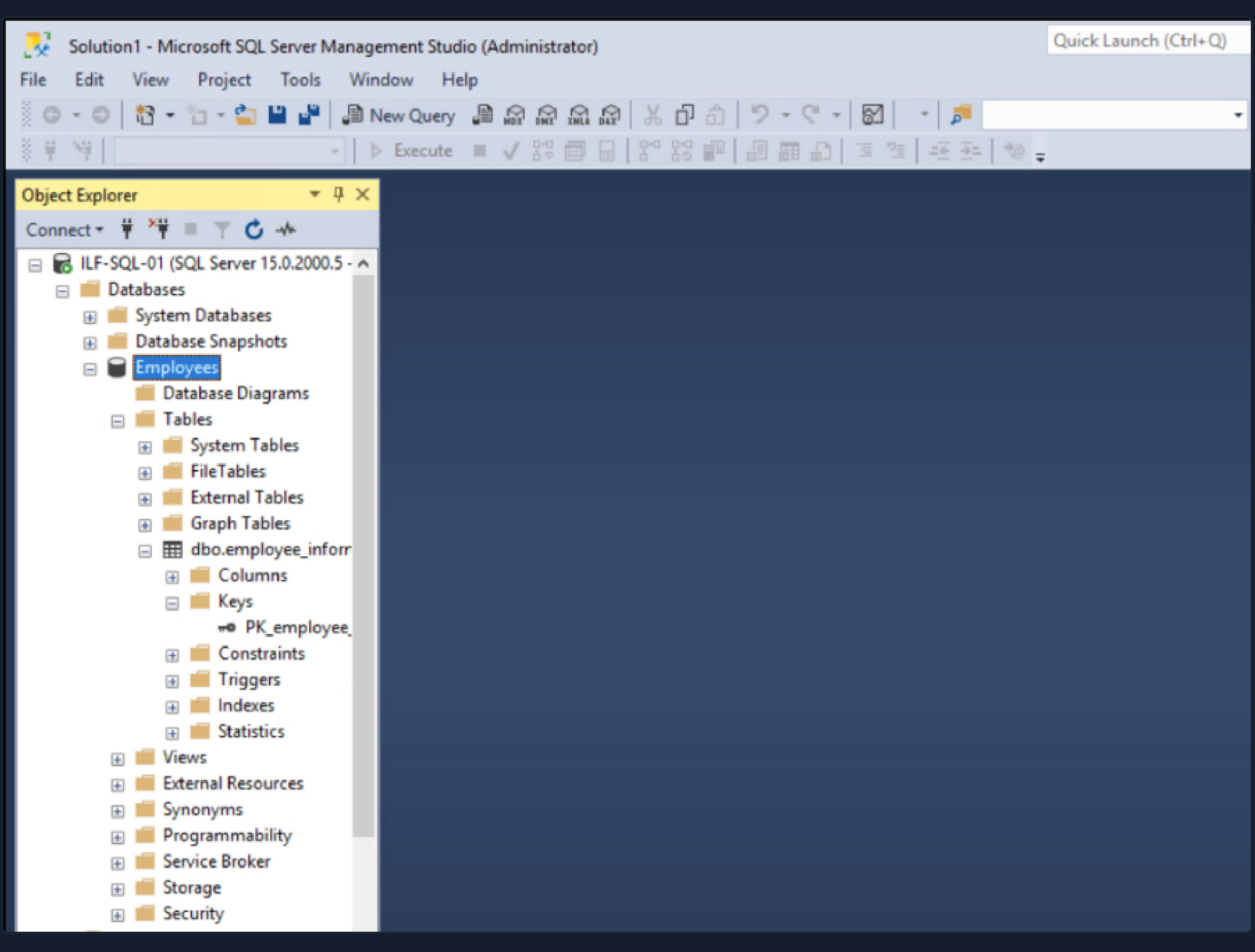
MSSQL

Microsoft SQL (`MSSQL`) is Microsoft's SQL-based relational database management system. Unlike MySQL, which we discussed in the last section, MSSQL is closed source and was initially written to run on Windows operating systems. It is popular among database administrators and developers when building applications that run on Microsoft's .NET framework due to its strong native support for .NET. There are versions of MSSQL that will run on Linux and MacOS, but we will more likely come across MSSQL instances on targets running Windows.

