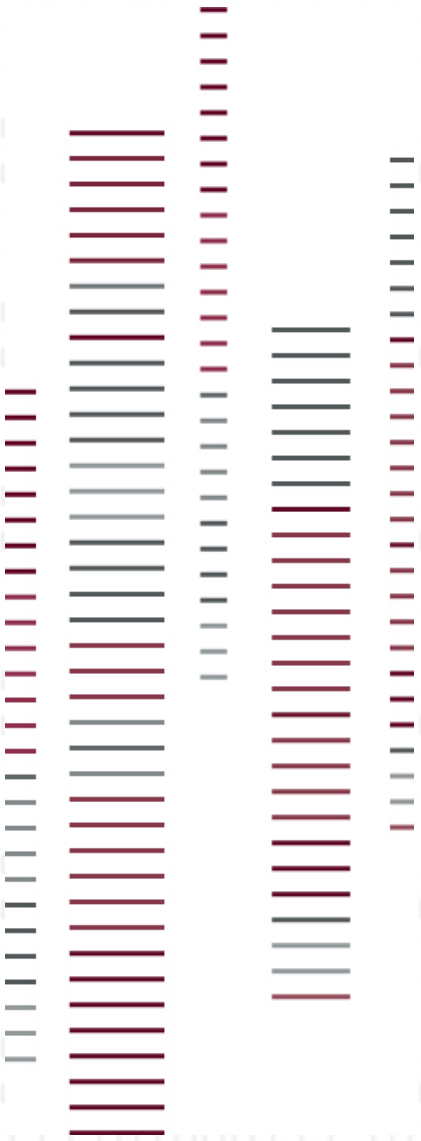


APPLICATION SECURITY, INC.

# SQL SERVER Anti-Forensics

Cesar Cerrudo

# Introduction

- Sophisticated attacks requires leaving as few evidence as possible
- Anti-Forensics techniques help to make forensics investigations difficult
- Anti-Forensics can be a never ending game
- Forensics techniques won't be detailed
- I will demonstrate that once you have DBA permissions, game is over, the whole server (OS, data, etc.) can be owned leaving almost no tracks.

# What is logged by SQL Server?

- By default SQL Server logs information on:
  - SQL Server error log
  - Windows application log
  - Default trace
  - Transaction log
- Also SQL Server saves data on:
  - Data files (databases)
  - Memory: data cache and procedure cache

# SQL Server and Windows application log

- Help to troubleshoot problems related to SQL Server
- This logging mechanism can't be disabled
- SQL Server error logs
  - Saved on LOG subforlder
  - 7 error log files are kept by default
    - Number of log files kept can be increased but no decreased
  - Named ERRORLOG, ERRORLOG.1, ERRORLOG.2,...
    - Current log file is ERRORLOG
    - New log file is created when SQL Server is restarted or error log is cycled
  - SQL Server administrators can delete them

# SQL Server and Windows application log

- Windows application log
  - Logs almost the same information as SQL Server error log
  - It also logs user name when Windows authentication is used
  - SQL Server administrators can't delete them
- What is saved?
  - Failed and successful login attempts (only if enabled)
  - Backup and restore information
  - Extended stored procedure DLL loading
  - Database (sp_dboption) and Server options (sp_configure) changes
  - Some DBCC commands
  - Error messages

# SQL Server and Windows application log

- What is not saved?
  - Extended stored procedure execution
  - Select statements
  - Some DBCC (Database Consistency Checker) commands
  - DDL (Data Definition Language) statements
  - DML (Data Manipulation Language) statements

# Default trace

- A trace is ran by default to log data necessary to diagnose and solve problems
- Trace files are saved on LOG sub folder
- Trace files are named log_X.trc where X is a number
  - A new trace files is created every time SQL Server is restarted or if the default trace option is enabled or if the files grows more than 20mb
  - 5 trace files are kept by default, when a new file is created the oldest one is deleted

# Default trace

- To enable/disable:

*EXEC sp_configure 'default trace enabled', 1*

*EXEC sp_configure 'default trace enabled', 0*

- To query if it's enabled:

*exec sp_configure 'default trace enabled'*

- Trace files can be read using SQL Server Profiler or with the next statement

*SELECT * FROM fn_trace_gettable*

*('C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\log.trc', default)*

# Default trace

- What is saved?
  - Failed login attempts
  - Login creation/deletion/modification
  - Use of trace related tables/functions/stored procedures
  - Objects creation and deletion
  - BACKUP and RESTORE statements
  - DBCC commands
  - DENY, GRAND and REVOKE statements
  - Etc.

