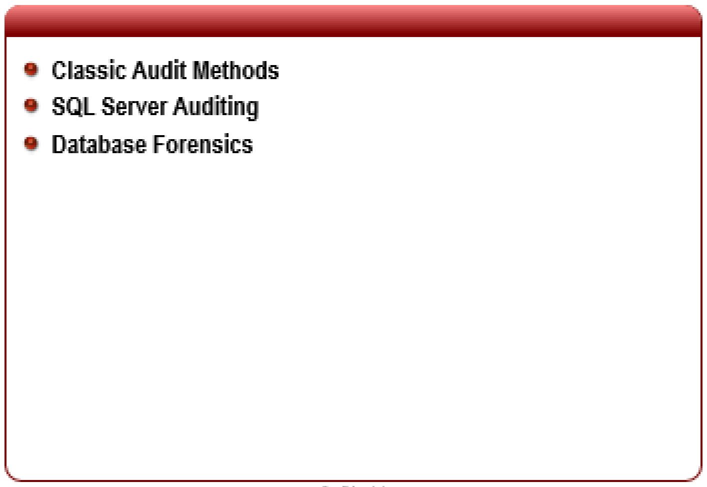
Module 5:
Auditing on SQL Server

Contents

Module Overview


- Classic Audit Methods
- SQL Server Auditing
- Database Forensics

It is now mandatory for every information system to have efficient access control elements. Without the necessary information, it is difficult to know what is occurring. Depending on the type of information system, the consequences for not having the necessary information can be small, medium or even catastrophic.

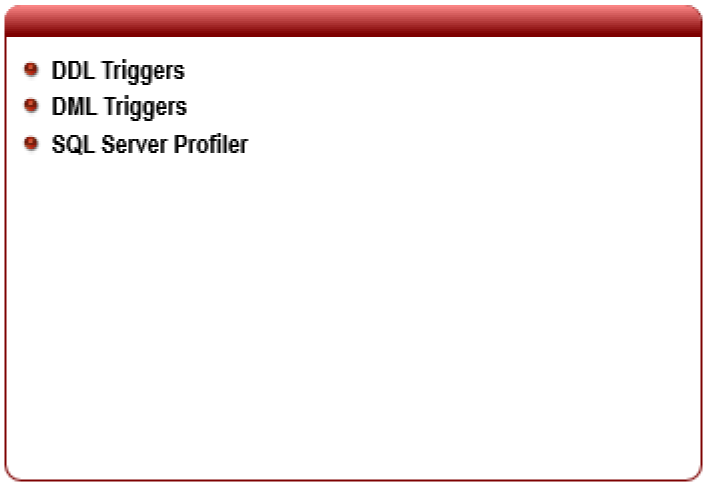
The database is the last element in every software architecture. It is crucial to implement access control on the database-level and prevent digital evidence tampering.

Objectives

After completing this module, you will be able to:

- Implement classic auditing on SQL Server
- Implement a new Audit feature in SQL Server 2012
- Understand the basics about database forensics

Lesson 1: Classic Audit Methods



- DDL Triggers
- DML Triggers
- SQL Server Profiler

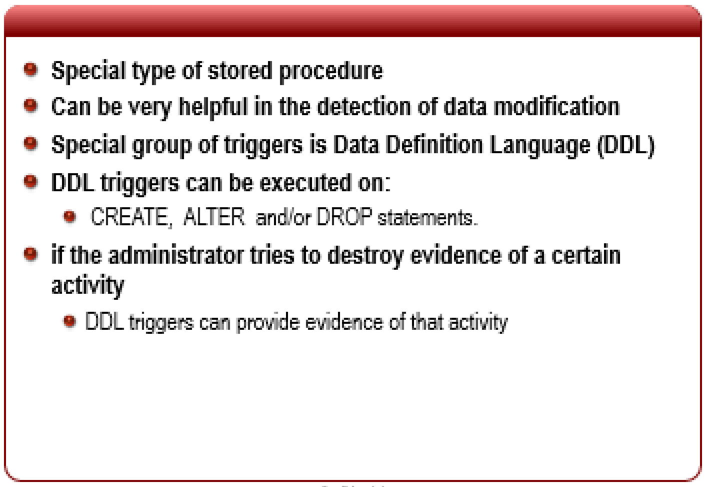
Business processes produce a large amount of data in government agencies, universities, and enterprises on a daily basis. Therefore, a secure environment for storing data is imperative. Cases in which data is maliciously modified (data tampering, data fraud and unauthorized data gathering) can produce serious and long-term consequences. Data tampering can be done with unauthorized access and, in some cases, through authorized users. Results of data tampering can be unpleasant for businesses and their clients. This lesson will explain some of the classic best case scenarios for auditing.

Objectives

After completing this lesson, you will be able to:

- Understand and use DDL triggers
- Understand and use DML triggers
- Use SQL Server Profiler for tracking user activity

DDL Triggers



- Special type of stored procedure
- Can be very helpful in the detection of data modification
- Special group of triggers is Data Definition Language (DDL)
- DDL triggers can be executed on:
  - CREATE, ALTER and/or DROP statements.
- if the administrator tries to destroy evidence of a certain activity
  - DDL triggers can provide evidence of that activity

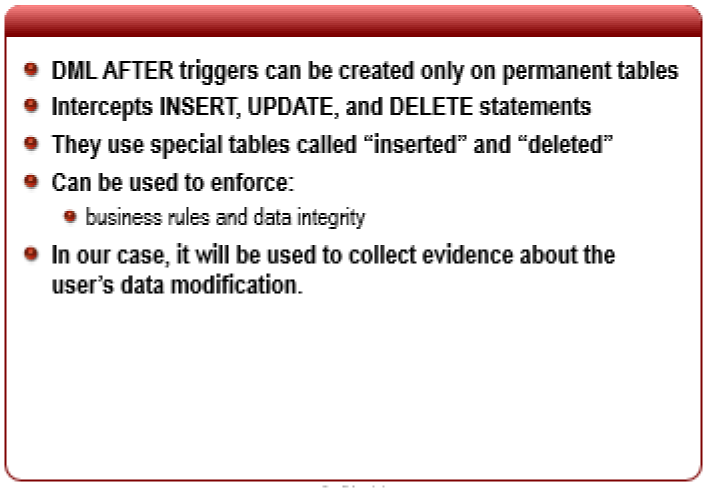
A trigger is a special type of stored procedure that is automatically executed when an event has occurred. If used properly, it can be very helpful in the detection of data modification. Trigger efficiency is based on the fact that each transaction passes through a layer for detecting data modification. This layer is positioned in the last line of defense of software environments—the database itself. Any other position can be manipulated or avoided.

