

Patch Verification of Oracle Database Servers

David Litchfield [davidl@ngssoftware.com]
9th July 2005



An NGSSoftware Insight Security Research (NISR) Publication
©2005 Next Generation Security Software Ltd
<http://www.ngssoftware.com>

Abstract

You've installed the latest Critical Patch Update from Oracle. You're secure and you comply with federal regulation and commercial standards. Or do you? This paper will show that there is a major disconnect between the expected patch level and the actual patch level and explore the impact this has on compliance with regulation such as HIPPA, GLBA, SOA and standards such as VISA CISP and MasterCard PCI. Finally the paper will conclude with on how best to address this disconnect and presents methods for doing so.

Patch Verification of Oracle Database Servers

Patching is now a board room level discussion. No longer is the worry that if you don't patch you might get hacked but rather that if you fail to patch you will fail to comply with governmental regulation or commercial standards. In the extreme such failures can end up with criminal charges being brought and a spell in jail for the organization's executives.

As part of being compliant you need to prove you have active patching policy and patches are being applied and being applied promptly. The VISA CISP and MasterCard PCI mandate for example that patches must be applied within one month of release (section 6.1.1 of the PCI). Any merchant that fails this test could end up losing the ability to process credit card transactions which would be disastrous. Part of the requirements of HIPPA, SOX and GLBA dictate that a secure network is to be maintained. If you know of a critical security patch and you don't have any plans to install it "soon" then this could be considered as negligence.

When it comes to your Oracle database servers who do you know whether you're patched? The answer is much more complex than one would imagine. Before answering though, let's place some historical context around this topic.

After a great deal of complaints due to the lack of information about the August 2004 mega-patch and the ad hoc basis on which Oracle used to release patches, Oracle instituted a quarterly release cycle for Critical Patch Updates or CPUs. The first CPU arrived in January 2005 and with it came the Risk Matrix. This matrix is supposed to enable DBAs to make better risk assessments. At the heart of the quarterly CPU is the opatch utility – the mechanism through which patches are applied to Oracle database servers. Each patch contains a number of files that contain fixed components and these are used to replace the older vulnerable components. Information about patches and what bugs have been fixed are stored in the Oracle Inventory, a flat XML file. Currently, if you want to determine whether an Oracle server is patched or not, you use the opatch utility to query the inventory.

In a survey conducted by NGSResearch it was discovered that 76% of Oracle database servers have anomalies between *expected* patch levels and *actual* patch levels. This is a staggering number. Servers that were thought to be compliant are actually not compliant. What becomes abundantly clear when you think about this is that simply using the Oracle inventory and the opatch utility is not enough to accurately determine patch status.

The reasons for this are many and simple:

Oracle supplied patches can be incomplete

There are times when the Critical Patch Update fails to install the updated fixed copies of the files. For example, the April 2005 CPU, fails in three areas. On all platforms, the new Java classes supplied to fix some SQL Injection vulnerabilities in DBMS_SUBSCRIBE and DBMS_ISUBSCRIBE are not actually loaded. On Windows platforms, the SQL script file containing a fixed version of the CTXSYS owned DRILOAD package is copied to the wrong directory so is never executed. Further, even if Ultrasearch is installed, the fixed versions of a number of WKSYS owned packages are not loaded into the database. In all of these cases it means that whilst you think you're protected against these flaws you are, in reality, still vulnerable and open to exploitation. To rectify these failures further manual steps are required - i.e. manually install the updated objects.

Oracle supplied patches can fail to fix the vulnerability

There have been many occasions where Oracle has supplied a "fix" for vulnerabilities discovered by NGSResearch yet on examining those "fixes" they fail to properly address the root of the problem. As such the flaws can still be exploited on a "patched" server.

Files can be updated or replaced directly

You don't need to use opatch to update or rollback patches; files can be replaced manually. For example, for whatever reason, let's say the updated fixed library is replaced with the older vulnerable library. Opatch will still report that the server is patched whereas it is no longer.

DBAs can forget to perform the post installation tasks

This is a fairly common mistake. After running opatch you're simply given a message "Opatch succeeded". Those that don't fully read the documentation may take this as that the patch installation has been completed by in actuality the post-installation steps need to be executed. These final steps update the PL/SQL packages, Java Class files, etc in the database and if these are not installed then the server will remain vulnerable *even though* running "opatch lsinventory" indicates that the server *is* patched.

Opatch can fail to update the inventory

Opatch often fails for various reasons. Permissions are wrong; files that are to be patched are still in use; environment variables are wrong; whatever the reason might be, and a quick search on Google reveals many more, opatch can often fail to update the inventory. If information in the inventory is wrong then so too are any observations made about patch status and levels.

