Oracle Security Alert #22

Dated: 29 November 2001
Updated: 23 September 2002 (see bottom)
Severity: 2

Security Implications of the Default Oracle9*i* Application Server v.1.0.2.2 Configuration

This note explains the security implication of the default SOAP configuration in Oracle9*i* Application Server v.1.0.2.2 (Oracle9*i*AS v.1.0.2.2). SOAP is installed and enabled with the 'typical' (non-custom) Oracle9*i*AS v.1.0.2.2installation.

Summary

SOAP is enabled by default (for the 'typical' install) on the Oracle9*i*AS v.1.0.2.2 release in order to provide convenient use of the SOAP samples. In this default installation, the ability to deploy/undeploy SOAP services is unrestricted and thus may pose a security threat especially in situations where the Oracle9*i*AS v.1.0.2.2 HTTP server might be accessed via the Internet. This note details the nature of the possible threats and also provides information that can be used to counter intrusions which might exploit the default SOAP installation in Oracle9*i*AS v.1.0.2.2.

It is strongly recommended that:

1. Where SOAP is not used, it be disabled OR
2. The deploy/undeploy feature be disabled OR
3. Access to the SOAP deploy/undeploy facility be restricted to administrative or test personnel in the manner detailed below.
   The SOAP default configuration in Oracle9*i*AS v.1.0.2.2 allows one to deploy/undeploy SOAP services by accessing an auto deployed service identified by **urn:soap-service-manager**. In addition, the Oracle SOAP configuration has a built–in Java provider that allows one to deploy Java classes as SOAP services. The default SOAP configuration will allow anyone who has access to the default SOAP URL **/soap/servlet/soaprouter** to deploy Java classes that are already available to the servlet, as SOAP services. If such a behavior is undesirable it is essential that the SOAP configuration be changed to protect access to **urn:soap-service-manager** or disable it. This is even more significant if the default SOAP URL is available from outside the firewall. Please note that adding SOAP services that are implemented as custom Java classes is not possible without write access to the file system.

Platforms affected

All platforms including Windows NT.

Default SOAP Configuration

The SOAP configuration in Oracle9*i*AS v.1.0.2.2. is installed as a servlet running in JServ, as part of the installation process. The default install enables two SOAP services:

1. **urn:soap-service-manager**
2. **urn:soap-provider-manager**

These two services allows one to deploy/undeploy other SOAP services and SOAP providers, respectively. The SOAP configuration in Oracle9*i*AS v.1.0.2.2., which is based on Apache SOAP, comes with a Java Provider, which is always deployed. This provider allows one to deploy Java classes as SOAP services. To deploy a Java class as a SOAP service, there are three restrictions:

1. The Java class that is used as a service class has to be available to the soap servlet (should be part of the JServ classpath or the servlet zone repository).

2. The Java class that is used as a service class has to have a public no args constructor.

3. The method in the service class that can be used by remote clients must have all the arguments deserializable and the return value seriliazable. The SOAP configuration in in Oracle9*i*AS v.1.0.2.2. contains serializers/deserilizers for the following Java types: Java inbuilt types/wrapper classes, JavaBeans, Hashtable, Vector, org.w3c.dom.Element, base64 binary, Parameter, QName, and arrays (of supported types). For any other type, the serializer/deserializers has to be custom written and deployed.

There are security implications of having the **urn:soap-service-manager** deployed. A client having access to the URL that hosts the service **urn:soap-service-manager** can deploy/undeploy other SOAP services. A client having access to the URL that hosts the service **urn:soap-provider-manager** can deploy/undeploy other SOAP providers. It is therefore essential to 'adequately' protect these services, where the meaning of 'adequately' depends on the environment and security requirements. It is important to note that the same SOAP servlet can host multiple services, i.e., an n-to-1 mapping between the SOAP services and the SOAP servlet. This means that Apache access control provides an all-or-nothing protection (as the service is identified in the XML payload - which Apache does not understand).

Consider a SOAP servlet with the sample 'addressbook' (**urn:AddressFetcher**) service deployed along with the default configuration. The default SOAP URL is **/soap/servlet/soaprouter**. Which means to access **urn:AddressFetcher** as well as to access **urn:soap-service-manager** (which deploys/undeploys new services) a client has to use the relative URI **/soap/servlet/soaprouter**. This means that if a client can access **urn:AddressFetcher** it can also access **urn:soap-service-manager**.

