Wreath

Table of Contents

[Executive Summary 3](#_Toc84857560)

[Summary of Results 4](#_Toc84857561)

[Attack Narrative 5](#_Toc84857562)

[First Machine (.200) 5](#_Toc84857563)

[Reconnaissance 5](#_Toc84857564)

[Webserver Compromise 6](#_Toc84857565)

[Reverse Shell 8](#_Toc84857566)

[Persistence 8](#_Toc84857567)

[Second Machine (.150) 9](#_Toc84857568)

[Pivoting 9](#_Toc84857569)

[Enumeration 9](#_Toc84857570)

[Privilege Escalation 12](#_Toc84857571)

[Third Machine (.100) 13](#_Toc84857572)

[Enumeration 13](#_Toc84857573)

[Exploitation 14](#_Toc84857574)

[Privilege Escalation 16](#_Toc84857575)

[Data Exfiltration 18](#_Toc84857576)

[Conclusion 19](#_Toc84857577)

[Recommendations 20](#_Toc84857578)

[Risk Rating 21](#_Toc84857579)

[Appendix 22](#_Toc84857580)

[Patch Management 22](#_Toc84857581)

[Password Reuse 22](#_Toc84857582)

# Executive Summary

Joker122402 was hired by Mr. Thomas Wreath to conduct a penetration test in order to determine his networks exposure to a targeted attack. All activities were conducted in a manner that simulates a malicious actor engaged in a target attack against Mr. Wreath with the goals of:

* Identifying if an attacker could breach Mr. Wreaths network
* Determining the Impact of a security breach on:
  + The safety of Mr. Wreaths Personal information
  + The internal infrastructure of Mr. Wreaths Network.

Efforts were placed on the identification and exploitation of security weaknesses that could allow an attacker to gain unauthorized access to data within the network. The attacks were conducted with the level of access a general internet user would have.

## Summary of Results

Initial reconnaissance revealed an out-of-date Webmin software that allowed for Remote Code Execution (RCE). The service was also insecurely configured to be running as the root user allowing us to leverage the RCE to gain full control of the server.

Using the compromised server as a pivot point, we began to attack the internal network. We first found a github server running an outdated version of the GitStack software. This software was vulnerable to an unauthorized upload vulnerability which allowed us to upload a malicious file and gain unauthorized RCE on the machine.

Using the second machine as a second pivot point, we were able to use an insecure upload vulnerability to bypass filters and upload a malicious file to the third machine. We were able to leverage this to gain access to the third machine. We were then able to escalate our privileges to the highest level by making use of an unquoted service path attack, in which we were able to replace one of the running services with a malicious file which gave us full access to the final machine. We were then able to retrieve passwords and information from that machine.

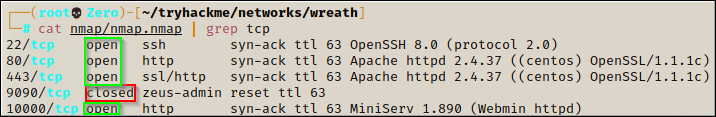
# Attack Narrative

## First Machine (.200)

When contracted by Mr. Wreath we were given the IP Address of his public webserver, which was hosting his website. This is the only machine we were able to access to begin so it was our initial target.

### Reconnaissance

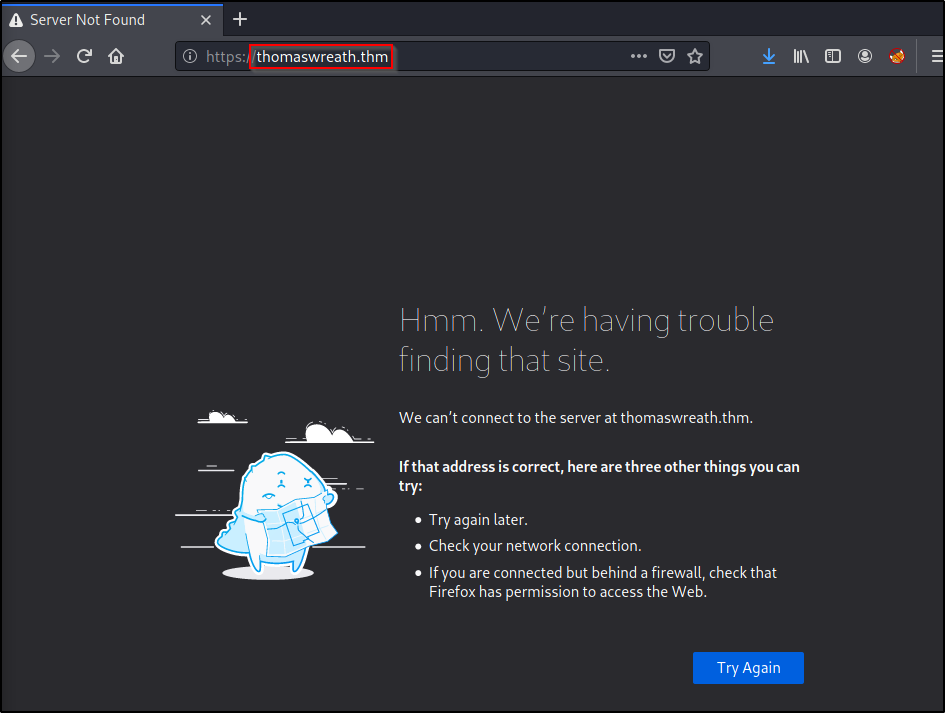
Mr. Wreath purposefully provided limited information about the network. Our initial scans were done using nmap and provided valuable insight as to what services were running and the target.