A special group of triggers is Data Definition Language (DDL). DDL triggers can be executed on a specific event when someone tries to change or create some database object (i.e. CREATE, ALTER and/or DROP statements). It is very rare that a user application can modify a database structure. But what if the administrator tries to destroy evidence of a certain activity? DDL triggers can provide evidence of that activity.

```
CREATE TRIGGER DDLeventAccessControl ON DATABASE
FOR DDL_DATABASE_LEVEL_EVENTS
AS
      DECLARE @data XML
      DECLARE @cmd NVARCHAR(1000)
      DECLARE @posttime NVARCHAR(24)
      DECLARE @hostname NVARCHAR(100)
      DECLARE @loginname NVARCHAR(100)
            SET @data = eventdata()
            SET @cmd = CONVERT(NVARCHAR(1000),
            @data.query('data(//TSQLCommand//CommandText)'))
            SET @posttime = CONVERT(NVARCHAR(24),
            @data.query('data(//PostTime)'))
            SET @hostname = HOST_NAME()
            SET @loginname = SYSTEM_USER

INSERT INTO dbo.DDLAuditTable(Cmd,PostTime,HostName,LoginName)
VALUES(@cmd, @posttime, @hostname, @loginname)
SELECT @data
GO
```

DML Triggers



- DML AFTER triggers can be created only on permanent tables
- Intercepts INSERT, UPDATE, and DELETE statements
- They use special tables called "inserted" and "deleted"
- Can be used to enforce:
  - business rules and data integrity
- In our case, it will be used to collect evidence about the user's data modification.

Data Manipulation Language (DML) AFTER triggers can be created only on permanent tables. This database objects cannot be created on views or temp tables. You create such triggers on a specific table and for a specific DML statement or statement list, including INSERT, UPDATE, and DELETE. Within DML triggers, you can access the old and new data of the affected rows through special tables called "*inserted*" and "*deleted*". The inserted table contains the new data of the affected rows, while the deleted table contains the old data. The inserted table will contain rows only for INSERT and UPDATE triggers. Also, the deleted table will contain rows only for DELETE and UPDATE triggers. The inserted and deleted tables have the same structure as the table on which the trigger was defined.

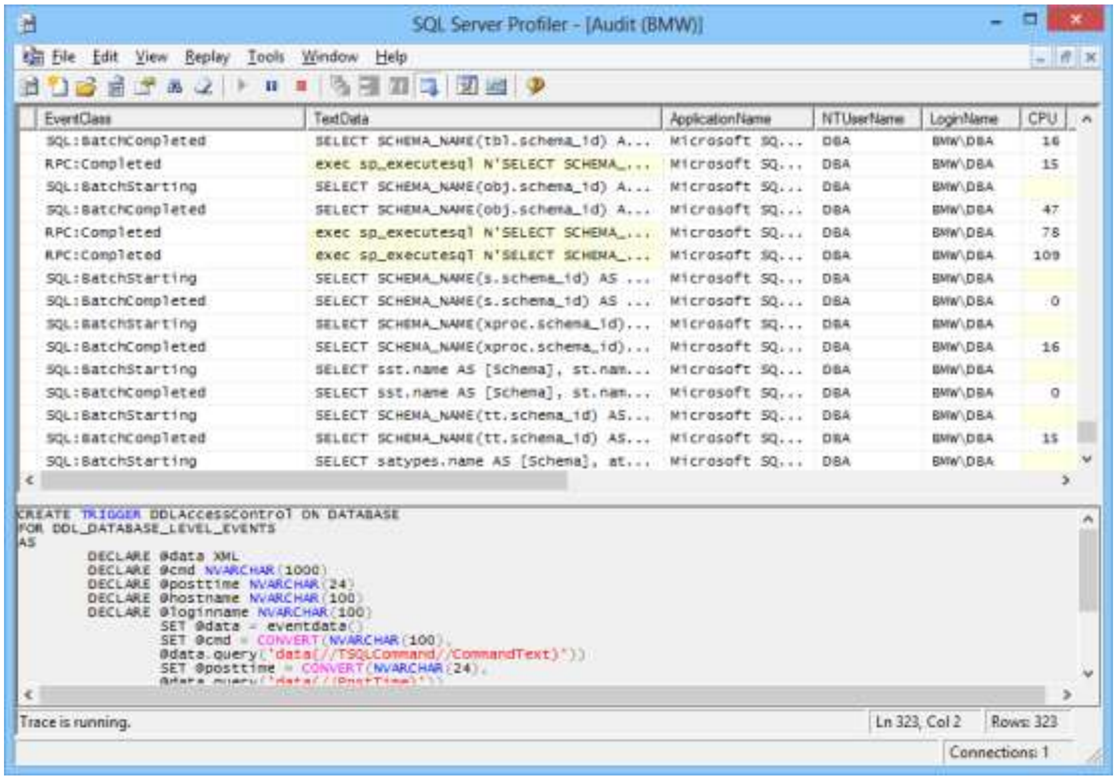
DML triggers can also be used to enforce business rules and data integrity. In this case, DML AFTER triggers will be used to collect evidence about the user's data modification.

SQL Server Profiler

- SQL Server Profiler is a standard tool for performance auditing and tuning
- Powerful tool for monitoring user activity
- Is not a replacement for triggers or other audit tools and methods

SQL Server Profiler is a rich interface to create and manage traces, as well as to analyze and replay trace results. The events are saved in a trace file that can later be analyzed or used to replay a specific series of steps when trying to diagnose a problem.

SQL Server Profiler is also a standard tool for performance auditing and tuning. Furthermore, it is also a powerful tool for monitoring user activity and tracking changes


