



# Self-Defending Databases

Alexander Kornbrust,  
Red-Database-Security GmbH



# Agenda

- Introduction
- Root Cause
- Anatomy of an attack
- Detection of an attack
- Countermeasures
- How to implement
- Discussion

# Introduction

This presentation shows how databases can defend themselves against SQL Injection attacks without human interaction.

Web applications/services are permanently attacked from the internet (successful/unsuccessful). A successful attack often leads to data loss (e.g. data is posted on websites like [pastebin.com](https://pastebin.com) ).

The majority of attacker are using tools to attack web applications and to download data that's why human reaction on these events is normally to slow.

# Introduction

Monitoring 30 web applications: (Imperva Trend Report #4, Sep 2011)

- on average 71 SQL injection attempts per hour
- 800-1300 injection attempts at peak times
- Use of highly automated SQL injection tools, e.g. **sqlmap**, **Havij**,...

# Hashdays 2011 – Protecting Databases with Trees

The presentation “A syntax-based approach to detect SQL injections” from Christian Bockermann showed how to use the parse tree to detect SQL Injection attacks.

This approach is smart but complex (SQL Parser, Training data, ...). Additionally it does not answer the problem what to in case of an SQL Injection.

# Root Cause

Problem:

Web applications are often vulnerable against SQL Injection

Solution:

Fix all vulnerable web applications and allow only the deployment of secure (after pentest) applications





Is this really realistic in a  
(large) organization?



# No !

- There is no secure code
- Majority of applications are not pentested
- Applications are longer used than expected (sometimes 10+ years)
- Application patches are difficult to get/patch or not available





# Anatomy of an SQL Injection Web Attack

1. Use a tool (e.g. Havij, Netsparker, Matrixay, Pangolin, SQLMap, ...) or google to find a SQL Injection vulnerability by crawling the entire website
2. Select the tables (data) in the tool
3. Download the data via the tool



# 1. Find a SQL Injection Vulnerability

# Google Hacking



ociexecute ora-01756

Suche

Ungefähr 19.700 Ergebnisse (0,23 Sekunden)

Alles

Bilder

Maps

Videos

Tipp: [Suchen Sie nur nach Ergebnissen auf Deutsch](#). Sie können Ihre Sprache in den [Einstellungen](#) festlegen.

[BLACK COFFEE | Stonepeak Ceramics](#)

[www.stonepeakceramics.com/...](#) - [Vereinigte Staaten](#) - [Diese Seite übersetzen](#)

Warning: ociparse() [function.ociparse]: **ORA-01756**: quoted string not properly ...

Warning: **ociexecute()** expects parameter 1 to be resource, boolean given in ...

## Scan Configuration Wizard

### Step 2/6 - Application Settings

Indicate the URL where the scan should start



Starting:

Show

For example: <http://demo.testdbapp.com.cn/>

Help

<< Back

Next >>

Cancel

## Scan Configuration Wizard

### Step 3/6 - Advance setting

Crawl advance rule setting!



#### Additional Servers and Domains

- ☒ Scan current URL      ☐ Scan current domain  
☐ Scan current subdomain      ☐ Scan any URL

#### Scan Level

count:  if set 0, don't control scan level

#### Thread count

count:  1~40

#### crawl mode

☐ Simple Mode

(Ignore parameter valuse differ in the crelw)

Exclude directory...

Exclude file

Help

<< Back

Next >>

Cancel

## Scan Configuration Wizard

### Step 6/6 - Inject settings

Select inject module



Default Method:

