

July 31, 2006

Security Analysis

Un-patched Oracle Database Security Bugs – July 2006 Oracle E-Business Suite Impact

Overview

Information about several un-patched security bugs in the Oracle Database has been published and these security bugs have varying degrees of impact on Oracle E-Business Suite 11i implementations. The most critical bug is a new attack vector regarding previously fixed SQL injection bugs in standard Oracle Database packages. The DBMS_ASSERT package used to fix security bugs can be bypassed. The second bug is the ability to bypass security controls on tables using specially crafted views. Database accounts with CREATE VIEW privilege are able to insert, update, or delete data in tables where the database account only has SELECT permission. The final security bug is an integer overflow in the ALTER SESSION statement, which an attacker can exploit to execute commands at the operating system level.

Assessment of Vulnerabilities

DBMS_ASSERT

Background

In order to fix SQL injection security bugs in standard Oracle functions and packages, Oracle developed the DBMS_ASSERT package as a common method to sanitize input like column, table, and schema names. The DBMS_ASSERT packages was introduced in the Critical Patch Update (CPU) October 2005, however, Oracle has not documented or otherwise published the functions within the package (except for the comments within the package specification itself). A number of SQL injection bugs in the October 2005, January 2006, April 2006, and July 2006 CPUs are fixed using the DBMS_ASSERT SIMPLE_SQL_NAME, QUALIFIED_SQL_NAME, and SQL_OBJECT_NAME functions.

On July 27, 2006, [Red Database Security](#) released an [advisory](#) regarding the ability to bypass the DBMS_ASSERT by using specially crafted SQL injection strings. The advisory provides examples of how to bypass the DBMS_ASSERT functions that verify passed in SQL object names.

Technical Discussion

The DBMS_ASSERT functions support both “nonquoted” and “quoted” identifiers. A nonquoted identifier is the common schema object naming with which most DBAs are familiar – all schema object names (tables, views, packages, etc.) must begin with an alphabetic character, must not be a reserved word, and can only contain alphanumeric, underscore, dollar sign, and pound sign. Quoted identifiers must be enclosed in double quotes and can include all available characters in the character-set, except for double quote. When DBMS_ASSERT SIMPLE_SQL_NAME and QUALIFIED_SQL_NAME are passed nonquoted values, the functions verify that the value is a valid schema object name according to the above schema object naming rules or that it conforms to the naming rules for a database link. For quoted identifiers, the functions verify that it is a valid quoted identifier, which is basically any string between double quotes.

The SIMPLE_SQL_NAME and QUALIFIED_SQL_NAME functions do not query the data dictionary to determine if the SQL object exists. The SCHEMA_NAME and SQL_OBJECT_NAME can be used to validate that a schema or SQL object actually exists.

It is important to note that the functions do not validate object length, so the input and returned names can be of unlimited length. For the Oracle Database, the actual limits are 30 bytes for schema objects names and 128 bytes for database link names.

The security issue is with the processing of the quoted identifiers and developer use of the DBMS_ASSERT functions. Using the SIMPLE_SQL_NAME and QUALIFIED_SQL_NAME functions, all quoted identifiers are valid with the only exception being an un-matched double quote. Our testing did not indicate any issues with the logic of the functions themselves. It appears that all invalid input combinations are properly handled and return an error (usually an “ORA-06502: PL/SQL: numeric or value error”). The only exception is that the function allows double-double quotes (e.g., TEST""TEST is a valid name), even though according to the Oracle SQL Reference manual this is not a valid name.

The problem is not with the DBMS_ASSERT functions themselves but with the use of the functions to sanitize input. In certain scenarios, it is possible to craft a SQL injection attack string using a quoted identifier that will be validated by SIMPLE_SQL_NAME and QUALIFIED_SQL_NAME, but still results in a successful SQL injection attack. This is primarily limited to when the SIMPLE_SQL_NAME and QUALIFIED_SQL_NAME functions are used in a concatenated SQL statement and the resulting identifier is enclosed with single quotes. This is the test case presented in the Red Database Security advisory.

A common use SIMPLE_SQL_NAME and QUALIFIED_SQL_NAME is to validate column names and table names in dynamic SQL statements. Bind variables can not be used as column names or table names, thus dynamic SQL must be used. When used with table names and column names in this fashion, there appears to be little risk of a SQL injection attack (unless the developer has done something stupid).

