



Encrypting Data

Is it possible to prevent access?

Pete Finnigan, Principal Consultant

SIEMENS

Insight Consulting

Introduction

- My name is Pete Finnigan
 - I specialise in researching, auditing and securing Oracle databases
- I am going to keep it reasonably simple and not too technical
- I will cover a lot of ground in 45 minutes - agenda next
- I want to give an overview of some of the issues surrounding encrypting data in the database and data that is intended to be stored in the database
- I want to talk about how realistic it is to ensure that data is protected in an Oracle environment

Agenda

- The purpose of encryption
- Where to encrypt – network, OS, database, middle tier
- Solutions: Oracle solutions, free solutions, commercial solutions
- The issues with encryption
- Sniffing, memory dumps and package interception
- Backups
- Key management
- Network encryption and file system encryption
- Transparent Data Encryption (TDE)

Why encrypt – what is the purpose of encryption?

- Regulatory needs – PCI, Visa, SoX, many more
- Sensitive data needs to be protected - HIV, AIDS, Top Secret data, Data Protection Act
- Departmental requirements
- HMG requirements
- To hide intellectual property, patents, copyrights
- Many more reasons
- Understand the threat first before embarking on encryption methods and technologies

Can we hide data from the DBA?

- “Can I hide data from the DBA?”
- This is one of the most regular questions I see or I am asked personally.
- There are two points to this
 - First: if you need to hide data from the DBA per se, then you probably need to review procedures, policies, HR etc
 - Second: If there is a real reason to hide the data, e.g. HIV status, then you can legitimately take action
- Hiding data from a DBA is virtually impossible – long term with standard Oracle
- Two solutions come to mind
 - Ensure that DBA’s do not have SYS or like privileges and use designed accounts, data is protected by VPD and all access is audited. **This can be bypassed and there is a need to have DBA’s use SYS from time to time**
 - The other option is to consider the new Oracle product – Oracle Database Vault -
http://www.oracle.com/technology/deploy/security/db_security/database-vault/index.html

Where to encrypt

- Consider and identify the data to be protected
- Review the complete data flow from source to destination
- Consider how the data enters the application / database
- Consider where it is used
- Consider where it is stored – database, files, data files, reports
- Consider how the data leaves the database – reports, extracts, feeds, backups, test databases (replication)
- Therefore, encrypt in the database, file system, storage media, backups, network...

Hackers like encrypted data

- Encrypted data is similar to protective marking
- Protective marking identifies key data (e.g. OLS)
- Encryption often also marks data
- Encrypted data can often be identified because it is encrypted
- Hackers will target encrypted or marked data as it says “**I am valuable data**”
- Consider dilution principals instead
 - If key data cannot be identified then do not mark it or encrypt it if the threat of marking is deemed to be higher than not encrypting

Some sample data

Name	Card_Number
=====	=====
Finnigan	5150879065437765
Kornbrust	5573578909861234
Litchfield	4853897665349861

- This is an example of a simple encryption scheme I have seen in a real system
- What's wrong with it (clues: column name, algorithm used, name and card stored together)

What solutions are available?

- Oracle database based solutions
 - DBMS_OBFUSCATION_TOOLKIT
 - DBMS_CRYPTO
 - Free packages, RC4, Blowfish available on the Net
- Free external solutions
 - C libraries for all encryption algorithms are available. E.g. <http://www.openssl.org/> includes code for most algorithms. These can be accessed via external procedures
 - Java classes for most algorithms are also available and can be used externally or from PL / SQL

What solutions are available (2)?

- Commercial solutions to protect data in the database
- Oracle Password Repository-
<http://sourceforge.net/projects/opr>
- Oracle Advanced Security Option (ASO)
- OpenSSL - <http://www.openssl.org/>
- OpenSSH – <http://www.openssh.com>
- Transparent Database Encryption

DBMS_OBFUSCATION_TOOLKIT

- Supports
 - Single DES
 - Triple DES
 - MD5
 - Generate DES keys desgetkey and des3getkey
- Issues:
 - Key generation but no key handling or key security
 - Oracle expects **you** to manage keys
 - No padding or chaining modes built in
- Oracle recommend to use DBMS_CRYPTO

DBMS_CRYPTO

- Supports
 - Single DES, 3DES, 3DES 2 key
 - AES
 - RC4
 - MD4, MD5 and SHA-1 hashes
 - HMAC_MD5 and HMAC_SH1
- Provides a much better random key generator
- Provides padding support
- Easier to use than older package – just pass text and key
- Oracle still expects you to manage the key though!!

