Interested in learning
more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

Automated Scanning of Oracle 10g Databases

Copyright SANS Institute
Author Retains Full Rights

AD

Automate more Web application scanning.

watchfire

© SANS Institute 2007, Author retains full rights.

Automated Scanning of Oracle 10g Databases

GSOC Gold Certification

Author: Rory McCune, rorym@mccune.org.uk

Adviser: Paul M. Wright

Accepted: January 23rd 2007

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Abstract

One of the points raised on the course is that whilst there are a number of scripts and tools which can be used as part of an Oracle Security review there isn't really a "security scanner" like tool which is available for free.

This paper analyses the various areas of Oracle security covered by the course and seeks to propose details of which checks could be carried out automatically and how (for example what parameters to check, and what the various resultant values would indicate about the security of the database).

Additionally as part of this, I expect to produce an initial beta copy of a scanning tool which implements the checks described in the paper.

This beta would likely be implemented in the Ruby Object-Oriented Scripting Language. In terms of Oracle database versions, the paper would look to primarily focus on 10g as it is the latest version available, however it is expected that many, if not all, of the techniques would apply equally to earlier versions.

This paper should be of interest to people tasked with auditing the security of Oracle databases and also with security tool authors who focus on Oracle.

© SANS Institute 2007, Author retains full rights.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Table of Contents

1 Introduction 5
2 Oracle Assessment. 6
Introducing RoraScanner 7
3 Oracle file-system checks 9
Unix file permissions & ownership 9
Windows file permissions & ownership 10
Automated checking of Oracle file permissions - UNIX 11
Automated checking of Oracle file permissions - Windows 12
4 Oracle Initialization file checks 13
Automated checking of Oracle initialization files 14
5 SQL Connection Checks 15
Default and easily guessed password checking 16
Oracle patch level checking 17
Password profile checking 20
User rights checking 22
Database parameter checking 23
6 Reporting and persistence 24
7 Conclusion & Alternatives 25
Appendix A - RoraScanner architecture overview 27
Anatomy of a Check 28

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Installation Instructions 30

© SANS Institute 2007, Author retains full rights.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

1 Introduction

Oracle's database products are arguably the most popular in the world and are used in most, if not all, large corporate organizations in one role or another. Part of the reason for this popularity is the product's flexibility and ability to fit into a number of roles from data warehousing applications to on-line transaction processing. However an inevitable consequence of increased flexibility and complexity is an increased "attack surface", meaning that there are more ways in which the system can be attacked due to the larger number of installed features.

Also as a result of the complexity of Oracle's database product, it can be a daunting task to know where to start in auditing or securing the database. For example, in the default Oracle 10gR2 installation, there are over 50000 database objects and almost 24000 table permissions already defined.

One way to help address this problem is to use automated tools which allow the auditor to focus more on the analysis of tool output and policy related issues that don't lend themselves well to automation.

This document is intended to look at the areas within the Oracle database which can be reviewed by automated tools to ascertain the current level of security of the database.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

2 Oracle Assessment.

In order to assess the security of an Oracle installation, there are a number of areas which can be analysed. This document will focus on the areas which are specific to the Oracle installation; however they are by no means sufficient to ensure the secure operating of an Oracle installation on its own. In particular, the security of the underlying operating system is vital, as if it is possible to get privileged access to the operating system through an unpatched vulnerability or poorly configured system daemon, then it will be subsequently possible to compromise the security of the database installed on that server.

For the purposes of this document the potential Oracle security checks have been split into three areas, depending on the method used to perform the check

1. Checks carried out at the file-system level (eg, permissions on Oracle program files)
2. Checks carried out on the Oracle Initialization files.
3. Checks carried out over an SQL connection to the database.

In terms of sources for the checks which our automated auditing can carry out, there are two good checklists which can be used for this. First is the SANS SCORE checklist available from http://www.sans.org/score/checklists/SCORE_Oracle_v3.1.xls and the Center for Internet Securitys Oracle Benchmark is also available from http://www.cisecurity.org/bench_oracle.html .

