

# SQL Server Forensics 2

Presented at AppSec Asia

November 18, 2009

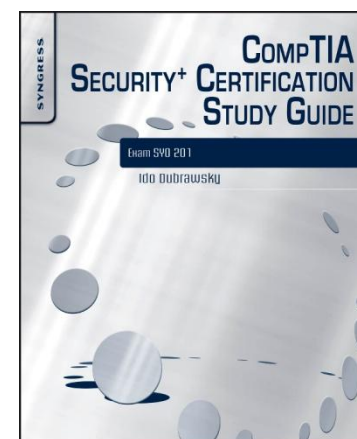
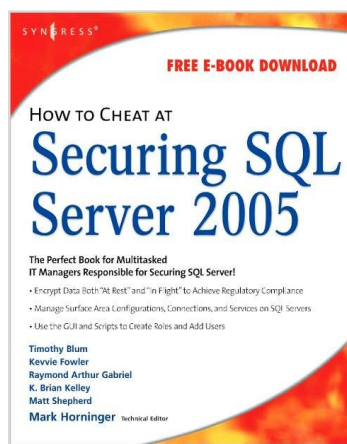
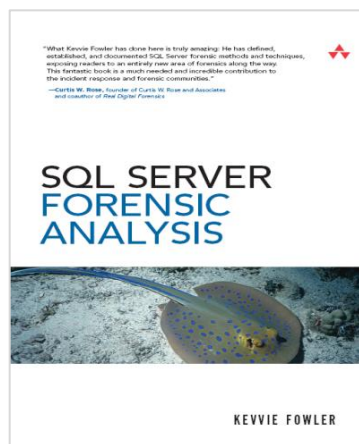


**Kevvie Fowler**, GCFA Gold, CISSP,  
MCTS, MCDBA, MCSD, MCSE



# About me

- ❖ Day job: Director Security Services, TELUS | backed by Emergis
- ❖ Night job: Security researcher, Ringzero
- ❖ Security industry contributions:



# What is database forensics?

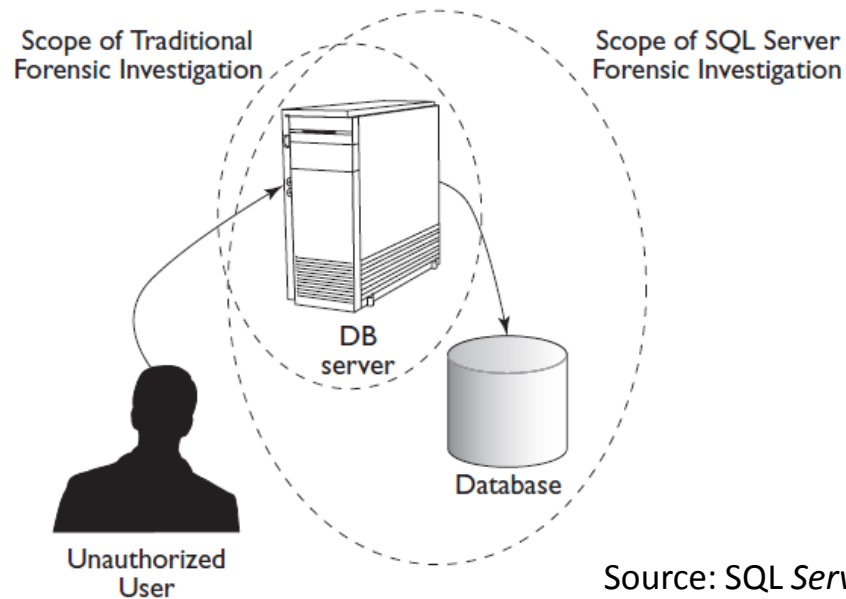
## ❖ Database forensics:

- The application of computer investigation and analysis techniques to gather database evidence suitable for presentation in a court of law
  - Retrace user DML & DDL operations
  - Identify data pre and post transaction
  - For the “real world”: No dependency on 3<sup>rd</sup> party auditing tools or pre-configured DML or DDL triggers

# What is database forensics? (2)

## ❖ Why do we need it?

- Typical investigations have little focus on the database



Source: *SQL Server Forensic Analysis*, Kevvie Fowler

## ❖ The right approach taken by a major AV vendor this year

# What is database forensics? (3)

- ❖ Aren't databases stored in the file system therefore part of a typical investigation?
  - 33 different data types many with differences in the way they are stored and retrieved (example value: 21976)

Data type	On disk value	Retreived value
CHAR	3231393736	21976
INT	D855	21976
DATETIME	D855	3/3/1960

- Random data without understanding the context is just random data:  
0x300004000100FE01004B0049006E007400650072006600610063006  
5003A0020003100390032002E003100360038002E0031002E0031003  
700360020002D002D002D002000300078003800

