CONFIGURING A SECURE *ORACLE9I APPLICATION* SERVER ENVIRONMENT

Oracle Corporation

Securing Oracle9iAS 1.0.2.x

Author:	Stephen Comstock
Creation Date:	January 15, 2002
Last Updated:	January 3, 2003
Version:	Final

Contents

CHAPTER

1 Overview

Oracle9i Application Server (9iAS) is a powerful and useful server to manage and deploy applications on the Internet and intranet. With this power come a variety of tools to implement a variety of applications ranging from the PL/SQL tool stack to SOAP implementations.

With these features comes the added responsibility of securing each tech stack that is used in the application environment. The security requirements for each will vary depending on application location, application use and corporate security policies.

While many security measures appear to be common sense, the difficulty is in identifying the security holes and knowing the options for securing those holes. This paper specifically addresses the holes presented by David Litchfield in his article "Hackproofing Oracle Application Server" (see References at the end of this document) and will add additional recommendations to his article. This article will also provide the reader with a checklist of tasks to tighten the security of their Oracle9iAS implementation.

Note that this document is intended to provide a set of security recommendations and "best practices" that have been found to be useful in many environments. It is not offered as a cookbook for securing an arbitrary system regardless of application, as many of the recommendations in this paper require that the reader know and understand their application environment. For instance, the steps to secure an application are different for a PL/SQL based application than for a strictly Java-based application. Also, hybrid applications require even more knowledge of the services of your application since disabling one service might affect another application service.

In this article, we will focus mainly on Oracle9iAS in an UNIX environment. Even though this is focused primarily for UNIX, many of the recommendation can be applied to a Windows™ environment.

Lastly, **always test any recommendations suggested in this article before implementing them in a production environment**.

CHAPTER

2 Operating System Security

Overview

Since this article primarily focuses on Oracle9iAS security, operating system security will only be briefly covered. However, since the operating system is a critical piece of web server environment, the operating security is briefly discussed here. There are many good publicly available sources for operating system security and these sites should be reviewed in addition to the recommendations made here.

In this article, we will primarily focus on UNIX (and UNIX-like) systems. With all recommendations listed below, you should consult your system administrator and/or operating system vendor to acquire more information on how to secure their operating system environment, and any specific recommendation they may have.

Security Patches

Visit and review your operating system vendors' security checklists and recommended security patches. Another good idea is to subscribe to your vendors' security newsletters. This will keep you abreast of the latest changes, and available patches.

Services

By default, most UNIX servers provide a variety of services in their /etc/inetd.conf and /etc/services files. These range from system services, such as login, to application services such as smtp. While in many environments these services are useful if not required, in a public webserver environment many of these services are unneeded, and if not disabled will provide potential holes to your operating system.

The list of services required for your webserver should be review by your system administrator and necessary action taken.

A couple of recommended services that should be reviewed include:

FTP/Telnet

FTP and Telnet provide needed access to your operating systems, but are well known services that are often used in attacks. A common recommendation is to run an encryption service such as SSH. What SSH provides is an encrypted transfer of data, so network sniffers will display encrypted packets instead of displaying clear text network packets. SSH can run on any port, so by configuring your SSH server to run on a port other than the default 22 would require a potential intruder to port scan your operating system to determine what port is available. SSH also requires a key combination that is needed for the initial authentication of the connection, which enhances security. More information on SSH can be acquired at http://www.openssh.org.

Port numbers of existing services

Some potential intruders will attack known ports of systems/applications. One way to make it more difficult is to change the port number of many common services. This will require the intruder to perform a port scan of your system, which is detectable and would alert you to potential security attack.

Once the services that are required for a specified host are identified, the system administrator should ensure those services are made as secure as possible.

To do this, it is a good idea are to configure each service so that it has the lowest privileges necessary for that service to operate. The fewer privileges for a service the more secure it will be. Consider implementing enhanced security packages that would be provided by the operating system vendor or third party vendor. One very common enhanced service implementation is the use of tcp_wrappers. Again, contact your operating system vendor for more information on security products and tools available for that operating system.

Remove samples and examples of applications and operating system facilities

Often when operating system systems are loaded or applications are loaded on an operating system, many example and sample files are loaded to assist with education of the system user. On production servers, these samples and examples should be removed. This will prevent the use of a sample application to gain access to the operating system or data source that is used by the sample.

Remove any username and code trees not in use

When managing or administrating a server, it is necessary to remove any users that are not necessary for the management or administration of the operating system or the application running on that operating system. By keeping the access list short, this will reduce the number of potentially cracked passwords, or ex-employees gaining access because their account was not revoked when they left the company.

Also, review any old or unused code that may exist on a system. For instance, an application may have been upgraded to eliminate security holes, but the old code is still available. The old code should be removed, since the good code could be accidentally replaced with the bad code. Also, by removing any unused code trees, it will reduce the number of potential tools available to an attacker.

CHAPTER

3 General Oracle9iAS Security

Overview

Web server security, regardless of the vendor, is common practice among the industry and should be reviewed on a regular basis. There are many resources that are available to assist with recommendations for general web server security and configurations. In this chapter, I will present some suggestions in relation to the Oracle9iAS server, and methods to secure some basic features.

Check MetaLink for Patches

The first thing an administrator should do when addressing security is to check Oracle Technology Network (OTN, http://otn.oracle.com) for any security alerts, and MetaLink (http://metalink.oracle.com)for patches that are available. These often will address issues that have appeared since product release, and will assist in securing your production server.

Remove unused services

This is where intimate knowledge of the Oracle9iAS services that are being used by your application is required.

If your application is strictly a PL/SQL based application, all references to any service but PL/SQL can be removed. Remember, the most secure service is one that is not running.

This can be accomplished by modifying two (2) files:

- ${ORACLE_HOME}/Apache/Apache/conf/httpd.conf

  If the application is strictly PL/SQL, the entry

  include "${ORACLE_HOME}/Apache/Jserv/etc/jserv.conf"

  should be either removed or commented out. The benefit of removing the entry is that there is not a chance of someone uncommenting the entry out by mistake, hence re-enabling the service. The benefit of commenting the service out is that if your application scope changes to include Java based services then all that is required is to remove the previously added comment. The best practice is to comment out the entry and add additional comments on to why the service is commented out.

