Securing an Oracle Database

Noel Yuhanna

Forrester Research Inc.

NOCOUG – Feb 2004

Agenda

- Trends in DBMS
- Why Secure your database?
- DBMS Security Framework
- Oracle Database Security
  - Basic Security - Password, Roles, Views
  - Adv. Security - VPD, FGA, Encryption
- Best Practices
- Summary

Trends in DBMS

- Database sizes are growing
  - Terabyte sized DB's are common
- Automation – Oracle, IBM, Microsoft
  - Self-Tuning, Self-Healing, Self-Managing
- Expanding scope of DBMS
  - XML, Web Services, Utility Computing, RFID
- Database Consolidation continues
  - To save money
- Security concerns grown
  - Increased intrusion, regulatory requirements

What does a Database contain?

- Non-sensitive Data
  - Not so interesting …
- Sensitive Data
  - Credit Card Numbers
  - Employees Salary/Bonus/Health
  - Social-Security Numbers
  - Medical records
  - Tax Information
  - Criminal Record
  - Account Information

Why secure your database?

- External attacks have grown
  - Steal data / disrupt business
  - Worms/Viruses
  - Vulnerabilities on OTN > 60 listed
- Internal attacks continue
  - Difficult to monitor
  - 70% of intrusion’s are internal
  - 20% of clients claimed being hacked

Regulatory requirements

- HIPAA
- Sarbanes Oxley
- California SB 1386
- GLB – Gramm-Leach-Biley Act
- Visa security compliance
- American Express requirements

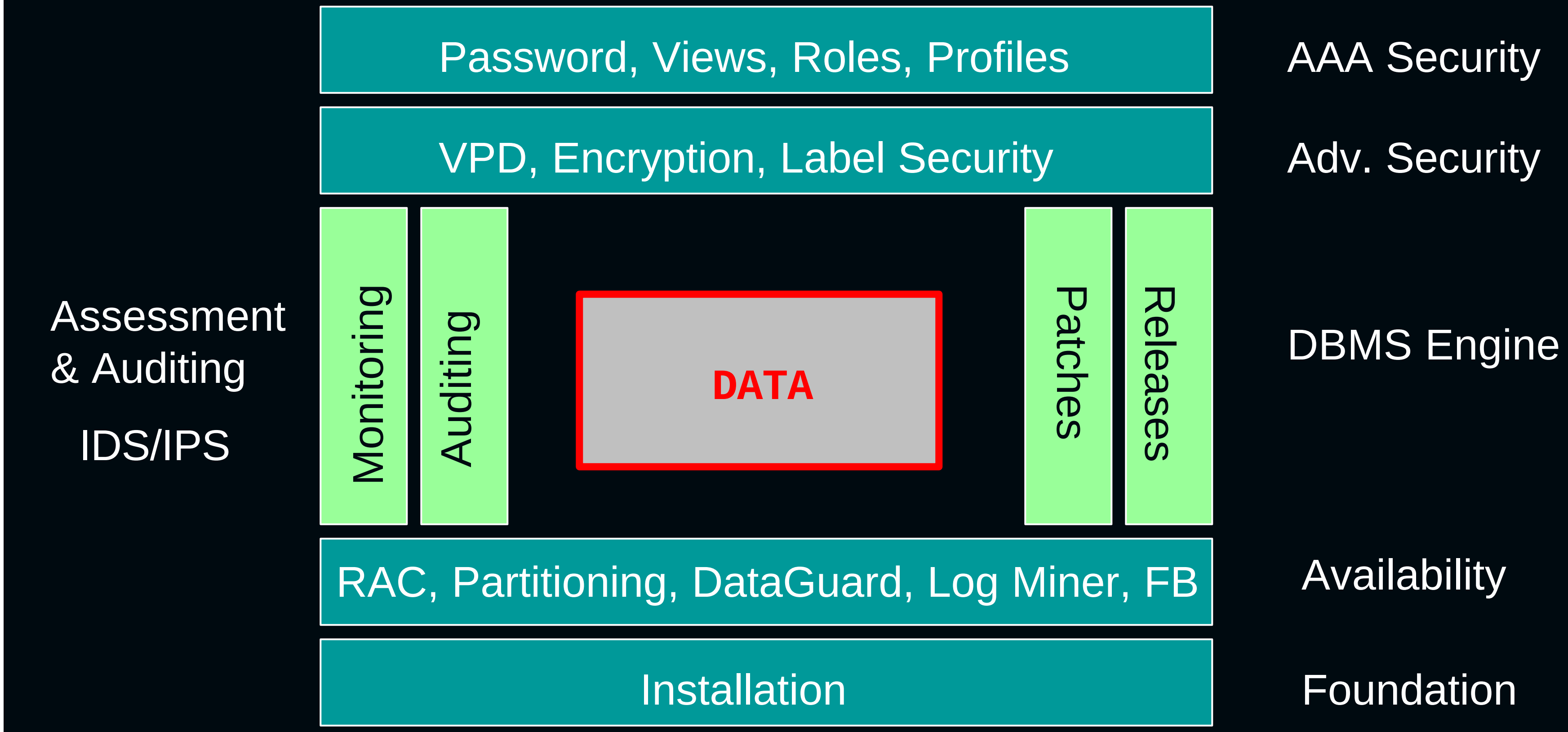
Risks – Business impact

- ✍ Law suits
- ✍ Loss of customer's confidence
- ✍ Loss of partner's confidence
- ✍ Impact in the revenue

Issues – DBMS and Admin

- DBMS software
  - DBMS bugs
  - OS bugs
  - Vulnerable services
- Administration
  - Default settings
  - Poor policies – roles, passwords,data access
  - Untrained DBA's
  - Insecure administration – backups, Test DB

DBMS Security Framework



Password, Views, Roles, Profiles

AAA Security

VPD, Encryption, Label Security

Adv. Security

Assessment & Auditing

IDS/IPS

Monitoring

Auditing

DATA

Patches

Releases

DBMS Engine

RAC, Partitioning, DataGuard, Log Miner, FB

Availability

Installation

Foundation

Security Standards?