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Introducing RoraScanner

The tool being used to implement the checks described in this paper is called RoraScanner. It is written in the Ruby scripting language (http://www.ruby-lang.org/en/) which is an object-oriented scripting language originally developed by Yukihiro "matz" Matsumoto.

Ruby has several characteristics which lend it to a project like this. The object-orientation lends itself to an extensible model, allowing new classes of checks to be added to the scanner without disturbing existing code. Ruby code also tends to be relatively readable unlike other some other scripting languages.

Also, using Ruby allows for the potential to add a web based front-end to the project relatively easily using Ruby on Rails (http://www.rubyonrails.org). This would also allow easy integration with the ActiveRecord framework which could be used to create a database of previous scans allowing for auditing of security relevant changes over time.

There are many excellent learning resources available online for those interested in learning more about ruby. Good starting points are

- The Ruby Homepage - http://www.ruby-lang.org/en/documentation/
- Learn to program Ruby - http://pine.fm/LearnToProgram/
- Why's poignant guide to Ruby - http://poignantguide.net/ruby/

RoraScanner currently provides a basic reporting & logging framework and allows for checks to be carried out over an SQL*PLUS connection to an Oracle database. Throughout the document, sections are included on how RoraScanner is implemented to cater for that class of check. In some cases (file-system scanning

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

and initialization file parsing) a separate script, rora-file-scanner, is used as the mechanism required to complete the check differs from that required by the main script.

Rora-file-scanner either parses the output of file listings to audit file permissions or Oracle initialization files to confirm values within these files are set appropriately.

Information about the general architecture of RoraScanner can be found in section 7 of this document. The latest version of RoraScanner can be found at http://rorascanner.rubyforge.org/ . The best way to get the latest version of RoraScanner is to use the subversion service available from RubyForge or download the latest release.

Installation information for RoraScanner and its pre-requisites can be found in the README file included with the release.

© SANS Institute 2007, Author retains full rights.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

3 Oracle file-system checks

Ensuring that the correct permissions are set on installed Oracle programs and data files, helps to ensure that only authorised operating system users can modify or read those programs and can also help to minimize the impact of any mistakes by users of the system ('*rm -rf **' has way less impact if the user doesn't have many rights!).

In terms of classification, there are two major environments where Oracle is installed that need to be considered, Unix based operating systems and Windows based operating systems. The permission structures and commands used to view and modify these permissions are quite different on each of these types of system and should be considered separately.

Unix file permissions & ownership

On Unix, the files which constitute the Oracle installation are found under the $ORACLE_HOME directory (this environment variable is set when Oracle is installed) which by default will be /opt/oracle/10GR2 for the current version.

The files under this directory should be accessible only to the Oracle database user, with read access allowed for any other local users who need to run the utilities installed in this area.

There are two main requirements for file permissions specified in the CIS and SCORE checklists. The general rule is that files under $ORACLE_HOME should have rights of 750 or less (Additional information on Unix file permissions can be found

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

at http://en.wikipedia.org/wiki/File_system_permissions). There is one exception to this general requirement, which is that files in $ORACLE_HOME/bin should be set to 755 or less, to allow users who are not in the oracle users group to run the utilities in that directory.

In terms of ownership, the Oracle program and data files should be owned by the Oracle user (by default this is "oracle") and the primary group of the Oracle user (by default this is "dba").

Windows file permissions & ownership

The first key check on a windows installation of Oracle is to confirm that the file-system, that Oracle is installed onto, is NTFS and not FAT32, as FAT32 does not allow for the setting of file permissions.

Windows file permissions should be set such that the Oracle User has full control of the files in the Oracle base directory and the "Everyone" windows group should have all permissions revoked from this directory tree.

© SANS Institute 2007, Author retains full rights.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Automated checking of Oracle file permissions - UNIX

On Unix, checking of file permissions automatically tends to be relatively straightforward due to the support of various scripting languages and the text readable output from the "ls" command.

There is an existing script which can be used to carry out file permission audits where the auditor has shell access to the database server. Fileprobe.sh can be downloaded from: http://www.evdbt.com/fileprobe.sh

An alternative approach, currently used by RoraScanner is to use the output of the command "ls -laR > <*text file*>" when run from the Oracle Installation directory (by default /opt/oracle/10gR2) as a root-level user will provide all the information needed to complete these checks. This approach also has the advantage that sysadmins do not need to run any custom scripts on their servers to get the output.

