Data Scrambling in Non-PROD Cloned Instances

John Peters
JRPJR, Inc.
john.peters@jrpjr.com

Before We Start A Quick Audience Survey

- How many of you have are on 11.0, 11i, 12?
- How many of you plan to upgrade to R12 in the next 18 months?

What I am going to cover

- Why obfuscate sensitive data and what is sensitive data
- OEM Application Management Pack
- A custom sensitive data backup and obfuscate methodology with sample code
- Review sensitive table columns

Examples of Sensitive Data

- Employee Data
  - Social Security Number
  - Salary Information
  - Review Information
  - Address and Phone Information
  - Age Information
  - Direct Deposit Bank Account Information
- Customer Data
  - Credit Card Numbers
- Vendor Data
  - Direct Deposit Bank Account Information
- Company Data
  - Bank Account Information

Why Obfuscate Sensitive Data

- Non-PROD instances have a lower level of access control
  - APPS password available to wider group
  - User responsibility control less restrictive
  - Data is sent to Oracle Support during debug
  - Non-Employees often have access to data, contract developers, consultants, etc.

Obfuscate Techniques

- Masking (done usually in UI)
  - Credit Card: NNNNNNNNNNNN1234
  - SSN: NNN-NN-1234
- Substitution
  - Substitute Digits and Characters with a constant
- Purge
  - Null out the sensitive data
- Giberish
  - Replace with random characters

Use Oracle Supplied Solutions

- When possible take advantage of Oracle Supplied Solutions to obfuscate sensitive data. They are already included in your licensing and are supported.

Example:

Credit Card Data

- Apply the credit card data encryption patches
- This secures data in both PROD and non-PROD instances
- Secures data in the UI and the Database
- "It's the law"

OEM Application Management Pack

- A data scrambling/purge framework introduced in version 2.0
- A generic engine that allows you to specify tables and columns of data that should be scrambled during a clone.
- Irreversible purge or scramble of the data.
- There is no seeded data. You need to decide what is scrambled.

What I Implemented

- A customization that allows sensitive data to be:
  - Backed Up
    - Original Data is stored in custom tables
    - Data in encrypted for security
    - Allows for a table by table reversal of obfuscation based on non-PROD instance testing requirements
  - Sensitive Data Obfuscated
    - Original numeric data is replaced by a 9
    - Character strings replaced by constant string 'Z' or 'N'
    - Key data requiring uniqueness is replaced by ID values
    - Obfuscated data is easily identifiable to users
    - Magnitude of data is still available "six figure salary"

Implementation

Two components:

- A PL/SQL Package to Encrypt/Decrypt Data
- An SQL Script that can be run during cloning that backs up and obfuscates source data

PL/SQL Package - GENERATE_KEY

- Generates a 64 character key value to encrypt/decrypt values
- Save this value in a safe location if you want to reverse encryption

```
select DBMS_CRYPTO.randombytes(256/8)from dual;
```

PL/SQL Package - DECODE_VARCHAR

```
FUNCTION DECODE_VARCHAR (p_in    in raw,
                         p_key   in raw)
RETURN VARCHAR2
IS
  l_ret     varchar2 (2000);
  l_dec_val raw (2000);
  l_mode     number := dbms_crypto.ENCRYPT_AES256
                     + dbms_crypto.CHAIN_CBC
                     + dbms_crypto.PAD_PKCS5;
BEGIN
  l_dec_val := dbms_crypto.decrypt (p_in,
                                    l_mode,
                                    p_key);
  l_ret:= UTL_I18N.RAW_TO_CHAR(l_dec_val, 'AL32UTF8');
  return l_ret;
END DECODE_VARCHAR;
```

PL/SQL Package - ENCODE_VARCHAR

```
FUNCTION ENCODE_VARCHAR (p_in    in varchar2,
                         p_key   in raw)
RETURN RAW
IS
  l_enc_val raw(2000);
  l_mode     number := dbms_crypto.ENCRYPT_AES256
                     + dbms_crypto.CHAIN_CBC
                     + dbms_crypto.PAD_PKCS5 ;
BEGIN
  l_enc_val := dbms_crypto.encrypt(UTL_I18N.STRING_TO_RAW(p_in,
                                                          'AL32UTF8'),
                                   l_mode,
                                   p_key);
  return l_enc_val;
END ENCODE_VARCHAR;
```

DBMS_CRYPTO

- In order to use DBMS_CRYPTO in the APPS schema you must first grant execute access to it.

```
sqlplus / as sysdba

  grant execute on dbms_crypto to APPS;

  create synonym apps.dbms_crypto for sys.dbms_crypto;
```

SQL Script

The script consists of three simple steps repeated on each table with Sensitive Data

1. Create Backup Table
2. Insert Sensitive Data Records into Backup Table
3. Obfuscate Sensitive Data

Runs as APPS during instance clone

SQL Script – Create Backup Table

- Columns in backup table
  - Primary Key of Source Table
  - Sensitive Source Data to Backup

```
create table XX_CUS.ENC_DATA_01
 (PERSON_ID NUMBER,
  EFFECTIVE_START_DATE DATE,
  EFFECTIVE_END_DATE DATE,
  ENC_NATIONAL_IDENTIFIER RAW(2000)
 );
```