Do not follow industry standards on Security

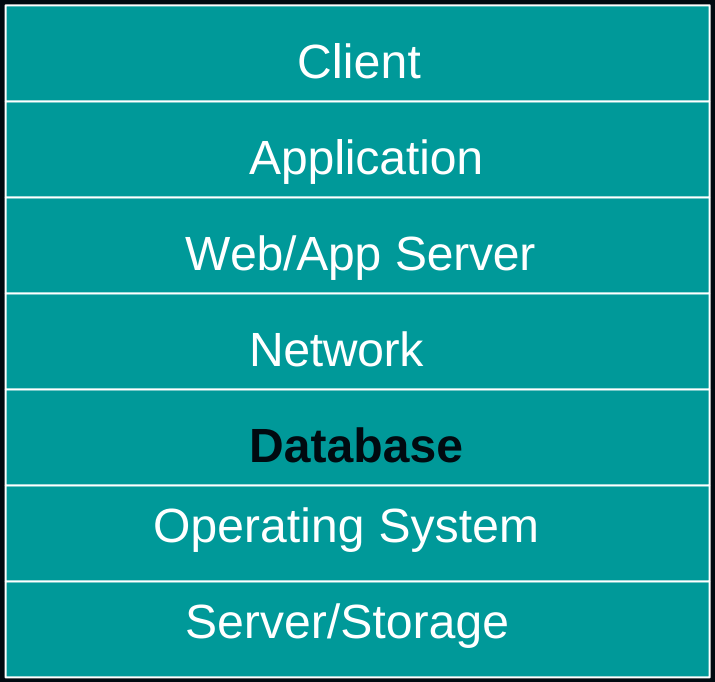
Create your own internal standards

Security is a continuous process, not a product

Develop a Security Plan “Its all about policies”

Database Layer

DBMS security is more than securing DB.



Client

Application

Web/App Server

Network

Database

Operating System

Server/Storage

How secure is your database?

- Production Database
- Test, Dev, Q&A, Stage – Databases
- Database backups – Tape, Disks

Database Installation

- Do not install options that are not needed
- Remove setup/install files created during Install.
- Disable all default user accounts – even Scott.
- Change system account passwords
- Disable system stored proc that are not used
- Remove privileges from PUBLIC on objects
- Control installation of Sqlplus/tools deployment
- Disable DBSNMP account if not used

Basic Security - Overview

- ✍ Password Management
- ✍ Using Profiles
- ✍ Creating Views
- ✍ Create Roles
- ✍ Listener Administration

Password Management

-Common vulnerabilities/attacks

-Blank passwords

-Weak Passwords

-Brute force attack

-Dictionary based attack

-Remove all default passwords

-Check for passwords in files

-Setup strong password policy for Admin & Users

Using Profiles

CREATE PROFILE ….. LIMIT

FAILED_LOGIN_ATTEMPTS	# of Attempts
PASSWORD_LIFE_TIME	# Days
PASSWORD_REUSE_TIME	# Days
PASSWORD_REUSE_MAX	# Changes
PASSWORD_LOCK_TIME	# Days
PASSWORD_GRACE_TIME	# Days
PASSWORD_VERIFY_FUNCTION	# Function

Example:

```
CREATE PROFILE app_user2 LIMIT
    FAILED_LOGIN_ATTEMPTS          5
    PASSWORD_LIFE_TIME             60
    PASSWORD_REUSE_TIME            60
    PASSWORD_REUSE_MAX             5
    PASSWORD_VERIFY_FUNCTION       verify_function
    PASSWORD_LOCK_TIME             1/24
    PASSWORD_GRACE_TIME            10;
```

Password Verification

UTLPWDMG.sql – password verification function

Checks:

1. If password is the same as username
2. If minimum length of password is x.
3. If password is simple. (checks words)
4. If password contains one letter & one digit.
5. If password differs from previous password by at least 3 letters.

User Account Lockout

```
CREATE PROFILE user_lockout_prof LIMIT
   FAILED_LOGIN_ATTEMPTS   5
    PASSWORD_LOCK_TIME      7;
```

No of tries

# of Days Locked

```
ALTER USER noel PROFILE user_lockout_prof;

ALTER USER noel ACCOUNT UNLOCK;
```

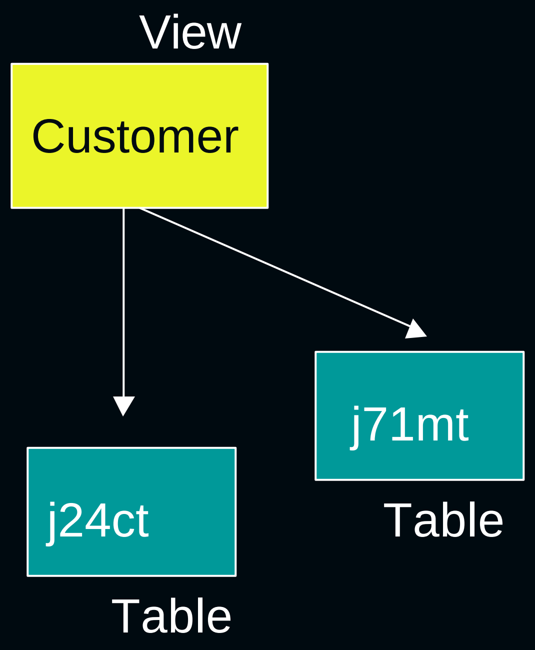
Views

-Minimize the use of direct table access

-Create views

-Table naming policy

-Hiding the base tables



Roles

- Collection of privileges
- Grant/Revoke roles
- Easier to manage

- Requires constant administration
- Use principle of least-privilege
- Setup policies on
  - Who, How, When, What

Listener

- Proxy between the client and database
- Is separate from the database
- Has its own commands and activities
- Has its own authentication and auditing
- Could stop access to database
- Buffer overflow attacks
  - Sending unexpected data in connection string
  - User=, Service=, command=x e.g.. Over 4096 chars.