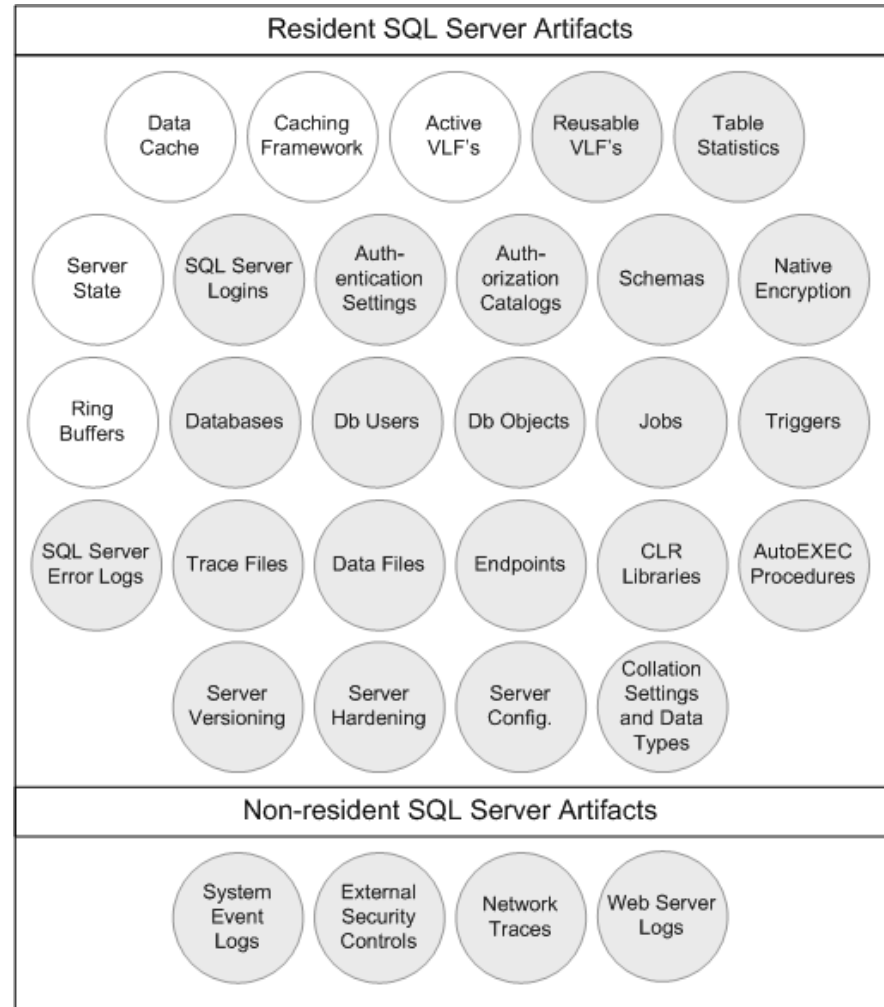
# Database artifacts

## ❖ 33 defined artifacts

- Volatile
- Non-volatile

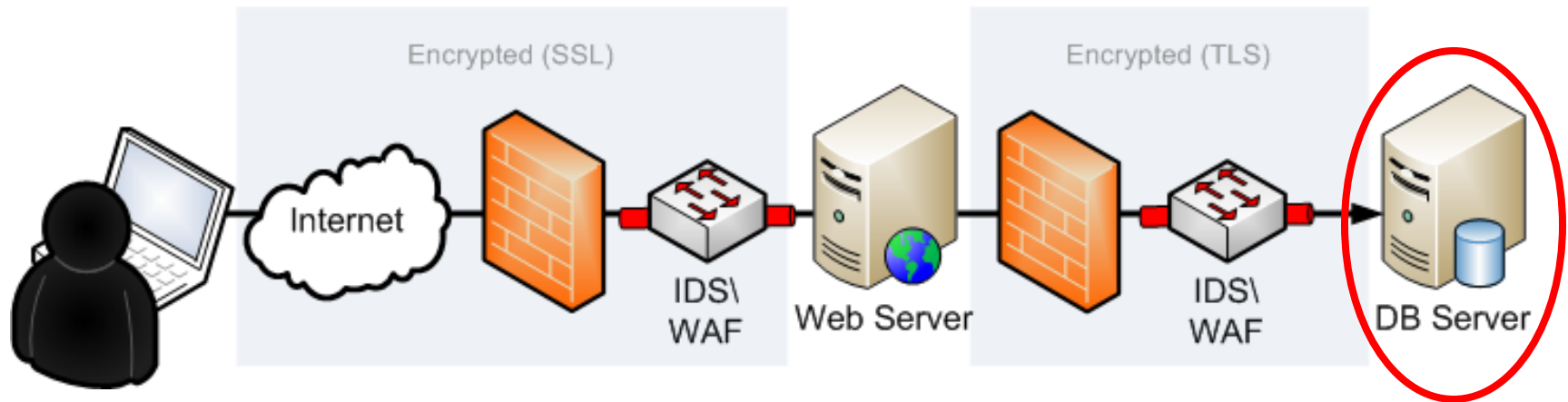
## ❖ Artifacts contain traces of a past database attack

