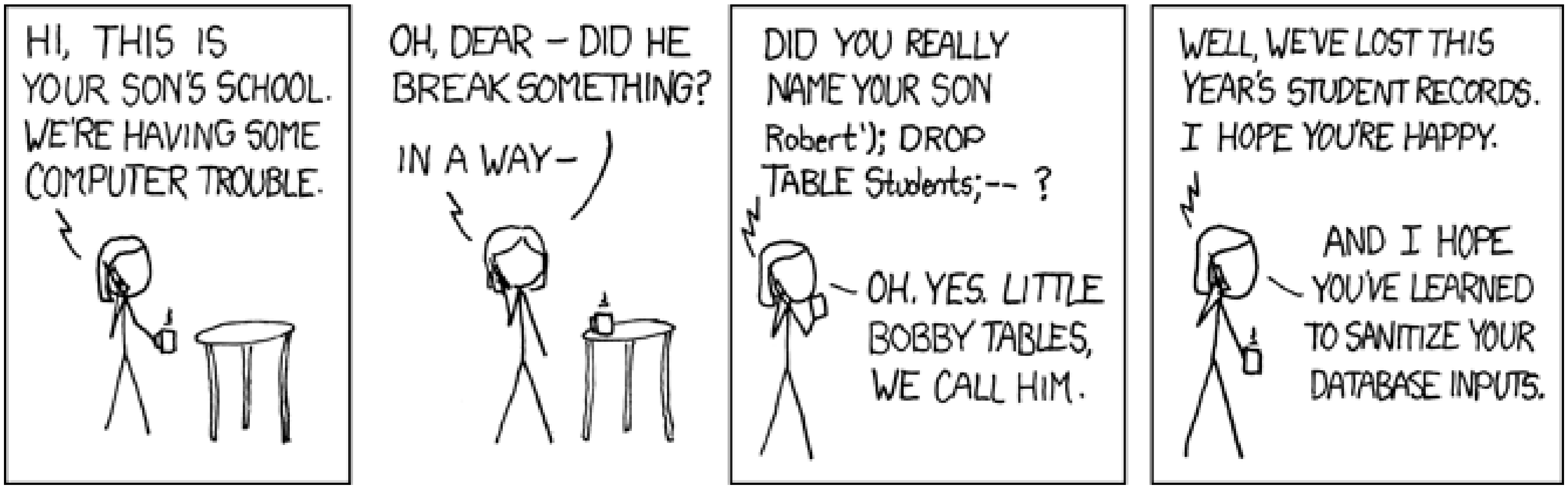
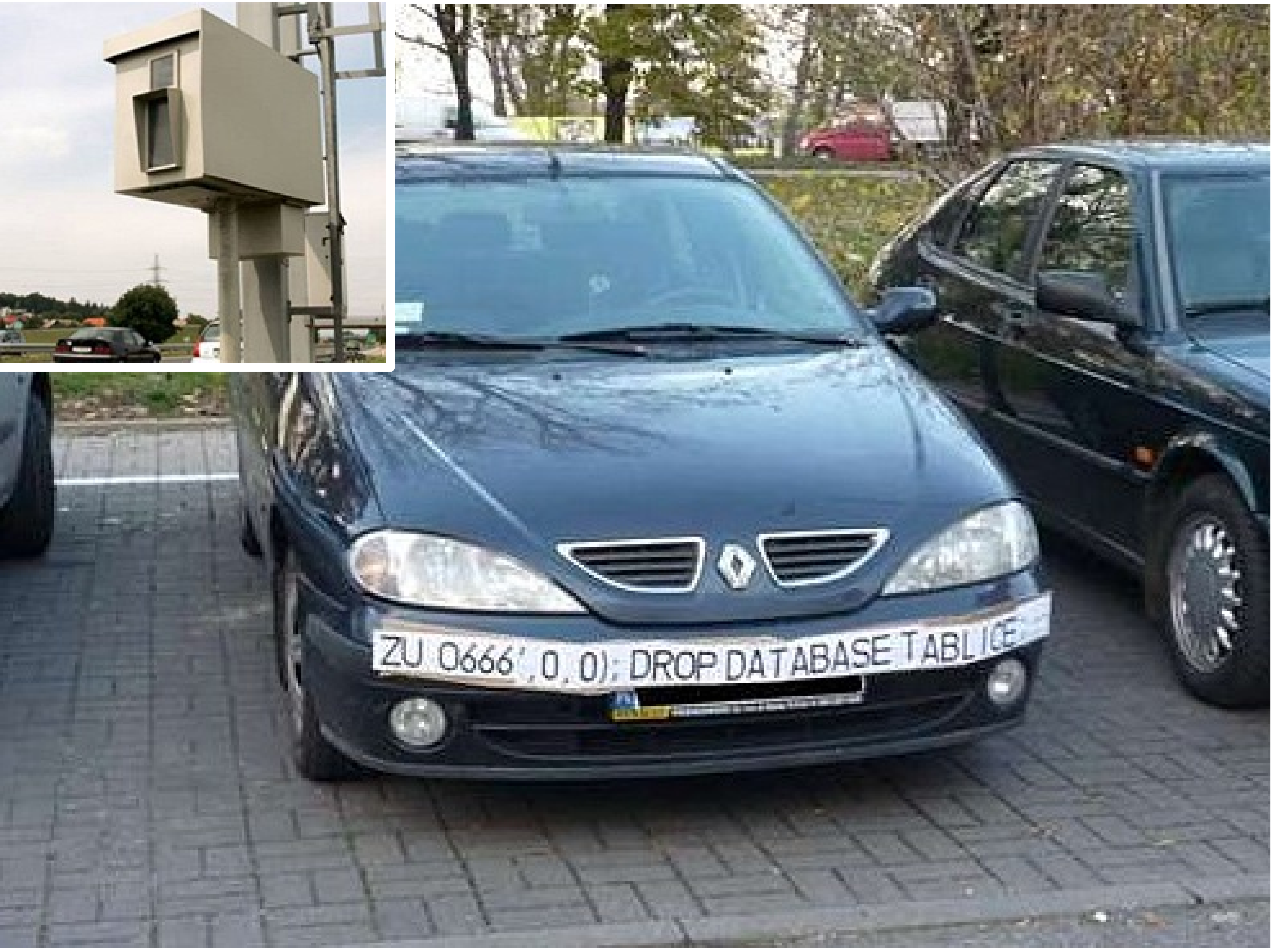

HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

AS Abakus As na disku.

Tom Kyte (asktom) je v svojem članku, ki govori o SQL vrivanju, naredil napako in omogočil SQL vrivanje.

http://tkyte.blogspot.com/2012/02/all-about-security-sql-injection.html


ZU 0666',0,0); DROP DATABASE TABLICE
[redacted]

Abakus ARBITER

ORACLE Gold Partner

Boris Oblak

Abakus plus d.o.o.

ORACLE CERTIFIED PROFESSIONAL

17. Strokovno srečanje

SIOUG 2012

Kongresni center Hotel Mons Ljubljana, 15. - 17. oktober

SIOUG Slovensko društvo Oracle uporabnikov

SQL vrivanje - kraja 130 milijonov kreditnih kartic

O podjetju

Zgodovina:

- od 1992, 20 zaposlenih
- Oracle zbirka podatkov, GNU/linux (1995)
- Dobitniki srebrnega priznanja za inovacije – Aerodrom Ljubljana: Flight Information System
- Dobitniki srebrnega priznanja za inovacije – Arbiter

Razvoj in vzdrževanje:

- Razvoj visoko razpoložljivih sistemov z OS GNU/linux
- Sistemska podpora in uglaševanje sistemov z OS GNU/linux
- Uglaševanje in administracija zbirk podatkov Oracle

Največji primer goljufije v zgodovini

- Albert Gonzalez – obsojen na 20 let zapora
- s sokrivci so uporabili SQL injection za vdor v sistem
- ~170.000.000 kartic
- http://en.wikipedia.org/wiki/Albert_Gonzalez

Zavedanje nevarnosti

- najpomembnejša nevarnost
- premalo ljudi se zaveda te nevarnosti
- aplikacija sprejme SQL stavek od nepreverjenih virov (uporabniški vnos) in ga izvede

Vezane spremenljivke

- bind variables
- brez uporabe je koda manj varna
- primer: vnos uporabniškega imena in gesla

```
select count(*)
  from user_table
 where username = <USER_NAME>
   and password = <PASSWORD>;
```

Vezane spremenljivke

```
create table user_table
   ( username varchar2(30),
    password varchar2(30) );
insert into user_table
   values ( 'boris',
   'strogo_zaupno' );
commit;

SQL> accept Uname prompt "Enter username: "
Enter username: boris

SQL > accept Pword prompt "Enter pass: "
Enter pass: nimam_pojma' or 'x' = 'x
```

Vezane spremenljivke

- brez uporabe vezanih spremenljivk

```
select count(*)
    from user_table
   where username = '&Uname'
     and password = '&Pword'
/

old   3:      where username = '&Uname'
new   3:      where username = 'boris'
old   4:        and password = '&Pword'
new   4:        and password = 'nimam_pojma' or 'x'
= 'x

  COUNT(*)
----------
         1
```

