Why So Serial?

Mateusz Niezabitowski - Ocado Technology

Object deserialization vulnerabilities vs Java

Mateusz Niezabitowski - Ocado Technology

$ whoami

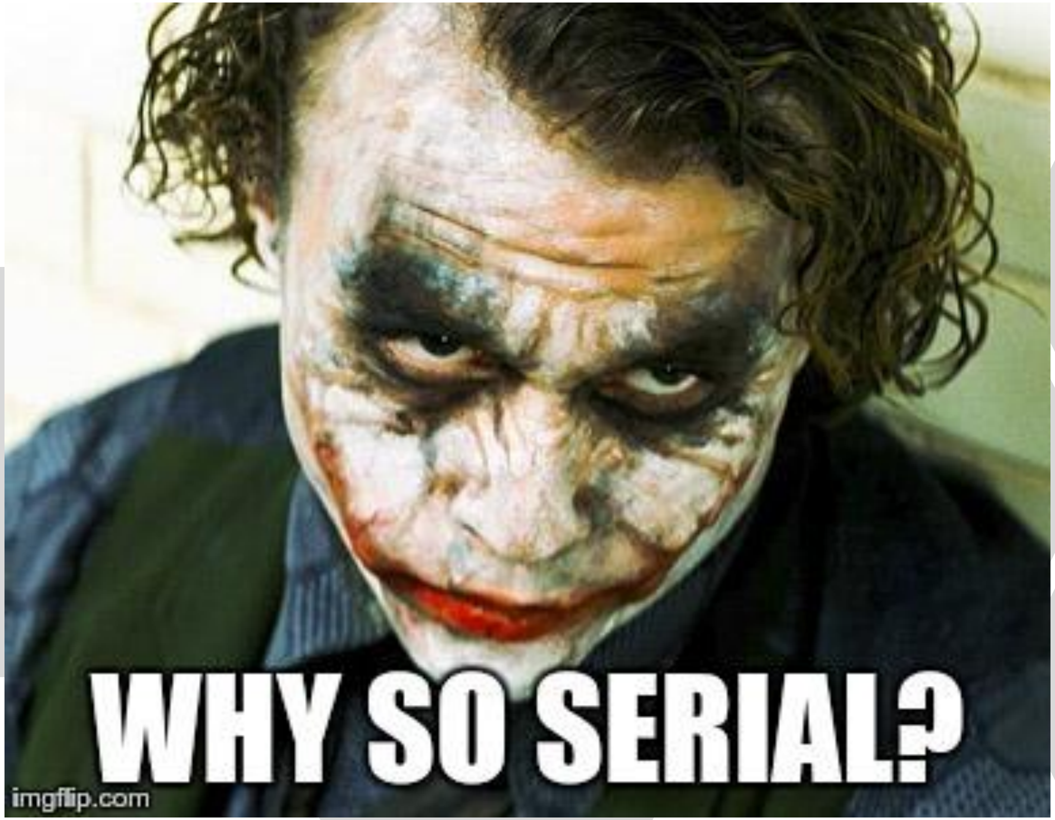
- 10 years as Software Developer (mostly Java & Web)
- Started moving to Application Security 4 years ago
- Application Security Engineer @ Ocado Technology
    - Mostly web technologies, a lot of Java

OWASP Top 10*?
CWE/SANS Top 25?

Source: http://vignette2.wikia.nocookie.net/uncyclopedia/images/1/1d/Absolutely_nothing.jpg/revision/latest?cb=20050420174213

OWASP Top 10*?
CWE/SANS Top 25?

*A8: 2017 - Community



Source: https://i.imgflip.com/1hjcgi.jpg

RCE = Remote Code Execution (a.k.a. Holy Grail)

You **OWN** the machine!

(think: SSH session)

Remember good ol' C?