Source: *SQL Server Forensic Analysis*,  
Kevvie Fowler



# A typical database attack

- ❖ Credit card data stolen via a SQLi attack
  - Pangolin exploitation video





# Incident Verification

# Verifying if an attack has occurred

## ❖ WFTSQL (wftsql <instance> <credentials>)

```
C:\Windows\system32\cmd.exe - wftsql
=====
wftSQL.bat v1.0.03
Copyright (C) 2007 Monty McDougal
Released under the terms of the GNU General Public License
=====
Usage: wftSQL.bat [SQLserver]
Usage: wftSQL.bat [SQLserver] [SQLuser] [SQLpass]
Enter target SQL Server Instance:

C:\Windows\system32\cmd.exe - wftsql
=====
Windows
Forensic
Toolchest™
(WFT) v3.0.01
=====
Copyright (C) 2003-2007 Monty McDougal. All rights reserved.
=====

C:\SQLCMD
06:40:19: Running 'sql\runsq1.bat' [#2/150]
      <md5=03049C3782A66064BACC95B614850643>
      COMPLETE
      'DBServerInfo.txt'
      <md5=07BAF0D870CCC5FFDD2CA01CD84BA626>
      'DBServerInfo.htm'
      <md5=E704FE2858BF9C99955BEA3D47A04189>

06:40:20: Verifying 'sql\SSFA_Configurations.sql' OK
      <md5=BD648BB0A9ECB9AB919BAB556855C7F0>
06:40:20: Verifying 'sql\runsq1.bat' OK
      <md5=03049C3782A66064BACC95B614850643>
06:40:20: Running 'sql\runsq1.bat' [#3/150]
      COMPLETE
      'Configuration.txt'
      <md5=50ED62A2FA7DDA8ED4E88289C7CF971A>
      'Configuration.htm'
      <md5=D2C372CAB63F5C49189753DA14F1F952>

06:40:22: Verifying 'sql\SSFA_RecentStatements.sql' OK
      <md5=D06315F6996BC797D0B8F28CDC850D8E>
06:40:22: Verifying 'sql\runsq1.bat' OK
      <md5=03049C3782A66064BACC95B614850643>
06:40:22: Running 'sql\runsq1.bat' [#4/150]
```

# Verifying if an attack has occurred (2)

## ❖ WFTSQL results

The screenshot displays the Windows Forensic Toolchest (WFT) interface. The top navigation bar includes 'Main', 'Log', 'Config', 'File Hashes', 'Tools', and 'Security Resources'. The left sidebar contains a tree view with categories like 'START', 'SQL SERVER', 'DB Objects & Users', and 'DB Configuration'. The main window shows the 'Plan Cache' tool results for a command executed on 'KEVVIE-PC'.

**Command** ✓ (md5=FEDD1E9DF1B76C877B8027B62E401B50)  
tools\sql\runcsql.bat -iSSFA\_PlanCache.sql > KEVVIE-PC\2008\_08\_16\09\_45\_19\txt\PlanCache.txt

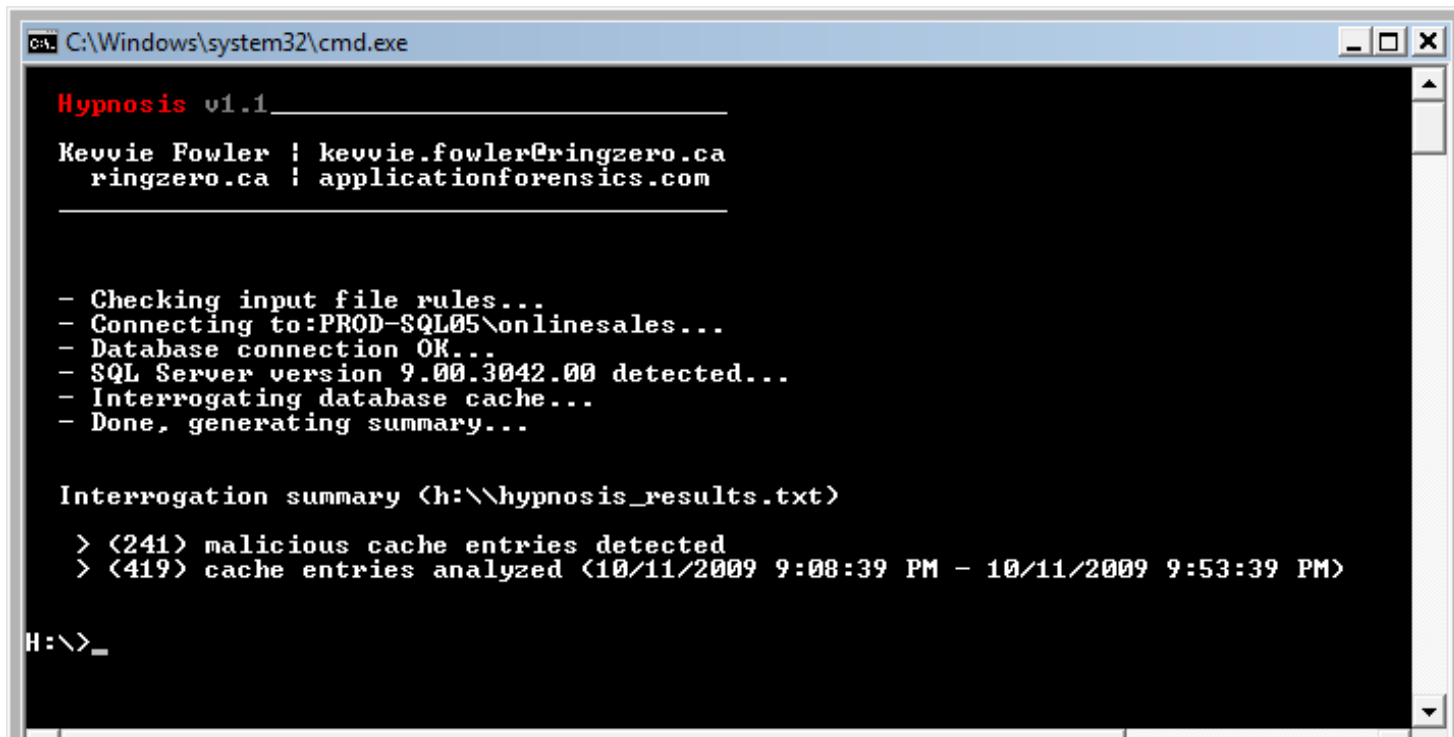
**Description**  
**Plan cache entries** For assistance in analyzing this data please go to: [www.applicationforensics.com/sql](http://www.applicationforensics.com/sql)

SSFA\_PlanCache.sql -- List of cached database plans organized by database and last\_execution\_time in descending order.

