

January 31, 2006

(updated February 10, 2006)

Security Analysis

Mod_plsql Security Bug Disclosure and Workaround Oracle E-Business Suite Impact = Critical

Executive Summary

Background

On January 25, 2006, David Litchfield of NGS released information about an unfixed security bug in Oracle's PLSQL Gateway (also referred to as mod_plsql). NGS classifies this as a serious vulnerability since it can be used to execute excluded database procedures and packages. In the disclosure are details of a workaround that is claimed to stop exploitation of this security bug. Integrigy has carefully reviewed the information, evaluated the potential impact to an Oracle Applications implementation, and has extensively tested the workaround.

Critical Vulnerability in Oracle Applications 11i

This is a critical security vulnerability in mod_plsql, which is used by Oracle Applications 11i. The vulnerability allows an attacker using only a web browser and having access to an Oracle Applications 11i application server to (1) execute any SQL statement or anonymous PL/SQL block as the APPS account or (2) retrieve and view any data accessible by the APPS account.

Oracle Applications 11i is Vulnerable

All versions of Oracle Applications 11i (11.5.1 – 11.5.10 CU2) are vulnerable to this security bug, even though Oracle Applications uses its own authentication mechanism to limit the database procedures and packages that mod_plsql can execute.

Also, Oracle has certified Oracle Applications 11i with Oracle Application Server 10g (10.1.2.0.2) for use of Oracle Internet Directory, Single Sign-On, Portal, Discoverer, Web Cache, and Oracle Integration in a standalone configuration. Oracle Application Server 10g (10.1.2.0.2) is vulnerable to the security bug in general and in a standalone configuration for Oracle Applications mod_plsql is configured by default for both the application server and metadata repository.

Workaround Will Cause Problems

The recommended workaround in the NGS disclosure suggests adding an Apache Rewrite rule to the `httpd.conf` file. The rule is very simplistic and does not account for the complex web architecture of most web applications. Our analysis and testing of the workaround shows it will cause problems and result in valid requests being denied in

Oracle Applications 11i, other packaged applications, and custom web applications.

Recommended Action

Due to the complexity of Oracle Applications 11i and the wide range of exploit possibilities, development of a workaround that blocks most attacks without causing problems in Oracle Applications is difficult. Under certain circumstances, `mod_plsql` can be disabled or the configuration can be changed to block such attacks. See the “Recommended Actions” section of this document for more information.

Assessment of the Vulnerability

Mod_plsql Background

The Oracle PL/SQL Gateway allows web applications to be created using Oracle database procedures and packages. It is an Apache loadable module, hence the name `Mod_plsql` (file name is `mod_plsql.so`). The called database packages use the PL/SQL Web Toolkit (`SYS.OWA_%`) to dynamically dynamic create web pages.

`Mod_plsql` was used initially by Oracle Applications 11i for the first generation of web-enabled modules. As Oracle has migrated to the OA Framework, the use of `mod_plsql` has greatly diminished and is mainly used in 11.5.10 to display on-line help – older implementations of Oracle Applications 11i still may extensively use `mod_plsql`, especially for sign-on and the personal home page.

For `mod_plsql` to access the database, a database access descriptor (DAD) is configured. The DAD contains the database connect string, username, and password, as well as other parameters that control the behavior of `mod_plsql`. For the Oracle Applications 11i the DAD information is stored in `$IAS_HOME/Apache/modplsql/cfg/wdbsvr.app` and the APPS account is used for all `mod_plsql` database connections. This configuration is standard for all Oracle Applications 11i implementations and is generated by AutoConfig.

In order to limit what database packages can be called by `mod_plsql`, Oracle Applications use `mod_plsql`'s custom authorization mechanism. The custom authorization calls the `APPS.OWA_CUSTOM.AUTHORIZE` function, which in turn looks in the table `FND_ENABLED_PLSQL` to see if the package or procedure is authorized. By default, `mod_plsql` limits access to standard database packages and procedures (e.g., `DBMS_%`, `SYS.%`, etc.), although, the Oracle Applications custom authorization mechanism is much more robust and blocks access to such packages and procedures prior to the `mod_plsql` authorization.