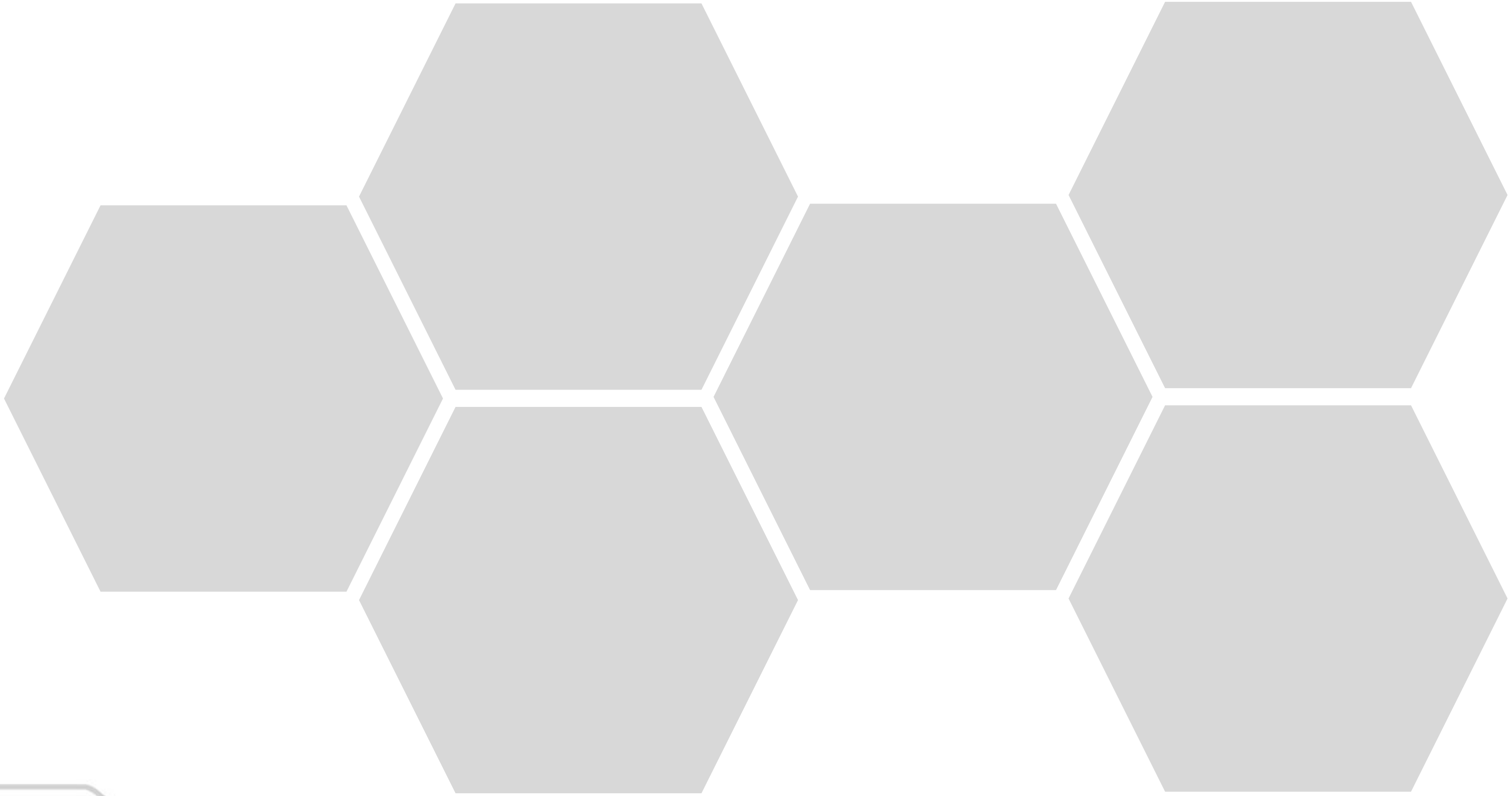
```
char   book_title[50];
char   book_author[50];
char   book_subject[100];
int    book_id;
```

Remember good ol’ C?

```c
char   book_title[50];
char   book_author[50];
char   book_subject[100];
int    book_id;
```

```c
struct Books {
   char   title[50];
   char   author[50];
   char   subject[100];
   int    book_id;
} book;
```

We’ve encapsulated state. What about behavior?



We’ve encapsulated state. What about behavior?

Source: https://cms-assets.tutsplus.com/uploads/users/34/posts/19916/preview_image/wordpress-oop.jpg

That’s awesome! But beware…

If untrusted party is sending you **Object**, they may define (or influence)

its behavior…

That’s awesome! But beware…

If untrusted party is sending you **Object**, they may define (or influence)

its behavior…

Which means: they **might** be able to execute arbitrary code!

But what is (de)serialization?

Simple: transforming in-memory object’s representation to the stream of bytes (and vice-versa) - e.g. to store on a hard drive, or send via network.

What do we need for exploitation?

Step 1:

Programming language must support (de)serialization (duh)

What do we need for exploitation?

Step 2:

Deserialization must be done in a *“dangerous”* way

What do we need for exploitation?

Step 3:

Some methods are being called during (or right after) deserialization

What do we need for exploitation?

Step 4:

There are some “*interesting*” classes “*available*”

What do we need for exploitation?

Step 5:

Target application must deserialize user-controlled objects

What do we need then?

What do we need then?

Source: http://vignette1.wikia.nocookie.net/inspectorgadget/images/b/be/Inspector_Gadget_Thinking.png/revision/latest?cb=20140311001122

So does Java contains some useful gadgets?

Sadly, by itself, it does not :-(

Great! I mean wait a minute… by itself?