File	PlanCache.txt (md5=DC1B8B9DE2CB3E479EEFE0F41CAC2D3F)
FROM ORDERS WHERE FirstName = ''	UNION ALL SELECT 9, name, '', createdate, 't', 't', 't', 't', 't', 't', 't', 't' FROM
FROM ORDERS WHERE FirstName = ''	UNION ALL SELECT 9, name, 'text', 'text', 'text', 'text', 'text', 'text', 'te
FROM ORDERS WHERE FirstName = ''	or 'a'='a'
FROM ORDERS WHERE FirstName = ''	or 1=1 or ''=''
FROM ORDERS WHERE FirstName = ''	and ''=''
FROM ORDERS WHERE FirstName = ''	or ''=''
FROM ORDERS WHERE FirstName = ''	UNION ALL SELECT 1, name, 0, 0, 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A' FROM sysobje
FROM ORDERS WHERE FirstName = ''	UNION ALL SELECT 1, name, 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 0 FROM sysobj
FROM ORDERS WHERE FirstName = ''	SELECT 1, name, 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 0 FROM sysobjects WHERE
FROM ORDERS WHERE FirstName = ''	UNION ALL SELECT 1, name, name, 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 0 FROM syso
FROM ORDERS WHERE FirstName = ''	UNION ALL SELECT 1, name, 0, 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 0 FROM sysobje
FROM ORDERS WHERE FirstName = ''	UNION ALL SELECT 1, xtype, 0, 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 0 FROM sysobj
FROM ORDERS WHERE FirstName = ''	UNION ALL SELECT 1, 0, 0, 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 0 FROM sysobje
FROM ORDERS WHERE FirstName = ''	UNION ALL SELECT 1, name, xtype, 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 0 FROM sys
FROM ORDERS WHERE FirstName = ''	UNION ALL SELECT 9. name, xtype, 'text', 'text', 'text', 'text', 'text', 'te

# Verifying if an attack has occurred (3)

## ❖ Hypnosis



```
C:\Windows\system32\cmd.exe

Hypnosis v1.1
-----
Kevvie Fowler | kevvie.fowler@ringzero.ca
ringzero.ca | applicationforensics.com
-----

- Checking input file rules...
- Connecting to:PROD-SQL05\onlinesales...
- Database connection OK...
- SQL Server version 9.00.3042.00 detected...
- Interrogating database cache...
- Done, generating summary...

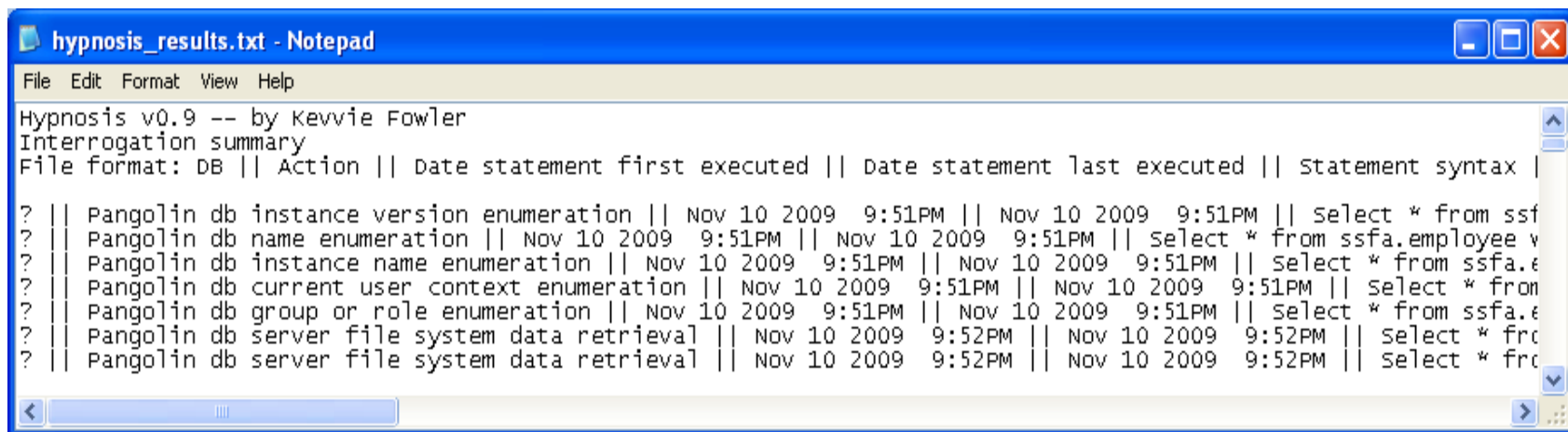
Interrogation summary (h:\\hypnosis_results.txt)
  > (241) malicious cache entries detected
  > (419) cache entries analyzed (10/11/2009 9:08:39 PM - 10/11/2009 9:53:39 PM)

H:\>_
```

# Verifying if an attack has occurred (4)

## ❖ Hypnosis results file (sample entry)

```
? || Pangolin db instance version enumeration || Nov 10 2009 9:51PM || Nov 10 2009 9:51PM || Select * from ssfa.employee where fname='Mikaela' and 1=2 union all select null,char(94)+char(94)+char(94)+cast(@@version as nvarchar(4000))+char(94)+char(94)+char(94),null,null,null --' || 1
```