Module

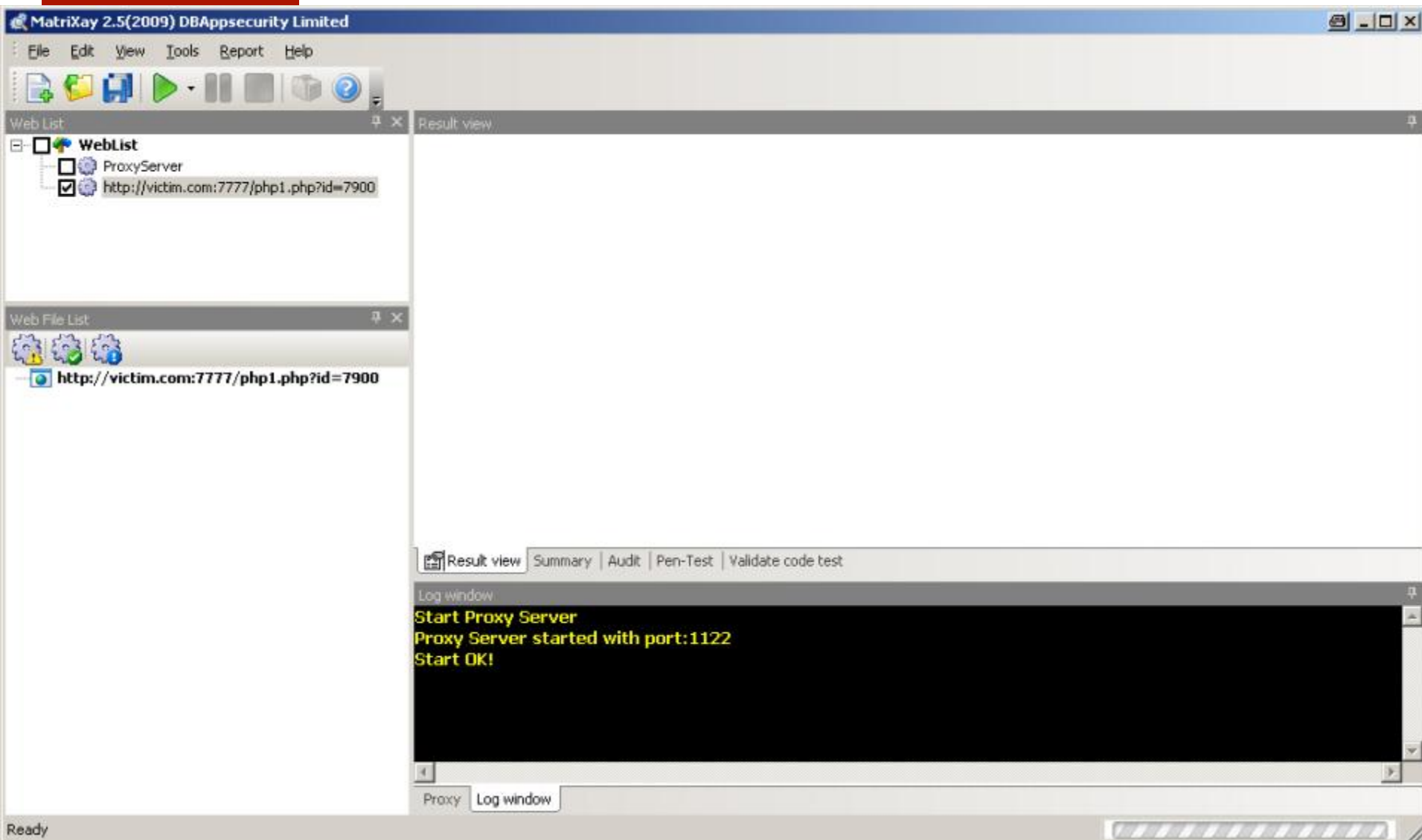
- |   |                                 |                                     |
|---|---------------------------------|-------------------------------------|
| <input checked="" type="checkbox"/> SQL injection | <input type="checkbox"/> Trojan | <input type="checkbox"/> Form Check |
| <input type="checkbox"/> Cross site-scripting     | <input type="checkbox"/> XPath  | <input type="checkbox"/> Web 2.0    |
| <input type="checkbox"/> Form Brute               | <input type="checkbox"/> Misc   |                                     |

Help

<< Back

Finished

Cancel





2. Select the data





**MatriXay 2.5(2009) DBAppsecurity Limited**

File Edit View Tools Report Help

Web List

- WebList
  - ProxyServer
  - http://victim.com:7777/php1.php?id=7900

Web File List

- http://victim.com:7777/php1.php?id=7900

Result view

- SQL Injection (1)**
  - http://victim.com:7777/php1.php?id=7900
    - (005)-SQL Injection type:number, DataBase type:Oracle, DataBase Name:ora11, DataBase User:SCOTT
      - (003):DEPT
        - DUMMY
        - EMP
        - SALGRADE
        - SOOD