SQL Server Profiler - [Audit (BMW)]
File Edit View Replay Tools Window Help

EventClass	TextData	ApplicationName	NTUserName	LoginName	CPU
SQL:BatchCompleted	SELECT SCHEMA_NAME(tbl.schema_id) A...	Microsoft SQ...	DBA	BMW\DBA	16
RPC:Completed	exec sp_executesql N'SELECT SCHEMA_...	Microsoft SQ...	DBA	BMW\DBA	15
SQL:BatchStarting	SELECT SCHEMA_NAME(obj.schema_id) A...	Microsoft SQ...	DBA	BMW\DBA	
SQL:BatchCompleted	SELECT SCHEMA_NAME(obj.schema_id) A...	Microsoft SQ...	DBA	BMW\DBA	47
RPC:Completed	exec sp_executesql N'SELECT SCHEMA_...	Microsoft SQ...	DBA	BMW\DBA	78
RPC:Completed	exec sp_executesql N'SELECT SCHEMA_...	Microsoft SQ...	DBA	BMW\DBA	109
SQL:BatchStarting	SELECT SCHEMA_NAME(s.schema_id) AS ...	Microsoft SQ...	DBA	BMW\DBA	
SQL:BatchCompleted	SELECT SCHEMA_NAME(s.schema_id) AS ...	Microsoft SQ...	DBA	BMW\DBA	0
SQL:BatchStarting	SELECT SCHEMA_NAME(xproc.schema_id)...	Microsoft SQ...	DBA	BMW\DBA	
SQL:BatchCompleted	SELECT SCHEMA_NAME(xproc.schema_id)...	Microsoft SQ...	DBA	BMW\DBA	16
SQL:BatchStarting	SELECT sst.name AS [Schema], st.nam...	Microsoft SQ...	DBA	BMW\DBA	
SQL:BatchCompleted	SELECT sst.name AS [Schema], st.nam...	Microsoft SQ...	DBA	BMW\DBA	0
SQL:BatchStarting	SELECT SCHEMA_NAME(tt.schema_id) AS...	Microsoft SQ...	DBA	BMW\DBA	
SQL:BatchCompleted	SELECT SCHEMA_NAME(tt.schema_id) AS...	Microsoft SQ...	DBA	BMW\DBA	15
SQL:BatchStarting	SELECT satypes.name AS [Schema], at...	Microsoft SQ...	DBA	BMW\DBA	

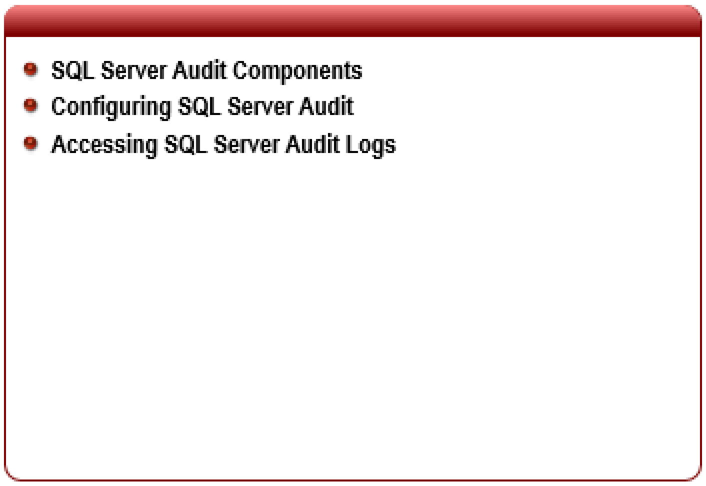
```
CREATE TRIGGER DDLAccessControl ON DATABASE
FOR DDL_DATABASE_LEVEL_EVENTS
AS
        DECLARE @data XML
        DECLARE @cmd NVARCHAR(1000)
        DECLARE @posttime NVARCHAR(24)
        DECLARE @hostname NVARCHAR(100)
        DECLARE @loginname NVARCHAR(100)
                SET @data = eventdata()
                SET @cmd = CONVERT(NVARCHAR(100),
                @data.query('data(//TSQLCommand//CommandText)'))
                SET @posttime = CONVERT(NVARCHAR(24),
                @data.query('data(//PostTime)'))
```

```
Trace is running.                                   Ln 323, Col 2   Rows: 323
                                                    Connections: 1
```

SQL Server Profiler is not a replacement for triggers or other audit tools and methods. It can be used as an add-on with good real-time monitoring to get quick inside information.

Keep in mind that SQL Server Profiler can affect performance on overloaded systems.

Lesson 2: SQL Server Auditing



- SQL Server Audit Components
- Configuring SQL Server Audit
- Accessing SQL Server Audit Logs

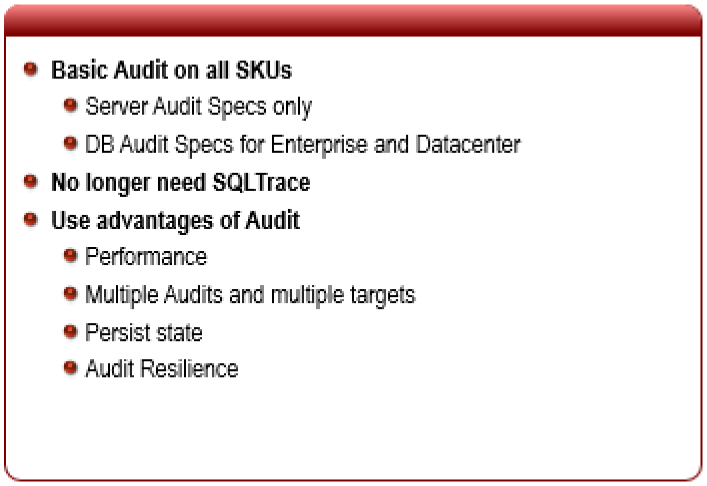
SQL Server 2012 brings a new way of collecting data about a user’s activity. It is built-in to the SQL Server Engine. This is a very good feature that will replace alternative ways of auditing (i.e. Triggers, Profiler and third parity tools).

Objectives

After completing this lesson, you will be able to:

- Recognize SQL Server Audit components
- Configure SQL Server audit
- Access SQL Server audit logs

SQL Server Audit Components



- **Basic Audit on all SKUs**
  - Server Audit Specs only
  - DB Audit Specs for Enterprise and Datacenter
- **No longer need SQLTrace**
- **Use advantages of Audit**
  - Performance
  - Multiple Audits and multiple targets
  - Persist state
  - Audit Resilience

Auditing of the SQL Server or database itself involves tracking and logging all events that occur on the server/database environment. SQL Server Audit helps you to create server audits (contain server audit specifications for server-level events) and database audit specifications (database-level events). Audited events can be written to the event logs or to audit files. All editions of SQL Server support server-level audits. Database-level auditing is limited to Enterprise, Developer, and Evaluation editions.

SQL Server Audit

The SQL Server Audit object collects a single instance of server- or database-level actions and groups of actions to monitor. The audit is at the SQL Server instance-level. You can have multiple audits per SQL Server instance.

Server Audit Specification

The Server Audit Specification object belongs to an audit. You can create one server audit specification per audit, because both are created at the SQL Server instance scope.

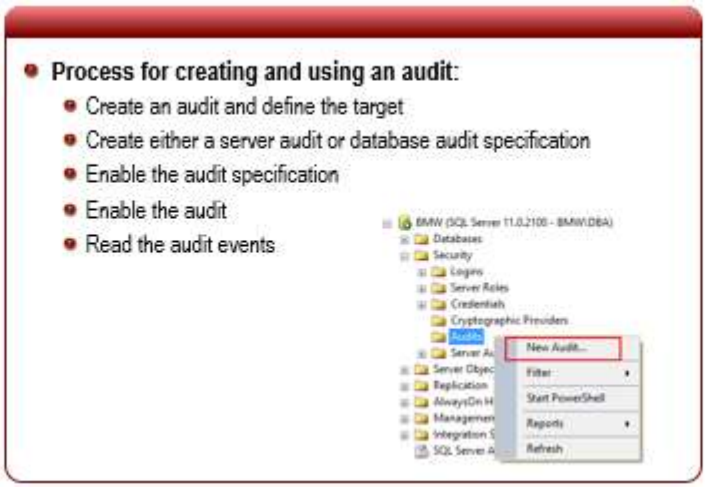
Database Audit Specification

The Database Audit Specification object belongs to a SQL Server audit. You can create one database audit specification per audit per SQL Server database. The Database Audit Specification collects database-level audit actions.

