Hacking SQL Server

Marek Chmel
Lead Database Administrator, SQL Team at&t Czech Republic
MVP Data Platform

Session Agenda

- SQL Server Security Overview
- Understanding Authentication Modes
- Getting SA access
- Protecting SQL Information
- Auditing SQL Instances
- Encrypting Information on the SQL Server

SQL Server Service Security Overview

- Startup accounts used to start and run SQL Server can be domain user accounts, local user accounts, managed service accounts, virtual accounts, or built-in system accounts – default is virtual account
- Virtual account is a local managed account
- Auto managed
- Can access network with credentials of computer account
  - Registers SPN

Working with SQL Logins

- We need to understand the linkage between logins, users, credentials, proxies, linked server logins etc.
- Two types of logins – Windows and SQL
  - SQL Logins have a hashed password stored locally in master DB
- SQL 2012+
  - `hashBytes = 0x0200 | fourByteSalt | SHA512(utf16EncodedPassword+fourByteSalt)`
- SQL 2008R2 and Older
  - `hashBytes = 0x0100 | fourByteSalt | SHA1(utf16EncodedPassword+fourByteSalt)`

Local System Security

- Local Windows Administrator has always access to SQL Server via several ways
  - Up to 2008r2 local system account has SA rights
  - 2012+ SQL server writer has SA rights

DEMO

System Security

SA Access

SQL Authentication

- 2 Modes for authentication
  - Windows
  - SQL and Windows
- Authentication mode configured via SSMS or via registry
  - Changing auth mode requires service restart

DEMO

SQL Authentication

SQL Server Single User Mode

- There are several parameters which can be used to start the SQL Server
  - Startup parameters –f and –m (single user and minimal configuration)
  - With any of these parameters local windows admins are sysadmins for SQL
- No logon triggers apply with this configuration

DEMO

SQL Startup Parameters

Using SysAdmin for Information Gathering

- Sysadmin has unlimited access to information stored within SQL server
  - Even while this information is encrypted
- Interesting sources of information
  - Login PWD
  - Linked Servers
  - Credentials
    - Encrypted using AES (2012+) or 3DES

Working without SysAdmin

- Enumerating Active Directory
  - Without SA rights you can't run xp_enumgroups and xp_logininfo, but there's still a way
  - Any user can execute a function SUSER_SID
  - Just use the returned SID, craft another one and run SUSER_SNAME
  - Those two are not limited to SA, and can be run by anyone

Escalation

- Under special circumstances you can escalate db_owner to SA
- Requires specific settings on the database (does not work in general)
- Can be automated with PowerShell to attack / scan multiple servers thanks to discovery

SQL Audit

Basic Audit on all SKUs

Server Audit Specs only
DB Audit Specs for Enterprise
Multiple Audits and multiple targets
Persist state
Audit Resilience

Improved Resilience

Automatically recover from most file or network errors
Added “ON_FAILURE = FAIL_OPERATION”
Added “MAX_FILES” option

Record Filtering

Tightly constrain info written to Audit log
Audit record generated but not written
Leverages Xevent filtering

Audit FAQ

- What is the performance impact?
- Can I protect the Audit log from the DBA?
- What happens if Audit fails to write?
- What do I do if the server fails to start because of SQL Server Audit?

DEMO

SQL Audit

Review the security info

- You can do your own “audit” of logins, roles, password changes etc
  - Regular role checks, alerts for role changes
  - Regular PWD aging checks
  - Logins with weak pwd

Protecting your data inside SQL

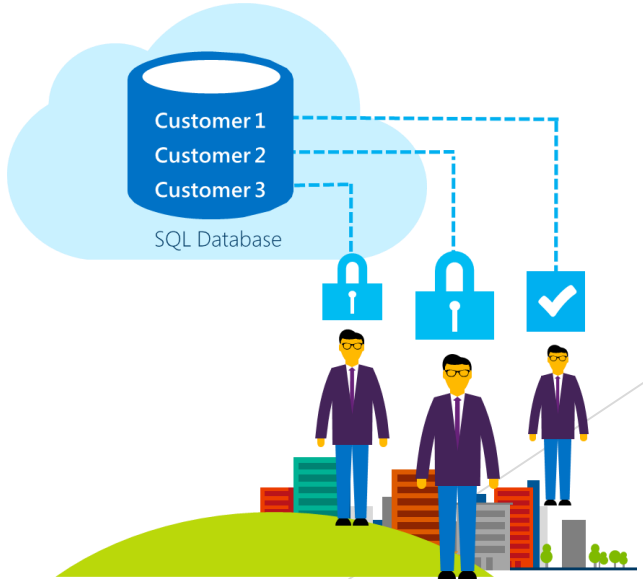
- Can you protect data stored on SQL server from unauthorized access?
- Transparent Data Encryption can protect data from nonSQL users
  - Encryption is done on SQLIO level, all data and log files are encrypted, including the backup