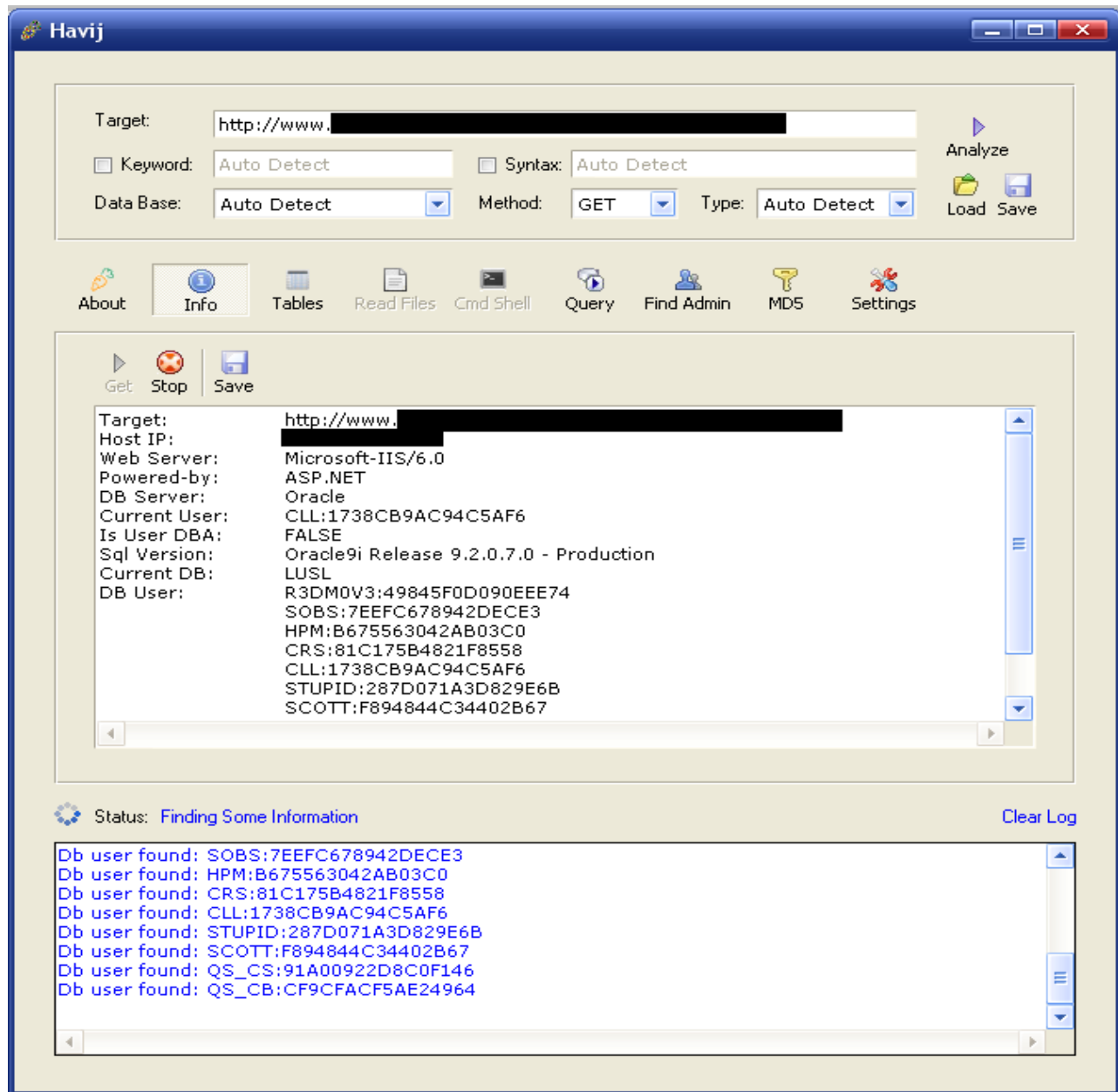
Result view | Summary | Audit | Pen-Test | Validate code test

Log window

```
Get TableName SOOD
(*)Fuzzing field Count
Short Smart Field count :3
(*)Fuzzing field count finished!
```

Proxy | Log window

Ready





### 3. Download the data

Pangolin -- Maded By Zwell -- <http://www.nosec.org>

URL:  GET

Type: Integer DB: Oracle KeyWord:

Check Pause Stop

Options Reset

Informations Datas Oracle Remote Data

Table/Column	DEPTNO	DNAME	LOC
<input type="checkbox"/> SOOD	10	ACCOUNTING	NEW YORK
<input type="checkbox"/> DUMMY	20	RESEARCH	DALLAS
<input type="checkbox"/> SALGRADE	30		
<input checked="" type="checkbox"/> DEPT			
<input checked="" type="checkbox"/> DEPTNO			
<input checked="" type="checkbox"/> DNAME			
<input checked="" type="checkbox"/> LOC			
<input type="checkbox"/> EMP			

Tables Columns Datas One by one 1=1 Save

Content is : DALLA  
Content is : DALLAS  
Processing records of 3/4  
Length is : 2  
Content is : 3  
Content is : 30

Running... Version 1.3.1.650



And the data is gone ...

