

Secure Coding. Practical steps to defend your web apps.

Copyright SANS Institute
Author Retains Full Rights

This paper is from the SANS Software Security site. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Defending Web Applications Security Essentials (DEV522)"
at <http://software-security.sans.org><http://software-security.sans.org/events/>

Forensic Analysis of a SQL Server 2005 Database Server

GCFA Gold Certification

Author: Kevvie Fowler, GCFA, CISSP, MCTS, MCSD, MCDBA, MCSE

kevvie.fowler@emergis.com

Adviser: Joey Niem

Accepted: April 1, 2007

Outline

| | |
|--|----|
| Investigation Introduction | 3 |
| Step 1: Verification | 3 |
| Step 2: System Description | 10 |
| Step 3: Evidence Collection | 11 |
| Step 4: Timeline Creation | 16 |
| Step 5: Media Analysis | 18 |
| Step 6: Data Recovery | 38 |
| Step 7: String Search | 43 |
| Investigation Summary | 44 |
| Appendix A | 46 |
| Appendix B | 48 |
| References | 50 |

Investigation Introduction

On March 1st, 2007, I received a call from a client who stated that they may have been a victim of a security incident sometime over the past 24 hours. They believed unauthorized modifications were made to their production database server which had resulted in erroneous product shipments and financial loss to the company. Due to the mission critical nature of the system, it could not be taken off-line unless significant evidence of system misuse could be identified.

Step 1: Verification

Upon arriving on scene, I was briefed on the situation and learned that the SQL Server 2005 database server contained a single user database which was the foundation of an online-sales application. The client also informed me that they had received a call from a credit card company regarding a suspicious transaction that was charged to a client card by their company.

Because the server could not be taken off-line, a live analysis was performed. During a live analysis volatile and non volatile data is viewed and acquired with the assistance of the live target operating system¹. During a forensic investigation you should utilize binaries on the target system as little as possible as they may be corrupt or tampered with thus skewing their output.

The incident response CD-Rom used in this investigation contains traditional incident response tools in addition to SQL utilities and libraries which allow ad-hoc query submission to SQL Servers using minimal assistance from the un-trusted host.

To begin the incident verification, Windows Forensic Tool v1.0.03 will be used with a customized configuration file. This configuration file will execute Distributed Management Views (DMV), Database Consistency Checker (DBCC) commands and other vendor issued procedures to gather data which can be used to prove or disprove the occurrence of an intrusion. For more information on the customized Windows Forensic Tool Chest configuration file, refer to Appendix A of this document.

At precisely 10:02 AM, server time, the client's system logged into the PRODSQL05 SQL Server interactively under the user context Administrator. Upon logging into the system, it was observed that the system tray contained no third party application icons and the operating system appeared to be Windows 2003 Standard Edition. At 10:03 AM, server time, I assumed command of the console to begin the investigation. My Forensic Response CD was inserted into the computer and a trusted command shell was launched by issuing the "*D:\FResponse\cmd.exe*" command. Using the full file path in addition to the binary name ensures that the binary is loaded from the trusted CD. The un-trusted host may contain binaries with matching names to the binaries contained on my response CD. If these binaries are present within a directory referenced in the path variable of the target host, the un-trusted binaries can be loaded in error. To eliminate the possibility of this occurring, the full file location in addition to the binary name will be used during this investigation.

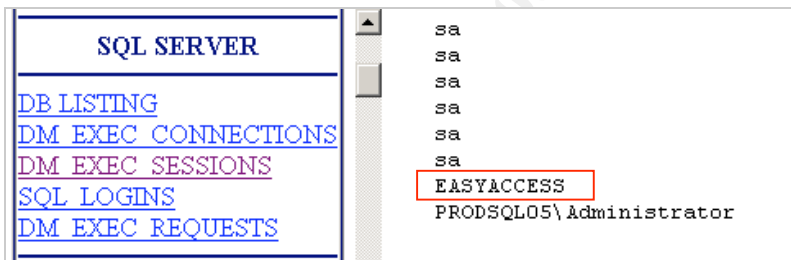
The outputs from the tools run during this investigation, will be saved on the trusted forensic workstation as opposed to the un-trusted target host. From the command shell, the "*D:\FResponse\net use *\\192.168.1.174\Acquisition*" command was issued to map a drive from the target host to sterile storage media located on my forensic laptop which was connected to the network under IP Address 192.168.1.174.

The "\$Acquisition" share is hidden and password protected to help ensure the integrity

of the data within. It was noted that the drive letter associated with the net use command was connected as “E:” on the target host. The “D:\FResponse\wft.exe -dst E:” command was issued to launch the customized Windows Forensic Toolchest v1.0.03 instance which gathered volatile database and operating system data from the target system and securely stored it on the forensic workstation.

Once Windows Forensic Toolchest was finished executing, the results were analyzed and the following notable events were identified.

SQL Server reserves Sessions #50 and lower for internal SQL Server processes, discounting these, it was identified that two sessions were currently active on the SQL Server. The first Session ID # 52 which belonged to the instance of WFT executing under the local Administrator context and the second was Session #51 belonging to an unknown user operating under the login EASYACCESS. This session had been established at 7:58 AM that morning. Because the login name was unconventional, it was flagged for client verification.



SQL Server 2005 maintains a record of the last SQL statement executed by a given session. Viewing this history for the connected users led to the identification of a suspicious transaction.

| SQL SERVER | local_tcp_port text |
|---|---|
| DB LISTING DM EXEC CONNECTIONS DM EXEC SESSIONS SQL LOGINS DM EXEC REQUESTS | <pre> 1433 delete from [orderhistory] where product = 'Volcano 62 inch Plasma TV VC2332' select c.session_id, c.connect_time, c.net_transport, c.last_read, c.last_write, c.client_net </pre> |

The audit policy active on the target system was configured to log successful logins only, and not login failures. However, SQL Server maintains its own log that records database related service errors in addition to authentication data. The error log was stored within the “*c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG*” directory of the target system. Review of the error log identified several hundred failed login attempts in succession against the sa account, followed by its successful login. This activity is normally attributed to evidence of a successful brute force attack against the database server.

| | |
|------------------------------|--|
| 2007-03-02 07:39:08.60 Logon | Login failed for user 'sa'. [CLIENT: 192.168.1.20] |
| 2007-03-02 07:39:08.80 Logon | Error: 18456, Severity: 14, State: 8. |
| 2007-03-02 07:39:08.80 Logon | Login failed for user 'sa'. [CLIENT: 192.168.1.20] |
| 2007-03-02 07:39:09.00 Logon | Error: 18456, Severity: 14, State: 8. |
| 2007-03-02 07:39:09.00 Logon | Login failed for user 'sa'. [CLIENT: 192.168.1.20] |
| 2007-03-02 07:39:09.20 Logon | Error: 18456, Severity: 14, State: 8. |
| 2007-03-02 07:39:09.20 Logon | Login failed for user 'sa'. [CLIENT: 192.168.1.20] |
| 2007-03-02 07:39:09.40 Logon | Error: 18456, Severity: 14, State: 8. |
| 2007-03-02 07:39:09.40 Logon | Login failed for user 'sa'. [CLIENT: 192.168.1.20] |
| 2007-03-02 07:39:09.60 Logon | Error: 18456, Severity: 14, State: 8. |
| 2007-03-02 07:39:09.60 Logon | Login failed for user 'sa'. [CLIENT: 192.168.1.20] |
| 2007-03-02 07:39:09.80 Logon | Error: 18456, Severity: 14, State: 8. |
| 2007-03-02 07:39:09.80 Logon | Login failed for user 'sa'. [CLIENT: 192.168.1.20] |
| 2007-03-02 07:39:10.00 Logon | Error: 18456, Severity: 14, State: 8. |
| 2007-03-02 07:39:10.00 Logon | Login failed for user 'sa'. [CLIENT: 192.168.1.20] |
| 2007-03-02 07:39:10.20 Logon | Error: 18456, Severity: 14, State: 8. |
| 2007-03-02 07:39:10.20 Logon | Login failed for user 'sa'. [CLIENT: 192.168.1.20] |
| 2007-03-02 07:39:10.40 Logon | Error: 18456, Severity: 14, State: 8. |
| 2007-03-02 07:39:10.40 Logon | Login failed for user 'sa'. [CLIENT: 192.168.1.20] |
| 2007-03-02 07:39:10.60 Logon | Error: 18456, Severity: 14, State: 8. |
| 2007-03-02 07:39:10.60 Logon | Login failed for user 'sa'. [CLIENT: 192.168.1.20] |
| 2007-03-02 07:39:10.80 Logon | Error: 18456, Severity: 14, State: 8. |
| 2007-03-02 07:39:10.80 Logon | Login failed for user 'sa'. [CLIENT: 192.168.1.20] |
| 2007-03-02 07:39:11.00 Logon | Error: 18456, Severity: 14, State: 8. |
| 2007-03-02 07:39:11.00 Logon | Login failed for user 'sa'. [CLIENT: 192.168.1.20] |
| 2007-03-02 07:39:11.20 Logon | Error: 18456, Severity: 14, State: 8. |
| 2007-03-02 07:39:11.20 Logon | Login failed for user 'sa'. [CLIENT: 192.168.1.20] |
| 2007-03-02 07:53:07.39 Logon | Login succeeded for user 'sa'. Connection: non-trusted. [CLIENT: 192.168.1.20] |

To further investigate the above findings, the configuration of the SQL Server needed to be obtained. SQLCMD, a Microsoft issued utility which allows the submission of ad-hoc SQL

statements and scripts to a MS SQL Server will be used from the trusted incident response CD. The ad-hoc query capabilities of this tool will be used during the remainder of this investigation.

The “*D:\FResponse\Sqlcmd -S PRODSQL05 -e -s,”*” command was executed from the trusted command prompt which opened a connection to the SQL Server using the interactive user context. The “*-e*” switch forces SQLCMD to echo our input statements into the SQL result files and the “*-s,”*” switch ensures the outputs are comma delimited which will allow the results to be imported into another application for deeper analysis.

