Oracle Applications

INTEGRIGY

January 9, 2007

# Security Analysis

## Oracle Applications 11i Password Decryption

## OVERVIEW

Most Oracle Applications 11i implementations are vulnerable to a significant security weakness in the encryption of passwords within the application where an insider may be able to circumvent all application controls by accessing any application account or obtain the APPS database account password. This issue is really a "perfect storm" with the convergence of (1) an inherent architectural weakness in the application, (2) generally accepted insecure operational procedures for ad-hoc query access and cloning, and (3) multiple examples of effective, easy to execute exploit code for decrypting application passwords.

The fundamental issue is that the Oracle Applications 11i application account passwords are stored in the database encrypted using the APPS database password as the encryption key rather than using a strong, one-way hash algorithm. In order to provide access to the APPS database password upon login and for other processes, it is stored encrypted in the database for each application account using the account username and password as the encryption key. In a well-controlled and secured Oracle Applications environment, this issue is difficult to exploit. However, most implementations allow some form of ad-hoc query access and have numerous cloned databases. Using published exploit code, an insider via ad-hoc query access or in a cloned database is able to decrypt easily both application account passwords and the APPS database account password. In our experience performing security assessments of Oracle Applications environments, virtually all Oracle Applications 11i implementations are vulnerable to this issue to some degree. The question is most often not if a production database is vulnerable rather can 10 or 500 people exploit it, can only trusted employees exploit it or every off-shore developer.

This inherent weakness of the Oracle Applications 11i user password encryption and storage is a topic that every so often gets some attention – it bubbles up and then is largely forgotten by most. However, it is very much present today even in 11.5.10.2 and this has been the design of Oracle Applications since at least 10.6 Smart Client. See the references at the end of this paper for a little history of the topic. This design (along with the APPLSYSPUB database account) is really a holdover from the client/server architecture and is an inherent design flaw in the current architecture of Oracle Applications.

Improving the Oracle Applications 11i user password storage is a complex change and we don't expect Oracle to fix it in 11i for the foreseeable future (remember 11.5.10 is supported until November 2012). Hopefully in Release 12, (1) a strong, one-way hash algorithm will be implemented, (2) the APPLSYSPUB database account will be eliminated, and (3) the GUEST account will be eliminated. These changes are feasible in R12 with the desupport of client-side tools like ADI, elimination of the PL/SQL Gateway (modplsql) web architecture, and support for recent versions of Oracle Forms and Oracle Discoverer.

# TECHNICAL BACKGROUND

Oracle Applications 11i stores passwords in two tables: FND_USER and FND_ORACLE_USERID. The FND_USER table stores application user account passwords and the FND_ORACLE_USERID table stores internal Oracle Applications database account passwords. Both tables use the same encryption algorithm to protect the passwords.

The information contained in this paper mostly applies to Oracle Applications implementation using "Local" authentication. Implementations using Single Sign-on (SSO) through Oracle Internet Directory (OID) for authentication do not store user account passwords in the Oracle Applications database rather they are maintained in the external directory. Nevertheless, the database account passwords are still stored in the FND_ORACLE_USERID table and the APPS password can still be obtained.

All references to the APPS database account in this paper may also be called one of the following in whitepapers, documentation, and other sources: Foundation Account, FNDNAM, AOL Account, and APPLSYS. The APPS and APPLSYS database accounts must always have the same password, thus are interchangeable in the context of this paper.

## FND_USER

The APPLSYS.FND_USER table contains all the application accounts. There are two password columns in this table: ENCRYPTED_FOUNDATION_PASSWORD and ENCRYPTED_USER_PASSWORD.

| Column | Value | Encryption Key |
|---|---|---|
| ENCRYPTED_FOUNDATION_PASSWORD | APPS password | username/password |
| ENCRYPTED_USER_PASSWORD | user password | APPS password |

These two columns provide for a two-way encryption of the passwords –

1. if you know a username and password, you can get the APPS password = ENCRYPTED_FOUNDATION_PASSWORD
2. if you know the APPS password, you can get any user's password = ENCRYPTED_USER_PASSWORD

 