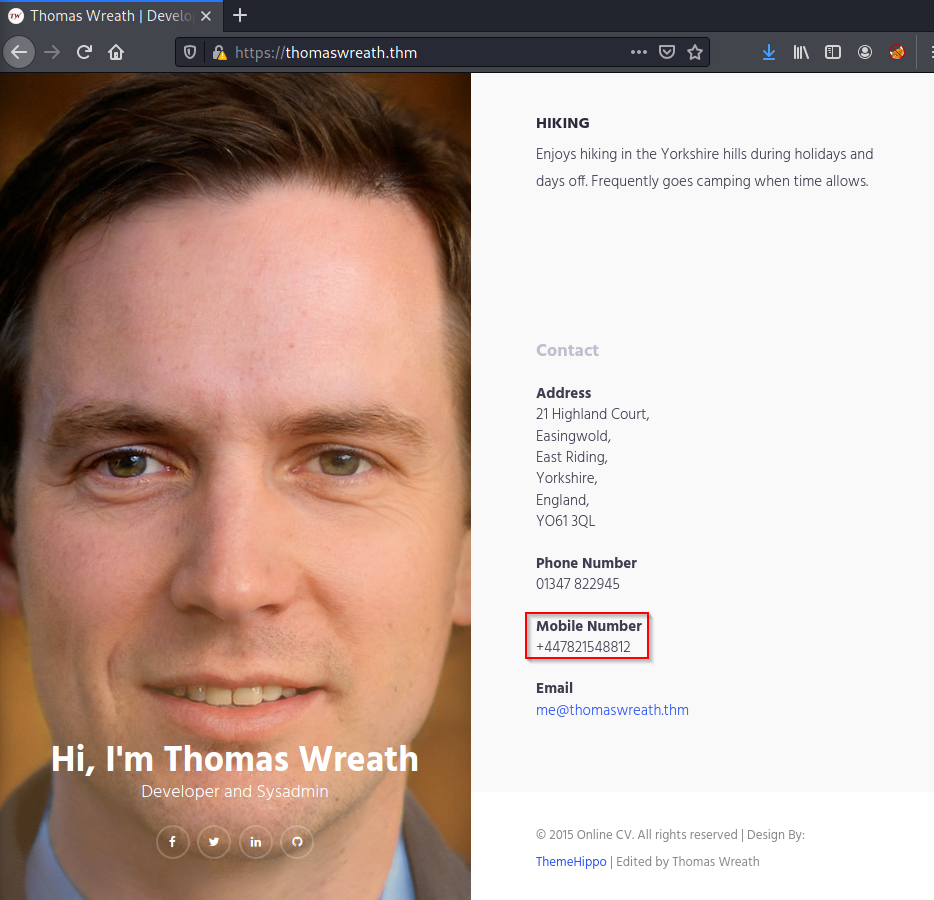
Our initial scans revealed that the target had 4 open ports. Port 22 running SSH, and port 80, 443 and 10000, running different HTTP and HTTPS services respectively.



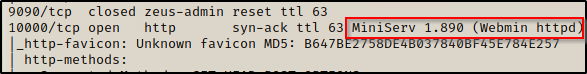
We were also able to determine that the target was running the CentOS Operating System.



When trying to navigate to the website, we found that the website did not have DNS properly set up. In order to get access the website, we needed to add thomeswreath.thm to our **/etc/hosts** file.



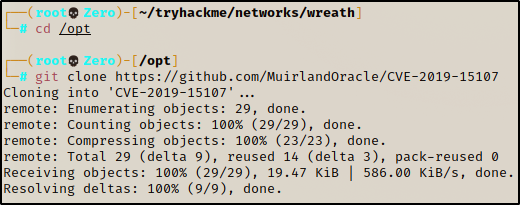
Accessing the website didn’t reveal any useful information to us, so we returned to our initial scan.