Vezane spremenljivke

- z uporabo vezanih spremenljivk

```
variable uname varchar2 (30);
variable pword varchar2 (30);
exec :uname := 'boris';
exec :pword := 'nimam_pojma'' or ''x'' = ''x';

select count(*)
  from user_table
 where username = :uname
   and password = :pword
/
   COUNT(*)
 ----------
          0
```

Vezane spremenljivke

- nezaželeni stranski efekti, če ne uporabljamo vezanih spremenljivk

```
accept pword prompt "Geslo: "
Geslo: kadri.odpusti_delavca (1234)

!?
```

SQL vrivanje – največji problem

- mogoče vse izgleda prenapihnjeno?
- www.google.com - „SQL injection“
  - 5.650.000 zadetkov (september 2012)
- ne samo VB (Active Server Pages), ne samo JavaServer Pages, php, …
- vsi jeziki, ki izvajajo SQL stavke, ki so vneseni od zunaj

Shranjene procedure

- shranjena procedura za brisanje zaposlenega

```
create or replace procedure remove_emp (p_schema in
varchar2, p_ename in varchar2)
is
  l_str clob;
begin
  l_str := '
  begin
    delete from ' || p_schema ||
     '.emp where ename = ''' || p_ename || ''';
    delete from ' || p_schema ||
     '.bonus where ename = ''' || p_ename || ''';
  end;';
  execute immediate l_str;
end;
/
```

Shranjene procedure

```
create table t (id int);

--Preverimo, koliko zapisov imamo:
SQL> select count (*) from emp where ename =
'KING';

  COUNT(*)
----------
         1

SQL> select count (*) from bonus where ename =
'KING';

  COUNT(*)
----------
         1
```

Shranjene procedure

```
begin
 remove_emp
 ( 'scott',
   'KING''; execute immediate ''drop table t'';
--' );
end;
/
begin
*
ERROR at line 1:
ORA-00942: table or view does not exist
ORA-06512: at line 4
ORA-06512: at "SCOTT.REMOVE_EMP", line 13
ORA-06512: at line 2

SQL> rollback;
```

Shranjene procedure

```
SQL> select count (*) from emp where ename =
'KING';

  COUNT(*)
----------
         0

SQL> select count (*) from bonus where ename =
'KING';

  COUNT(*)
----------
         1
```

Kako odkriti vdor

- zelo težko
- forenzično raziskovanje po izvršenem dejanju – pogoj: vklopljen AUDIT
- raziskovanje po v$sql – ne dela, če je CURSOR_SHARING = FORCE/SIMILAR
- odkriti, od kje literali prihajajo
- če so kot uporabniški vnos, potem imamo resne težave

Abakus **ARBITER**

Zaupanja vredne sledi

Težko odkrivanje

```
CREATE OR REPLACE PROCEDURE inj(p_date IN DATE)
IS
   u_rec all_users%ROWTYPE;
   c     SYS_REFCURSOR;
   l_sql CLOB;
BEGIN
   l_sql := 'select * from all_users
      where created = ''' || p_date || '''';
   dbms_output.put_line(l_sql);
   OPEN c FOR l_sql;
   FOR i IN 1 .. 5
   LOOP
      FETCH c INTO u_rec;
      EXIT WHEN c%NOTFOUND;
      dbms_output.put_line(u_rec.username);
   END LOOP;
   CLOSE c;
END;
```

Težko odkrivanje

- parameter je datum in ne string (odpade or 1=1)
- `where created = ''' || p_date || '''';`
- `where created = to_date (to_char (p_date));`
- 2x implicitna konverzija
- pogosto videna koda

Implicitna konverzija je zlo

- neželeni stranski efekti
  - trunc (datum)
- logične napake
  - 14.10.2012 in 14.10.1912?
  - NLS_DATE_FORMAT = 'dd.mm.rr' (Slovenian)

Implicitne konverzije so zlo

```
SQL> set serveroutput on;
SQL> exec inj (sysdate);

      select *
       from all_users
      where created = '25.09.12'

PL/SQL procedure successfully completed.
```

Težko preprečiti

```
SQL> alter session set nls_date_format =
'dd.mm.yyyy"'' or ''a'' = ''a"';

SQL> exec inj (sysdate);

      select *
       from all_users
      where created = '25.09.2012' or 'a' = 'a'
APPM
ABAKUS
U1
BORIS
REV_SRC_USER

PL/SQL procedure successfully completed.
```