After logging in, an output file was established to log the SQL statements and their associated results securely to my forensics workstation.

```
:out e:\initialconnection.txt
```

A MD5 hash will be created on each output file to ensure data integrity. When a connection is made to SQL Server the default database context configured under the user Login Properties will be used. To ensure the database context was indeed set for the OnlineSales database the following command was issued:

```
use OnlineSales  
go
```

Results: initialconnection.txt

SQL Server 2005 can be configured to use either Windows Authentication, which allows the host operating system to authenticate users, or Mixed Mode authentication, which allows authentication to occur at either the Operating System or independently within SQL Server⁴. There are also various logging options within SQL Server to log successful and/or failed login attempts. To verify the active configuration settings of the subject server the following command was run:


```
xp_loginconfig
```

Results: xp_loginconfig--onlinesales.txt

The following results were produced and show that the server is set for Mixed Mode authentication and is configured to log both successful and failed login attempts.

| name | config_value |
|----------------|------------------|
| login mode | Mixed |
| default login | guest |
| default domain | ESALECO |
| audit level | all |
| set hostname | false |
| map _ | domain separator |
| map \$ | NULL |
| map # | - |

Authorization within SQL Server 2005 is controlled by two gates. The first gate ensures that users are authenticated at the database instance and the second ensures that users have the appropriate permissions to access the various databases and database objects. During the verification step of this investigation we identified that the SQL server login EASYACCESS was logged into the server. However because the investigation is on the OnlineSales database the database permissions will need to be checked to ensure that the EASYACCESS account has access to this specific database. The following query was run to gather a list of all database users within the OnlineSales database:

```
Select * from sys.database_principals where type = 'S' or type = 'U' order by create_date,
modify_date
```

Results: db_principals-onlinesales.txt

This query produced the following results which show that the EASYACCESS user account does have access to the OnlineSales database:

| name | principal_id | type | type_desc | default_schema_name | create_date | modify_date |
|------------|--------------|------|--------------|---------------------|-------------|-------------|
| sys | 4 | S | SQL_USER | NULL | 38639 | 38639 |
| Lance | 5 | U | WINDOWS_USER | dbo | 39140 | 39140 |
| EASYACCESS | 6 | S | SQL_USER | dbo | 39143 | 39143 |

The Microsoft extended procedure *“xp_cmdshell”* allows users to execute dos commands within the underlying host operating system using their SQL client. This can allow an attacker who compromises the SQL Server to then launch attacks against the underlying host operating system. However, this procedure is disabled by default in SQL Server 2005. To verify its current state, the following command was executed:

```
Select * from sys.configurations
```

Results: sys.configurations.txt

The results showed that this procedure was disabled therefore the assumption is made that database users are unable to execute operating system level commands on the host.

| configuration_id | name | value | minimum | maximum | value_in_use | description | ... |
|------------------|----------------------------|-------|---------|---------|--------------|--|-----|
| 16390 | xp_cmdshell | 0 | 0 | 1 | 0 | Enable or disable command shell | ... |
| 16391 | Ad Hoc Distributed Queries | 0 | 0 | 1 | 0 | Enable or disable Ad Hoc Distributed Queries | ... |

At approximately 10:45 AM, the initial findings were presented to the client who verified that the EASYACCESS account was an anomaly and not created for any legitimate business purpose. It was also disclosed that the Online Sales application was down for maintenance therefore no one should have been logged on to the OnlineSalesdatabase or have executed the identified delete statement.

At 11:01 AM the client authorized a full forensic investigation to be performed on the server to determine the scope and impact of the intrusion. At 11:05 AM The SQL Server was

disconnected from the production network and plugged into a 4 port DLINK hub to isolate the server and prevent further modification by the unknown user.

Step 2: System Description

As previously stated in the verification section of this document, upon login to the target server the default Microsoft background was visible on the server console and there were no third party applications visible within the system tray. The following system profile was gathered from information provided by the client as well as investigator findings gathered during the verification step:

| | |
|-------------------------|--|
| System Name | PRODSQL05 |
| Serial Number | US822301223 |
| System Operating System | Microsoft Windows Server 2003 Service Pack 1 |
| Database Version | 9.00.1399.06 |
| System Function | Function as a backend database to an online order processing system |
| Physical Description | The system contained 3 peripheral network cards, one appeared to be a video card, and the remaining two appeared to be network cards, however, only a single network card was actually connected to the network. |

Asset Photographs:**Step 3: Evidence Collection**

As time elapses after a security incident, evidence can be overwritten by legitimate and/or malicious system activity. Databases can contain large data stores which result in a high data acquisition cost. To help ensure priority is given to the data sources most likely to contain relevant data to support the investigation, it's my expert opinion that relevant data sources be

assigned a significance and also a volatility value between 1 -5 with, 5 being of higher significance and/or volatility. The following values should be used in the following formula to determine priority [10 - (significance rating) + (volatility rating) = priority]. Using the above formula, the data stores relevant in this investigation were prioritized as follows:

| Item | Importance | Volatility | Priority |
|-----------------------------------|------------|------------|----------|
| SQL Server Connections & Sessions | 5 | 5 | 0 |
| Transaction Log(s) | 5 | 4 | 1 |
| SQL Server Logs | 4 | 3 | 3 |
| SQL Server Database Files | 3 | 2 | 5 |
| System Event Logs | 2 | 2 | 6 |

Now that data stores have been identified and prioritized, the actual data acquisition can take place.

SQL Server connection & session data

Related information was successfully captured via the customized Windows Forensic Tool chest tool executed during the verification stage of this investigation.

Transaction Logs

The SQL Server transaction log contains a record of all insert, update and delete statements made within the database. For performance reasons SQL Server does not immediately write these events to the physical data files. Instead changes are written to the log file to buffer and later written to the data files.

A single SQL Server database can utilize multiple database files and multiple transaction logs. The number of files and locations will need to be identified for the OnlineSales database.

Using the trusted SQLCMD session, the following SQL query is executed to gather the database file information:

```
sp_helpdb OnlineSales
```

Results: sp_helpdb-onlinesales.txt

The below results were returned from the above SQL query and show that the OnlineSales database is currently using one physical data file ending with the ".mdf" extension and two transaction log files ending with the ".ldf" extension. These files are contained within separate Windows file locations.

| name | fileid | filename | ... |
|------------------|--------|--|-----|
| OnlineSales | 1 | C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\OnlineSales.mdf | ... |
| OnlineSales_log | 2 | C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\OnlineSales_log.ldf | ... |
| OnlineSales_log2 | 3 | C:\OtherLogs\OnlineSales_log2.ldf | ... |

The following SQL query was then executed to dump the contents of the OnlineSales log file to the trusted forensic workstation:

```
dbcc log(OnlineSales)
```

Results: dbcclog-onlinesales.txt

Although a SQL Server database can use multiple physical transaction logs internally, SQL Server splits each physical log file into 4-16 Virtual Log Files (VLFs)⁵. Selected VLFs are marked active at any given time and used to record transactions. SQL Server periodically completes a checkpoint process which flushes changes recorded in the log file to the physical disk file. Once this is complete, SQL Server marks the VLFs containing the fully committed transactions reusable and will overwrite them as required with new log records.

The following SQL Server command was run from within the OnlineSales context to view the logical allocation status of the physical transaction log:

```
dbcc loginfo
```

Results: dbccloginfo-onlinesales.txt

The results of this command may be helpful later in the investigation when it will be determined if the physical transaction log file will be split into virtual log files to separate the active VLFs from the reusable VLFs which may contain historical data relevant in this investigation. In order to obtain a true bit-to-bit copy of the transaction log, the SQL Server service will need to be shutdown in order to release the locks held on the target files. At SQL Server shutdown and startup the database checkpoint process is automatically triggered⁵ which, as previously stated before will flush the non committed changes to disk and mark the records as reusable. The following command was executed to force the shutdown of SQL Server.

```
Shutdown
```

Results: shutdown.txt

After the SQL SERVER processes were shutdown, the physical log files were acquired using the dcfldd disk imaging tool which also generated MD5 hashes for the acquired data. These hashes were compared to the hashes of the on disk files to ensure the data was not altered during duplication.

Database files

Using the database file locations retrieved from the results of the "sp_helpdb OnlineSales" command executed earlier in the investigation, the OnlineSales database file was also acquired using the dcfldd tool.

Default SQL Server Trace File

The default configuration of SQL server runs a trace which captures limited activity within the database. This configuration is enabled by default, but can be disabled by a user with sufficient privileges. Using the SQL Server configuration gathered earlier in this investigation, the default trace was confirmed to be enabled.

| configuration_id | name | value | minimum | maximum | value in use | description |
|------------------|------------------------------|----------|----------|----------|--------------|--|
| 1567 | ft crawl bandwidth (max) | 100 | 0 | 32767 | 100 | Max number of full-text crawl buffers |
| 1568 | default trace enabled | 1 | 0 | 1 | 1 | Enable or disable the default trace |
| 1569 | blocked process threshold | 0 | 0 | 86400 | 0 | Blocked process reporting threshold |

During review of the SQL Server installation directory several trace files using the default Microsoft trace naming convention "log_###" were identified. These log files were acquired using the dcfldd tool as they may contain information relevant in this investigation.

SQL Server Error Logs

In addition to the current error log used by SQL Server, historical log data is also maintained. Each time the SQL Server service is restarted, a new error log is created and the existing log is backed up. SQL Server maintains the current error log in addition to 6 log backups. All 7 error logs were acquired using the dcfldd tool. Once all data had been acquired the SQL Server services were restarted.