There are 2 general rules that should be stated for this series of checks:

1. If the file checked has group permissions of greater than Read/Execute or there are "other" permissions on the file, it should be flagged as a problem.
2. If the file is owned by any user other than the oracle user or the oracle group, then it should also be flagged as a problem file.

The main exception to these rules is that the files in the $ORACLE_HOME/bin directory, which should have rights of 755 or less.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Automated checking of Oracle file permissions – Windows

On windows, it is a bit trickier to get access to permission information in a programmatic fashion, so initially RoraScanner uses a relatively simple check to confirm whether the "Everyone" group has any access to the files in the ORACLE_HOME directory. To provide a relatively straightforward way of doing this, a program called Dumpsec can be used (http://www.systemtools.com/somarsoft/index.html). This is a commonly used file-system auditing tool and allows us to create CSV output of all the rights assignments for a directory tree. The process of using Dumpsec to get the information we need in the correct format is:

1. Change the Report -> Permission Report Options setting to get Dumpsec to "show all directories and files".
2. Then Run Report -> Dump permissions for File System. Choose the Oracle Base directory as the starting point for the report.
3. Once the report has completed running, choose the File -> Save Report As.. option and select Comma Separated columns as the file type to save.

This provides us with a report which lists every file in the directory tree and each rights assignment for that file on a separate line which is a relatively easy format to process automatically.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

4 Oracle Initialization file checks

Oracle 10G uses a number of initialization files to control the settings of variables related to the configuration of the database and related systems like the TNS listener. The main log files that we're interested in for the purposes of these audits, are listener.ora and sqlnet.ora. init.ora also contains several parameters but the easiest way to view these is by selecting information from v$parameter over a database connection.

Within these three files there are a number of checks from the SCORE and CIS checklists that can be reviewed relatively easily by an automated tool.

© SANS Institute 2007, Author retains full rights.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Automated checking of Oracle initialization files

There are a couple of techniques required to review the contents of these initialization files depending on how the relevant sections are laid out.

In some instances there are name/value pairs on a line by themselves. These lines can be easily parsed and the value extracted and compared against the desired value.

Additionally there are bracketed sections which start with a "*parameter =*" for these we need to parse the multi-line statements however, the checks described in the CIS and SCORE checklists don't require us to parse these.

One point that has to be considered in these checks, is the default value of the parameter if it is not found in the file, as it is important not to flag a parameter that's not found, if its default configuration is secure. Fortunately the CIS checklist provides some guidance on this.

Currently, the rora-file-scanner script reads in the .ora file supplied as a parameter to the script, reads the lines in the file into an Array variable then passes this array to each of the tests to be completed. In order to maintain readability in the code, the checks for each initialization have been split into separate module files.

© SANS Institute 2007, Author retains full rights.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

5 SQL Connection Checks

This section is by far the largest of the three we outlined in the introduction, in terms of the number and types of checks that can be carried out and there are a number of sub-sections that will be considered:

1. **Default and easily guessed passwords.** One of Oracle's main problems traditionally has been the large number of default accounts which ship on the database with default passwords.
2. **Oracle patch level.** Oracle have released a large number of security patches for 10g which resolve a range of security related issues. These are currently released as quarterly Critical Patch Updates (CPUs).
3. **Password profiles.** Closely related to reviewing database passwords, is checking the password profiles assigned to database users to ensure that they meet good practice requirements.
4. **User rights.** It is possible to review the rights assigned to security critical database tables and also system rights.
5. **Database parameters.** There are a number of security related parameters that can be checked over a database connection including the ones from init.ora that were mentioned in the previous section.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Default and easily guessed password checking

> **Existing Tools** – There are a range of existing Oracle password checkers available. Some of these just check for default passwords whilst other look for easily guessed ones as well.
>
> Orabf - http://www.toolcrypt.org/index.html?orabf
>
> bfora - http://www.digitalsec.net/stuff/tools+misc/bfora.pl
>
> http://www.trantechnologies.com/pass_cracker.zip
>
> http://802.11ninja.net/code/hashattack-0.2.0.tgz

One of the easier checks to implement is default password checking. It is relatively straightforward to extract the usernames and password hashes from a database and compare them with those contained in one of the lists available on the Internet. In RoraScanner, the list used comes from Pete Finnigan's site and can be found at http://www.petefinnigan.com/default/oracle_default_passwords.csv).