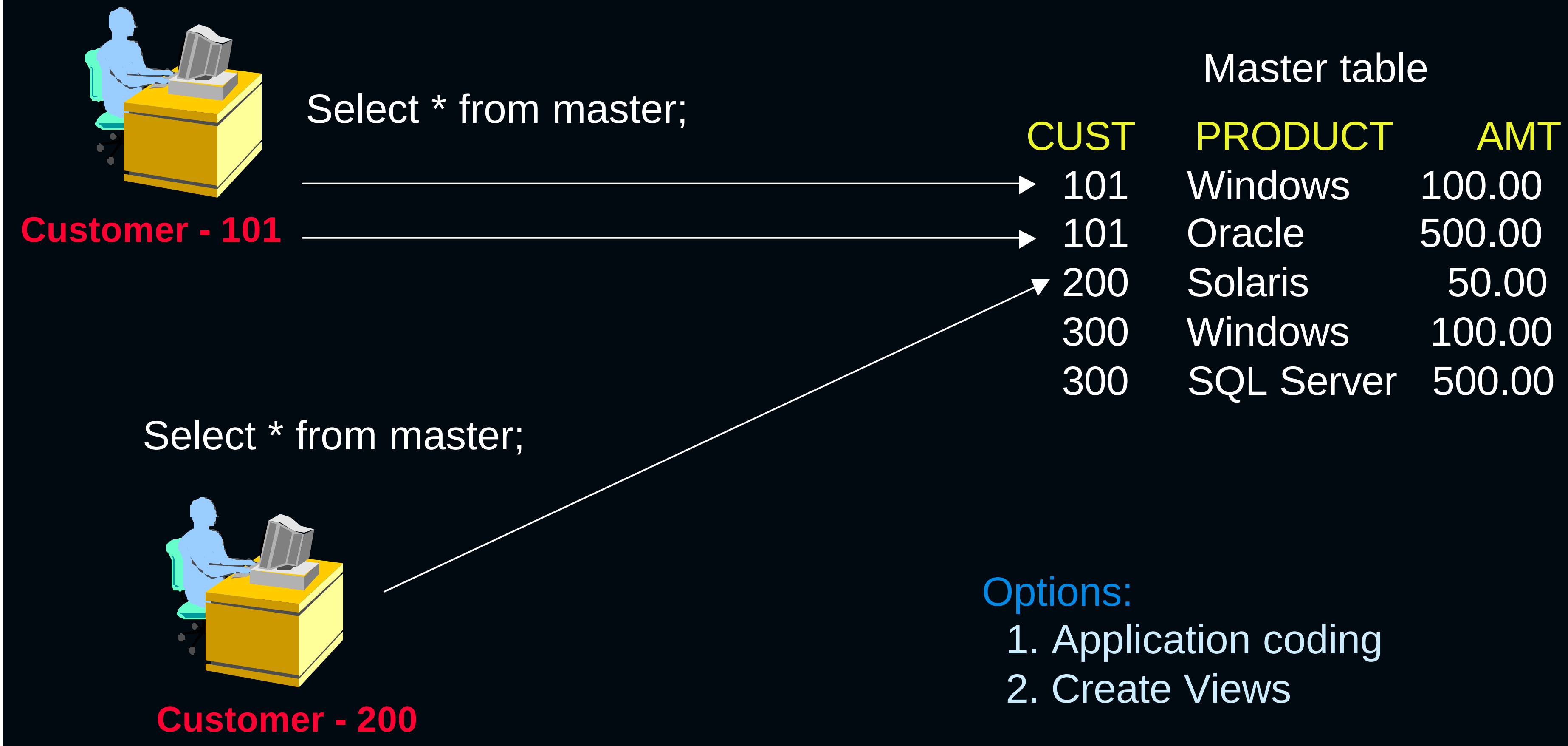
Listener - Recommendations

- Secure listener with a password
- Protect the listener.ora file
- Change the default port 1521/TCP
- Blocks all ports on firewall except port 80
- Use TCP network – is fastest and secure
- Use only network libraries needed, remove others
- Enable SSL encryption for highly sensitive DB
- Prevent unauthorized admin of Listener

Advanced Security - Overview

- Virtual Private Database (VPD)
- Label Security
- Data Encryption

Virtual Private Database (VPD)



Select * from master;

Select * from master;

