Global Information Assurance Certification Paper

Incident Response in a Microsoft SQL Server Environment

GIAC (GCIH) Gold Certification

Author: Juan M. Walker CISSP, jwalker@emfbroadcasting.com
Adviser: Robert Vandenbrink

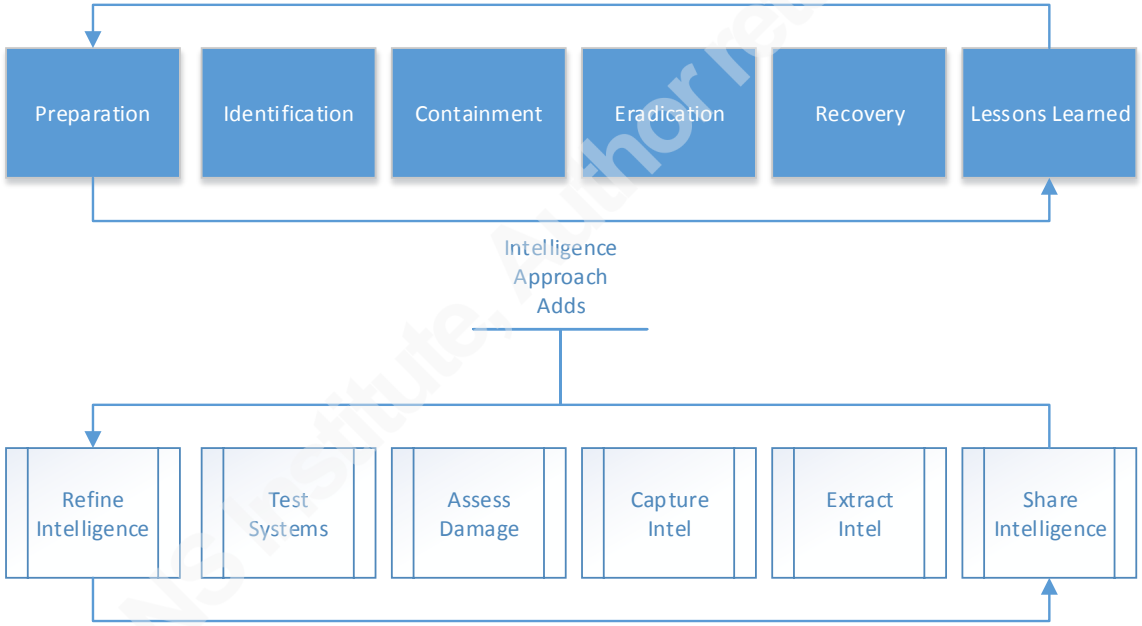
Abstract

Microsoft SQL Server provides several capabilities that control or affect the security of the data stored within its files. This document discusses layers of security that help protect DBMSs. The security and response to an incident depends on how well each of these items is configured and monitored. This document presents the security options for DBMSs specifically Microsoft SQL Server.

1.0 Introduction

Incident Response in a Microsoft SQL Server environment starts with planning and requires the Intelligence approach. This approach can be applied to most relational and NoSQL database management systems. The focus here is Microsoft SQL Server.

Continual monitoring is a very important part of protecting a Microsoft SQL Server environment. If monitoring uncovers intelligence of an incident, database administrators must investigate them thoroughly and with rigor. As attacks continue to become more sophisticated incident response must grow from ad hoc to incident discovery "hunting". (Chuvakin 2013)



Before hunting define incidents by establishing a baseline for each instance, creating policy, and establishing guidelines. Also establish a risk framework to prioritize response to incidents. In some cases people outside of information security and database administration will be the originator of incidents. Strong relationships with these groups help with forensics and investigations. By clearly defining response actions and who is responsible, organizations will be able to determine a course of action regarding legal requirements and system availability. The sharing of information with safe groups mature and strengthen the whole community, and very important. (Chuvakin 2013)

In many cases incident response is ad-hoc. Since attacks are becoming more continuous incidence response requires moving seamlessly from fire fighting to continual then continuous incident response. Continuous response layers an advanced mix of people, processes, tools, and feedback over current threats and assets. Indicators are examined under several contexts, threat intelligence, vulnerability, and user data. This can be captured in Microsoft SQL Server with layers of defense. (Chuvakin 2013)

2.0 What Is an Incident?

The goal of incident response (IR) and the intelligence approach is early notification. As outsiders become trusted insiders, attacks can originate from internal and external sources. For most an incident can be defined as a deviation from the norm of a system or process. Finding the deviations requires expertise in the systems or processes protected.

For example, if the security information and event management (SIEM) captures intelligence and detects a deviation from the norm in login failures on a Microsoft SQL Server within a given time frame further investigation is triggered. This could be a symptom of a database server on the verge of compromise. It isn't officially an "incident" yet conducting the investigation is triggering it. A tool or process and a deviation from the norm approach are a popular one for defining incidents. This requires extracting intelligence from individual server logs and refining them to reveal a deviation from known server logins. (Gartner 2011)

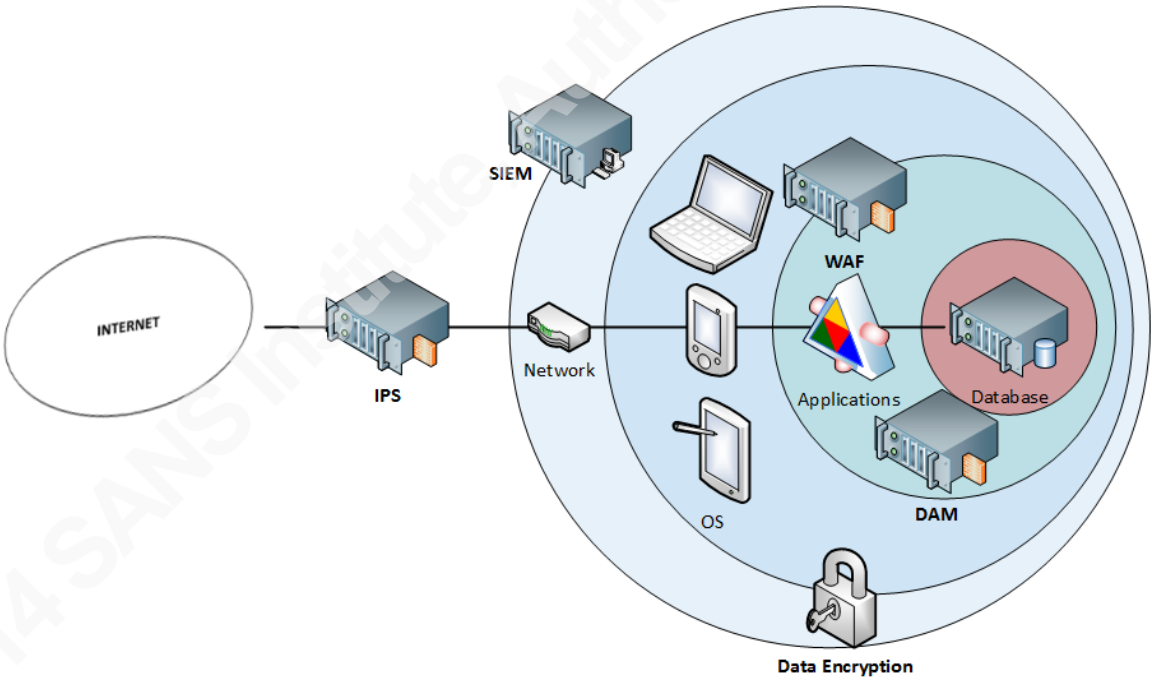
This is still very tactical. A better approach is a deviation from the norm plus a risk-analysis approach that accounts for the damage an event can cause. The IR Team evaluates which incidents have the greatest risk to a database server and watch for indicators and warnings of when these might occur. Events with the most impact should be quickly and thoroughly investigated. Organizations might use tools like database activity monitors to accomplish this. (Gartner 2011)

Most organizations operational database teams are the first responders for security activity, because they are the first to observe deviations in availability, network, user activity, or other events that should be marked for additional follow-up.