Step 4: Timeline Creation

Constructing an initial timeline will map out the notable digital events which have been identified thus far and establish an investigation scope which will be used during the Media Analysis phase. Review of the SQL Server error logs obtained during the Evidence Collection step show that the SQL Server instance was restarted on March 01, 2007.

```
2007-03-01 07:26:22.53 Server      SQL Server is now ready for client connections. This is an informational message..
```

This will be the first entry in the timeline. As discovered during the verification step of this investigation on March 2nd, 2007 several hundred failed SQL Server login attempts were recorded within the error log between 7:01 AM to 7:39 AM from IP address 192.168.1.20. Following these failed login attempts were successful logins by the SA account at 7:54 AM and the EASYACCESS account at 8:09 AM from the same IP address.

```
2007-03-02 07:39:11.00 Logon      Error: 18456, Severity: 14, State: 8.
2007-03-02 07:39:11.00 Logon      Login failed for user 'sa'. [CLIENT: 192.168.1.20]
2007-03-02 07:39:11.20 Logon      Error: 18456, Severity: 14, State: 8.
2007-03-02 07:39:11.20 Logon      Login failed for user 'sa'. [CLIENT: 192.168.1.20]
2007-03-02 07:53:07.39 Logon      Login succeeded for user 'sa'. Connection: non-trusted. [CLIENT: 192.168.1.20]
2007-03-02 08:09:37.60 Logon      Login succeeded for user 'EASYACCESS'. Connection: non-trusted. [CLIENT: 192.168.1.20]
```

These events will be added to the timeline in addition to the associated Server Process Identifier (SPID). A SPID is a unique number used by SQL Server to track a given session within the database server². The trace files obtained during the evidence collection phase of this investigation were imported into MS SQL Profiler on my forensic workstation for analysis. During review, the following notable events were identified:

- (1) Creation of EASYACCESS account
- (2) EASYACCESS account is granted access to OnlineSales database
- (3) EASYACCESS account is added to ONLINESALES db_owner role

(4-6) Unknown transactions are executed by EASYACCESS account which required tempdb usage. Often DML operations require tempdb usage² therefore it is likely that SPID 51 issued DML operations which required object or interim result storage.

| LoginName | SPID | StartTime | EventSubClass | DatabaseName | TransactionID | TargetLoginName | RoleName |
|-------------------------|------|-------------------------|---------------------------|--------------|---------------|-----------------|----------|
| sa | 51 | 2007-03-02 07:39:10.200 | | master | | | |
| sa | 51 | 2007-03-02 07:39:10.400 | | master | | | |
| sa | 51 | 2007-03-02 07:39:10.600 | | master | | | |
| sa | 51 | 2007-03-02 07:39:10.800 | | master | | | |
| sa | 51 | 2007-03-02 07:39:11.003 | | master | | | |
| sa | 51 | 2007-03-02 07:39:11.203 | | master | | | |
| sa | 51 | 2007-03-02 07:54:07.180 | 1 - Add | master | 3559 | EASYACCESS | |
| sa | 51 | 2007-03-02 07:54:34.030 | 1 - Commit | tempdb | 3615 | | |
| sa | 51 | 2007-03-02 07:54:35.740 | | master | 3722 | | |
| sa | 51 | 2007-03-02 07:54:35.903 | 0 - Begin | tempdb | 3787 | | |
| sa | 51 | 2007-03-02 07:54:35.913 | 1 - Commit | tempdb | 3787 | | |
| sa | 51 | 2007-03-02 07:55:52.783 | 3 - Grant database access | OnlineSales | 4126 | EASYACCESS | |
| sa | 51 | 2007-03-02 07:56:18.440 | 1 - Add | OnlineSales | 4171 | EASYACCESS | db_owner |
| EASYACCESS | 51 | 2007-03-02 08:09:33.773 | 1 - Commit | tempdb | 4860 | | |
| EASYACCESS | 2 | 2007-03-02 08:13:29.350 | 1 - Increase | tempdb | | | |
| EASYACCESS | 51 | 2007-03-02 08:13:31.433 | 1 - Commit | tempdb | 5651 | | |
| EASYACCESS | 51 | 2007-03-02 08:13:32.667 | 1 - Commit | tempdb | 5848 | | |
| PRODSQL05\Administrator | 52 | 2007-03-02 10:17:38.283 | 1 - Commit | tempdb | 12166 | | |
| PRODSQL05\Administrator | 52 | 2007-03-02 11:05:24.943 | 1 - Commit | tempdb | 14988 | | |

Based on the events identified thus far in the investigation, the following timeline was constructed:

| Time | User | SPID | Action |
|----------------------|------------|------|---|
| March 1, 2007 | | | |
| 7:26 AM | UNKNOWN | N/A | SQL Server instance is restarted |
| March 2, 2007 | | | |
| 7:01 AM – 7:39 AM | UNKNOWN | 51 | SQL Server Brute Force attack launched against PRODSQL05 server |
| 7:54 AM | SA | 51 | SA SQL Server user account logs into PRODSQL05 server |
| 7:54 AM | SA | 51 | EASYACCESS account created |
| 7:55 AM | SA | 51 | EASYACCESS account granted access to OnlineSales database |
| 7:56 AM | SA | 51 | EASYACCESS account added to OnlineSales db_owner role |
| 8:09 AM | EASYACCESS | 51 | EASYACCESS SQL Server account logs into |

| | | | |
|----------|---------------|-----|---|
| | | | PRODSQL05 server |
| 8:09 AM | EASYACCESS | 51 | EASYACCESS account executes unknown transaction within ONLINESALES db |
| 8:13 AM | EASYACCESS | 51 | EASYACCESS account executes unknown transaction within OnlineSales database |
| 8:13 AM | EASYACCESS | 51 | EASYACCESS account executes unknown transaction within OnlineSales database |
| 10:17 AM | Administrator | 52 | Start of Forensic Investigation of database server |
| 11:05 AM | Administrator | N/A | PRODSQL05 server removed from network |
| 11:16 AM | Administrator | 52 | SQL Server instance shutdown |

The application connected to SPID 51 was recorded by SQL Server as "OSQL-32". Performing a *Google™* search on this name identified the application as a legacy Microsoft command line query tool called OSQL. This will be noted as it may be relevant in the future if an investigation is performed on the unauthorized user's computer.

| | | | | | | | |
|------------|-------------------------|----|-------------------------|---------------|---|----------|---|
| squeld 1.0 | sa | 51 | 2007-03-02 07:39:11.003 | | 0 | | 0 |
| squeld 1.0 | sa | 51 | 2007-03-02 07:39:11.203 | | 0 | | 0 |
| OSQL-32 | sa | 51 | 2007-03-02 07:54:07.180 | 1 - Add | 1 | 0x01 | 0 |
| OSQL-32 | sa | 51 | 2007-03-02 07:54:34.030 | 1 - Commit | | 0x01 | 0 |
| OSQL-32 | sa | 51 | 2007-03-02 07:54:35.740 | | | 0x01 | 0 |
| OSQL-32 | sa | 51 | 2007-03-02 07:54:35.903 | 0 - Begin | | 0x01 | 0 |
| OSQL-32 | sa | 51 | 2007-03-02 07:54:35.913 | 1 - Commit | | 0x01 | 0 |
| OSQL-32 | sa | 51 | 2007-03-02 07:55:52.783 | 3 - Grant ... | 1 | 0x01 | 0 |
| OSQL-32 | sa | 51 | 2007-03-02 07:56:18.440 | 1 - Add | 1 | 0x01 | 0 |
| OSQL-32 | EASYACCESS | 51 | 2007-03-02 08:09:33.773 | 1 - Commit | | 0xB89... | 0 |
| | | 2 | 2007-03-02 08:13:29.350 | 1 - Increase | | | 0 |
| OSQL-32 | EASYACCESS | 51 | 2007-03-02 08:13:31.433 | 1 - Commit | | 0xB89... | 0 |
| OSQL-32 | EASYACCESS | 51 | 2007-03-02 08:13:32.667 | 1 - Commit | | 0xB89... | 0 |
| SQLCMD | PRODSQL05\Administrator | 52 | 2007-03-02 10:17:38.283 | 1 - Commit | | 0x010... | 0 |

Step 5: Media Analysis

The timeline established in the previous step will now be used to set boundaries on the scope of media analysis. Using the timeline, the focus of the investigation will be on activities executed by SPID 51 between 7:54 AM March 1st, 2007 when the unauthorized access was gained to SQL Server and later in the day at 11:05 AM when the system was isolated from the production network.

Before looking at any of the raw SQL Data files, the data types in use within the OnlineSalesdatabase will need to be identified. Unicode is a standard method of mapping SQL Server byte representations (code points) to ASCII characters. The Unicode standard is inclusive of characters which map to all languages throughout the world. SQL Server uses various data types which store Unicode data, however there are some data types used by SQL Server (char(n), varchar(n) & text) which store non-Unicode values³. When non-Unicode values are stored within SQL Server, they are converted to a supported data type using the collation setting of the respective table column³. If this data is viewed by a computer using a code page which does not cover the range of characters used within the collation setting of the database, data loss can occur which can skew the results³. To determine if non-Unicode data was being used by the Order table and the collation setting in place, the following procedure was run:

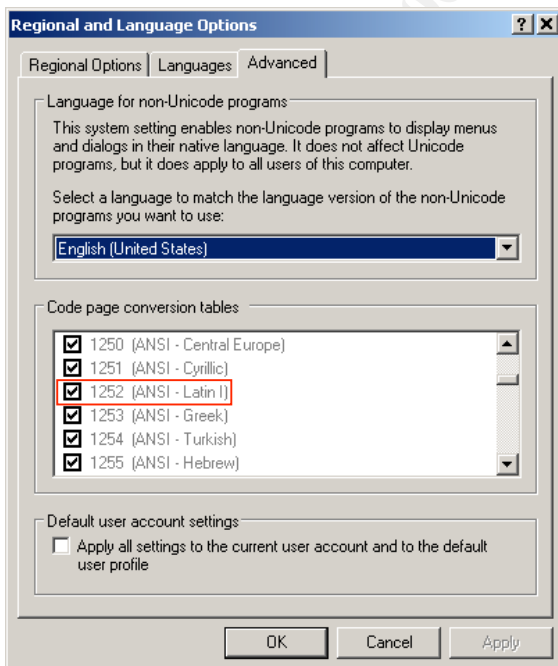
```
sp_tablecollations 'order'
```

Results: sp_tablecollations-onlinesales.txt

The results below show that both Unicode and non-Unicode data is stored within the Order table. The columns storing non-Unicode data are using the SQL_Latin1_General_CP1_CI_AS collation setting.

| colid | name | tds_collation | collation |
|-------|--------------|---------------|------------------------------|
| 1 | OrderID | NULL | NULL |
| 2 | FirstName | 0x0904D00034 | SQL_Latin1_General_CP1_CI_AS |
| 3 | LastName | 0x0904D00034 | SQL_Latin1_General_CP1_CI_AS |
| 4 | Address | 0x0904D00034 | SQL_Latin1_General_CP1_CI_AS |
| 5 | City | 0x0904D00034 | SQL_Latin1_General_CP1_CI_AS |
| 6 | State | 0x0904D00034 | SQL_Latin1_General_CP1_CI_AS |
| 7 | ZIP | 0x0904D00034 | SQL_Latin1_General_CP1_CI_AS |
| 8 | CCType | 0x0904D00034 | SQL_Latin1_General_CP1_CI_AS |
| 9 | CCNumber | 0x0904D00034 | SQL_Latin1_General_CP1_CI_AS |
| 11 | ShipStatusID | NULL | NULL |
| 12 | OrderDate | NULL | NULL |
| 13 | Product | 0x0904D00034 | SQL_Latin1_General_CP1_CI_AS |
| 14 | Price | 0x0904D00034 | SQL_Latin1_General_CP1_CI_AS |

This collation setting was researched on *SQL Server 2005 Books Online* which showed that this collation maps to code page 1252⁴. To verify the code page in use on my forensic workstation, the regional and language options application within control panel on my forensic workstation was viewed. This identified that the forensic workstation was using a compliant code page in order to correctly translate the code points used by SQL Server.



The transaction log acquired during the evidence collection phase was imported into Microsoft Excel using code page 1252. A SQL Server 2005 transaction log contains over 100 columns however only a few columns will contain relevant data based on the scope of this investigation. The following table outlines target columns and their function within this investigation.

| Column | Description |
|------------------|--|
| Operation | The type of operation which was performed |
| PageID | The data page affected by the transaction |
| SlotID | The row within the data page affected by the transaction |
| Offset in Row | The first position within the data row affected by the transaction |
| SPID | The Server Process Identifier |
| Begin Time | Indicates the transaction start time (server time) |
| Transaction Name | Classification of the active transaction |
| End Time | Indicates the transaction end time (server time) |
| RowLogContents0 | The value which was updated by the transaction (Insert, Update statements) |
| RowLogContents1 | The value which was written to disk (Insert, Update statements) |

For a listing of all columns within the transaction log, please see Appendix B of this document.

The imported data set was filtered to display only records which were executed by SPID 51 and between the date/time ranges captured in the timeline. The first two transactions identified, were associated with the creation and permission augmentation of the EASYACCESS account which was identified during the trace file review.

| Server UID | UID | SPID | Beginlog Status | Begin Time | Transaction Name | Transaction SID | End Time | Transaction Begin |
|------------|------|------|-----------------|-------------------------|------------------|-----------------|-------------------------|------------------------|
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 0 | -1 | 51 | 0x01000000 | 2007/03/02 07:55:52.813 | CREATE USER | 0x01 | NULL | NULL |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| 0 | -1 | 51 | 0x01000000 | 2007/03/02 07:56:18.440 | user_transaction | 0x01 | 2007/03/02 07:55:52.833 | 00000010:00000e1c:0001 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | 2007/03/02 07:56:18.450 | 00000010:00000e1f:0001 |

The third transaction executed by SPID 51 was an update statement. The transaction log details show that a database transaction ID 0000:0000032e which was an update statement affecting 3 records within 3 separate data pages within the database.



A SQL Server data page is an 8192 byte structure which stores database data⁵. A data page can contain multiple rows and each database contains multiple data pages. Data pages are organized into logical groups of 8 called extents⁶. Using the transaction log dump, the first update statement was analyzed, identifying a record on row 20 of Data Page 0001:000000d3. Both the

Page ID and Transaction ID values are stored in hex and when converted to decimal produce the following values:

| Identifier | Hex | Decimal |
|----------------|---------------|---------|
| Transaction ID | 0000:0000032e | 0:814 |
| Data Page | 0001:000000d3 | 1:211 |

In order to view the raw data pages, the OnlineSales database was attached within SQL Server Management Studio (SSMS) version 9.00.1399.00 on my forensic workstation. Within the newly added OnlineSales database, Microsoft-issued commands and procedures will be used to examine the raw data pages which have been modified.

The following command was issued from within the OnlineSales database context

```
dbcc page (OnlineSales, 1, 211, 1)
```

The above command dumped data page 211 which contained the row which had been modified. The header of the table was examined to identify the base table to which the data page belonged.


```

PAGE HEADER:

Page @0x04304000

m_pageId = (1:211)                m_headerVersion = 1                m_type = 1
m_typeFlagBits = 0x0              m_level = 0                          m_flagBits = 0x0
m_objId (AllocUnitId.idObj) = 87  m_indexId (AllocUnitId.idInd) = 256
Metadata: AllocUnitId = 72057594043629568
Metadata: PartitionId = 72057594039500800
Metadata: ObjectId = 629577281    m_prevPage = (1:314)                Metadata: IndexId = 1
pminlen = 108                     m_slotCnt = 22                       m_nextPage = (1:315)
m_freeData = 5918                 m_reservedCnt = 0                     m_freeCnt = 3263
m_xactReserved = 0                m_xdesId = (0:0)                     m_lsn = (16:3686:2)
m_tornBits = -1731484635          m_ghostRecCnt = 0

Allocation Status

GAM (1:2) = ALLOCATED              SGAM (1:3) = NOT ALLOCATED
PFS (1:1) = 0x60 MIXED_EXT ALLOCATED  O_PCT_FULL                          DIFF (1:6) = CHANGED
ML (1:7) = NOT MIN_LOGGED
    
```

Objectid 629577281 was used as an argument in the following query which was run to resolve the name of the object.

```
Select * from sysobjects where id = 629577281
```

This produced the following output which confirmed that the data page belonged to the Order table.

| name | id | xtype | uid | info | status | base_sch | replinfo | parent_obj | crdate | ... |
|-------|-----------|-------|-----|------|--------|----------|----------|------------|-----------------|-----|
| Order | 629577281 | U | 1 | 0 | 0 | 0 | 0 | 0 | 2/26/07 4:08 PM | ... |

The method used by SQL Server to store data depends on the data types in use, the size of each column and the order in which the columns were specified when the table was created. Before the raw data pages were examined, the table schema was first gathered by executing the following command:

```
SELECT sc.colorder, sc.name, st.name as 'datatype', sc.length FROM syscolumns sc,
```

```

systypes st
WHERE sc.xusertype = st.xusertype and sc.id = 629577281
ORDER BY colorder

```

The following output was produced which illustrates the schema of the Order table:

| colorder | name | datatype | length |
|----------|--------------|----------|--------|
| 1 | OrderID | int | 4 |
| 2 | FirstName | varchar | 20 |
| 3 | LastName | varchar | 20 |
| 4 | Address | varchar | 50 |
| 5 | City | nchar | 40 |
| 6 | State | nchar | 4 |
| 7 | ZIP | nchar | 10 |
| 8 | CCType | varchar | 15 |
| 9 | CCNumber | varchar | 20 |
| 11 | ShipStatusID | int | 4 |
| 12 | OrderDate | datetime | 8 |
| 13 | Product | nvarchar | 100 |
| 14 | Price | nchar | 30 |

Using slotID: 20 and rowoffset 80 which were obtained previously from the transaction log, the specific point within the data row was identified in which the transaction began.

```
Slot 20 Offset 0x147f Length 237

Record Type = PRIMARY_RECORD          Record Attributes = NULL_BITMAP VARIABLE_COLUMNS

Memory Dump @0x2F3AD47F

00000000: 30006c00 6f000000 53007000 72006900 +0.l.o...S.p.r.i.
00000010: 6e006700 4c006100 6b006500 20002000 tn.g.L.a.k.e. . .
00000020: 20002000 20002000 20002000 20002000 + . . . . .
00000030: 41005a00 31003400 34003100 30000a00 t.A.Z.1.4.4.1.0...
00000040: 00000100 00000000 0000e498 00003300 t.....3.
00000050: 2e003500 30002000 20002000 20002000 t..5.0. . . . .
00000060: 20002000 20002000 20002000 0e0000c0 + . . . . .
00000070: 06008400 88009900 9d00ad00 ed00416e t.....An
00000080: 6f736f6e 456d696c 37322053 74617266 tosonEmil72 Starf
00000090: 656c6c20 44726976 65566973 61343931 tell DriveVisa49l
000000A0: 36383833 38343033 38323330 3056006f +6883840382300V.o
000000B0: 006c0063 0061006e 006f0020 00360032 t.l.c.a.n.o. .6.2
000000C0: 00200069 006e0063 00680020 0050006c t. .i.n.c.h. .P.l
000000D0: 00610073 006d0061 00200054 00560020 t.a.s.m.a. .T.V.
000000E0: 00560043 00320033 00330032 00+++++++t.V.C.2.3.3.2.
```

Using the table schema obtained earlier, the data type within this row offset is the Price column which contains a 30-byte nchar data type. From the transaction log, the hexadecimal value from the Rowlog0 and Rowlog1 columns were extracted and converted to decimal representation.

RowLog0

| | | | | | | | | | | | |
|--------------|----|----|----|----|----|----|----|----|----|----|----|
| Hex | 35 | 00 | 30 | 00 | 30 | 00 | 2E | 00 | 30 | 00 | 30 |
| ASCII | 5 | | 0 | | 0 | | . | | 0 | | 0 |

RowLog1

| | | | | | | | | | | | |
|--------------|----|----|----|----|----|----|----|----|----|----|----|
| Hex | 2E | 00 | 35 | 00 | 30 | 00 | 20 | 00 | 20 | 00 | 20 |
| ASCII | . | | 5 | | 0 | | SP | | SP | | SP |

Mapping the data page determined that the offset for the price column is 0x4f (79), as identified, the update statement began at offset 80. This was done so SQL Server did not have to overwrite a value in which it would need to rewrite as part of the transaction. Therefore the offset was augmented by SQL Server from 79 to 80 to compensate. Taking this into consideration, the

statement executed under transaction 0000:0000032e (0:814) was to update the price column from "3500.00" to "3.50":

```

Slot 20 Offset 0x147f Length 237

Record Type = PRIMARY_RECORD          Record Attributes = NULL_BITMAP VARIABLE_COLUMNS

Memory Dump @0x2F3&D47F

00000000:  30006c00 6f000000 53007000 72006900 +0.l.o...S.p.r.i.
00000010:  6e006700 4c006100 6b006500 20002000 tn.g.L.a.k.e. . .
00000020:  20002000 20002000 20002000 20002000 + . . . . .
00000030:  41005a00 31003400 34003100 30000a00 +A.Z.1.4.4.1.0...
00000040:  00000100 00000000 0000e498 00003300 +.....3.
00000050:  2e003500 30002000 20002000 20002000 +..5.0. . . . .
00000060:  20002000 20002000 20002000 0e0000c0 + . . . . .
00000070:  06008400 88009900 9d00ad00 ed00416e +.....An
00000080:  6f736f6e 456d696c 37322053 74617266 tosonEmil72 Starf
00000090:  656c6c20 44726976 65566973 61343931 tell DriveVisa491
000000A0:  36383833 38343033 38323330 3056006f +6883840382300V.o
000000B0:  006c0063 0061006e 006f0020 00360032 +.l.c.a.n.o. .6.2
000000C0:  00200069 006e0063 00680020 0050006c +. .i.n.c.h. .P.1
000000D0:  00610073 006d0061 00200054 00560020 +.a.s.m.a. .T.V.
000000E0:  00560043 00320033 00330032 00+++++++V.C.2.3.3.2.
    
```

Using the same steps outlined above, the remaining 2 records updated during this transaction were identified.