Target

The results of an audit are sent to a target: file, Windows Security event log or Windows Application event log.

Configuring SQL Server Audit

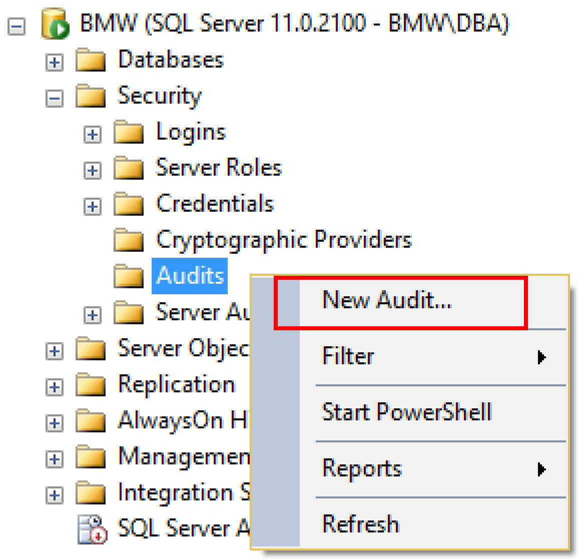


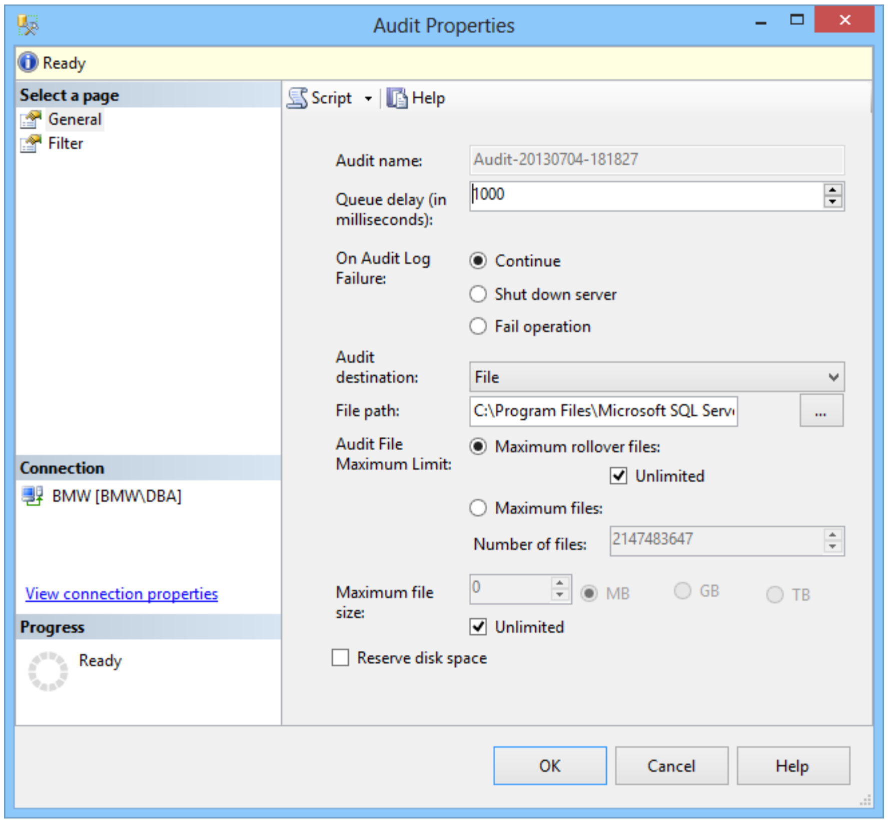
- Process for creating and using an audit:
  - Create an audit and define the target
  - Create either a server audit or database audit specification
  - Enable the audit specification
  - Enable the audit
  - Read the audit events

You can use SQL Server Management Studio or T-SQL to define an audit. After the audit is created and enabled, the target will receive entries.

The general process for creating and using an audit is as follows:

1. Create an audit and define the target.
2. Create either a server audit specification or database audit specification that maps to the audit. Enable the audit specification.
3. Enable the audit.
4. Read the audit events by using the Windows Event Viewer, Log File Viewer, or the fn_get_audit_file function.





BMW [BMW\DBA]

Audit name: Audit-20130704-181827

Queue delay (in milliseconds): 1000

On Audit Log Failure: [✓ Continue] Shut down server Fail operation

Audit destination: File

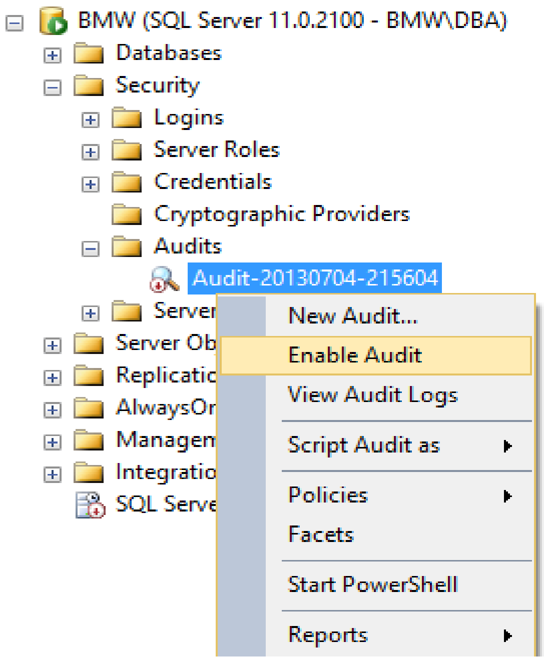
File path: C:\Program Files\Microsoft SQL Serv

Audit File Maximum Limit: [✓ Maximum rollover files:] [✓ Unlimited] Maximum files:

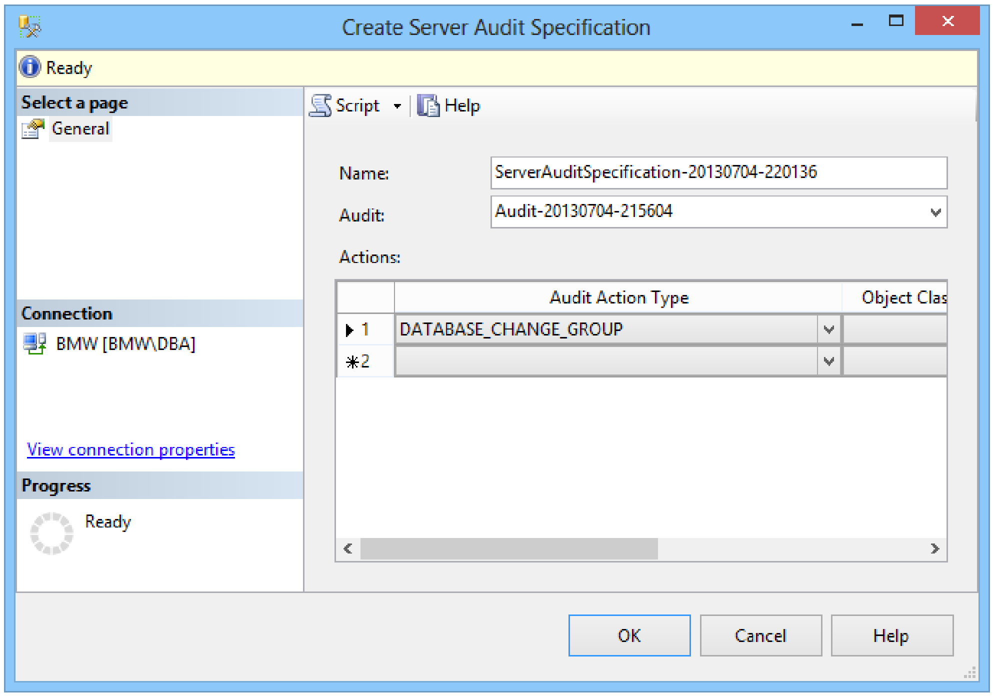
Number of files: 2147483647

Maximum file size: 0 [✓ MB] GB TB [✓ Unlimited]

Reserve disk space



After finishing those initial steps, we can create either a server audit specification or database audit specification.



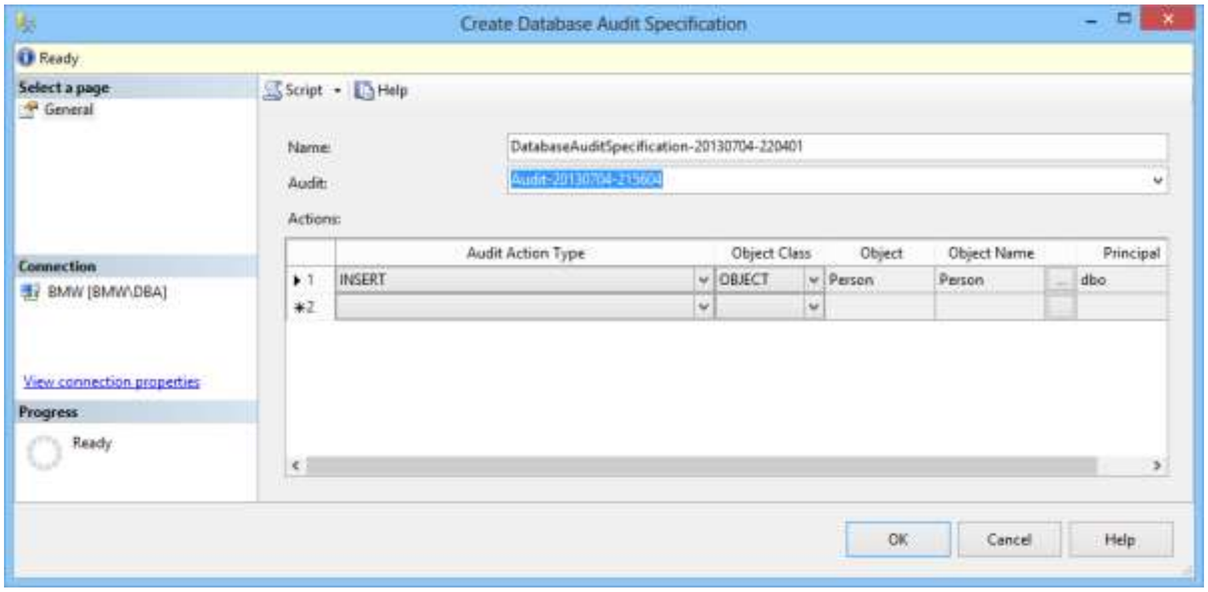
BMW [BMW\DBA]

Name: ServerAuditSpecification-20130704-220136

Audit: Audit-20130704-215604

Actions:

	Audit Action Type	Object Clas
1	DATABASE_CHANGE_GROUP	
2		



BMW [BMW\DBA]

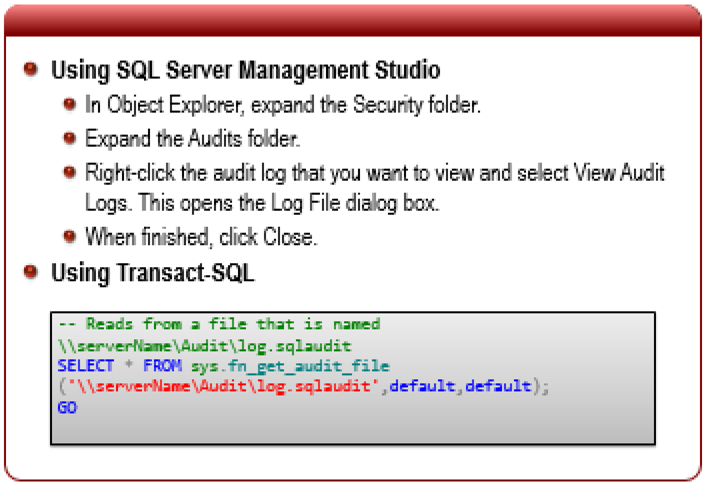
Name: DatabaseAuditSpecification-20130704-220401

Audit: Audit-20130704-215604

Actions:

	Audit Action Type	Object Class	Object	Object Name	Principal
1	INSERT	OBJECT	Person	Person	dbo
2					

Accessing SQL Server Audit Logs



- Using SQL Server Management Studio
  - In Object Explorer, expand the Security folder.
  - Expand the Audits folder.
  - Right-click the audit log that you want to view and select View Audit Logs. This opens the Log File dialog box.
  - When finished, click Close.
- Using Transact-SQL

```
-- Reads from a file that is named
\\serverName\Audit\log.sqlaudit
SELECT * FROM sys.fn_get_audit_file
('\\serverName\Audit\log.sqlaudit',default,default);
GO
```

Accessing collected data is quite simple. It can be accessed through GUI or T-SQL, like any other operation on SQL Server.