Vulnerability Analysis

The vulnerability is a classic SQL injection bug. `Mod_plsql` dynamically generates an anonymous PL/SQL block in order to authorize and execute the requested package or procedure. The package/procedure name and passed parameter names are inserted into the PL/SQL block without proper validation [see Red Database Security reference]. It is interesting to note but not relevant with this issue that the Oracle Applications custom authorization will not reject an invalid object call, rather it will assume a non-existent package/procedure call is “path alias” and accept it.

Since the SQL injection bug exists in an anonymous PL/SQL block, an attacker is able to execute virtually any SQL statement, function, procedure, or package -- the anonymous PL/SQL block is executed as APPS. There seems to be few limitations on what can be executed and with a little creativity even dynamic web pages displaying data can be generated. As the APPS account has full privileges on all Oracle Applications 11i data, the ability to execute any Oracle Applications package, and access to almost all standard database packages, full control of the database is possible.

Threat Analysis

Mod_plsql has extensive debugging and logging capabilities, which even records the dynamically generated PL/SQL block [see Red Database Security reference] and greatly reduces the effort to develop a workable exploit. Because an understanding of how mod_plsql functions is required, the complexity of creating a workable and effective exploit should be viewed as moderate. With the information released by NGS and Red Database Security, an Oracle knowledgeable attacker most likely can develop a working exploit in 1-3 days.

For an exploit to work, a valid package/procedure call is required. All standard database packages and procedures are blocked, therefore, an exploit must be customized to each mod_plsql application (Oracle Applications, Oracle HTMLDB, Oracle Portal, Oracle Workflow, and other third party applications). This limits the effectiveness and probability of an automated attack (i.e., worm).

The probability of an Oracle Applications specific automated attack is very low due to the small number of implementations connected to the Internet. However, due to the high probability of success of an attack and the ability to identify targets using Google and other search engines, targeted attacks against Oracle Applications implementations by external and internal attackers must be considered. The chance of an internal attack is probably greater than an external attack since employees, consultants, and contractors have knowledge and understanding of the implementation and often the necessary skills to carry out such an attack.

Analysis of Workaround

The original disclosure and a follow-up clarification by NGS suggested workarounds in order to block any potential attacks. The workarounds suggest including a Rewrite rule in the Oracle HTTP Server (Apache) configuration in order to block all query strings (passed parameters) that include a right parenthesis. The Rewrite rules are as follows --

Original Workaround

```
RewriteEngine on
RewriteCond %{QUERY_STRING} ^.*\).*|.*%29.*$
RewriteRule ^.*$ http://127.0.0.1/denied.htm?attempted-attack
RewriteRule ^.*\).*|.*%29.*$ http://127.0.0.1/denied.htm?attempted-attack
```

Revised Workaround

```
RewriteEngine on
RewriteCond %{QUERY_STRING} ^.*\).*=|.*%29).*=$
RewriteRule ^.*$ http://127.0.0.1/denied.htm?attempted-attack
RewriteRule ^.*\).*|.*%29.*$ http://127.0.0.1/denied.htm?attempted-attack
```

These rewrite rules will cause problems for Oracle Applications and other web

applications since the rules block any request where the user enters a right parenthesis into the application and the data is passed to the application in a HTTP GET request – Oracle Applications and most other mod_plsql applications generally use GET requests. There is no recommendation on placement of the Rewrite in the Apache configuration file – if it is placed in the `httpd.conf` file rather than the mod_plsql configuration file (`$IAS_HOME/Oracle/ModPlsql/cfg/plsql.conf`), then non-mod_plsql applications will be affected by the Rewrite rules.

Also, mod_plsql is capable of sending data to the database using either HTTP GET or POST requests. Rewrite rules do not process POST data, therefore, will not block attacks using POST requests.

The NGS workarounds should not be used under any circumstances. On February 8, NGS “withdrew” the recommendation of their workarounds.

Recommended Action

Internet Accessible and 11.5.9 or 11.5.10

Only for Oracle Applications 11i application servers implemented using Metalink Note 287176.1 “Oracle E-Business Suite 11i Configuration in a DMZ August 2005”

The standard Oracle Applications DMZ implementation includes the configuration of a URL firewall, which blocks access to all Oracle Applications web pages except those absolutely required by the Internet accessible modules. In the URL firewall configuration file available in Metalink Note 287176.1, all mod_plsql access is blocked by default. Only on-line help (PLS Help), iReceivables, and iRecruitment require use of mod_plsql.