DEMO

Transparent Data Encryption

Row Level Security

- Fine-grained access control over specific rows in a database table
- Help prevent unauthorized access when multiple users share the same tables, or to implement connection filtering in multitenant applications
- Administer via SQL Server Management Studio or SQL Server Data Tools
- Enforcement logic inside the database and schema bound to the table.



Benefits of row-level security

Fine-grained access control

Keeping multi-tenant databases secure by limiting access by other users who share the same tables.

Application transparency

RLS works transparently at query time, no app changes needed.

Compatible with RLS in other leading products.

Centralized security logic

Enforcement logic resides inside database and is schema-bound to the table it protects providing greater security. Reduced application maintenance and complexity.

DEMO

Row level security

Dynamic Data Masking

Regulatory Compliance

A strong demand for cloud applications to meet **privacy standards** recommended by regulating authorities.

Sensitive Data Protection

Protects against unauthorized access to sensitive data in the application or using direct access to the database.

Minimal Impact on Existing Apps

- No need to modify existing application queries
- Complimentary to other data protection features

```
ALTER TABLE [Employee] ALTER COLUMN [SocialSecurityNumber]
ADD MASKED WITH (FUNCTION = 'SSN()')

ALTER TABLE [Employee] ALTER COLUMN [Email]
ADD MASKED WITH (FUNCTION = 'EMAIL()')

ALTER TABLE [Employee] ALTER COLUMN [Salary]
ADD MASKED WITH (FUNCTION = 'RANDOM(1,20000)')

GRANT UNMASK to admin1
```

DEMO

Dynamic Data Masking

Always Encrypted

Prevents Data Disclosure

End-to-end encryption of individual columns in a table with keys that are *never* given to the database system.

Supports Computation on Encrypted Data

In v1, support for equality operations incl. join, group by and distinct operators. Support for additional computations post v1.

Application Transparency

Minimal application changes via server and client library enhancements.

Allows customers to store sensitive data outside of their trust boundary.

Data remains protected from high-privileged, yet unauthorized users incl. rouge admins & hackers.