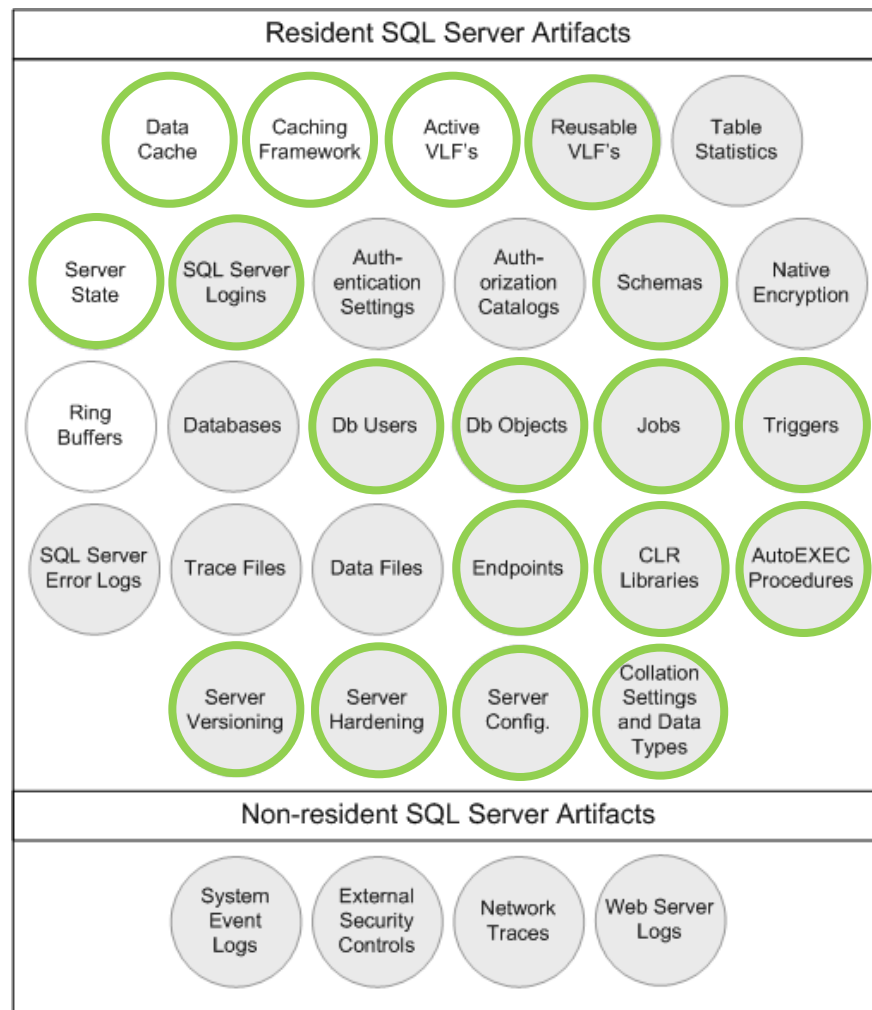
```
hypnosis_results.txt - Notepad
File Edit Format View Help
Hypnosis v0.9 -- by Kevvie Fowler
Interrogation summary
File format: DB || Action || Date statement first executed || Date statement last executed || Statement syntax ||
? || Pangolin db instance version enumeration || Nov 10 2009 9:51PM || Nov 10 2009 9:51PM || select * from ssfa.employee where fname='Mikaela' and 1=2 union all select null,char(94)+char(94)+char(94)+cast(@@version as nvarchar(4000))+char(94)+char(94)+char(94),null,null,null --' || 1
? || Pangolin db name enumeration || Nov 10 2009 9:51PM || Nov 10 2009 9:51PM || select * from ssfa.employee v
? || Pangolin db instance name enumeration || Nov 10 2009 9:51PM || Nov 10 2009 9:51PM || select * from ssfa.e
? || Pangolin db current user context enumeration || Nov 10 2009 9:51PM || Nov 10 2009 9:51PM || select * from
? || Pangolin db group or role enumeration || Nov 10 2009 9:51PM || Nov 10 2009 9:51PM || select * from ssfa.e
? || Pangolin db server file system data retrieval || Nov 10 2009 9:52PM || Nov 10 2009 9:52PM || select * fro
? || Pangolin db server file system data retrieval || Nov 10 2009 9:52PM || Nov 10 2009 9:52PM || select * fro
```



# Artifact Collection

# SQL Server Artifact collection

- ❖ WFTSQL has already preserved 18 artifacts
- ❖ Collect other applicable artifacts