Zabava se začne

```
SQL> nls_date_format = '"''union select
tname,0,null from tab--"';

SQL> exec inj (sysdate);

      select *
       from all_users
      where created = ''union select
tname,0,null from tab--'
BIN$yn+TlSkTCtDgQIrBPS9M3w==$0
BONUS
DEPT
EMP
SALGRADE

PL/SQL procedure successfully completed.
```

Number

```
CREATE OR REPLACE PROCEDURE inj (p_num IN NUMBER)
IS
   l_sql CLOB;
BEGIN
   l_sql := 'select object_name from all_objects
where object_id = ' || p_num;
   EXECUTE IMMEDIATE l_sql;
END;
```

- implicitna konverzija number -> char: to_char (p_num)

Number

```
SQL> select to_number ('1.01', '9d99') from dual;

TO_NUMBER('1.01','9D99')
------------------------
                    1.01


SQL> alter session set nls_numeric_characters='P ';
SQL> select to_number ('1P01', '9d99') from dual;

TO_NUMBER('1P01','9D99')
------------------------
                    1P01

SQL> select to_number ('0P01', '9d99') from dual;

TO_NUMBER('0P01','9D99')
------------------------
                     P01
```

Number

```
CREATE OR REPLACE FUNCTION p01 RETURN NUMBER
AUTHID CURRENT_USER IS
BEGIN
   FOR x_rec IN (SELECT tname
                   FROM tab)
   LOOP
      dbms_output.put_line(x_rec.tname);
   END LOOP;
   RETURN (1);
END;
/

grant execute on p01 to public;
create public synonym p01 for scott.p01;
```

Number

```
SQL> exec inj (.01);

BIN$yn+TlSkTCtDgQIrBPS9M3w==$0
BONUS
DEPT
EMP
SALGRADE
TTT
USER_TABLE
X

 'select object_name from all_objects where
 object_id = ' || p_num;

 select object_name from all_objects where
 object_id = P01;
```

Kako se zaščititi?

- težja in lažja pot :-)
- preveriti vso kodo
- testirati na različne možnosti vnosa
- dobri standardi kodiranja
    - nikoli implicitnih konverzij
    - vedno uporabiti eksplicitne datumske formate

Kako se zaščititi?

- lažja pot
  - uporabljati vezane spremenljivke
- vezane spremenljivke niso podvržene SQL vrivanju!

```
l_sql := '
  select *
    from all_users
   where created = :d';
open c for l_sql USING p_date;
```

Vezane spremenljivke?

```
CREATE OR REPLACE FUNCTION check_user (
   p_user  IN VARCHAR2,
   p_table IN VARCHAR2)
RETURN BOOLEAN IS
   l_ret NUMBER;
   l_sql VARCHAR2 (4000);
BEGIN
   -- we cannot use bind variable for table name!
   l_sql := 'SELECT COUNT (*) FROM '
      || p_table
      || ' WHERE USERNAME = :user'
   dbms_output.put_line (l_sql);
   EXECUTE IMMEDIATE l_sql
      INTO l_ret
      USING p_user;
   RETURN (l_ret != 0);
END;
```

Vezane spremenljivke

```
BEGIN
    IF NOT check_user (
        'MIHA', 'MY_USERS WHERE evil_funct() = :a1 --') THEN
        dbms_output.put_line ('Uporabnik ne obstaja!');
    END IF;
END;
/
```

Kako preprečiti

- dostop do baz omogočiti samo preko PL/SQL API
- ne uporabljati dinamičnega SQL-a, če je le mogoče
- uporabljati vezane spremenljivke
- uporabiti varen SQL text

PL/SQL API

- uporabnik nima dostopa do tabel in/ali pogledov
- uporaba privatnih sinonimov (če so potrebni – bolje »set current schema«)
- sinonimi lahko kažejo samo na PL/SQL kodo
- lastnik PL/SQL kode je tudi lastnik tabel/pogledov, zato dodatni privilegiji niso potrebni

PL/SQL API - 2

- dodatna prednost: odpade uporaba triggerjev (vse preko PL/SQL paketov)
- razširjeno pravilo
  - omogočiti dostop do pogledov
  - na pogledih uporabiti INSTEAD OF triggerje (ugodni stranski efekti: ni mutating table)

Vezane spremenljivke

- VEDNO, razen:
  - dinamično določanje imena Oracle objekta
  - pred 11g pretežno v podatkovnih skladiščih
    - 11g: variable peeking
  - nastavljanje parametrov v seji (»`alter session set optimizer_mode=all_rows`«)
    - v teh primerih so priporočljive konstante

```
c_all_rows constant varchar2 (60) :=
    'alter session set optimizer_rule=all_rows';
…
execute immediate c_all_rows;
```