The ENCRYPTED_USER_PASSWORD and/or ENCRYPTED_FOUNDATION_PASSWORDS columns may also contain one of the following values instead of an encrypted string –

| Value | Description |
|---|---|
| **EXTERNAL** | User authentication is delegated external to the application through a Single Sign-On server running Oracle Internet Directory (OID). The user's password is not stored in the FND_USER table. |
| **INVALID** | For some default Oracle Applications accounts, such as APPSMGR, account access is blocked by setting the password column to INVALID. This is done by directly updating the FND_USER table rather than through any application function or utility. |
| **X** | This value may be seen in some database, especially Vision, and is the same as INVALID. |
| **ZG_<error message>** | If the encryption algorithm fails, then the error message is stored in the password column. The user is unable to logon in the case of an encryption failure. This can be tested by setting the user password to greater than 100 characters, which will result in a ZG_ENCRYPT_FAILED_CHARSET_CLIP error. |

## FND_ORACLE_USERID

The APPLSYS.FND_ORACLE_USERID table contains all the Oracle Applications related database accounts – there is one database account for each Oracle Applications module (i.e., GL = General Ledger). The application needs access to these database schemas to perform various functions, thus it must have access to the database account password. All the passwords in the FND_ORACLE_USERID table are encrypted using the APPS password as the key.

| Column | Value | Encryption Key |
|---|---|---|
| ENCRYPTED_ORACLE_PASSWORD | Database account password | APPS password |

The password values in this table are maintained independent of the database, therefore, any changes to the database account password using the ALTER USER or PASSWORD SQL statements may not be reflected in the FND_ORACLE_USERID table. Normally, the passwords in FND_ORACLE_USERID are changed using either the FNDCPASS utility or the system administration "Oracle Users" form. It is not uncommon for standard database accounts like CTXSYS or custom database accounts to not have the correct password set in FND_ORACLE_USERID.

## ENCRYPTION ALGORITHM

The password encryption algorithm appears to be based on Triple-DES according to patch information and bug descriptions, although we did not attempt to verify what encryption algorithm is actually being used. Oracle Applications passwords can be 1 to 100 characters in length and

 

longer passwords are truncated at 100 characters. The encrypted string appears to be padded with random characters to the maximum length.

Most references to the encryption algorithm point to either the PL/SQL package APPS.FND_WEB_SEC or the Java class "oracle.apps.fnd.security.WebSessionManagerProc". For decryption and encryption, the following calls are made –

APPS.FND_WEB_SEC →
oracle.apps.fnd.security.WebSessionManagerProc →
oracle.apps.fnd.security.AolSecurity →
oracle.apps.fnd.security.AolSecurityPrivate

The actual encryption and decryption routines are in the "oracle.apps.fnd.security.AolSecurityPrivate" Java class. This Java class is stored both in the database as a Java Stored Procedure and in the operating system directory $COMMON_TOP/java.

Oracle changed the password encryption in October 2001 (Oracle Patch 2013414) in order to strengthen the password encryption. This patch was included in all new versions starting with 11.5.7 and FND.E. The new passwords can be identified in the FND_USER and FND_ORACLE_USERID tables as beginning with 'ZG'. It is important to note that passwords had to be converted to the new algorithm using a manual step, therefore, some unused applications account passwords may not be stored using the stronger encryption (64 characters and do not begin with 'ZG'). To force re-encryption of all application passwords, simply change the APPS password using your standard procedure for changing the password or using the FNDCPASS utility.

## ORACLE APPLICATIONS LOGIN PROCESS

The general Oracle Applications login process involves the following general steps –

1. The ENCRYPTED_FOUNDATION_PASSWORD and ENCRYPTED_USER_PASSWORD are retrieved from FND_USER if the account exists and is active.
2. The APPS password is obtained from ENCRYPTED_FOUNDATION_PASSWORD by using the username and password as the decryption key.
3. The user password is decrypted from ENCRYPTED_USER_PASSWORD by using the APPS password as the decryption key.
4. The decrypted user password is compared to the entered user password.

