

THE ORACLE XSQL SERVLET

The XSQL Servlet is a smart technology to develop dynamic websites with XML and XSLT.

OVERVIEW

The XSQL Servlet based on the XML Java Framework from Oracle. You can install the XDK with the Oracle IAS or download the whole XDK framework from technet (technet.oracle.com / Java Section).

First the web server must be configured to run servlets.

Then the classpath for the XSQL Servlet must be defined in the webserver configuration files.

If the Jserv cartidge is used:

`$ORACLE_HOME/Apache/Jserv/jserv.properties`

```
...
# Oracle XSQL Servlet
wrapper.classpath=D:\oracle\oraiaas\jdk\lib\oraclexsql.jar
# Oracle JDBC (8.1.7)
wrapper.classpath=D:\oracle\oraiaas\jdbc\lib\classes12.zip
# Oracle XML Parser V2 (with XSLT Engine)
wrapper.classpath=D:\oracle\oraiaas\jdk\lib\xmlparserv2.jar
# Oracle XML SQL Components for Java
wrapper.classpath=D:\oracle\oraiaas\jdk\lib\xsu12.jar
# XSQLConfig.xml File location
wrapper.classpath=D:\oracle\oraiaas\oracore\admin
...
```

In the next step, a applikation handler must be defined if you run the XSQL Servlet with the Jserv cartridge:

`$ORACLE_HOME/apache/jServ/config/Jserv.conf (8i)`

```
..
ApJServAction .xsql /servlets/oracle.xml.xsql.XSQLServlet
...
```

The configuration file of the XSQL Servlet is also written in XML. In this file the connection description to the database is configured.

`$ORACLE_HOME/jdk/lib/XSQLConfig.xml`

```
<connectiondefs>
  <connection name="myapp">
    <username>system</username>
    <password>manager</password>
    <dburl>jdbc:oracle:thin:@localhost:1521:ORCL</dburl>
    <driver>oracle.jdbc.driver.OracleDriver</driver>
  </connection>
</connectiondefs>
```

Now it is possible to call the first XSQL Page

Example for an Oracle XSQL Page:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="emp.xsl"?>
<page connection="demo" xmlns:xsql="urn:oracle-xsql">
<xsql:include-request-params/>
<xsql:query find="% " sort="ENAME" null-indicator="yes" >
    SELECT *
    FROM EMP
    WHERE ENAME LIKE UPPER('%{@find}%')
    ORDER BY {@sort}
</xsql:query>
</page>
```

To get this page, the following URL must be called:

```
Http://mywebserver/mydemo/myApp.xsql?find=SCOTT&sort=1
```

The result of the query is send back to the browser as XML file. (best viewed with the MS IS => 5.0)

To get a HTML output from the generated XML, is is possible to transfrom the XML with a XSL stylesheets

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:import href="../common/rowcol.xsl"/>
<xsl:template match="page">
  <html>
    <head><link rel="stylesheet" type="text/css" href="../common/rowcol.css" />
    </head>
    <body class="page">
      <center><from action="emp.xsql" method="post">Value of 'find' Parameter to
Match ENAME:<input type="text"
      value="{/page/request/parameters/find}" name="find"/>
    </from></center>
    <xsl:apply-templates select="ROWSET"/>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

TEST THE EXISTENTS

1. ARE SERVLETS KONFIGURED?

Test the path to execute a sevlet:

```
http://mytestserver/servlet/IsItWorking
http://mytestserver/servlets/IsItWorking
```

Call the XSQL Class over the browser:

```
http://mytestserver/servlet/oracle.xml.xsql.XSQLServlet
```

```
or
http://mytestserver/servlets/oracle.xml.xsql.XSQLServlet
```

If the Servlet Class exists, the version will shown.

If the version is => 1.0.0.0, than you have found a beautiful backdoor to the system , search in the web about Georgi Guninski and XSQL Servlet. He have a good Example to call programes on the remote system with a stylesheet from an other server.

Solution:

Delete all Classfiles in the default servlet directory like the IsItWorking.class.

Change the servlet URL /servlet and /servlets in the jserv.conf file.

If you use the newer JServ Cartridge with Apache 1.3.19, be carefull to change ALL ocurance of the URL's in this file!

THE VULNERABILITIES

1. PASSWORD IN THE CONFIGFILE

The passwords are not encrypted in the configuration file.

First try to find the XMLConfig.xml file in the \$ORACLE_HOME Tree

Some locations are: xdk/lib, oracore/admin

In the file you find:

```
<connectiondefs>
  <connection name="myapp">
    <username>system</username>
    <password>manager</password>
    <dburl>jdbc:oracle:thin:@localhost:1521:ORCL</dburl>
    <driver>oracle.jdbc.driver.OracleDriver</driver>
  </connection>
</connectiondefs>
```

Solution

You can set the location for this file as a warper classpath property in the jserv.properties or the zone.propertis files.

\$ORACLE_HOME/Apache/Jserv/jserv.properties

```
...
# XSQLConfig.xml File location
wrapper.classpath=D:\oracle\oraias/oracore/admin
...
```

Protect the file with OS stuff (easy to say, as if Servlet runs with a nobody account!)

Maybe it is better, to use the externally authentication feature of the database.

2. CHANGING THE SQL (SQL POISING)

The place holders in the sql statement @USER or @PWD will be replaced with the value of the parameter

Example:

```
<?xml version="1.0" >
<xsql:query connection="myapp" xmlns:xsql="urn:oracle-xsql">
  select id,account,name from user where username='{@USER}' and pwd='{@PWD}'
</xsql:query>
```

The Url is normally called in this way:

```
Http://mywebserver/mydemo/myApp.xsql?USER=jue&PWD=my
```

To poisoning the URL the parameter PWD can know change.