Incident response is more than a security and forensics activity. Operational staff should be trained how to handle and document incidents with low consequences. Low-risk malware infection must be remediated, but need not escalate to the security or incident response teams. The operations teams might be asked to conduct preliminary analysis before escalating to the security team. (Gartner 2011)

Policies, incident response plans, and incident investigations origin and responsibility are usually the security teams. Higher-level tools such as SIEM, IPS, database activity monitors, and web application firewalls are used to understand the root cause, or discover an incident. During an incident it is important to know if it is acceptable to remove a Microsoft SQL Server from the network or block a suspected incoming attack. (Gartner 2011)

3.0 The Framework



Defense in depth is the use of multiple security countermeasures to protect the enterprise. In a Microsoft SQL Server environment data is at the innermost point of a diagram. Today enterprise security architectures are designed with easy access from the network thru the operating system to applications then where the data is stored. Multiple

controls can provide more security than one. Protecting Microsoft SQL Server requires implementing steps from policy to continual incident discovery to continuous incident response.

Detecting deviations from the norm requires knowing what's not right in a Microsoft SQL Server environment. All bad actors leave tracks. Continuous incident response is an individual or team who uses advance preparation to identify when to declare an incident, its priority, and the knowledge of what information to share inside and outside the enterprise. The following will help with creating a baseline. (Gartner 2011)

Incidents are happening now. Mastering ad-hoc response to deviations from normal operation, and implementing manual log monitoring of high value equipment and processes achieve incident discovery. Then, adding tactical monitoring tools and fine-tuning incident response processes to gain more visibility into your Microsoft SQL Server environment. (Gartner 2011)

Incident discovery or identification is used in a Microsoft SQL Server database environment due to constant infection rates, ongoing attack campaigns, persistent threats, and compliance. Examples of this are ongoing scans for intelligence from external and internal sources, and includes malware and malicious software reversing. These processes help create a baseline and actionable intelligence. The intelligence mindset explores a direction instead of an entity, and creates insight through research. (Gartner 2011)

With the rise of Internet attacks that target web applications using sql injection, cross-site scripting, cross-site request forgery, and botnets bad actors have a powerful arsenal. SIEM, next generation firewalls, web application firewalls, and database activity monitors help with continual incident response and to mitigate threats. While protecting Microsoft SQL Server, the data and server should be protected from internal and external threats this includes monitoring privileged users, secure web development, and other risks to data. (Gartner 2011)

Insight can be found in endpoint forensic tools unlike antivirus are tools that can check files, processes, and memory on live running systems. Examples are products by Mandiant, RSA, Guidance, Carbon Black, and Helix. Endpoint sensors determine the

hash for a process. These sensors help determine if a process is running and, if files like pdf, docs, and images execute this process anywhere. (Gartner 2011)

Using SIEM and network forensics helps uncover small incidents. These tools used for IR are a collection of logs that can be quickly and easily searched. The caveat is incident detection isn't automatic. One of the most useful is a packet capture or network forensic tools (NFT). Packets are a single truth. Network forensics tools help collect packets to extract and analyze data. This gives the ability to dig deep into application protocols. The challenge with SIEM and NFT is seeing everything, dealing with the flood of information, virtual environments, and private clouds. (Gartner 2011)

Continuous monitoring is an important component of database protection. When an incident occurs DBAs must thoroughly investigate and contain them. There are many reasons to monitor database activity. For example looking for internal or external malicious or unauthorized activity. As the attacks become more sophisticated, continuous incident response is key to discovering database anomalies and fraud. To prevent sensitive data from leaving the organization inspect outbound traffic and block sensitive data like credit cards, social security numbers, and intellectual property. Also, protect application data stored in databases to protect the application from database exploits.

Sensitive information can be spread across different RDMSs and systems throughout the world, and visibility into these locations sensitive data helps enterprises reduce risk. Discovering where databases are located, types of information, and who has access to sensitive data is a critical step to securing them.

Sensitive data can be classified as structured or unstructured. Structured data is data that is organized in a structure so that it is identifiable. A Microsoft SQL Server database is an example of structured data storage. Microsoft SQL Server using T-SQL (Transact Structured Query Language) allows the selection of information based on columns and rows. Transact-SQL (T-SQL) is Microsoft's extension to Structured Query Language (SQL). SQL is a computer language originally developed for querying, altering and defining relational databases. T-SQL expands on the SQL standard. Sending Transact-SQL commands to the server does all communication with a SQL Server.

This is structured data, because it is organized and searchable by data type within the content. This content might contain credit cards or social security numbers. Unstructured data has a loose structure and includes documents, images, objects and other data types that are not part of a Microsoft SQL Server Database. An email and word documents are examples of unstructured data.

Vulnerability assessments helps identify known vulnerabilities in applications and supporting Microsoft SQL Server databases. Detect and patch vulnerabilities to analyze risk and exposure to potential threats. Unpatched systems expose Microsoft SQL Server databases and servers to a variety of attacks and more. In order to protect these systems and its sensitive data a vulnerability management solution is needed. These solutions scan applications and data systems to help close security gaps by scanning systems for known vulnerabilities and identifying systems that aren't configured correctly. Database servers should be scanned for sensitive data, and to identify unmanaged and unknown database servers and repositories.

To protect a Microsoft SQL Server and its data from threats we must block known and zero-day attacks by using a combination of known attack signatures, and deviations of normal application and database usage. Also, stop malicious users before they can attack by identifying and blocking known malicious sources.

Malicious insiders require identifying deviations from the norm in user activity then alert and/or block the suspicious activity. Monitoring sensitive data usage includes privileged and non-privileged users to the file system and databases. When a user's session is blocked or causes an alert it could signal abuse or a deviation from normal activity. These bad actors use privilege escalation, as a common technique the result is unexpected and unauthorized privileges changes.

Data is commonly accessed through an application interface, browsers, and database administrator tools. This data can be compromised through known and zero day attacks, malicious users both internal and external, leakage of sensitive data, exploits, and application and database vulnerabilities with web application vulnerabilities accounting for a large number of these attacks. One of the most common web application vulnerabilities is an injection attack.