- ${ORACLE_HOME}/Apache/Apache/conf/oracle_apache.conf

If this application is strictly PL/SQL based, then all entries in this file can be commented out, **except**

include "${ORACLE_HOME}/Apache/modplsql/cfg/plsql.conf".

This entry is required for PL/SQL and will not operate without that entry.

If your application is strictly Java based with no XML usage or PL/SQL references, then the following can simplify the "securing process" by modifying one (1) file:

- ${ORACLE_HOME}/Apache/Apache/conf/oracle_apache.conf

In this file, you can comment out all the entries, **except**

include "${ORACLE_HOME}/Apache/jsp/conf/ojsp.conf" .

This configuration file is used for JSP applications. If you are not using JSP applications, then this entry is not needed.

Note: This only reduces the number of services that you will need to secure and does not secure the remaining services. Instructions to secure each individual service will be available in upcoming chapters.

Also, note that Portal uses both Java and PL/SQL, therefore the services should not be disabled.

Utilize RedirectMatch and Rewrite Rules

RedirectMatch and Rewrite rules can provide a simple method to prevent individuals from accessing restricted areas or potentially open areas. This option is used throughout this document and is a common practice. In Oracle9iAS v1.02.x, there is a documented bug regarding Rewrite and Redirect rules. Instead of the destination address being a simple URI, the destination must be a full URL path. For instance, if you wanted to redirect access to /something to /nothing you would have to do the following:

RedirectMatch ^/something http://my.server.com/nothing

When utilizing Allow/Deny rules, use IP addresses

When configuring Location and Directory directives, it is possible to allow or deny access based on hostname, ip address or subnet. To prevent a DNS spoof, use IP addresses when possible for the Allow/Deny rule.

Reduce the Timeout setting

One common method of executing a Denial of Service attack is to continually send packets to a webserver. What happens is that the server will create a client to service the request, and will remain available for a period of time (Timeout). Denials of Service attacks rely on this feature and will send a mass of packets to a site, locking all of the clients into a service state. Now, there are no more clients available to service any new requests. This makes a server appear to be unavailable. These services will not become available again until the timeout is reached. Once the timeout is reached, the connection is closed and new requests can be processed.

This value can be set in the

${ORACLE_HOME}/Apache/Apache/conf/httpd.conf

file by changing the parameter TimeOut. This should be set as low as possible for your application. A common value is 300, which is 5 minutes.

Configure appropriate logging to identify attacks

Oracle9iAS server provides a variety of logging services to help identify who has accessed your system, when and what was their request. These logging features utilize the normal Apache logging parameters and should be configured to log, at the minimum, server access.

Along with logging, there are many tools available to analyze those logs. These should be considered to assist in your log file analysis.

Protect Administrative URIs

Within Oracle9iAS, there is a number of URIs that provides administrative or testing functionalities. These URIs will often provide information about various services and will provide information on the server and its configuration. While useful in debugging, this information would assist someone who is trying to determine your webserver configuration.

These tools can often be secured by either using rewrite rules or allow / deny rules. There are definite benefits to each. Rewrite rules will send all requestors, including administrators, to a site / location of your choosing. Unfortunately, some URIs cannot be protected via allow / deny rules. For those URIs, rewrite rules are the only available choice for protecting access.

Below is a list of URIs that should be protected via rewrite rules:

RewriteEngine On
RedirectMatch ^/dms0 http://my.server.com/errors/forbidden.html
RedirectMatch ^/dms0.*$ http://my.server.com/errors/forbidden.html
RedirectMatch ^/servlet/DMSDUMP http://my.server.com/errors/forbidden.html
RedirectMatch ^/servlet/DMSDUMP.*$ http://my.server.com/errors/forbidden.html
RedirectMatch ^/servlet/Spy.*$ http://my.server.com/errors/forbidden.html
RedirectMatch ^/fcgi-bin/echo http://my.server.com/errors/forbidden.html
RedirectMatch ^/fcgi-bin/echo.*$ http://my.server.com/errors/forbidden.html
RedirectMatch ^/fcgi-bin/echo2 http://my.server.com/errors/forbidden.html
RedirectMatch ^/fcgi-bin/echo2.*$ http://my.server.com/errors/forbidden.html

As the security measures for individual services are discussed, this list will change. Security implementation is application and customer specific, and the degree of protection implemented will vary based on the application, the customer's security budget, and their tolerance for risk.

Remove any samples and examples from the Oracle9iAS code tree

Oracle9iAS provides a number of samples and examples to assist application developers in designing and developing new applications. On production servers, these samples / examples should be removed. This will reduce the number of potential access points to your webserver environment.

The following is a list of samples / examples that should be removed. Note that, depending on the applications you have installed on your system, other samples / examples may exist; these should also be considered for removal.

- ${ORACLE_HOME}/Apache/Jsdk/examples
- ${ORACLE_HOME}/Apache/fastcgi/examples
- ${ORACLE_HOME}/Apache/BC4J/samples
- ${ORACLE_HOME}/panama/WebIntegration/Server /samples
- ${ORACLE_HOME}/panama/WebIntegration/Server/HmUtil/samples
- ${ORACLE_HOME}/panama/sample
- ${ORACLE_HOME}/ldap/demo
- ${ORACLE_HOME}/soap/samples
- ${ORACLE_HOME}/Apache/Jserv/servlets/Hello*
- ${ORACLE_HOME}/Apache/Jserv/servlets/IsIt*

In the ${ORACLE_HOME}/Apache/Apache/conf/httpd.conf file the DocumentRoot directory is set by default to ${ORACLE_HOME}/Apache/Apache/htdocs. This should either be set to the applications DocumentRoot or should be protected in the following manner:

```
DocumentRoot "${ORACLE_HOME}/Apache/Apache/
<Directory />
   AllowOverride None
   Order deny,allow
   Deny from all
</Directory>
<Directory "${ORACLE_HOME}/Apache/Apache/htdocs">
   Options Indexes FollowSymLinks
   AllowOverride None
   Order deny,allow
   Deny from all
                        </Directory>
```

This will protect all of the documents that are pre-installed with the Oracle9iAS installation. This will save you some time in removing all the content from under that directory.