```
Http://mywebserver/mydemo/myApp.xsql?USER=jue&PWD=my' %20union%20select%20id,
account,name%20from%20user%20where%20'A'='A
```

The statement in the XSQL page will be now transformed into:

```
select id,account,name from user where username='jue' and pwd='my' union select id,
account, name from user where 'A'='A'
```

If the SQL Statement is written in one single line it is possible to disable the last parameter with an comment.

Example:

```
ttp://mywebserver/mydemo/myApp.xsql?USER='admin'%20--
```

The statement in the XSQL page will be now transformed into

```
select id,account,name from user where username='admin' -- and pwd=''
```

and it is possible to connect!

Solution (and also in the most cases the better way to code)

Use bind Variables!

It is not possible to poisoning the statement and if the statement is often called the execution becomes quicker.

Example:

```
<?xml version="1.0" >
<xsql:query connection="myapp"
  xmlns:xsql="urn:oracle-xsql" bind-variable="USER PWD">
  select id,account,name from user where username=? and pwd=?
</xsql:query>
```

3. DISABLE THE USE OF THE STYLESHEET

The Parameter *xml-stylesheet=none* in the URL disable the style sheet process.

The is fine to debug, but in a production environment, you should disable this feature.

Solution

Edit the XMLConfig.xml file and set the Value *allow-client-style* to no!

```
<defaults>
  <allow-client-style>no</allow-client-style>
</defaults>
```

4. A LOOK ON THE STATEMENT

Normally the sql Statement is unknown and not visible for the user of the web app.

But if point 3 is possible, the XML output is visible.

So, as the first thing, try to get an error with a bad parameter, like a number like '2n2'

The Statement will be become visible.

Solution:

No Solution : Try to use bind variables and think on point 3.

5. THE DEMO INSTALLATION

Maybe you can find the url:

/xsql/demo/adhocsql/query.xsql?sql=<what ever you want>

If this url can be found, that is like a SQLPLus prompt for the Web.

To escape the ' ' use the + or %20:

```
/xsql/demo/adhocsql/query.xsql?sql=select%20*%20from%20user_objects
```

6. SET YOUR OWN STYLESHEET

You can call a stylesheet over the url with the xml-stylesheet parameter and load the stylesheet from your server if it is not forbidden in the configuration file.

Search in the web about Georgi Guninski and XSQL servlet. He have a good example to call something on the remote system with a stylesheet from an other server.

Solution:

Don't disable the stylesheet security in the configuration File! See Point 3.

7. SET YOUR OWN XSQL FILE

I try to do something in this way, but with no success at the moment. It's look like it is not possible.

8 USE THE COMMAND LINE UTILITY. (THE ORACLEXML SQL UTILITY)

If you have an account on the system you can use the command line utility to get XML data exports from the database.

If you have readable access to the configuration file, you can get the password from there and maybe you don't need more than this.

The class you must call is: oracle.xml.xsql.XSQLCommandLine.

Example:

```
java -cp
/opt/oracle/ora9i/lib/oraclexsql.jar:/opt/oracle/ora9i/lib/xmlparserv2.jar:/opt/oracle/ora9i/xdk/admin:/opt/oracle/ora9i/jdbc/lib/classes12.jar:/opt/oracle/ora9i/rdbms/jlib/xsul2.jar oracle.xml.xsql.XSQLCommandLine bad.xsql
```

Pay attention to the right classpath!

In the classpath must be the path to the configuration file.

=> Install the Oracle XDX local to your home and use only the path to the original configuration file!

In the XML file you call the connection descriptor is used to connect to the right database.

Example bad.xsql

```
<?xml version="1.0"?>
<page connection="demo" xmlns:xsql="urn:oracle-xsql">
<xsql:include-request-params/>
<xsql:query>
      SELECT * from user_objects
</xsql:query>
</page>
```

This tool and all classes you need, can you download from technet.

With Oracle XSQL it is possible to insert or change data. You can built very easy a powerful sql loader utility with this tools.

Solution:

Delete the XML configuration file XMLConfig.xml in a production enviroment.

Don't install the XDK, if you don't need it.

If you must use it, than see point 1.

Use the externally authentication feature.

THE JSERV CARTRIDGE:

THE PROCESS MANAGER

PROBLEM:

Call the url `http://mytestserver/oprocmgr-status`

It is not possible to disable this url, because this url is internally used!

Solution

Allow only the local ip of the webserver to call the url.

Example

```
<IfModule mod_oprocmgr.c>
  ProcNode bhgk 7780
  <Location /oprocmgr-service>
    SetHandler oprocmgr-service
    order deny,allow
    deny from all
    allow from 192.168.10.99
  </Location>
  <Location /oprocmgr-status>
    SetHandler oprocmgr-status
    order deny,allow
    deny from all
    allow from 192.168.10.99
  </Location>
</IfModule>
```

THE DEFAULT URL /SERVLET OR /SERVLETS

Over this url you can call every servlet in the class path of the Jserv cartridge.

Examples:

`/servlet/DMSDump`

`/servlet/Spy`

Solution:

Change the mapping to the url `servlet`.

If it is possible with your application change the URL to maybe `xservlets` or something else.

If you use the Apache > 1.3.19, you must configure the Jserv configuration File and the `xml.conf` file.

In the Jserv config file change ALL occurrence of `/servlet` with the new url!

If you have defined elsewhere an application handler to use a servlet over a file extension, like the XSQL servlet in `xml.conf`, change this!

THE DYNAMIC MONITORING SERVICES

The default URL's:

/dms0
/dms/DMSDump
/servlet/DMSDump
/servlet/Spy
/dms/AgreeSpy

Solution

Disable the URL /sevlet if possible.

Deny the access to the dms0 URL only to localhost.