The scan revealed that the target was running the Webmin HTTP service MiniServ 1.890, which is an outdated software.

### Webserver Compromise

Looking up the MiniServ version we found revealed that Miniserv 1.890 is an outdated version of the Webmin software and that it is vulnerable to CVE-2019-15107[[1]](#footnote-1). This vulnerability is categorized as a 9.8 critical vulnerability. Exploiting it allows the attacker unauthorized remote code execution (RCE) due to a flaw in the password reset function. Upon looking into this vulnerability a bit further we found a github repository[[2]](#footnote-2) for the CVE which is good because GitHub repositories are usually kept up to date and well maintained.

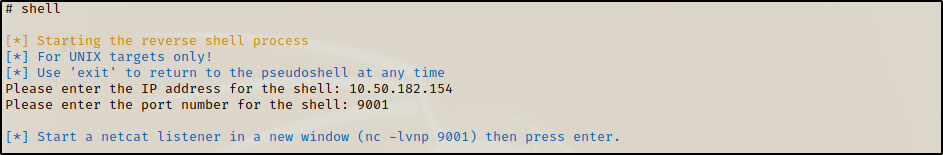


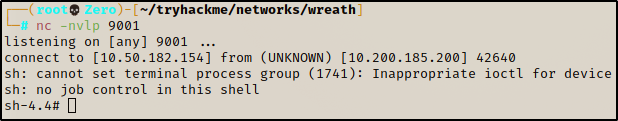
Inside the repository there is also a script that when run, will ensure you have all the necessary requirements to make use of this exploit.



By supplying the target IP to the exploit script, we are granted pseudo shell on the machine that will execute commands for us. Executing “whoami” revealed that the server was running root.

## Reverse Shell

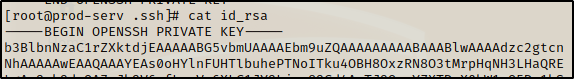
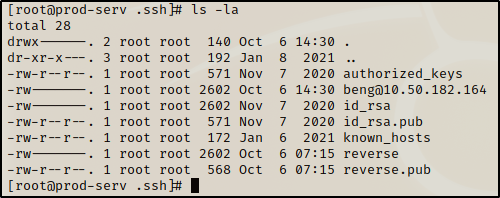




This exploit also provided a way for us to gain a full reverse shell on the target system. We set up a listener of our own and then upon executing the command “shell” the exploit script connected to our listener giving us us a full reverse shell. Since the server is running as root, we got a shell as the root user (root is the highest privileged user on linux systems).

## Persistence

As the root user there are no restrictions as to what we can and can’t do on the server. The easiest way to maintain access would be to crack the password of the root user, however our efforts to do so were not successful.



So instead, we copied the id\_rsa key from the /root/.ssh. This allowed us us to login via SSH as the root user without ever needing to supply a password.

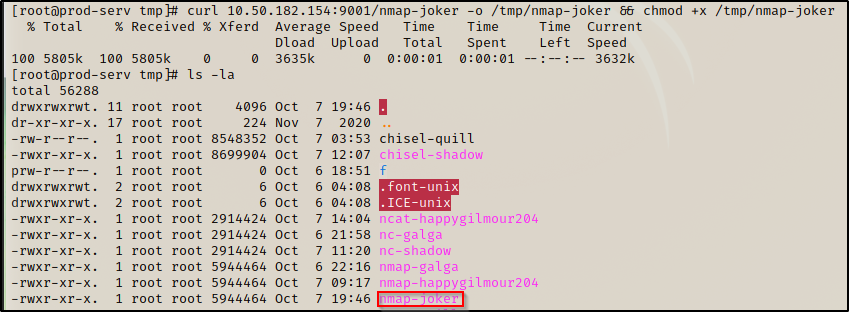
## Second Machine (.150)