Please note the ${ORACLE_HOME} will be the actual directory tree for your Oracle9iAS installation.

Delete all user and codes which are not used

Analogous to the recommendations in the Operating System Security chapter, any users and code trees that are defined in the Oracle9iAS environment should be limited and managed actively. This means that any users that are no longer required to exist on the system should be removed, and any old or unused code should be removed from your system.

Port Management

Normal webserver operation depends on two ports: 80 for http traffic, and 443 for SSL traffic. The recommendations in this section are similar to the services section, meaning, do not run services you do not need on other ports.

While it would be easy to have a testing and production environment on the same machine utilizing different ports, this provides more entry points into your system and is therefore not recommended.

For example, you may wish to run a configuration on the normal webserver ports 80 and 443, but also want to run services on 82 and 444 to test other configurations. This would require that all of those configurations are as secure as your normal production environment, which in turn may cause a duplication of work and create unknown exposure as new code is introduced.

Run the httpd daemon as nobody/nobody

By default, Oracle9iAS is installed with the running user of the httpd daemons as *oracle* with a group of *dba*. This should be changed to a user with little or no privileges. This will help prevent an attacker who would use a buffer overflow attack from being able to doing something that would require a greater amount of privileges. It is recommended that this user be nobody (the lowest user in Unix).

This can be accomplished by modifying the ${ORACLE_HOME}/Apache/Apache/conf/httpd.conf file:

```
User nobody
Group nobody
```

Under no circumstances should the User be root or anyone with root-like privileges. This is dangerous as the httpd daemon would have all the privileges of *root*. The user that is chosen should only have enough access to manage the application.

Replace standard error pages with custom error pages

By providing customized error pages, this allows the administrator to have control over what information is provided to the potential user.

To accomplish this, the custom pages should first be designed. Implementation of custom error pages requires the modification of the ${ORACLE_HOME}/Apache/Apache/conf/httpd.conf file adding the following:

```
ErrorDocument 204 /errordocs/204.html
ErrorDocument 400 /errordocs/400.html
ErrorDocument 403 /errordocs/403.html
ErrorDocument 404 http://some.server.com/cgi-bin/404.cgi?404
ErrorDocument 500 /errordocs/500.html
```

Note: On the ErrorDocument 404 you can actually reference another application/url. This can be useful if you are tracking the number of errors coming from a specific set of sites, or urls, allowing for further site management.

Remove any unnecessary directories from the httpd.conf file.

When reviewing your security, you should review the ${ORACLE_HOME}/Apache/Apache/conf/httpd.conf file for any directory references that you may not be using. This is especially true of intrusive directories such as cgi-

bin, etc. which allow script execution. If these are not in use by the main application, they should be commented out of the file. The most secure directory is one that isn't available.

Remove actual server name from the ServerName directive

One key bit of information that a potential intruder will need to start probing a system is the name of a system. This is especially true if you are using an external load balancer such as F5 BigIP or Cisco Local Director that would provide IP masquerading. To avoid giving the system name to a potential intruder, you can use your application name instead of the actual server name in the

${ORACLE_HOME}/Apache/Apache/conf/httpd.conf file:

ServerName my.app.com

instead of

ServerName my.server.com

Remove the actual banner from Oracle9iAS.

When a customer or user accesses a site that does not have custom pages or an error occurred, Oracle9iAS will display the servername in the ServerName reference as well as the version number of Oracle9iAS. This can be demonstrated by executing the following simple command:

```
telnet my.server.com 80
Trying 1.2.3.4
Connected to my.server.com
Escape character is '^]'.
get /
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>501 Method Not Implemented</TITLE>
</HEAD><BODY>
<H1>Method Not Implemented</H1>
get to /index.html not supported.<P>
Invalid method in request get /<P>
<HR>
<ADDRESS>Oracle HTTP Server Powered by Apache/1.3.12 Server at hosted.server.com>
</BODY></HTML>
Connection closed by foreign host.
```

What an intruder can tell from this is the following:

- The server is running Oracle HTTP Server version 1.3.12
- The server is actually running on hosted.server.com

From this information, a potential intruder can search for current vulnerabilities of Oracle HTTP Server 1.3.12, as well as Apache. The intruder also knows what server to attack since it is clear in the telnet output (i.e. hosted.server.com)

To limit the information provided to a potential intruder, you should modify

${ORACLE_HOME}/Apache/Apache/conf/httpd.conf

and make the following changes:

ServerSignature Off

ServerTokens Prod

By doing this you have now removed that information from a telnet query, as show below:

```
telnet my.server.com 80
Trying 1.2.3.4...
Connected to my.server.com
Escape character is '^]'.
get /
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>501 Method Not Implemented</TITLE>
</HEAD><BODY>
<H1>Method Not Implemented</H1>
get to /index.html not supported.<P>
Invalid method in request get /<P>
</BODY></HTML>
Connection closed by foreign host.
```

Notice that all that extra information is now gone. This will make the potential intruder have to find another way to identify your system. Please note that this will only remove the version information. This will still display the following on services such as Netcraft®:

Oracle HTTP Server Powered by Apache

Place all files in a location inaccessible by the DocumentRoot, AliasMatch, LocationMatch directives unless explicitly needed

It is critical to remove access to any file unless it is explicitly needed for the external presentation of the application. This would especially include password files to protect directories, configuration files, log files or any other file that is not necessarily needed for the external presentation of the application. By removing access to all files except for the ones explicitly needed, the administrator is controlling the view of the user. The most secure directory is one that doesn't exist. By making the directories inaccessible, they will not exist to the external user.

Remove any unnecessary Directives, such as document directories, etc.

This has been partially covered above. In addition to any example or sample directories, you should remove any references to documentation directories or added directives that are not needed to operate your application. If a directive is not critical for your application (or the server itself), then it should be commented out.

Two directives that should be commented out on a production server are the following:

- Alias /jservdocs/ "/u01/app/oracle/product/IAS10222/Apache/Jserv/docs/"
- Alias /soapdocs/ "/u01/app/oracle/product/IAS10222/soap/"

Turn off Indexing of Directories.

