Live-Hacking von Oracle-Datenbanken
Agenda

- Introduction
- Typical Database Attackers
- Exploits
- Countermeasure
Databases in the real world
The ivory tower architecture

Simple architecture
- Clients accessing a database via application server
- No direct access to the database
- Security and business rules are enforced in the application server
- Password change on database and application server
The ivory tower solution in the real world

- Complex architecture
- All types of clients are accessing the database
- Security and business rules only enforced in the first application server
- Passwords are stored in many places. Normally not documented

Real World

- Some people must connect with special tools like TOAD
- New project
- Another project
- We need a reporting solution
- You have nice data we will use it
- We just do a database link
- Yet another project
How difficult is it to hack an Oracle database?
It depends...

• Easy:
  • Old or unpatched versions
  • Database not hardened (weak passwords, unsecure code, ...)
  • Many exploits

• Difficult:
  • Latest, fully patched version
  • Hardened database
  • Database Activity Monitoring running
  • Custom exploit needed
Sorted by Exploit Type

SQL Injection Basics

- [Introduction to SQL Injection](#) via SQL Shell (e.g. SQL*plus)

Privilege Escalation

- [mdsys.reset_progs_index](#) (bug, 10.2, 11.1, 11.2)
- [dbms_job](#) (bug, 10.2)
- [dbms_sql_hash](#) (bug, 10.2)
- [dbms_cdc_publish](#) (bug, 10.1, 10.2, 11.1, 11.2)
- [dbms_cdc_publish](#) (bug, 10.1, 10.2, 11.1, 11.2)
- [dbms_jvm_exp_perm & dbms_java](#) (bug, 10.2)
- [dbms_jvm_exp_perm & dbms_java](#) (bug, 11.1-11.2)
- [alter_session set NLS](#) (bug, 8-10.2)
- [sys.dbms_metadata.get_granted_xml](#) (bug, SQL)
- [sys.dbms_metadata.get_xml](#) (bug, SQL)
- [sys.dbms_metadata.get_granted_xml](#) (bug, SQL)
- [sys.dbms_metadata.get_ddl](#) (bug, SQL)
- [sys.dbms_cdc_subscribe](#) (bug, SQL)
- [sys.dbms_export_extension](#) (bug, SQL)
- [sys.dbms_cdc_imdmp](#) (bug, SQL)
- [sys.kunmimp](#) (bug, SQL)
- [sys.kunlmk/work](#) (bug, SQL)
- [sys.kumnisfr](#) (bug, SQL)
- [sys.ul.findrcset](#) (bug, SQL)
- [sys.it.createworkspace](#) (bug, SQL)
- [wmsys.it.createworkspace](#) (bug, SQL)
- [sys.it.removeworkspace](#) (bug, SQL)
- [wmsys.it.removeworkspace](#) (bug, SQL)
- [ctsys.dlload](#) (bug, SQL)
- [xdb.xdb_pitrn_pig](#) (bug, SQL)

Bypass Access Rights

- [Bypass_access privileges using xmldb_transform](#) (bug, XMLDB, HTTP)
- [Bypass_access privileges using inline views](#) (bug, 8-10g)
- [Bypass_access privileges using normal views](#) (bug, 8-10g)
- [Bypass_access privileges using ANSI join](#) (bug, 9.1)
Who attacks a database?
Classification of Attackers

- Curious DBA or Employee
- Criminal employee
- Leaving employee
- External hacker
- Intelligence agency / Organized crime
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 politicians)
- **Identification:** Mostly select statements, Few/No traces without audit, Difficult to spot
Curious DBA or Employee

Example:

- Search data of colleagues

  SQL> select * from hr.emp
  where salary > 10000;

Example:

- Search data of celebrities

  SQL> select * from customers
  where lastname='Cruiser'
  and prename = 'Tom';

Tom Cruiser, 27.12.1963,
Account 123, 123.00
Curious DBA or Employee

**Example: (Demo)**

- Change identity (all versions of Oracle)

```sql
SQL> exec kupp$proc.change_user('HR');
```
### Privilege Escalation

**System ID**: 1138  
**Name**: Privilege Escalation in package SYS.KUPP$PROC; ID:1138  
**Description**: A Privilege Escalation is an attack in which a malicious user gains privileges they previously did not have.  
A vulnerability exists in Oracle 9, Oracle 10 and Oracle 11 which can be exploited to perform privilege escalation.  
The vulnerability is in procedure CHANGE_USER of package SYS.KUPP$PROC.  

**External References**:  

**Official patch**: CPU Jul2008  
**CVE**: CVE-2008-2602  
**CVSS**: 4.6  

**Exception(s)**:  
**Add Exception**

| Action                  |  
|-------------------------|---|
| Send alert              | HIGH |
| McAfee Database Security Console |  
| SNMP Trap               |  
| Twitter                 |  
| Terminate user session  |  
| Quarantine user for     | 50 min. |
Countermeasure

- Use McAfee Database Activity Monitoring to audit sensitive data
- Use and audit fake data (honey table) to catch curious people
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
Example

- Reset bill of friends aka “Friends & Family”
  
  SQL> update billing set amount=34 where userid=47111;
  
  ➔ Monitor direct updates without using the application

- Change Health Insurance account number and bypass SAP completely

  SQL> update sapr3.tsdlk
  set blzsz='50550020', KNRZS = '35921'
  where KUSCH=17;
  
  ➔ Monitor the integrity of sensitive data
Example 3

It is normally easy to follow financial transactions. That’s a challenge in (perfect) computer crimes. The following approach steals money without leaving financial traces. The attacker is not stealing money, instead of he is deleting his debts.

- Apply for credit for a house (e.g. 350,000 EUR)
- Get the money from the bank and buy the house
- Pay the rates for the credit for a few months.
- Set the credit to zero.
Countermeasure

**Example:**

- Use McAfee Database Activity Monitoring to audit/monitor sensitive data

- Use McAfee Security Scanner for Databases to search sensitive data (Data Discovery)
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
Leaving Employees

Example

- Extract sensitive data (e.g. using Excel, normal reports...)
  
  select * from customers

- Export entire Database (especially developers)

  exp.exe userid=grips/grips@grips full=y
Countermeasure

Example:

- Use McAfee Database Activity Monitoring to audit sensitive data or export utilities
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
Example – SQL Injection
Countermeasure

Example

- Use McAfee Database Activity Monitoring to audit sensitive data and typical views/tables used in an attack (e.g. DBA_TAB_COLUMNS)
Intelligence Agency / Organized Crime

- **Type:** Intelligence Agency / Organized Crime
- **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
Intelligence Agency / Organized Crime

Examples

- Buy customer list with black money (Germany vs. Liechtenstein/Switzerland)
- Stuxxnet
More information & demos at the McAfee booth...
Thank you

- Contact:

Red-Database-Security GmbH
Bliesstr. 16
D-.66538 Neunkirchen
Germany