The Tale of Oracle e-Business Suite 11.5 and the missing security patches
22nd March 2016
(dlitchfield@google.com)

On the 31st December 2015, the Exception to Sustaining Support came to an end [1] for Oracle's eBusiness Suite 11.5.10.2 and reverted to Sustaining Support. This effectively means that there will be no "new updates, fixes, security alerts, data fixes, and critical patch updates" which is a bit of a shame really given I've just reported a large number of security issues to Oracle in this product. As such, anyone that is still using this version of eBusiness Suite will have to take steps to mitigate the risk themselves. To that end, this document will detail the vulnerabilities and what you can do to protect yourself against exploitation. Also included in *Appendix A* are a list of issues the author reported to Oracle in November 2015 and were patched in the January 2016 CPU and *Appendix B*, a list of issues silently patched in the 2016 CPU that readers might still be vulnerable to if they have not installed this patch.

Protecting against exploitation (and bypassing some protection methods)
Protecting against exploitation is particularly troublesome because there may be multiple ways to gain access to a particular vulnerability. Consider the following: bisakrgn.jsp is trivially vulnerable to SQL injection:

https://example.com/OA_HTML/bisakrgn.jsp?pSearchBy=%2530D%27||chr(65)||%27Y%25

Assume we block direct access to bisakrgn.jsp using a Location or mod_rewrite directive an attacker may still be able to get access. In the case of bisakrgn.jsp it has a corresponding function ID in the FND_FORM_FUNCTIONS table so an attacker can still access it by the run function JSP - RF.JSP:

https://example.com/OA_HTML/RF.jsp?function_id=11091&pSearchBy=%2530D%27||chr(65)||%27Y%25

So let's assume we block access to this, again using mod_rewrite, unless it's done correctly we may be able to gain access again using one or more leading 0s in front of the function_id:

https://example.com/OA_HTML/RF.jsp?function_id=00011091&pSearchBy=%2530D%27||chr(65)||%27Y%25

Or by substituting characters for the hex equivalent:

https://example.com/OA_HTML/RF.jsp?fun%63tion_id=00011091&pSearchBy=%2530D%27||chr(65)||%27Y%25

Or by using a POST request instead of a GET.

Then there's an even hackier way of still gaining access. Many extant JSPs execute a JSP forward based on user input. For example, if a JSP contains similar to the following code

[redacted]

It can be used to gain access to bisakrgn.jsp again. The following JSPs perform arbitrary JSP forwards and so can be abused to gain access to "forbidden" content:

https://example.com/OA_HTML/qotSCopAddSvc.jsp?qotFrmMainFile=bisakrgn.jsp&pSearchBy=%25%27||CHR(LENGTH(USER)%2B28)||%27%25

https://example.com/OA_HTML/qotSCopIBSrch.jsp?qotFrmMainFile=bisakrgn.jsp&pSearchBy=%25%27||CHR(LENGTH(USER)%2B28)||%27%25

https://example.com/OA_HTML/qotSCopModSvc.jsp?qotFrmMainFile=bisakrgn.jsp&pSearchBy=%25%27||CHR(LENGTH(USER)%2B28)||%27%25

https://example.com/OA_HTML/qotSCopPOSrch.jsp?qotFrmMainFile=bisakrgn.jsp&pSearchBy=%25%27||CHR(LENGTH(USER)%2B28)||%27%25

https://example.com/OA_HTML/qotSSppSalesSupplement.jsp?qotFrmMainFile=bisakrgn.jsp&pSearchBy=%25%27||CHR(LENGTH(USER)%2B28)||%27%25

https://example.com/OA_HTML/qotSSrpSvdSrch.jsp?qotFrmMainFile=bisakrgn.jsp&pSearchBy=%25%27||CHR(LENGTH(USER)%2B28)||%27%25

https://example.com/OA_HTML/qotSSrpSvdSrchList.jsp?qotFrmMainFile=bisakrgn.jsp&pSearchBy=%25%27||CHR(LENGTH(USER)%2B28)||%27%25

https://example.com/OA_HTML/qotSTppTmplCreate.jsp?qotFrmMainFile=bisakrgn.jsp&pSearchBy=%25%27||CHR(LENGTH(USER)%2B28)||%27%25

https://example.com/OA_HTML/jtfbinperzedit.jsp?event=save&jtfBinId=1&jtfbinperzfavorName=X&jtfbinperzfavorDesc=foo&jtfbinperzfavorId=1&&jtfbinreturnURL=bisakrgn.jsp&pSearchBy=%25%27||CHR(LENGTH(USER)%2B28)||%27%25

The only foolproof method of protecting against these flaws is to delete the offending JSP.