# Default trace

- What is not saved?
  - Extended stored procedures execution
  - SELECT statements
  - DML statements

# Transaction log

- It's a log that records all transactions and database modifications
- Transaction log management depends on configured recovery model
  - Full recovery model
    - All transactions are logged, logs backup are required, database can be restored at any point
  - Simple recovery model
    - Minimal information is logged, the log is not backed up, log space is reused frequently so records are overwritten, system databases use this model

# Transaction log

- It's implemented as a separate file or set of files
  - Log file extension is .ldf
  - Can be in different location that database files
  - The next statement can be used to determine the location and name of log files of current database:

Select * from sysfiles

- Size and grow of the log can be set at database creation time or with ALTER DATABASE
- It can grow filling all available disk space so it must be backed up and truncated periodically

# Transaction log

- When the log is truncated the space of its internals structures is marked as free for reuse
  - Data is not deleted, it's overwritten
- Truncating does not reduce the size of the file
  - In order to reduce log file size it must be shrunk
    - DBCC SHRINKFILE (log_name_or_id, size)
    - Space of internal unused structures is released to OS
- Logs records for the current database can be displayed with:

*SELECT * FROM ::fn_dblog(null, null)*

# Transaction log

- What is saved?
  - The start and end of each transaction
  - Every data modification (DDL, DML)
  - Rollback operations
  - The transaction SID (Login security ID)
  - Etc.
- What is not saved?
  - SELECT statements
  - Extended stored procedure execution

# Data files

- They are files where the database data is saved
  - One database can have multiple data files
  - The main data file has an extension of .mdf
  - Their structure is not publicly known
- Data files store tables and indexes, every DDL or DML statement executed causes modification on data files.
- Data can be retrieved from data files by running queries using T-SQL.

# Data files

- Deleted data is not completely removed
  - Deleted records will remain in data files until overwritten by new records
- They can be shrunk in the same way as transaction log files
- What is saved?
  - User data, metadata
  - Results of DDL or DML statements
- What is not saved?
  - SELECT statements
  - Extended stored procedures execution
  - DBCC commands

# SQL Server memory

- SQL Server caches data on memory
- Most important caches are data and procedure cache
  - Data cache is used to store data read and written from/to data files
    - Information can be retrieved by DBCC PAGE command
  - Procedure cache is used to store execution plans of executed statements
    - Information can be retrieved by executing the next statement:

      *SELECT * FROM sys.syscacheobjects*

# SQL Server memory

- Memory addresses allocated by SQL Server can be displayed by running the next statement:
  - SELECT * FROM sys.dm_os_virtual_address_dump
- SQL Server memory can be directly read by running DBCC BYTES command
  - It is possible to read clear text passwords from recently created or modified logins
- What is saved?
  - Actually everything at some point is in SQL Server memory

# SQL Server Anti-Forensics

- From Forensics Wiki : "Anti-forensic techniques try to frustrate forensic investigators and their techniques..."
- Leave as few tracks as possible of non authorized activity, evil actions, attacks, etc.
  - The breach can't be detected
  - If breach is detected these techniques can also be used to confuse investigators.
- Sysadmin privileges are required
  - Attacker can get them: Exploiting a vulnerability, Brute forcing/guessing user and pass, Trojan, Being an evil DBA, Etc.
- The scenario discussed is a default installation of SQL Server 2005 SP 3

# SQL Server Anti-Forensics

- Some important facts in a default installation
  - Failed logging attempts are logged
  - Logging is always done to SQL Server error log and Windows application log
  - Default trace is running
  - Recovery model is set to simple in system databases (except model) and to simple or full on user databases
  - SQL Server runs under a low privileged account

# SQL Server Anti-Forensics

- Some actions an attacker will want to do
  - Steal data, modify data, install a backdoor, rootkit, etc.
  - Own the Windows server (Windows admin!=SQL Server admin)
  - Leave as few evidence as possible, preferably no evidence
- How to accomplish attacker desired actions?
  - Don't care about failed logins (attacker has user/pass, exploits SQL injection, etc.)
  - Some actions will be logged on 3 places, some on 2 places and some on 1 place, also on transaction logs and datafiles if DML or DDL command are executed, and always on memory