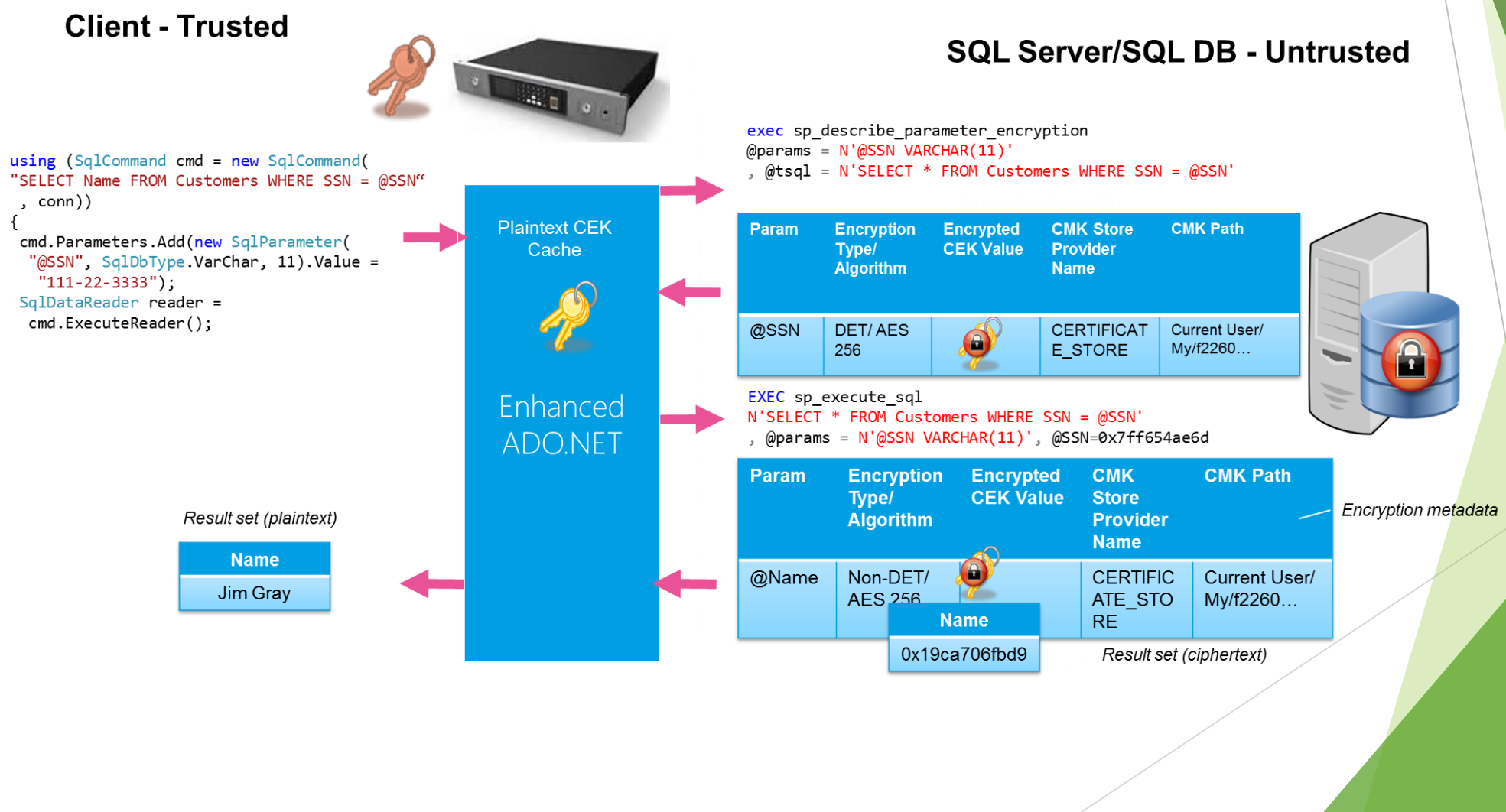
Encryption Types

- Two types of encryption available
  - Randomized encryption uses a method that encrypts data in a less predictable manner
  - Deterministic encryption uses a method which always generates the same encrypted value for any given plain text value

```
Randomized encryption
    Encrypt('123-45-6789') = 0x17cfd50a
    Repeat: Encrypt('123-45-6789') =
    0x9b1fcf32
    Allows for transparent retrieval of
    encrypted data but NO operations
    More secure

Deterministic encryption
    Encrypt('123-45-6789') = 0x85a55d3f
    Repeat: Encrypt('123-45-6789') =
    0x85a55d3f
    Allows for transparent retrieval of
    encrypted data AND equality
    comparison
        E.g. in WHERE clauses and
        joins, distinct, group by
```

How It Works


Client - Trusted
SQL Server/SQL DB - Untrusted
using (SqlCommand cmd = new SqlCommand(
"SELECT Name FROM Customers WHERE SSN = @SSN"
, conn))
{
cmd.Parameters.Add(new SqlParameter(
"@SSN", SqlDbType.VarChar, 11).Value =
"111-22-3333");
SqlDataReader reader =
cmd.ExecuteReader();
Plaintext CEK Cache
Enhanced ADO.NET
exec sp_describe_parameter_encryption
@params = N'@SSN VARCHAR(11)'
, @tsql = N'SELECT * FROM Customers WHERE SSN = @SSN'
Param
Encryption Type/ Algorithm
Encrypted CEK Value
CMK Store Provider Name
CMK Path
@SSN
DET/ AES 256
CERTIFICATE_STORE
Current User/ My/f2260…
EXEC sp_execute_sql
N'SELECT * FROM Customers WHERE SSN = @SSN'
, @params = N'@SSN VARCHAR(11)', @SSN=0x7ff654ae6d
Param
Encryption Type/ Algorithm
Encrypted CEK Value
CMK Store Provider Name
CMK Path
@Name
Non-DET/ AES 256
CERTIFICATE_STORE
Current User/ My/f2260…
Encryption metadata
Result set (plaintext)
Name
Jim Gray
Name
0x19ca706fbd9
Result set (ciphertext)

DEMO

Always Encrypted

Summary

- Without strong encryptions of your data local OS admin has a way inside to your SQL server
- Keep your local admins at minimum
- Keep your sysadmins at minimum
- Protect your backups
- Keep your instance patched with latest updates

Session End

Ing. Marek Chmel, MSc

Lead Database Administrator, Cloud, Platform, Application & Data Layer Team

mc654x@att.com or marek.chmel@technet.ms