Juan M. Walker CISSP, jwalker@emfbroadcasting.com

SQL Injection seeks to exploit vulnerabilities where untrusted data is sent to a Web application. In Microsoft SQL Server these flaws are often found in T-SQL queries sent to a web application. The attacker's objective is to trick the web application into executing commands like *'or 1=1 --'or* to gain access to unauthorized data.

Incorrectly configured security happens at all levels from the web to Microsoft SQL Server these flaws can give attackers access to sensitive data. Encryption helps protect sensitive data such as credit cards, social security numbers, and health care information.

Implementing secure coding best practices will secure the web development environment. This is an iterative approach that includes application design, source control, implementation, vulnerability testing, and monitoring. OWASP secure coding principles states that application design should contain the necessary controls to prevent unauthorized activity and include confidentiality, integrity, and availability. Once written, they should be tested with rigor for vulnerabilities using web application scanning, penetration testing, and code review.

Monitoring web applications for attacks reveal areas of the web site that hackers target, illustrates attack trends, and uncovers exploit techniques in real time. Database administrators and application developers can leverage this knowledge to architect more resilient web applications. To protect Microsoft SQL Server from these types of vulnerabilities a web application firewall in combination with a database activity monitor can be used. These devices have the ability to block or alert on attacks.

4.0 Microsoft SQL Server Database Security

Databases are an important asset for storing and providing data. Database administrators ensure the data stored in them is secure, available, and have integrity. Performing continual vulnerability and patch management help protect against known and unknown attacks. Monitoring privileged user access helps protect the Microsoft SQL Server environment from sensitive data leakage. Also, ensuring extra features are not installed; there are no system defaults, and unknown connections protect Microsoft SQL Servers from being compromised. Where data is sensitive it should be encrypted using

products that encrypt at the disk or database level. Microsoft SQL Server has this functionality built in. The first step is finding what to protect.

Refine Intelligence by knowing what to protect

A security label is information that describes the sensitivity of a data. This data is tagged with sensitivity levels mapped to user permissions. To determine access to data, the users' label is compared against the label on the data. The labeling of stored data provides information on how to handle, protect, and identify misuse. Without this information a user can accidently disclose sensitive data to unauthorized users.

Protect production and development Microsoft SQL Server environments

Least privilege is applied to ensure processes run at the level necessary to accomplish a specific task. The creation of processes, roles, and accounts with the right permissions are used to accomplish this. Least privilege is applied to design thru development and implementation. Developers are often granted elevated privileges within the Microsoft SQL Server or at the operating system level during development. These elevated privileges in production can affect operation and security. Elevated privileges in the production environment should not be allowed. The appendix has sample T-SQL code to insure the proper permissions are assigned to appropriate roles.

Identify and remove developer accounts from the production environment, because developer accounts should not change or alter database objects or data in the production environment. If there is a finding, this is an indicator. Eradicate by using the DROP LOGIN command in Microsoft SQL Server.

Microsoft SQL Server has other permissions that should be granted to the appropriate roles. For example, alter server state permissions are high server privileges that should only be granted to individual administration accounts through roles to insure they don't go unmanaged. Assignment of privileges via roles helps protect against privilege assignments that are unauthorized.

Identify users with alter server state permissions. If the user is unauthorized to have permissions, this is an indicator and intelligence. Assess the damage on all database servers in the environment. Capture additional intelligence from these systems, and

eradicate by denying privileges to users not authorized. This permission lets a user stop, start, or pause Microsoft SQL Server. Alter server state is granted by default on sysadmin.

Another high server privilege is alter any. The alter any event session permission allows a user to start and stop an event session such as Microsoft SQL Server Trace and System Monitor as well as the Windows event log or the SQL Server error log. Logs are critical to forensic investigations on a Microsoft SQL Server.

Apply least privilege to server roles like view any, control, shutdown, and administer. These permissions will allow a user to see all metadata, gain high-level access, and shutdown an instance of SQL Server. If the user is not authorized to have the permission, this is intelligence. If an indicator is found on a single instance of SQL Server, assess the damage on all database servers in the environment. Capture additional intelligence from these systems, and eradicate by revoking privileges to users not authorized. The next steps are to limit access to system tables, configuration information, consider renaming the SA account, and sharing intelligence with need to know personnel.

Refine Intelligence by monitoring unauthorized changes

Changes to the SQL Server environment can affect the overall security. Change control is important for the operating system, hardware, software, and firmware. Only individuals authorized and qualified should be allowed to gain access, change, upgrade, and modify the SQL Server environment. Unplanned and unauthorized changes can lead to system compromise. Implement a process to automatically check and alert on all changes to the system and database objects.

Extract and capture intelligence to insure database objects are owned by authorized accounts.

In SQL Server ownership of databases is a high level privilege. This owner has full rights to everything in the database and can grant privileges to other users. The owner is also a SQL Server user. Unmanaged ownership can cause unauthorized changes to database objects, and unauthorized privileges. To identify these accounts capture intelligence from the sys.databases table. Extract the name, owner_sid, and state_desc

from this table. If the user is not authorized owner, this is an indicator. Eradicate by changing the ownership to an authorized user. See appendix for code examples.

"Public" accounts might be assigned privileges that would allow unauthorized access to the SQL Server environment. These accounts should be removed or set to zero privileges, if there is no data being made available to the public. To obtain a list of public accounts capture intelligence from the sys.sysusers and sys.server_principals tables. If public accounts are found we should determine whether they have excessive privileges, if so this is an indicator. If there are findings, remove references and excessive privileges. See appendix for code examples.

"Guest" accounts might be assigned privileges that give an attacker access to the SQL Server environment. Well-known accounts like guest are likely targets and prone to give unauthorized access to the data within SQL Server. The guest account should have its privileges set to the minimum. Capture intelligence from the sys.sysusers and sys.server_principals tables. See appendix for code examples.

Refine Intelligence by removing unnecessary database objects and functionality

Unnecessary components in the SQL Server environment increase the attack vector and possible vulnerabilities. Disabling unnecessary services, components, and database objects reduces the attack surface. Document all approved database components and remove unused functionality and database objects.

Use encryption to protect sensitive information

As part of a layered defense strategy sensitive data should be encrypted at rest. When not encrypted this data is subject to compromise. Sensitive information stored in databases, tables, and columns should be encrypted using the highest level of encryption.

When a compensating control does not provide confidentiality encryption should be used for sensitive data protection. Encryption can be applied at the disk or single row and column level. Encrypted data should be made available to need-to-know users only. Safe-net and IBM can be used to encrypt in a SQL Server environment.

When using cryptography the use of weak algorithms puts data at risk. To insure confidentiality and non-repudiation only use the highest levels encryption. AES 256 is currently the strongest method. AES 128, AES 192, AES 256 are acceptable algorithms. The encryption example uses Microsoft SQL Server to store and mange keys. The keys below are Symmetric and offer the best protect when encrypting column level data.