Source: https://media.giphy.com/media/26xBBqz39mhKKASAM/giphy.gif

Enter: Apache’s `commons-collections` library

Enter: Apache's `commons-collections` library

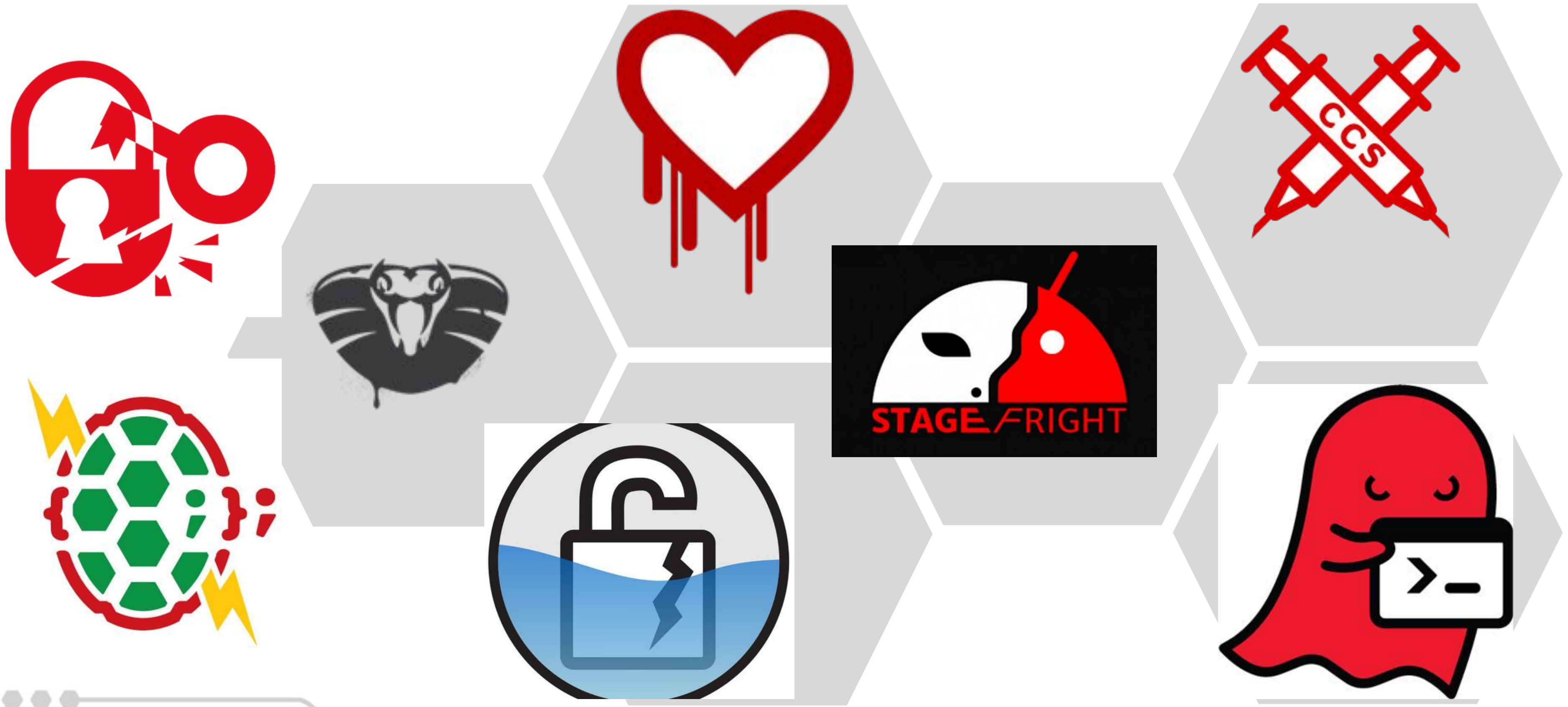
But how common is `common`?

Enter: Apache’s `commons-collections` library

But how common is `common`?