Vezane spremenljivke - 2

- ko SQL stavek podamo kot parameter v eno izmed Oracle funkcij
  - dbms_utility.exec_ddl_statement()
  - dbms_ddl.create_wrapped()
  - dbms_hs_passthrough (SQL stavki v drugih bazah)
  - owa_util (generiranje HTML strani)

DBMS_ASSERT

- ko se generira SQL stavek, uporabiti
  - dbms_assert.simple_sql_name()
  - dbms_assert.enquote_literal()
  - to_char (x f, 'NLS_NUMERIC_CHARACTERS='',.''')
    - x: variable of a numeric datatype
    - f: format model 'TM' (text minimum number format model)

DBMS_ASSERT

- dodana v 10.2, backport na 8.1.7 -->
- enquote_literal
- enquote_name
- SIMPLE_SQL_NAME
- QUALIFIED_SQL_NAME
- SCHEMA_NAME
- SQL_OBJECT_NAME

enquote_literal

- podan parameter obda z enojnimi narekovaji, če še ni obdan
- dopušča gnezdenje enojnih narekovajev
- če najde en „enojni narekovaj“, vrne napako:
  ORA-06502: PL/SQL numeric or value error

enquote_literal

```
SQL> SELECT DBMS_ASSERT.enquote_literal('literal without quotes') FROM dual;

DBMS_ASSERT.ENQUOTE_LITERAL('LITERALWITHOUTQUOTES')
---------------------------------------------------
'literal without quotes'

1 row selected.

SQL> SELECT DBMS_ASSERT.enquote_literal('literal without ''''quotes') FROM dual;

DBMS_ASSERT.ENQUOTE_LITERAL('LITERALWITHOUT''''QUOTES')
---------------------------------------------------
'literal without ''quotes'

1 row selected.

SQL> SELECT DBMS_ASSERT.enquote_literal('literal without ''quotes') FROM dual;
SELECT DBMS_ASSERT.enquote_literal('literal without ''quotes') FROM dual
       *
ERROR at line 1:
ORA-06502: PL/SQL: numeric or value error
ORA-06512: at "SYS.DBMS_ASSERT", line 308
ORA-06512: at "SYS.DBMS_ASSERT", line 358

SQL>
```

enquote_name

- podan parameter obda z dvojnimi narekovaji, če še ni obdan
- dopušča gnezdenje dvojnih narekovajev
- privzeto spremeni parameter v velike črke (to se lahko spremeni s parametrom 'capitalize')
- če najde en „dvojni narekovaj“, vrne napako:
  ORA-06502: PL/SQL numeric or value error

enquote_name

```
SQL> SELECT DBMS_ASSERT.enquote_name('quoted and uppercase') FROM dual;

DBMS_ASSERT.ENQUOTE_NAME('QUOTEDANDUPPERCASE')
----------------------------------------------
"QUOTED AND UPPERCASE"

SQL> SELECT DBMS_ASSERT.enquote_name('"remains quoted and lowercase"') FROM dual;

DBMS_ASSERT.ENQUOTE_NAME('"REMAINSQUOTEDANDLOWERCASE"')
------------------------------------------------------
"remains quoted and lowercase"

SQL> SELECT DBMS_ASSERT.enquote_name('pairs of ""quotes"" are allowed') FROM dual;

DBMS_ASSERT.ENQUOTE_NAME('PAIRSOF""QUOTES""AREALLOWED')
------------------------------------------------------
"PAIRS OF ""QUOTES"" ARE ALLOWED"
```

enquote_name

```
SQL> SELECT DBMS_ASSERT.enquote_name('individual "quotes" are not allowed') FROM
dual;
SELECT DBMS_ASSERT.enquote_name('individual "quotes" are not allowed') FROM dual
       *
ERROR at line 1:
ORA-06502: PL/SQL: numeric or value error
ORA-06512: at "SYS.DBMS_ASSERT", line 308
ORA-06512: at "SYS.DBMS_ASSERT", line 343
ORA-06512: at line 1

SQL> SET SERVEROUTPUT ON
SQL> EXEC DBMS_OUTPUT.put_line(DBMS_ASSERT.enquote_name('quoted and remains
lowercase', FALSE));

"quoted and remains lowercase"

PL/SQL procedure successfully completed.

SQL>
```

simple_sql_name

- checks the input string conforms to the basic characteristics of a simple SQL name:
  - The first character of the name is alphabetic.
  - The name only contains alphanumeric characters or the "_", "$", "#"
  - Quoted names must be enclosed by double quotes and may contain any characters, including quotes provided they are represented by two quotes in a row ("").