There are two goals when protecting your webserver:

- Reduce the amount of information available
- Reduce the amount of access to non application related areas

Directory indexes will display the contents of a directory if there is not an index.htm or similar file available. By commenting this out, you will prevent an intruder from viewing the files in a directory, potentially finding a file that may be of use in their quest to access your system. The quickest way to disable this feature is to modify

${ORACLE_HOME}/Apache/Apache/conf/httpd.conf

and comment out the following line:

IndexOptions FancyIndexing

If directory indexing is required, then it is recommended that this be set only at the Directory level using:

<Directory /u01/app/something>

IndexOptions FancyIndexing

</Directory>

In this case, only files in the above location can be browsed.

Always test before you implement

As always, these recommendations should always be testing in your application environment before implementing on a production server.

CHAPTER

4 PL/SQL Security

Overview

PL/SQL access from a webserver has been available since the early days of Oracle webservers. Most users who have experience with Oracle web or application servers have managed, developed or used an application that utilizes the PL/SQL toolkit. Securing PL/SQL Data Access Descriptors (DADs) is critical for securing applications that use the PL/SQL toolkit. Securing PL/SQL DADs requires a combination of good DBA skills and good web server knowledge.

Remove Administration access

In Oracle9iAS, there is a nice tool that allows an administrator to create and administer DADs. This tool is available by default and utilizes a default configuration in the ${ORACLE_HOME}/Apache/modplsql/cfg/wdbsvr.app file. There are a number of methods of protecting this.

One method is not to use this tool at all, therefore disabling any threat. This can be accomplished by removing the entries from the ${ORACLE_HOME}/Apache/modplsql/cfg/wdbsvr.app file. There is a definite benefit to this solution. If the administration tool is disabled, your configuration is inaccessible over the web. The configuration tool is quite easy to use - both by administrators and by potential intruders. It displays information on your database connect strings and usernames that are configured to access those DADs. This would provide an intruder with two pieces of valuable knowledge: the connect information for a database, and a username. All they need now is a password and they are in your database.

Unfortunately, there are some drawbacks to this approach. The easy-to-use tool for configuring access to your database is now unavailable.. You also lose the encryption of DAD passwords that the tool provides. There exists a command line executable that can encrypt the passwords for you, so encryption of the passwords is still possible even if the tool is disabled.

The second method is to secure the DAD administration screen with a username/password that has no privileges on the database. By configuring the username without the password, an individual wanting to gain access will need to know the password of the DAD administration user. Since the site is protected, and the DAD administration user has limited privileges, the risk of break in and damage to your database is minimal. Unfortunately, if they guess the password, then they have access to the administration screen and can modify your DAD configurations.

The third method is to protect without a username/password. An intruder must guess a username and a password, but as Mr. Litchfield points out in his article (see references), there are many username/passwords that are installed by default, and an intruder can gain access by just traversing the list and using the default username/password combinations until one works. Since the PL/SQL DAD only relies on a correct username/password, access to the administrative tool would be granted. In this case, it is more difficult to gain access if the database's default username and passwords have been changed.

Protect PL/SQL toolkit

The PL/SQL toolkit is a set of packages that allow a developer to create a PL/SQL procedure or package to be executed over the Internet. With the introduction of Oracle 9iAS, the toolkit is now owned by SYS and should be secured using multiple mechanisms.

The first step is to include all of the PL/SQL toolkit packages in the exclusion_list of the ${ORACLE_HOME}/Apache/modplsql/cfg/wdbsvr.app file. Also, it is good practice to include packages owned by *sys*, packages that are *dbms* packages or *account* packages.

The exclusion list will throw a 403 or Forbidden access exception if anyone attempts to access packages in that list.

The second step is to find procedures and packages executed by the application. To determine these procedures and packages, log into your database as the username and password as entered in ${ORACLE_HOME}/Apache/modplsql/cfg/wdbsvr.app.

Once logged in, run the following SQL:

select distinct substr(1,3,name) from user_source;

This will provide you with a list of the first three letters of the packages and procedures that the DAD user executes. With that list, you can expand your exclusion list even further.

This should be done for each DAD in the ${ORACLE_HOME}/Apache/modplsql/cfg/wdbsvr.app file.

Once that is accomplished, the next step is to add a few location directives in the ${ORACLE_HOME}/Apache/modplsql/cfg/plsql.conf file. This should be similar to the following:

<Location /pls/<DAD>/<exclusion list>>
SetHandler modplsql
Order deny, allow
Deny all
<Location>

This will now reject any incoming request that isn't available in the DAD.

Lastly, create a RedirectMatch. Create a rewrite rule to rewrite every request that is in the exclusion list to a custom page. This RedirectMatch should be entered before the include "${ORACLE_HOME}/Apache/modplsql/cfg/plsql.conf" entry in the httpd.conf file.

For instance, let's say you have an application that tracks cars, and you have decided to name all your packages and procedure to begin with the word "car." The first thing you would do is create the exclusion list. By default, you should always include the entries that David Litchfield mentioned in his report which are:

account*, sys.*, dbms_*, owa*

If you really wanted to exclude any entries, you could do this instead:

Exclusion_list=a*,b*,d*,e*,f*,g*,h*,i*,j*,k*,l*,m*,n*,o*,p*,q*,r*,s*,t*,u*,v*,w*,x*,y*,z*

I would also include the Capital letters as well, just in case. This will only allow packages that start with C to be executed. Since the standard was for all packages and procedures to start with the word "car," this was quite simple to protect.

Next, we will add the location directive to the ${ORACLE_HOME}/Apache/modplsql/cfg/plsql.conf file.

<Location /pls/car/[A-BD-Za-bd-z][b-zB-Z][a-qs-zA-QS-Z].*$>
SetHandler modplsql
Order deny, allow
Deny all
</Location>

Here what we did was to spell out the word "car" in multiple ways, so if CAR, Car, CaR, cAr, etc., is entered, the entry is allowed, else the request is denied.