Trusted.conf cannot be trusted

As mentioned earlier, using Location directives may not be 100% effective. Indeed, in the default install, the Apache configuration file trusted.conf contains a number of Location directives that are used to block access to specific URLs. These location directives cannot be trusted. For example, the trusted.conf contains the following directive:

This is supposed to deny access to /dms0 but it can be trivially bypassed by adding an extra forward slash:

https://example.com//dms0

Other examples of ineffective Location directives include
https://example.com/oa_servlets//IsItWorking
https://example.com/oa_servlets//oracle.apps.fnd.oam.jserv.OAMJservSumm?host=example.com&port=8102&proc=http
https://example.com/OA_HTML//bin//sqlnet.log
https://example.com/oa_servlets//oracle.xml.xsql.XSQLServlet
https://example.com/oa_servlets//oracle.xml.xsql.XSQLServlet/OA_HTML/jtfwrepo.xml

You'll need to update these Location directives to protect against abuse.

The Vulnerabilities - currently unpatched
The following JSP pages contain SQL injection vulnerabilities, some of which can be exploited without a username or password and allow a total compromise of the database server.

- CVE-2016-3662
- CVE-2016-3662
- CVE-2016-3662
- CVE-2016-3662
- CVE-2016-3662
- CVE-2016-3663
- CVE-2016-3466

There are 2 more JSPs vulnerable to SQL injection that are not listed here but eBusiness Suite 12.2 and earlier are also vulnerable so these will be disclosed after patches for those versions are made available by Oracle.

As already indicated, there may be multiple ways to access these files, either through RF.jsp or by using an ad-hoc JSP forward. The only sure fire way of protecting against exploitation is to delete them. However, this may not be possible if the JSPs are actually used. A quick grep of the access logs will help determine this. If any of these JSPs are used then you should patch the JSPs in question.

The SQL injection vulnerabilities in [illegible] and [illegible] all lie in the IEXReconciliationReport Java class underlying the JSP:

```
[illegible]  [illegible]  =
      [illegible]
                  [illegible]  ,
                  [illegible]  ,
                  [illegible]  ,
                  [illegible]  ,
                  [illegible]  ,
                  [illegible]  ,
                  [illegible]  ,
                  [illegible]  ,
                  [illegible]  ,
                  [illegible]  ,
                  "  ,
                  [illegible]  ,
                  [illegible]  ,
                  [illegible]  ,
                  [illegible]  ,
                  [illegible]  ,
                  [illegible]  ,
                  [illegible]
                  [illegible]
                  [illegible]
                  [illegible]  ;
```

To prevent the SQL injection flaws, before the [illegible] method is executed, for each string parameter, use [illegible] to remove every occurrence of a single quote.

[illegible] it takes a “where” parameter and, if present, is embedded in a query then executed:

```
[illegible]
[illegible]
[illegible]
}
...
[illegible]
[illegible]
.
[illegible]
```

As is, this effectively allows an attacker to embed their own where clause and from there they can execute entirely arbitrary SQL and gain full control of the database server. There is no safe way to do this and it is strongly recommended that this section of code be commented out:

[redacted]
[redacted]
[redacted]
}

In Summary
There are often multiple ways to gain access to a given JSP or URL. If the JSP is not being used, by far the safest option is to delete the vulnerable file if that is possible. If it is not then you need to patch the JSP yourself as Oracle will no longer provide patches.

Appendix A
In November 2015 I reported over 40 security flaws to Oracle in eBusiness Suite 11.5.10.2 and these were patched in the January 2016 Critical Patch Update [2]. The full list can be found in [3] and I discuss exploitation of some of the more interesting issues in [4].