(in often less than 2 minutes)

# Potential Reaction for SQL Inj. Attacks



## Solution:

Within 2 minutes after the attack started, the Manager on Duty is receiving an alert and automatically stops the attack by shutting down the service



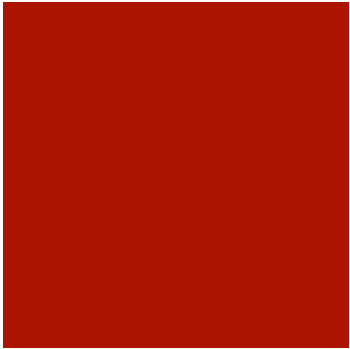
Is this really realistic in a  
(large) organization?



# No !

- Time for a human reaction is too short
- Several companies do not have a Security information and event Management ( SIEM ) and Security Operation Center (SOC) in place to forward these kind of alert,
- Even with a SIEM system 2 minutes are a challenge for most organizations
- Side effects of a stopping a system/service is not documented in most companies
- A manager on duty would normally not stop a system





We don't live in a perfect  
world that's why we need a  
different real-world  
approach against SQL  
Injection attacks

# Approach

1. The system itself has to detect the hacking attempt
2. Appropriate counter measures have to be taken

# Detection

How and where can we detect a SQL Injection hacking attempt?

- Web Application Firewall (WAF/IDS)
  - Can block some of the attacks by filtering the input
- Webserver
  - Can block some of the attacks by filtering the input (mod\_security)
- Application
  - Not without changing the application itself (which is difficult)
- Database
  - Yes, by detecting SQL errors

# Detection



Out-of-the-box Databases like Oracle or Microsoft SQL Server are able to detect SQL specific error messages and can run (custom) code (=countermeasure) after the detection.

MySQL could use this technique via a MySQL proxy.

These specific database errors only occur if a vulnerability exists and this vulnerability was triggered by a specific string (e.g. `“or 1=1--”`)

False positives are rare. A false positive could occur if a developers are deploying applications with incorrect SQL statement (e.g. missing single quote).

# Implementation

The implementation of this detection has to be done in different ways depending from the underlying database:

Oracle:

- **Database Error Trigger**

Microsoft SQL Server:

- **Event Notification**

MySQL:

- **MySQL Proxy or MySQL Audit Plugin\***

\* <http://dev.mysql.com/doc/refman/5.5/en/writing-audit-plugins.html>

# Detection of SQL Injection attacks via error messages

- Depending from the used attack method (UNION, extend query, create error messages to retrieve data, ...) a specific error will be created  
e.g.  
*ORA-01789: query block has incorrect number of result columns*
- Or  
*Microsoft OLE DB Provider for ODBC Drivers error  
'80040e07' [Microsoft][ODBC SQL Server Driver][SQL Server]Syntax  
error converting the nvarchar value '**mypassword**' to a column of  
data type int. /Administrator/login.asp, line 27*

# Typical SQL Injection Attack I

## Original SQL command


```
select custname, custid, custorder from customer;
```

## SQL command extended by an attacker

```
select custname, custid, custorder from customer  
union  
select username, null, password from dba_users;
```

# Typical SQL Injection Attack II



Address  <http://192.168.2.172:8889/reports/rwservlet?report=c:\project\sqlinject3.rdf+userid=scott/tiger@ora9206+destype=CACHE>

SQL Injection

Empno	Ename	Sal
7369	SMITH	800
7499	ALLEN	1600
7521	WARD	1250
7566	JONES	2975
7654	MARTIN	1250
7698	BLAKE	2850
7782	CLARK	2450
7788	SCOTT	3000
7839	KING	5000
7844	TURNER	1500
7876	ADAMS	1100
7900	JAMES	950
7902	FORD	3000
7934	MILLER	1300



# Typical SQL Injection Attack III



:\project\sqlinject3.rdf+userid=scott/tiger@ora9206+destype=CACHE+desformat=HTML+paramform=yes

Submit Query

Reset

**Berichtparameter**

**Geben Sie die Parameterwerte ein**

**P Where**

ORDER BY 1

# Typical SQL Injection Attack IV



---

t=c:\project\sqlinject3.rdf+userid=scott/tiger@ora9206+destype=CACHE+desformat=HTML+paramform=yes

---

Submit Query

Reset

**Berichtparameter**

**Geben Sie die Parameterwerte ein**

**P Where**

UNION select NULL,USERNAME

# Typical SQL Injection Attack V



## Injected:

*Union null,username from all\_users--*

## Error message:

ERROR at line 1:


ORA-01789: query block has incorrect number of result columns

➔ Attacker (or tool) is adding NULLs until a proper SQL statement was created and executed

## Next attempt:

*Union null,null,username from all\_users—*

# Typical SQL Injection Attack VI

Address  http://192.168.2.172:8889/reports/rwervlet?