# SQL Server Anti-Forensics

- How to accomplish attacker desired actions?
  - Attacker can't delete Windows application log but she can delete SQL Server error log
    - But needs to cycle error log which also gets logged
  - Attacker can delete default trace file
    - But he needs to disable default trace which also gets logged
  - Attacker can run SELECT  statements, but they are logged on procedure cache in SQL Server memory
    - Can be cleaned by DBCC FREESYSTEMCACHE('ALL')
      - But the command is logged on default trace

# SQL Server Anti-Forensics

- How to accomplish attacker desired actions?
  - Attacker can modify data but it will be logged on transaction logs
    - Transaction logs can be truncated and shrunk
      - This gets logged on SQL Server and Windows logs and on default trace
      - Breaks backup chain
      - Transaction logs will have unusual sizes
  - It seems that it's pretty impossible to accomplish attacker desired actions

# SQL Server Anti-Forensics

- Accomplishing attacker desired actions
  - Logging mechanisms must be disabled (of course without being logged)
  - SQL Server provides Extended Stored Procedures (XPs)
  - Similar to stored procedures but implemented in a Windows DLL
    - DLL is loaded by SQL Server when XP is used
    - DLLs can execute code when loaded (DllMain())
  - SQL Server version < 2008 will only log information after XP is used the first time
  - XP can be used to patch memory to avoid logging and also to provide needed functionality for the attacker

# SQL Server Anti-Forensics

- Accomplishing attacker desired actions
  - When loaded XP DLL will patch:
    - ReportEventW API from Advapi32.dll to avoid logging on Windows application log
    - NTWriteFile API from Ntdll.dll to avoid logging on SQL Server error log
  - When the XP is added to SQL Server
    - It gets logged on default trace
      - Default trace should be disabled after DLL is loaded
      - Default trace file should be overwritten to erase tracks
    - Some records are created in master database
      - After removing XP, master database must be “cleaned”

# SQL Server Anti-Forensics

- Accomplishing attacker desired actions
  - Cleaning master database and transaction log (order does matter)

```
WHILE @i<1000
BEGIN
   BEGIN TRAN
   ... (code setting @randomvalue in each iteration)
   DBCC  addextendedproc('randomvalue', 'randomvalue')
   ROLLBACK TRAN
   SET @i=@i+1
END
--Shrinking master.mdf data file
DBCC SHRINKFILE (1,1)
DBCC SHRINKFILE (1,0)
DBCC SHRINKFILE (1,1)
```

# SQL Server Anti-Forensics

- Accomplishing attacker desired actions
  - Cleaning master database and transaction log (order does matter)

```
--Shrinking master.ldf transaction log
DBCC SHRINKFILE (2,1)
DBCC SHRINKFILE (2,0)
DBCC SHRINKFILE (2,1)

WHILE @i<1000
BEGIN
   CHECKPOINT --Emptying master.ldf transaction log
   SET @i=@i+1
END
```

# SQL Server Anti-Forensics

- Accomplishing attacker desired actions
  - Cleaning procedure cache
    - The next could raise alerts because slow down

      DBCC FREESYSTEMCACHE('ALL')
    - Execute statements only from master database avoiding views and stored procedures:

      SELECT * FROM targetdatabase..table

      UPDATE targetdatabase..table set data=1
    - Then just clean master database proc. cache

      DBCC FLUSHPROCINDB(1)
  - Cleaning data cache (query results, etc.)

    CHECKPOINT

    DBCC DROPCLEANBUFFERS

# SQL Server Anti-Forensics

- Accomplishing attacker desired actions
  - Modifying user databases
    - Cleaning transaction logs will break backup chain
    - DML and DDL statements can be run using a different account
      - SQL Server service account or Windows user accounts can be used
        - » Actions will be logged under a different account everywhere
      - SETUSER and EXECUTE AS
        - » Actions will be logged under a different account in transaction log

# SQL Server Anti-Forensics

- Accomplishing attacker desired actions
  - XP can provide the next functionality
    - Elevating privileges
    - Running OS commands under different Windows accounts
    - Removing tracks
    - Insert a backdoor in SQL Server memory

# SQL Server Anti-Forensics

- Elevating privileges
  - SQL Server process has impersonation tokens
    - If an Windows administrator or SYSTEM token is found then OS can be owned.
  - Token kidnapping technique
    - SQL Server service account can impersonate so it's possible to get impersonation tokens from other processes
    - 100% ownage guaranteed, DBA=Windows admin
  - After OS is compromised it's possible to clean even more tracks
    - Disk can be wiped, any OS tracks removed, install a rootkit, etc.