Vulnerability Information

A number of previously “fixed” SQL injection vulnerabilities in standard Oracle database packages are vulnerable by using a double quoted SQL injection that can “bypass” the DBMS_ASSERT functions. Since the introduction of the DBMS_ASSERT package in the

October 2005 CPU, Oracle has fixed 40+ SQL injection bugs. We did not perform a detailed analysis to determine the exact scope and impact of this issue across all previously fixed SQL injection bugs, however, there are at least several of the previously fixed bugs that can be exploited using double quoted identifiers. Some of these functions are granted to PUBLIC.

The easiest method to exploit these SQL injection bugs is for the attacker to create a PL/SQL function that will be executed. The simplest scenario is to have the attacker's function execute something like "GRANT DBA TO SCOTT". This works because the majority of SYS, CTXSYS, and MDSYS packages execute with definer rights, thus the attacker's function executes as SYS, CTXSYS, or MDSYS. The attacker's database account must have CREATE PROCEDURE database system privilege in order to create such a function. Prior to Oracle 10.2, the CREATE PROCEDURE system privilege is granted to the RESOURCE role.

E-Business Suite Impact

*This is a critical vulnerability for most Oracle E-Business Suite 11i implementations and can be readily exploited. Due to the application design, an attacker is able to compromise the entire application if direct SQL*Net access is permitted and it is not necessary to have CREATE PROCEDURE system privilege.*

The most likely attack scenario for the E-Business Suite is to use the APPLSYS PUB account if direct SQL*Net access is permitted. As of 11.5.10, the default is for "Managed SQL*Net Access" that limits direct SQL*Net access to the database, however, few organizations have implemented this feature. The APPLSYS PUB account does not have the CREATE PROCEDURE system privilege, but there are a number of application packages that can be easily exploited.

The following actions should be considered to mitigate risk associated with this vulnerability –

1. if possible, "Manage SQL*Net Access" should be implemented
2. database session auditing should be enabled to identify all connections using APPLSYS PUB
3. CREATE PROCEDURE system privilege should be revoked from all non-application database accounts, especially any query or read-only accounts.

View Security Bypass

Background

Oracle mistakenly published on Metalink information on an un-patched security vulnerability in the Oracle Database. On April 6, 2006, Oracle Support published a Metalink Note (Note ID 363848.1 – "A User with SELECT Object Privilege on Base Tables Can Delete Rows from a View") containing detailed information on the bug and a working example. Oracle removed the Metalink Note after about 24 hours. On April 11, 2006, Alexander Kornbrust of Red Database Security released an [advisory](#) to a security mailing list on the nature of the vulnerability, however, did not provide exploit code or a working example. This security advisory received media attention and was widely distributed.

This bug was NOT fixed in the July 2006 CPU. Oracle has not released any information as to when this bug will be fixed.

Technical Discussion

Any database account with CREATE VIEW system privilege and at least SELECT access to the base table can create a specially crafted view that will allow update, insert, and delete access to the base table. Andrew Max has [reported](#) that this bug can be exploited without even using a view. This issue appears to affect all supported Oracle Database versions from 8.1.7.4 to 10.2. We have verified this bug has not been fixed on 9.2.0.7 after applying the July 2006 CPU.

We are not aware of a working exploit being publicly available. Although based on the number of media reports and blog entries on the topic, someone with Oracle SQL knowledge can probably create a working exploit in a matter of hours.

E-Business Suite Impact

Any database account with CREATE VIEW system privilege is able insert, update, or delete data in any table that the database account has SELECT permission. Any database account with CREATE VIEW system privilege should be assumed to have full permissions on any tables where the account has SELECT permission until this security bug can be patched. Of particular concern are any read-only database accounts (i.e., APPS_RO, APPS_READ, READ_ONLY, etc.) that are widely accessed or known.

The APPLSYSPUB account does not have CREATE VIEW system privilege, therefore, the ability to exploit this bug is limited. Even if the bug can be exploited without using a view, APPLSYSPUB only has SELECT permissions on a few tables. The worst impact would be a denial of service attack by deleting all the rows in the FND_APPLICATION or similar tables.