SQL Injection

Empno	Ename	Sal
7369	SMITH	800
7499	ALLEN	1600
7521	WARD	1250
7566	JONES	2975
7654	MARTIN	1250
7698	BLAKE	2850
7782	CLARK	2450
7788	SCOTT	3000
7839	KING	5000
7844	TURNER	1500
7876	ADAMS	1100
7900	JAMES	950
7902	FORD	3000
7934	MILLER	1300

ANONYMOUS

CTXSYS

DBSNMP

HR

MDSYS

ODM

# SQL Injection Error Codes Oracle - I

Error code	Error Message	Typical Command
ORA-00900	invalid SQL statement	
ORA-00906	missing left parenthesis	
ORA-00907	missing right parenthesis	
ORA-00911	invalid character	e.g. PHP MAGIC_QUOTES_GPC activated and attempt to inject a single quote
ORA-00917	missing comma	
ORA-00920	invalid relational operator	
ORA-00923	FROM keyword not found where expected	
ORA-00933	SQL command not properly terminated	
ORA-00970	missing WITH keyword	
ORA-01031	insufficient privileges	Attempted privilege escalation
ORA-01476	divisor is equal to zero	Blind SQL Injection attempt (e.g. sqlmap)
ORA-01719	outer join operator not allowed in operand of OR or IN	
ORA-01722	invalid number	Enumeration with rownum and current rownum does not exist

# SQL Injection Error Codes Oracle - II

Fehlernr	Fehlermeldung	Auslöser
<b>ORA-01742</b>	comment not properly terminated	inline comment, e.g optimizer hint is not properly terminated
<b>ORA-01756</b>	quoted not properly terminated	single quote not properly terminated
<b>ORA-01789</b>	query block has incorrect number of result columns	Attempt to use UNION SELECT
<b>ORA-01790</b>	expression must have same datatype as corresponding	Attempt to use UNION SELECT
<b>ORA-24247</b>	network access denied by access control list	Oracle ACL has blocked the usage of UTL_INADDR (or similar)
<b>ORA-29257</b>	Host %S unknown	Attempted SQL Injection via utl_inaddr
<b>ORA-29540</b>	Class does not exist	Attempted utl_inaddr attempt but Java is not installed
<b>ORA-31011</b>	XML parsing failed	SQL Injection attempt via xmltype
<b>ORA-19202</b>	Error occurred in XML processing	SQL Injection via extractvalue

# SQL Injection Error Codes MSSQL

Error Message	Typical Command
Unclosed quotation mark before the character string "	Usage of single quotes
Syntax error converting the varchar value 'test' to a column of data type int.	<u>Usage of --</u>
Column '[COLUMN NAME]' is invalid in the select list because it is not contained in an aggregate function and there is no GROUP BY clause.	http://[site]/page.asp?id=1 having 1=1--
Syntax error converting the nvarchar value '[DB USER]' to a column of data type int.	http://[site]/page.asp?id=1 or 1=convert(int,(USER))--

# React on errors

- The system could react on the errors caused by SQL Injection attempts
  - Detection only / Audit the event
  - Send an email to the manager-on-duty/DBA/Security Department
  - Lock the database account
  - Terminate this session or terminate all sessions
- To minimize the impact, different database accounts should be used for different applications (e.g. Internet, Intranet, Android, iOS, ..).
- If the application is blocked from the internet, intranet users can still work with the application.
- Only errors caused by the application server should create such a reaction (i.e. ORA-01756 from SQL\*Plus, TOAD or SQL Management Studio will be ignored)



# Additional options to react

- After the account was locked the user can't use the webapp
- Send an email to the operating and/or Manager on Duty
  - Analyze error ( 'or 1=1 oder O'Leary)
  - Unlock account in case of false positive
  - In case of a real alert try to identify the type of attacker (Amateur, Pro, Skript-Kiddie with tool, ...)
  - Option to lock ip ranges (lock only people outside of hungary)
  - Fix the software bug and/or search a workaround.