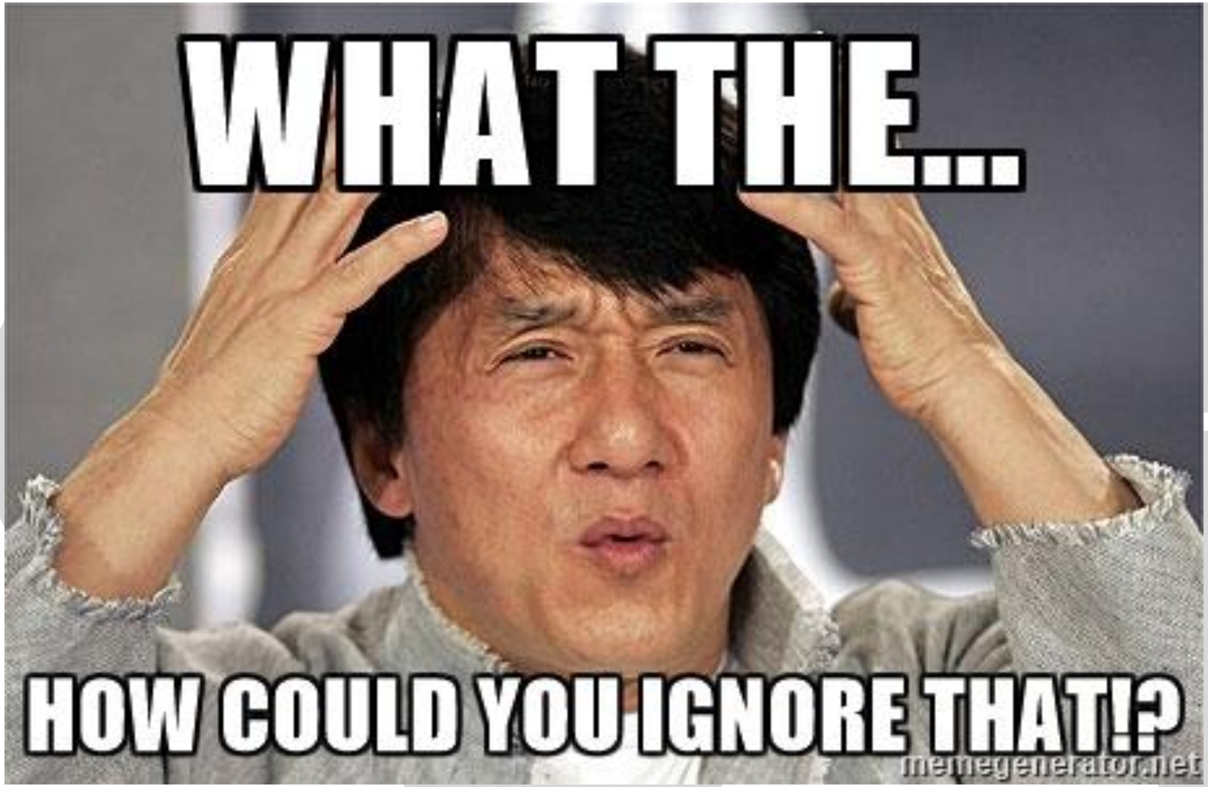
Turns out: pretty common…

In a world when every vulnerability needs logo…



“The most underrated, underhyped vulnerability of 2015”

“The most underrated, underhyped vulnerability of 2015”



Source: https://cdn.meme.am/cache/instances/folder397/500x/74666397.jpg

Fortunately, 9 months (!) later…

IBM WebSphere

openNMS®

ORACLE WebLogic

Jenkins

JBoss™

PayPal

Source: https://cdn.meme.am/instances/61927462.jpg

Let’s talk about the fix...

Source: http://www.dark-circuit.com/random/data/media/1/i%20will%20fix%20it%202.jpg

Approach #1: don’t use serialization!

Approach #1: don’t use native serialization!

But what if half of your code depends on it?

Approach #2 (proposed by foxglove)

Step 1:

```
grep -Rl InvokerTransformer .
```

(Yeah, really…)

Approach #2 (proposed by foxglove)

Step 2:

Delete all occurrences of `commons-collections`

OR

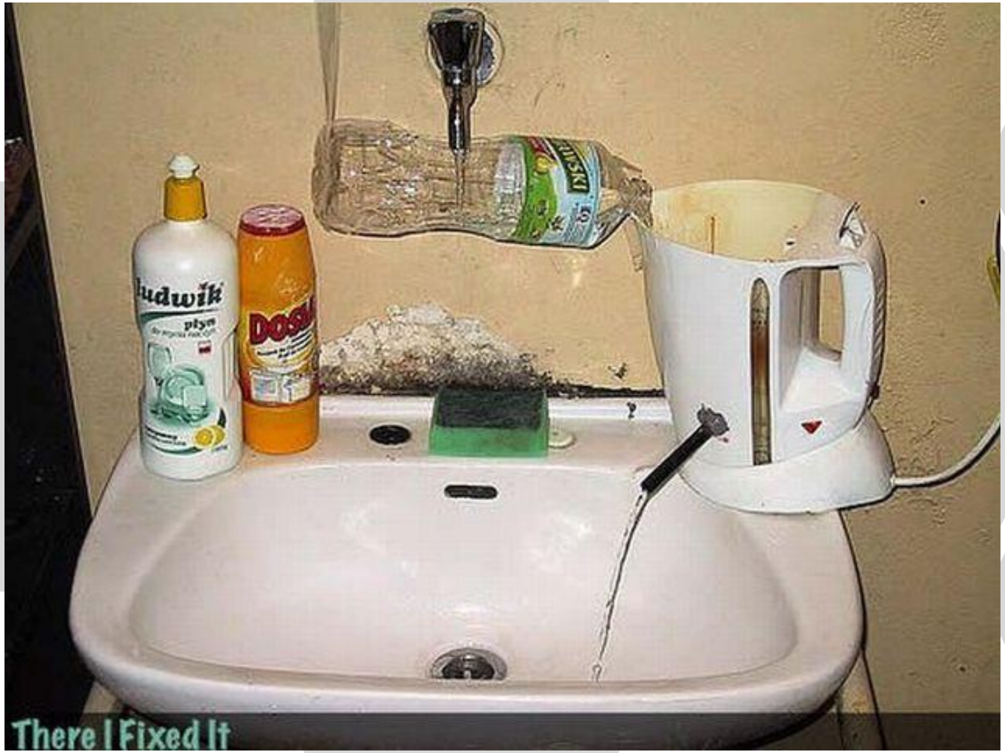
Delete `InvokerTransformer.class` from all jars