With the exception of development and testing situations, such all-or-nothing access is rarely the right choice. In addition to custom solutions such as deploying one servlet per service or having a front end to the SOAP servlet, theSOAP configuration in Oracle9*i*AS v.1.0.2.2 provides the following features to protect **urn:soap-service-manager** and **urn:soap-provider-manager** services, which customers might want to consider when deploying for production:

1. The SOAP in Oracle9*i*AS v.1.0.2.2 allows one to specify the URLs that MUST be used to access **urn:soap-service-manager** and **urn:soap-provider-manager**. If those URLs are not used then the servlet will reject the request. Using Apache, one can configure two (or more) URLs to point to the same servlet. Say 'A' and 'B'. 'A' can have a one level of protection and can be specified in the soap config to be used for the service manager. 'B' can have another level of protection. This means that if a client can access 'A', it can deploy/undeploy services and can also access other services. A client that can access 'B' but not 'A', cannot deploy/undeploy services, but can access other services. To configure the required URI for **urn:soap-service-manager** underneath the **serviceManager** element (in file $ORACLE_HOME/soap/webapps/soap/WEB-INF/config/soapConfig.xml) add the following element:

   <osc:option name="requiredRequestURI" value="relative-url" />

   To configure the required URI for **urn:soap-provider-manager** underneath the **providerManager** element add the following element:

   <osc:option name="requiredRequestURI" value="relative-url" />

   For example, the URL **/soap/servlet/soaprouter** can be used to access custom soap services (such as **urn:AddressFetcher**). Apache and JServ directives can be used to configure URLs /**soap/admin/servicemanager** and **/soap/admin/providermanager** to point to the same servlet (which hosts **urn:AddressFetcher**). Standard Apache and

JServ authentication/authorization facilities can be used to protect **/soap/servlet/soaprouter**, /**soap/admin/servicemanager** and **/soap/admin/providermanager** (such as allowing only local host to access **/soap/admin/servicemanager** and **/soap/admin/providermanager** or requiring SSL with client authentication etc.).

2. The SOAP configuration in Oracle9*i*AS v.1.0.2.2. also has the concept of predeployed services (and providers). What this means is that all the services (and providers) are deployed and no new services (and providers) are to be deployed and deployed services (and providers) are not to be undeployed. This is done by adding the following element underneath the **serviceManager** element in the soap config file ($ORACLE_HOME/soap/werbapps/soap/WEB-INF/config/soapConfig.xml) :

   <osc:option name="autoDeploy" value="false" />

   This undeploys the **urn:soap-service-manager** and **urn:soap-provider-manager** services. The default configuration has the **autoDeploy** attribute set to true. This means that services can be deployed/undeployed. This means that any client having access to **/soap/servlet/soaprouter** can deploy/undeploy service. Such a client can deploy classes (which have no args constructors) available in the JServ classpath (e.g.: classes in packages java.lang.*, java.util.* etc) and the servlet zone repository. To invoke the method in such a deployed class requires that the appropriate serializer/deserializer also be made available as noted above.

Workarounds

If SOAP is not being used it can be disabled by editing the file $ORACLE_HOME/Apache/Jserv/etc/jserv.conf and commenting out the following four lines:

(Please note that unlike in the four lines below, jserv.conf in an Apache install will have $ORACLE_HOME expanded and the ajp port may not be 8200)

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

Update of 13AUG2002: Additional Issue

There is an additional issue with the SOAP default install that has been corrected in future releases. It is possible to read the install scripts due to a configuration error in the default httpd.conf. Note that it is not possible to execute those install scripts, only read them. The result is that it is possible to determine the name of the disk where Oracle is installed. To correct this problem, the configuration should be changed to make the install scripts inaccessible to the OHS server.

Workarounds

The problem is that the install scripts for 1.0.2.2 and 1.0.2.3 reside in the docs folder mapped by the following httpd.conf entry:

Alias /soapdocs/ "F:\Oracle\iSuites/soap/"

This should be changed to the following on 1.0.2.2 and 1.0.2.3 and the docs moved:

Alias /soapdocs/ $ORACLE_HOME/soap/docs/

In the patchset for iAS9.0.2, the corresponding line in httpd.conf should be deleted.