Using SQL Server Management Studio

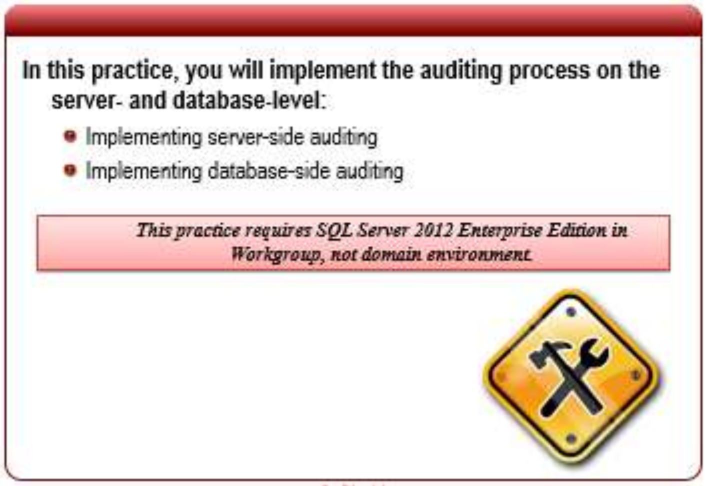
1. In **Object Explorer**, expand the **Security** folder.
2. Expand the **Audits** folder.
3. Right-click the audit log that you want to view and select **View Audit Logs**. This opens the **Log File** dialog box.
4. When finished, click **Close**.

Using Transact-SQL

1. In **Object Explorer**, connect to an instance of **Database Engine**.
2. On the **Standard** bar, click **New Query**.
3. Write T-SQL code:

```
-- Reads from a file that is named \\serverName\Audit\log.sqlaudit
SELECT * FROM sys.fn_get_audit_file
('\\serverName\Audit\log.sqlaudit',default,default);
GO
```

Practice: Auditing Server and Database Activity



In this practice, you will implement the auditing process on the server- and database-level:

- Implementing server-side auditing
- Implementing database-side auditing

This practice requires SQL Server 2012 Enterprise Edition in Workgroup, not domain environment.

In this practice, you will implement the auditing process on the server- and database-level.

This practice requires SQL Server 2012 Enterprise Edition in Workgroup, not domain environment.

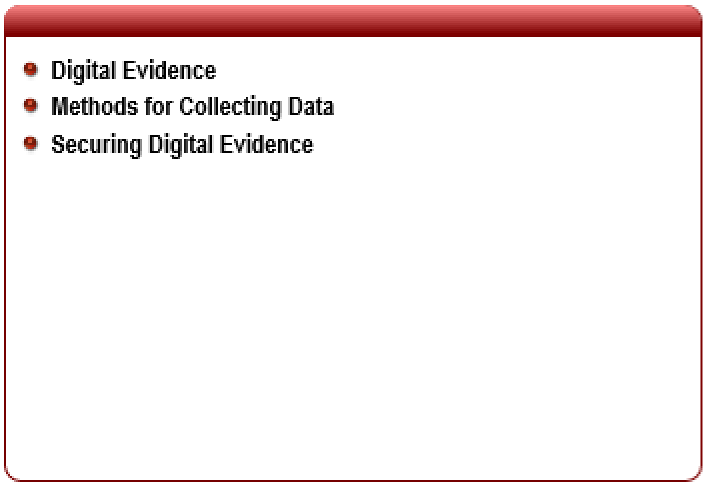
Exercise 1: Implementing server-side auditing

1. Open **Security** node in **Object Explorer** of your SQL Server instance.
2. Expand the **Audit** node.
3. Right-click and select **New Audit**, then name it "**Global Server Audit**".
4. Select a location for the audit log, click **File**, and then click **OK**.
5. Right-click **Global Server Audit** and select **Enable Audit**.
6. Right-click **Server Audit Specification** and click **New Server Audit Specification**.
7. In the new dialog window, do the following:
   i. Select **Global Server Audit** as a base audit object;
   ii. Select the elements on the server-level that you would like to audit; and
   iii. Click **OK**.

Exercise 2: Implementing database-side auditing

1. Open **Security** node in **Object Explorer** of your AdventureWorks2012 sample.
2. Right-click **Database Audit Specification** and select **New Database Audit Specification**.
3. Select **Global Server Audit** as a base audit object.
4. Select the elements on the database-level that you would like to audit, and click **OK.**
5. Click **OK**.

Lesson 3: Database Forensics



- Digital Evidence
- Methods for Collecting Data
- Securing Digital Evidence

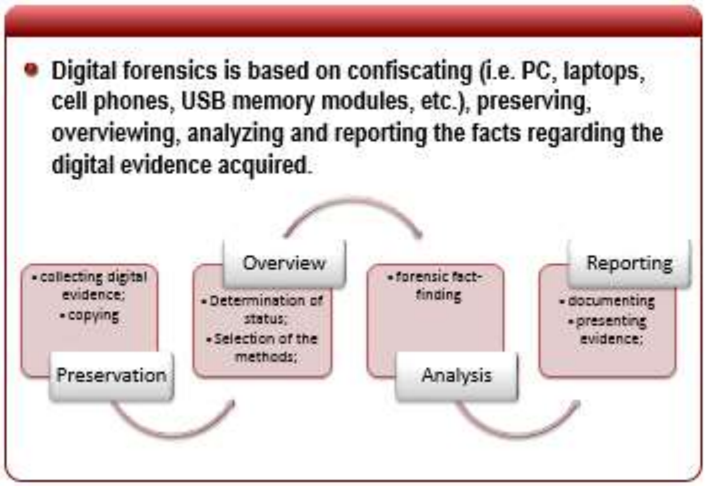
Contemporary information systems, such as eLearning, eUnivesity, eVoting, eHealth, etc., are frequently used and misused for irregular data changes known as data tampering. As such, our security measures must be improved to address this issue. Proving a computer crime act requires very complicated processes which are based on digital evidence collecting, forensic analysis and the investigation process. Forensic analysis of database systems is a very specific and demanding task. In this lesson, you will learn some basic elements about how to protect collected digital evidence at the database-level.

Objectives

After completing this lesson, you will be able to:

- Identify and understand digital evidence
- Select methods for collecting digital evidence
- Secure digital evidence

Digital Evidence



- Digital forensics is based on confiscating (i.e. PC, laptops, cell phones, USB memory modules, etc.), preserving, overviewing, analyzing and reporting the facts regarding the digital evidence acquired.