simple_sql_name

  - The function ignores leading and trailing white spaces are ignored
  - The length of the input string is not validated.
- when input string does not conform, ORA-44003 is raised:

ORA-44003: Invalid SQL name

qualified_sql_name

- <local qualified name> ::= <simple name> {'.' <simple name>}
- <database link name> ::= <local qualified name> ['@' <connection string>]
- <connection string> ::= <simple name>
- <qualified name> ::= <local qualified name> ['@' <database link name>]
- [SCHEMA-NAME.]OBJECT-NAME[@DBLINK-NAME]

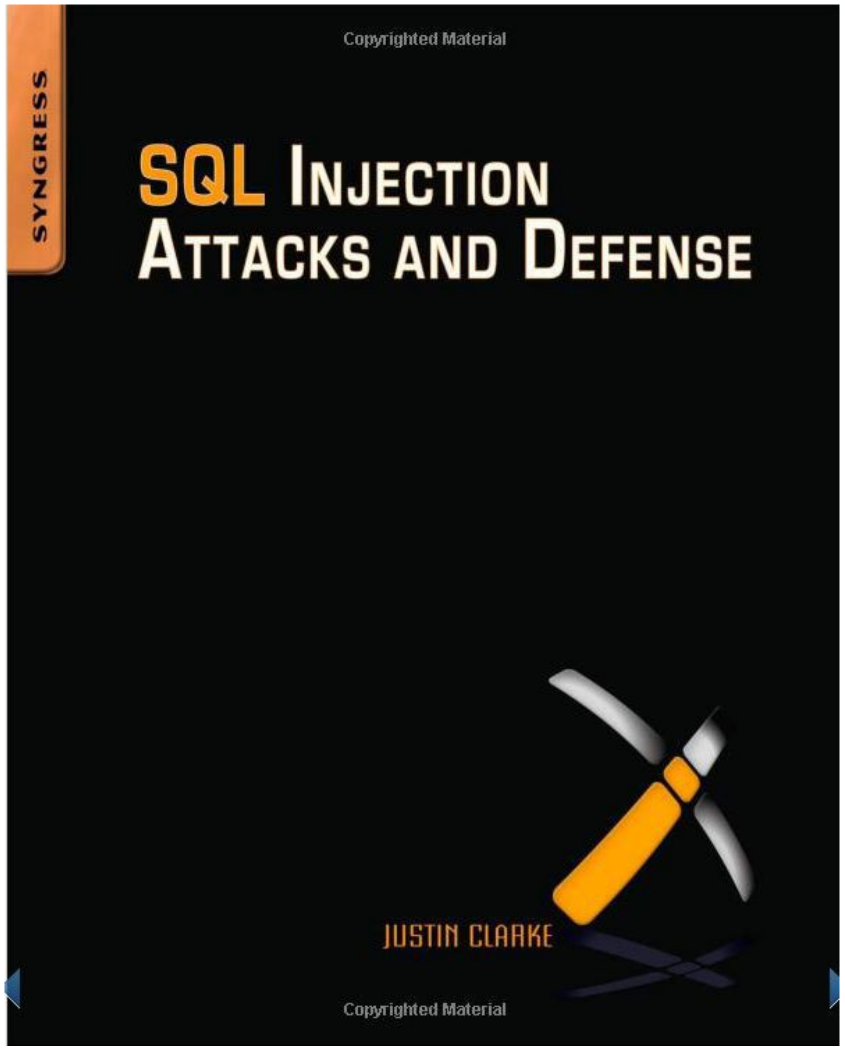
schema_name

- existing schema name
- case sensitive!

sql_object_name

- existing object, not case sensitive
  - 'dbms_assert'
  - 'sys.dbms_assert'
  - 'sys.dbms_assert@db_link'
- with db link only syntax is checked!

Več o tem ...