# SQL Server Anti-Forensics

- Running OS commands under different Windows accounts
  - XP can let the attacker to run any command
  - An impersonation token can be used to execute commands under any available Windows account

Removing tracks

  - After finishing attacker desired actions tracks must be removed
  - XP can provide functionality to remove all the tracks and remove itself

# SQL Server Anti-Forensics

- More advanced techniques
  - Insert a backdoor in SQL Server memory
    - When connecting in an specific way or running some SQL statement
      - Avoid logging automatically
      - Allow to steal other user sessions at will
  - Schedule attacks
    - Wait for victim user connection
      - Hijack connection
      - All actions logged as victim user
  - Edit logs instead of erasing or avoiding them

# SQL Server Anti-Forensics

- Attack steps
  - Add XP and execute it
    - SQL Server error log and Windows log get disabled
  - Disable default trace
  - Corrupt or overwrite default trace
  - Run desired commands
  - Execute XP to remove tracks and itself
    - Enable default trace without running it
    - Remove XP
    - Remove tracks (datafile, transaction log, caches, etc.)
    - Set default trace to run
    - Unload XP DLL
      - SQL Server error and Windows log get enabled

# Attack scenarios

- DBA is afraid of upcoming lay-offs (sounds familiar?)
  - Want to keep his job
  - Need to get rid of another DBA
  - Disable logging with a XP or with xp_cmdshell if enabled
  - Execute commands as victim DBA
    - Do things that will make look bad victim DBA
  - Remove tracks, go home and wait

Or

  - Install a SQL Server backdoor
  - If “X” command is not ran in 10 days
  - Fire payload
  - Corrupt data bit by bit, can take weeks to detect

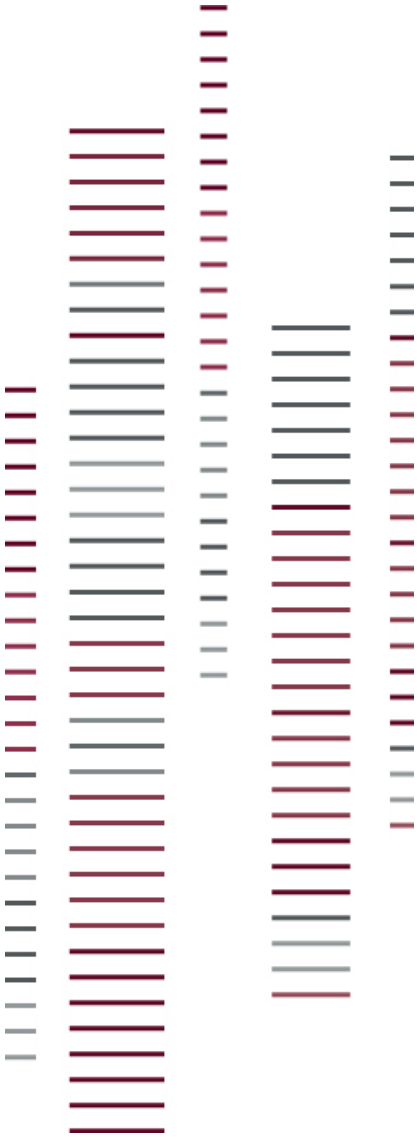
# Protections

- Use a third party database activity monitoring solution
  - DBA activity must be monitored
  - Built in database logging mechanisms can't be trusted
- Periodically scan databases for missing patches, misconfiguration, vulnerabilities, etc.
- Implement a strong password policy
  - Teach users to use pass phrases

# Conclusions

- If an attacker can connect to SQL Server as administrator the game is over
  - Attacker can complete manipulate database server leaving almost no tracks
  - Attacker can also own Windows server too
- Third party monitoring and logging mechanisms must be used
  - If not used then  your data is at SQL administrators will
  - Can't trust on SQL Server logging mechanisms

# Fin



APPLICATION SECURITY, INC.

Questions?

Thanks

Contact:
cesar>at<appsecinc>dot<com