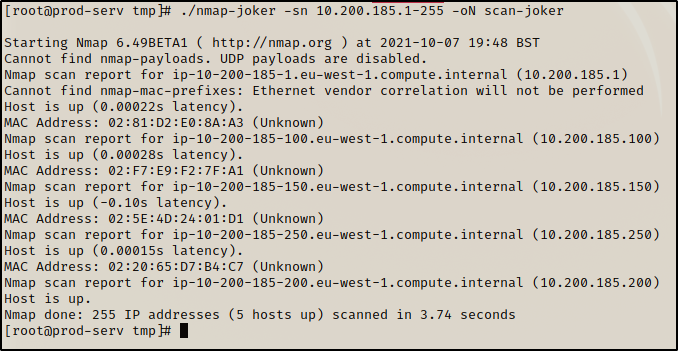
### Pivoting



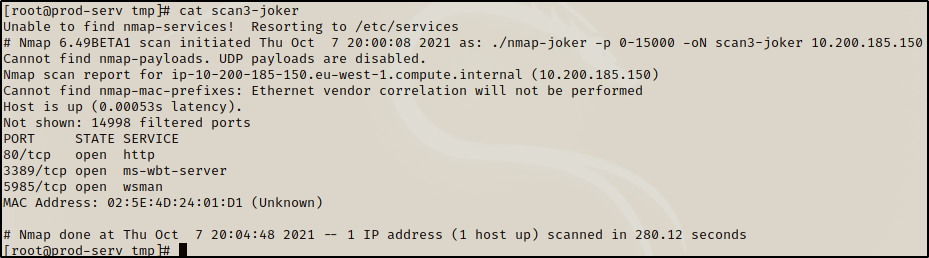
Using the compromised webserver, we utilized sshuttle[[3]](#footnote-3), a publicly available too, create an ssh tunnel to the webserver. Doing this allowed us to expand our attack surface to include the second machine on the network which was previously unreachable.

### Enumeration

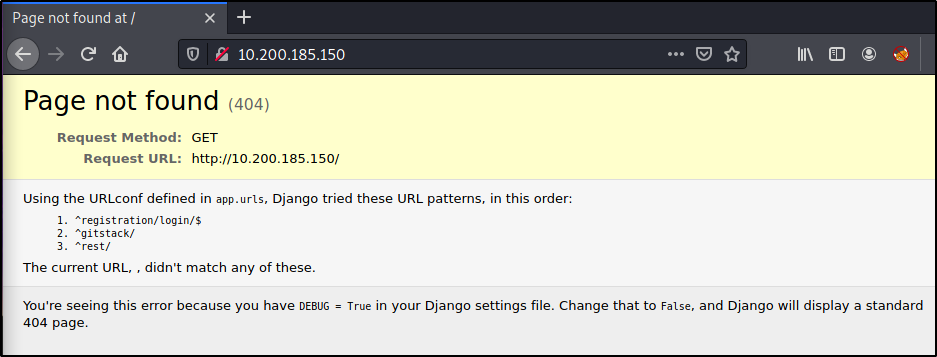




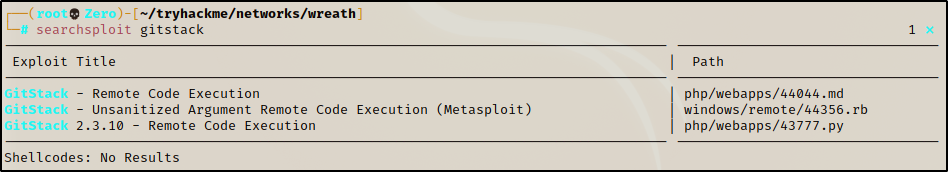
We again utilized nmap to do our scanning. We did this by uploading a static binary of the tool to the compromised webserver. With the static binary we scanned the subnet for alive hosts. Knowing that .1 and .250 were out of scope of the assessment and that we had already compromised .200 we knew our next target was .150.



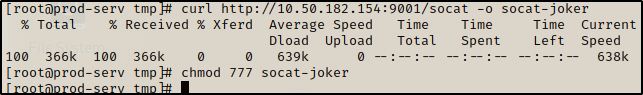
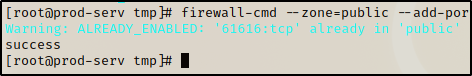
Our scan revealed that the next target was also hosting a web server on port 80.



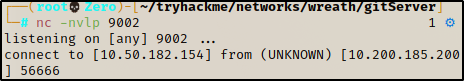
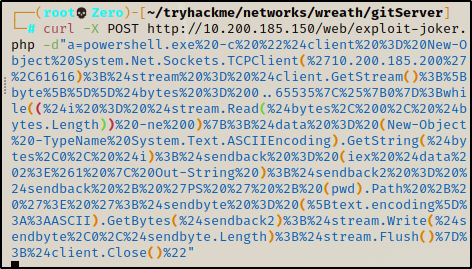
Navigation the website revealed that we needed to visit a subdirectory of the website to actually access anything. Through trial and error we that the /gitstack directory was where we needed to go. Naviagting there reveled a login page that displayed default credentials. The credentials however did not work.



Basic research on the gitstack service revealed that some versions are vulnerable to exploit 43777[[4]](#footnote-4), yet another RCE vulnerability. After making some changes to this publicly available exploit, we were able to use it to upload a malicious file to the webserver which allowed us to execute commands on the machine.

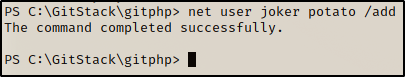


We then opened up the firewall on the previously compromised webserver to allow us to relay a shell back to our attacking machine using a tool called socat.