CVE-2016-0510 SQL INJECTION IN APPS.BIS_BUSINESS_VIEWS_CATALOG
CVE-2016-0511 SQL INJECTION IN BIS_LOV_PUB ANDBIS_PORTLET_PMREGION
CVE-2016-0512 SQL INJECTION IN HR_MISC_WEB
CVE-2016-0514 SQL INJECTION IN JTF_BISFAVORITEPLUG_PUB
CVE-2016-0515 SQL INJECTION IN JTF_BISUTILITY_PUB
CVE-2016-0516 SQL INJECTION IN QA_SS_CORE
CVE-2016-0517 SQL INJECTION IN HR_UTIL_DISP_WEB
CVE-2016-0518 SQL INJECTION IN HRHTML
CVE-2016-0589 SQL INJECTION IN ORACLESSWA
CVE-2016-0578 SQL INJECTION VIA JTF_BISUTILITY_PUB.LOV_VALUES
CVE-2016-0581 SQL INJECTION AND XSS IN AME_UI
CVE-2016-0576 MULTIPLE SQL INJECTION AND XSS INICX_UTIL.LOVVALUES
CVE-2016-0520 XSS IN ICX_ASK_ORACLE
CVE-2016-0519 XSS IN ARW_TOOLBAR
CVE-2016-0521 XSS VULNERABILITIES IN POR_REDIRECT
CVE-2016-0584 XSS IN JTF_BISCHARTPLUG_PUB
CVE-2016-0582 XSS IN JTF_BISRELATED_PVT
CVE-2016-0583 XSS IN JTF_BIS_CHART_PLUG
CVE-2016-0588 XSS IN GL_WEB_PLSQL_CARTRIDGE
CVE-2016-0513 XSS IN ORACLEPLUGS.PLUGRENAME
CVE-2016-0507 XSS IN ARW_UTILITIES
CVE-2016-0509 XSS IN AP_WEB_UTILITIES_PKG
CVE-2016-0575 MULTIPLE XSS IN OT_UTIL_SKILLS_WEB
CVE-2016-0579 MULTIPLE XSS IN JTF_BISJAVASCRIPT_PUB
CVE-2016-0586 MULTIPLE XSS IN ICX_ADMIN_SIG
CVE-2016-0544 SQL INJECTION IN AMSSEGMENTLOV.JSP

CVE-2016-0543 SQL INJECTION IN AMSQUERYPREVIEW.JSP
CVE-2016-0548 SQL INJECTION IN BISAKRGN.JSP
CVE-2016-0549 SQL INJECTION IN BISAKRIU.JSP
CVE-2016-0547 SQL INJECTION IN BISAKRGI.JSP
CVE-2016-0552 SQL INJECTION IN BICRLUPD.JSP (Affects EBS 12x, too)
CVE-2016-0545 SQL INJECTION IN BICCFGD2.JSP (Affects EBS 12x, too)
CVE-2016-0550 SQL INJECTION IN JTFWTOST.JSP (Affects EBS 12x, too)
CVE-2016-0580 DOS IN ADI_BINARY_FILE
CVE-2016-0585 DOS IN ICX_ADMIN_SIG

Appendix B

In the January 2016 Critical Patch update Oracle silently patched a number of SQL injection flaws, Cross-Site-Scripting, Open Redirects and directory traversal issues. A full accounting of these security flaws can be found in Appendix A.

amsLOVDelivPreViewPopUp.jsp - SQLi

```
String fileId = request.getParameter("fileId");
String content = l_objeb.readFile(conn,fileId);
```

bicstud1.jsp - SQLi

```
String    org_id    = request.getParameter("org_id");
String    sql  = "UPDATE bic_measures_all set description=?, weight=?, last_update_date=SYSDATE,
last_updated_by=FND_GLOBAL.login_id, last_update_login=FND_GLOBAL.user_id where measure_id = ?
and org_id="+org_id;
PreparedStatement pstmt = _connection.prepareStatement(sql);
```

bicstusw.jsp - SQLi

```
String    org_id    = request.getParameter("org_id");
String    sql  = "UPDATE bic_measures_all set description=?, weight=?, last_update_date=SYSDATE,
last_updated_by=FND_GLOBAL.login_id, last_update_login=FND_GLOBAL.user_id where measure_code
= ? and org_id="+org_id;
PreparedStatement pstmt = _connection.prepareStatement(sql);
```