Approach #2 (proposed by foxglove)

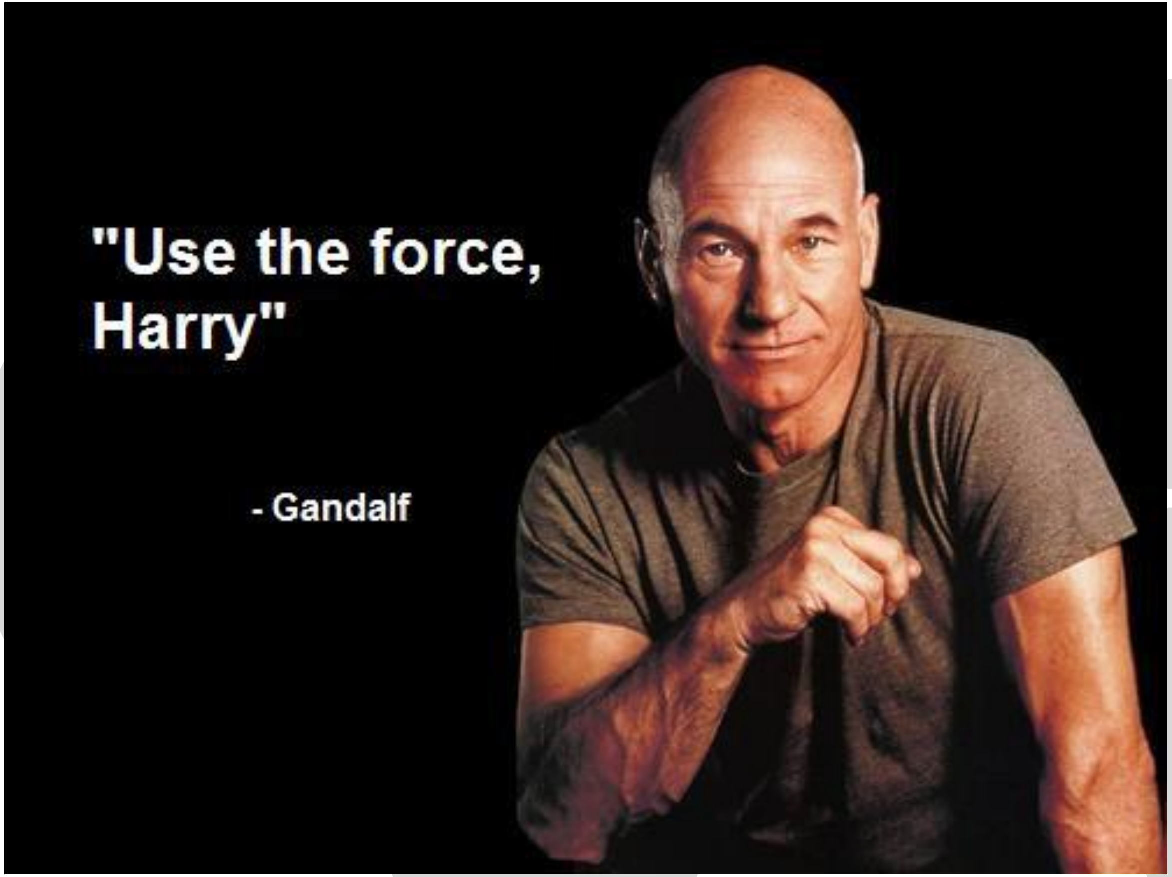
Step 3:

Profi^H^H^H^H^HWait, WTF?

“Fix”



Source: https://www.appliancesonline.com.au/academy/wp-content/uploads/2012/11/Sink.jpg



Source: https://absurdlynerdly.files.wordpress.com/2011/10/offensive.jpg?w=300&h=225

So let’s say you don’t rely on native serialization

We are good, right?

Solved! Hold on… native?

Source: https://media.giphy.com/media/26gsuXyfQKiy315ao/giphy.gif

Serialization is inevitable!