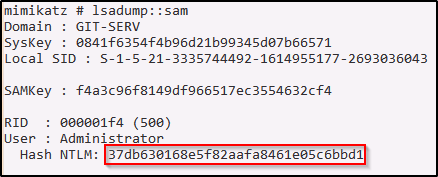
This in turn allowed us to send a command to the target machine that gave us full access to the underlying OS.

### Privilege Escalation

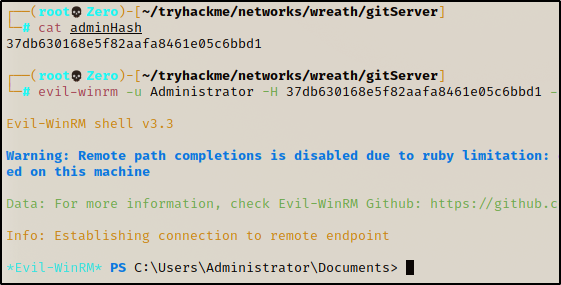


In order to maintain access to this newly compromised Github server, we simply added a new user to the machine. We then used RDP to gain full graphical access to this machine.

Utilizing RDP we were able to upload a malicious tool called mimikatz that is designed to steal as much information from windows machines as possible.

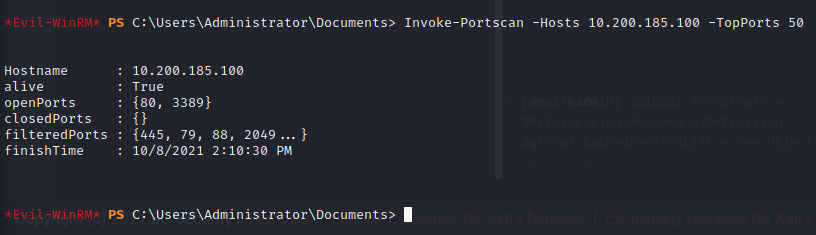
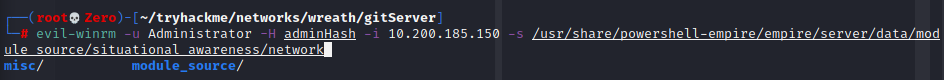


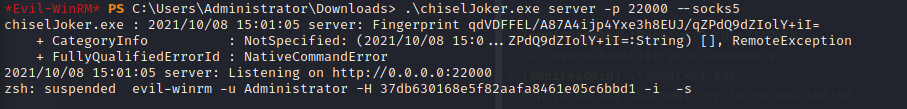
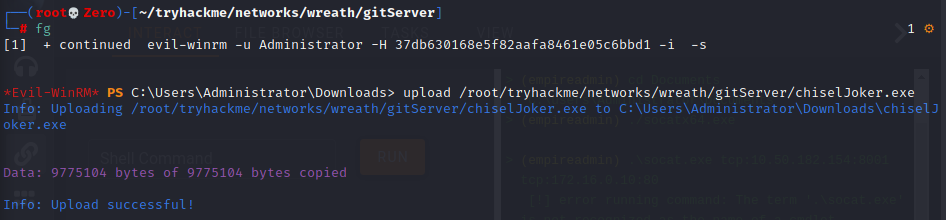
Using mimikatz, we were able to dump the SAM Database, which is where windows stores password hashes. We then cracked this hash using an online tool called Crackstation[[5]](#footnote-5) to reveal the administrator password.

 However aside from being able to crack the hash, we were able to use a tool called evil-winrm to execute a pass the hash attack, meaning that even if we couldn’t crack the hash, we would still have persistent Administrator access to the now compromised second machine.

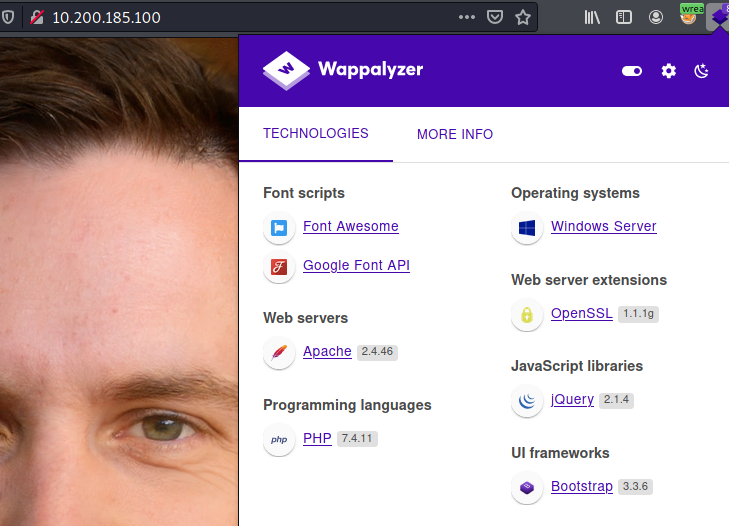
## Third Machine (.100)

### Enumeration



Utilizing evil-winrm we were able to load some scripts directly into the memory of the compromised machine allowing us to use a powerful port scanning tool without ever writing anything to the disk of the GitHub server. The port scan revealed that the only open ports on the target machine were ports 80, and 3389. 