MSSQL Clients

SQL Server Management Studio (`SSMS`) comes as a feature that can be installed with the MSSQL install package or can be downloaded & installed separately. It is commonly installed on the server for initial configuration and long-term management of databases by admins. Keep in mind that since SSMS is a client-side application, it can be installed and used on any system an admin or developer is planning to manage the database from. It doesn't only exist on the server hosting the database. This means we could come across a vulnerable system with SSMS with saved credentials that allow us to connect to the database. The image below shows SSMS in action.



Many other clients can be used to access a database running on MSSQL. Including but not limited to:

mssql-cli	SQL Server PowerShell	HediSQL	SQLPro	Impacket's mssqlclient.py

Of the MSSQL clients listed above, pentester's may find Impacket's mssqlclient.py to be the most useful due to SecureAuthCorp's Impacket project being present on many pentesting distributions at install. To find if and where the client is located on our host, we can use the following command:

MSSQL Clients

```
Govardhan Gujji22@htb[/htb]$ locate mssqlclient

/usr/bin/impacket-mssqlclient
/usr/share/doc/python3-impacket/examples/mssqlclient.py
```

MSSQL Databases

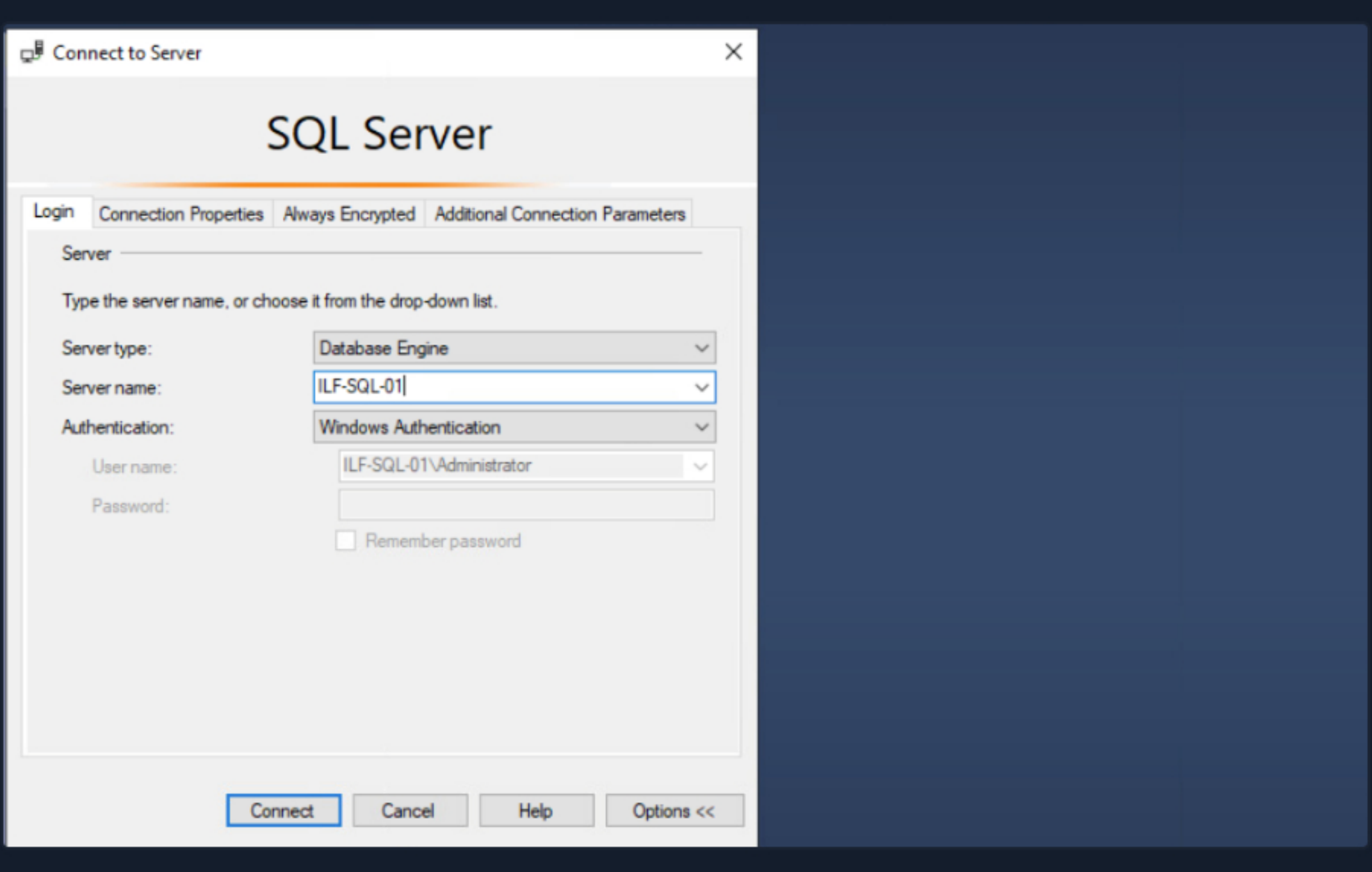
MSSQL has default system databases that can help us understand the structure of all the databases that may be hosted on a target server. Here are the default databases and a brief description of each:

