SQL Server Database Forensics
Kevvie Fowler, GCFA Gold, CISSP, MCTS, MCDBA, MCSD, MCSE
Black Hat USA 2007
Why are databases critical assets?

- Databases hold critical information
- Industry trends are scaling in versus out
- Database servers today hold more sensitive information than ever before
- Data security legislations & regulations dictate that security breaches must be reported
- Database security breaches are “Front Page” news
  - T.J. Maxx | 45.7 million credit/debit cards disclosed
  - CardSystems Solutions | 200,000 credit/debit cards disclosed
Traditional investigations often exclude databases
Database Forensics

The application of computer investigation and analysis techniques to gather database evidence suitable for presentation in a court of law

Benefits

- Retrace user DML & DDL operations
- Identify data pre and post transaction
- Recover previously deleted data rows
- Can help prove/disprove a data security breach
- Can help determine the scope of a database intrusion
- For the “real world”: No dependency on 3rd party auditing tools or pre-configured DML or DDL triggers
Database files
- Data files (.mdf) contain the actual data
- Consists of multiple data pages

Data rows can be fixed or variable length
Log files (.ldf) hold all data required to reverse transactions and recover the database
Physical log files consist of multiple Virtual Log Files (VLF)

A VLF is the unit of truncation for the transaction log
According to Microsoft:

“Although you might assume that reading the transaction log directly would be interesting or even useful, it’s just too much information.”

Inside the transaction log:

1. CurrentLSN
2. Operation
3. Context
4. Transaction ID
5. Tag Bits
6. Log Record Fixed Length
7. Log Record Length
8. PreviousLSN
9. Flag Bits
10. AllocUnitID
11. AllocUnitName
12. Page ID
13. Slot ID
14. Previous Page LSN
15. PartionID
16. RowFlags
17. NumFlags
18. Offset in Row
19. Checkpoint Begin
20. CHKPT Begin DB Version
21. MaxXDESID
22. Num Transactions
23. Checkpoint End

24. CHKPT End DB Version
25. Minimum LSN
26. Dirty Pages
27. Oldest Replicated Begin LSN
28. Next Replicated End LSN
29. Last Distributed End LSN
30. Server UID
31. UID
32. SPID
33. BeginLogStatus
34. Begin Time
35. Transaction Name
36. Transaction SID
37. End Time
38. Transaction Begin
39. Replicated Records
40. Oldest Active LSN
41. Server Name
42. Database Name
43. Mark Name
44. Master XDESID
45. Master DBID
46. PrepLogBegin LSN
47. PrepareTime
48. Virtual Clock
49. Previous Savepoint
50. Savepoint Name
51. Rowbits First Bit
52. Rowbits Bit Count
53. Rowbits Bit Value
54. Number of Locks
55. Lock Information
56. LSN Before Wrties
57. Pages Written
58. Data Pages Delta
59. Reserved Pages Delta
60. Used Pages Delta
61. Data Rows Delta
62. Command Type
63. Publication ID
64. Article ID
65. Partial Status
66. Command
67. Byte Offset
68. New Value
69. Old Value
70. New Split Page
71. Rows Deleted
72. Bytes Freed
73. CI Table ID
74. CI Index ID
75. NewAllocationUnitID
76. FilegroupID
77. Meta Status
78. File Status
79. File ID
80. Physical Name
81. Logical Name
82. Format LSN
83. RowsetID
84. TextPtr
85. Column Offset
86. Flags
87. Text Size
88. Offset
89. Old Size
90. New Size
91. Description
92. Bulk allocated extent count
93. Bulk rowinsertID
94. Bulk allocationunitID
95. Bulk allocation first IAM Page ID
96. Bulk allocated extent ids
97. RowLog Contents 0
98. RowLog Contents 1
99. RowLog Contents 2
100. RowLog Contents 3
101. RowLog Contents 4
Server Process ID (SPID)
- A unique value used by SQL Server to track a given session within the database server
- Transaction log activity is logged against the executing SPID