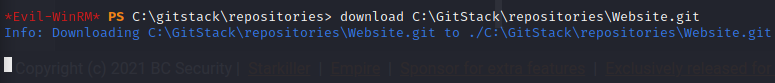
Using a tool called chisel[[6]](#footnote-6) we were able to proxy our network traffic through the compromised GitHub server to allow us to access the website running on port 80 of the target.



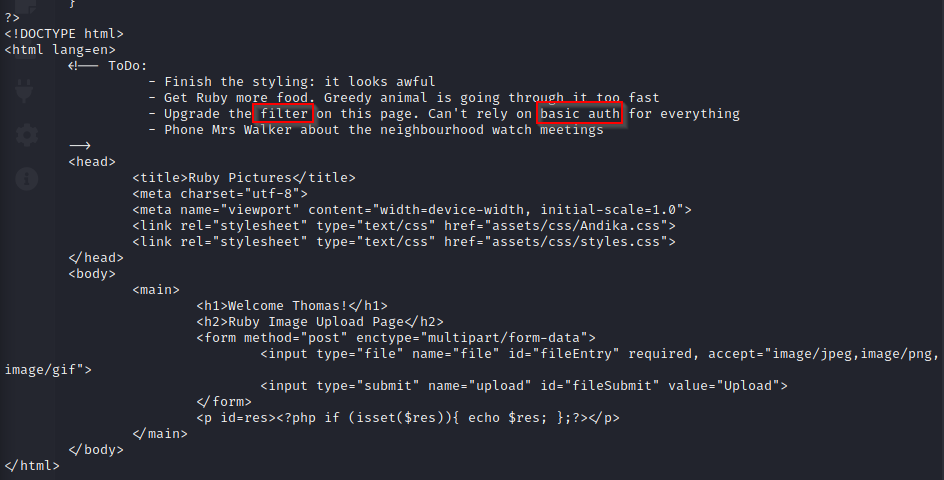
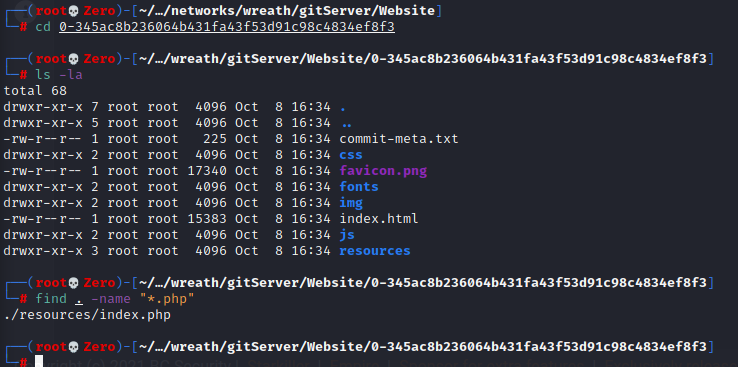
A tool called wappalyzer revealed to us that the website was running the PHP programming language.

This website appeared to be a carbon copy of the previous site that was being hosted on the GitHub server.

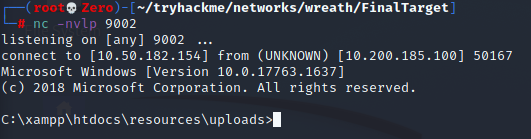
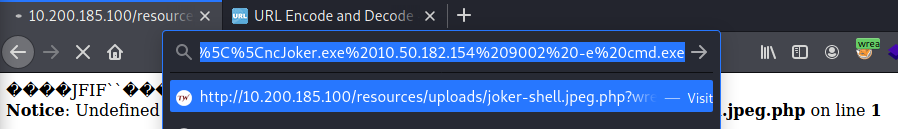
### Exploitation



Again using evil-winrm we were able to download the Website.git file which hosts the website. We used a suite of tools called GitTools[[7]](#footnote-7), to allow us to locally create the entire GitHub repository based off of the Website.git file. We were then able to determine which commit was the most up to date version of the website and we searched it for vulnerabilities

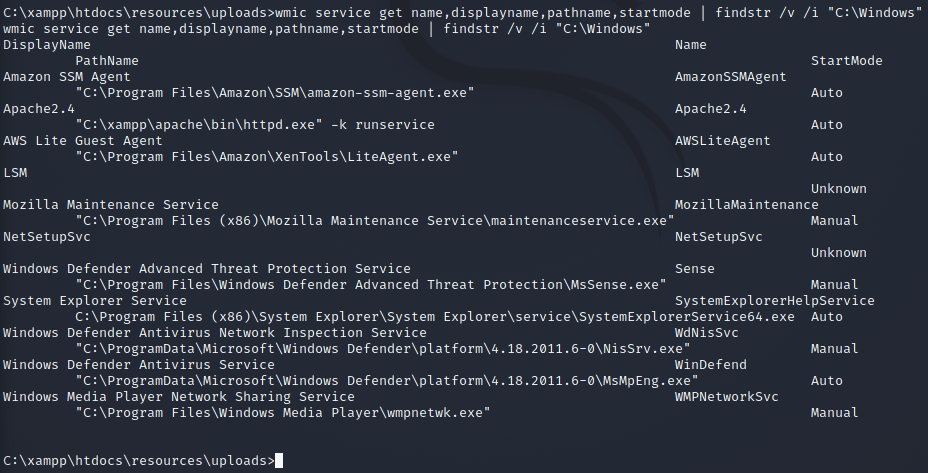


We found a single php file which appeared to be code for some sort of upload page. The To-Do list at the bottom revealed that we would need to bypass basic authentication and a content filter, in order to exploit this machine. The content filter was a mix of a very basic extension filter, and an image size filter. These 2 filters were easily bypassed by embedding our malicious code into an actualy picture, and naming it with the double extension of .jpeg.php.

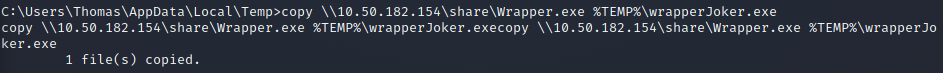


We were able to bypass the anti-virus software on the target with some basic code obfuscation which allowed us to gain a remote shell on the target machine.

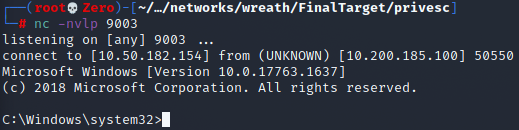
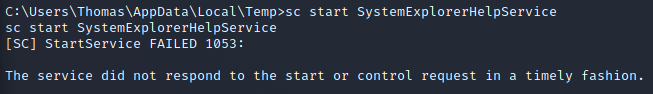
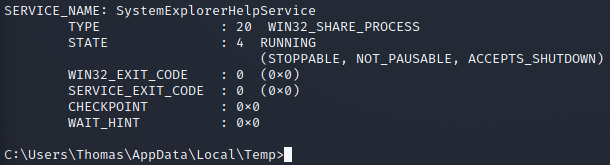
### Privilege Escalation



Once we had a shell on the machine, we needed to find a way to escalate our privileges. While enumerating the running services running on the machine, we noticed that the SystemExplorerHelpService did not have quotes around its path. This indicated that it might be vulnerable to an unquoted service path attack[[8]](#footnote-8), if we had write permission for any of the directories listed in the path. Due to a very clear misconfiguration of permissions, we were able to erite to one of the directories.

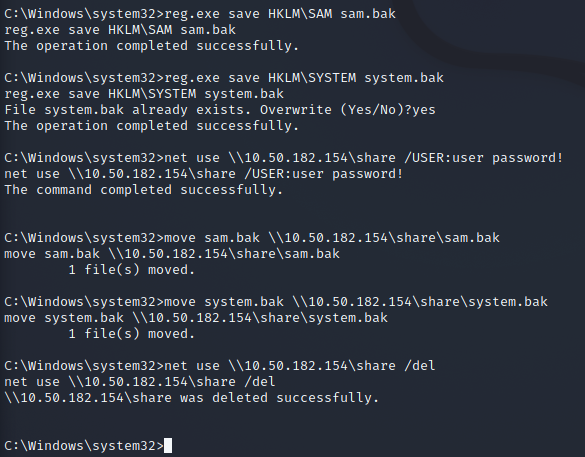


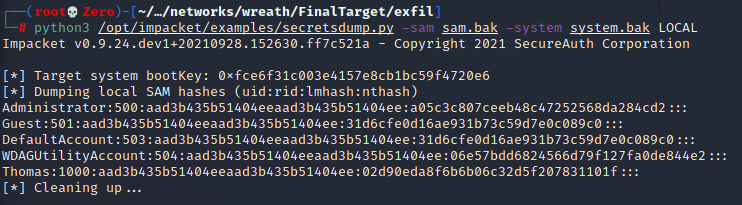
We uploaded a simple wrapper class that would execute a command as the service when executed. Since the service was running as the system user, this could be used to give us a System Authority shell (highest level shell on windows). We placed our wrapper program in C:\Program Files (x86)\System Explorer\ and called it System.exe thus tricking windows into thinking that our wrapper is the file its supposed to execute rather than the actual service file at the end of the path.



We are able to restart the service and get a System shell on the target.

### Data Exfiltration



For our exfiltration the only things in scope were password hashes. Since the system we were on had an anti-virus running on it, we couldn’t simply use mimikatz to dump the SAM database again. Instead we had to download the SAM hive and the SYSTEM hive files, to allow us to recreate the SAM database locally. We then used a script called secretsdump.py from the Impacket tool suite[[9]](#footnote-9) to rebuild the SAM Database and dump the different password hashes it stores.

# Conclusion

The Wreath network suffered a series of control failures which led to a complete compromise of the entire network and its data. These failures would have drastic results on the Wreath Network if a malicious party had exploited them. Current policies regarding software maintenance and password reuse are not adequate to mitigate the impact of the discovered vulnerabilities.

The specific goals of the penetration test were stated as:

* Identifying if an attacker could breach Mr. Wreaths network
* Determining the Impact of a security breach on:
  + The safety of Mr. Wreaths Personal information
  + The internal infrastructure of Mr. Wreaths Network

These goals were met. A targeted attack against Mr. Wreaths network can result in a complete compromise of his entire network, and all the data within it. It’s important to note that this compromise of security infrastructure, can be greatly attributed to outdated software across a majority of the network. Appropriate efforts should be taken to keep software updated which could help mitigate the effect of security failures thought the Wreath Network.

## Recommendations

Due to the impact to the overall network as uncovered by this penetration test, appropriate resources should be allocated to ensure remediation efforts are accomplished in a timely manner. While a comprehensive list of items that should be implemented is beyond the scope of this engagement, some high-level items are important to mention.

1. **Implement a patch management program:** The compromise of the Wreath Network was drastically impacted by the outdated software present throughout the network. Operating a consistent patch management program is an important part of maintaining strong security. This will help to limit the possible attack surface that results from running unpatched internal services.
2. **Ensure that strong credentials are used everywhere within the network:** It is recommended that strong password policy be implemented across the entire network. While this issue wasn’t the complete cause of the network being compromised, it was still and issue and should be addressed

## Risk Rating

The overall risk identified to the Wreath Network as a result of the penetration test is **High**. A direct path from external attacker to full system compromise was discovered. It is reasonable to believe that a malicious entity would be able to successfully execute an attack against the Wreath Network through targeted attacks.

# Appendix

## Patch Management

**Rating: High**

**Description:** The Wreath network’s internal and external environments contain a number of unpatched systems and applications.

**Impact:** A combination of weak authentication and unpatched hosts, which contain known vulnerabilities with publicly available exploits, allows an attacker to gain access to a large majority of the Wreath Networks assets. Specifically, the outwards facing webserver, as well as the internal GitHub server were both compromised due to unpatched software.

**Remediation:** All assets on the network should be kept current with the latest vendor supplied security patches. This can be achieved with vendor supplied tools or third-party applications which can provide an overview of all missing patches. In many instances, third-party tools can also be used for patch deployment throughout the environment.

### Password Reuse

**Rating: High**

**Description:** The Wreath network’s user “**Thomas**” was found to be reusing passwords throughout the network. This was used to gain access to the upload page on the final target machine.

**Impact:** Password reuse in general is a practice which should be highly discouraged and prevented to the extend possible. Though it wasn’t a problem for a majority of the network, the password reuse indirectly led to the compromise of the final machine on the network.

**Remediation:** All assets on the network should be kept current with the latest vendor supplied security patches. This can be achieved with vendor supplied tools or third-party applications which can provide an overview of all missing patches. In many instances, third-party tools can also be used for patch deployment throughout the environment.

1. <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-15107> [↑](#footnote-ref-1)
2. <https://github.com/MuirlandOracle/CVE-2019-15107> [↑](#footnote-ref-2)
3. <https://github.com/sshuttle/sshuttle> [↑](#footnote-ref-3)
4. <https://www.exploit-db.com/exploits/43777> [↑](#footnote-ref-4)
5. <https://crackstation.net> [↑](#footnote-ref-5)
6. <https://github.com/jpillora/chisel> [↑](#footnote-ref-6)
7. <https://github.com/internetwache/GitTools> [↑](#footnote-ref-7)
8. <https://gracefulsecurity.com/privesc-unquoted-service-path/> [↑](#footnote-ref-8)
9. <https://github.com/SecureAuthCorp/impacket> [↑](#footnote-ref-9)