Slot 13, Offset 0x1450, Length 239, DumpStyle BYTE

Record Type = PRIMARY_RECORD

Record Attributes = NULL_BITMAP VARIABLE_COLUMNS

Memory Dump @0x2F7AD450

```

00000000: 30006c00 08010000 53007000 72006900 +0.l.....S.p.r.i.
00000010: 6e006700 4c006100 6b006500 20002000 tn.g.L.a.k.e. . .
00000020: 20002000 20002000 20002000 20002000 + . . . . .
00000030: 41005a00 31003700 30003000 33000100 +A.Z.1.7.0.0.3...
00000040: 00000100 00000000 0000e498 00003300 +.....3.
00000050: 2e003500 30002000 20002000 20002000 +..5.0. . . . .
00000060: 20002000 20002000 20002000 0e0000c0 + . . . . .
00000070: 06008400 8a009b00 9f00af00 ef00436f +.....Co
00000080: 72796e6e 466f776c 65723732 20537461 trynnFowler72 Sta
00000090: 7266656c 6c204472 69766556 69736135 trfell DriveVisa5
000000A0: 35313835 33303030 30303030 30303056 +518530000000000V
000000B0: 006f006c 00630061 006e006f 00200036 +.o.l.c.a.n.o. .6
000000C0: 00320020 0069006e 00630068 00200050 +.2. .i.n.c.h. .P
000000D0: 006c0061 0073006d 00610020 00540056 +.l.a.s.m.a. .T.V
000000E0: 00200056 00430032 00330033 003200++++. .V.C.2.3.3.2.
    
```

RowLog0

| | | | | | | | | | | | |
|--------------|----|----|----|----|----|----|----|----|----|----|----|
| Hex | 35 | 00 | 30 | 00 | 30 | 00 | 2E | 00 | 30 | 00 | 30 |
| ASCII | 5 | | 0 | | 0 | | . | | 0 | | 0 |

RowLog1

| | | | | | | | | | | | |
|--------------|----|----|----|----|----|----|----|----|----|----|----|
| Hex | 2E | 00 | 35 | 00 | 30 | 00 | 20 | 00 | 20 | 00 | 20 |
| ASCII | . | | 5 | | 0 | | SP | | SP | | SP |

```
Slot 7, Offset 0x11c6, Length 240, DumpStyle BYTE

Record Type = PRIMARY_RECORD      Record Attributes = NULL_BITMAP VARIABLE_COLUMNS

Memory Dump @0x2F2AD1C6

00000000: 30006c00 46010000 53007000 72006900 +0.l.F...S.p.r.i.
00000010: 6e006700 4c006100 6b006500 20002000 tn.g.L.a.k.e. . .
00000020: 20002000 20002000 20002000 20002000 + . . . . .
00000030: 41005a00 31003400 34003100 30000a00 +A.Z.1.4.4.1.0...
00000040: 00000100 00000000 0000e498 00003300 +.....3.
00000050: 2e003500 30002000 20002000 20002000 +..5.0. . . . .
00000060: 20002000 20002000 20002000 0e0000c0 + . . . . .
00000070: 06008200 8b009c00 a000b000 f0004162 +.....Ab
00000080: 6965a04f 274e6569 6c6c2037 32205374 tie.0'Neill 72 St
00000090: 61726665 6c6c2044 72697665 56697361 tarfell DriveVisa
000000A0: 34393136 38383030 30303030 30303030 +4916880000000000
000000B0: 56006f00 6c006300 61006e00 6f002000 +V.o.l.c.a.n.o. .
000000C0: 36003200 20006900 6e006300 68002000 +6.2. .i.n.c.h. .
000000D0: 50006c00 61007300 6d006100 20005400 +P.l.a.s.m.a. .T.
000000E0: 56002000 56004300 32003300 33003200 +V. .V.C.2.3.3.2.
```

RowLog0

| | | | | | | | | | | | |
|--------------|----|----|----|----|----|----|----|----|----|----|----|
| Hex | 35 | 00 | 30 | 00 | 30 | 00 | 2E | 00 | 30 | 00 | 30 |
| ASCII | 5 | | 0 | | 0 | | . | | 0 | | 0 |

RowLog1

| | | | | | | | | | | | |
|--------------|----|----|----|----|----|----|----|----|----|----|----|
| Hex | 2E | 00 | 35 | 00 | 30 | 00 | 20 | 00 | 20 | 00 | 20 |
| ASCII | . | | 5 | | 0 | | SP | | SP | | SP |

It is noted that all 3 records updated during this transaction were associated with the “Volcano 62 inch Plasma TV VC2332” product.

The fourth transaction executed by SPID 51 was another update statement. The transaction log details show that transaction ID: 0000:0000032f was an update statement affecting 2 records located on 2 separate data pages.

| Operation | Context | Transaction ID | Page ID | Slot ID | ... | Offset in Row |
|-----------------|---------------|----------------|---------------|---------|-----|---------------|
| LOP_BEGIN_XACT | LCX_NULL | 0000:0000032f | NULL | NULL | | NULL |
| LOP_MODIFY_ROW | LCX_CLUSTERED | 0000:0000032f | 0001:000000d3 | 20 | | 66 |
| LOP_MODIFY_ROW | LCX_CLUSTERED | 0000:0000032f | 0001:000000d6 | 7 | | 66 |
| LOP_COMMIT_XACT | LCX_NULL | 0000:0000032f | NULL | NULL | | NULL |

The same process used previously was followed to identify the affected records. The row offset and page ID values obtained from the transaction log were used to identify the specific value updated within the following records:

```
Slot 7 Offset 0x12b6 Length 243

Record Type = PRIMARY_RECORD          Record Attributes = NULL_BITMAP VARIABLE_COLUMNS

Memory Dump @0x0E64D2B6

00000000: 30006c00 46010000 42006500 6c006c00 +0.1.F...B.e.1.1.
00000010: 65007600 75006500 20002000 20002000 +e.v.u.e. . . . .
00000020: 20002000 20002000 20002000 20002000 + . . . . .
00000030: 43005400 32003100 30003600 37000a00 +C.T.2.1.0.6.7...
00000040: 00000200 00000000 0000e398 00003300 +.....3.
00000050: 35003000 30002e00 30003000 20002000 +5.0.0...0.0. . .
00000060: 20002000 20002000 20002000 0e0000c0 + . . . . .
00000070: 06008200 8b009f00 a300b300 f3004162 +.....Ab
00000080: 6965a04f 274e6569 6c6c2032 32372057 +ie.0'Neill 227 W
00000090: 494e4448 4156454e 20535452 45455456 +INDHAVEN STREETV
000000A0: 69736134 39313638 38303030 30303030 +isa4916880000000
000000B0: 30303056 006f006c 00630061 006e006f +000V.o.l.c.a.n.o
000000C0: 00200036 00320020 0069006e 00630068 +. .6.2. .i.n.c.h
000000D0: 00200050 006c0061 0073006d 00610020 +. .P.l.a.s.m.a.
000000E0: 00540056 00200056 00430032 00330033 +.T.V. .V.C.2.3.3
000000F0: 003200+++++.....2.
```

The data type within this offset of the row is the ShipStatusID which is a 4-byte integer value.

RowLog0

| | | | | |
|-------|----|----|----|----|
| Hex | 00 | 01 | 00 | 00 |
| ASCII | 00 | 01 | 00 | 00 |

RowLog1

| | | | | |
|-------|----|----|----|----|
| Hex | 00 | 02 | 00 | 00 |
| ASCII | 00 | 02 | 00 | 00 |

```

Slot 20 Offset 0x156c Length 238

Record Type = PRIMARY_RECORD          Record Attributes = NULL_BITMAP VARIABLE_COLUMNS

Memory Dump @0x0E6CD56C

00000000: 30006c00 6f000000 57006800 69007400 +0.l.o...W.h.i.t.
00000010: 62007900 20002000 20002000 20002000 +b.y. . . . .
00000020: 20002000 20002000 20002000 20002000 + . . . . .
00000030: 46004c00 33003200 37003000 31000a00 +F.L.3.2.7.0.1...
00000040: 00000200 00000000 00008e98 00003300 +.....3.
00000050: 35003000 30002e00 30003000 20002000 +5.0.0...0.0. .
00000060: 20002000 20002000 20002000 0e0000c0 + . . . . .
00000070: 06008400 88009a00 9e00ae00 ee00416e +.....An
00000080: 6f736f6e 456d696c 37342048 65726963 +osonEmil74 Heric
00000090: 6b736f6e 20445249 56455669 73613439 +kson DRIVEVisa49
000000A0: 31363838 33383430 33383233 30305600 +16883840382300V.
000000B0: 6f006c00 63006100 6e006f00 20003600 +o.l.c.a.n.o. .6.
000000C0: 32002000 69006e00 63006800 20005000 +2. .i.n.c.h. .P.
000000D0: 6c006100 73006d00 61002000 54005600 +l.a.s.m.a. .T.V.
000000E0: 20005600 43003200 33003300 3200+++++ .V.C.2.3.3.2.
    
```

RowLog0

| | | | | |
|-------|----|----|----|----|
| Hex | 00 | 01 | 00 | 00 |
| ASCII | 00 | 01 | 00 | 00 |

RowLog1

| | | | | |
|-------|----|----|----|----|
| Hex | 00 | 02 | 00 | 00 |
| ASCII | 00 | 02 | 00 | 00 |

It is noted that after querying the ShipStatus table the ShipStatusID value of 1 indicates that an order has been shipped and a value of 2 indicates that the order has yet to be shipped. It is the investigator's belief that the value was updated from 2 to 1 in an attempt to have the customer

repeat shipment of the referenced product to the designated address.

The fifth transaction executed by SPID 51 was an insert statement. The transaction log details show that a database transaction 0000:00000330 affected a single row.

| Operation | Context | Transaction ID | Page ID | Slot ID | ... | Offset in Row |
|-----------------|---------------|----------------|---------------|---------|-----|---------------|
| LOP_BEGIN_XACT | LCX_NULL | 0000:00000330 | NULL | NULL | | NULL |
| LOP_INSERT_ROWS | LCX_CLUSTERED | 0000:00000330 | 0001:00000138 | 8 | | NULL |
| LOP_COMMIT_XACT | LCX_NULL | 0000:00000330 | NULL | NULL | | NULL |

The same procedure used to map the previous update statements to a data pages was followed to identify the inserted record:

```
Slot 8 Offset 0xc6 Length 188

Record Type = PRIMARY_RECORD      Record Attributes = NULL_BITMAP VARIABLE_COLUMNS

Memory Dump @0x0E64CCF6

00000000: 30006c00 a1010000 53007000 72006900 +0.1.....S.p.r.i.
00000010: 6e006700 4c006100 6b006500 20002000 +n.g.L.a.k.e. . .
00000020: 20002000 20002000 20002000 20002000 + . . . . .
00000030: 41005a00 31003400 34003100 3000827c +A.Z.1.4.4.1.0..|
00000040: 23fb0200 00000000 0000e498 00003400 +#. . . . .4.
00000050: 2e003000 30002000 20002000 20002000 +.0.0. . . . .
00000060: 20002000 20002000 20002000 0e0000c2 + . . . . .
00000070: 06008200 87009800 9c00ac00 bc004e69 +. . . . .Ni
00000080: 6e6f426c 61636b37 32205374 61726665 +noBlack72 Starfe
00000090: 6c6c2044 72697665 56697361 35353138 +11 DriveVisa5518
000000A0: 35333030 30303030 30303030 58004200 +530000000000X.B.
000000B0: 4f005800 20003300 36003000 ++++++++0.X. .3.6.0.
```

Querying the remainder of the transactions showed that no future modifications were made to this slot within the data page 0000:00000330 therefore the data currently residing on the data page remains unchanged from its state as inserted during this transaction. The values contained within this record are as follows:

| | |
|-----------------------------|-----------------------------------|
| OrderID = 417 | CCType = Visa |
| FirstName = Nino | CCNumber = 5518530000000000 |
| LastName = Black | ShipStatusID = 2 |
| Address = 72 Starfell Drive | OrderDate = March 1, 2007 12:00AM |
| City = SpringLake | Product = XBOX 360 |
| State = AZ | Price = 4.00 |
| ZIP = 14410 | |

The price associated with this item seems inaccurate, and will be flagged for review by the client. It was also noted that the credit card number used in this insert statement was also associated with one of the records updated during transaction 815.

The sixth transaction executed by SPID 51 was transaction 0000:00000331 an update statement affecting 3 records.

| Operation | Context | Transaction ID | Page ID | Slot ID | ... | Offset in Row |
|-----------------|---------------|----------------|---------------|---------|-----|---------------|
| LOP_BEGIN_XACT | LCX_NULL | 0000:00000331 | NULL | NULL | | NULL |
| LOP_MODIFY_ROW | LCX_CLUSTERED | 0000:00000331 | 0001:000000d3 | 20 | | 74 |
| LOP_MODIFY_ROW | LCX_CLUSTERED | 0000:00000331 | 0001:0000013c | 13 | | 74 |
| LOP_MODIFY_ROW | LCX_CLUSTERED | 0000:00000331 | 0001:000000d6 | 7 | | 74 |
| LOP_COMMIT_XACT | LCX_NULL | 0000:00000331 | NULL | NULL | | NULL |

The same procedure used earlier to map the previous update statements to a data pages was followed here and resolved to the Order table. Using the table schema obtained earlier, the data type within this row offset is the OrderDate column which contains an 8-byte datetime data type. The first record updated during this transaction was located on data page 211, slot 20 and the updated column began at offset 74.

```

Slot 20 Offset 0x147f Length 237

Record Type = PRIMARY_RECORD          Record Attributes = NULL_BITMAP VARIABLE_COLUMNS

Memory Dump @0x0E01D47F
OrderDate Column
00000000: 30006c00 6f000000 53007000 72006900 +0.l.o...S.p.r.i.
00000010: 6e006700 4c006100 6b006500 20002000 +n.g.L.a.k.e. . .
00000020: 20002000 20002000 20002000 20002000 + . . . . .
00000030: 41005a00 31003400 34003100 30000a00 +A.Z.l.4.4.l.O...
00000040: 00000100 00000000 0000e498 000003300 +.....3.
00000050: 2e003500 30002000 20002000 20002000 +..5.O. . . . .
00000060: 20002000 20002000 20002000 0e0000c0 + . . . . .
00000070: 06008400 88009900 9d00ad00 ed00416e +.....An
00000080: 6f736f6e 456d696c 37322053 74617266 +osonEmil72 Starf
00000090: 656c6c20 44726976 65566973 61343931 +ell DriveVisa49l
000000A0: 36383833 38343033 38323330 3056006f +6883840382300V.o
000000B0: 006c0063 0061006e 006f0020 00360032 +.l.c.a.n.o. .6.2
000000C0: 00200069 006e0063 00680020 0050006c +. .i.n.c.h. .P.l
000000D0: 00610073 006d0061 00200054 00560020 +.a.s.m.a. .T.V.
000000E0: 00560043 00320033 00330032 00+++++++V.C.2.3.3.2.
    
```

Updated Value

The method in which computers store multiple-byte values vary, some use little-endian ordering (LEO) and others use big-endian ordering (BEO)⁷. With little-endian ordering, the most significant byte of the number is placed in the first storage byte; big-endian does the reverse and stores the least significant byte in the first storage byte. Microsoft operating systems use little-endian ordering⁷, which is also true in the way SQL Server stores numeric values.

From the transaction log the hexadecimal values from the Rowlog0 and Rowlog1 columns were extracted, switched into LEO and converted to decimal representation.

RowLog0

| | |
|-----------|--------------------|
| Hex (BEO) | 0x0000000000BD9800 |
| Hex (LEO) | 0x000000000098BD00 |
| Decimal | 39101 |

RowLog1

| | |
|------------------|--------------------|
| Hex (BEO) | 0x0000000000E49800 |
| Hex (LEO) | 0x000000000098E400 |
| Decimal | 39140 |

The datetime data type within SQL Server breaks an 8-byte date value into 2 fragments, the first being the number of days before or after January 1st, 1900 and the second being the number of clock computer ticks after midnight with a tick occurring every 3.33 milliseconds⁵. Applications using the datetime data type to store date values only, will have a default time value of 00:00:00:000 which represents midnight⁵. The decimal representation of the RowLog1 column is 39140 which when added in days to January 1st, 1900 gives us the date of March 01, 2007. The order date of this record was updated from January 21, 2007 to March 01, 2007.

This procedure was used to identify the remaining two values which were updated within transaction 0000:00000331.

Slot 13 Offset 0x1450 Length 239

Record Type = PRIMARY_RECORD Record Attributes = NULL_BITMAP VARIABLE_COLUMNS

Memory Dump @0x0E8DD450

| Address | Hex | ASCII |
|-----------|-------------------|-------------------------------------|
| 00000000: | 30006c00 08010000 | 53007000 72006900 +0.1.....S.p.r.i. |
| 00000010: | 6e006700 4c006100 | 6b006500 20002000 tn.g.L.a.k.e. . . |
| 00000020: | 20002000 20002000 | 20002000 20002000 + |
| 00000030: | 41005a00 31003700 | 30003000 33000100 +A.Z.1.7.0.0.3... |
| 00000040: | 00000100 00000000 | 0000e498 00000300 +.....3. |
| 00000050: | 2e003500 30002000 | 20002000 20002000 +..5.0. |
| 00000060: | 20002000 20002000 | 20002000 0e0000c0 + |
| 00000070: | 06008400 8a009b00 | 9f00af00 ef00436f +.....Co |
| 00000080: | 72796e6e 466f776c | 65723732 20537461 +rynnFowler72 Sta |
| 00000090: | 7266656c 6c204472 | 69766556 69736135 +rfell DriveVisa5 |
| 000000A0: | 35313835 33303030 | 30303030 30303056 +518530000000000V |
| 000000B0: | 006f006c 00630061 | 006e006f 00200036 +.o.l.c.a.n.o. .6 |
| 000000C0: | 00320020 0069006e | 00630068 00200050 +.2. .i.n.c.h. .P |
| 000000D0: | 006c0061 0073006d | 00610020 00540056 +.l.a.s.m.a. .T.V |
| 000000E0: | 00200056 00430032 | 00330033 003200++++. .V.C.2.3.3.2. |

OrderDate Column

Updated Value

RowLog0 (on disk value prior to transaction)

| | |
|-----------|--------------------|
| Hex (BEO) | 0x0000000000BD9800 |
| Hex (LEO) | 0x000000000098BD00 |
| Decimal | 39101 |

RowLog1 (committed transaction value)

| | |
|-----------|--------------------|
| Hex (BEO) | 0x0000000000E49800 |
| Hex (LEO) | 0x000000000098E400 |
| Decimal | 39140 |

Slot 7 Offset 0x11c6 Length 240

Record Type = PRIMARY_RECORD Record Attributes = NULL_BITMAP VARIABLE_COLUMNS

Memory Dump @0x0E74D1C6

| Address | Hex | ASCII |
|-----------|-------------------------------------|-------------------|
| 00000000: | 30006c00 46010000 53007000 72006900 | +0.l.F...S.p.r.i. |
| 00000010: | 6e006700 4c006100 6b006500 20002000 | +n.g.L.a.k.e. . . |
| 00000020: | 20002000 20002000 20002000 20002000 | + |
| 00000030: | 41005a00 31003400 34003100 30000a00 | +A.Z.1.4.4.1.0... |
| 00000040: | 00000100 00000000 0000e498 00000330 | +.....3. |
| 00000050: | 2e003500 30002000 20002000 20002000 | +..5.0. |
| 00000060: | 20002000 20002000 20002000 0e0000c0 | + |
| 00000070: | 06008200 8b009c00 a000b000 f0004162 | +.....Ab |
| 00000080: | 6965a04f 274e6569 6c6c2037 32205374 | +ie.0'Neill 72 St |
| 00000090: | 61726665 6c6c2044 72697665 56697361 | +arfell DriveVisa |
| 000000A0: | 34393136 38383030 30303030 30303030 | +4916880000000000 |
| 000000B0: | 56006f00 6c006300 61006e00 6f002000 | +V.o.l.c.a.n.o. . |
| 000000C0: | 36003200 20006900 6e006300 68002000 | +6.2. .i.n.c.h. . |
| 000000D0: | 50006c00 61007300 6d006100 20005400 | +P.l.a.s.m.a. .T. |
| 000000E0: | 56002000 56004300 32003300 33003200 | +V. .V.C.2.3.3.2. |

OrderDate Column

Updated Value

RowLog0 (on disk value prior to transaction)

| | |
|-----------|--------------------|
| Hex (BEO) | 0x0000000000CE9800 |
| Hex (LEO) | 0x000000000098CE00 |
| Decimal | 39118 |

RowLog1 (committed transaction value)

| | |
|-----------|--------------------|
| Hex (BEO) | 0x0000000000E49800 |
| Hex (LEO) | 0x000000000098E400 |
| Decimal | 39140 |

The seventh transaction executed by SPID 51 was transaction 0000:00000332, a delete statement

affecting a single record.

| Operation | Context | Transaction ID | Page ID | Slot ID | ... | Offset in Row |
|-----------------|-------------------|----------------|---------------|---------|-----|---------------|
| LOP_BEGIN_XACT | LCX_NULL | 0000:00000332 | NULL | NULL | ... | |
| LOP_DELETE_ROWS | LCX_MARK_AS_GHOST | 0000:00000332 | 0001:00000158 | 24 | | |
| LOP_SET_BITS | LCX_PFS | 0000:00000000 | 0001:00000001 | 0 | | |
| LOP_COMMIT_XACT | LCX_NULL | 0000:00000332 | NULL | NULL | | |

This record will be further examined during the data recovery stage of this investigation.

Step 6: Data Recovery

The seventh transaction executed by SPID 51 was transaction 0000:00000332, a delete statement affecting a single record. When a record is deleted within SQL Server, it is marked as a ghost⁵, which tells the database engine to hide it from future query results even though the underlying data still resides within the data page. A garbage clean-up process runs periodically within SQL Server to physically remove the ghost records within the data pages so the space can be reused. Ghost records contained within a data page are flagged within the page header. Examining the header of the page 0001:0000000158 (1:344) containing the deleted row showed that the `m_ghostRecCnt` value was set at 0 indicating that the ghost records had already been physically removed from the data page.

```

Page @0x043D0000

m_pageId = (1:344)                m_headerVersion = 1                m_type = 1
m_typeFlagBits = 0x4              m_level = 0                        m_flagBits = 0x8200
m_objId (AllocUnitId.idObj) = 78  m_indexId (AllocUnitId.idInd) = 256
Metadata: AllocUnitId = 72057594043039744
Metadata: PartitionId = 72057594039042048
Metadata: ObjectId = 245575913     m_prevPage = (1:190)               Metadata: IndexId = 1
pminlen = 108                    m_slotCnt = 27                    m_nextPage = (1:191)
m_freeData = 6899                m_reservedCnt = 0                  m_freeCnt = 2876
m_xactReserved = 0               m_xdesId = (0:818)                m_lsn = (16:3626:1)
m_tornBits = -1097693874         m_ghostRecCnt = 0

```

Using the same procedure used earlier in this document to map a data page to the owning table identified that the data page associated in this transaction mapped to the OrderHistory table. This table had an identical schema to that of the Order table. Within the transaction log, the following value was obtained from the RowLog0 column of the delete statement:

```

“0x30006C009F000000500061007900650074007400650020002000200020002000
200020002000200020002000200046004C00310036003600300032000100000000000003A980
00033003500300030002E003000300020002000200020002000200020000E0000C0060082
00860098009C00AD00CD004275727443617665323237205374617267656C6C2044726976655
66973613635393033343030333433323233323030566F6C63616E6F20363220696E636820506
C61736D6120545620564332333332”

```

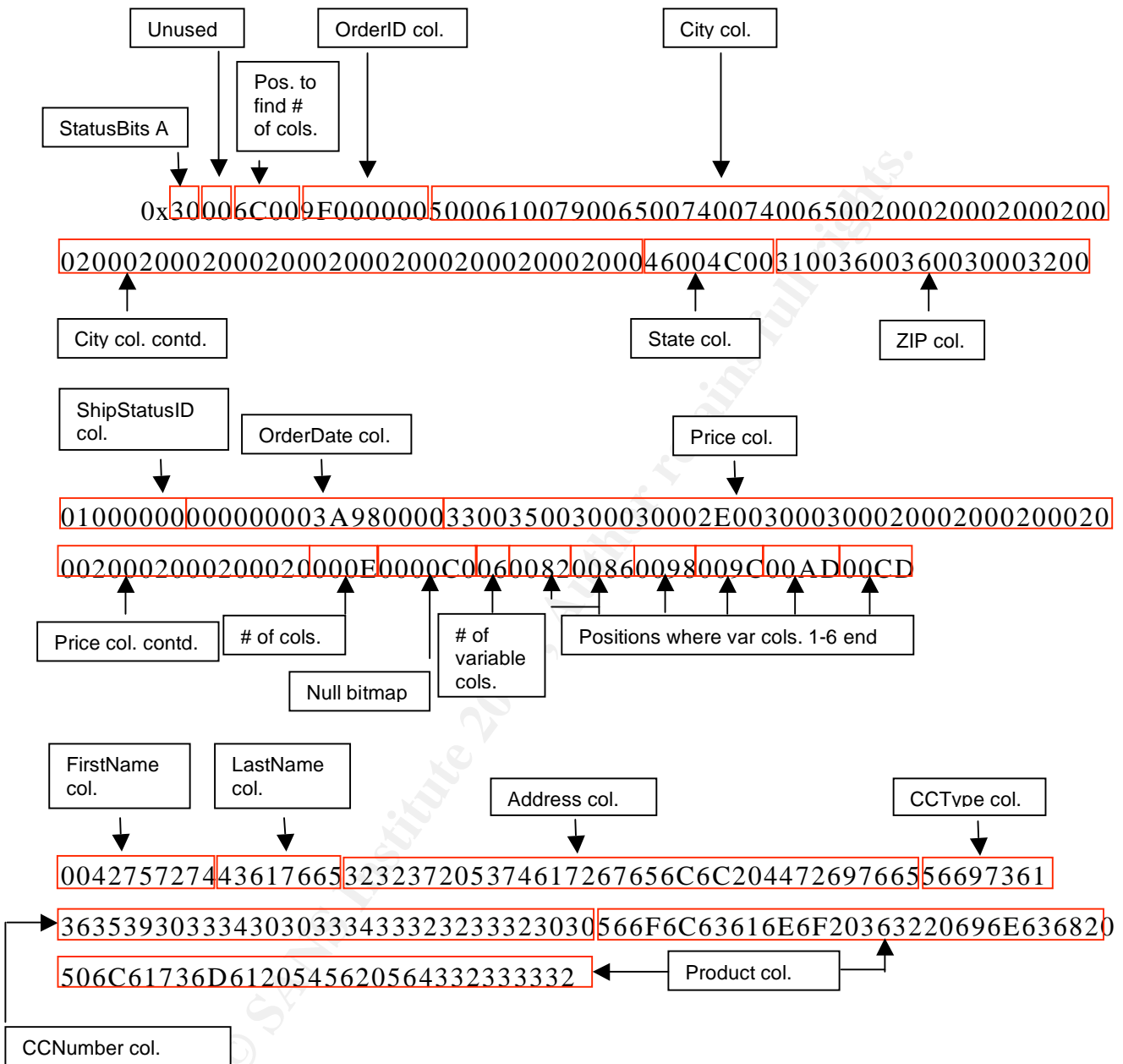
The data above is the actual data row deleted from the data page during the transaction. To determine exactly what customer data had been deleted, it was necessary to reconstruct the data row. SQL Server uses two different data row structures, one for rows which contain fixed length columns only, and another for rows containing variable length columns and/or fixed length columns. Based on the schema obtained earlier in this investigation we know that the Order table contains both fixed and variable length data types. The data row structure for a variable length row is as follows:

| | | | | | | | | |
|---|---|---|----------------------|---|---|---|---|-------------------------|
| 1 | 2 | 3 | Fixed length columns | 4 | 5 | 6 | 7 | Variable length columns |
|---|---|---|----------------------|---|---|---|---|-------------------------|

Source: Inside SQL Server 2005 The Storage Engine⁵

| Legend | | |
|-------------------------|--|---|
| Item | Storage Allocation | Description |
| 1 | 1 byte | StatusBits A contains data row properties ⁵ |
| 2 | 1 byte | Unused in SQL Server 2005 ⁵ |
| 3 | 2 bytes | Row offset to in row location containing the number of columns in the data row ⁵ |
| Fixed length columns | Fixed column length for all fixed columns | Location of in row fixed length data columns ⁵ |
| 4 | 2 bytes | Total number of columns in data row ⁵ |
| 5 | 1 bit for each row column | Null Bitmap ⁵ |
| 6 | 2 bytes | Number of variable length columns within data row ⁵ |
| 7 | 2 bytes for each variable length column | Row offset marking the end of each variable length column ⁵ |
| Variable length columns | Used length of all variable length columns | Location of in row variable length data columns ⁵ |

Using the above row structure, and the data obtained from the RowLog0 column of the transaction log, the data row was reconstructed.



Switching the appropriate hex values into LEO, and converting the values to decimal/ASCII representation produced the following.

OrderID: 159

FirstName: Burt

LastName: Cave

Address: 227 Stargell Drive

City: Payette

State: FL

ZIP: 16602

CCType: Visa

CCNumber: 65903400343223200

ShipStatusID: 1

OrderDate: September 11th, 2006

Product: Volcano 62 inch Plasma TV VC2332

Price: 3500.00

Now that all of the executed transactions have been identified, the timeline was updated to reflect the notable discoveries.

| Time | User | SPID | Action |
|----------------------|------------|------|---|
| March 1, 2007 | | | |
| 7:26 AM | UNKNOWN | N/A | SQL Server instance is restarted |
| March 2, 2007 | | | |
| 7:01 AM – 7:39 AM | UNKNOWN | 51 | SQL Server Brute Force attack launched against PRODSQL05 server from IP 192.168.1.20 |
| 7:54 AM | SA | 51 | SA SQL Server user account logs into PRODSQL05 server from IP address 192.168.1.20 using OSQL.exe |
| 7:54 AM | SA | 51 | EASYACCESS account is created |
| 7:55 AM | SA | 51 | EASYACCESS account is granted access to OnlineSales database |
| 7:56 AM | SA | 51 | EASYACCESS account added to OnlineSales db_owner role |
| 8:09 AM | EASYACCESS | 51 | SQL Server account logs into PRODSQL05 server |

| | | | |
|----------|---------------|-----|---|
| | | | from IP address 192.168.1.20 |
| 8:17 AM | EASYACCESS | 51 | Transaction 814 is executed which updates the price of 3 Volcano Plasma TV orders from \$3500.00 to \$3.50 |
| 8:20 AM | EASYACCESS | 51 | Transaction 815 is executed which updates the shippingstatusID column on Volcano Plasma TV orders from 1 (product shipped) to 2 (product not shipped) |
| 8:31 AM | EASYACCESS | 51 | Transaction 816 is executed which inserts an order for an XBOX 360 billed to the credit card of another database customer. |
| 8:37 AM | EASYACCESS | 51 | Transaction 817 is executed which sets the orderdate on Plasma TV orders to February 28, 2007 |
| 8:38 AM | EASYACCESS | 51 | Transaction 818 is executed which deletes a previous order from the OrderHistory table for a Volcano Plasma TV at a price of \$3500.00 |
| 10:17 AM | Administrator | 52 | Start of Forensic Investigation of database server |
| 11:05 AM | Administrator | N/A | PRODSQL05 server removed from network |
| 11:16 AM | Administrator | 52 | SQL Server instance shutdown |

Step 7: String Search

As stated previously in this report, a single physical transaction log file is logically partitioned and split into 4-16 Virtual Log Files (VLFs) by SQL Server. Only a subset of these VLFs will be active at any one point. It is possible that the inactive VLFs at one point in time were active and may contain past transaction data which is relevant within this investigation. Review of the active transaction log file used throughout this investigation identified that the earliest log entry was 7:26 AM March 1st, 2007 and the latest was 11:16 AM March 2nd, 2007. This date range is inclusive of the scope of the investigation therefore further review of VLFs is not required.

The published Microsoft tools which interpret transaction logs support only active VLFs. Further investigation into transactions which occurred outside of the scope of this investigation will require sting searches to be performed on the inactive areas of the transaction log to identify rows for reconstruction.

Investigation Summary

In conclusion, after gathering and analyzing all evidence, it is in the investigator's expert opinion that on the Morning of March 1st, 2007, an unauthorized user connecting from IP 192.168.1.20 executed a successful brute force attack against the PRODSQL05 server. Once access was gained to the database, a connection was made using the Microsoft OSQL client to create a backdoor account named EASYACCESS. This account was used by the user to fraudulently insert an erroneous product order for an XBOX 360 with the incorrect price of \$4.00. This order was billed to Visa card number 5518530000000000 which belongs to another customer within the database. It is noted that the mailing addresses used within the fraudulent order differs from the address of the compromised user and may belong to the unauthorized user.

In addition to inserting a fraudulent order, the unauthorized user performed the following updates to existing Volcano 62 inch Plasma TV VC2332 orders within the Order table.

- Order dates were set to February 28th, 2007
- Prices were updated from \$3500.00 to \$3.50
- The shippingstatusID column was updated from 2 to 1

A single record was also deleted from the Order table for a past Volcano 62 inch Plasma TV VC2332 by the user.

The unauthorized user is believed to have had a general understanding of Transact-SQL (TSQL) syntax in order to have been capable of executing the database transactions via the OSQL command line interface and moderate knowledge of the OnlineSales database schema.

© SANS Institute 2007, Author retains full rights.

Appendix A

The following was text was added to WFT configuration file

```
#####
# SQL SERVER #
#####

M    NA    NA    NA    NA    SQL SERVER NA

V    SQLCMD.RLL    341369b133a26556d963427384ca89ba    NA    NA    NA
    Required by sqlcmd.exe

EVH  SQLCMD.exe 28731c04b854cc1570dbdacc89a6c3f2    %s -E -Q "sp_helpdb" >
%s%s%s    sp_helpdb    DB LISTING SP_HELPDB SQL SERVER

EH   SQLCMD.exe 28731c04b854cc1570dbdacc89a6c3f2    %s -E -Q "select c.session_id,
c.connect_time, c.net_transport, c.last_read, c.last_write, c.client_net_address, c.local_tcp_port,
s.text from sys.dm_exec_connections c cross apply sys.dm_exec_sql_text
(c.most_recent_sql_handle) s" > %s%s%s    dm_exec_connections
    DM_EXEC_CONNECTIONS    DM_EXEC_CONNECTIONS    SQL SERVER

EH   SQLCMD.exe 28731c04b854cc1570dbdacc89a6c3f2    %s -E -Q "select * from
sys.dm_exec_sessions" > %s%s%s    dm_exec_sessions    DM_EXEC_SESSIONS
    DM_EXEC_SESSIONS    SQL SERVER

EH   SQLCMD.exe 28731c04b854cc1570dbdacc89a6c3f2    %s -E -Q "select name,
type_desc, create_date, modify_date from sys.sql_logins order by create_date, modify_date" >
%s%s%s    sql_logins    SQL_LOGINS    SQL_LOGINS
    SQL SERVER

EH   SQLCMD.exe 28731c04b854cc1570dbdacc89a6c3f2    %s -E -Q "select * from
```

```
sys.dm_exec_requests " > %s%s%s dm_exec_requests DM_EXEC_REQUESTS  
DM_EXEC_REQUESTS SQL SERVER
```

© SANS Institute 2007, Author retains full rights.

Appendix B

Transaction Log Column listing:

| | | | | | |
|----|-------------------------|----|--------------------------|----|----------------------|
| 1 | CurrentLSN | 24 | CHKPT End DB Version | 46 | PrepLogBegin LSN |
| 2 | Operation | 25 | Minimum LSN | 47 | PrepareTime |
| 3 | Context | 26 | Dirty Pages | 48 | Virtual Clock |
| 4 | Transaction ID | | Oldest Replicated Begin | 49 | Previous Savepoint |
| 5 | Tag Bits | 27 | LSN | 50 | Savepoint Name |
| 6 | Log Record Fixed Length | 28 | Next Replicated End LSN | 51 | Rowbits First Bit |
| 7 | Log Record Length | 29 | Last Distributed End LSN | 52 | Rowbits Bit Count |
| 8 | PreviousLSN | 30 | Server UID | 53 | Rowbits Bit Value |
| 9 | Flag Bits | 31 | UID | 54 | Number of Locks |
| | AllocUnitID | 32 | SPID | 55 | Lock Information |
| 11 | AllocUnitName | 33 | BeginLogStatus | 56 | LSN Before Writes |
| 12 | Page ID | 34 | Begin Time | 57 | Pages Written |
| 13 | Slot ID | 35 | Transaction Name | 58 | Data Pages Delta |
| 14 | Previous Page LSN | 36 | Transaction SID | 59 | Reserved Pages Delta |
| 15 | PartionID | 37 | End Time | 60 | Used Pages Delta |
| 16 | RowFlags | 38 | Transaction Begin | 61 | Data Rows Delta |
| 17 | Num Elements | 39 | Replicated Records | 62 | Command Type |
| 18 | Offset in Row | 40 | Oldest Active LSN | 63 | Publication ID |
| 19 | Checkpoint Begin | 41 | Server Name | 64 | Article ID |
| 20 | CHKPT Begin DB Version | 42 | Database Name | 65 | Partial Status |
| 21 | MaxXDESID | 43 | Mark Name | 66 | Command |
| 22 | Num Transactions | 44 | Master XDESID | 67 | Byte Offset |
| 23 | Checkpoint End | 45 | Master DBID | 68 | New Value |

GIAC Gold Template

| | | | | | |
|----|---------------------|----|-----------------------------|-----|---------------------------|
| 69 | Old Value | 71 | Rows Deleted | 73 | CI Table ID |
| 70 | New Split Page | 72 | Bytes Freed | 74 | CI Index ID |
| 75 | NewAllocationUnitID | 85 | Column Offset | | Bulk allocation first IAM |
| 76 | FilegroupID | 86 | Flags | 95 | Page ID |
| 77 | Meta Status | 87 | Text Size | 96 | Bulk allocated extent ids |
| 78 | File Status | 88 | Offset | 97 | RowLog Contents 0 |
| 79 | File ID | 89 | Old Size | 98 | RowLog Contents 1 |
| 80 | Physical Name | 90 | New Size | 99 | RowLog Contents 2 |
| 81 | Logical Name | 91 | Description | 100 | RowLog Contents 3 |
| 82 | Format LSN | 92 | Bulk allocated extent count | 101 | RowLog Contents 4 |
| 83 | RowsetID | 93 | Bulk rowinsertID | | |
| 84 | TextPtr | 94 | Bulk allocationunitID | | |

References

- ¹ Keith J. Jones, Richard Bejtlich, Curtis W. Rose. Real Digital Forensics, Addison-Wesley, 2006
- ² “MSDN Blog Pages” <http://blogs.msdn.com/sqlserverstorageengine/default.aspx>
- ³ Microsoft Developer Network “MSDN” <http://msdn2.microsoft.com/en-us/default.aspx>
- ⁴ SQL Server 2005 Books Online, <http://msdn2.microsoft.com/en-us/library/ms130214.aspx>
- ⁵ Kalen Delaney. Inside SQL Server 2005 The Storage Engine, Microsoft Press, 2007
- ⁶ Kalen Delaney and Jim Gray. Inside SQL Server 2000. Microsoft Press, 2001
- ⁷ Brian Carrier. File System Forensic Analysis. Addison-Wesley, 2005

Upcoming SANS App Sec Training



| | | | |
|---|------------------------|-----------------------------|----------------|
| SANS London July 2017 | London, United Kingdom | Jul 03, 2017 - Jul 08, 2017 | Live Event |
| SANSFIRE 2017 | Washington, DC | Jul 22, 2017 - Jul 29, 2017 | Live Event |
| Community SANS Minneapolis DEV534 | Minneapolis, MN | Aug 25, 2017 - Aug 28, 2017 | Community SANS |
| Community SANS San Francisco DEV541 | San Francisco, CA | Aug 28, 2017 - Aug 31, 2017 | Community SANS |
| SANS Network Security 2017 | Las Vegas, NV | Sep 10, 2017 - Sep 17, 2017 | Live Event |
| Secure DevOps Summit & Training | Denver, CO | Oct 10, 2017 - Oct 17, 2017 | Live Event |
| SANS Seattle 2017 | Seattle, WA | Oct 30, 2017 - Nov 04, 2017 | Live Event |
| SANS Cyber Defense Initiative 2017 | Washington, DC | Dec 12, 2017 - Dec 19, 2017 | Live Event |
| SANS Cyber Defense Initiative 2017 - DEV522: Defending Web Applications Security Essentials | Washington, DC | Dec 14, 2017 - Dec 19, 2017 | vLive |
| SANS Security East 2018 | New Orleans, LA | Jan 08, 2018 - Jan 13, 2018 | Live Event |
| SANS OnDemand | Online | Anytime | Self Paced |
| SANS SelfStudy | Books & MP3s Only | Anytime | Self Paced |