Recommendation

All access to mod_plsql can be disabled by blocking access to mod_plsql using the URL firewall. In the `url_fw.conf` file, comment out all references that start with `/pls/`. There are three groups of Rewrite rules with `/pls/` -- (1) on-line help or PLS Help, (2) iReceivables, and (3) iRecruitment. On-line help will be disabled, but so will all access to mod_plsql from the Internet. After Oracle releases a patch to fix the vulnerability in mod_plsql, re-enable the on-line help.

If the iReceivables or iRecruitment modules are being used, carefully review the Rewrite rules to see if these web pages are being used in your implementation by checking the Apache log files in the production environment.

High Risk Oracle Applications 11i Implementations

Only for implementations that can not wait for Oracle to release a patch (most likely on April 18, 2006) and can extensively test the workaround

The mod_plsql configuration can be modified to force mod_plsql to validate the package/procedure call and all the parameters – the configuration setting is `always_describe` and is set on each application server. There exist two potential issues with setting this parameter – (1) a potential performance impact since mod_plsql has to describe each package/procedure call before executing and must loop through all the parameters and (2) valid calls to path aliases may be blocked as Oracle Application

custom validation will authorize such a call as `findgfm`. We have tested this in 11.5.9 and 11.5.10 and did not encounter any problems, however, our testing was limited and did not test any functional modules.

Recommendation

In the `$IAS_HOME/Apache/modplsql/cfg/wdbsvr.app` on each application server, add the following setting to each DAD in the file and restart the Apache web server –

```
always_describe = Yes
```

The `always_describe` should be set on each Internet accessible application server and a determination should be made if the threat to internal facing application servers also requires the change. There are no ramifications or issues if `always_describe` is set on some application servers and not on other servers.

For Oracle Applications environments running AutoConfig, the `wdbsvr.app` file is managed by AutoConfig and any changes will be overwritten by AutoConfig. Metalink Note 270519.1 has information on how to customize AutoConfig files.

This workaround should be thoroughly tested before using in a production environment. Review the production Apache log files from each application server for any requests containing `/pls/` (handler name for `mod_plsql`). Use `grep` (`grep -i /pls/access_log`) to list all the calls to `mod_plsql` in the current and rotated Apache log files. Of most concern are URLs that contain `findgfm` or do not conform to the standard format of `/pls/<sid>/package.procedure?parameter=xxxx`.

All Other Oracle Applications 11i Implementations

Due to the complexity of Oracle Applications 11i, the wide range of exploit possibilities, and potential issues with any workaround, we recommend most Oracle Applications implementations wait for Oracle to release a patch. Most likely Oracle will fix this issue in the Critical Patch Update scheduled for release on April 18, 2006 – Oracle has clearly stated that a patch will not be released prior to the April Critical Update.

Implementation of intrusion detection and/or intrusion prevention systems may offer some additional protection as these systems can detect or block potential SQL injection attacks. However, the standard configuration and rules for many of these products (e.g., Snort) are often not appropriately configured for Oracle Applications.

References

References

NGS – “Workaround for unpatched Oracle PLSQL Gateway flaw” –
<http://archives.neohapsis.com/archives/fulldisclosure/2006-01/0853.html>

Red Database Security – “SQL Injection via mod_plsql” –
http://www.red-database-security.com/advisory/oracle_modplsql_injection.html

Oracle – Oracle9i Application Server Oracle, “Using the PL/SQL Gateway”, Release 1 (v1.0.2.2), Part No. A90099-01
http://download.oracle.com/docs/cd/A97335_02/apps.102/a90099.pdf

Oracle – “FAQ for Oracle PL/SQL Gateway Security Issue Released by David Litchfield”, Metalink News and Notes

History

- January 31, 2006 ▪ Initial Release
- February 1, 2006 ▪ Updated recommendations to include information on the “always_describe” parameter
- February 10, 2006 ▪ Added information from and added reference to Oracle’s FAQ about the vulnerability posted on Metalink
- Updated David Litchfield’s recommendations
- Added information to Integrity’s recommendation #2 “High Risk Oracle Applications 11i Implementations” about changing the “always_describe” parameter only on Internet accessible servers

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.

For more information, visit www.integrigy.com

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.

Authors: Stephen Kost and Jack Kanter

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.