bisdefsc.jsp - SQLi

```
String schedule = request.getParameter("schedule");
...
...
scheduleDesc = getScheduleDesc(conn, schedule);

public String getScheduleDesc(Connection conn, String schedule) throws SQLException
{

StringBuffer sqlBuf = new StringBuffer(100);
sqlBuf.append("select description from fnd_conc_release_classes_vl where release_class_name = '");
sqlBuf.append(schedule);
sqlBuf.append("'");
String sql = sqlBuf.toString();
```

```
...
...
stmt = conn.prepareStatement(sql);
OracleStatement ostmt = (OracleStatement) stmt;
ostmt.defineColumnType(1, java.sql.Types.VARCHAR, 80);

rs   = stmt.executeQuery(sql);
```

bispgraph.jsp - File access vuln

bisschcf.jsp - SQLi

```
  String pStartDate = request.getParameter("startDate");
  pStartDate = (pStartDate == null) ? "" : pStartDate;
  String pStartHour = request.getParameter("startHour");
  pStartHour = (pStartHour == null) ? "" : pStartHour;
  String pStartMinute = request.getParameter("startMinute");

... etc ...


  String startTime = getTime(pStartDate, pStartHour, pStartMinute, pStartAmPm);
  String endTime = getTime(pEndDate, pEndHour, pEndMinute, pEndAmPm);

...
...

public String getTime(String pDate, String pHour, String pMinute, String pAmPm)
{
  String time = null;
  String hour = null;

  //Bug Fix 2503143, 2584641
  String icxDateFormat = webAppsContext.getNLSDateFormat();
  String date = "";
  if (icxDateFormat != null && !"".equals(icxDateFormat))
    date = getConcurrentDate(conn, pDate, icxDateFormat);

...
...


public String getConcurrentDate(Connection conn, String date, String icxDateFormat)
{
  String concurrentDate = "";
  StringBuffer sqlBuf = new StringBuffer(100);
  // Fix for bug 3363676 - gscc format mask
  sqlBuf.append("SELECT to_char(to_") ;
  sqlBuf.append("date('");
```

```
  sqlBuf.append(date);
  sqlBuf.append("','");
  sqlBuf.append(icxDateFormat);
  sqlBuf.append("'),'DD-MON-RR') FROM dual ");
  String sql = sqlBuf.toString();
  PreparedStatement pstmt = null;
  ResultSet rs = null;

  try
  {
    pstmt = conn.prepareStatement(sql);
    OracleStatement ostmt = (OracleStatement) pstmt;
    ostmt.defineColumnType(1, java.sql.Types.VARCHAR, 15);
    rs = pstmt.executeQuery();
...
...
```

bissched.jsp - XSS

bixappdep.jsp - XSS

```
String appCode = request.getParameter("appCode");
<label>   <input type=hidden name=appCode value="<%=appCode%>"> </label>
...
...
<td headers="c1" height="12" class="errorMessage" colspan="2"
nowrap><%=request.getParameter("errMsg")%></td>
```

bixappdepupdate.jsp - SQLi

```
String apps_installed[] = request.getParameterValues("add");
...
apps_list = apps_list + "'" + apps_installed[i] + "'" + ",";
...
String query1 = " UPDATE bix_dm_apps_dependency "+
              " SET application_installed = 'Y'," +
              "    last_update_date = SYSDATE,  " +
              " last_updated_by =  ?  WHERE application_short_name IN ";
       query1 = query1 + "(" + apps_list + ") ";
...
query2 = query2 + "(" + apps_list + ") ";
...
PreparedStatement stmt = conn.prepareStatement(query1);
stmt.setString(1,userID);
stmt.executeUpdate();
...
stmt = conn.prepareStatement(query2);
stmt.setString(1,userID);
stmt.executeUpdate();
```

bscpgraph.jsp - Arbitrary file access

csifLOVScripts.jsp - XSS

dddAM3DViewer.jsp - XSS

dddCallChMgmt.jsp - XSS

iebTitleAppName.jsp - SQLI

```
ResultSet _rset = _stmt.executeQuery ("select application_name from fnd_application_tl " +
                                "where application_id=" + _APP_ID +
                                " and language='" + _LANG + "'");
if(_rset.next())
```