Copyrighted Material

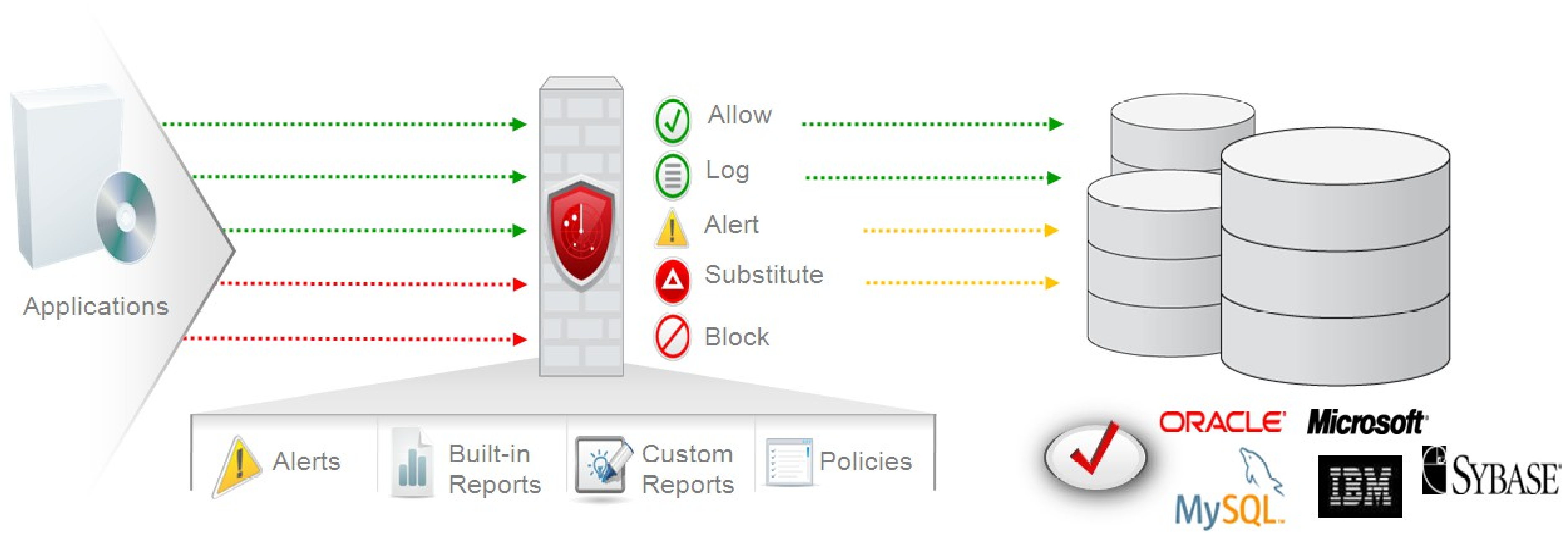
SYNGRESS

SQL INJECTION ATTACKS AND DEFENSE

JUSTIN CLARKE

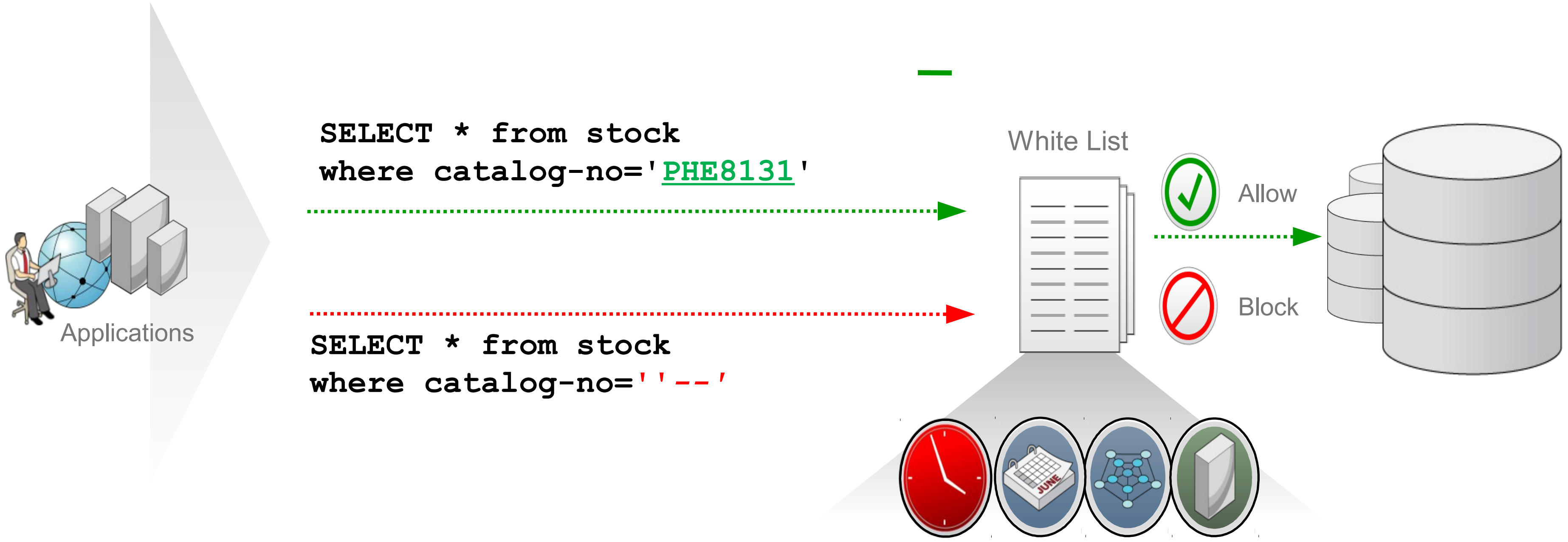
Copyrighted Material

Oracle Database Firewall
First Line Of Defense



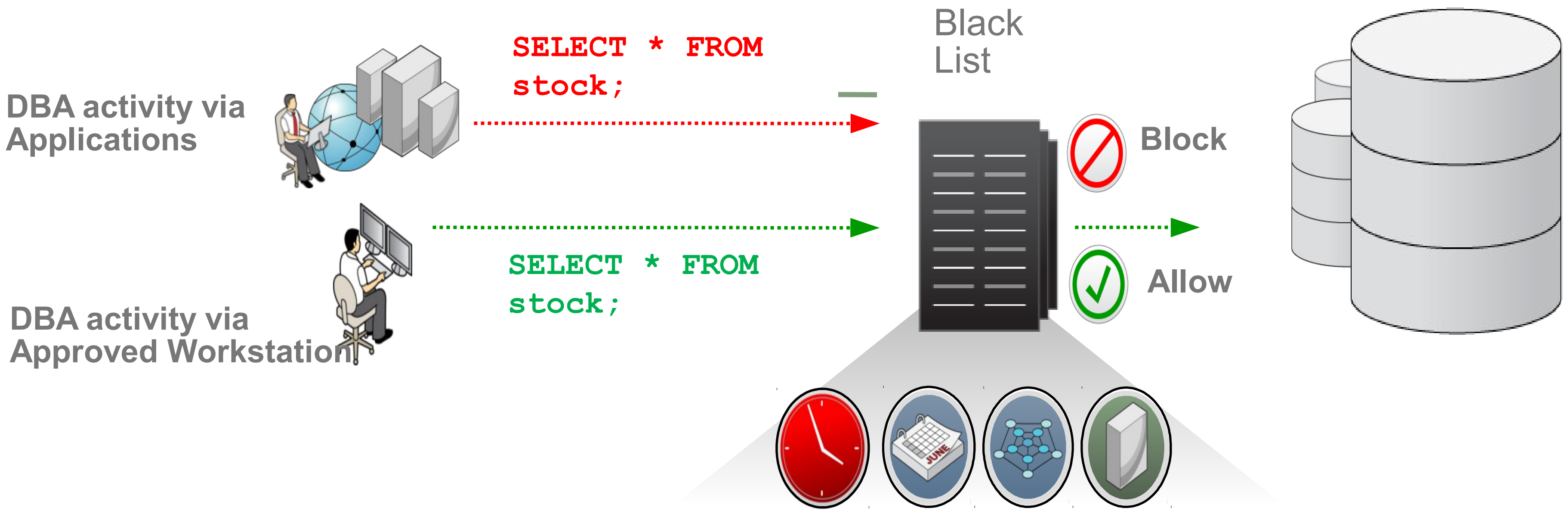
- Monitors database activity, prevents attacks and SQL injections
- White-list, black-list, and exception-list based security policies based upon highly accurate SQL grammar based analysis, no disruptive false positives
- In-line blocking and monitoring, or out-of-band monitoring modes

Positive Security Model



- "Allowed" behavior can be defined for any user or application
- Automated whitelist generation for any application
- Many factors to define policy (e.g. network, application, etc)
- Out-of-policy database network interactions instantly blocked

Negative Security Model



DBA activity via Applications
DBA activity via Approved Workstation
SELECT * FROM stock;
SELECT * FROM stock;
Black List
Block
Allow
JUNE

- Stop specific unwanted SQL interactions, user or schema access
- Ensures database interactions originate from appropriate sources
- Blacklist can take into account session factors such as time of day, day of week, network, application, etc
- Provide flexibility to authorized DBAs while still monitoring activity

SQL vrivanje predstavlja največjo nevarnost za SQL baze podatkov, saj ga je izredno težko odkriti oziroma preprečiti!

ORA-03113: end-of-file on communication channel

Boris Oblak
Abakus plus d.o.o.

ORACLE CERTIFIED PROFESSIONAL

ORACLE Gold Partner

17. Strokovno srečanje
SIOUG 2012
Kongresni center Hotel Mons Ljubljana, 15. - 17. oktober

SIOUG Slovensko društvo Oracle uporabnikov

SQL vrivanje - kraja 130 milijonov kreditnih kartic