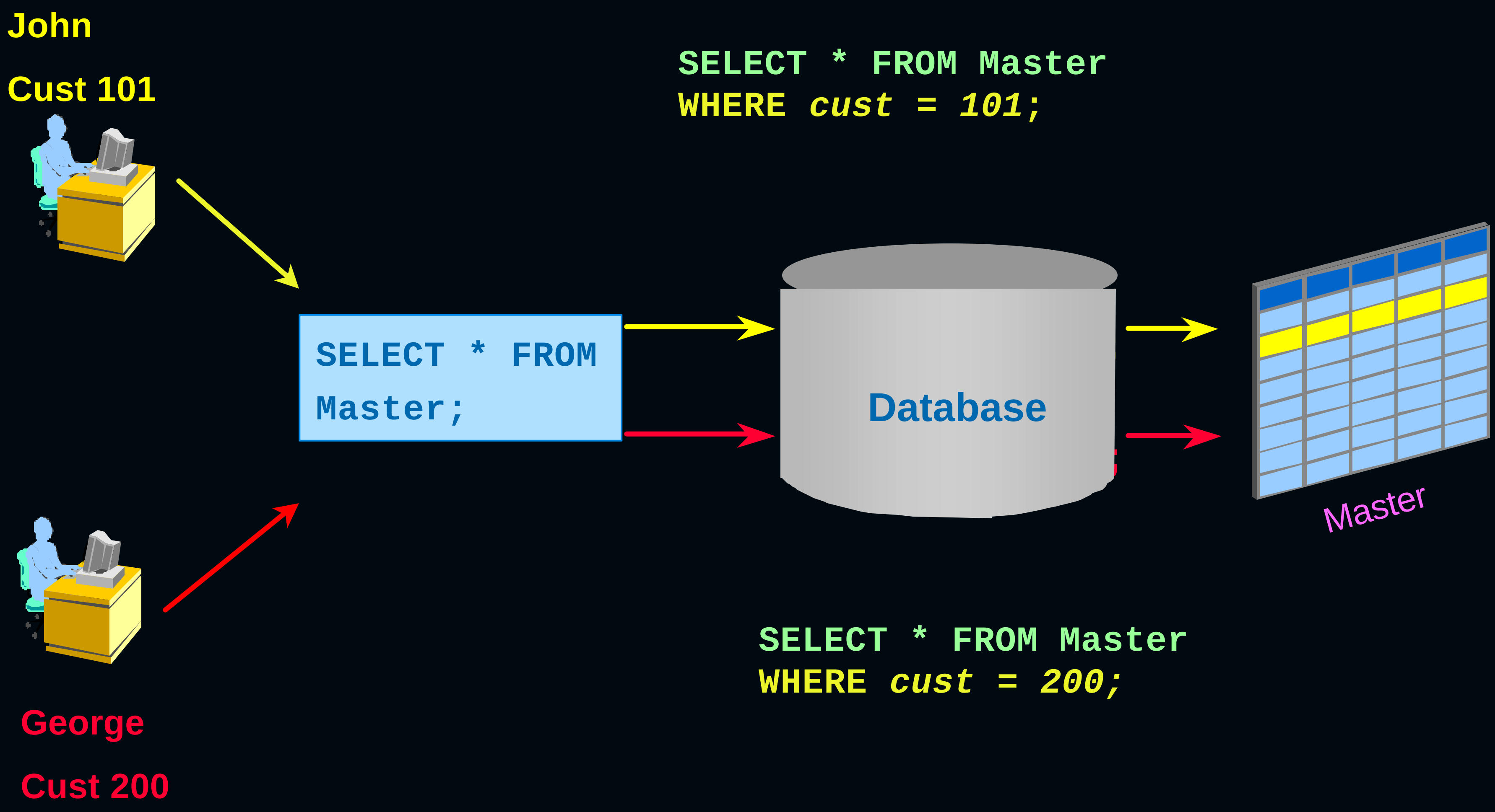
Master table

CUST	PRODUCT	AMT
101	Windows	100.00
101	Oracle	500.00
200	Solaris	50.00
300	Windows	100.00
300	SQL Server	500.00

Options:

1. Application coding
2. Create Views

Virtual Private Database (VPD)



John

Cust 101

```
SELECT * FROM Master
WHERE cust = 101;
```

```
SELECT * FROM
Master;
```

```
SELECT * FROM Master
WHERE cust = 200;
```

George

Cust 200

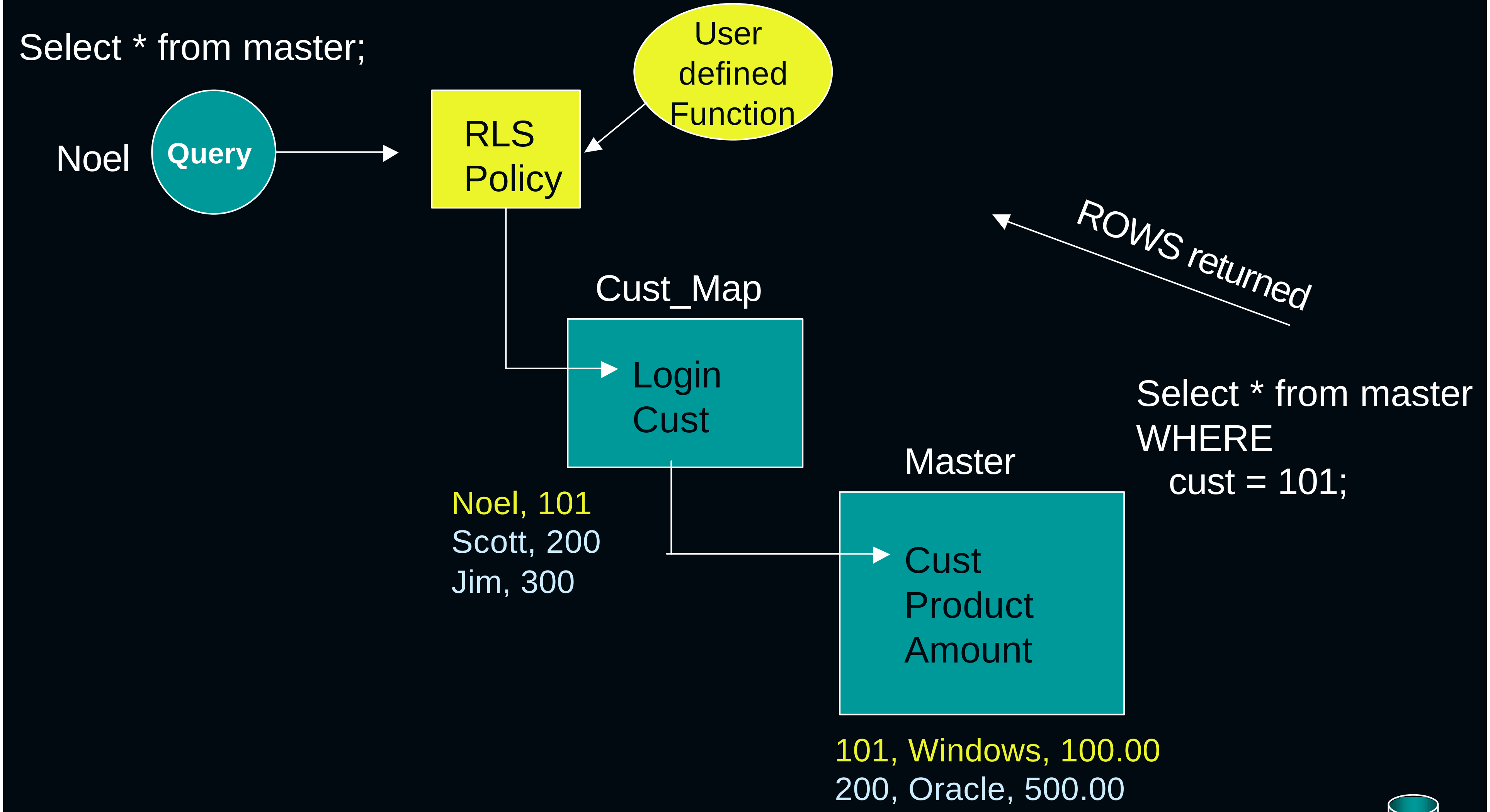
Virtual Private Database (VPD)