Lastly, we will add a RewriteRule. Again, this should come before the inclusion of the plsql.conf file. This file is located in the oracle_apache.conf file which is included in the

${ORACLE_HOME}/Apache/Apache/conf/httpd.conf file

Rewrite on
RedirectRule /pls/*/[A-BD-Za-bd-z][b-zB-Z][a-qs-zA-QS-Z].*$ http://my.server.com/errordocs/forbidden.html

Each of these methods works well independently, and works well together. Further discussion of security issues associated with the PL/SQL toolkit may be found in Oracle Security Alert #28, available on OTN.

Remove un-needed DAD configurations

One interesting feature of PL/SQL is that any VirtualHost or Server configuration can access and utilize the dads listed in the ${ORACLE_HOME}/Apache/modplsql/cfg/wdbsvr.app file. This provides easy configuration especially for applications that utilize the same DAD configuration to access data. It is important to remove any un-needed DAD configurations. Old configurations tend to be forgotten and will sometimes leave a hole where one was not intended. The most secure DAD is no DAD. If you don't need it, remove it.

Restrict DAD access

This has been covered regarding PL/SQL toolkit access. The same rules apply to individual DAD configurations or non-existent DADs. One recommendation is to create location directives and rewrite rules to exclude all attempts except for the DADs you have listed in the

${ORACLE_HOME}/Apache/modplsql/cfg/wdbsvr.app.

The location directive will only allow traffic for the entries you made and exclude any new entries that may not be authorized.

Limit privileges of DAD owners

The owner of the application in the database should only have enough permission to execute the application. Also, never give a DAD owner DBA privileges. This is similar to running Oracle9iAS as *root*. This would allow the user full permissions to do anything in the database.

If you are using database authentication to secure your database, the PRODUCT_USER_PROFILE table should be used to restrict the ability to create new objects, delete object and most DDL commands. This feature of the Oracle Database is very nice and can really close some holes that might otherwise exist in an application.

DAD users must be sufficiently constrained on the database that if someone did guess the password, little or no damage should come from that.

Use administrative tool to encrypt passwords

If you have properly secured your administrative pages using database authentication, then the administrative tool is a good resource to encrypt passwords. That way if someone did get your wdbsvr.app file, they would still have to guess the password to get to the no privilege account. (see above). If you don't want to have the administrative tool available on your system, consider using the command line tool to encrypt the passwords. It is just as good for satisfying this requirement, and you do not need to have the administration screen available.

Apply PL/SQL Patch

In early 2002, Oracle released a PL/SQL toolkit patch to resolve security problems associated with cross-site scripting and buffer overflow (these are described in Oracle Security Alert #28 on OTN). If you haven't applied this patch yet, this is a must have and should be applied immediately. This does require a PL/SQL toolkit upgrade, but fixes quite number of issues, and is worth the effort. The patch number is 2209455 and is available at http://metalink.oracle.com

Always test before you implement

As always, these recommendations should always be tested in your application development environment before implementing them on a production server.

CHAPTER

5 Java Security

Overview

With the popularity ever increasing for Java applications, it is necessary to understand and secure your Oracle9iAS Java processes. Oracle provides a wide range of Java tools, built on JSPs and Servlets. As Java technology increases in use, the Oracle9iAS Java technology stack is sure to grow in use as well.

JServ

Secure Administrative/Informative URIs

With Oracle 9iAS™, Oracle introduced tools for the automatic monitoring and management of Jserv processes. Those who have ever configured Jserv for manual mode know how difficult it is to perform manual JDK death detection and reactivation, so automatic management is a welcome feature. Many tools have been provided in Oracle9iAS to monitor and manage those processes. These tools should be secured and monitored so private configuration information is only accessed by members of the administration team.

Some of the important URIs are listed below and should be secured. The most logical method of securing these URIs is via the allow / deny portion of the Location directive. In this directive, only the server hosting the machine, any virtual IP address on the machine, and any administrative machines should be able to access these URIs with all others being restricted. I will go over what each URI does and show the directive to protect it. Note that these are normal allow / deny directives that are used in the Oracle9iAS and Apache. These entries are all located in the ${ORACLE_HOME} / Apache / Apache / conf / httpd.conf file.

Server Status – This provides statistics on the machine and the current status of each process.

```
<Location /server-status>
   SetHandler server-status
   Order allow,deny
   Allow from only_those_I_trust

</Location>
```

DMS – This tool provides information on the number of requests a Java process is receiving, the number of instances available and the number of processes in use.

```
<Location /dms0>
 SetHandler dms-handler
 Order allow,deny
 Allow from only_those_I_trust
</Location>
```

Oproc manager – This is the process that monitors and manages Java processes and determines when to restart them if they have died. Without appropriate entries for oprocmgr in the httpd.conf file, your Java processes will not start correctly. The allow directive is very important as any Java process has to be able to access this service (which it does through httpd). If a process cannot access Oproc manager or access is denied, then the process will not run.

Be sure to include all the IP addresses that are in use on the hosting server. This means that if you use 2-3 IP addresses to do multi-hosting, those IP addresses must be in the Allow list. Also, if you add a Deny all after the allow, it will also cease to function.

```
<Location /oprocmgr-service>
 SetHandler oprocmgr-service
 Order allow,deny
 Allow from only_those_I_trust
</Location>

<Location /oprocmgr-status>
 SetHandler oprocmgr-status
 Order allow,deny
 Allow from only_those_I_trust    </Location>
```

Jserv – This is the only entry that is not located in the ${ORACLE_HOME}/Apache/Apache/conf/httpd.conf file. This is located in the ${ORACLE_HOME}/Apache/Jserv/etc/jserv.conf file. This provides a user with statistics on the running Jserv processes, ranging from memory usage to CLASSPATH configuration. This should be restricted as it provides configuration information.

```
<Location /jserv/>
 SetHandler jserv-status
 order allow,deny
 allow from only_those_I_trust    </Location>
```

Security Allowed Addresses

There is an entry in the associated properties file of the Jserv that allows for the restriction of IP addresses that can access the running JDK. This entry is security.allowedAddresses and is similar to the allow directive in the httpd.conf. This entry is located in the ${ORACLE_HOME}/Apache/Jserv/etc/jserv.properties file and should contain the IP addresses that will be accessing the server. Normally, this will only be local host and perhaps a Virtual IP address if you are adding Virtual Interfaces to your machine.

```
security.allowedAddresses=127.0.0.1, myhost.server.com
```

Manual method

If you don't want to go through the effort of securing the status and monitoring URIs above, you can always run in manual mode. In this mode, the status and monitoring URIs provided for use in automatic mode are not used, and are typically replaced by custom-developed status and monitoring applications. These can be shell scripts or other programs that monitor and manage the Jserv processes. It is usually much simpler to run in secure automatic mode than configure in manual mode and create the monitoring yourself, but since this is an available option I feel obligated to present it.