Default System Database	Description
`master`	Tracks all system information for an SQL server instance
`model`	Template database that acts as a structure for every new database created. Any setting changed in the model database will be reflected in any new database created after changes to the model database
`msdb`	The SQL Server Agent uses this database to schedule jobs & alerts
`tempdb`	Stores temporary objects
`resource`	Read-only database containing system objects included with SQL server

Table source: System Databases Microsoft Doc

Default Configuration

When an admin initially installs and configures MSSQL to be network accessible, the SQL service will likely run as `NT SERVICE\MSSQLSERVER`. Connecting from the client-side is possible through Windows Authentication, and by default, encryption is not enforced when attempting to connect.



Authentication being set to `Windows Authentication` means that the underlying Windows OS will process the login request and use either the local SAM database or the domain controller (hosting Active Directory) before allowing connectivity to the database management system. Using Active Directory can be ideal for auditing activity and controlling access in a Windows environment, but if an account is compromised, it could lead to privilege escalation and lateral movement across a Windows domain environment. Like with any OS, service, server role, or application, it can be beneficial to set it up in a VM from installation to configuration to understand all the default configurations and potential mistakes that the administrator could make.

Dangerous Settings

It can be beneficial to place ourselves in the perspective of an IT administrator when we are on an engagement. This mindset can help us remember to look for various settings that may have been misconfigured or configured in a dangerous manner by an admin. A workday in IT can be rather busy, with lots of different projects happening simultaneously and the pressure to perform with speed & accuracy being a reality in many organizations, mistakes can be easily made. It only takes one tiny misconfiguration that could compromise a critical server or service on the network. This applies to just about every network service and server role that can be configured, including MSSQL.

This is not an extensive list because there are countless ways MSSQL databases can be configured by admins based on the needs of their respective organizations. We may benefit from looking into the following:

- MSSQL clients not using encryption to connect to the MSSQL server
- The use of self-signed certificates when encryption is being used. It is possible to spoof self-signed certificates
- The use of named pipes
- Weak & default `sa` credentials. Admins may forget to disable this account

Footprinting the Service

There are many ways we can approach footprinting the MSSQL service, the more specific we can get with our scans, the more useful information we will be able to gather. NMAP has default mssql scripts that can be used to target the default tcp port `1433` that MSSQL listens on.

The scripted NMAP scan below provides us with helpful information. We can see the `hostname`, `database instance name`, `software version of MSSQL` and `named pipes are enabled`. We will benefit from adding these discoveries to our notes.