Example use of DBMS_CRYPTO

```
create or replace function des_crypt
(pv_text in varchar) return raw is
  lv_key raw(128);
  lv_text raw(2000);
begin
  lv_text:=sys.utl_i18n.string_to_raw(
    pv_text,'AL32UTF8');
  lv_key:=sys.utl_i18n.string_to_raw(
    sys.dbms_crypto.randombytes(16),'AL32UTF8');
  return(sys.dbms_crypto.encrypt(
    lv_text,sys.dbms_crypto.DES3_CBC_PKCS5,lv_key));
end des_crypt;
/
```

Example use of DBMS_CRYPTO (2)

```
SQL> @crypt
```

Function created.

```
SQL> set serveroutput on size 1000000
```

```
SQL> exec dbms_output.put_line('Crypted text:  
' || des_crypt('test crypt'));
```

Crypted text:

```
8640FE54ED48429423E5EABB01AA3334
```

Free solutions and commercial solutions

- Jared Still has some good stuff -
<http://www.jaredstill.com/content/oracle-encryption.html>
- And
<http://web.archive.org/web/20050207112853/http://www.cybcon.com/~jkstill/util/encryption/encryption.html>
- <http://www.openssl.org> – most C algorithms
- DbEncrypt from Application Security Inc –
<http://www.appsecinc.com/products/dbencrypt/oracle/> -
- Encryption Wizard - Relational Database Consultants, Inc. -
http://www.relationalwizards.com/html/database_encryption.html

Password encryption

- Two issues with password encryption and use of passwords
 - Database passwords can be leaked in memory, text files, SQL and PL/SQL code, from the network
 - When an application also includes authentication then passwords can be leaked from binaries, middle tier, clients, network, in the database (shared memory, tables...)
- Oracle Password Repository (OPR) - <http://sourceforge.net/projects/opr>
- Carefully design authentication systems

An example of weak encryption

```
SQL> select view_username,sysman.decrypt(view_password)
      2 from sysman.mgmt_view_user_credentials;
```

```
VIEW_USERNAME  SYSMAN.DECRYPT(VIEW_PASSWORD)
-----
MGMT_VIEW      A4F5F18AD3B5080A307182A4EE3936
```

```
SQL>select credential_set_column,sysman.decrypt(credential_value)
      2 from sysman.mgmt_credentials2;
```

```
CREDENTIAL_SET_COLUMN  SYSMAN.DECRYPT(CREDENTIAL_VALUE)
-----
UserName               dbsnmp
Password               manager
```

Thanks to Alex for
the idea

An example of weak encryption (2)

- The previous example shows that the database password for the MGMT_VIEW user is stored in the SYSMAN schema.
- Other database usernames and passwords are also stored
- Metalink passwords are also stored
- The PL / SQL behind the SYSMAN schema is wrapped with the 9i wrapper – why?
- There is an easy to use decrypt function!
- The encryption seed is also stored in clear text
- Some lessons on how not to store critical data

9i and below wrapped PL / SQL weaknesses

```
SQL> create or replace procedure encode (credit_card in varchar2,  
    str out varchar2) is  
  2  key varchar2(16):='01234567890ABCDEF';  
  3  begin  
  4  null;  
  5  end;  
  6  /
```

Procedure created.

```
SQL> save encode.sql  
{snipped}
```

```
G:\code>wrap iname=encode.sql oname=encode.plb
```

```
PL/SQL Wrapper: Release 9.2.0.1.0- Production on Fri Jun 23 15:43:47  
2006
```

```
Copyright (c) Oracle Corporation 1993, 2001. All Rights Reserved.
```

```
Processing encode.sql to encode.plb
```

```
2 :e:  
1ENCODE:  
1CREDIT_CARD:  
1VARCHAR2:  
1STR:  
1OUT:  
1KEY:  
116:  
101234567890ABCDEF:
```