J S P

Connect Strings, Usernames and Password

When developing JSP applications, you should never store username, password or connection information in your JSP. This is because JSP compilation can expose this information. When a JSP is requested, the JSP is taken by the Jserv engine and is converted to a .java file, then compiled into a .class servlet file. After that is complete, the .class file is then executed as a servlet and information is returned to the user. During that compilation process, the username, password and connection information is stored in the .java file that is generated and can be accessed. So if someone was able to get the .java file, they would have all they would need to connect to your database. A much better idea is to use a properties file and store username, password, and connection information in there. As mentioned above, this file should be stored somewhere outside the scope of the presentation portion of the Oracle9iAS server to prevent someone from downloading the file.

Protecting your generated .java and .class files

In the article that was presented by David Litchfield, it is recommended to write a Location directive to protect the _pages directory where the JSP generated files are stored. In a single application environment, this is a very good idea and recommended. Another solution that will work in a single or multi-application environment is the user of a RedirectMatch. This can be easily achieved in your redirect file by adding the following:

RedirectMatch ^/_pa.*$ http://my.server.com/errordocs/forbidden.html

This should resolve the issue of individuals trying to download the JSP generated files.

S e r v l e t s

Class file protection

Similar to the JSP generated files, Servlets are just .class files. Now while these are compiled code, there are reverse engineering tools available to de-compile these files. So it is recommended to only allow access to the Class files that make up your application. This can be achieved with a RedirectMatch. This is similar to the RedirectMatch that we did with the PL/SQL in which we only allow the Class files that are used by the application.

G e n e r a l

ClassPathing

Since many features are made available to the developer using Oracle9iAS, there are a lot of classes and groups of classes included in the ${ORACLE_HOME}/Apache/Jserv/etc/jserv.properties file. While many of these are convenient, they might provide unwanted access to areas or pieces of the configuration. If a classpath is not needed by your application, don't use it, and don't include it. A potential intruder cannot execute a class file that doesn't exist.

While I would like to give you a list of what to comment out, I cannot, as each application has requirements for execution. This file is pretty well documented and explains what each section is used for. If you are not using it, you will probably be pretty safe to comment it out.

A l w a y s t e s t b e f o r e y o u i m p l e m e n t

As always, these recommendations should always be tested in your application development environment before implementing them on a production server.

CHAPTER

6 XSQL/SOAP Security

SOAP

Overview

Simple Object Access Protocol or SOAP is a mechanism where objects and data can be shared via services using XML. This allows XML requests to be generated and submitted to a SOAP service, and that service will return the requested data. While this is an interesting technology, it is still a young technology and there are some security considerations that should be observed when using it. Oracle provides a SOAP configuration with its Oracle9iAS product. This configuration is by default open and should be reviewed and secured before use in a production system.

Protect SOAP-SERVICE-MANAGER

SOAP requests are managed via requests to specific URL/URIs which manage requests for those services that are available. A request is submitted to the URL/URI SOAP service, the request is processed and the data is returned. The SOAP service manager allows any class file that matches the requirements of SOAP to be registered as a SOAP service, making it available to process requests. The class file has to match a certain number of requirements which are listed below:

- The class file has to be available to the SOAP servlet. This means that it either needs to be included in the CLASSPATH of the soap.properties file, or it is located in the directory that serves that the SOAP servlet repository.
- The class file has to have a public no args constuctor
- The method in the service class that is accessible by remote clients must have all arguments deserializable and return values serializable. Oracle9iAS configuration contains the serializers/deserialisers for a number of Java types and can be found in the Oracle9i Application Server Release Notes Version 1.0.2.2.2.

If the classes match these requirements, then through the service manager, they can be registered/deregistered as SOAP services.

With that said, there are a few things that you need to be cautious of.

The URI that is used to access these available services is the same as the service manager URI. If this is not protected, then individuals can register and deregister existing services as well as create new services if class files are uploaded. This can be an administrative nightmare, since anyone could enable or disable a service without the knowledge of the administrator.

There are a couple steps that can be done to protect the service manager or remove the ability to enable or disable services.

- Specify a specific URI that should be used to access the serviceManager. This can be done by editing the ${ORACLE_HOME}/soap/webapps/soap/WEB-INF/config/soapConfig.xml file. To specify a specific URI, you will want to modify the following entries below:
  - Under serviceManager: <osc:option name=”requiredRequestURI” value=”your new uri” />
  - Under providerManager: <osc:option name=”requiredRequestURI” value=”your new uri” />

Now, create specific Location directives in your ${ORACLE_HOME}/Apache/Apache/conf/httpd.conf file that would only allow access from a location or password protect those locations. This would prevent any individual from viewing your existing services or modifying any configuration of those services.

- If the step just described is not an option, then the next choice is to utilize the concept of pre-deployed services and providers. In Oracle9iAS, there is an option to enable/disable the ability to add/delete/enable/disable new services. This can be accomplished by modifying the ${ORACLE_HOME}/soap/webapps/soap/WEB-INF/config/soapConfig.xml. The entry to modify is in the serviceManager area and consists of the following
  - <osc:option name="autoDeply" value="false">

By default, this is set to true. Setting this to false enables the pre-deployed services and will not allow anyone to enable or disable services, or add services. This does provide security in that it disables one way to modify services, but it still provides access to the serviceManager for individuals who wish to snoop.