Data type storage and retrieval
- 31 different data types
- Data types are stored and retrieved differently within SQL Server
- Storing and retrieving value: 21976 in various data types results in the following:

<table>
<thead>
<tr>
<th>Data type</th>
<th>On disk value</th>
<th>Retrieved value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR</td>
<td>3231393736</td>
<td>21976</td>
</tr>
<tr>
<td>INT</td>
<td>D855</td>
<td>21976</td>
</tr>
<tr>
<td>DATETIME</td>
<td>D855</td>
<td>3/3/1960</td>
</tr>
</tbody>
</table>

- Big endian ordering (BEO) is applicable to number formats

Procedure Cache
- Ah-hoc statement and procedure execution plans
SQL Server data resides natively within SQL Server and stored externally within the native Windows operating system.

Evidence repositories:
- **SQL Server**
  - Volatile database data
  - Database data files
  - Database log files
  - Plan cache
  - Data cache
  - Indexes
  - Tempdb
  - Version store

- **Operating System**
  - Trace files
  - System event logs
  - SQL Server error logs
  - Page file
  - NTFS journal
  - Memory
SQL Server Forensics | Investigation Tools

- SQL Server Management Studio Express
- SQLCMD
- Windows Forensic Toolchest
- DD\DCFLDD
- MD5SUM
- Netcat\CryptCat
- WinHex
- Native SQL Server views, functions and statements
  - Dynamic Management Views (DMV)
  - Database Consistency Checker (DBCC) commands
  - FN_*
- Lots of sanitized acquisition media
Evidence Collection
Determine the scope of evidence collection

Prioritize evidence collection
1. Volatile database data (sessions/connections, active requests, plan cache, etc.)
2. Transaction logs
3. Database files
4. SQL Server error logs
5. System event logs
6. Trace files
### Collecting volatile database data
- Can be automated using WFT & command line SQL tools
- GUI front end, binary validation and thorough logging
- Gathers volatile data internal and external to SQL Server

```
<table>
<thead>
<tr>
<th>Command</th>
<th>md5=28731C04E854CC1570DEBACC89A6C3E2</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SQLCMD.exe -E -Q &quot;select name, type_desc, create_date, modify_date from sys.sql_logins order by create_date modify_date&quot; &gt; e\sql_logins.txt</code></td>
<td></td>
</tr>
</tbody>
</table>
```

<table>
<thead>
<tr>
<th>File</th>
<th><code>sql_logins.txt</code></th>
<th>md5=42AB71BA778EBDD2F89C7587902705C0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EASYACCESS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 rows affected</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Computer Name:</th>
<th>PRODSQL05</th>
<th>Date/Time: 03/02/2007 10:17:49</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Forensic Toolchest (WFT) v1.0.03 (2003.09.20)</td>
<td>Copyright (C) 2003 Monty McDougall</td>
<td>All rights reserved</td>
</tr>
</tbody>
</table>
SQL Server Forensics | Evidence Collection

- **SQLCMD**
  - Load command line tool and establish logging

  ```
  D:\FResponse>Sqlcmd -S RZ-SQL-2005 -e -s"" -E
  1> :out z:\initialconnection.txt
  ```

- **Collecting the active transaction log**
  - Determine on disk locations of the transaction log files

  ```
  D:\FResponse>Sqlcmd -S RZ-SQL-2005 -e -s"" -E
  1> sp_helppdb OnlineSales
  2> go
  ```

Results:

<table>
<thead>
<tr>
<th>name</th>
<th>fileid</th>
<th>filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnlineSales</td>
<td>1</td>
<td>C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\OnlineSales.mdf</td>
</tr>
<tr>
<td>OnlineSales_log</td>
<td>2</td>
<td>C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\OnlineSales_log.ldf</td>
</tr>
<tr>
<td>OnlineSales_log2</td>
<td>3</td>
<td>C:\OtherLogs\OnlineSales_log2.ldf</td>
</tr>
</tbody>
</table>
Collecting the active transaction log (cont’d)