Assessing whether a user is using an easily guessed password is somewhat more complex. To do this we need a dictionary file and also an implementation of the Oracle password hashing algorithm. The details of the algorithm are well known, having been published in a paper by Joshua Wright and Carlos Cid (available here http://www.sans.org/reading_room/special/index.php?id=oracle_pass&ref=911&portal=af19b59fe003baf8ff4d03f329fe480d). An automated scanner can either choose to implement the algorithm as described or use one of the existing password checking programs which are available. RoraScanner currently uses checkpwd from Red database Security (http://www.red-database-security.com/software/checkpwd.html) as it works from the command line and allows for results to be parsed easily.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Oracle patch level checking

Determining the patch level of an Oracle installation has two major components. Firstly, confirming the version of the software running, which is easily done by querying the v$version view and returning the results and secondly ascertaining what, if any, security patches have been applied to the system.

One technique for doing this is based on the work described in Paul M. Wrights paper "Using Oracle Forensics to determine vulnerability to Zero Day exploits". That paper describes a process for creating md5 hashes of the DDL of PACKAGE objects in the database using the DBMS_METADATA.GET_DDL function. By calculating the hashes for all appropriate objects in the database for each CPU level, it should be possible to come up with signatures which uniquely describe each level. A variation on this technique which is mentioned in "Oracle Forensics Part 4: Live Response" by David Litchfield, is to get the source from SYS.SOURCE$ but in principle the results should be the same.

In order to enable this technique, the first step is to create a set of hashes for each patch level available. To do this, one way is to iterate over all the objects which have DDL selectable, using the GET_DDL process. From a review of an Oracle 10gR2 server, the following object type have DDL which can be selected :-

CLUSTER,DIMENSION,FUNCTION,INDEX,INDEXTYPE,LIBRARY,OPERATOR,PACKAGEPROCEDURE,SEQUENCE,TRIGGER,VIEW

Using the sample database, it is then possible to create a list for each CPU-level and run a comparison to establish which MD5 hashes change from one to the next, thus establishing the changed packages.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

One object type which does change frequently but which is not suitable for use, is the SEQUENCE object as the alteration in the DDL there is just the number to start.

Following from this we get this sequence of changes for the various CPU levels from 10.2.0.1 with no CPUs installed through to the April 2006 CPU level.

Changes from original level to January 2006 CPU

Object Type	Object Name	SCHEMA OWNER	ORIGINAL HASH	NEW HASH
PACKAGE	OWA_OPT_LOCK	SYS	378dc65d663566e15a3acd132048714a	cb965c814e58518c01797cd5fa06f73b
PACKAGE	OWA_UTIL	SYS	fef91af3adac5ac00a2d800c62314bac	b8feebd79e05a14ce4cf813d31aee481
PACKAGE	HTP	SYS	aceee71f7a2dc864682cdfe548b93e03	0737c8fe10c981896d232c457c564908

Changes from January 2006 CPU to April 2006 CPU

Object Type	Object Name	SCHEMA OWNER	ORIGINAL HASH	NEW HASH
PACKAGE	DBMS_EXPORT_EXTENSION	SYS	0b2761a5835ed97f683ac8bf3ad54dc3	2f115cb22c80785c3c08e34642c8a63f
PACKAGE	DBMS_REGISTRY_SYS	SYS	82f759c1038eeba30bb56d8a887476e4	dda8a5bebf3615ecb0f200aedc061f78

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Reviewing the changes for the later CPUs in July 2006, October 2006 and January 2007, reveals that the changes are primarily in PACKAGE objects with some changes in trigger objects. The exception to this is the application of the January 2007 CPU which doesn't seem to make any changes that are picked up by reviews of DDL.