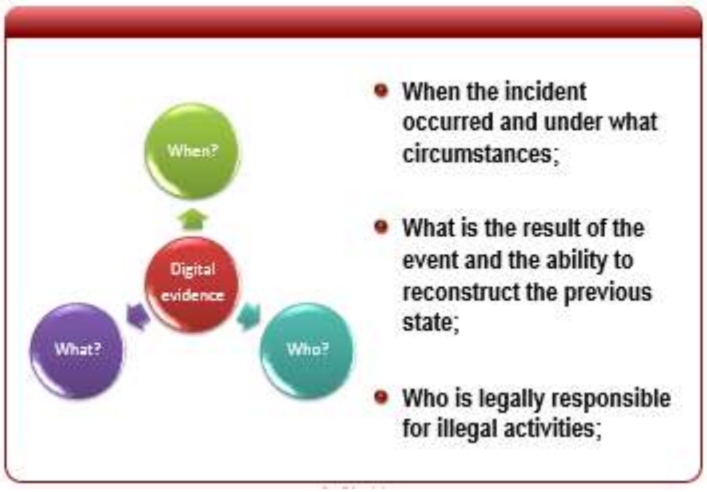
Digital forensic is the most important part of the investigation process. Facts collected in this process need to be presented in a court of law. The process of collecting, analyzing and preserving digital data is based on scientific methods, which are the only methods to ensure the validity of the evidence. On the other hand, digital evidence is defined as any data stored on a computer or transmitted over a network that can support or quash a theory in relation to a digital crime.

Digital forensics is based on confiscating (i.e. PC, laptops, cell phones, USB memory modules, etc.), preserving, overviewing, analyzing and reporting the facts regarding the digital evidence acquired.

Why is a database such an important asset in the monitoring and collection of digital evidence?

- Databases contain critical information for business;
- Database servers keep private, sensitive and secure information;
- Industrial espionage (data stealing and reselling to third parties);
- Database is the last line of defense in the IS and where the damage is greatest;
- Loss of credibility;
- Etc.

Methods for Collecting Data

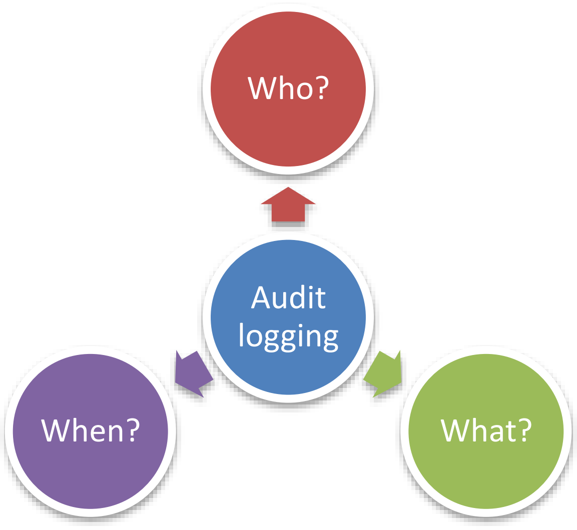


- When the incident occurred and under what circumstances;
- What is the result of the event and the ability to reconstruct the previous state;
- Who is legally responsible for illegal activities;

It is a well-known fact that modern information society has a strong need to provide a secure data storage environment. Keep in mind that a single change in a database can result in a criminal charge.

There is only one way to provide valid forensic analysis of a database. Such validity is achieved by creating an audit which logs all aspects of the information system. Based on that audit log, the data investigation process can reconstruct what has really happened. Depending on the application architecture (web, desktop, combined), there is a wide range of data that is needed. For example, user name, IP address, time stamp are just some of the data required. Please note that not all data is required in each situation. IP address of an internet provider is not needed in a classical desktop environment. In contrast, the IP address of an internet provider can be crucial in determining the geographical location of the criminal.

The basic task of audit logging is to answer three questions: Who, What, and When.



Depending on how much detail is necessary about changed data, audit logging can be divided in two groups: simple and advanced access control.

Simple access control collects basic data about actions in information systems. As this is a common task, we are not going to describe it in more detail. Simple access control gives answers to the “Who” and “When” questions. It is important to remember that simple access control is not enough to provide data for a valid forensic analysis. Ultimately, every system requires at least this group of access control.

On the other hand, there are systems where integrity, data precisions, privacy and security are the primary areas of concern. Such systems are ideal candidates for advanced access control. Answers to the “Who” and “When” questions are not sufficient for the advanced access control environment. The biggest issue for this group of access control is the “What” question.

Who

The first and basic element in this category is the identity of the user. Every information system has its own user implementation, but all of them are based on real user identity. Regardless of the tool used for accessing data, application or direct access to the database layer; a user account is required. Authentication and authorization are the first clues to answer the “Who” question.

The second element is the location of the user. The location and data about the geographic location is important regardless of whether the user accessed the system from the web or a desktop environment.

When

Data collected in the investigation process is invalid without a time dimension. From the aspect of the database, timeframes of a user’s activity can support or quash a user’s alibi. The date and time of login, logout, and the period of activity are all required in this segment of audit logging.

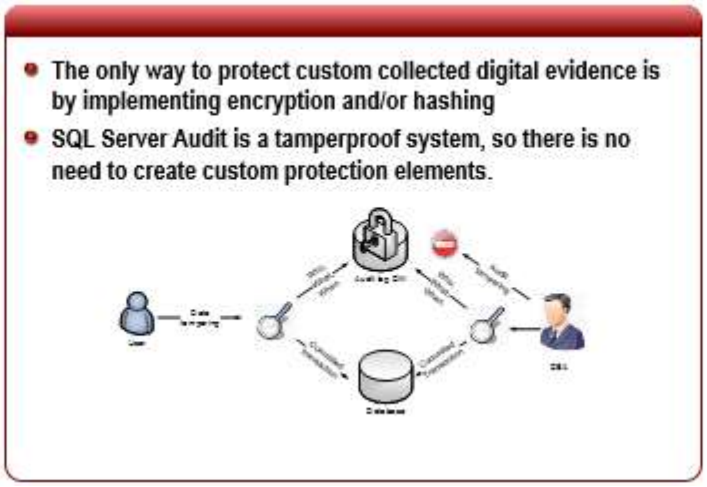
What

The answer to the “What” question is the major difference between the simple and advanced access control groups. The answer to this question plays a key role in this process, and can contain such information as: tampered data (old data), modified data (new data) and/or a validation element. The validation element is a security mechanism for providing an extra layer of protection from users with high access rights (i.e. administrators, DBA, power users, etc.). From the digital forensic point of view, the validation element is a key element. Without the validation element, the forensic analysis is irrelevant.

Digital forensic is the part of the investigation which involves the scientific analysis of the media, storage devices and/or devices for data processing (i.e. servers, PCs, laptops, tablets, mobile devices, storage media, etc.), with the goal of finding traces of the activities that led to the crime.

Digital forensics is a very complex area which is beyond the scope of this course.