It is important to note that there are actually multiple login processes based on the history of Oracle Applications including Forms, Self-Service (HTML), ADI, Concurrent Manager, and Discoverer to name a few. Depending on the type of login and the version, the entire login process may be through Java classes on the application server or may involve calls to PL/SQL packages like APPS.FND_WEB_SEC. Ultimately, all the login processes call

 

"oracle.apps.fnd.security.AolSecurityPrivate" either as a Java stored procedure (Java class in the database) or through the application server (operating system Java class).

### Internal Application Processes and GUEST Password

Internal application processes may require the APPS database password for such tasks as creating a new user. These processes use the GUEST user password that is stored in the system profile option "Guest User Password" (GUEST_USER_PWD) and then obtains the APPS password by decrypting the ENCRYPTED_FOUNDATION_PASSWORD column for the GUEST user in FND_USER. The internal function to obtain the APPS password from the GUEST user is APPS.FND_WEB_SEC.GET_FOUNDATION_PASSWORD.

The GUEST_USER_PWD system profile option stores the GUEST password in clear-text in the format "username/password" in the APPLSYS.FND_PROFILE_OPTION_VALUES table. It can also be obtained using the function APPS.FND_PROFILE.VALUE('GUEST_USER_PWD').

### Re-Encrypting All Passwords

Since all the passwords in FND_USER and FND_ORACLE_USERID are based on the APPS password, any changes to the APPS password forces all the passwords in both tables to be re-encrypted. This is only achieved when the APPS password is changed using the "Oracle Users" form or the FNDCPASS utility.

## Decrypting Oracle Applications Passwords

In the past year, several instances of working and easy to use exploit code [2][4] have been published for decrypting Oracle Applications user passwords. The exploit code relies on the GUEST user password stored as a system profile option value to obtain the APPS password as the key to other users' passwords. Even though the exploit code requires significant database privileges in terms of access to the FND_USER table, FND_PROFILE_OPTION_VALUES table, and FND_WEB_SEC package, almost all Oracle Applications implementations are vulnerable to the decryption of production application user passwords for a number of reasons.

### SQL Access to Production Database

It is common for query access to the production database to be provided to a set of users or IT staff for troubleshooting and ad-hoc reporting. Also, query tools like Brio, Crystal Reports, and Discoverer are regularly used for supplemental reporting. Due to the complexity of the Oracle Applications data model and DBA time constraints, usually this ad-hoc query access is fairly broad in nature and generally the database account is granted the SELECT ANY TABLE system privilege rather than a narrow set of individual table grants. These accounts furthermore tend to be poorly

 

controlled and are regularly shared among users. You should assume any user with such access is capable of obtaining any application account password or the APPS password.

## CLONED DATABASES

In almost all implementations, the production database is cloned to test, development, and/or training databases on a periodic basis. Usually, the application account passwords are not changed and the access controls for the cloned database are much less restrictive. Often with development databases, the APPS password is reset to the default of APPS. When the APPS password is changed for the cloned database, Oracle Applications will re-encrypt all the application user passwords using the new APPS password.

In such cloned databases, especially where the APPS password is known, it is trivial to decrypt all the application account passwords. If the application account passwords have not been changed during the cloning process, then a developer is able to obtain all users' passwords for the production database.

Also, all 250+ database schema passwords (FND_ORACLE_USERID table) may not always be changed in the cloned database, thus a developer will be able to obtain these passwords for the production database.

## ADDITIONAL CONSIDERATIONS

### *OFF-LINE DECRYPTION*

A point overlooked in the discussion of decrypting the passwords is that the decryption is independent of the database. The encryption and decryption routines are in the "oracle.apps.fnd.security.AolSecurityPrivate" Java class that may be obtained from any 11.5.7+ Oracle Applications implementation or from Metalink by downloading any major FND patch. The attacker just has to obtain the encrypted password strings from the production or cloned database and can use the decryption routines in a standalone Java program. In general, the attacker only needs SELECT privileges on FND_USER to be able to decrypt the passwords.

### *ANY USERNAME/PASSWORD COMBINATION WORKS*

The focus of the exploit code has been on the GUEST user password as being a key to the decryption. The GUEST user password is useful in the decryption not because it is a special key or has significant meaning as some authors have incorrectly asserted but because it is a known username/password combination and can be readily obtained from the system profile options. Any valid username/password combination will work in obtaining the APPS passwords. This means a user with an application account could use their own username/password combination to decrypt the ENCRYPTED_FOUNDATION_PASSWORD for their account.