Source: *Server Forensic Analysis*,  
Kevvie Fowler



# Artifact Analysis

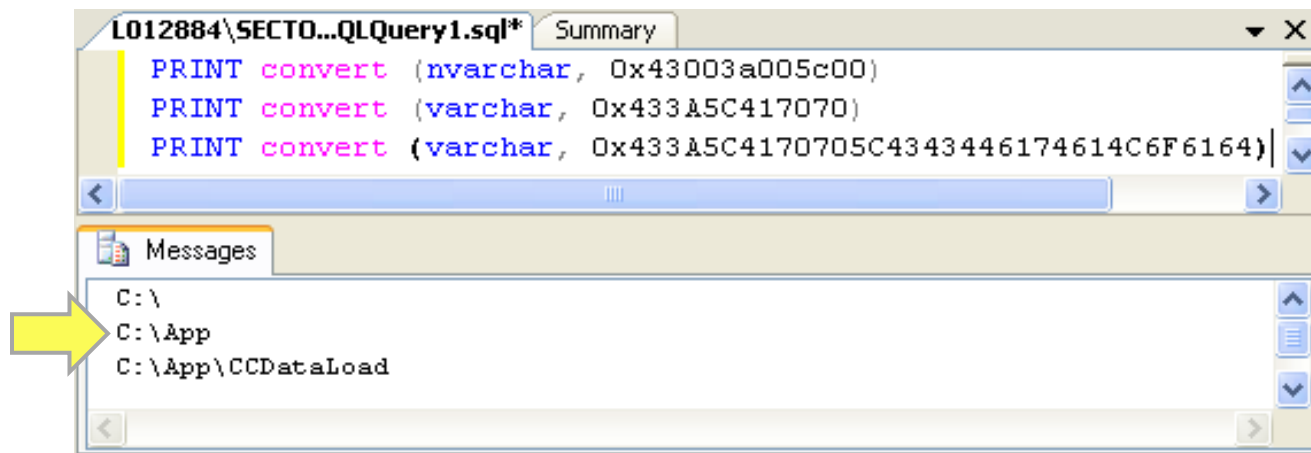


# Analyzing collected artifacts (2)

## ❖ Plan Cache (continued)

- Sample finding from Hypnosis:

```
? || Pangolin db server file system data retrieval || Oct 3 2009 9:23 PM ||  
Oct 3 2009 9:23PM || Select * from ssfa.employee where status = 1 and  
fname='Mikaela' ;declare @z nvarchar(4000) set @z=0x43003a005c00 insert  
pangolin_test_table execute master..xp_dirtree @z,1,1--' || 1
```



```
LO12884\SECTO...QLQuery1.sql* Summary  
PRINT convert (nvarchar, 0x43003a005c00)  
PRINT convert (varchar, 0x433A5C417070)  
PRINT convert (varchar, 0x433A5C4170705C4343446174614C6F6164)  
Messages  
C:\  
C:\App  
C:\App\CCDataLoad
```

# Analyzing collected artifacts <sup>(3)</sup>

## ❖ Transaction log

- DML statements
- DDL statements
- Login failures...

## ❖ Walk through of reconstructing an INSERT operation

# Analyzing collected artifacts (4)

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. Num Elements
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

# Analyzing collected artifacts (5)

## ❖ INSERTED data row (Rowlog contents 0)

```
0x300004000100FE01004B0049006E007400650072006600610063006500  
3A0020003100390032002E003100360038002E0031002E0031003700360  
020002D002D002D002000300078003800
```

## ❖ “Before you can analyze data you need to first understand the format in which it was written”

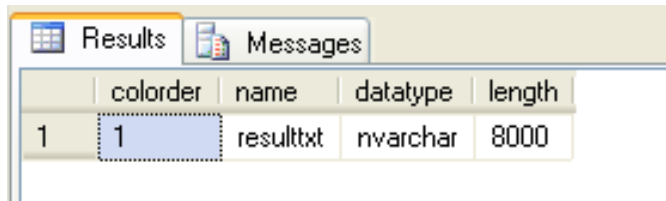
Kevvie Fowler quoting himself...

# Analyzing collected artifacts (6)

## ❖ Understanding the data row structure

- Syscolumns (existing object)

```
SELECT sc.colorder, sc.name, st.name as 'datatype', sc.length FROM syscolumns  
sc, systypes st WHERE sc.xusertype = st.xusertype and sc.id =  
Object_ID('Object_Name') ORDER BY colorder
```



colorder	name	datatype	length
1	1	resulttxt	nvarchar 8000

- Plan cache (existing or deleted object)

```
Select * from ssfa.employee where fname='Mikaela' ;create table  
[pangolin_test_table]([resulttxt] nvarchar(8000) null);--'
```

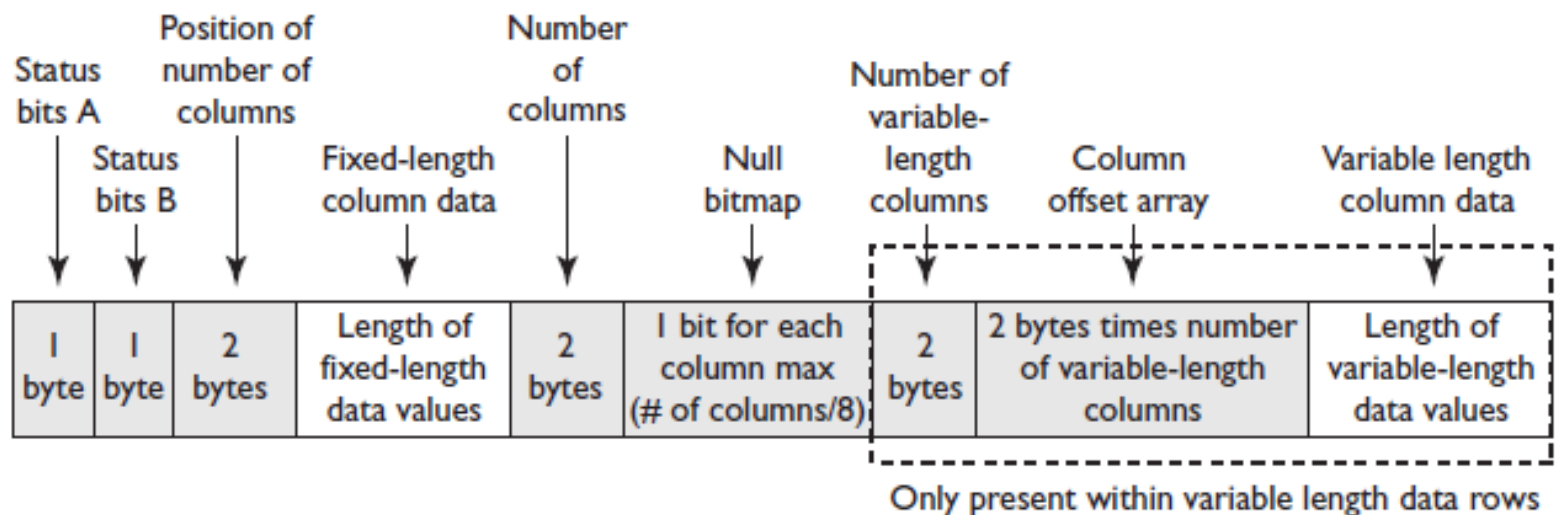
- Transaction log (existing or deleted object)

```
0x30002D0064EFCA5D000001000000E7E7000000401F000008D0000002000000401F00000  
000000000000000000000100000800100470072006500730075006C007400740078007400
```