Do not allow requests to be forwarded

As recommended in the W3C article "Simple Object Access Protocol (SOAP) 1.1" (see references), a service should not have the capability to forward requests beyond itself. XML requests can embed multiple service paths, similar to source routing of network packets, to service providers. What this can do is allow a request to come into a service that is being hosted, and potentially be forwarded on to a rogue service. The recommendation is to have requests die at your service, preventing any request forwarding.

Disable if not in use

If you are not using SOAP in any application you are running, it is best to disable the service all together. The simplest way to do this is to modify the ${ORACLE_HOME}/Apache/Jserv/etc/jserv.conf file and comment out the following:

```
# SOAP Entries
ApJServGroup group2 1 1 ${ORACLE_HOME}/Apache/Jserv/etc/jservSoap.properties
ApJServMount /soap/servlet ajpv12://localhost:8200/soap
ApJServMount /dms2 ajpv12://localhost:8200/soap
ApJServGroupMount /soap/servlet balance://group2/soap
```

Protect soapConfig.xml

Since this file holds all the configuration information for SOAP, this is a file worth protecting. By default this is in an accessible area. This file should be moved to a non-presentation area of the Oracle9iAS server, and the location should be noted in the soap.properties file. If this is not an option, another suggested security technique is to place a Directory directive in the ${ORACLE_HOME}/Apache/Apache/conf/http.conf file and only allow access to that directory from your localhost or private subnet. For more information on SOAP configuration-related security considerations in 9iAS, refer to Oracle Security Alert #22 (available on OTN).

X S Q L

Overview

XSQL is enhanced tool provided by the Oracle9iAS product that allows access to data sources via SQL and presenting the data in XML format.

Disable if not in use

In his paper (see references), Mr. Litchfield describes a method of acquiring a XSQLConfig.xml file via the XSQLServlet. This is possible because of the inclusion of two files in the classpath of the servlet: One being oraclexmlsql.jar and the other being oraclexsql.jar. By default these are included in the standard configuration of Oracle9iAS. If you have configured a separate servlet for your XSQL applications, then comment this entry out on all other servlets, to prevent the access of that file.

Also, if you are not utilizing XSQL, then the XSQLConfig.xml file would most likely contain no useful information, but this should be commented out just the same.

Protect the XSQLConfig.xml file

Now this is a tricky one. Because the servlet engine executes the package ignoring rewrite rules and httpd directives, this cannot be protected via our normal methods. The best recommendation is to move the file to a location other than the common ${ORACLE_HOME}/xdk/lib directory. This will at least make the hunt for the file a little more exciting. For more information please refer to Oracle Security Alert #28 (available on OTN).

Always test before you implement

As always, these recommendations should always be tested in your application development environment before implementing them on a production server.

CHAPTER

7 Application Security

Overview

Usually when applications are designed, security is the last item on the checklist. Customer requirements, functionality and usability are normally on the top of the list. Once these are satisfied, there is often no time left for security considerations, especially if security has not been taken into account during application design. When designing an application with Oracle9iAS, there are a number of good rules that can be applied to improve security with relatively little impact to functionality and usability.

General

Passwords, Users, and configurations

One common mistake is not utilizing property and configuration files, or not securing them correctly. As mentioned in the Oracle9iAS section, all configuration files, etc, should be stored outside the presentation scope of Oracle9iAS. Configuration files can be included in a Jserv configuration from any location. Also, as mentioned previously, do not store clear text usernames, passwords or connection information in your code itself, as this can be easily de-compiled and the sensitive information obtained.

Restrict Downloads

Unless absolutely necessary, restrict all downloads and DirectoryIndexing unless they are required by the application, then restrict and protect those directories using Location or Directory directives. Do not set DirectoryIndexing globally.

Monitoring Scripts

Any monitoring scripts that are viewed over the web should be secured through Location or Directory directives using Allow / Deny rules. These should only be accessible by the local host and any monitoring system that you are using. All other access should be restricted.

Reduce the amount of information provided

Make sure that the data you are providing is the data you want your user to see. All errors should present one of the standard error pages. A user should not see any code related error messages or Java exception messages, as this will provide a potential intruder with information on class trees, class names and locations. When dealing with error messages, less data is more. If you can control the data your user community sees, then you will protect yourself greatly. If the information is needed for debugging purposes, consider sending the information to log files.

SSL

Consider using SSL for your confidential data like username, password, etc. A common hacker trick is to locate a webserver that is offering this information in non-SSL format, and place a network sniffer on that line. After that, it is a matter of sitting back and collecting data - sooner or later, the snooper will get what they are looking for. SSL makes this much more difficult since data is protected by encryption, hence many give up the chase.

Location

Choosing a suitable network location for your webserver is a crucial security decision. If you manage content for external viewing, consider in investing in a firewall technology. This can be very useful in preventing unwanted snoops of machines, ports and applications. This can also be very effective in implementing a DMZ, in which there is a firewall on both sides of your webserver, one on the Internet side, one on the intranet side. This can ensure that the only Internet traffic which crosses the first firewall is that directed to webserver ports 80 and 443, and this data cannot pass through the second firewall. It is a common misconception that the use of WebCache eliminates the need for a firewall. This is incorrect as traffic still flows to the origin server (Oracle 9iAS), and the server is still available, if Webcache is not. Firewalls are good investments if you have an external presence. It can also be useful in protecting your data source, as is discussed in the next section.

Data Source

If you implement the DMZ approach as mentioned above, it is then possible to have your data source completely within your firewall (intranet). This allows for your webserver which is in between firewalls to access the server, but any requests from outside the firewall infrastructure (Internet) to that data source is refused. So even if someone did get information on your database, they would have to break through one firewall, break into your hosting machine, then guess the username/password combination for your data source. That is a lot of work, and often most individuals would be discouraged at that point. Another recommendation is to reduce the number of data source locations that are accessed. If there is only one hole in the internal firewall, it is harder to find than if there are multiple holes available. This will also help in intrusion detection. If there is only one way into your internal network, it is pretty easy to figure out how someone entered, versus if there are multiple paths.

Presentation Data

You should always keep a back up of your presentation and configuration data in a secure location. This will provide you with the ability to quickly recover from a defacing attack, by just restoring the data from the secure location.