From this it is obvious that this check on its own isn't sufficient to determine patch level however, there are some other areas of the database that can be reviewed for corroborating data.

One of these is the DBA_REGISTRY_HISTORY view which stores the current applied CPU level in the comment field. Another technique which could be used to augment this checking, is based on the The LAST_DDL_TIME field in the USER_OBJECTS view which stores a timestamp for modifications of the DDL for each object in the database . Combined with a process similar to the one described above for creating lists of packages DDL hashes, it is possible to determine the current patch level.

© SANS Institute 2007, Author retains full rights.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Password profile checking

With any Oracle database there can be a number of password profiles setup for different classes of user. These profiles specify security related parameters like the maximum lifetime of a password and the number of failed login attempts that can be made before the account is locked out.

Automated checking of these profiles is primarily a matter of checking current values for each relevant setting and then flagging any exceptions compared to the values specified in the CIS and SCORE checklists.

The only complication to this is that profiles other than the default one may have a setting of "DEFAULT" rather than a value, which means that they have the same setting as the DEFAULT profile. In those cases it is important to flag those parameters only where the DEFAULT profile is also being flagged.

A listing of potential parameters to check based on the material in the SANS Securing Oracle course is:

- password_life_time
- password_reuse_time
- password_reuse_max
- failed_login_attempts
- password_lock_time
- password_grace_time
- password_verify_function

Of these parameters, the only one which falls outside of our base approach to

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

checking is the "password_verify_function" as it is not set to a numeric value that we can check. Instead, the easiest check to perform would be whether this parameter is not null, to indicate that there is a password verification function present. This isn't ideal of course, as it doesn't attest to the strength or not of the function.

© SANS Institute 2007, Author retains full rights.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

User rights checking

This is a potentially tricky area to evaluate automatically as the rights provided to users and roles will vary from database to database and what will be appropriate for one system may not be for another.

There are however, some high level privileges that we can review that would give some indications that excessive rights have been granted.

Firstly there is the area of default role assignments. There are several default roles which ship with Oracle which provide an excessive amount of permissions. The CONNECT role may be assumed to be required for authentication to the database but in reality provides additional privileges that aren't required for most application users. Similarly, the RESOURCE and DBA roles shouldn't be used as roles with only the privileges required but should be defined on a database by database basis.

Secondly, it is possible to report back assignation of system privileges like "DROP ANY TABLE" or "ALTER ANY PROCEDURE" primarily as information for the auditor to review whether these are being used appropriately. Also reviewing access to sensitive tables like DBA_USERS where the password hashes are stored is a good idea.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Database parameter checking

There are a wide range of database parameters for the configuration of the database which can be checked across a sql*net connection.

Within the v$parameter view there are a series of checks that can be made by comparing the values returned with the suggested values in the CIS and SCORE checklists. In terms of techniques for checking, the primary thing to be aware of is that (similarly to the process for checking values in initialization files) it is important to be aware of whether the default value is considered acceptable, for the purposes of reporting the result if that parameter is not set.

© SANS Institute 2007, Author retains full rights

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

6 Reporting and persistence

At a basic level, the techniques described in this document can be used for one-off reporting which would likely be suitable for the purposes of security auditors (either external to an organisation or internal). One of the key design elements that has been implemented in RoraScanner as it stands, is the consistent format that the results from each scan module are returned in. This allows for the reporting module to simply iterate over the listing of complete checks and apply whatever transformation is required to get the desired output format.

At the moment, this takes the form of a basic HTML report which outputs results from the various plugins in HTML table format.

However, for other use scenarios such as a security team monitoring, the security of a database server over time, it would be desirable to be able to compare results of previous scans to the current one.

In this case, adding in functionality to save scan results to a database file will provide the best solution. In the case of RoraScanner, the easiest way to do this is to make use of an existing Object-Relational Mapping (ORM) framework such as ActiveRecord. This allows for easy translation of the output arrays from the scanner into a variety of different relational databases.

© SANS Institute 2007, Author retains full rights.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

7 Conclusion & Alternatives

From this paper it is possible to see that by automating the process of auditing the security of an Oracle database, significant results can be produced which allow the auditor to focus on other areas of the security of the installation which don't lend themselves to automation, such as reviewing the administrative processes for handling removal of accounts from databases for users who have left.