# Oracle Error Trigger

## - Sample code

```
CREATE OR REPLACE TRIGGER after_error
AFTER SERVERERROR ON DATABASE
DECLARE
sql_text ORA_NAME_LIST_T;
v_stmt CLOB;          -- SQL statement causing the problem
n NUMBER;             -- number of junks for constructing the sql statement causing the
error
v_program VARCHAR2(64);
v_serial number;
v_sid number;
BEGIN
-- Version 1.00
select program,serial#,sid into v_program,v_serial,v_sid from v$session where
sid=sys_context('USERENV', 'SID');
-- construct the sql text
n := ora_sql_txt(sql_text);
--
IF n >= 1
THEN
FOR i IN 1..n LOOP
v_stmt := v_stmt || sql_text(i);
END LOOP;
END IF;
--
```

```
FOR n IN 1..ora_server_error_depth LOOP
```

```
IF (lower(v_program) = 'iis.exe') -- add your own application server
```

```
and (ora_server_error(n) in
```

```
('942','900','906','907','911','917','920','923','933','970','1031','1476','1719','1722','1742','1756','1789','1790','19202','24247','29257','29540','31011'))
```

```
THEN
```

```
-- Potential attack was detected
```

```
-- 1. Monitor the attack
```

```
-- 2. Send an email to the responsible person (DBA/MoD)
```

```
-- send_email (e.g. via utl_smtp )
```

```
-- 3. Lock database user used by the webapp
```

```
execute immediate ('ALTER USER /* Error_Trigger */ "" |  
sys_context('USERENV','SESSION_USER') || "" account lock');
```

```
-- 4. Terminate Session
```

```
execute immediate ('ALTER SYSTEM /* Error_Trigger */ KILL SESSION "" | | v_sid | | ','' | |  
v_serial | | "" account lock');
```

```
alter system kill session 'session-id,session-serial'
```

```
-- 5. Other countermeasures
```

```
END IF;
```

```
END LOOP;
```

```
--
```

```
END after_error;
```

```
/
```



SQL Server

# Concept SQL Server

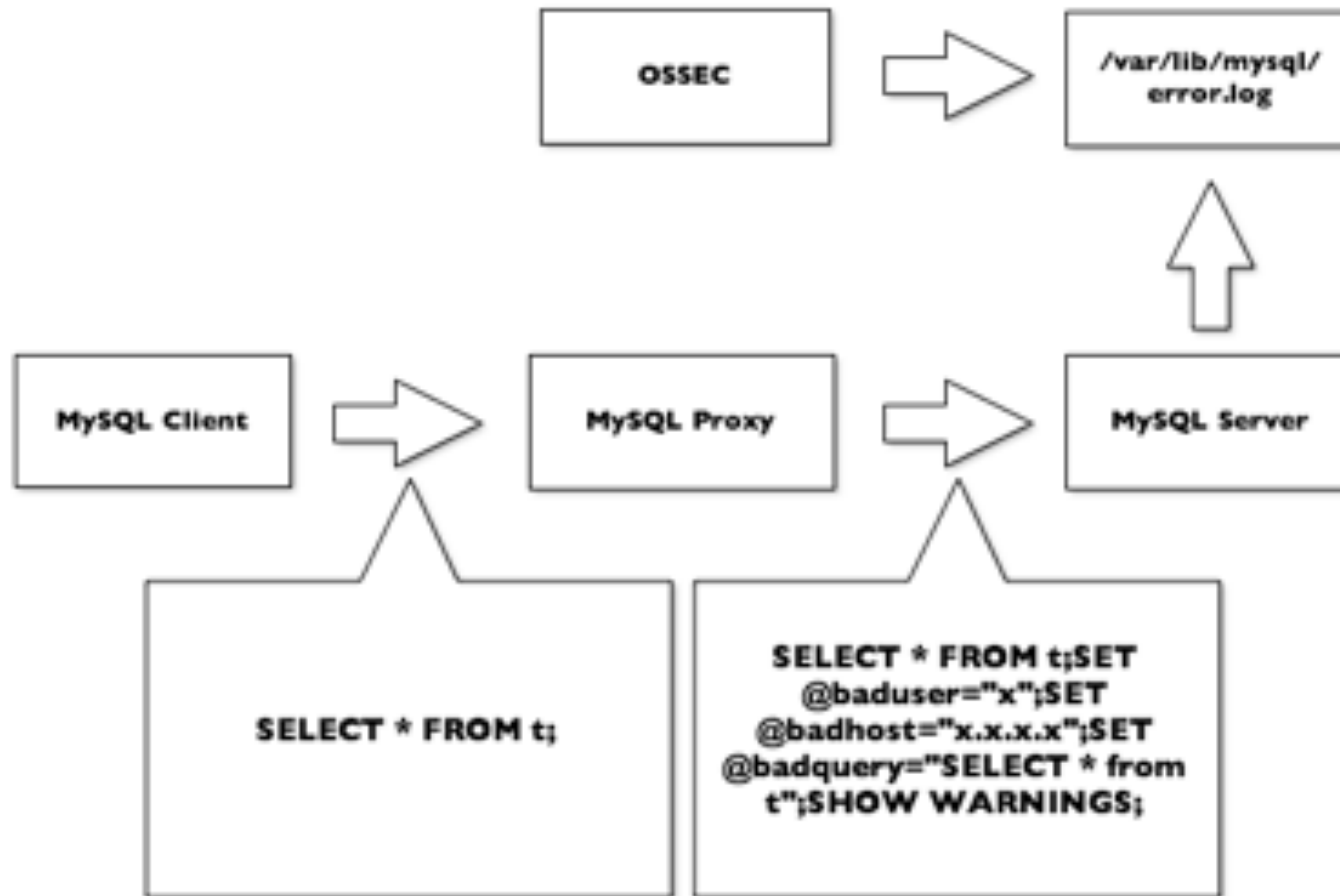
- Event notifications are a special kind of database object that send information about server and database events to a Service Broker service.
- Create events for typical SQL injection errors