Remote publishing tools

Another recommendation is to resist using remote publishing tools. If you use remote publishing tools, other individuals could potentially use the same tools to publish to your site, hence changing the appearance or content of your data.

Analyze all CGI applications

CGI scripts are a easy and quick method for distributing information. CGI applications are also among the most notorious for allowing unauthorized access. It is good practice to analyze each CGI application to ensure that the following cannot be done:

- Verify that if passing parameters to the CGI, it doesn't provide information that you do not wish advertised. A couple things to try: display your password file, display other files, etc.
- Verify that if a large amount of data is passed to the CGI, it handles the request appropriately with error messages, and doesn't crash. This can prevent buffer overflow attacks, which can otherwise be exploited to crash the system or execute unauthorized instructions.

Consider Load Balancing

There are many commercially available load balancing tools and products. These allow many servers to service the requests of one application, and perform IP masquerading. The benefit is that there is only one URL known which doesn't actually manage the service, but passes service request to a pool of middle tier servers which actually manage the request. A direct benefit is a redundant configuration where if one server is attacked, it can be removed from the pool of servers managing the application until the affected server is secured. This provides a consistent image to the external community of application availability. This type of service, similar to firewalls, is worth the investment.

Always test before you implement

As always, these recommendations should always be tested in your application development environment before implementing them on a production server.

CHAPTER 8 Conclusion

While every system is vulnerable to attack, there are steps you can take to minimize the probability of attacks being successful, and also to limit the damage that may occur if an attacks are successful. As shown in this article, there are primarily two methods to reduce the impact of an attack.

One method is to limit the amount of information provided to a would be intruder. The second method is to reduce the number of options available to the intruder access a system. By implementing each of the methods to the greatest extent for your operating system, webserver and application, the ability of potential intruders to access your systems and application is greatly decreased. As new technologies are created, and new information is accessible via these technologies, the need to address security concerns will increase. Schedule security audits often to assist in identifying potential areas of concern, then address those concerns.

In appendix 1, I have included a checklist of the recommendation provided in this paper for convenience.

APPENDIX

1 Security Checklist

General

Always test before you implement

Operating system

- ❑ Apply operating system Vendor recommended security patches
- ❑ Disable unused or unneeded services
- ❑ Remove any operating system samples or example code
- ❑ Remove any users that are not in use
- ❑ Consider moving to SSH and disable telnet/ftp

General Oracle9iAS

- ❑ Check OTN for relevant Oracle Security Alerts
- ❑ Check MetaLink for patchset releases
- ❑ Remove unused services
- ❑ Utilize RedirectMatch and Rewrite Rules
- ❑ When using Allow/Deny rules, use IP addresses
- ❑ Reduce the TimeOut Setting
- ❑ Configure appropriate logging
- ❑ Protect Administrative URIs
- ❑ Remove any samples and examples from Oracle9iAS code tree
- ❑ Delete all users and code not in use
- ❑ Configure only the ports that are necessary for the application
- ❑ Change user of httpd daemon to unprivileged user
- ❑ Replace standard error pages with custom error pages
- ❑ Remove any unnecessary directory references from the httpd.conf
- ❑ Remove actual hosting server name from httpd.conf
- ❑ Disable Oracle9iAS banner from server
- ❑ Place all files that are not necessary for the application in a location inaccessible by a web user
- ❑ Remove any unnecessary directives such as documentation directives, etc.
- ❑ Turn off indexing for directories

PL/SQL

- ❑ Remove or Restrict Administration access
- ❑ Protect PL/SQL Toolkit
- ❑ Remove un-needed DAD configurations
- ❑ Restrict DAD Access
- ❑ Limit Privileges of DAD Owners and Users
- ❑ Encrypt DAD Passwords
- ❑ Apply necessary toolkit/PLSQL patches

Java

- Secure Administrative and Informative URIs
- Restrict Allowed Addresses
- Remove all username, passwords, connect strings
- Protect generated .java and .class files
- Protect application .class files
- Remove any unnecessary class pathing

XSQL/SOAP

- Protect SOAP-SERVICE-MANAGER
- Do not allow forward requests
- Disable if not in use
- Protect and move the soapConfig.xml
- Protect and move the XSQLConfig.xml
- Disable XSQL by removing classpath entries

Application

- Remove passwords, usernames and connection information from source code
- Protect configuration files by moving them outside the presentation area of Oracle9iAS
- Restrict downloads and directory indexing
- Ensure that any monitoring scripts that are accessed via the web are protected via Allow/Deny rules
- Reduce the amount of information provided when application errors occur.
- Use SSL for sensitive information such as login pages, etc.
- Use firewall technology to encapsulate your webserver from the Internet and your intranet from your webserver
- Move your data source inside your internal firewall. Allow only access to that data source from the webserver.
- Create a secure copy of your presentation data and configuration files
- Disallow use of Remote Publishing Tools
- Analyze all CGI Applications
- Consider using a Load Balancer or Load Balancing application

References

Litchfield, David. *Hackproofing Oracle Application Server (A Guide to Securing Oracle 9).* Available at http://www.ngssoftware.com

Kossakowski, Klaus-Peter & Allen, Julie. *Securing Public Web Servers*. (CMU/SEI-SIM-011). Pittsburgh, Pa: Software Engineering Institute, Carnegie Mellon University, 2000. Available at http://www.cert.org/security-imporvement/modules/m11.html

Oracle Corporation. *Oracle9i Application Server Release Notes Release 1 (v1.0.2.2.2) for Sun Sparc Solaris*. Part number A95823-01. Available at http://technet.oracle.com or http://metalink.oracle.com

Box, Don, et al. *Simple Object Access Protocol (SOAP) 1.1*. Available at http://www.w3.org/TR/SOAP

Oracle Corporation. *Oracle8i Application Developer's Guide - XML Release 3 (8.1.7)* . Part Number A86030-01. Available at http://technet.oracle.com or http://metalink.oracle.com

Scambray, Joel, McClure, Stuart & Kurtz, George. *Hacking Exposed: Network Security Secrets and Solutions Second Edition*. Berkeley, California:Osbourne/McGraw-Hill, 2001