# Analyzing collected artifacts (7)

## ❖ Understanding the data row structure (continued)



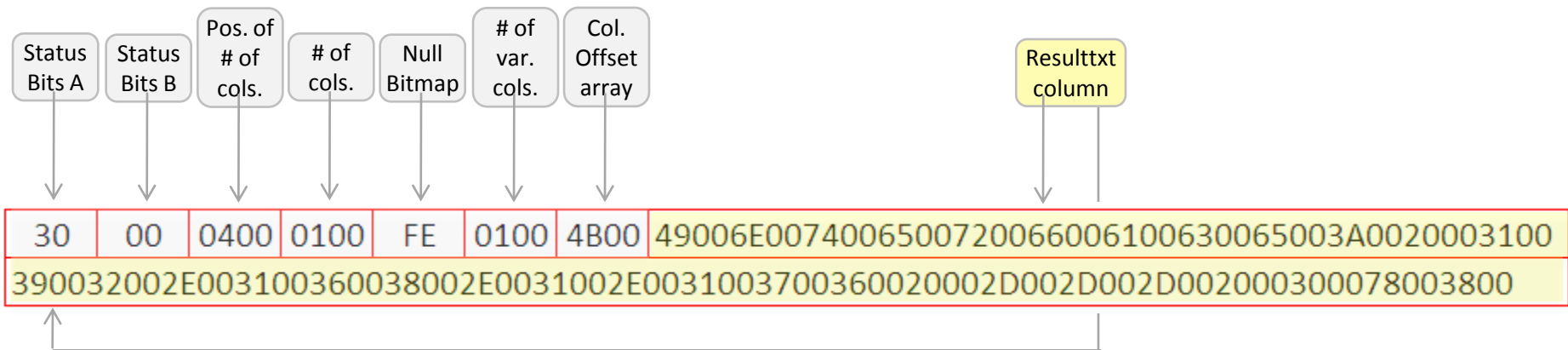
Remember the row structure?

Source: *Server Forensic Analysis*,  
Kevvie Fowler



# Analyzing collected artifacts (8)

## ❖ Applying the structure to our data row



## ❖ Reconstructing the insert operation

```
PROD-SQL05\ONLL.. SQLQuery9.sql* | PROD-SQL05\ONLL... SQLQuery8.sql* | ADMIN:PROD-SQL0... SQLQuery4.sql*
```

```
PRINT convert (nvarchar(max), 0x49006E0074006500720066006100630065003A0020003100390032002E00310036...  
0038002E0031002E0031003700360020002D002D002D002000300078003800)
```

Results | Messages

(No column name)

Interface: 192.168.1.176 --- 0x8

# Analyzing collected artifacts <sup>(9)</sup>

## ❖ 5 sample insert operations retrieved

- 1) 0x300004000100FE01004B0049006E0074006500720066006100630065003A0020003100390032002E003100360038002E0031002E0031003700360020002D002D002D002000300078003800
- 2) 0x300004000100FE01006F002000200049006E007400650072006E00650074002000410064006400720065007300730020002000200020002000200050006800790073006900630061006C00200041006400640072006500730073002000200020002000200020005400790070006500
- 3) 0x300004000100FE01007B00200020003100390032002E003100360038002E0031002E00320020002000200020002000200020002000200020002000200020002000300030002D00300030002D00320031002D00640030002D00620034002D0033003700200020002000200020002000640079006E0061006D0069006300200020002000
- 4) 0x300004000100FE01007B00200020003100390032002E003100360038002E0031002E00330020002000200020002000200020002000200020002000200020002000300030002D00360030002D00620030002D00650065002D00320034002D0063006500200020002000200020002000640079006E0061006D0069006300200020002000
- 5) 0x300004000100FE01007B00200020003100390032002E003100360038002E0031002E00320030002000200020002000200020002000200020002000200020002000300030002D00360030002D00620030002D00620034002D00640035002D0035003700200020002000200020002000640079006E0061006D0069006300200020002000

# Analyzing collected artifacts <sup>(10)</sup>

- ❖ Reconstruction of data rows inserted into the table and extracted line-by-line by the attacker:

(1) Interface: 192.168.1.176 --- 0x8

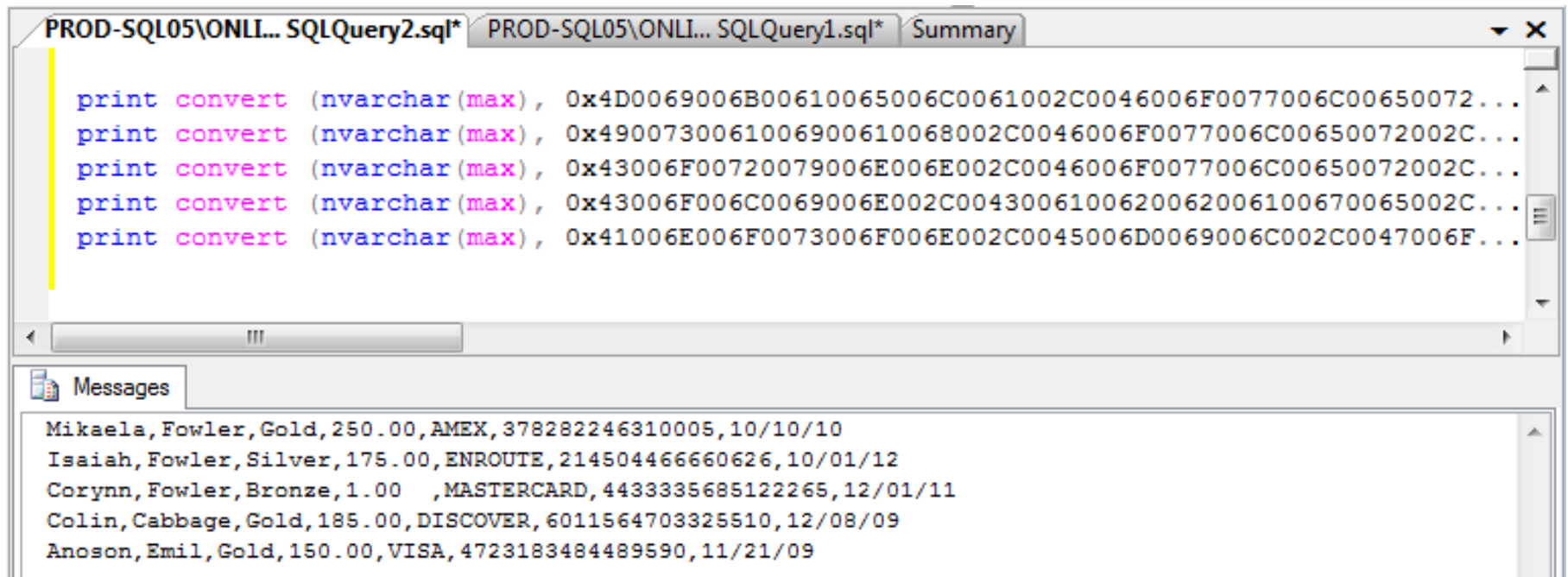
(2)	Internet Address	Physical Address	Type
(3)	192.168.1.2	00-00-21-d0-b4-37	dynamic
(4)	192.168.1.3	00-60-b0-ee-24-ce	dynamic
(5)	192.168.1.20	00-60-b0-b4-d5-57	dynamic

...

Can you tell what command was executed during the attack ?

# Analyzing collected artifacts (10)

- ❖ Using the same process allows us to reconstruct the credit card data viewed by the attacker:



```
PROD-SQL05\ONLI... SQLQuery2.sql*  PROD-SQL05\ONLI... SQLQuery1.sql*  Summary
```

```
print convert (nvarchar(max), 0x4D0069006B00610065006C0061002C0046006F0077006C00650072...
print convert (nvarchar(max), 0x4900730061006900610068002C0046006F0077006C00650072002C...
print convert (nvarchar(max), 0x43006F00720079006E006E002C0046006F0077006C00650072002C...
print convert (nvarchar(max), 0x43006F006C0069006E002C0043006100620062006100670065002C...
print convert (nvarchar(max), 0x41006E006F0073006F006E002C0045006D0069006C002C0047006F...
```

Messages

```
Mikaela, Fowler, Gold, 250.00, AMEX, 378282246310005, 10/10/10
Isaiah, Fowler, Silver, 175.00, ENROUTE, 214504466660626, 10/01/12
Corynn, Fowler, Bronze, 1.00, MASTERCARD, 4433335685122265, 12/01/11
Colin, Cabbage, Gold, 185.00, DISCOVER, 6011564703325510, 12/08/09
Anoson, Emil, Gold, 150.00, VISA, 4723183484489590, 11/21/09
```



# Antiforensics

# The Anti-antiforensics

- ❖ SQL Server antiforensics research released at Black Hat DC, 2009
  
- ❖ Traditional forensic practices can overcome some of the published antiforensics techniques
  
- ❖ Top 3 points for discussion
  1. Fill transaction logs with CHECKPOINT entries to overwrite DML\DDL data
    - Active transactions and replication is not factored in

# The Anti-antiforensics (2)

2. Using `sp_cycle_errorlog` to overwrite SQL Server errorlog
  - SQL Server doesn't securely delete overwritten error logs which can be easily recovered
  - Number of historical error logs are user configurable
  
3. Filling default trace files with 20MB of data to force trace rotation
  - SQL Server doesn't securely delete overwritten traces which can be easily recovered



# 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 additional evidence needed to support your case
- ❖ Additional info on SQL Server Forensics can be found on [www.applicationforensics.com](http://www.applicationforensics.com)
  - SQL Server Forensics (1.0)
  - Double Trouble - Database rootkits and native encryption flaws
  - To Cache a Thief – Cache based SQL injection attack detection
  - Database incident response tools WFTSQL and Hypnosis

# Thank-you | additional information

Thank-you! - additional information and questions:

[kevvie.fowler@ringzero.ca](mailto:kevvie.fowler@ringzero.ca)