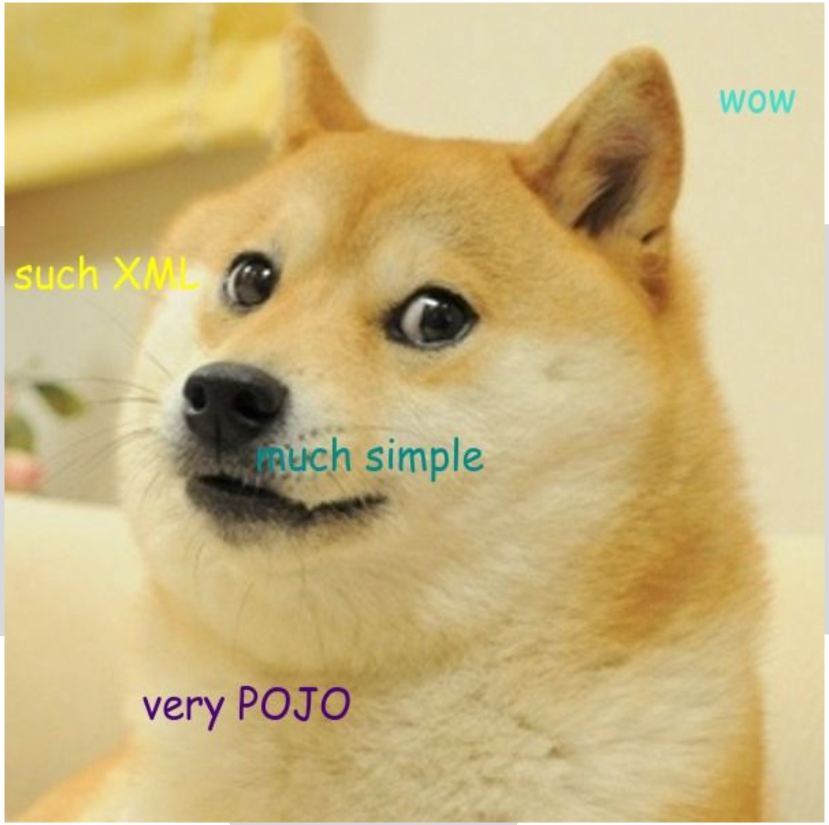
JSON

XML

PROTOBUF

… (a lot more)

Let’s talk about XStream



XStream is so simple and powerful!

```java
import java.util.Date;
import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.io.xml.DomDriver;

public class XStreamTest {

    private String name = "hey ma, look, I'm string! ";

    private int age = 6;

    private Date birthDate = new Date();

    public static void main(String[] args) {
        System.out.println(new XStream(new DomDriver()).toXML(new XStreamTest()));
    }

}
```

XStream is so simple and powerful!

```
<XStreamTest>
  <name>hey ma, look, I&apos;m string! </name>
  <age>6</age>
  <birthDate>2016-04-26 16:54:50.773 UTC </birthDate>
</XStreamTest>
```

Source: https://cdn.meme.am/cache/instances/folder665/500x/68008665.jpg

No RCE? Still no good - auth bypass/EoP

```
{
    “username”: “testuser123”,
    “age”: 23,
    “acceptedCookie”: true,
    “role”: “user”
}
```

No RCE? Still no good - auth bypass/EoP

```
{
    “username”: “testuser123”,
    “age”: 23,
    “acceptedCookie”: true,
    “role”: “admin”
}
```

No RCE? Still no good - DoS

```
static byte[] payload() throws IOException {
   Set root = new HashSet();
   Set s1 = root;
   Set s2 = new HashSet();
   for (int i = 0; i < 100; i++) {
     Set t1 = new HashSet();
     Set t2 = new HashSet();
     t1.add("foo"); // make it not equal to t2
     s1.add(t1);
     s1.add(t2);
     s2.add(t1);
     s2.add(t2);
     s1 = t1;
     s2 = t2;
   }
   return serialize(root);
 }
```

No RCE? Still no good - Pretty much everything*

SQLi, File/Directory removal, XSS…

No RCE? Still no good - Pretty much everything*

*It all depends on available gadgets!

Seriously though - we need a fix!

Source: http://funnydumpster.com/wp-content/uploads/2010/10/there-i-fix-it-15.jpg

Approach #3: Blacklisting/Whitelisting!

Idea: deserialize only “safe” classes

Approach #3: Blacklisting/Whitelisting!

Problem: blacklisting doesn’t work :-(

Let’s play Gadget whack-a-mole!

Source: https://usatftw.files.wordpress.com/2014/06/cryingkid1.gif?w=1000

Approach #3: Blacklisting/Whitelisting!

Problem: whitelisting could also fail!