# MySQL

(all credits go to Xavier Mertens)

# Concept MySQL





# Concept MySQL

- Create an UDF to write errors in a log file.
- A LUA script will rewrite the query by appending the “SHOW WARNINGS” statement + some variables at the end of the query.
- Then the query results of the modified query will be read by the LUA scripts and written to a log-file
- It is also possible to lock the MySQL user account (similar to the other databases)



Countermeasures after  
detecting an attack

# Countermeasures after detection I

Monitor the attempt and send an email to the security officer

Pro:

- Small footprint
- No side effect on the application

Cons:

- Fast response needed
- What happens during the night, vacation, ...

# Countermeasures after detection II

Lock the database account and kill all already running processes

Pro:

- Attack is immediately stopped
- No data is lost

Cons:

- Side effect on the application (Denial-of-Service)
- Potential false positive

# Countermeasures after detection III

Get the IP address from the web application server and start a denial-of-service against the IP where the attack was coming from

Pro:

- Database strikes back ;-)

Cons:

- Is this legal?



What do you prefer?

Service not available

or

Data is lost/published?

# What happens after logout?

Manager on Duty can decide if the database account should be re-enabled / unlock.

This is normally an easier decision instead of stopping a service to stop an on-going attack?

# Potential false positives

If a web application is vulnerable against SQL Injection attacks, an “accidental” string like “**O’Leary**” can trigger the account lockout.

In this case the string was not part of an attack. In such a case it could be an option to exclude the string from the detection and re-enable the service again.

The vulnerability should be fixed as soon as possible.

Or

Developer has deployed wrong SQL code (e.g. Single Quote is missing)



# SQL Injection Errors from Web Application Scanner

If a web application is vulnerable against SQL Injection attacks, and a web application scanner is performing a scan we know that we are under attack.

How can we detect that a webapp scanner was causing the error?

# SQL Injection Errors from Web Application Scanner

## Vulnerable URL:

*php3.php?ename=test*

Webapp Scanner is trying to inject patterns and analyzes the result

*php3.php?ename=' and 1=0 union select 1,password from dba\_users where username='SYSTEM'--*

## ERROR:

*ORA-01789 - query block has incorrect number of result columns*

## SQL Statement:

*Select \* from emp where ename='' and 1=0 union select 1,password from dba\_users where username='SYSTEM'--*

➔ Acunetix was used

# Acunetix – Forensic Traces



Test strings (partial)
<code>   (select username from dual) --</code>
<code>' union select username,password from dba_users--</code>
<code>'union select user, sysdate from dual --</code>
<code>and 1=0 union select 1,2 from dual--</code>
<code>and 1=1 Union select null,banner from v \$version--</code>

# SQL Injection Errors from Web Application Scanner

## Vulnerable URL:

*php1.php?id=7900*

Webapp Scanner is trying to inject patterns and analyzes the result

*php1.php?id=' **OR 'ns'='ns***

## ERROR:

*ORA-00933: SQL command not properly ended*

## SQL Statement:

*Select \* from emp where id=' **OR 'ns'='ns***

➔ **Netsparker (ns) was used**

# Netsparker – Forensic Traces



Test strings (partial)
OR 17-7=10
+CHAR(95)+CHAR(33)+CHAR(64)
' OR 'ns'='ns
OR 1=1
/**/AND/**/1=/**/ CHAR(95)+CHAR(33)+CHAR(64)+SUBSTRING(CA ST((SELECT/**/@@version)/**/AS/**/ varchar(3000)), 0,343)+CHAR(95)+CHAR(33)+CHAR(64)

# Matrixay – Forensic Traces



Test strings (partial)
AnD 1=1
AnD AsC(1)<65535
AnD user<Chr(0)
AnD (SeLEcT CoUNt(TaBLe_NaME) FrOM user_tables)>0
AnD AsCIi(DaTAbAsE())=0

