Creating a successful auditing strategy for your SQL Server databases

By Minette Steynberg

The purpose of security auditing is to identify all attacks, unlawful or malicious activities which might be taking place on your server. Criminals have become increasingly inventive and as a DBA you might not have considered or even be aware of all activities which may be putting your data at risk.

To this end, keeping track of who has access, who has attempted access, and what has been done by people with access has become a crucial activity in identifying and pinpointing vulnerabilities in your security strategy.

In this white paper I will provide some guidelines as to how you can create a suitable auditing strategy for your environment and which SQL Server features may be useful to you in your quest for the perfect auditing strategy.

The importance of auditing

Why do we audit? The number one goal of auditing should be to ensure that your security policy is sufficient in keeping your business’s and customers’ information secure and that your risks are being managed.

The government and other regulatory bodies of specific industries all have prescribed requirements which must be adhered to, in order to be considered compliant or to avoid fines and other punishments, such as PCI, HIPAA, SOX etc. In some cases, the only regulations might be those imposed by the business itself.

If your SQL Server is not being audited, you are exposing yourself to a large amount of potential issues. The most common issues are external attacks, unsanctioned activities by authorised users and mistakes.

Having auditing in place, allows for forensic analysis after an incident has been identified. Which in turn will allow you to implement suitable counter measures and hold employees accountable for their actions.

Auditing strategy

Devising a solid auditing strategy is at the heart of any successful auditing endeavour. It is quite tempting to just audit everything, but in a real world environment this is just not feasible and may lead to problems such as

- An overabundance of data, which will make it hard to identify any issues.
- Large amounts of storage space required.
- Affecting performance adversely.

Before implementing any auditing solutions the following needs to be considered:

1. What are you legally required to audit?
2. How long are you legally required to keep the audit data?
3. How much storage space do you have?
4. What is the purpose of the audited data, is it just for compliance, or will someone actively review the accumulated data to identify vulnerabilities?
5. How critical is it that auditing happens?
Creating a successful auditing strategy for your SQL Server databases

What to audit

Identifying what really needs to be audited is the biggest challenge in creating an auditing strategy. Understanding the data in your database is a fundamental to the success of your auditing strategy.

From a logical perspective data that needs to be protected and to which all access should be audited generally falls into these categories:

- Data which can be used to identify a person such as a passport number or social security number.
- Data which provides personal information about a person, such as his hair and eye colour or even something as simple as the type of movies he likes to watch.
- Data which can be considered sensitive. This basically refers to information which can negatively or in some cases favourably affect those involved should it fall into the wrong hands. This typically includes information such as a person’s health record or financial information such or financial information pertaining to a company or person.

From a database perspective, these are the actions that should be considered for auditing:

- Logons
- Configuration
- Audit configuration
- Schema modification
- Data modifications

There is a fine balance between auditing too little and auditing too much. This depends to a large extend on your reasons for auditing. Even though your first objective should be to keep your customers data safe, many companies take this only as far as following the prescribed measures enforced by compliance rules for their specific industry.

If you measure the security of your data by your ability to pass your audit, then by all means just audit what is prescribed, since this is exactly what these requirements are for. Complying with your industries regulation for security compliance should already cover you for most vulnerabilities.

As a best practice suggestion, also audit your audits, to ensure that no one has tampered with what is being audited, and that you are covered for all eventualities.

If you intend to use the information gathered to fortify your server, make sure you also audit successful events as well as unsuccessful events. Not only will it help you to determine who is abusing their privileges, but it will also allow you to see if failed attacks may have succeeded after multiple attempts. SQL Attacks are most commonly directed at the SA user, so if you see multiple logon failures for the SA user and then a successful logon, it’s not a stretch to think that someone may have obtained unauthorised access to your system.

Auditing your auditing implementation

It is not an uncommon occurrence for DBA’s to disable auditing if the server suddenly starts performing poorly.

When this occurs DBAs may forget to re-enable all the tasks which have been temporarily disabled, and this may then lead to increased vulnerability. Auditing your auditing implementation can help to circumvents this issue.

Ideally a notification should be sent if changes have been made to the auditing configuration, whether as a reminder
Creating a successful auditing strategy for your SQL Server databases

to the DBA to re-enable auditing or to the auditor to let them know that someone has modified the auditing implementation.

This will also highlight any unsanctioned activities performed by authorised users who had the forethought to disable auditing prior to performing their activity.

Selective auditing
In some cases it may be necessary to apply selective auditing either on a per user, or a per object basis.
I.e. A Company may want to keep track of who looks at their salary data or a specific employee has raised some red flags, and all of his activities needs to be tracked to determine his guilt or his innocence.

Archiving auditing information
Because of the potential volume of audit information, it will become necessary to archive the data. Having a central repository for all auditing data will also make it easier to identify trends and to process the information.

Additionally, keeping archived auditing information will allow you review past audits again, in the light of new transgressions which may only recently have been discovered.

Separation of duties to ensure that only authorised personnel which excludes system administrators have access to the auditing information also helps to prevent data tampering of audit information.

