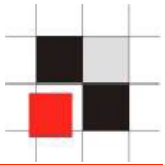
red database security

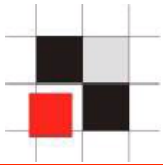
# Oracle Auditing

Alexander Kornbrust
29-Oct-2008

# Table of Content



# Introduction

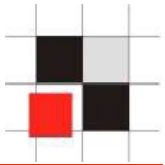


Some numbers from a German survey (741 companies) – End of 2007

| | |
|---|---|
| Damage | 2.8 Billion EUR (Germany only!) |
| Espionage Growth | 10% per year |
| Espionage incidents | 18.9% |
| Assumed incidents | 35.1% |
| Affected Departments | Sales (20%), R&D (16.1%), HR (14.7%), MFG (13.3%) |
| Attackers | Internal Employees (20%), Competitor (15%) |
| Police involved | <25% |
| Offender | Admin. (31.3%), Technician (22.9%), Manager (17.1%) |

http://bc1.handelsblatt.com/news/loadbin/ShowImage.aspx?img=1567932&typ=handelsblatt.pdf

# Introduction

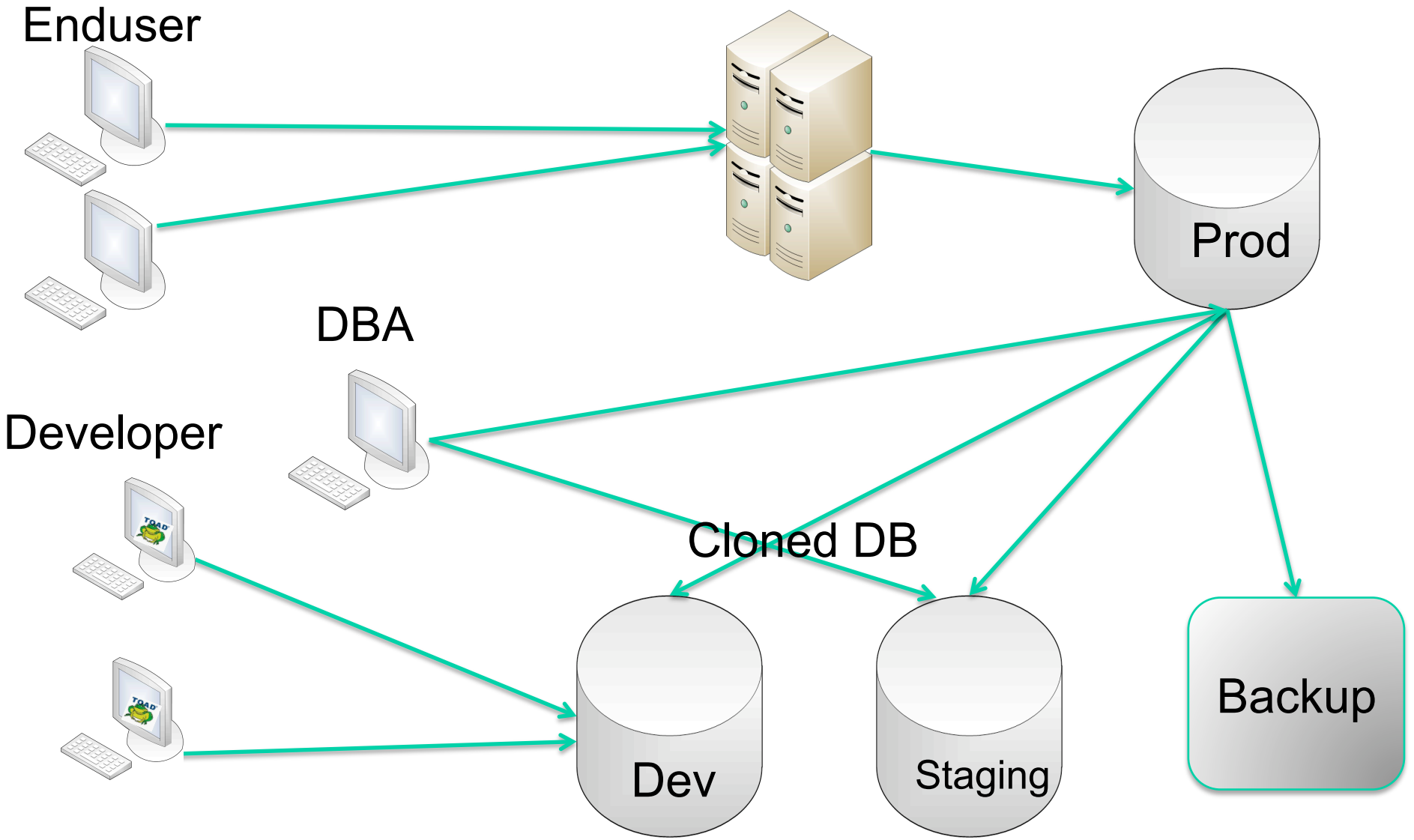


Most Oracle Forensics approaches are bottom-up approaches. Start at the bottom of the database to find traces of an attack (e.g. Find deleted data in data blocks, using logminer to search in archive logs, …).

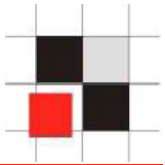
From my experience this approach is often not useful because different attackers are causing different traces in the system.

To find attackers fast and cost efficient it is necessary to classify the attackers because different attackers are using different kind of attacks.

# Introduction – Simplified Company Environment



# Classification Attackers

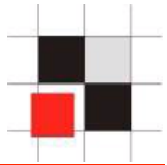


There are different types of attackers  and we need different approaches to catch theses guys because they are leaving different tracks in the system

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.

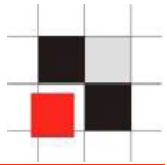
**Samples:**

- 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

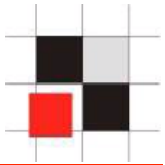
**Scenario:** Try to remove evidence about a (serious) fault.
Probably it's not a good approach to ask the DBA to do the forensics

**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, ….

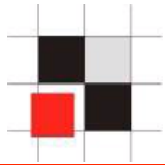
**Samples:**

- 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

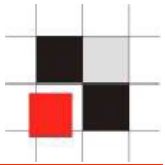
**Scenario:** Get as much data/information for the new job as possible.
Most common attack