NMAP MSSQL Script Scan

NMAP MSSQL Script Scan

```
Govardhan Gujji22@htb[/htb]$ sudo nmap --script ms-sql-info,ms-sql-empty-password,ms-sql-xp-cmdshell,ms-s

Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-08 09:40 EST
Nmap scan report for 10.129.201.248
Host is up (0.15s latency).

PORT     STATE SERVICE  VERSION
1433/tcp open  ms-sql-s Microsoft SQL Server 2019 15.00.2000.00; RTM
| ms-sql-ntlm-info:
|   Target_Name: SQL-01
|   NetBIOS_Domain_Name: SQL-01
|   NetBIOS_Computer_Name: SQL-01
|   DNS_Domain_Name: SQL-01
|   DNS_Computer_Name: SQL-01
|_  Product_Version: 10.0.17763

Host script results:
| ms-sql-dac:
|_  Instance: MSSQLSERVER; DAC port: 1434 (connection failed)
| ms-sql-info:
|   Windows server name: SQL-01
|   10.129.201.248\MSSQLSERVER:
|     Instance name: MSSQLSERVER
|     Version:
|       name: Microsoft SQL Server 2019 RTM
|       number: 15.00.2000.00
|       Product: Microsoft SQL Server 2019
|       Service pack level: RTM
|       Post-SP patches applied: false
|     TCP port: 1433
|     Named pipe: \\10.129.201.248\pipe\sql\query
|_    Clustered: false

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.52 seconds
```

We can also use Metasploit to run an auxiliary scanner called `mssql_ping` that will scan the MSSQL service and provide helpful information in our footprinting process.

MSSQL Ping in Metasploit

MSSQL Ping in Metasploit

```
msf6 auxiliary(scanner/mssql/mssql_ping) > set rhosts 10.129.201.248

rhosts => 10.129.201.248


msf6 auxiliary(scanner/mssql/mssql_ping) > run

[*] 10.129.201.248:       - SQL Server information for 10.129.201.248:
[+] 10.129.201.248:       -    ServerName      = SQL-01
[+] 10.129.201.248:       -    InstanceName    = MSSQLSERVER
[+] 10.129.201.248:       -    IsClustered     = No
[+] 10.129.201.248:       -    Version         = 15.0.2000.5
[+] 10.129.201.248:       -    tcp             = 1433
[+] 10.129.201.248:       -    np              = \\SQL-01\pipe\sql\query
[*] 10.129.201.248:       - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Connecting with Mssqlclient.py

If we can guess or gain access to credentials, this allows us to remotely connect to the MSSQL server and start interacting with databases using T-SQL (`Transact-SQL`). Authenticating with MSSQL will enable us to interact directly with databases through the SQL Database Engine. From Pwnbox or a personal attack host, we can use Impacket's mssqlclient.py to connect as seen in the output below. Once connected to the server, it may be good to get a lay of the land and list the databases present on the system.

Connecting with Mssqlclient.py

```
Govardhan Gujji22@htb[/htb]$ python3 mssqlclient.py Administrator@10.129.201.248 -windows-auth

Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

Password:
[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(SQL-01): Line 1: Changed database context to 'master'.
[*] INFO(SQL-01): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (150 7208)
[!] Press help for extra shell commands

SQL> select name from sys.databases

name

--------------------------------------------------------------------------------------

master

tempdb

model

msdb

Transactions
```

Start Instance

∞ / 1 spawns left

Waiting to start...

Questions

Answer the question(s) below to complete this Section and earn cubes!

Cheat Sheet
Get VPN Key

Target: Click here to spawn the target system!

+ 0 Enumerate the target using the concepts taught in this section. List the hostname of MSSQL server.

ILF-SQL-01 — Submit — No Hint

+ 0 Connect to the MSSQL instance running on the target using the account (backdoor:Password1), then list the non-default database present on the server.

Employees — Submit — Hint

Previous Next Mark Complete & Next

Cheat Sheet
Resources
Go to Questions

Table of Contents

Introduction
Enumeration Principles
Enumeration Methodology
Infrastructure Based Enumeration
Domain Information
Cloud Resources
Staff
Host Based Enumeration
FTP
SMB
NFS
DNS
SMTP
IMAP / POP3
SNMP
MySQL
MSSQL
IPMI
Remote Management Protocols
Linux Remote Management Protocols
Windows Remote Management Protocols
Skills Assessment
Footprinting Lab - Easy
Footprinting Lab - Medium
Footprinting Lab - Hard

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left