Encryption Example

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '!DB123456789'
USE <<Database Name>>;
CREATE CERTIFICATE CertEmail_001
  WITH SUBJECT = 'Email Records',
  EXPIRY_DATE = '12/31/2016';
USE <<Database Name>>;
CREATE SYMMETRIC KEY EmailAES256_001 WITH ALGORITHM =
AES_256
   ENCRYPTION BY CERTIFICATE CertEmail_001;
USE <<Database Name>>;
GO
CREATE TABLE [dbo].[etblEmailAddress](
        [id] [bigint] IDENTITY(1,1) NOT NULL,
        [emailaddress] [char](45) NULL
) ON [PRIMARY]
GO
INSERT INTO [etblEmailAddress]([emailaddress]) VALUES
('xxx12345@gmail.com')
GO
INSERT INTO [etblEmailAddress]([emailaddress]) VALUES
('xxx12345@yahoo.com')
GO
INSERT INTO [etblEmailAddress]([emailaddress]) VALUES
('xxx12345@hotmail.com')
GO
INSERT INTO [etblEmailAddress]([emailaddress]) VALUES
('xxx12345@aol.com')
GO
INSERT INTO [etblEmailAddress]([emailaddress]) VALUES
('xxx12345@yourcompany.com')
GO

/* -- Test Encryption By Running the below. */

OPEN SYMMETRIC KEY EmailAES256_001
DECRYPTION BY CERTIFICATE CertEmail_001;

 SELECT
```

```
 Emailaddress,
 EncryptByKey(Key_GUID('EmailAES256_001'), emailaddress) as cyphertext,
 CONVERT(char(45),
DecryptByKey(EncryptByKey(Key_GUID('EmailAES256_001'), emailaddress)))
as decrypted_cyphertext
 FROM [dbo].[etblEmailAddress]
```

The above example can be expanded to support systems like PeopleSoft where encrypting data is not supported. An encrypted credit card example follows that is used in a PeopleSoft database.

Base tables and views:

```
CREATE TABLE [FZ103HSH_NEW](
        [HSZ103ID] [bigint] IDENTITY(1000,1) NOT NULL,
        [HSZ103HSH] [char](32) NULL,
        [HSZ103CCE] [char](30) NULL,
        [HSZ103LSTUSED] [datetime] NULL,
        [HSZ103ACNO] [float] NULL,
        [HSZ103EXPR] [char](10) NULL,
        [HSZ103CCE_NEW] [varbinary](32) NULL,
        [HSZ103NOC] [varchar](70) NULL,
        [HSZ103CID] [varchar](10) NULL,
 CONSTRAINT [PK_FZ103HSH_1] PRIMARY KEY CLUSTERED
(
        [HSZ103ID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

CREATE VIEW [FZ103HSH_IDV] AS
SELECT
[HSZ103ID],
[HSZ103HSH],
CAST ([encryptCreditCard([HSZ103CCE_NEW]) AS CHAR(30)) COLLATE
SQL_LATIN1_GENERAL_CP1_CI_AS [HSZ103CCE],
[HSZ103LSTUSED],
[HSZ103ACNO],
[HSZ103EXPR],
[HSZ103NOC],
[HSZ103CID]
FROM [FZ103HSH_NEW]
```

```
CREATE VIEW [FZ103HSH]
AS
SELECT
CASE [HSZ103ID] WHEN NULL THEN NULL ELSE [HSZ103ID] END
[HSZ103ID],
[HSZ103HSH] ,
[HSZ103CCE] ,
[HSZ103LSTUSED] ,
[HSZ103ACNO] ,
[HSZ103EXPR] ,
[HSZ103NOC] ,
[HSZ103CID]
FROM [FZ103HSH_IDV]
GO

CREATE TRIGGER [FZ103HSH_INS_TRIG]
ON [FZ103HSH]
INSTEAD OF INSERT
AS

BEGIN

SELECT  inserted.* INTO #temp FROM inserted

INSERT INTO[FZ103HSH_NEW]
(
[HSZ103HSH],
[HSZ103CCE],
[HSZ103CCE_NEW],
[HSZ103LSTUSED],
[HSZ103ACNO],
[HSZ103EXPR],
[HSZ103NOC],
[HSZ103CID]
)

SELECT
[fn_md5_2005](rtrim(#temp.[HSZ103CCE])),
NULL,
encryptCreditCard(#temp.[HSZ103CCE]),
#temp.[HSZ103LSTUSED],
#temp.[HSZ103ACNO],
#temp.[HSZ103EXPR],
#temp.[HSZ103NOC],
#temp.[HSZ103CID]
```

```
FROM #temp
                WHERE
                NOT EXISTS (SELECT * FROM FZ103HSH WHERE
HSZ103HSH = [fn_md5_2005](rtrim(#temp.[HSZ103CCE])))
                AND NOT EXISTS (SELECT * FROM FZ103HSH WHERE
HSZ103ID = #temp.HSZ103CCE)
                AND len(#temp.[HSZ103CCE]) <> 0


END

ALTER TRIGGER [FZ103HSH_UPD_TRIG]
ON [FZ103HSH_IDV]
INSTEAD OF UPDATE
AS
BEGIN  UPDATE [FZ103HSH_NEW]
SET [HSZ103HSH] = [INSERTED].[HSZ103HSH],
[HSZ103CCE_NEW] = case CAST( [INSERTED].[HSZ103CCE] AS
VARBINARY(8000) ) when CAST( [DELETED].[HSZ103CCE] AS
VARBINARY(8000) ) then [HSZ103CCE_NEW] else encryptCreditCard
([INSERTED].[HSZ103CCE) end,
[HSZ103LSTUSED] = [INSERTED].[HSZ103LSTUSED],
[HSZ103ACNO] = [INSERTED].[HSZ103ACNO],
[HSZ103EXPR] = [INSERTED].[HSZ103EXPR],
[HSZ103NOC] = [INSERTED].[HSZ103NOC],
[HSZ103CID] = [INSERTED].[HSZ103CID]
FROM [FZ103HSH_NEW],
INSERTED,
DELETED
WHERE [FZ103HSH_NEW].[HSZ103ID] = [INSERTED].[HSZ103ID] and
[FZ103HSH_NEW].[HSZ103ID] = [DELETED].[HSZ103ID]
OPTION(ROBUST PLAN)
END
```

Place the code below in a job to continually update credit cards from your credit card vault.