Opatch can be told not to update the inventory

By specifying "no_inventory" as an argument to opatch when applying fixes it instructs opatch not to update the inventory. Whilst this does put the installation in an unsupported state it, nonetheless is occasionally used.

Opatch can be used to rollback patches

When a patch is rolled back, it has been observed that these changes are not reflected in the Oracle inventory so whilst the server appears to be patched it is not.

Solving the Problem

What is clear is that we can't rely on the information from opatch to tell us whether we're really patched or not. We need a more proactive approach for that.

After a great deal of digging and analysis, NGSResearch have come up with a number of methods that can be used to accurately determine patch status. By following this methodology you can determine whether patch installations have been successful or whether some of the components have failed to take for whatever reason. When the auditor asks you to show her your patching policy and prove that it's working you'll be able to do so with greater confidence by adopting these methods. Not only that, but you'll also be to feel greatly more assured that the vulnerabilities in your server are actually being fixed properly. If you don't have the time or resource to do this yourself [NGSSQuirreL for Oracle](#) contains a Patch Verification module that will quickly tell you what you need to know.

Patch Analysis

Firstly download the patch in question from the Metalink website (<http://metalink.oracle.com>) and then analyze it. You're looking to ascertain what SQL scripts are supplied and what each do then document this. For example, the August 2004 mega-patch (the transitional patch that led to the Critical Patch Update program.) contains many SQL scripts that update vulnerable PL/SQL packages in the SYS, CTXSYS, MDSYS and WKSYS schemas. Taking the latter, WKSYS, and exploring further we can see that these SQL scripts replace the WK_ACL, WK_ADM and WK_UTIL packages. The reason that these packages are specifically replaced is due to the fact that they contain SQL injection vulnerabilities that allow low

privileged users to gain SYSDBA privileges. Document any such replacements as we'll come back to them.

Turning to an un-patched server we now get the encrypted source for these same packages. (Note that at this stage we may find that there is no equivalent object already in the database; the reason for this is because the package is new. For example, the April 2005 Critical Patch Update creates an entirely new package called OWA_MATCH.) Once we have the encrypted source we now compare the new with the old. Due to the way that these PL/SQL packages are encrypted one small change in the code can make the two encrypted texts entirely different, especially on 10g.

If we were to run the following query on 10g (10.0.1.2) without the August 2004 patch

```
SELECT OWNER,NAME FROM SYS.DBA_SOURCE WHERE  
OWNER='WKSYS' AND  
NAME='WK_ACL' AND  
TEXT LIKE '%a676 4557%'
```

we'd get the following result set:

OWNER	NAME
WKSYS	WK_ACL

After applying the patch and re-running the same query, nothing *should* be returned. The reason for this is simple. In the query above, the TEXT at the end, "a676 4557", refers to the checksum and size of the old package. This information is stored in the TEXT of the encrypted source of the package. The new package has a different checksum and size - namely "abe6 4aa5" - we extracted this information from the patched SQL file. As such, when we run the query again on the patched server, no record set is returned because the package has been replaced. In other words, for this component, the patch installation was successful. If, however, when you run this query on a patched server and a recordset *is* returned, then the patch has failed for this component; you still have the older, vulnerable version. The same procedure is then applied to every component that patch replaces or, in certain cases, drops. An example of this is the CTXSYS DRILOAD vulnerability. This package, prior to August 2004, contains a vulnerability whereby an attacker can gain DBA privileges by passing arbitrary SQL to the VALIDATE_STMT procedure. The August patch fixes this vulnerability by simply dropping the procedure. Before August, if you were to run the query

```
SELECT OWNER,PACKAGE_NAME FROM ALL_ARGUMENTS WHERE  
OWNER= ' CTXSYS ' AND  
PACKAGE_NAME= ' DRILOAD ' AND  
OBJECT_NAME= ' VALIDATE_STMT ' ;
```

you'd get the following recordset

OWNER	PACKAGE_NAME
CTXSYS	DRILOAD

After August, provided the patch has been installed correctly you'd get no recordset. This analysis showed its strength after the April 2005 Critical Patch Update. It was noted that this query returned a recordset when it should not have. On probing further it was discovered that the April 2005 CPU was faulty and though opatch and the inventory stated the server was not vulnerable, the server actually was. In response to this Oracle updated the Metalink notes for the April CPU.

Basically what we're doing here with this analysis is comparing the *expected* end state of the database server with the *actual* end state. If the two do not marry up then the patch installation was *not* successful.

One of the great advantages of performing this patch verification is being able to isolate *exactly* what component didn't install properly. Armed with this knowledge the problem can be rectified.

In summary here's how to perform patch verification:

Analyze the patch to see what files and objects are being replaced. Examine the same objects in the database. Note the differences then apply the patch. Once applied, query each component to see whether the server has the older un-patched version or the newer patched version. Once done, fix any problems that may come to light.