SQL Server features available to facilitate security auditing
SQL Server has a host of features to facilitate auditing. Depending on your auditing requirements one or a combination of these features may be the best way to achieve your goal.

C2 Auditing
SQL Server has offered C2 auditing since SQL Server 2000 in order to be classified under the C2 auditing criteria prescribed by the Department of Defence.

Auditble events includes the execution of stored procedures, the creation or deletion of objects and user login and logout activity. In the case of C2 auditing, it is not possible to choose what to audit. It basically audits everything or nothing.

C2 auditing data is saved to a log file in the MSSQL\DATA directory. Each file is limited to 200MB and a new file is created, each time the limit is reached, until the directory runs out of space or auditing is disabled.

To turn on C2 Auditing you can

- Check the Enable C2 audit tracing check box in the security tab of the server properties dialog.
Creating a successful auditing strategy for your SQL Server databases

- Use the following T-SQL code

  ```sql
  sp_configure 'show advanced options', 1;
  GO
  RECONFIGURE;
  GO

  sp_configure 'c2 audit mode', 1;
  GO
  RECONFIGURE;
  GO
  ```

An important fact to note about C2 auditing, is that when it is enabled, and the server is for some reason not able to log an event, the server will shut down. This is to prevent any activities taking placed which are not audited.

When this happens, there are only 2 solutions to the problem.
Creating a successful auditing strategy for your SQL Server databases

1. You need to make some space on the server that auditing can continue before restarting the server or
2. You need to restart SQL Server with the –f flag to bypass auditing.

C2 auditing will be deprecated in a future version and will be replaced with Common Criteria Compliance.

Common Compliance Criteria
Common criteria compliance was introduced in SQL Server 2005 SP1 but is unfortunately only available in Enterprise, Development and Evaluation Editions of SQL Server.

Common compliance criteria was created by a group of nations to achieve the following:

- Increase availability of security enhanced IT products
- To help users to evaluate IT products
- To improve consumer confidence in IT product security.

An international body which is recognised by the International Standards Organisation (ISO) is responsible for maintaining the common compliance criteria.

When you enable the common compliance enabled options this is what will happen:

- Residual information Protection will take place. This means that memory allocations will be over-written with a known pattern of bits before it can be allocated to a new resource.
- Logins will be audited. This includes successful and unsuccessful logins.
- Table level DENYs will take precedence over column level GRANT.

To enable the common criteria enable option you can

- Check the Enable Common Criteria compliance check box in the security tab of the server properties dialog.
Creating a successful auditing strategy for your SQL Server databases

- Use the following T-SQL Script

```sql
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'common criteria compliance enabled', 1;
GO
RECONFIGURE
GO
```

SQL Server Auditing

SQL Server Auditing was first released in SQL Server 2008 as an enterprise only feature. In SQL Server 2012 server level auditing is now also available to Standard Edition users.

The auditing feature was designed to provide a secure auditing platform with a minimal performance impact, which is easy to manage and able to satisfy all audit queries.
Creating a successful auditing strategy for your SQL Server databases

In order to use SQL Auditing, 3 components may be configured:

- **Server Audit (required)**
  The server audit resides in the master database and is used to define where audit information will be stored, whether the auditing should be synchronous or asynchronous, how the files should be handled, and what happens if an audit event cannot be logged.
  This is available to all editions of SQL Server from 2012 onwards.

- **Server Audit specification (optional)**
  This is where server level events can be audited. In order to create a Server Audit specification a Server Audit needs to be created first. An Audit specification depends on a Server Audit and there can be only one Server Audit Specification per Server Audit. Server Audit specifications allows you to audit things like, successful logins made to the server, DBCC actions etc.
  This is also available on all SQL Server Editions from 2012 onwards.

- **Database Audit specification (optional)**
  The Database audit specification also depends on the server audit. It allows for more granular auditing within a database, including actions performed on specific objects such as which user performed a select on a specific table. Multiple database audit specifications can be linked to a single Server Audit.

The information recorded in the audit logs can be read by using the `fn_get_audit_file` function, as indicated below.

```sql
SELECT * FROM fn_get_audit_file('D:\Audits\*', default, default)
```

**SQL Trace**

Currently SQL Trace is still the preferred method for performing auditing on SQL Server, it has been around since SQL 2000 and most DBA's are familiar with it, or at least with using Profiler.

SQL trace uses a set or stored procedures that record events which have been defined in the trace definition. Traces are highly configurable and can achieve very fine grained auditing.

SQL Server provides Profiler as a front end to SQL Trace, which makes it easy to create traces and run them. Unfortunately profiler does come with some performance penalties and as such it's not always a good idea to run it against your production server.

If you do need to run traces against your production server, do so by using the Transact-SQL statements to manage and run traces. The following are the procedures which can be used to create and read trace information:

- **sp_trace_create**
  This procedure is used to create a trace definition. Trace options such as file rollover or server shutdown and the location of the trace files can also be specified here.