RoraScanner currently (June 2007) implements a small number of checks in each of the categories mentioned however, the current intention is to, where possible, implement all of the checks listed in the CIS Benchmark for Oracle 10g.

In terms of alternative products or tools which could be used to achieve similar results, there are some free and proprietary options available. There are three main commercial options for database vulnerability assessment, each of which cover a variety of database engines in addition to Oracle. NGSSQuirrel from NGSSoftware (http://www.ngssoftware.com), AppDetectivePro from AppSecInc (http://www.appsecinc.com/products/appdetective/) and IpLocks (http://www.iplocks.com/html/p/capabilities_va/). There are also some other free options available as listed below

- Scuba (http://www.imperva.com/application_defense_center/scuba/) - This is a free database security scanner recently released by Imperva. It covers several database engines (DB2, Oracle, MS SQL and Sybase) and creates HTML reports. Scuba focuses on checks carried out over an SQL connection.
- CIS Oracle Database Benchmark (http://www.cisecurity.org/bench_oracle.html) - At the moment this tool

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

only covers version 1.2 of the CIS Oracle benchmark and is restricted to the scanning of Oracle 8i databases.

- Oscanner (http://www.cqure.net/tools.jsp?id=20) - GPL java based Oracle Security scanner. At the moment this project doesn't seem to be very active. Oscanner is focused on enumeration of security relevant information over an Oracle database connection.

© SANS Institute 2007, Author retains full rights.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Appendix A - RoraScanner architecture overview

RoraScanner is a series of ruby scripts designed to audit an Oracle database installation for security related information and provide a report of relevant findings back to the user. There are a couple of key design goals which have influenced the setup of the system

1. Extensibility - With any security scanner, one of the key attributes needs to be the ability to add new security checks to the program easily.
2. Minimal database rights - RoraScanner tries to minimize the quantity of database rights required to complete its checks, in order to ease the process of being granted approval to run scans. Specifically, this has meant that using a lot of PL/SQL techniques that involve creating database tables have been ruled out, as have techniques requiring direct access to the underlying Operating System.

There are essentially two groups of files within RoraScanner currently. Those that deal with file level scanning and those that deal with scanning done over the sql*net connection. With each group there is an initial script that deals with reading in command line objects and setting up the scanner object and then calling the appropriate methods to carry out the checks an generate the reports.

Following on from that, there are Class and Module definition files. Where appropriate, these have been split up into separate physical files. As the check base grows it may be a good idea to split them up further.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Anatomy of a Check

Each check that RoraScanner carries out, has a number of variables setup in a specific format to allow for the reporting module to parse the results into the correct format.

```
def version_scan
      def find_version(version_string)
      version = version_string[/\d+\.\d+\.\d+\.\d+\.\d+/]
      return version
    end
    @version_scan_source = "SCORE"
    @version_scan_description = "This check returns the version of Oracle in use on the database"
    @version_scan_sql = "select * from v$version"
    @version_scan_implications = " "
    @version_scan_banner = "Oracle Version"
    @version_scan_vuln_class = "Informational"
    @version_scan_version = "all"
    @version_scan_column_names = %w[Component_Name Version_Number]
    @version_scan_results = Array.new
    @conn.exec('select * from v$version') do |r|
      if r[0]=~/Oracle/
           @version_scan_results << ['ORACLE', find_version(r[0]) ]
      end
      if r[0]=~/PL\/SQL/
           @version_scan_results << ['PLSQL', find_version(r[0]) ]
      end
         if r[0]=~/CORE/
        @version_scan_results << ['CORE', find_version(r[0]) ]
      end
      if r[0]=~/TNS/
           @version_scan_results << ['TNS', find_version(r[0]) ]
      end
      if r[0]=~/NLSRTL/
        @version_scan_results << ['NLSRTL', find_version(r[0]) ]
      end
    end
    @completed_checks << "version_scan"
  end
```

In this example, the check is designed to pull the version information out of the database for information (and also so it can be used in later checks if needed). Once these variables have been specified, we can execute the sql required for the check using the existing Oracle connection which is setup when we created the RoraScanner object. The results from this are passed into a ruby block and the relevant information extracted. In this case the code looks through the results for

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

certain keywords and if found, adds them to the results array.