 

# PROTECTING ORACLE APPLICATIONS PASSWORDS

The Oracle Applications encrypted passwords must be protected in order to prevent decryption. The goal is to limit access to the FND_USER table and the encrypted passwords, just as should be done with the DBA_USERS view.

## GENERAL

### 1. VERIFY APPLSYSPUB DOES NOT HAVE ACCESS TO FND_USER_VIEW [CRITICAL]

Verify APPLSYSPUB and PUBLIC do not have SELECT privileges on the view APPS.FND_USER_VIEW. This is especially an issue with instances that were upgraded from 11.5.6 and prior. FND_USER_VIEW shows all application accounts and the ENCRYPTED_FOUNDATION_PASSWORD. Prior to 11.5.7, APPLSYSPUB may have been granted SELECT privileges on this view to support ADI. This view is not required by APPLSYSPUB, except for old, desupported versions of ADI.

### 2. CHANGE GUEST ACCOUNT PASSWORD

The password for the GUEST account should be changed from the default of ORACLE or GUEST. Follow the solution in Metalink Note ID 396537.1 for details on changing the password and check that the password was also changed in the System Profile Option "Guest User Password".

### 3. CHANGE PASSWORDS FOR ALL SEEDED ORACLE APPLICATIONS ACCOUNTS

Change the passwords for all Oracle Applications 11i seeded accounts (SYSADMIN, WIZARD, APPSMGR, etc.) even though these accounts may be already be disabled. At the same time, make sure all accounts except for SYSADMIN and GUEST are disabled (note a few accounts may be required by a specific module). See Metalink Note ID 189367.1 for the most up to date list of seeded user accounts. For years clients have questioned why we recommend always changing the seeded account passwords even though the accounts may be disabled – this is the reason why.

### 4. CHANGE PASSWORDS FOR ALL DATABASE ACCOUNTS [CRITICAL]

Change the passwords for every database account including all 250+ Oracle Applications schemas. Even though a module is not being used, the database account password must be changed. Use the FNDCPASS utility to change all the database passwords on a periodic basis. In 11.5.10 RUP3, FNDCPASS includes a new option (ALLORACLE) to change all the schema passwords in a single FNDCPASS call (see Metalink Note ID 398942.1).

 

### 5. CREATE ALL NEW APPLICATION ACCOUNTS WITH STRONG PASSWORDS

Create all new user accounts with unique and strong passwords. In 11.5.10, User Management (UMX) can be used to securely create new users with strong passwords.

### 6. SET SERVER SECURITY TO SECURE

Oracle Applications Server Security when set to SECURE requires servers connecting to the Oracle Applications database to provide a secure server ID. This is only used when actually logging into Oracle Applications through a SQL*Net and is not related to database authentication. By setting Server Security to SECURE may prevent an attacker from obtaining the encrypted foundation password under certain circumstances. See the section "AdminAppServer Utility" of the *Oracle Applications 11.5.10 System Administrator's Guide – Configuration* manual for detailed instructions on setting up Server Security.

### 7. IMPLEMENT MANAGED SQL*NET ACCESS

11.5.10 introduced a new feature called Managed SQL*Net Access. Managed SQL*Net Access limits the hosts that can connect to the database server using SQL*Net by implementing Oracle TNS Listener valid node checking. Valid node checking is a list of IP addresses that are permitted to connect to the database server. Unfortunately, it is very difficult to implement this feature since a large number of hosts often require access to the database server. See Metalink Note ID 291897.1 for more information on configuring this feature.

## SQL ACCESS

### 8. LIMIT ACCESS TO FND_USER AND FND_ORACLE_USERID

Limit access to the APPLSYS.FND_USER and APPLSYS.FND_ORACLE_USERID tables by all non-DBA accounts including any query-only accounts. Often an APPSREAD or similar database account is created for support purposes or end-user ad-hoc query use. These accounts tend to be created with SELECT ANY TABLE system privilege, which allows access to FND_USER. Instead, all non-DBA accounts should be created with a limited set of database privileges for only those tables absolutely required for the business function.