The following actions should be considered to mitigate risk associated with this vulnerability –

1. if possible, “Manage SQL*Net Access” should be implemented
2. CREATE VIEW system privilege should be revoked from all non-application database accounts, especially any query or read-only accounts.

ALTER SESSION Integer Overflow

Background

On July 27, 2006, information was posted to a [security mailing list](#) regarding an integer overflow in the ALTER SESSION SQL statement. In particular, the SET EVENTS event ID parameter is vulnerable to an integer overflow. An Integer overflow is a buffer overflow that is exploited by passing an overly large integer value. The integer overflow may allow execution of arbitrary commands at the operating system level, however, we did not verify if the integer overflow is truly exploitable.

This vulnerability is exploitable by any database user with ALTER SESSION system privilege. Prior to Oracle 10.2, the CONNECT role has ALTER SESSION system privilege. With the pervasive and common practice of assigning the CONNECT role to new database accounts, most database accounts will have the necessary system privileges to exploit this vulnerability.

E-Business Suite Impact

This is a difficult vulnerability to exploit and working exploit code has not yet been published. The initial disclosure of the vulnerability only contains a sample SQL statement that will trigger the integer overflow, but performs no action. If working exploit code is published, it would be trivial for anyone with minimal Oracle knowledge to exploit this vulnerability.

Again, the most likely attack scenario for the E-Business Suite is to use the APPLSYSPUB account if direct SQL*Net access is permitted. The APPLSYSPUB account does have ALTER SESSION system privilege.

The following actions should be considered to mitigate risk associated with this vulnerability –

1. if possible, “Manage SQL*Net Access” should be implemented
2. database session auditing should be enabled to identify all connections using APPLSYSPUB

We do not recommend revoking the ALTER SESSION system privilege from APPLSYSPUB or any other database account without extensive testing. Theoretically, most database accounts probably do not require ALTER SESSION system privilege. CREATE SESSION allows for a database account to perform most session-level ALTER SESSION statements and the ALTER SESSION system privilege only is required to enable SQL*Trace or for SET EVENTS statements. See [Metalink Note ID 204699.1](#) for more information.

References

References

DBMS_ASSERT

Red Database Security – “Bypassing Oracle dbms_assert” –
http://www.red-database-security.com/wp/bypass_dbms_assert.pdf

View Security Bypass

Red Database Security – “Read-only user can modify data” –
http://www.red-database-security.com/advisory/oracle_modify_data_via_views.html

Andrew Max – “Yet Another Security Alert” –
<http://andrewmax.blogspot.com/2006/04/yet-another-security-alert.html>

Alter Session Integer Overflow

hasecorp@hotmail.com – “Alter Session Integer Overflow” –
<http://archives.neohapsis.com/archives/fulldisclosure/2006-07/0654.html>

Metalink Note ID 204699.1 “How to revoke ALTER SESSION Privilege” –
https://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_database_id=NOT&p_id=204699.1

History

- July 31, 2006 ■ Initial Release

About Integrigy Corporation

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

Copyright © 2006 Integrigy Corporation.

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

Integrigy, AppSentry, and AppDefend are trademarks of Integrigy Corporation. Oracle is a registered trademark of Oracle Corporation and/or its affiliates.

Integrigy's Vulnerability Disclosure Policy - Integrigy adheres to a strict disclosure policy for security vulnerabilities in order to protect our clients. We do not release detailed information regarding individual vulnerabilities and only provide information regarding vulnerabilities that is publicly available or readily discernable. We do not publish or distribute any type of exploit code. We provide verification or testing instructions for specific vulnerabilities only if the instructions do not disclose the exact vulnerability or if the information is publicly available.

The Information contained in this document includes information derived from various third parties. While the Information contained in this document has been presented with all due care, Integrigy Corporation does not warrant or represent that the Information is free from errors or omission. The Information is made available on the understanding that Integrigy Corporation and its employees and agents shall have no liability (including liability by reason of negligence) to the users for any loss, damage, cost or expense incurred or arising by reason of any person using or relying on the information and whether caused by reason of any error, negligent act, omission or misrepresentation in the Information or otherwise.

Furthermore, while the Information is considered to be true and correct at the date of publication, changes in circumstances after the time of publication may impact on the accuracy of the Information. The Information may change without notice.