- Gather the VLF allocations

Results:

<table>
<thead>
<tr>
<th>FileId</th>
<th>FileSize</th>
<th>StartOffset</th>
<th>FSeqNo</th>
<th>Status</th>
<th>Parity</th>
<th>CreateLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>14352384</td>
<td>8192</td>
<td>16</td>
<td>2</td>
<td>64</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>14352384</td>
<td>14360576</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>14352384</td>
<td>28712960</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>14606336</td>
<td>43065344</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>14352384</td>
<td>8192</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>14352384</td>
<td>14360576</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>14352384</td>
<td>28712960</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>14606336</td>
<td>43065344</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

2 = Active
0 = Recoverable or unused
Collecting the active transaction log (cont’d)

- `Fn_dblog` filters transactions by:
  - Target database object
  - Specific columns
  - SPID and/or date/time range

  ```
  Select * from ::fn_dblog(NULL, NULL)
  ```

- DBCC Log
  - More resource intensive
  - Dumps transaction log in its entirety

  ```
  dbcc log(<databasename>, 3)
  ```
  - 0 = minimal info
  - 1 = slightly more info
  - 2 = detailed info including (page id, slot id, etc.)
  - 3 = full information about each operation
  - 4 = full information on each operation in addition to hex dump of current data row
Collecting the database plan cache

- Collecting the plan cache
  
  ```sql
  select * from sys.dm_exec_cached_plans cross apply
  sys.dm_exec_sql_text(plan_handle)
  ```

- Collect additional plan cache specifics
  
  ```sql
  select * from sys.dm_exec_query_stats
  ```

Collecting database data files & logs (\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\)

Collecting default trace files and error logs (\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\)

Collecting SQL Server error logs (\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\)

Collecting system event log (WFT)
Evidence Analysis
Windows event log
- SQL Server Windows-based authentication data (failures, successful Log-on/off)
- Server startup and shutdown
- IP addresses of SQL Server client connections

Error log
- Complete authentication history
- Server startup and shutdown
- IP addresses of SQL Server client connections

17-03-02 07:39:10.80 Logon  Login failed for user 'sa'. [CLIENT: 192.168.1.20]
17-03-02 07:39:11.00 Logon  Login failed for user 'sa'. [CLIENT: 192.168.1.20]
17-03-02 07:39:11.20 Logon  Login failed for user 'sa'. [CLIENT: 192.168.1.20]
17-03-02 08:09:37.60 Logon  Login succeeded for user 'EASYACCESS'. Connection: non-trusted. [CLIENT: 192.168.1.20]
SQL Server Forensics | Evidence Analysis

- Default database trace
  - Complete authentication history
  - DDL Operations (Schema changes)
  - IP addresses of SQL Server client connections