**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:** Get as much data/information for the new job as possible.

**Samples:**

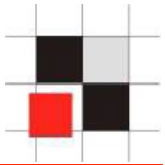
- Steal data for a competitor
- Steal credit card information
- Steal Source Code
- Break in just for fun

**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, …

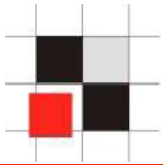
**Known Incidents:**

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

**Known Suspects:**

- China, France, Israel, Russia, US

# Typical Oracle Forensic Questions

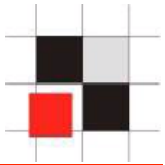


Instead of following the bottom-up-approach we are starting from top-down.

Here some typical forensic questions

- What happened between 1.00 am and 6.00 am in our database?
- What happened between 1.Sep 2007-13-Oct 2007?
- Who changed the password of user SYSTEM?
- What was done by the former employee Mr. Smith in the last 3 months before he left the company?
- We have a fraud incident. Is the the leak coming from the/what database?
- There was a hacker in our network. Are our database infiltrated?

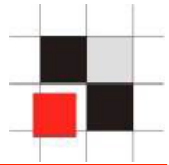
# Typical Oracle Forensic Problems



During a forensic analysis you are often running into the following problems

- DBs are often cloned from the production system. Multiple targets. Development/Staging Database are often easier to hack.
- Database not patched, many entry points
- Logfiles (e.g. listener.log) non existent or deleted
- No audit information
- Forensic analysis starts too late (1-2 years after the incident)
- Employee workstations completely erased

# Typical Traces of Attackers – Curious DBA or Employee

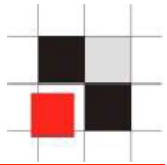


These users have often already all the privileges and know the system (+ security precaution) quite good. User and/or database enumeration is not necessary. Difficult to track because often only (a few) select statements where executed

**Attack from the database and/or application**

- Log-Entries listener.log (outside of the business hours)
- Audit entries if auditing is enabled (find unusual operation)
- Oracle Workload Repository (sys.WRH$*)
- V$sql contains last executed SQL statements if the database was not restarted

# Typical Traces of Attackers – DBA covering tracks

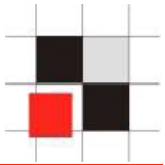


To avoid problems/trouble in the job, sometimes DBA's are deleting traces of their faults.

**Attack from Database and/or OS**

- Backup from the time of the incident removed/deleted
- Archive log deleted/corrupted
- Audit-logs modified (Oracle rowids in the right sequence)

# Typical Traces of Attackers – Criminal Employee



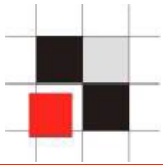
Difficult to track. Knows the system quite well. Large amount of different possibilities

Implements sometimes logic bombs, triggers, backdoors, …

**Attack from Database and/or OS**

- Modified / added source or backdoor (Use checksums against DB objects)

# Typical Traces of Attackers – Leaving Employees

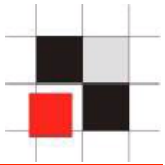


Steals everything he can get (files, large reports, …). Normally no experience in covering tracks.

**Attack from Database and/or Application**

- Listener.log (e.g. exp.exe)
- Audit-Information
- Analysis of the users workstation (e.g. what USB device was attached).

# Typical Traces of Attackers – External Hacker



Often coming from an external system (webserver, SQL Injection, …). This kind of attack is causing a lot of trace information, depending were the attacker was coming from

**Attack from a Webserver**

- Access.log / Error.log of the webserver

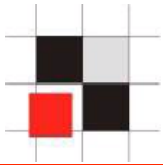
**Attack from OS**

- Listener.log
- Audit entries if auditing is enabled
- Backdoors in the database (glogin.sql, rootkits, …)

**Attack from Application**

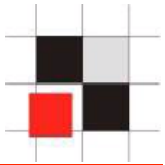
- Privilege escalation, SQL Injection, …

# Starting



A good starting point for many of these incidents is the listener.log. The following slides explain how to start, what to do and how to interpret the results

# Analyzing the listener.log

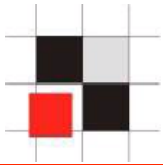


The Oracle listener.log should be analyzed on a regular basis to find out:

- Who is accessing the database when
- Programs used to access the DB (e.g. TOAD on a production database, licensing issues)
- Database links accessing the DB
- D.O.S. attempts (stop TNS listener, rare)
- What remote apps must be changed during a password change

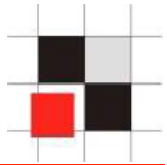
Keep in mind that most of the entries in the TNS protocol (like program, username, …) can be forged but most attackers are not doing this

# Monitor Listener.log with external table



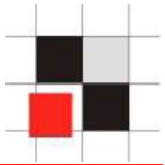
```
create table listener_log (
   log_date date,
   connect_string varchar2(300),
   protocol_info varchar2(300),
   action varchar2(15),
   service_name varchar2(15),
   return_code number(10)
)
organization external (
   type oracle_loader
   default directory LISTENER_LOG_DIR
   access parameters   (
      records delimited by newline
      nobadfile
      nologfile
      nodiscardfile
      fields terminated by "*" lrtrim
      missing field values are null  (
          log_date char(30) date_format
          date mask "DD-MON-YYYY HH24:MI:SS",
          connect_string,
          protocol_info,
          action,
          service_name,
          return_code      )    )
   location ('listener.log'))
reject limit unlimited
/
```

# Monitor Listener.log with external table



```
create or replace function parse_listener_log_line
   (
       p_in varchar2,
       p_param in varchar2
   )
   return varchar2
   as
       l_begin     number(3);
       l_end       number(3);
      l_val       varchar2(2000);
  begin
       if p_param not in (
                'SID', 'SERVICE_NAME','PROGRAM','SERVICE',
                'HOST','USER', 'PROTOCOL','TYPE',
                'METHOD','RETRIES', 'DELAY','PORT','COMMAND'
       ) then
           raise_application_error (-20001,'Invalid Parameter Value '|
|p_param);
       end if;
       l_begin := instr (upper(p_in), '('||p_param||'=');
       l_begin := instr (upper(p_in), '=', l_begin);
       l_end := instr (upper(p_in), ')', l_begin);
       l_val := substr (p_in, l_begin+1, l_end - l_begin - 1);
       return l_val;
  end;
 /
```

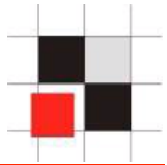
# Show all programs accessing the DB



```
select parse_listener_log_line(connect_string,'PROGRAM') program,
       count(1) cnt
from listener_log
group by parse_listener_log_line(connect_string,'PROGRAM');
```

| | |
|---|---|
| C:\InstalledPrograms\Quest Software\TOAD\TOAD.exe | 1 |
| C:\Program Files\Actuate7\Server\operation\fctsrvr7.exe | 25,796 |
| C:\Program Files\Embarcadero\DBA700\DBArt700.exe | 53 |
| C:\Program Files\Informatica PowerCenter 7.1\Client\pmdesign.exe | 1 |
| C:\Program Files\Microsoft Office\OFFICE11\EXCEL.EXE | 20 |
| C:\Program Files\Microsoft Office\Office10\MSACCESS.EXE | 4 |
| C:\Program Files\Oracle\jre\1.1.8\bin\jrew.exe | 9 |
| C:\Program Files\Quest Software\TOAD\TOAD.exe | 846 |
| c:\9I_CLIENT\bin\sqlplus.exe | 5 |
| exp@odsddb01 | 2 |
| oracle | 31 |
| oracle@stcdwhdd | 4 |
| sqlplus | 20 |

# Analyzing Results – used programs



1. Connect to the Oracle database

```
C:\> sqlplus system/mypw@mydb/mysid
```

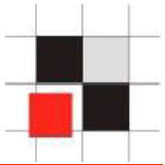
2. Run the listener.log Analysis scripts for single or multiple instance

```
SQL> @http://www.mycompany/listenersingle.sql
SQL> @http://www.mycompany/listenermulti.sql
```

3. Check and analyze the results

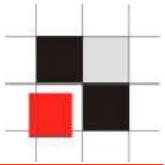
```
notepad listener_single_analysis.txt
```

# Analyzing Results – used programs



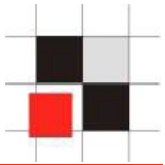
- What programs are used?
- Is the entire used software licensed?
- Are the "unofficial" applications in use? (e.g. developed by a student)
- Is the path of all application compliant with your company standard (e.g. c:\Program Files)
- Look for highs and lows numbers
  A brute force/dictionary attack tool is normally connecting quite often.
  A tool to exploit the database is normally called only once

# Analyzing Results – used programs



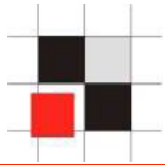
| | |
|---|---|
| | 14,392 |
| C:\tools\sidguess\sidguess.exe | 1,062 |
| E:\wintools\sidguess.exe | 970 |
| c:\tools\sidguess.exe | 528 |
| C:\repscan2008\repscan.exe | 173 |
| C:\oracle\ora101mid\Apache\Apache\Apache.exe | 53 |
| C:\tools\OracSec\OracSec.exe | 31 |
| C:\tools\checkpwd.exe | 8 |
| c:\tools\checkpwd.exe | 8 |
| D:\wintools\sqlplus.exe | 4 |
| C:\oracle\ora101\bin\emagent.exe | 3 |
| C:\oracle\ora101\bin\sqlplus.exe | 3 |
| C:\repscan2008\repscan180.exe | 3 |
| c:\oracle\ora101\bin\ORACLE.EXE | 3 |
| C:\Programme\repscan\sqlplus.exe | 2 |
| C:\Users\rbalupar\Documents\VMWare?XP?Shared?Folder\Oracle?RDS?Trainin g\instantclient_win_10_2\sqlplus.exe | 2 |
| C:\oracle\ora101\perl\5.6.1\bin\MSWin32-x86\perl.exe | 2 |
| C:\tools\checkpwd123.exe | 2 |
| c:\tools\sqlplus.exe | 2 |
| C:\oracle\ora101\bin\EXP.EXE | 1 |
| C:\oracle\ora101\bin\expdp.exe | 1 |
| C:\oracle\ora101\bin\impdp.exe | 1 |
| C:\repscan2008\repscanconfig.exe | 1 |
| E:\instantclient_win_10_2\sqlplus.exe | 1 |
| E:\wintools\sqlplus.exe | 1 |
| c:\tools\alex.exe | 1 |

# Analyzing Results – used Usernames



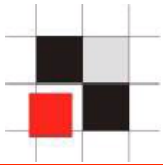
- Compliant to your company naming convention?
- Unusual names (like hacker666 or xyz)

# Analyzing Results – used Usernames



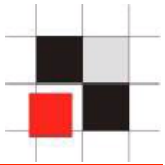
| | |
|---|---|
| oracle | 1,855 |
| dpino | 532 |
| Administrator | 440 |
| SYSTEM | 60 |
| root | 6 |
| kkqq | 5 |
| alexanderkornbrust | 3 |
| frh | 2 |
| rbalupar | 2 |

# Analyzing Results – used Export Utilities



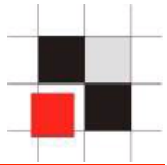
- Who is using export utilities?
- What export utilities are used?

# Analyzing Results – used export utilities



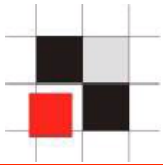
```
Kkqq     C:\oracle\ora101\bin\EXP.EXE                  1
Oracle   C:\oracle\ora101\bin\expdp.exe                1
```

# Analyzing Results – used Hacker Utilities



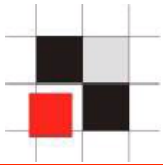
- Was someone using hacker/security utilities?
- Was this part of a normal audit?

# Analyzing Results – used hacker utilities



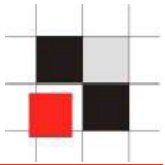
```
Administrator E:\wintools\sidguess.exe    439
dpino E:\wintools\sidguess.exe 531
frh C:\Programme\repscan\sqlplus.exe 2
oracle C:\repscan2008\repscan.exe  173
oracle C:\repscan2008\repscan180.exe 3
oracle C:\repscan2008\repscanconfig.exe 1
oracle C:\tools\checkpwd.exe   8
kkqq C:\tools\checkpwd123.exe   2
kkqq C:\tools\sidguess\sidguess.exe  1,062
oracle c:\tools\checkpwd.exe   8
oracle c:\tools\sidguess.exe 528
```

# Analyzing Results – Usage of listener commands



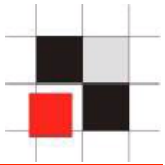
- Listener commands are often used in the enumeration phase of an attack
- This could help to find the time when the attack started

# Analyzing Results – Usage of listener commands



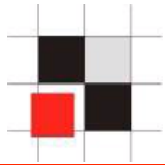
```
Stop     1
status   30
```

# Analyzing Results – Usage per day



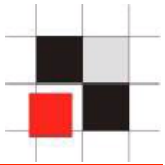
- Most of the connects should happen on working days (e.g. Mon-Fri or Sun-Thu)
- High numbers (at the weekend) could be an indication for an attack

# Analyzing Results – Usage of per day



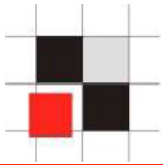
| | | | |
|---|---|---|---|
| 2007-11-28 | 53 | 2008-01-14 | 1 |
| 2007-11-29 | 144 | 2008-01-27 | 21 |
| 2007-12-02 | 102 | 2008-01-28 | 226 |
| 2007-12-03 | 1,185 | 2008-01-29 | 115 |
| 2007-12-04 | 463 | 2008-01-30 | 11 |
| 2007-12-05 | 20 | 2008-02-04 | 47 |
| 2007-12-06 | 261 | 2008-02-05 | 42 |
| 2007-12-07 | 72 | 2008-02-11 | 29 |
| 2007-12-10 | 14 | 2008-02-14 | 1 |
| 2007-12-13 | 17 | 2008-02-21 | 17 |
| 2007-12-14 | 1 | 2008-02-24 | 4 |
| 2007-12-17 | 150 | 2008-03-12 | 159 |
| 2007-12-18 | 193 | 2008-03-13 | 177 |
| 2007-12-19 | 172 | 2008-03-14 | 8,236 |
| 2007-12-20 | 73 | 2008-03-15 | 135 |
| 2007-12-22 | 7 | 2008-03-16 | 147 |
| 2007-12-26 | 3 | 2008-03-17 | 215 |
| 2007-12-27 | 20 | 2008-03-22 | 7 |
| 2008-01-03 | 31 | 2008-03-27 | 36 |
| 2008-01-04 | 12 | 2008-04-09 | 40 |
| 2008-01-05 | 3 | 2008-04-13 | 21 |
| 2008-01-09 | 1 | 2008-04-14 | 1 |
| 2008-01-10 | 3 | 2008-04-15 | 230 |
| 2008-01-11 | 18 | 2008-04-16 | 7 |
| | | 2008-04-17 | 4,087 |

# Analyzing Results – Usage per hour



- Look for high and lows
- Depending from the application there are normally peaks in the morning/after the lunch break

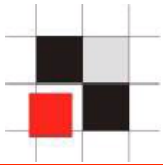
# Analyzing Results – Usage per hour



| | |
|---|---|
| 00 | 526 |
| 01 | 36 |
| 02 | 45 |
| 03 | 75 |
| 04 | 104 |
| 05 | 93 |
| 06 | 61 |
| 07 | 41 |
| 08 | 4,229 |
| 09 | 133 |
| 10 | 142 |
| 11 | 145 |
| 12 | 152 |

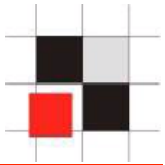
| | |
|---|---|
| 12 | 152 |
| 13 | 172 |
| 14 | 262 |
| 15 | 259 |
| 16 | 6,009 |
| 17 | 2,359 |
| 18 | 305 |
| 19 | 183 |
| 20 | 195 |
| 21 | 167 |
| 22 | 141 |
| 23 | 1,196 |

# Analyzing Usage



- Small collection of scripts to get a report for the following questions:
  - What happened from 13-apr-2008 11:07 til 13-apr-2008 14:12
  - What was done in the database by Mr. Smith?
  - What was done in the database by Mr. Smith in the timeframe (13-feb-2008-13-apr 2008)?
  - Backdoors installed

# Usage – Analyze timeframe



1. Connect to the Oracle database

```
C:\> sqlplus system/mypw@mydb/mysid
```

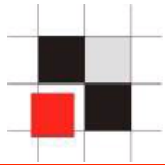
2. Run the script analyze_usage

```
SQL> @http://www.mycompany/
analyzetimeframe.sql
SQL> enter Starttime: 13-apr-2008 11:07
SQL> enter Endtime: 13-apr-2008 14:12
SQL> generating report timeframe_analysis.txt…
```

3. Check and analyze the results

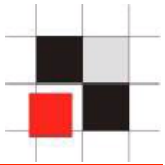
```
notepad timeframe_analysis.txt
```

# Analyzing Results – Analyze timeframe



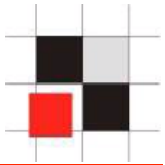
- Object creation (e.g. procedures, tables, …)
- Entries in Database log-tables (SYS.AUD$, SYS.FGA_AUD$, dbms_scheduler_log, …)

# Analyzing Results – Analyze timeframe



```
Objects created:
13-apr-2008 12:01  CREATE FUNCTION SCOTT.F1 (sys.object$)
13-apr-2008 12:02  GRANT DBA TO PUBLIC    (sys.aud$)
13-apr-2008 12:07  drop user system cascade;   (sys.aud$)
```

# Usage – Analyze User Activity



1. Connect to the Oracle database

```
C:\> sqlplus system/mypw@mydb/mysid
```

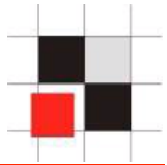
2. Run the script analyze_usage

```
SQL> @http://www.mycompany/analyzeactivity.sql
SQL> Enter username
SQL> generating report activity_analysis.txt…
```

3. Check and analyze the results

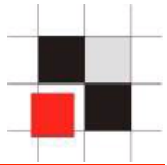
```
notepad activity_analysis.txt
```

# Analyzing Results – Analyze User Activity over a timeframe



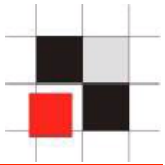
- If an employee left company/is suspicious
- Longer timeframe than previous reports
- Object creation (e.g. procedures, tables, …)
- Entries in Database log-tables (SYS.AUD$, SYS.FGA_AUD$, dbms_scheduler_log, …)

# Analyzing Results – Analyze User Activity over a timeframe



```
Objects created:
19-mar-2008 12:01  CREATE TABLE SMITH.TMP (sys.object$)
19-mar-2008 12:01  SELECT APP.CREDITCARD  (sys.aud$)
21-marc-2008 12:02 CREATE TABLE SMITH.TMP2(sys.object$)
```

# Usage – Analyze User Activity over a period



1. Connect to the Oracle database

```
C:\> sqlplus system/mypw@mydb/mysid
```

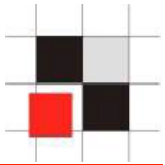
2. Run the script analyze_usage

```
SQL> @http://www.mycompany/
analyzeactivitytimeframe.sql
SQL> enter Starttime: 13-feb-2008
SQL> enter Endtime: 13-apr-2008
SQL> enter Username: SMITH
SQL> generating report
activitytimeframe_analysis.txt…
```

3. Check and analyze the results

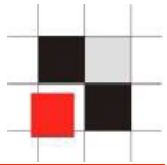
```
notepad activitytimeframe_analysis.txt
```

# Analyzing Results – Backdoors



- Typical Backdoors & Payloads (e.g. function containing the string "grant dba to")
- Found an Oracle vulnerability. Under special circumstances (materialized view), Oracle was resetting the password of the user outln and granted DBA privileges to this user.
  ➔ Fixed yesterday with the Oracle April 2008 CPU - Critical Patch Update

# Usage – Find Backdoors



1. Connect to the Oracle database

```
C:\> sqlplus system/mypw@mydb/mysid
```

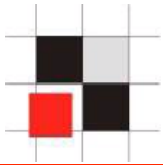
2. Run the script analyze_usage

```
SQL> @http://www.mycompany/analyzeactivity.sql
SQL> Enter username
SQL> generating report activity_analysis.txt…
```

3. Check and analyze the results

```
notepad backdoor_analysis.txt
```

# Usage – Find Backdoors



## Scanned databases

| Database Name | Signature | Result |
|---|---|---|
| ora10203 | signatures\ora10203_sig.csv | failed |

## 3: BD028 - Oracle Directory pointing to C:\ (WIN) detected. in ora10203

| OWNER | DIRECTORY_NAME | DIRECTORY_PATH |
|---|---|---|
| SYS | MYDIR | C:\ |
| SYS | MY | C:\ |

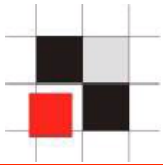
## 3: BD059 - Table(s) with whitespaces detected. in ora10203

| OWNER | TABLE_NAME |
|---|---|
| TEST | \|\| XMLREF() \|\| |

## 2: BD069 - PL/SQL code which contains "grant dba" in ora10203

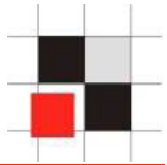
| NAME | NAME | OBJ# |
|---|---|---|
| TEST | V1 | 54106 |

## Other promising approaches



- Search the log-tables (e.g. SYS.AUD$ or custom tables) for holes in rowids.
- Every row in a table contains a unique number called rowid which can not be modified (assigned by the system) and not affected by updates
- Rowid are based on a sequence.
- Deleting an entry from a log-table creates a hole

# Other promising approaches



What Rowid is missing?

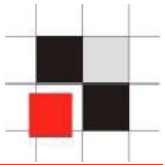
```
SQL> select rowid from sys.aud$;

ROWID
------------------
AAAAAIkAABAAAABDiAAA
AAAAAIkAABAAAABDiAAB
AAAAAIkAABAAAABDiAAD
AAAAAIkAABAAAABDiAAE
```

Take the creation time of the first entry and the creation time after the modification and you know in what timeframe the entry was removed.

Lookup with logminer/flashback query was was there before.

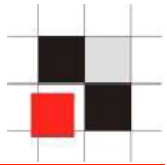
# Other promising approaches



The following SQL statement reports if there are holes in a table

```
select l.first + 1 hole, l.next - l.first - 1 distance, l.rowa , l.rowb
from
(select a.id  first,b.id  next ,a.rowid  rowa, b.rowid  rowb
 from mylogtable a ,mylogtable b
 where
      b.id   > a.id
  and b.id   = (select min(x.id)
                from mylogtable x
               where x.id   > a.id)
order by 1 ) l
where l.next - l.first  > 1
```

# Looking up in the memory (V$SQL) of the DB

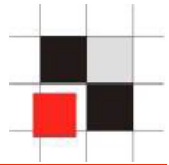


The memory view v$sqlarea contains a list of most SQL statements plus the number of executions/rows returned executed against the database in the last few minutes/hours.

If the incident happened few hours ago (which is rare) the statements could be still in this cache. Shutting down the database would destroy this data.

That's why you should preserve this data..

# Looking up in the memory (V$SQL) of the DB



```
create directory forensic1 as '\tmp';
  CREATE TABLE ext_write (sql_text)
  ORGANIZATION EXTERNAL
  (TYPE oracle_datapump
  DEFAULT DIRECTORY forensic1
  LOCATION ('vsqlarea.txt'))  PARALLEL AS
  SELECT sql_text from v$sqlarea;
```

## 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@red-database-security.com