```
CREATE PROCEDURE [dbo].[prcPay01CreditCards_e]
AS
SET NOCOUNT ON
BEGIN

INSERT INTO FZ103HSH(HSZ103CCE,HSZ103EXPR)
SELECT
CreditCardNumber AS HSZ103CCE ,
```

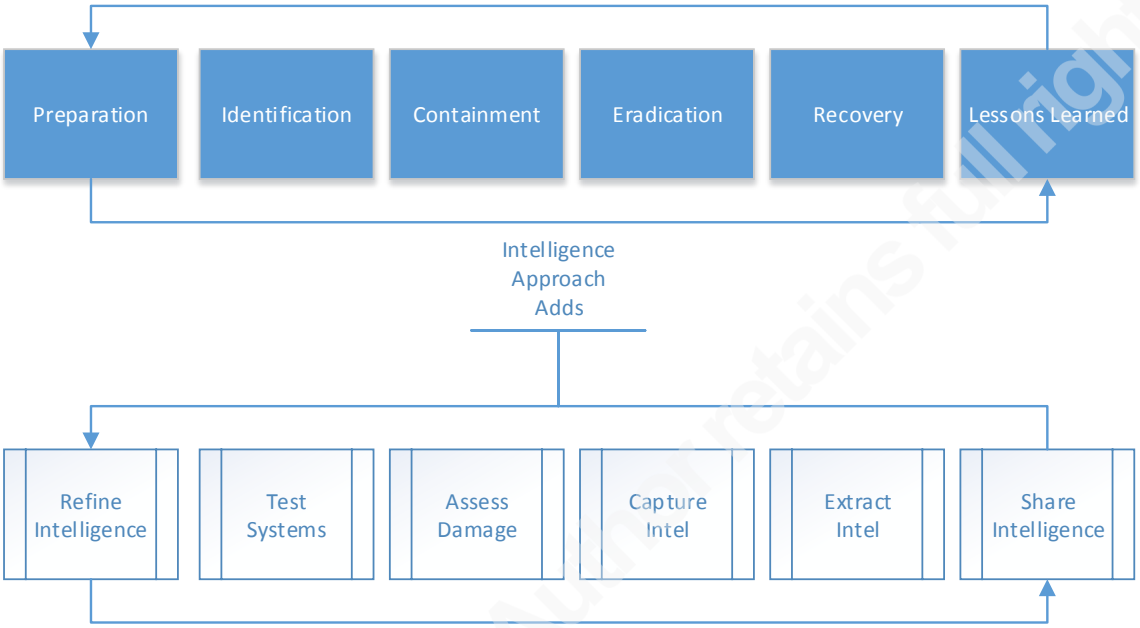
```
ExpirationMonth+ExpirationYear AS HSZ103EXPR
FROM   [Pay01_CreditCards] pay01 WITH (NOLOCK)
WHERE
NOT EXISTS
(
        SELECT *
        FROM FZ103HSH_NEW  WITH (NOLOCK)
        WHERE [HSZ103CCE_NEW] = encryptCreditCard
(CONVERT(CHAR(30),CreditCardNumber))
)
AND NOT EXISTS
(
        SELECT *
        FROM FZ103HSH_NEW  WITH (NOLOCK)
        WHERE HSZ103ID = CreditCardNumber
)
AND LEN(rtrim(CreditCardNumber)) <> 0
AND LEN(rtrim(CreditCardNumber)) BETWEEN 13 AND 16
AND ISNUMERIC(rtrim(CreditCardNumber)) = 1

UPDATE [Pay01_CreditCards]
        SET CreditCardNumber = isnull(dbo.fn_getref
(rtrim(CreditCardNumber)),CreditCardNumber )
        FROM [dbo].[Pay01_CreditCards] p
        WHERE LEN(rtrim(CreditCardNumber)) BETWEEN 13 AND 16

END
SET NOCOUNT OFF
```

It is common that data used for development is transferred from a production database server. Procedures should be in place to ensure sensitive production data is properly handled.

Capture Intelligence by auditing the SQL Server Environment to help with Incident Identification and Response

Audit records are generated by many systems within the enterprise. The operating system, network, and Server SQL Server environment are some examples. Audit logs help with forensic investigations regarding malicious activity. Capture intelligence from ::FN_TRACE_GETINFO('0'). If logging is not started, this is an indicator. See appendix for code examples to check for event auditing and to start basic logging.

Incident Example



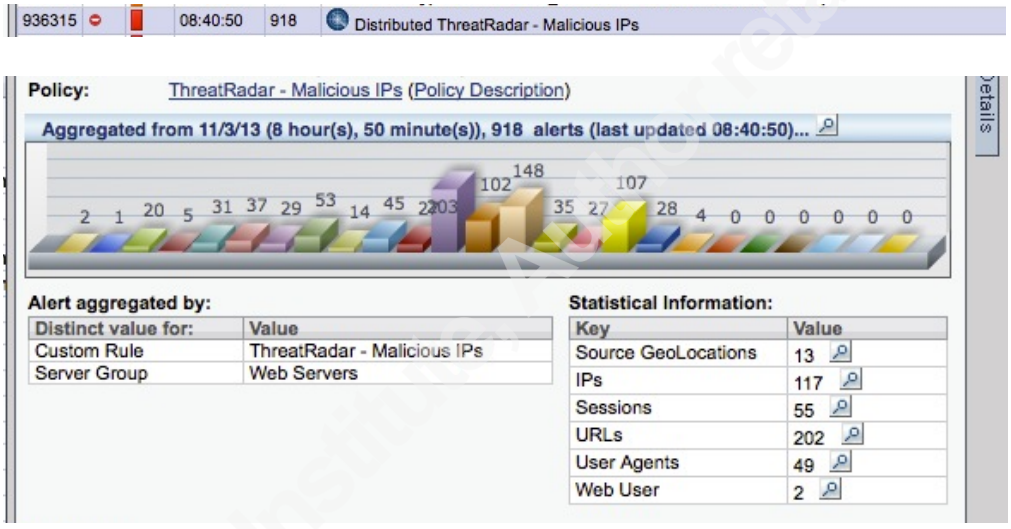
In this example unencrypted data has been detected, and this unencrypted data is available via Microsoft SQL Server public account. The goal is to quickly identify public accounts with excessive privileges and access to the encrypted data. Using the traditional incident response approach, first assess the damage on the current server by capturing intelligence from sys.sysusers and sys.server_principals tables. Extract the name column these tables. See appendix for a code example

Identify each public account and determine if it has excessive privileges. If there are findings, contain and eradicate by removing references to the public account and excessive privileges. After removal, test applications that use the database and go through lessons learned with need to know staff.

The intelligence approach expands the identification and moves into discovery by hunting for actionable intelligence. Capturing and extracting intelligence from sys.sysusers and sys.server_principals tables across all databases and servers in the environment to assess the damage can achieve this. Continue the assessment by determining if the public account has excessive privileges. If there are findings, remove

references to the public account and excessive privileges. Then test applications that use the database and share the intelligence with need to know staff.

Protection at the network helps restrict access to SQL Server and users with the appropriate permissions. When a direct connection to the SQL Server is used instead of a middle tier, a wide range of users must be allowed. The identification of malicious IPs by location can help protect a SQL Server. Tools like iMPERVA have built in profiles for malicious IPs. After extracting intelligence, for the SQL Server and network, the below example should result in an alert, block, or both, and logged as an indicator in the continuous incident response process.



If a malicious IP above is targeting the SQL Server that contains sensitive information, the result is an alert, block, or both, and logged as an indicator in the continuous incident response process.

5.0 Conclusion

In summary, it is important to assess the damage and define when and why an incident is occurring and when a formalized response is required. The declaration of an incident triggers resources and processes that limit harm and determine the parameters for escalation. Many incidents originate from observations by business personnel or simply through calls to the help desk. This behavior should be encouraged both for the reporting and handling of incidents. Reporting to management should be prioritized and streamlined being sure to include threats, lessons learned, and steps to remediation.

Juan M. Walker CISSP, jwalker@emfbroadcasting.com

The purpose of these products and procedures is to secure SQL Server's sensitive data. This includes auditing, alerting, and blocking in real time. These products and services give the ability to detect and patch SQL Server vulnerabilities. They also help identify malicious users, users with excess permissions, and ones that should be removed because they are inactive. Finally, they help accelerate incident response and forensic investigations with an intelligence point of view.

6.0 References

Naidu, Stree. Security - Data: Data Suffers from Skewed Security Priorities [online]. Government News, Vol. 30, No. 4, Aug/Sept 2010: 58-59. Availability: <http://search.informit.com.au/documentSummary;dn=274800041729954;res=IELHSS> ISSN: 1447-0500. [cited 10 Mar 14].

Microsoft TechNet (2014). Microsoft SQL Server. Dynamic Management Views and Functions (Transact-SQL). Retrieved from http://technet.microsoft.com/en-us/library/ms180163.aspx

Microsoft TechNet (2014). Microsoft SQL Server. Database and Files Catalog Views. Retrieved from http://technet.microsoft.com/en-us/library/ms180163.aspx

Defense Information System agency (2014, January 28). Application Security – Database (SQL Server). Retrieved from http://iase.disa.mil/stigs/app_security/database/sql.html

Joint Task Force Transformation Initiative (2013). Security and Privacy Controls for Federal Information Systems and Organizations (NIST Special Publication 800-53r4, 2013 Edition). Retrieved November 1, 2013, from National Institute of Standards and Technology Website: http://dx.doi.org/10.6028/NIST.SP.800-53r4

Cichonski, P., Millar, T., Grance, T., & Scarfone, K. (2012). Computer Security Incident Handling Guide (NIST Special Publication 800-61r2, 2012 Edition). Retrieved November 1, 2013, from National Institute of Standards and Technology Website: http://csrc.nist.gov/publications/nistpubs/800-61rev2/SP800-61rev2.pdf

Souppaya, M., & Scarfone, K. (2013). Guide to Malware Incident Prevention and Handling for Desktops and Laptops (NIST Special Publication 800-83r1, 2013 Edition). Retrieved November 1, 2013, from National Institute of Standards and Technology Website: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-83r1.pdf

Kent, K., Chevalier, S., Grance, T., & Hung, D. (2006). Guide to Integrating Forensic Techniques into Incident Response (NIST Special Publication 800-86, 2006 Edition). Retrieved November 1, 2013, from National Institute of Standards and Technology Website: http://csrc.nist.gov/publications/nistpubs/800-86/SP800-86.pdf

Henry, T. (2012, August 21). Acting on Security Monitoring: Incident Response and Forensics (ID: G00210015). Retrieved from Gartner database.

McMillan, R., & Walls, A. (2013, July 02). Seven Steps to Creating an Effective Computer Security Incident Response Team (ID: G00225512). Retrieved from Gartner database.

McMillan, R. (2013, November 19). Prepare for the Inevitable With an Effective Security Incident Response Plan (ID: G00236455). Retrieved from Gartner database.

Chuvakin, A. (2013). Top 7 Most Effective Incident Detection and Response Practices and Tools. Symposium conducted at the meeting of Gartner for Technical Professionals, San Diego, CA.

7.0 Appendix

The following T-SQL code will help test systems to identify and then remove Developer Accounts:

Identification:

```
SELECT
        name AS 'AccountName'
       ,create_date AS 'CreateDate'
       ,LOGINPROPERTY(name, 'PasswordLastSetTime') AS
'PasswordChanged'
  FROM sys.server_principals
 WHERE TYPE NOT IN ('C', 'R', 'U') -- ('C', 'G', 'K', 'R', 'S', 'U')
  AND name NOT IN ('##MS_PolicyEventProcessingLogin##',
'##MS_PolicyTsqlExecutionLogin##')
  AND sid <> CONVERT(VARBINARY(85), 0x01) -- exclude SA account
  AND is_disabled <> 1
 ORDER BY name
```

Eradication:

```
USE master
DROP LOGIN <<'account name'>>
```

(Defense Information System agency (STIG), 2014)

Apply least privilege to 'Alter…' permissions.

Use the T-SQL code below to find server roles that are authorized to 'Alter…' permissions of a Microsoft SQL Server.

Indentification:

```
SELECT
        pr.name
       ,pe.permission_name
```

```
        ,pe.state_desc
        ,pr.[type]
FROM sys.server_permissions  AS pe
JOIN sys.server_principals       AS pr
ON pe.grantee_principal_id = pr.principal_id
WHERE pr.[type] = 'R' AND permission_name =  'Alter server state'
```

Deny privileges to roles not authorized.

Eradication:

REVOKE <<permission name>> TO <<role name>>

Apply least privilege to 'Alter Any' permissions.

Use the T-SQL code below to find server roles that are authorized to 'Alter Any' of a Microsoft SQL Server.

```
Indentification:
SELECT
        pr.name
        ,pe.permission_name
        ,pe.state_desc
        ,pr.[type]
FROM sys.server_permissions AS pe
JOIN sys.server_principals      AS pr
ON pe.grantee_principal_id = pr.principal_id
WHERE pr.[type] = 'R' AND permission_name  LIKE 'Alter any%'
```

Deny privileges to roles not authorized.

Eradication:

REVOKE <<permission name>> TO <<role name>>

Apply least privilege to 'View…' permissions.

Use the T-SQL code below to find server roles that are authorized to 'View…' permissions of a Microsoft SQL Server.

```
Indentification:
SELECT
        pr.name
        ,pe.permission_name
        ,pe.state_desc
        ,pr.[type]
FROM sys.server_permissions AS pe
```

```
JOIN sys.server_principals      AS pr
ON pe.grantee_principal_id = pr.principal_id
WHERE pr.[type] = 'R' AND permission_name LIKE 'VIEW%'
```

Deny privileges to roles not authorized.

Eradication:

REVOKE <<permission name>> TO <<role name>>

Apply least privilege to 'Create…' permissions.

Use the T-SQL code below to find server roles that are authorized to 'CREATE…' permissions of a Microsoft SQL Server.

Indentification:

```
SELECT
        pr.name
       ,pe.permission_name
       ,pe.state_desc
       ,pr.[type]
FROM sys.server_permissions AS pe
JOIN sys.server_principals      AS pr
ON pe.grantee_principal_id = pr.principal_id
WHERE pr.[type] = 'R' AND permission_name LIKE 'CREATE%'
```

Deny privileges to roles not authorized.

Eradication:

REVOKE <<permission name>> TO <<role name>>

Apply least privilege to 'Authenticate…' permissions.

Use the T-SQL code below to find server roles that are authorized to '**Authenticate** …' permissions of a Microsoft SQL Server.

Indentification:

```
SELECT
        pr.name
       ,pe.permission_name
       ,pe.state_desc
       ,pr.[type]
FROM sys.server_permissions AS pe
JOIN sys.server_principals      AS pr
ON pe.grantee_principal_id = pr.principal_id
WHERE pr.[type] = 'R' AND permission_name LIKE 'Authenticate%'
```

Deny privileges to roles not authorized.

Eradication:

REVOKE <<permission name>> TO <<role name>>

Apply least privilege to 'Control…' permissions.

Use the T-SQL code below to find server roles that are authorized to '**Control** …' permissions of a Microsoft SQL Server.

Indentification:

```
SELECT
        pr.name
       ,pe.permission_name
       ,pe.state_desc
       ,pr.[type]
FROM sys.server_permissions AS pe
JOIN sys.server_principals      AS pr
ON pe.grantee_principal_id = pr.principal_id
WHERE pr.[type] = 'R' AND permission_name LIKE 'Control%'
```

Deny privileges to roles not authorized.

Eradication:

REVOKE <<permission name>> TO <<role name>>

Apply least privilege to 'UNSAFE…' permissions.

Use the T-SQL code below to find server roles that are authorized to '**UNSAFE**…' permissions of a Microsoft SQL Server.

Indentification:

```
SELECT
        pr.name
       ,pe.permission_name
       ,pe.state_desc
       ,pr.[type]
FROM sys.server_permissions AS pe
JOIN sys.server_principals      AS pr
ON pe.grantee_principal_id = pr.principal_id
WHERE pr.[type] = 'R' AND permission_name LIKE 'UNSAFE%'
```

Deny privileges to roles not authorized.

Eradication:

REVOKE <<permission name>> TO <<role name>>

Apply least privilege to 'SHUTDOWN…' permissions.

Use the T-SQL code below to find server roles that are authorized to '**SHUTDOWN**…' permissions of a Microsoft SQL Server.

```
Indentification:
SELECT
        pr.name
       ,pe.permission_name
       ,pe.state_desc
       ,pr.[type]
FROM sys.server_permissions AS pe
JOIN sys.server_principals      AS pr
ON pe.grantee_principal_id = pr.principal_id
WHERE pr.[type] = 'R' AND permission_name LIKE 'SHUTDOWN%'
```

Deny privileges to roles not authorized.

Eradication:

REVOKE <<permission name>> TO <<role name>>

Apply least privilege to 'External…' permissions.

Use the T-SQL code below to find server roles that are authorized to '**External**…' permissions of a Microsoft SQL Server.

```
Indentification:
SELECT
        pr.name
       ,pe.permission_name
       ,pe.state_desc
       ,pr.[type]
FROM sys.server_permissions AS pe
JOIN sys.server_principals      AS pr
ON pe.grantee_principal_id = pr.principal_id
WHERE pr.[type] = 'R' AND permission_name LIKE 'External%'
```

Deny privileges to roles not authorized.

Eradication:

REVOKE <<permission name>> TO <<role name>>

Apply least privilege to 'Connect…' permissions.

Use the T-SQL code below to find server roles that are authorized to '**Connect**…' permissions of a SQL Server.

```
Indentification:
SELECT
        pr.name
       ,pe.permission_name
       ,pe.state_desc
       ,pr.[type]
FROM sys.server_permissions AS pe
JOIN sys.server_principals      AS pr
ON pe.grantee_principal_id = pr.principal_id
WHERE pr.[type] = 'R' AND permission_name LIKE 'Connect%'
```

Deny privileges to roles not authorized.

Eradication:

REVOKE <<permission name>> TO <<role name>>

Apply least privilege to 'Administer…' permissions.

Use the T-SQL code below to find server roles that are authorized to '**Administer**…' permissions of a SQL Server.

```
Indentification:
SELECT
        pr.name
       ,pe.permission_name
       ,pe.state_desc
       ,pr.[type]
FROM sys.server_permissions AS pe
JOIN sys.server_principals      AS pr
ON pe.grantee_principal_id = pr.principal_id
WHERE pr.[type] = 'R' AND permission_name LIKE 'Administer%'
```

Deny privileges to roles not authorized.

Eradication:

REVOKE <<permission name>> TO <<role name>>

Extract and capture intelligence to insure database objects are owned by authorized accounts.

To identify these accounts run the T-SQL query that follows.

```
SELECT name AS 'DBName'
    , SUSER_SNAME(owner_sid) AS 'DBOwner'
    , state_desc AS 'DBStatus'
 FROM sys.databases
```

To remediate and change the ownership to an authorized database owner do the following:

1. Go to SQL Server Management Studio
2. Then Object Explorer
3. Select SQL Server Name
4. Click Databases
5. Right click Database Name
6. Click Properties
7. Click Files
8. Select new database Owner
9. Click ok to commit all changes

(Defense Information System agency (STIG), 2014)

Public account management

To obtain a list of public accounts run the following code.

```
EXEC sp_MSforeachdb '
IF NOT "?" IN ("master", "tempdb", "model", "msdb")
BEGIN
  USE ?
  SELECT "?" AS "Database"
      , su.name   AS "dbAccountName"
      , sp.name   AS "SQLServerAccountName"
    FROM sys.sysusers su
    LEFT JOIN sys.server_principals sp
     ON su.sid = sp.sid
   WHERE ( su.name like "publ%"
 OR sp.name like "publ%" )
     AND NOT su.sid = CONVERT(VARBINARY(85), 0x)
END'
```

Below are the steps to find excessive privileges of public accounts.

1. Go to SQL Server Management Studio
2. Then Object Explorer
3. Select SQL Server Name
4. Click databases
5. Select <<Database Name>>
6. Select security
7. Select users
8. Right click public account
9. Right click Database Name

10. Click properties
11. Check owned schemas
12. Then membership
13. Next securables
14. Then extended properties

(Defense Information System agency (STIG), 2014)

Public account removal:

```
USE <'databasename'>
DROP USER  <'publicaccountname'> -- Removes user from database
DROP LOGIN <'publicaccountname'> -- Removes user from system
GO
```

Public account scheme ownership removal:

```
-- Remove Owned Schemas by assigning schema to another user
USE <'databasename'>
ALTER AUTHORIZATION ON SCHEMA::<'schemaname'> TO
<'accountname'>
GO
```

Public account direct membership of a system role removal:

```
USE <'databasename'>
ALTER ROLE <'rolename'> DROP MEMBER <'publicaccountname'>
GO
```

Public account direct Securables access removal:

```
USE <'databasename'>
REVOKE <'securablename'> ON <'object_name'> TO <'publicaccountname'> AS
<'grantorname'>
GO --<'grantorname'> is usually "[dbo]"
```

Public account direct Extended Properties removal:

1. Go to SQL Server management studio
2. Then oject explorer
3. Select SQL Server name
4. Click databases
5. Select <<database name>>
6. Select security
7. Select users
8. Right click public account
9. Right click database name
10. Click properties
11. Select extended properties
12. Click “Delete” to remove the extended property

The code below will check for guest accounts.