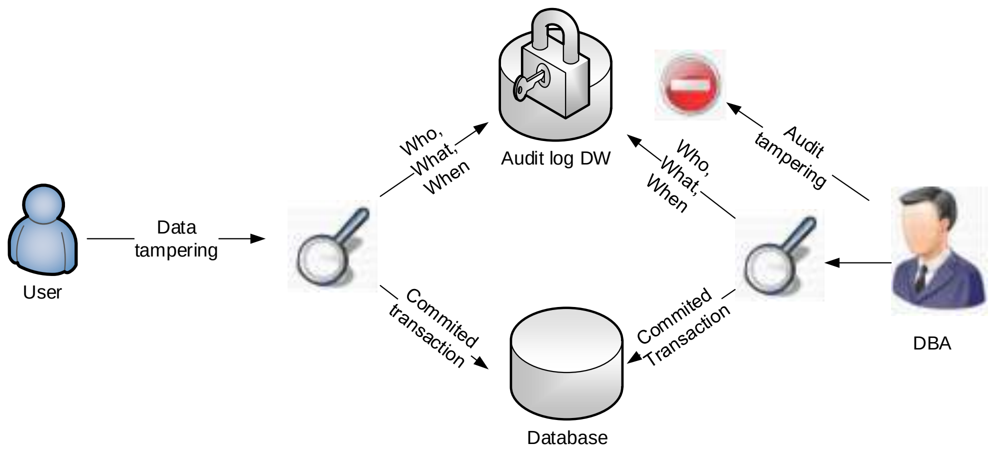
Securing Digital Evidence



- The only way to protect custom collected digital evidence is by implementing encryption and/or hashing
- SQL Server Audit is a tamperproof system, so there is no need to create custom protection elements.

There are many reasons why someone should attempt to use unauthorized access and make malicious data modifications. The Hospital Information System (HIS) is a good example. After examining a patient, a physician gives a diagnosis and prescribes therapy which may or may not incorporate medication. Mistakes in that process can produce serious consequences and even death to a patient. In both cases, medical personnel are highly “motivated” to falsify data in information system.

We can expand this scenario further. We have audit logging in a trigger-based environment. The user with the highest privilege can access the digital evidence repository and tamper with the collected data. As this is a big problem, a process needs to be created whereby the DBA can be detected in the process of tampering data.



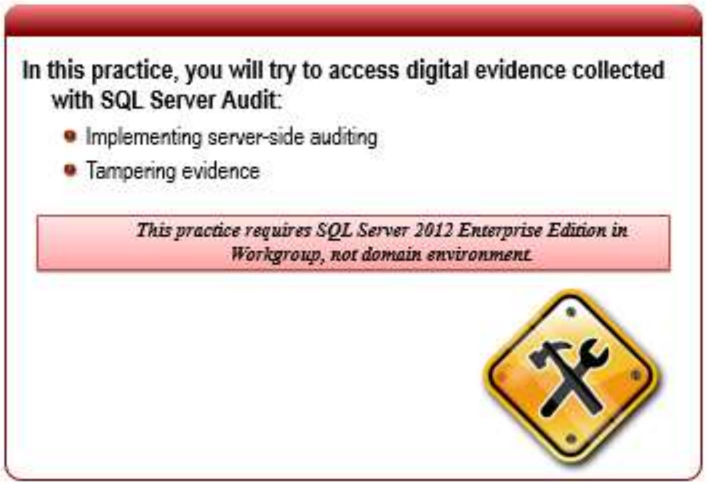
If we use something other than SQL Server Audit, then we need to ensure that the digital evidence cannot be tampered.

The only way to protect the digital evidence is by implementing a combination of cryptographic elements (i.e. encryption and hashing).

SQL Server Audit is a tamperproof system, so there is no need to create custom protection elements.

It is good practice to keep your digital evidence in more than one location. A remote location would be best as it will increase the integrity of the whole system.

Practice: Securing Digital Evidence


In this practice, you will try to access digital evidence collected with SQL Server Audit:

- Implementing server-side auditing
- Tampering evidence

This practice requires SQL Server 2012 Enterprise Edition in Workgroup, not domain environment.

In this practice, you will try to access digital evidence collected with SQL Server Audit.

This practice requires SQL Server 2012 Enterprise Edition in Workgroup, not domain environment.

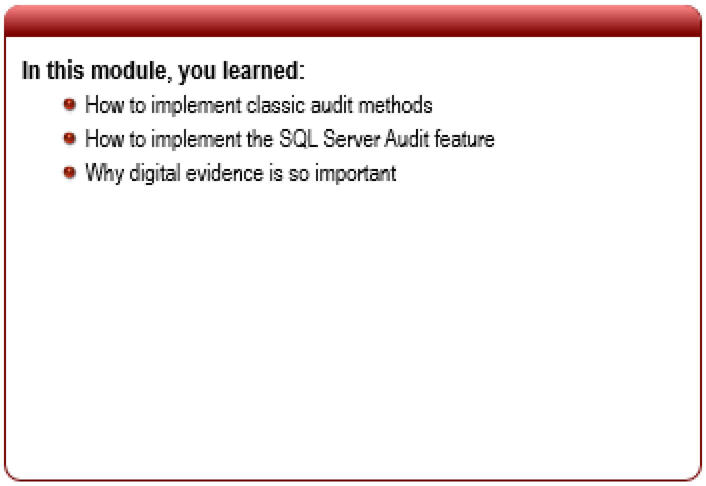
Exercise 1: Implementing server-side auditing

1. Open **Security** node in **Object Explorer** of your SQL Server instance.
2. Expand the **Audit** node.
3. Right-click and select **New Audit**, then name it “**Tamper protected**”.
4. Select a location for the audit log, select **Security** log, then click **OK**.
5. Right-click **Tamper protected**, and then click **Enable Audit**.
6. Right-click **Server Audit Specification**, select **New Server Audit Specification**.
7. In the new dialog window, do the following:
   i. Select **Global Server Audit** as a base audit object;
   ii. Select the elements at the server-level that you would like to audit; and
   iii. Click **OK**.

Exercise 2: Tampering evidence

1. Locate **Security** log in your Windows environment.
2. Try to make some modifications.

Summary


In this module, you learned:

- How to implement classic audit methods
- How to implement the SQL Server Audit feature
- Why digital evidence is so important

In today's IT world, collecting data about a user's activity has many security, privacy, and ethical issues. However, such data collection is required by law and requested by governmental institutions. In many cases, the collection of data is beyond the user's right to privacy. As such, there are many situations where the collection of data is not justified. Yet still, systems which are not monitored are at risk of various damage. In any case, SQL Server provides you with strong features to collect and secure you digital evidence repository.

Objectives

After completing this module, you learned:

- How to implement classic audit methods
- How to implement the SQL Server Audit feature
- Why digital evidence is so important