Hacking wrapped PL / SQL – pre-9i

- The symbol table is visible
- For the previous example it is possible to:
 - Deduce the purpose of the procedure
 - Find out the encryption algorithm used using `DBA_DEPENDENCIES` unless it is implemented internally to the procedure
 - Decrypt Credit Cards – in this case
- Critical code values can be read – the key
- Wrapped source can be modified without un-wrapping
 - Example: Fixed `DBMS_OUTPUT` limits problem
- PL/SQL can be unwrapped

Issues with encryption

- Sniffing network connections
- Memory dumps – can reveal keys – TDE had a number of bugs initially – idea was based on the library cache issue but using optimizer trace and dumpsga.
- Package interception to steal keys or data
 - Packages can be replaced / trojaned
- Package interception to steal computed keys
- Backups
 - If not encrypted the data can be read
 - If encrypted there are issues in maintaining old keys
- Key management is the biggest problem for encryption

Sniffing

- What is sniffing?
- What can you sniff?
 - Keys, data, passwords, much more
- Package interception is also a form of sniffing
- Capture the package used, key passed in, data passed in
- Tools for network packet capture:
 - ethereal
 - Sql*net trace
 - Snoop on Solaris....
- Keyloggers – software or hardware based
- OCI and Java interception – OCISPY and P6spy

Sniffing an ALTER USER

```
TRACE_FILE_SERVER=oug.trc  
TRACE_DIRECTORY_SERVER=d:\temp  
TRACE_LEVEL_SERVER=SUPPORT
```

➡ Add to the sqlnet.ora file

```
SQL> alter user scott identified by secretpassword;
```

User altered.

🔍 In the trace file you will find the password

```
[19-SEP-2005 14:29:52:814] nsprecv: 00 00 00 00 00 2D 61 6C | .....-al |  
[19-SEP-2005 14:29:52:814] nsprecv: 74 65 72 20 75 73 65 72 | ter.user |  
[19-SEP-2005 14:29:52:814] nsprecv: 20 73 63 6F 74 74 20 69 | .scott.i |  
[19-SEP-2005 14:29:52:814] nsprecv: 64 65 6E 74 69 66 69 65 | dentifie |  
[19-SEP-2005 14:29:52:814] nsprecv: 64 20 62 79 20 73 65 63 | d.by.sec |  
[19-SEP-2005 14:29:52:814] nsprecv: 72 65 74 70 61 73 73 77 | retpassw |  
[19-SEP-2005 14:29:52:814] nsprecv: 6F 72 64 01 00 00 00 01 | ord..... |
```

Memory dumps

- When package / view based encryption is used there can be two issues:
 - Keys passed and used could be grabbed from memory
 - The SQL or PL / SQL statements can be read from the SGA and may contain decrypted data or keys
- Any user with ALTER SESSION (default on most systems) can dump most memory structures and use trace
- Oradebug, orapatch, bbed can all be used to access data in memory
- Direct SGA access is possible – see <http://www.petefinnigan.com/other.htm> for a few papers

Dump example

```
SQL> alter session set events 'immediate  
trace name library_cache level 10';
```

Session altered.

```
SQL>
```

- Dump commands can dump most memory and file structures
- Trace will show SQL and binds
- Hackers are not ethical, ensure when you use encryption that nothing is visible!

Package Interception

- What is package interception?
- If encrypt / decrypt is implemented via PL / SQL packages then it could be possible to first call your own package (log the keys, clear data...) and then call the original package
- I talked about this in “Exploiting and protecting Oracle” in 2001 (not for key stealing!)
- This works because Oracle resolves
 - Local packages first, then private synonyms, then public, then the real object
 - This is the same ideas as Oracle root kits – See Alex Kornbrust BH 2005 and 2006
- Programmed keys could also be stolen

Package interception example

- Create local `dbms_crypto`
- Add code to store the parameters passed to `DBMS_CRYPTO` or write them to a file or network device
- Call the original `DBMS_CRYPTO` passing the arguments to it
- Install the local package
- Can be done via a synonym (private or public)
- Block the issue by ensuring applications call encryption packages with a full path
- The same issue applies to computed keys