<table>
<thead>
<tr>
<th>ssID</th>
<th>ApplicationName</th>
<th>LoginName</th>
<th>SPID</th>
<th>StartTime</th>
<th>EventSubClass</th>
<th>Success</th>
<th>LoginSid</th>
<th>RequestID</th>
<th>EventS</th>
</tr>
</thead>
<tbody>
<tr>
<td>160</td>
<td>sqeu1da 1.0</td>
<td>sa</td>
<td>51</td>
<td>2007-03-02 07:39:11.003</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>sqeu1da 1.0</td>
<td>sa</td>
<td>51</td>
<td>2007-03-02 07:39:11.203</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1300</td>
<td>OSQL-32</td>
<td>sa</td>
<td>51</td>
<td>2007-03-02 07:54:07.180</td>
<td>1 - Add</td>
<td>1</td>
<td>0X01</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1300</td>
<td>OSQL-32</td>
<td>sa</td>
<td>51</td>
<td>2007-03-02 07:54:34.030</td>
<td>1 - Commit</td>
<td>0X01</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1300</td>
<td>OSQL-32</td>
<td>sa</td>
<td>51</td>
<td>2007-03-02 07:54:35.740</td>
<td></td>
<td>0X01</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1300</td>
<td>OSQL-32</td>
<td>sa</td>
<td>51</td>
<td>2007-03-02 07:54:35.903</td>
<td>0 - Begin</td>
<td>0X01</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1300</td>
<td>OSQL-32</td>
<td>sa</td>
<td>51</td>
<td>2007-03-02 07:54:35.913</td>
<td>1 - Commit</td>
<td>0X01</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1300</td>
<td>OSQL-32</td>
<td>sa</td>
<td>51</td>
<td>2007-03-02 07:55:52.783</td>
<td>3 - Grant ...</td>
<td>1</td>
<td>0X01</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1300</td>
<td>OSQL-32</td>
<td>sa</td>
<td>51</td>
<td>2007-03-02 07:56:18.440</td>
<td>1 - Add</td>
<td>1</td>
<td>0X01</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1660</td>
<td>OSQL-32</td>
<td>EASYACCESS</td>
<td>51</td>
<td>2007-03-02 08:09:33.773</td>
<td>1 - Commit</td>
<td>0XBB9...</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1660</td>
<td>OSQL-32</td>
<td>EASYACCESS</td>
<td>51</td>
<td>2007-03-02 08:13:29.350</td>
<td>1 - Increase</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1660</td>
<td>OSQL-32</td>
<td>EASYACCESS</td>
<td>51</td>
<td>2007-03-02 08:13:31.433</td>
<td>1 - Commit</td>
<td>0XBB9...</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1660</td>
<td>OSQL-32</td>
<td>EASYACCESS</td>
<td>51</td>
<td>2007-03-02 08:13:32.667</td>
<td>1 - Commit</td>
<td>0XBB9...</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- **Data files & Log files**
  - Attach files
  - Use to obtain on-demand schema info, data page contents, etc.

- **Active transaction log**
  - Import into Excel / Access for viewing
  - Identify DML & DDL statements
  - Map transactions to a SPID
Transaction Log - Update operations

- Marks the beginning of a transaction
- Unique transaction identifier
- Data page identifier for row containing the updated record
- Marks the end of a transaction
- Type of transaction performed
- On data page row location of record
- In row data offset of modification

<table>
<thead>
<tr>
<th>Operation</th>
<th>Context</th>
<th>Transaction ID</th>
<th>Page ID</th>
<th>Slot ID</th>
<th>...</th>
<th>Offset in Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOP BEGIN XACT</td>
<td>LCX_NULL</td>
<td>0000:0000032e</td>
<td>NULL</td>
<td>NULL</td>
<td>...</td>
<td>NULL</td>
</tr>
<tr>
<td>LOP MODIFY_COLUMNS</td>
<td>LCX_CLUSTED</td>
<td>0000:0000032e</td>
<td>0001:00000d3</td>
<td>20 ...</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>LOP MODIFY_COLUMNS</td>
<td>LCX_CLUSTED</td>
<td>0000:0000032e</td>
<td>0001:0000013c</td>
<td>13 ...</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>LOP MODIFY_COLUMNS</td>
<td>LCX_CLUSTED</td>
<td>0000:0000032e</td>
<td>0001:00000d6</td>
<td>7 ...</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>LOP COMMIT XACT</td>
<td>LCX_NULL</td>
<td>0000:0000032e</td>
<td>NULL</td>
<td>NULL</td>
<td>...</td>
<td>NULL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Hex</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction ID</td>
<td>0000:0000032e</td>
<td>0.814</td>
</tr>
<tr>
<td>Data Page</td>
<td>0001:0000000d3</td>
<td>1.211</td>
</tr>
</tbody>
</table>
DBCC Page will pull up the modified data page

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

Viewing the page header will detect the owning object

```
Page 80x04304000
m_pageId = (1:211)
m_typeFlagBits = 0x0
m_objId (AllocUnitId.idObj) = 87
m_headerVersion = 1
m_level = 0
m_indexId (AllocUnitId.idInd) = 256
Metadata: AllocUnitId = 72057594043629568
Metadata: AllocUnitId = 720575940439500800
Metadata: ObjectID = 629577281
m_prevPage = (1:314)
m_slotCnt = 22
m_reservedCnt = 0
m_xactReserved = 0
m_tornBits = -1731484635
```

Lookup the owning object: `Select * from sysobjects where id = 629577281`

Results:

```
<table>
<thead>
<tr>
<th>name</th>
<th>id</th>
<th>xtype</th>
<th>uid</th>
<th>info</th>
<th>status</th>
<th>base_schema_ver</th>
<th>replinfo</th>
<th>parent_obj</th>
<th>crdate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order</td>
<td>629577281</td>
<td>U</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2007-02-26 16:08:21.320</td>
</tr>
</tbody>
</table>
```
Gather the object schema

“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”

Results:

<table>
<thead>
<tr>
<th>colorder</th>
<th>name</th>
<th>datatype</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OrderID</td>
<td>int</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>FirstName</td>
<td>varchar</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>LastName</td>
<td>varchar</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>Address</td>
<td>varchar</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td>City</td>
<td>nchar</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>State</td>
<td>nchar</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>ZIP</td>
<td>nchar</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>CCType</td>
<td>varchar</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>CCNumber</td>
<td>varchar</td>
<td>20</td>
</tr>
<tr>
<td>11</td>
<td>ShipStatusID</td>
<td>int</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>OrderDate</td>
<td>datetime</td>
<td>8</td>
</tr>
<tr>
<td>13</td>
<td>Product</td>
<td>nvarchar</td>
<td>100</td>
</tr>
<tr>
<td>14</td>
<td>Price</td>
<td>nchar</td>
<td>30</td>
</tr>
</tbody>
</table>
Viewing data page 1:211 modified using Slot 20 & Row offset 80
SQL Server Forensics | Evidence Analysis

- Price column pre and post transaction modification

<table>
<thead>
<tr>
<th></th>
<th>RowLog Contents 0</th>
<th>RowLog Contents 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0x3500300030002E003000030</td>
<td>0x2E00350030002000200020</td>
</tr>
</tbody>
</table>

- Price column pre and post transaction modification

RowLog0

<table>
<thead>
<tr>
<th>Hex</th>
<th>35</th>
<th>00</th>
<th>30</th>
<th>00</th>
<th>30</th>
<th>00</th>
<th>2E</th>
<th>00</th>
<th>30</th>
<th>00</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RowLog1

<table>
<thead>
<tr>
<th>Hex</th>
<th>2E</th>
<th>00</th>
<th>35</th>
<th>00</th>
<th>30</th>
<th>00</th>
<th>20</th>
<th>00</th>
<th>20</th>
<th>00</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII</td>
<td>.</td>
<td>5</td>
<td>0</td>
<td>SP</td>
<td>SP</td>
<td>SP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 1st record affected by transaction 814 had the price column updated from “3500.00” to “3.50” Including leading byte “33”
Transaction Log - Insert Operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Context</th>
<th>Transaction ID</th>
<th>Page ID</th>
<th>Slot ID</th>
<th>...</th>
<th>Offset in Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOP_BEGIN_XACT</td>
<td>LCX_NULL</td>
<td>0000:00000330</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td>NULL</td>
</tr>
<tr>
<td>LOP_INSERT_ROWS</td>
<td>LCX_CLUSTERED</td>
<td>0000:00000330</td>
<td>0001:00000138</td>
<td>8</td>
<td></td>
<td>NULL</td>
</tr>
<tr>
<td>LOP_COMMIT_XACT</td>
<td>LCX_NULL</td>
<td>0000:00000330</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td>NULL</td>
</tr>
</tbody>
</table>

Reconstruct the data row

RowLog Contents 0:

“0x30006C00A1010000530072069006E0067004C0061006B00650020002000200 020002000200020002000200041005A0031003400340031003000100000000000 00E49800003400200300030002002000200020002000200020002000200020002000 0E000C02008200870098009C00AC00BC04E6966F426C61636B3732205374617266 656C6C20447269766556697361353531333030303030303058004200F0058002000 330036003000”
- Lookup the schema and reconstruct the data row
- Structure of a variable length data row:

![Diagram showing the structure of a variable length data row](image)

Legend:

<table>
<thead>
<tr>
<th>Item</th>
<th>Storage Allocation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 byte</td>
<td>StatusBits A contains data row properties</td>
</tr>
<tr>
<td>2</td>
<td>1 byte</td>
<td>Unused in SQL Server 2005</td>
</tr>
<tr>
<td>3</td>
<td>2 bytes</td>
<td>Row offset to in row location containing the number of columns in the data row</td>
</tr>
<tr>
<td>4</td>
<td>2 bytes</td>
<td>Total number of columns in data row</td>
</tr>
<tr>
<td>5</td>
<td>1 bit for each row column</td>
<td>Null Bitmap</td>
</tr>
<tr>
<td>6</td>
<td>2 bytes</td>
<td>Number of variable length columns within data row</td>
</tr>
<tr>
<td>7</td>
<td>2 bytes for each variable length column</td>
<td>Row offset marking the end of each variable length column</td>
</tr>
<tr>
<td></td>
<td>Variable length columns</td>
<td>Used length of all variable length columns</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Location of in row variable length data columns</td>
</tr>
</tbody>
</table>

Source: Inside SQL Server 2005 The Storage Engine
Swap the bytes (endian ordering)
Translate data types
The inserted record was:

- OrderID: 4122
- FirstName: Nino
- LastName: Black
- Address: 72 Starfell Drive
- City: SpringLake
- State: AZ
- ZIP: 14410
- CCType: Visa
- CCNumber: 5518530000000000
- ShipStatusID: 1
- OrderDate: March 1st, 2007
- Product: XBOX 360
- Price: 4.00
SQL Server Forensics | Evidence Analysis

- **Transaction Log - Delete operations**
- **Ghost records**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Context</th>
<th>Transaction ID</th>
<th>Page ID</th>
<th>Slot ID</th>
<th>...</th>
<th>Offset in Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOP_BEGIN_XACT</td>
<td>LCX_NULL</td>
<td>0000:0000032</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOP_DELETE_ROWS</td>
<td>LCX_MARK_AS_GHOST</td>
<td>0000:0000032</td>
<td>0001:00000158</td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOP_SET_BITS</td>
<td>LCX_PFS</td>
<td>0000:00000000</td>
<td>0001:0000001</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOP_COMMIT XACT</td>
<td>LCX NULL</td>
<td>0000:0000032</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RowLog Contents 0:

```
"0x30006C009F00000050006100790065007400740065002000200020002000200020002000200020002000200002002000200020002000200046004C003100360036003000320002000200000000003A98000033003500300030002E0030003000200020002000200020002000200020000000E0000C006008200860098009C00AD00CD004275727443617665323237205374617267656C6C204472697665566973613635393033343030333433323233323030566F6C63616E6F203620696E636820506C61736D6120564332333332"
```

- **Reconstruct the data row**
Swap the bytes (endian ordering)

Translate data types

The deleted record was:
- OrderID: 159
- FirstName: Burt
- LastName: Cave
- Address: 227 Stargell Drive
- City: Payette
- State: FL
- ZIP: 16602
- CCType: Visa
- CCNumber: 65903400343223200
- ShipStatusID: 1
- OrderDate: September 12th, 2006
- Product: Volcano 62 inch Plasma TV VC2332
- Price: 3500.00
## Plan cache

- Review for applicable statements within scope of investigation (date, objects, etc.)
- Look for **non-standard** statements