At the end of the check, the @completed_checks array is updated to include the check name. It is important to note that the text added to the @completed_checks array must be the same as the naming used for the check variables, as this is used in the reporter module.

© SANS Institute 2007, Author retains full rights.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Installation Instructions

Note that all the components of RoraScanner only need to be installed on a scanning workstation, it is not necessary to install anything on the target server.

Windows Install instructions

- **Install the RoraScanner scripts** - http://rorascanner.rubyforge.org
- **Install Ruby on scanning machine** - http://rubyforge.org/projects/rubyinstaller/
- **Install the Oracle Instant client files** - http://www.oracle.com/technology/software/tech/oci/instantclient/index.html
- **Install the Ruby Oracle client** - http://ruby-oci8.rubyforge.org/en/
- **Install ruport from a command line** - "gem install ruport"
- **Run RoraScanner** (usage instructions are in the README)

Linux Install Instructions

- **Install the RoraScanner scripts** - http://rorascanner.rubyforge.org
- **Install Ruby and rubygems** on the scanning machine using the package manager for your distribution eg, Fedora Core - yum install ruby, yum install rubygems

- **Install the Oracle Client Files** - http://www.oracle.com/technology/software/tech/oci/instantclient/index.html
- **Install the Ruby Oracle client** - http://ruby-oci8.rubyforge.org/en/
- **Install ruport from a command line** - "gem install ruport"
- **Run RoraScanner** (usage instructions are in the README)

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Last Updated: May 23rd, 2013

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Malaysia @ MCMC 2013	Cyberjaya, MY	Jun 03, 2013 - Jun 08, 2013	Live Event
SANS Pen Test Berlin 2013	Berlin, DE	Jun 03, 2013 - Jun 08, 2013	Live Event
Industial Control Systems Security Training - Houston	Houston, TXUS	Jun 10, 2013 - Jun 15, 2013	Live Event
Security Impact of IPv6 Summit 2013	Washington, DCUS	Jun 14, 2013 - Jun 16, 2013	Live Event
SANSFIRE 2013	Washington, DCUS	Jun 14, 2013 - Jun 22, 2013	Live Event
SANS Canberra 2013	Canberra, AU	Jul 01, 2013 - Jul 13, 2013	Live Event
Digital Forensics & Incident Response Summit 2013	Austin, TXUS	Jul 09, 2013 - Jul 16, 2013	Live Event
SANS London Summer 2013	London, GB	Jul 09, 2013 - Jul 16, 2013	Live Event
SANS Rocky Mountain 2013	Denver, COUS	Jul 14, 2013 - Jul 20, 2013	Live Event
SANS Mumbai 2013	Mumbai, IN	Jul 22, 2013 - Jul 27, 2013	Live Event
SEC528 SANS Training Program for the CompTIA® New Advanced Security Practitioner (CASP) Certification	Washington, DCUS	Jul 22, 2013 - Jul 26, 2013	Live Event
SANS San Francisco 2013	San Francisco, CAUS	Jul 29, 2013 - Aug 03, 2013	Live Event
SANS SEC 560: Network Penetration Testing @ Bangalore 2013	Bangalore, IN	Aug 05, 2013 - Aug 10, 2013	Live Event
SANS Boston 2013	Boston, MAUS	Aug 05, 2013 - Aug 10, 2013	Live Event
Critical Security Controls Summit	Washington, DCUS	Aug 12, 2013 - Aug 18, 2013	Live Event
Industrial Control Systems Security Training - DC	Washington, DCUS	Aug 12, 2013 - Aug 16, 2013	Live Event
SANS Thailand 2013	Bangkok, TH	Aug 19, 2013 - Aug 31, 2013	Live Event
SANS Virginia Beach 2013	Virginia Beach, VAUS	Aug 19, 2013 - Aug 30, 2013	Live Event
Mobile Device Security Summit 2013	OnlineCAUS	May 30, 2013 - Jun 06, 2013	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced