Domain admin in 30 minutes

This is an experience from a real internal domain pentest. Customer name has obviously been removed.

Goal here is going from network access only to domain admin (DA). This process takes about 30 min and was performed on a real pentest.

A few key points:

- This particular customer had removed regular users as local admins from workstations, so using responding and relaying didn't give us much here. This is after all a pretty old attack vector, although it has a very good success rate when regular users are local admins on their workstations.
- Certain amout of luck involved, but with such a huge domain and so many admins there will always be a way to DA
- Policy did not allow domain admins to remote into workstations.
- There was lockout on both admin and domain admin users after five tries.
- Important to not lock anybody out!
- Use HTTPS meterpreter shells to avoid getting busted
- Don't drop exe files on boxes, and if do clean up afterwards

Null session on domain controller (DC)

Did a fast scan with Nessus, could have done other things to find a domain controller like nmap or other windows commands.

Tried to check it the DC supported null sessions using the following command:

```
enum4linux -a <ip address>
```

Domain controller gives us everything we need.

A list of every machine, every user, including admins and domain admins.

Bonus: every group membership and the password policy.

PW policy reveals there is no lockout on regular users and no requirement for complexity.

It's a party!

Password spraying

Every regular user has three letter usernames. Let's make a list of all those. We don't want to block out admin users (they usually have lockout). About 500 usernames on this test.

No complexity requirements = shit passwords. This test was performed during christmas 2017 so we make a list of relevant passwords and check it twice:

- Winter2017
- winter2017
- christmas2017
- Christmas2017

Now we need a place to try the passwords. Trying to authenticate with a lot of failures on SMB may be detected and this stuff is way easier to set up in Burp anyway. We load up a list of usernames and passwords with Burp Intruder and point it to the external OWA e-mail server that really wasn't that hard to guess.

`www.mail.customer.com`

Now we just run it for all 500 usernames with those four passwords for all of them. Spray and pray!

Takes about 5 seconds to filter for 200 OK in Burp and see where the pw is accepted. It's also possible to sort by size of the response to easily find the positives. Got a few regular users login credentials now. We can get busy! Remember there was no lockout on passwords, so we could have tried until somebody came

running through the door of the tiny meeting room we were sitting in. Food for thought .... So we get the credentials of a user called `ank`.

Make a map of the domain with regular user privs

Now remember this user is not an admin but the way Active Directory works allows us to map out the whole domain.The syntax is:

`runas /netonly /FQDN\user cmd.exe`

So we ran the following command:

`runas /netonly /user:customer\ank powershell`

Type in password `winter2017` when prompted. This query gets executed at the domain controller, and if successfull we can basically execute queries that are forwarded and executed on the DC. This is not remote code execution or any sort of vuln, just inherent functionality. We now want to make a map with Bloodhound, so if you haven't get that set up using the guide from this book. Then go ahead and load up SharpHound.ps1 and execute it. Depending on the size of the domain and activity, this should take some time. On busy daytime in a corporate office with a few thousand workstations and servers this took 30 minutes to complete.

```
Powershell -exec bypass
Import-module SharpHound.ps1
Invoke-BloodHound -CollectionMethod ACL,ObjectProps,Default -CompressData –Skip
```

Wait 20 minutes for a full map of the entire domain (3000 hosts). Now load it all up in bloodhound using the import feature, just put the entire ZIP file in there. Now BloodHound has a few default queries which are quite useful. Since we want domain admin we wanna see if its an easy way.

`Find shortest path to domain admin`

Now we notice a few things that we kind of knew already, but now realized. The number of domain admins is too damn high! Also, by cross referencing the output from the null session enumeration on the DC we see domain admin usernames that are highly susceptible as service accounts. A thing we didn't try before

way later in the engagement was logging in as domain admins with the same username as password. Believe it or not, this actually worked for one of the domain admins, which is plain ridiculous. We could have reduced our effort getting DA from 30 minutes to 30 seconds, but of course we didn't wanna go wild password spraying domain admins as this would get us detected really fast and we could potentially block out domain admins preventing them from doing their jobs. That's not good!

Pivoting around to find a box that has domain admin

So now that we have a proper map we need to try to get closer to a box with domain admin so we can capture that users privileges or credentials somehow. Because this customer had removed all kinds of local admin from workstation's the regular users we had access to basically has no good privs, except as Bloodhound so elegantly reveals one single box. WTF? Apparently in their rigoorous removal of local admins from workstations they had forgot this one WS which is still in use. So before we can get to a box with domain admin we need an intermediary step we jump on to a workstation the user has access to that also has an admin logged on with psexec (SMB share login) and RDP (remote desktop). Because this box was accessible by every user in the domain it was just jumping on it. By some stroke of luck an admin user is also logged onto this box. We use in Metasploit with the user's creds on that workstation:

```
exploit/windows/smb/psexec_psh
```

We now have shell as local admin on that workstation, and hence we can run mimikatz. We load and execute mimikatz straight into memory from metasploit:

```
load mimikatz
mimikatz_command -f sekurlsa::logonPasswords full
```

We now dump the admin user's password and hash in plaintext (easily identifiable as all admin-users in this domain has "admin" in the username). We check the privileges of this admin user and find a box with multiple domain admins on that this user also has access to. On to the final step!

Privilege escalation from admin to domain admin

We shell in with `psexec_psh` to the box that has domain admins. We are not local admin on this box, so we can't do mimikatz. Luckily, there are other ways to get privileges. Rotten Potato is a privilege escalation technique that works well for this kind of scenario. Using this technique, we can elevate our privilege on a Windows workstation from the lowest levels to “NT AUTHORITY\SYSTEM” – the highest level of privilege available on a Windows machine. We do the following in our Meterpreter:

```
getuid
getprivs
upload /root/just_dce_64.exe c:\\temp\\rot.exe
load incognito
list_tokens -u
execute -Hc -f ./rot.exe
impersonate_token "NT AUTHORITY\\SYSTEM"
getprivs
```

We are now SYSTEM on this box and can run mimikatz do dump the plaintext creds of whatever domain admin is logged on the box. Alternatively, we can use rotten potato to impersonate a domain admin. Just replace the username of the impersonate_token command from above with the username you want to impersonate. Wild!

Congrats you are now domain admin!

Previous
War stories

Next
Password cracking and auditing

Last updated 6 years ago