# HP Webinspect – Forensic Traces



Test strings (partial)
value' OR 5=5 OR 's'='0
value' AND 5=5 OR 's'='0
value' OR 5=0 OR 's'='0
value' AND 5=0 OR 's'='0
0+value
value AND 5=5
value AND 5=0
value OR 5=5 OR 4=0
value OR 5=0 OR 4=0

# Pangolin – Forensic Traces



Test strings (partial)

```
union all select null from dual-- and  
1=1
```

```
union all select null,null from dual--  
and 1=1
```

```
and (select length(table_name) from  
(select rownum r,table_name from  
(select rownum r,table_name from  
user_tables where rownum<=1 order by 1  
desc)
```



# SQL Injection Errors from Web Application Scanner

A stored procedure in the database could be used to identify the common SQL Injection tools and block the access.

This stored procedure is called inside the trigger/event notification.

# Improve the concept I

- The application could use dedicated connections for dedicated services (e.g. internal users and external users are using a different connection or special connections for IOS and Android Apps). A lockout of the external users does not affect the internal users
- Block only IP's or IP ranges from specific blocks/regions/countries (e.g. if majority of customers is coming from CH, block requests from the outside of CH)

# Improve the concept II

- The database mechanism to detect attacks could communicate with the webserver to block specific IPs already at the webserver level instead of blocking the entire account
- Depends how much time and effort is done for the implementation

# Improve the concept III

- Use different weights for the decision what to do
  - Error comes from a TOR network +10
  - Error comes from the intranet +4
  - Error comes from an uncommon country (e.g. Turkmenistan) +7
  - Injected string contains a -- +10
  - Know attack string from webapp scanner +20
  - Single quote comes after D, O +1 (Palm D'or, O'Connor) +1
  - Single quote comes after a nonD/non-O +3
  - ...
- Add all weights
- If a certain weight is reached lock the user



Other ways to detect an  
SQL Injection attack

# Fake Data (Honey Data)



- Using fake data could help to identify attacks which are not triggered by error messages (e.g. if attacker uses a known exploit for standard software (e.g. Wordpress, ...))
- Fake data (Honey data) is data (e.g. Passwords, Credit card numbers, ...) in tables which is never used by the application. If someone from the web application server is accessing this kind of data this is often part of the data discovery process of the attacker.

# Fake-Data (Honey-Data)

- Creation a table or tables containing unused data with juicy names (e.g. PASSWORD, CREDITCARD, SALARY). Such interesting data is often the target of attackers.
- During the attack, attackers are often accessing the view ALL\_TAB\_COLUMNS (Oracle) or INFORMATION\_SCHEMA.COLUMNS (MSSQL) to get the column names of interesting data
- Attackers are normally downloading the data of interesting tables found via the column name in further attacks.
- You could monitor such an access and could react (send email, lock user, ...)
- Oracle can implement this monitoring via Virtual Private Database (VPD)

# Fake-Data (Honey-Data)

-- Create Honeytable

```
create table app.userdata (username varchar2(30), password varchar2(30));
```

-- Fill Honeytable with data

```
insert into app.userdata values ('WEBUSER','WEBUSER01');
```

```
insert into app.userdata values ('WEBADM','ADMADM01');
```

```
insert into app.userdata values ('WEBREAD','READUSER01');
```

-- create predicate function

```
create or replace function perfcheck (pv_schema in varchar2, pv_object in  
varchar2)
```

```
return varchar2 as  
begin
```

```
dbms_output.put_line('Send email to the security team or lock the database user...');
```

-- return always true. Attacker will see all results

```
return '1=1';
```

```
end;
```

```
/
```

-- now we activate VPD for this table

```
exec dbms_rls.add_policy(object_schema => 'APP', object_name => 'USERDATA',  
policy_name => 'PERFCHECK', policy_function => 'PERFCHECK');
```



# Fake Functions

- Developers are typically using obvious function names.
- A common function name for encrypting/decrypting data is `encrypt()/decrypt()`.
- If an attacker finds a encrypted password column and a function called `decrypt`, he will probably use the `decrypt` function:  
  
`Select decrypt(password) from app.appusers;`
- Instead of decrypting the the password, this function is sending an email to the security officer.

# Summary

- Self-Defending databases can be a cheap and fast step to protect databases.
- Implementation is transparent. No need to change the application (but it could be useful)
- Can also be used during pentests to monitor if SQL Errors were triggered by Pentesters
- Data loss can be prevented
- Dilemma for the management:  
Stop the service or stop the data loss



Q & A?

# Thanks

- Contact:

Red-Database-Security GmbH

Bliesstr. 16

D-.66538 Neunkirchen

Germany