lemsa_docpreview.jsp - XSS

lemsa_docshowimage.jsp - SQLi

```
// what is cid?  It is the FileID field in FND_LOBS
kb.downloadFile(request.getParameter("cid"), response);
```

ieuiMeetingErrorDisplay.jsp - XSS

```
String status = request.getParameter("Status");
String errCode = request.getParameter("ErrCode");
...
<td class="errorMessage"><img src="../OA_MEDIA/erroricon_pagetitle.gif" width="32"  height="32"
border="0"> <%=status%> </td>
<td colspan="2" class="errorMessage"> <%=errCode%> : <%=messageText%> </td>
```

ieuTitleAppName.jsp - SQLi

```
ResultSet _rset = _stmt.executeQuery ("select application_name from fnd_application_tl " +
                                "where application_id=" + _APP_ID +
                                " and language='" + _LANG + "'");
```

lexagfilter1.jsp - SQLi

```
sEntity = (request.getParameter("entity") == null)? "" : request.getParameter("entity");
sColumn = (request.getParameter("column") == null)? "" : request.getParameter("column");
...
...
String str1 = " select count(' ";
String str2 = " ') from ";
//clchang 10/15/2003 updated for jdk14 bug 3169260
//PreparedStatement stmt2 = connVal.prepareStatement ();
PreparedStatement stmt2 = connVal.prepareStatement (
     str1 + sColumn + str2 + sEntity );
```

```
    // "select count('" + sColumn + "') from " + sEntity );
try {
   ResultSet rset2 = stmt2.executeQuery ();
```

lexstdetupd.jsp - SQLi

```
String sStartTime = (request.getParameter("sStartTime") == null)? "" : request.getParameter("sStartTime");
...
sql.append( "select trunc(to_date('" + sEndTime + "', '" + sDateFormat + "')) - trunc(sysdate) from dual " );
...
qAllStmt = connVal.prepareStatement(sql.toString());
ResultSet rset = qAllStmt.executeQuery();

...

sql.append( "select trunc(to_date('" + sStartTime + "', '" + sDateFormat + "')) - trunc(to_date('" +
   sEndTime + "', '" + sDateFormat + "')) from dual " );

etc
```

jtfalout.jsp - XSS

jtfbAssignResps.jsp - XSS

jtfwemlt.jsp - XSS

```
String caller = request.getParameter("caller");
<SimpleHref target="<%=caller%>.jsp?key=<%=key%>" type="accept" label="Ok">Ok</SimpleHref>
```

jtfwmesg.jsp - XSS

```
String msg = request.getParameter("msg");
<%=msg%>
```

jtfwtpol.jsp - SQLi

```
String keyStr = request.getParameter("key");

String query =
   "SELECT party_id, party_name " +
   "FROM hz_parties " +
   "WHERE party_id IN (SELECT object_id FROM hz_party_relationships WHERE subject_id = " + keyStr +
                  " AND (PARTY_RELATIONSHIP_TYPE = 'EMPLOYEE_OF'" +
                  " OR PARTY_RELATIONSHIP_TYPE = 'CONTACT_OF')" +
                  ")";

try {
   stmt = connection.createStatement();
   rs = stmt.executeQuery(query);
```

oksAutoRenewalHelp.jsp - Open redirect / XSS

```
String thanks = request.getParameter("thanks");
if(thanks != null){
  response.sendRedirect(thanks);
```

titleAppName.jsp - SQLI

```
ResultSet _rset = _stmt.executeQuery ("select application_name from fnd_application_tl " +
                                      "where application_id=" + _APP_ID +
                                      " and language='" + _LANG + "'");
```


[1] https://blogs.oracle.com/stevenChan/entry/ebs11i_sustaining_support
[2] http://www.oracle.com/technetwork/topics/security/cpujan2016-2367955.html
[3] http://www.davidlitchfield.com/DetailsforOraclesJanuary2016CPU.pdf
[4] http://www.davidlitchfield.com/AssessingOraclee-BusinessSuite11i.pdf