Unfortunately, the FND_USER table is fundamentally required by many reporting and ad-hoc queries, thus it is difficult to directly exclude this table from such database accounts. Also, over 500 standard Oracle Applications views are dependent on FND_USER. A careful review of ad-hoc query privileges should be performed to determine the exact business requirements and privileges required.

Query accounts should not normally require access to FND_ORACLE_USERID, therefore, this table should be easy to exclude from such database accounts.

 

## CLONED DATABASES

All user and database passwords should be immediately changed in all cloned databases to prevent decryption of the production passwords.

### 9. CHANGE ALL APPLICATION ACCOUNT PASSWORDS DURING CLONING [CRITICAL]

As part of the cloning process, change all application account passwords to a random string using a PL/SQL script that calls FND_USER_PKG.CHANGEPASSWORD. An operational issue then exists in that users of the cloned instance will need to obtain the new password. One solution is to use the "Reset Password Functionality" in 11.5.10 and UMX.H. Users needing access will then have to reset their password after each clone of a development or test database. See Metalink Note ID 399766.1 for more information on the UMX Reset Password Functionality.

### 10. CHANGE THE GUEST ACCOUNT PASSWORD DURING CLONING

The GUEST password requires additional steps in order to change including updating the password in the AutoConfig XML file. As part of the cloning process, follow the steps in Metalink Note ID 396537.1 to change the GUEST password.

### 11. CHANGE ALL DATABASE ACCOUNT PASSWORDS DURING CLONING [CRITICAL]

All database account password should be changed as part of the cloning process (with the exception of APPLSYSPUB). Even though a module is not being used, the database account password must be changed. In 11.5.10 RUP3, FNDCPASS includes a new option (ALLORACLE) to change all the schema passwords in a single FNDCPASS call (see Metalink Note ID 398942.1). Also, all standard database account passwords (CTXSYS, DBSNMP, etc.) should also be changed as part of each clone.

 

# REFERENCES

1. Martin Schlafer and Michael Ackerman Michael, "Oracle Unbreakable? Part I: Ten Hot Security Spots in Oracle Applications 11i", March 2002, Spring 2002 OAUG Conference, http://www.oaug.com/conferencesandeducation/papers/spring2002/SCHLAFEW.pdf
2. http://www.erp100.com/viewthread.php?action=printable&tid=1592, November 2005, (Simplified Chinese -> English Translation)
3. http://521102yz.itpub.net/post/5095/84876, May 2006, (Simplified Chinese -> English Translation)
4. Johan Louwers, "Oracle Applications Passwords Decryption Vulnerability", December 2006, http://johanlouwers.blogspot.com/2006/12/oracle-applications-passwords.html
5. Oracle Corporation, "Best Practices for Securing the E-Business Suite 3.0.4", October 2006, Metalink Note ID 189367.1, https://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_database_id=NOT&p_id=189367.1 [Sections of this document were written by Integrigy Corporation]
6. Oracle Corporation, "After RUP4 Unable To Login After Password Reset Or Password Expiration", 3 December 2006, Metalink Note ID 396537.1, https://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_database_id=NOT&p_id=396537.1
7. Oracle Corporation, "Reset Password Functionality FAQ", 15 November 2006, Oracle Metalink Note ID 399766.1, https://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_database_id=NOT&p_id=399766.1
8. Oracle Corporation, "11.5.10 New Features : Managed SQL*Net Access from Hosts", 21 January 2005, Metalink Note ID 291897.1, https://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_database_id=NOT&p_id=291897.1

# HISTORY

January 9, 2007 – Initial Version

 

INTEGRIGY

# About Integrigy

Integrigy Corporation is a leader in application security for large enterprise, mission critical applications. Our application vulnerability assessment tool, AppSentry, assists companies in securing their largest and most important applications. AppDefend is an intrusion prevention system for Oracle Applications and blocks common types of attacks against application servers. Integrigy Consulting offers security assessment services for leading ERP and CRM applications.

Integrigy Corporation
P.O. Box 81545
Chicago, Illinois 60602 USA
888/542-4802
www.integrigy.com



Authors: Stephen Kost and Jack Kanter

If you have any questions, comments or suggestions regarding this document, please send them via e-mail to alerts@integrigy.com.