<table>
<thead>
<tr>
<th>bucketid</th>
<th>refcounts</th>
<th>usecounts</th>
<th>cacheobjtype</th>
<th>obtype</th>
<th>plan_data</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td>6514</td>
<td>2</td>
<td>1</td>
<td>Compiled Plan</td>
<td>Adhoc</td>
<td>0x06000...</td>
<td>SELECT CAST(tli.is_enabled AS bit) AS [IsEnabled], OBJECTPROPERTY(tli.object_id,'TableFullT...</td>
</tr>
<tr>
<td>3485</td>
<td>2</td>
<td>1</td>
<td>Compiled Plan</td>
<td>Adhoc</td>
<td>0x06000...</td>
<td>SELECT col.name AS [Name] FROM sys.tables AS tbl INNER JOIN sys.fulltext_indexes AS fti ON ...</td>
</tr>
<tr>
<td>942</td>
<td>2</td>
<td>1</td>
<td>Compiled Plan</td>
<td>Adhoc</td>
<td>0x06000...</td>
<td>SELECT dtb.is_fulltext_enabled AS [IsFullTextEnabled] FROM master.sys.databases AS dtb WHERE...</td>
</tr>
<tr>
<td>6312</td>
<td>2</td>
<td>1</td>
<td>Compiled Plan</td>
<td>Adhoc</td>
<td>0x06000...</td>
<td>select CCNumber, Firstname, Lastname from OrderHistory where OrderID = 1967</td>
</tr>
<tr>
<td>551</td>
<td>2</td>
<td>1</td>
<td>Compiled Plan</td>
<td>Adhoc</td>
<td>0x06000...</td>
<td><strong>select * from OrderHistory</strong></td>
</tr>
<tr>
<td>8567</td>
<td>2</td>
<td>4</td>
<td>Compiled Plan</td>
<td>Adhoc</td>
<td>0x06000...</td>
<td>select * from sys.dm_exec_cached_plans cross apply sys.dm_exec_sql_text(plan_handle)</td>
</tr>
<tr>
<td>944</td>
<td>2</td>
<td>1</td>
<td>Compiled Plan</td>
<td>Adhoc</td>
<td>0x06000...</td>
<td>select CCNumber, Firstname, Lastname from OrderHistory where OrderID = 1</td>
</tr>
<tr>
<td>9441</td>
<td>2</td>
<td>1</td>
<td>Compiled Plan</td>
<td>Adhoc</td>
<td>0x06000...</td>
<td>select CCNumber, Firstname, Lastname from OrderHistory where OrderID = 5</td>
</tr>
<tr>
<td>8278</td>
<td>2</td>
<td>1</td>
<td>Compiled Plan</td>
<td>Adhoc</td>
<td>0x06000...</td>
<td>select CCNumber, Firstname, Lastname from OrderHistory where OrderID = 22</td>
</tr>
<tr>
<td>6313</td>
<td>2</td>
<td>1</td>
<td>Compiled Plan</td>
<td>Adhoc</td>
<td>0x06000...</td>
<td>select CCNumber, Firstname, Lastname from OrderHistory where OrderID = 1823</td>
</tr>
<tr>
<td>3005</td>
<td>2</td>
<td>1</td>
<td>Compiled Plan</td>
<td>Adhoc</td>
<td>0x06000...</td>
<td>select CCNumber, Firstname, Lastname from OrderHistory where OrderID = 1639</td>
</tr>
</tbody>
</table>
Investigation Pitfalls
What to look out for!

- Know the schema you're working with
- Data type storage formats
- Reduce large data sets
- Correlate on-disk values with transaction log data
- Encryption
- This takes time so be patient!
Conclusion
Conclusion

- Don’t ignore the database when conducting computer forensics investigations
- Database forensics techniques learned today can augment traditional forensics skills to uncover the evidence needed to support your case

Additional information within the presentation white paper

- Real world database forensics scenario
- Database forensics methodology
- Additional evidence collection and analysis techniques
  - Code pages and collation settings
  - Obtaining server configuration
  - Identifying user account creation and elevation of privilege activity
  - Transaction log data carving
  - And more…
Questions
Presentation References