- Introduced in Oracle 8i
- Controls access to data
- Add policy to any Table/View
  - Bind PL/SQL pkg (DBMS_RLS) to Table
- Dynamically rewrites SQL
  - Query modification takes place
  - WHERE clause appended to SQL Stmt

Policy Function

- Takes two arguments
    - Table Owner
    - Table Name
- Return a valid predicate
- WHERE clause should not be returned

VPD Flow - Example



Select * from master;

Noel

Query

RLS
Policy

User
defined
Function

Cust_Map

Login
Cust

Noel, 101
Scott, 200
Jim, 300

Master

Cust
Product
Amount

101, Windows, 100.00
200, Oracle, 500.00

ROWS returned

Select * from master
WHERE
cust = 101;

Policy Function

```
CREATE or REPLACE FUNCTION get_master (
   v_table_owner in varchar2,  v_table_name in varchar2
)
return varchar2
IS
 customer_number number;
  my_predicate varchar2(80);

BEGIN
  SELECT cust into customer_number from CUST_MAP
   WHERE login = USER;

  my_predicate := ‘CUST = ‘ || customer_number;

  return my_predicate;
END;
/
```

Add a Policy

```
BEGIN
 DBMS_RLS.ADD_POLICY (
     Object_schema => ‘scott’,
     Object_name    => ‘master’,
     Policy_name     => ‘my_policy’,
     Policy_function  => ‘get_master’, (as shown in previous slide)
     Function_schema => ‘scott’,
     statement_types => ‘SELECT, UPDATE, DELETE, INSERT’
);
END;
/
```

Application Context

Named set of attributes/values

Default context is USERENV –name,host

Can define your own context

Set application context

DBMS_SESSION.set_context package

e.g.. SET_CONTEXT(‘HR_CTX’,’EMPID’, value);

-Fetch the application context in policy function

SYS_CONTEXT function:

e.g.. SYS_CONTEXT(‘USERENV’,’SESSION_USER’);

Benefits - VPD

Customize: Multiple policies per table

Scaleable: Rewritten queries are optimized

Flexible: Predicates generated dynamically

Transparent: No application changes

Security: Cannot bypass the policy

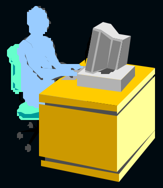
2-Tier/3-Tier: Works with any type apps

Lower Cost: Build once

Oracle Label Security

- Enterprise Edition Add-on Security Option
- Out-of-the-box, row level security
- Design based on Government req.
- Also used by commercial org.
- Data access is based on sensitivity labels and customizable enforcement options

Oracle Label Security (OLS)



Oracle Label Security Authorization:
Secret

OLS

Project Table

Project	Location	Department	Sensitivity Label	
AX703	Chicago	Finance	Unclassified	OK
B789C	Dallas	Engineering	Secret	OK
JFS845	Chicago	Legal	Top Secret	✗
SF78SD	Miami	Human Resource	Highly Confidential	✗

Label Components

Label =

Level plus

***Optional* Compartments plus**

***Optional* Groups**

In Military establishments:

TopSecret:US_Only:D20

Benefits - OLS

- Enables Data privacy by default
- Runs on all Operating systems
- Extends VPD
- No programming necessary
- Granular level of data security

Comparing VPD/OLS

VPD

- Part of Enterprise Edition
- You define security policy

OLS

- EE Security option
- Oracle provides security policy

How are they the same?

- Both supply API's
- OPM can manage both
- Suitable for hosting
- Centralized Security in database
- Restrict access at the row level

Database Encryption

- Selective encrypting sensitive data
  - Credit card numbers
  - Passwords
  - Personal Information – Health, Account, etc
- Options:
  - DBMS_OBFUSCATION_TOOLKIT PL/SQL
  - DBMS_CYRPTO – 10g
  - Third Party Vendors