SQL Script – Insert into Backup Table

```
insert into xx_cus.ENC_DATA_01
 (PERSON_ID,
  EFFECTIVE_START_DATE,
  EFFECTIVE_END_DATE,
  ENC_NATIONAL_IDENTIFIER
 )
select PERSON_ID,
       EFFECTIVE_START_DATE,
       EFFECTIVE_END_DATE,
       XX_CUS_CLONE_UTILITY.ENCODE_VARCHAR(NATIONAL_IDENTIFIER,
                                           &&KEY) enc
  from PER_ALL_PEOPLE_F;
```

SQL Script – Obfuscate Data

```
update PER_ALL_PEOPLE_F a
   set NATIONAL_IDENTIFIER = 'NNN-NN-NNNN'
 where substr(NATIONAL_IDENTIFIER,4,1) = '-'
   and exists (select 'Y'
                 from xx_cus.ENC_DATA_01 b
                where b.PERSON_ID = a.PERSON_ID
                  and b.EFFECTIVE_START_DATE = a.EFFECTIVE_START_DATE
                  and b.EFFECTIVE_END_DATE = a.EFFECTIVE_END_DATE
              );
```

SQL Script – Restore Data

```
update PER_ALL_PEOPLE_F a
   set NATIONAL_IDENTIFIER = select
   XX_CUS_CLONE_UTILITY.DECODE_VARCHAR(ENC_NATIONAL_IDENTIFIER,
                                       &&KEY) enc
                  from xx_cus.ENC_DATA_01 b
                 where b.PERSON_ID = a.PERSON_ID
                   and b.EFFECTIVE_START_DATE = a.EFFECTIVE_START_DATE
                   and b.EFFECTIVE_END_DATE = a.EFFECTIVE_END_DATE
 where exists (select 'Y'
                  from xx_cus.ENC_DATA_01 b
                 where b.PERSON_ID = a.PERSON_ID
                   and b.EFFECTIVE_START_DATE = a.EFFECTIVE_START_DATE
                   and b.EFFECTIVE_END_DATE = a.EFFECTIVE_END_DATE
               );
```

Sensitive Data Tables – Employees 11i

PER_ALL_PEOPLE_F.NATIONAL_IDENTIFIER

PER_PAY_PROPOSALS.PROPOSED_SALARY_N

PER_PAY_PROPOSAL_COMPONENTS.CHANGE_AMOUNT_N

PAY_ELEMENT_ENTRY_VALUES_F. SCREEN_ENTRY_VALUE

PER_PERFORMANCE_REVIEWS.PERFORMANCE_RATING

PAY_EXTERNAL_ACCOUNTS.SEGMENT1 to SEGMENT30

Sensitive Data Tables – Payables 11i

AP_BANK_ACCOUNTS_ALL.BANK_ACCOUNT_NUM

AP_BANK_BRANCHES.EFT_USER_NUMBER and EFT_SWIFT_CODE

AP_CARDS_ALL.CARD_NUMBER

AP_INVOICE_PAYMENTS_ALL.BANK_ACCOUNT_NUM

AP_CHECKS_ALL.BANK_ACCOUNT_NUM

Sensitive Data Tables - Credit Card Data 11i

ASO_PAYMENTS.PAYMENT_REF_NUMBER

IBY_CREDITCARD. CCNUMBER

OE_ORDER_HEADERS_ALL. CREDIT_CARD_NUMBER

OKS_K_HEADERS_B. CC_NO

OKS_K_LINES_B.CC_NO

CS_INCIDENTS_ALL_B.CREDIT_CARD_NUMBER

Sensitive Data Tables Summary

- You need to work with the functional users to try to find sensitive data in your version of the E-Business Suite based on how your company uses the E-Business Suite.
- Take a look at DBA_TAB_COLS
- Write scripts to look at all VARCHAR2 columns for sensitive data patterns
  - 16 characters (could be a credit card)
  - 3 '-' 2 '-' 4 Social Security Number

Don’t Forget Old Tables With Sensitive Data

As we upgrade from 10 to 11 to 11i to R12 Oracle leaves obsolete tables in the database. These still contain valid data in some cases.

Example:

SO_HEADERS_ALL was replaced by OE_ORDER_HEADERS_ALL and contains the column CREDIT_CARD_NUMBER

Things to Remember

- Take advantage of Oracle Supplied data obfuscation functionality first
- Ask user community to verify that all sensitive data has been found and obfuscated in non-PROD instance
- Preserve you encryption keys in a safe place, these are the keys to the kingdom

- My contact information:

  John Peters
  john.peters@jrpjr.com
  http://www.jrpjr.com

- Additional reference papers can be found at:
  http://www.norcaloaug.org
  http://www.jrpjr.com