```
EXEC sp_MSforeachdb '
IF NOT "?" IN ("master", "tempdb", "model", "msdb")
BEGIN
  USE ?
  SELECT "?" AS "Database"
      , su.name AS "dbAccountName"
      , sp.name AS "SQLServerAccount"
   FROM sys.sysusers su
   LEFT JOIN sys.server_principals sp
     ON su.sid = sp.sid
   WHERE ( su.name like "gues%"
       OR sp.name like "gues%" )
END  '  (Defense Information System agency (STIG), 2014)
```

PUBLIC Account Access

```
Select
    su.name   AS 'dbAccountName'
      , sp.name   AS 'SQLServerAccountName'
    FROM sys.sysusers su
    LEFT JOIN sys.server_principals sp
      ON su.sid = sp.sid
   WHERE ( su.name like 'publ%'
OR sp.name like 'publ%')

     AND NOT su.sid = CONVERT(VARBINARY(85), 0x)
```

CONVERT(VARBINARY(85), 0x) filters the guest account. The sid column in sys.sysusers is varbinary(85). You must convert 0x to varbinary which is the value of guest in sys.sysusers. (Defense Information System agency (STIG), 2014)

Check for event auditing:

```
SELECT DISTINCT(traceid) FROM ::FN_TRACE_GETINFO('0')
```

Basic logging is started with the code below.

```
CREATE PROCEDURE start_audit AS
-- Create a Queue
DECLARE @rc INT
DECLARE @traceID INT
DECLARE @maxfilesize BIGINT
DECLARE @audit_log NVARCHAR(128)

SET @maxfilesize = 5
SET @audit_log = <<'Log File Path'>>
```

```
EXEC @rc = SP_TRACE_CREATE @TraceID output, 6, @audit_log,
@maxfilesize, NULL

-- Audit Login
EXEC SP_TRACE_SETEVENT @TraceID, 14, 1, 1 -- TextData
EXEC SP_TRACE_SETEVENT @TraceID, 14, 6, 1 -- NTUserName
EXEC SP_TRACE_SETEVENT @TraceID, 14, 7, 1 -- NTDomainName
EXEC SP_TRACE_SETEVENT @TraceID, 14, 8, 1 -- HostName
EXEC SP_TRACE_SETEVENT @TraceID, 14, 10, 1 -- ApplicationName
EXEC SP_TRACE_SETEVENT @TraceID, 14, 11, 1 -- LoginName
EXEC SP_TRACE_SETEVENT @TraceID, 14, 12, 1 -- SPID
EXEC SP_TRACE_SETEVENT @TraceID, 14, 14, 1 -- StartTime
EXEC SP_TRACE_SETEVENT @TraceID, 14, 23, 1 -- Success
EXEC SP_TRACE_SETEVENT @TraceID, 14, 26, 1 -- ServerName
EXEC SP_TRACE_SETEVENT @TraceID, 14, 35, 1 -- DatabaseName
EXEC SP_TRACE_SETEVENT @TraceID, 14, 41, 1 -- LoginSid
EXEC SP_TRACE_SETEVENT @TraceID, 14, 60, 1 -- IsSystem
EXEC SP_TRACE_SETEVENT @TraceID, 14, 64, 1 -- SessionLoginName
-- Audit Logout
-- Occurs when a user logs out of SQL Server.
EXEC SP_TRACE_SETEVENT @TraceID, 15, 6, 1 -- NTUserName
EXEC SP_TRACE_SETEVENT @TraceID, 15, 7, 1 -- NTDomainName
EXEC SP_TRACE_SETEVENT @TraceID, 15, 8, 1 -- HostName
EXEC SP_TRACE_SETEVENT @TraceID, 15, 10, 1 -- ApplicationName
EXEC SP_TRACE_SETEVENT @TraceID, 15, 11, 1 -- LoginName
EXEC SP_TRACE_SETEVENT @TraceID, 15, 12, 1 -- SPID
EXEC SP_TRACE_SETEVENT @TraceID, 15, 13, 1 -- Duration
EXEC SP_TRACE_SETEVENT @TraceID, 15, 14, 1 -- StartTime
EXEC SP_TRACE_SETEVENT @TraceID, 15, 15, 1 -- EndTime
EXEC SP_TRACE_SETEVENT @TraceID, 15, 23, 1 -- Success
EXEC SP_TRACE_SETEVENT @TraceID, 15, 26, 1 -- ServerName
EXEC SP_TRACE_SETEVENT @TraceID, 15, 35, 1 -- DatabaseName
EXEC SP_TRACE_SETEVENT @TraceID, 15, 41, 1 -- LoginSid
EXEC SP_TRACE_SETEVENT @TraceID, 15, 60, 1 -- IsSystem
EXEC SP_TRACE_SETEVENT @TraceID, 15, 64, 1 -- SessionLoginName
-- Audit Server Starts and Stops
EXEC SP_TRACE_SETEVENT @TraceID, 18, 6, 1 -- NTUserName
EXEC SP_TRACE_SETEVENT @TraceID, 18, 7, 1 -- NTDomainName
EXEC SP_TRACE_SETEVENT @TraceID, 18, 8, 1 -- HostName
EXEC SP_TRACE_SETEVENT @TraceID, 18, 10, 1 -- ApplicationName
EXEC SP_TRACE_SETEVENT @TraceID, 18, 11, 1 -- LoginName
EXEC SP_TRACE_SETEVENT @TraceID, 18, 12, 1 -- SPID
EXEC SP_TRACE_SETEVENT @TraceID, 18, 14, 1 -- StartTime
EXEC SP_TRACE_SETEVENT @TraceID, 18, 23, 1 -- Success
EXEC SP_TRACE_SETEVENT @TraceID, 18, 26, 1 -- ServerName
EXEC SP_TRACE_SETEVENT @TraceID, 18, 41, 1 -- LoginSid
```

EXEC SP_TRACE_SETEVENT @TraceID, 18, 60, 1 -- *IsSystem*
EXEC SP_TRACE_SETEVENT @TraceID, 18, 64, 1 -- *SessionLoginName*
-- *Audit Login Failed*
EXEC SP_TRACE_SETEVENT @TraceID, 20, 1, 1 -- *TextData*
EXEC SP_TRACE_SETEVENT @TraceID, 20, 6, 1 -- *NTUserName*
EXEC SP_TRACE_SETEVENT @TraceID, 20, 7, 1 -- *NTDomainName*
EXEC SP_TRACE_SETEVENT @TraceID, 20, 8, 1 -- *HostName*
EXEC SP_TRACE_SETEVENT @TraceID, 20, 10, 1 -- *ApplicationName*
EXEC SP_TRACE_SETEVENT @TraceID, 20, 11, 1 -- *LoginName*
EXEC SP_TRACE_SETEVENT @TraceID, 20, 12, 1 -- *SPID*
EXEC SP_TRACE_SETEVENT @TraceID, 20, 14, 1 -- *StartTime*
EXEC SP_TRACE_SET
Verify the logs are started with the code below.
SELECT DISTINCT(eventid) **FROM** ::FN_TRACE_GETEVENTINFO('14')
SELECT DISTINCT(eventid) **FROM** ::FN_TRACE_GETEVENTINFO('15')
SELECT DISTINCT(eventid) **FROM** ::FN_TRACE_GETEVENTINFO('18')
SELECT DISTINCT(eventid) **FROM** ::FN_TRACE_GETEVENTINFO('20')

(Defense Information System agency, 2014)