Hiding parameters in PL/SQL calls

```
create or replace function test_param(ipstr in
  varchar2, ks in varchar2) return varchar2 as
  input_str varchar2(8):=''; output_str
  varchar2(16):=''; key_str varchar2(8):='';
begin
  input_str:=ipstr; key_str:=ks;
  dbms_obfuscation_toolkit.DESEncrypt(
  input_string => input_str,
  key_string => key_str,
  encrypted_string => output_str);
  return output_str;
end; /
```

- See

http://www.petefinnigan.com/ramblings/dbms_obfuscation_toolkit.htm

Backups

- TDE is supported through RMAN. Three modes supported:
 - Transparent mode (default)
RMAN> configure encryption for database on;
 - Password encryption
RMAN> set encryption on identified by pwd;
 - Dual mode
- Oracle Secure backup
- Various third party solutions, hardware and software
- TDE / RMAN has the advantage of being selective
- Issues with old keys

Key management

- For the built in solutions (DBMS_CRYPTO and DBMS_OBFUSCATION_TOOLKIT) Oracle does not provide any key management support
- Free solutions, free PL / SQL, Java or C external procedures also do not provide key management out of the box
- Commercial solutions provide key management
- Keys can be fixed or computed
- Key management can be handled in many ways:
 - With the client
 - The server file system
 - In the database

Key management in the client

- The user must pass the key to the encryption routines (in the database?)
- The key could be
 - Typed in
 - Held in a file
 - Held in the registry
 - Compiled into a binary
- This implies that
 - The key is known to many people
 - The key could be hard coded into many clients

Key management on the file system

- The key could be held on the server file system
- This is better as it's held in one place
- Would need to protect from the DBA – external file system?
- The database would need to access the key
 - External procedure
 - Database file read
- The DBA could access the key as its read in
- More secure than the client
- The model could be extended to a secure device

Key management in the database

- The key would be stored in:
 - A database table
 - A secure global context
- The DBA can read the key in most circumstances
- The key could be held “with” the data
- Changing keys could be easier
- If held in memory the key could be read from shared memory segments
- Seems less secure than the file system
- A model could work where each row has a key and the keys are “unlocked” with a global pass phrase entered at start up
- Essentially the model used for TDE

Network and file system encryption

- There are a number of possibilities for network encryption
- Advanced Security Option (ASO) is from Oracle and offers a number of levels of encryption and algorithms. SSL is also available
- OpenSSL – see white papers page
- Windows EFS -
<http://www.microsoft.com/technet/prodtechnol/winxppro/deploy/cryptfs.mspx>
- No consistency for Unix systems

Transparent Database Encryption – an overview

- Available only with the ASO – costly?
- Protects the data on the storage media
- Operates on the database table columns
- The column keys are stored in the data dictionary
- A master key is held in a wallet and entered on startup
- Columns to encrypt can be chosen
- How does it work?

TDE – an overview

- Set up a wallet directory in the sqlnet.ora
- Set a master key

```
ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY  
  "PWD" ;
```

- This creates a default wallet
- Open the wallet

```
ALTER SYSTEM SET ENCRYPTION WALLET OPEN  
  IDENTIFIED BY "PWD" ;
```

- Add encryption to a table

```
ALTER TABLE EMP MODIFY (ENAME ENCRYPY NO  
  SALT) ;
```

Conclusions

- Design carefully
- Consider the data flow, the data in transit and the data at rest
- Use secure storage for keys – stick, bio, THALES, Eracom
- If writing in PL / SQL use a single package that accepts no inputs. If keys are not passed then they cannot be stolen.
- Don't rely on wrapping PL / SQL
- Use full paths to any system packages
- You cannot protect data from a DBA if internal solutions are used
- The internal packages cannot be used to secure data. It is impossible to have secure key management

Questions and Answers

- Any questions, please ask
- Later?
 - Contact me via email peter.finnigan@siemens.com
 - Or via my website <http://www.petefinnigan.com>



www.siemens.co.uk/insight

+44 (0)1932 241000

Insight Consulting

Siemens Enterprise Communications Limited

**Security, Compliance, Continuity
and Identity Management**

SIEMENS



Choose with confidence
use with ease



INVESTOR IN PEOPLE



013