Also, this is really painful from developer's point of view

Approach #3: Blacklisting/Whitelisting!

How could you even do that? Java doesn’t support black/white lists…

Approach #3: Blacklisting/Whitelisting!

Solution 1: Wrap/Subclass `ObjectInputStream`

Example: **SerialKiller**

Approach #3: Blacklisting/Whitelisting!

Solution 2: Modify JVM (Java Agent)

Example: **NotSoSerial**

Approach #4: Use safe library

But really, really safe!

(I’m reluctant to name any “safe” library here, but hmm maybe J*****n)

Approach #5: Crypto for the rescue - signing!

Basic idea:

Every serialized object is cryptographically signed

(MACed)

Approach #5: Crypto for the rescue - signing!

Result:

User **can’t modify** and send object back to server

(actually - he can, but server will know that object has been tampered with)

Approach #5: Crypto for the rescue - signing!

Result:

Server only sees objects serialized by itself

(should be safe)

Approach #5: Crypto for the rescue - signing!

Looks awesome… But there are still problems :-(

Approach #5: Crypto for the rescue - signing!

Can’t be applied to all serialization problems

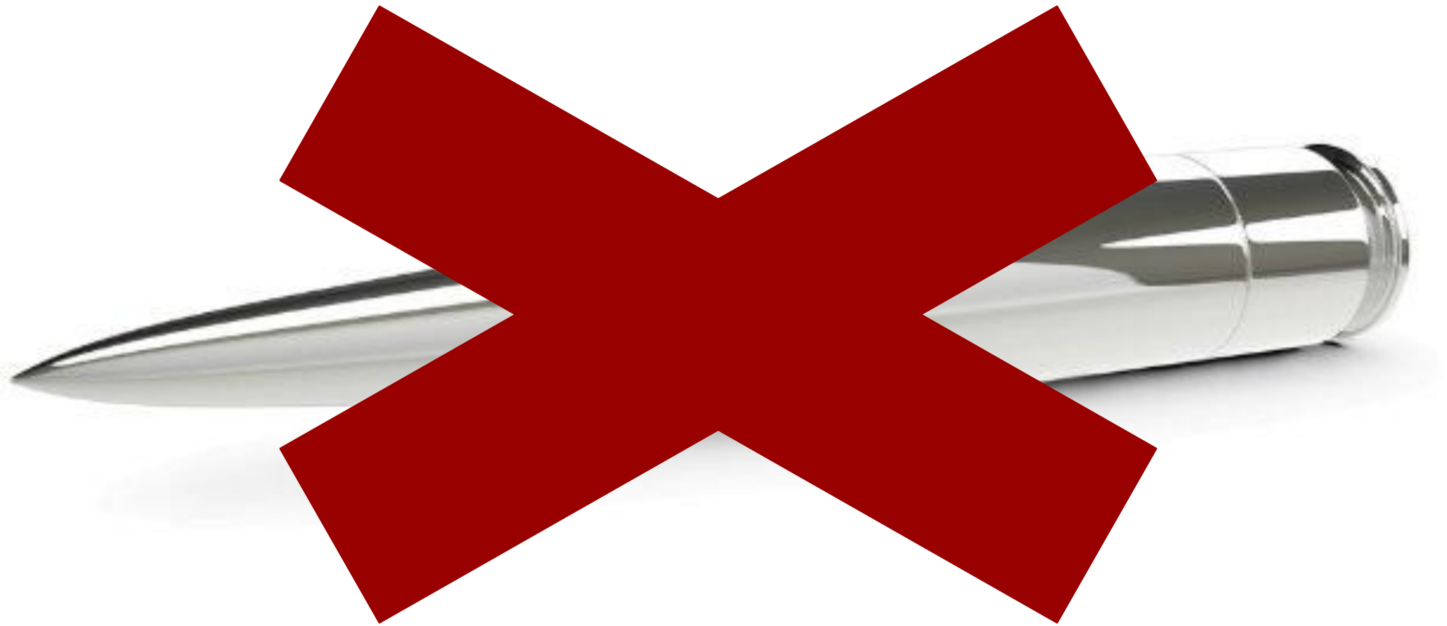
Approach #5: Crypto for the rescue - signing!