- DBMS_OBFUSCATION_TOOLKIT is granted to PUBLIC by default

Encryption algorithms supported

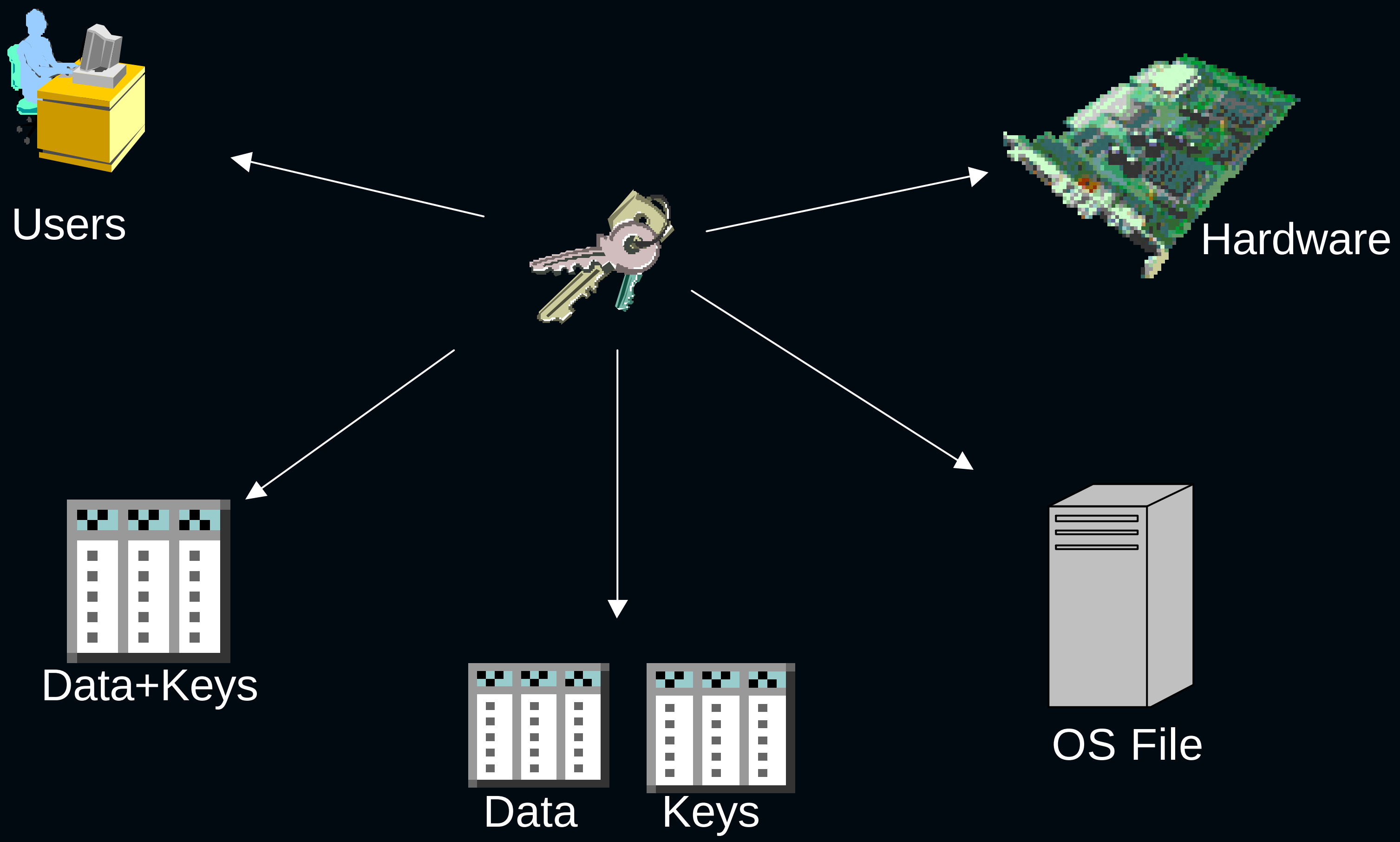
- Data Encryption Standard (DES)
- Triple DES (3DES)
- Advanced Encryption Standard (AES)
- MD5, MD4, and SHA-1 cryptographic hashes
- MD5 and SHA-1 Message Authentication Code (MAC)

DBMS_OBFUSCATION

```
DBMS_OBFUSCATION_TOOLKIT.DES3ENCRYPT (
  input_string =>
  key_string =>
  encrypted_data =>
);
DBMS_OBFUSCATION_TOOLKIT.DES3DECRYPT (
  input_string =>
  key_string =>
  decrypted_data =>
);
```

✍ Supports RAW and Varchar2 only

Where do you store the keys?

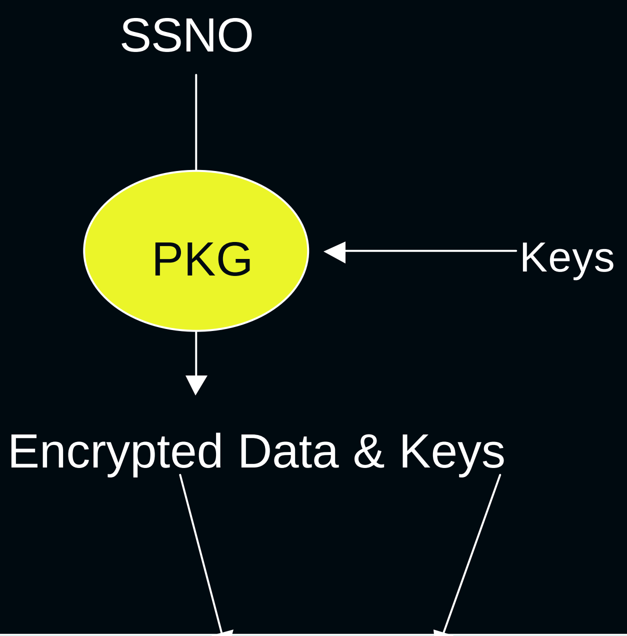


Encrypting Data

44557878	01234567890123456	=	ÍP? 9'
Data	Key		Encrypted Data
ÍP? 9'	01234567890123456	=	44557878
Encrypted Data	Key		Data

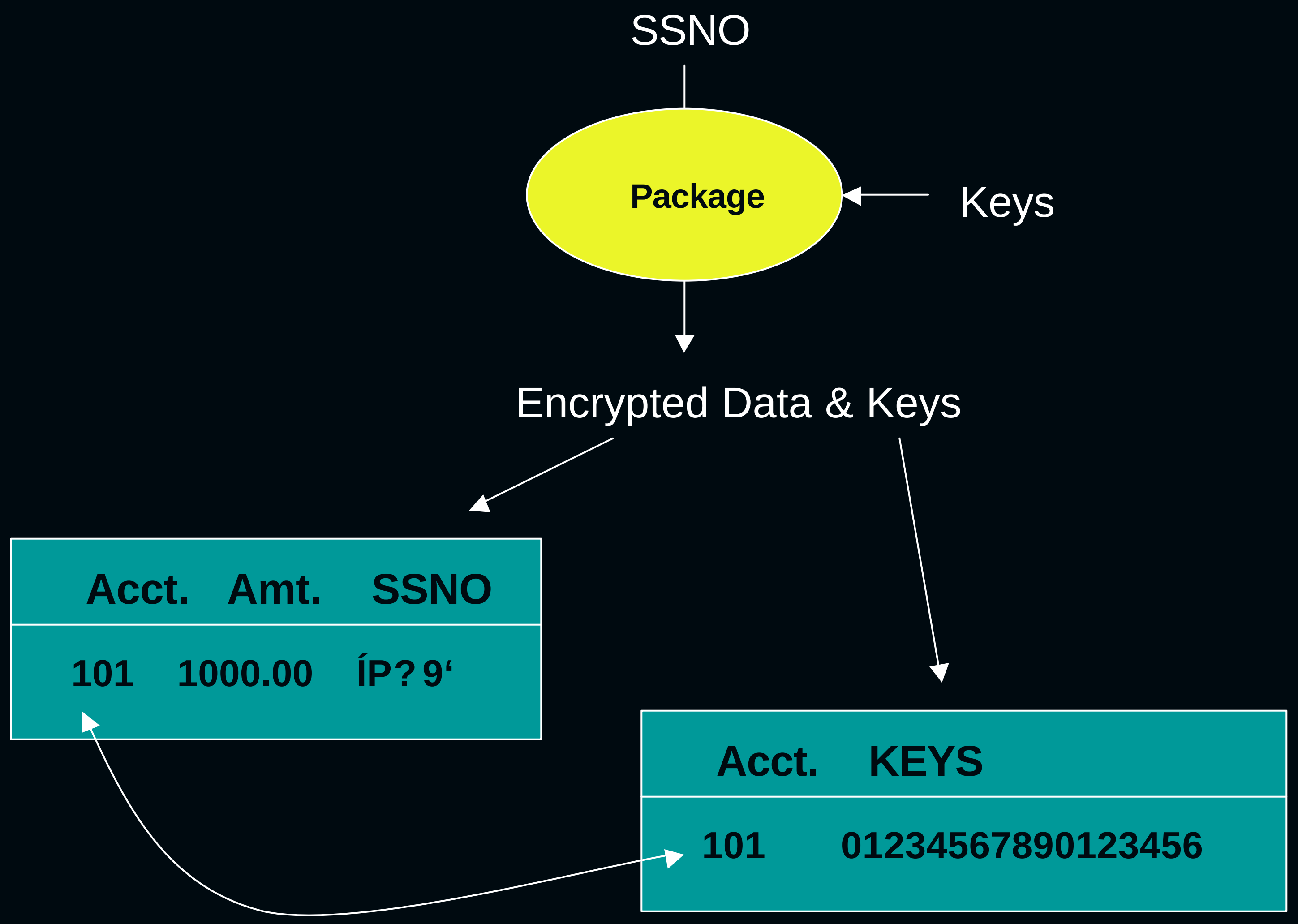
Symmetric encryption – Same key is used to encrypt/decrypt
Asymmetric encryption – One used to encrypt another to decrypt

Storing keys in same table



Acct.	Amt.	SSNO	KEYS
101	1000.00	ÍP?9‘	01234567890123456

Storing keys in another table



Acct.	Amt.	SSNO
101	1000.00	ÍP?9‘

Acct.	KEYS
101	01234567890123456

Encryption Example
Inserting data

```
CREATE or REPLACE PROCEDURE INSERT_ACCOUNT
(  account_id IN number,
   account_amt IN number,
   unencrypted_SSNO IN varchar2,
   encrypt_key in varchar2) AS
  encrypted_SSNO varchar2(2000);
BEGIN
DBMS_OBFUSCATION_TOOLKIT.DES3Encrypt(
                        input_string =>          unencrypted_SSNO,
                        key_string =>            encrypt_key,
                        encrypted_string =>      encrypted_SSNO);

INSERT into account_table values (
                        account_id, account_amt,encrypted_SSNO);
COMMIT;
END;
/
```

Storing Data – Cont’d

```
set serveroutput on

DECLARE
   password varchar2(64);

BEGIN
insert_account(101,1000,'44557878', '01234567890123456');
END;
/

-- UN: 44557878
-- EN: ÍP? 9'
```

Retrieving Encrypted Data

```
CREATE OR REPLACE PROCEDURE RETRIEVE_SSNO
(   account_id IN number,
    encrypt_key IN varchar2,
    unencrypted_SSNO OUT varchar2)  AS
     v_encrypted_SSNO varchar2(2000);

BEGIN

  select SSNO into v_encrypted_SSNO
   from account_table where account_id = account_id;

  dbms_obfuscation_toolkit.DES3Decrypt(
                        input_string =>          v_encrypted_SSNO,
                        key_string =>            encrypt_key,
                        decrypted_string =>      unencrypted_SSNO);
END;
/
```

Retrieving Data – Cont’d

```
set serveroutput on

DECLARE
   password raw(256);
   unencrypted_SSNO varchar2(64);

BEGIN

   RETRIEVE_SSNO(101, '01234567890123456',unencrypted_SSNO);

   DBMS_OUT.PUT_LINE ('UN: ' || unencrypted_SSNO);

END;
/

-- UN: 44557878
```

What about encrypting index?

- You can encrypt the index data
- Not Recommended
- You can only do equality checking (=)
- Others such as range scan will not work

10g Enhancements

- **DBMS_CRYPTO Function**

  - **Easier to use and manage**
  - **Additional encryption algorithms**
  - **Block cipher chaining modes – CBC, CFB..**
  - **Takes care of space issues**
  - **Intended to replace DBMS_OBFUSCATION Pkg**
  - **Supports RAW, CLOB and BLOB**
  - **Does not support varchar2**

DBMS_CRYPTO

```
DECLARE
v_data_raw RAW(80);
v_key_raw RAW(80);
strings varchar2(80);
encrypted_data RAW(80);
unencrypted_data_raw RAW(80);
unencrypted_data varchar2(80);

BEGIN
strings := ‘THIS IS TOP SECRET';
v_data_raw := UTL_I18N.STRING_TO_RAW (strings, 'AL32UTF8');
my_keys := '012345678901234567890123456789O1';
v_key_raw := UTL_I18N.STRING_TO_RAW (my_keys, 'AL32UTF8');

encrypted_data := DBMS_CRYPTO.ENCRYPT
(v_data_raw, DBMS_CRYPTO.DES3_CBC_PKCS5, v_key_raw);

unencrypted_data_raw := DBMS_CRYPTO.DECRYPT
(encrypted_data, DBMS_CRYPTO.DES3_CBC_PKCS5, v_key_raw);

unencrypted_data := UTL_I18N.RAW_TO_CHAR (unencrypted_data_raw, 'AL32UTF8');
dbms_output.put_line(unencrypted_data);
END;
/
```

DBMS_CRYPTO

```
SQL> @crypto_test@dbms_crypto_test

1. CHAR UNENCRYPTED DATA: THIS IS TOP SECRET

2. RAW UNENCRYPTED DATA: 5448495320495320544F5020534543524554

3. RAW ENCRYPTED DATA: 2C05A8EF1539D519F558B2B2D70C8BBC3CE365A5D5D42A15

4. CHAR ENCRYPTED DATA: ,^E????X?????<??*^U

5. RAW UNENCRYPTED DATA: 5448495320495320544F5020534543524554

6. CHAR UNENCRYPTED DATA: THIS IS TOP SECRET

PL/SQL procedure successfully completed.

SQL>
```

Third party vendors - Encryption

- Application Security
- Communication Horizons
- nCipher
- Protegrity

Monitoring & Auditing - Overview

- Assessment
- Auditing
- Monitoring
  - Intrusion Detection System (IDS)
  - Intrusion Prevention System (IPS)

Oracle Auditing

- Purpose of auditing
  - Check for suspicious activity
  - Gather statistical information
- Run cataudit.sql script
- Tables: AUD$ - owned by SYS.

Examples:

- Audit SELECT, INSERT, DELETE on <table> BY <username>
- Audit SESSION WHENEVER NOT SUCCESSFUL;

Sys/DBA Auditing

- Writes audit record for all operation by DBAs
- Audit records are written to O/S files
- AUDIT_SYS_OPERATIONS = TRUE

Fine Grained Auditing

- Set auditing policy based on
  - Columns accessed
  - Kind of rows accessed

- Associate PL/SQL procedure with audit policy
  - Send external notification whenever audit event is triggered

- DBMS_FGA.ADD_POLICY(

```
 Object_schema =>  ‘HR’,
 Object_name =>     ‘EMP’,
 Policy_name =>      ‘CheckSalary’,
 Audit_column =>     ‘SALARY’
 Audit_condition =>  ‘SALARY > 10000’
 Handler_schema => ‘COMP_CC’,
 Handler_module => ‘PageHRAdmin’
 Statement_Types => ‘SELECT’);
```

10g Auditing Enhancements

- FGA support for DML
  - It was previously only available for SELECT
  - Now includes INSERT, UPDATE and DELETE

- Uniform Audit Trail
  - New view DBA_COMMON_AUDIT_TRAIL added
  - Presents standard and FGA records in single view

Assessment – 3rd party vendors

- IP Locks – Assessment products
- Symantec – Enterprise Security Manager
- NetIQ – Vigilent security
- NGsSoftware - Squirrel
- Computer Associates – eTrust Policy/Access Control
- ISS – Database Scanner

IDS & IPS

- IP Locks
- Lumigent
- Guardium
- Symantec

DBMS Engine Security

- Security Patches
- Database Releases/upgrades
- Secure policies

Known Vulnerabilities

- Oracle Listener Denial of Service (DOS)
- Oracle LD_PRELOAD Privilege Escalation -
- Buffer Overflow in Oracle Database Server Binaries -
- Buffer Overflow in XML Database
- Buffer Overflow in EXTPROC function of the Database
- Buffer Overflow in Net Services
- Buffer Overflow in iSQL*Plus product
- Denial of Services security vulnerability
- Oracle Net Listener vulnerability
- OpenSSL Buffer Overflow

-Vulnerability in PL/SQL EXTPROC

-SQL Injection (No SQL validation in applications)

-DLLs/EXEs often have weak permissions

SQL Injection vulnerability

- Web application
  - Username or password or any inputs
- Input:
  - User = scott
  - Password = Z’ OR ‘1’=‘1
- Changes this:
  - Select * from master where
    username = :x and password = :y;
- To:
  - Select * from master where
    username = ‘**scott’** and password = ‘**Z’ OR ‘1’=‘1’**;

Application Best Practices

- Check for input – validate them
- Check the length of the string
- Check the expected value
- Check for single quotes or double quotes
- Use stored procedures and Views
- Minimize the use of dynamic SQL
- Application should not use system/sys accounts
- Create separate usernames with roles defined

Patches/Releases

- Security Patches
  - Essential
  - Test and deploy
- New Releases/updates
  - Improved version
  - Greater security

Availability - Overview

- RAC
- DataGuard
- Log Miner
- Flashback query
- Partitioning

Final thoughts

- DBMS Security is important
- Start by creating a Security Plan
- Define policies and procedures
- Create your own standards
- Use Oracle security features
- Third party vendor tools

Questions or Comments

Noel Yuhanna

Noel_yuhanna@hotmail.com