- **sp_trace_setevent**
  This procedure is used to define which trace events you would like to capture and which columns of each event to record.
Creating a successful auditing strategy for your SQL Server databases

- sp_trace_setstatus
  Since all traces are created in stopped state, this procedure is required to start the trace and then stop it again. It can also be used to remove the trace altogether.

One thing to note about traces, is that they are not persistent. If your instance is restarted the trace will not automatically be re-created.

SQL Trace will be deprecated in future versions of SQL Server and will be replaced with Extended Events.

Extended Events
Extended events were introduced in SQL 2008, it is a highly configurable events framework, which will ultimately replace SQL Trace. In SQL 2008 the events available in the extended events engine were limited, but in SQL Server 2012, all of the event which were available in SQL Trace are now also available in Extended Events.

The extended events engine has a very low performance overhead which makes it preferable over SQL Trace. The engine only manages the packages which contains the event information making it extremely flexible. It is also possible to correlate date between applications, SQL server and Windows using Extended Events.

SQL Server 2012 also introduces a user interface for extended events which makes it a lot easier to use without having to understand the underlying architecture of Extended Events framework. The extended events wizard can be found under the management node of your server.
Creating a successful auditing strategy for your SQL Server databases

The wizard will guide you through the creation of a new session using an existing template or help you to create your own selecting the events and event fields you require.

Alternatively you can use Transact-SQL to create traces. The following are the three main commands for creating and managing Extended Event Sessions:

- **CREATE EVENT SESSION**
  This command creates and event session and allows you to specify the events, actions and targets which should be associated to the session. You can see the available events, actions and targets by selecting from `sys.dm_xe_objects` with the object types of event, action and target respectively.

- **ALTER EVENT SESSION**
  This command is used to modify the event sessions configuration and to start or stop the event session.

- **DROP EVENT SESSION**
  This command is used to dispose of an event session when it is no longer required.
Creating a successful auditing strategy for your SQL Server databases

Change Data Capture
Change data capture was introduced to SQL Server in 2008. This enterprise only feature provides practically real time, low impact information gathering on DML changes which can be used for auditing purposes.

This feature harvests the SQL Server transaction log as part of the sp_replcmds process, for changes to data on the tables for which it was enabled. CDC depends on the SQL Server agent which must be running at all times to ensure that data capture takes place. Two jobs are created when CDC is implemented, one is used to capture the data an populate the respective tables, and the other is used to perform the clean-up.

Change data capture can be enabled on your database and specific tables, without having to make any changes to the objects themselves.

The following commands can be used to implement CDC:

- `sys.sp_cdc_enable_db`
  This command enables CDC for use on a specific database, and is required before any tables can be enabled for CDC
- `sys.sp_cdc_enable_table`
  This command enables CDC on a specific table. This creates an associated capture instance of the table, including a change table and query functions. The first 5 columns in the change table contains Meta data to be used by the CDC process and the rest of the columns are based on the columns in the source table to be captured.

DML, DDL and Logon Triggers
Triggers can be used to log logons or changes to objects and data within your database.

DML triggers fire in response to INSERT, UPDATE and DELETE statements while DDL triggers fire in response to changes made to the data definition such as CREATE, DROP, ALTER etc. Logon triggers fire each time a user logs on to SQL Server which can be used to audit access to your server. Unfortunately the trigger does not fire if the user is not authenticated, and as such will only be useful in auditing successful logins.

In order to use DML triggers to audit data changes, triggers will need to be added to each table you wish to audit. This can be cumbersome and hard to maintain. Applications developed on SQL Server often make use of DML triggers for internal auditing.

In the same way DDL triggers can be implemented to record any changes to database objects. DDL triggers are created on database or server level and fires when the specifically configured event occurs.

The following Transact-SQL Statements can be used to create or maintain DML or DDL triggers:

- `CREATE TRIGGER`
  This statement can be used to create either a DML a DDL or a logon trigger. When creating a DDL trigger specifying the scope of the trigger as ON ALL SERVER or ON DATABASE is required.
- `ENABLE/DISABLE TRIGGER`
Creating a successful auditing strategy for your SQL Server databases

Triggers are enabled by default when created. These commands help you to disable a trigger without removing it from the database and then to re-enable it later. You can also disable DML triggers by using the ALTER TABLE command.

Conclusion

However tempting it may be to audit everything that happens in your server, the key to a successful auditing strategy is knowing your data, understanding what needs to be audited, having processes in place to ensure that auditing data is reviewed and that when vulnerabilities are identified, they are addressed and counter measures are implemented.

The best strategy is a strategy tailored to your business, which audits not only the information prescribed by your industrial authority, but also everything you need to identify potential internal and external threats to the security of your sensitive information, while taking into account performance, your compliance requirements as well as your available storage space.

With the myriad of features provided by SQL Server to help you implement the perfect auditing strategy, securing your business’s and your customers’ sensitive data has never been easier. Although some features such as SQL Audit is still fairly young and not yet fully developed, the right combination of features will help you walk the tightrope between compliance, performance, storage requirements and security.

References

SQL Server Common Criteria Certifications
Auditing in SQL Server 2008
Using SQL Server 2008 Extended Events