Implementation of this mechanism in existing application might be tricky

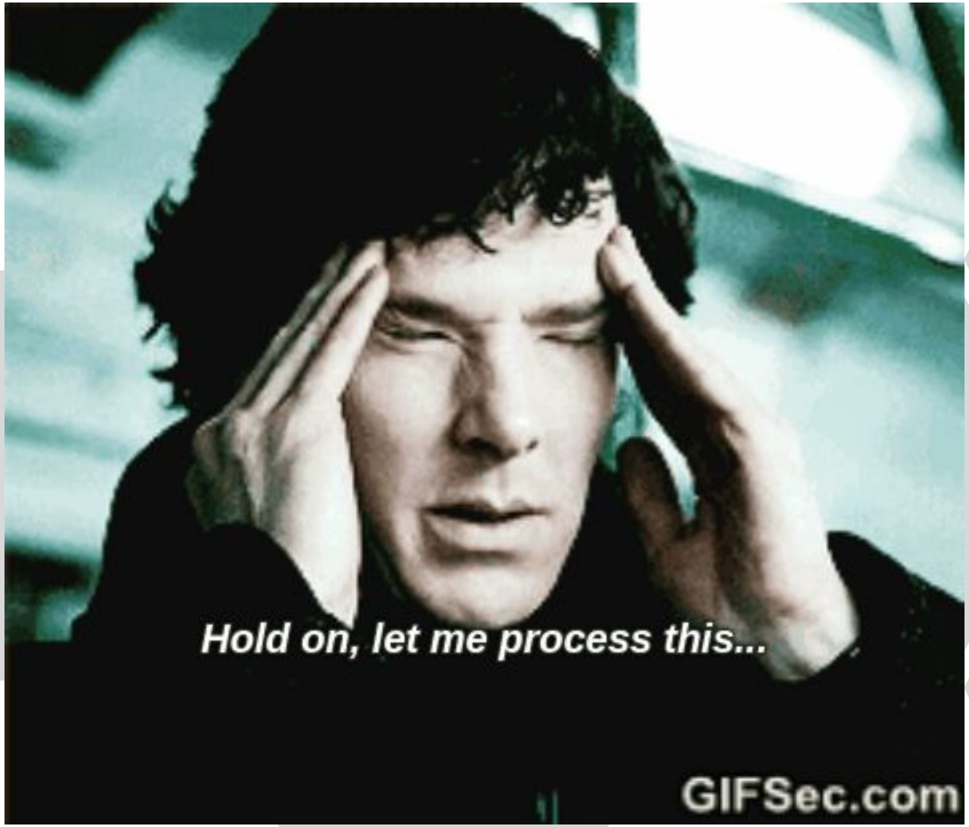
Approach #5: Crypto for the rescue - signing!

Also - Crypto is Hard™

Source: http://exchangeleads.io/wp-content/uploads/2015/09/Silver-Bullet.jpg



Source: http://exchangeleads.io/wp-content/uploads/2015/09/Silver-Bullet.jpg



Source: https://media.giphy.com/media/l3q2A4LfFi4Crt4Vq/giphy.gif

Serialization problems are everywhere

Also in Java. They are language agnostic. They are format agnostic.

Serialization problems usually are very dangerous

Very often they lead to RCE, but other attacks are possible (depending on gadgets)

Serialization problems are hard to fix

Don’t blacklist, don’t play gadget whack-a-mole. Think

before applying fix - there’s no silver bullet.

Links

General

- [EN] OWASP on deserialization of untrusted data
- [EN] Original presentation by @frohoff & @gebl about serialization problems

Java

- [PL] Deserialization vulnerabilities in Java explained, part 2, part 3 (my :-))
- [EN] Article about deserialization problems in Java which raised awareness (with bugs found in Jenkins, JBoss, WebLogic, WebSphere, OpenNMS...)
- [EN] ysoserial tool by @frohoff & @gebl
- [EN] Recent presentation on deserialization in Java from @frohoff - finder of commons-collections gadget chain
- [EN] Matthias Kaiser - Exploiting Deserialization Vulnerabilities in Java talk (with great explanation of commons-collections gadget chain)
- [EN] Deserialization vulnerability in PayPal
- [EN] Article on java deserialization vulnerabilities by Contrast Security
- [EN] Explanation of commons-collection gadget chain
- [EN] Recent gadget chain targeting OpenJDK, using nothing but JRE (by Matthias Kaiser)!
- [EN] Summary of deserialization problems with proposed solutions, and why most of them (don't) work
- [EN] Old deserialization problems in XStream library
- [EN] Recently found deserialization problems with Kryo library
- [EN] Recently found deserialization problems with XStream library
- [EN] April's Fool - remove java serialization
- [EN] Recent deserialization vulnerability - again in Jenkins (CVE-2017-1000353)
- [EN] Even more recent deserialization issue in Struts2, part 2 (CVE-2017-9805)
- [EN] Comprehensive whitepaper describing status quo of deserialization vulnerabilities (gadget chains, libraries) in 2017

Defense

- [EN] Why blacklisting doesn't work
- [EN] Why blacklisting doesn't work - again
- [EN] SerialKiller - wrapper of ObjectInputStream with black/whitelisting
- [EN] NotSoSerial - Java Agent with serialization black/whitelists

Questions?

m.niezabitowski@ocado.com | appsec@ocado.com

Source: http://az616578.vo.msecnd.net/files/2016/08/08/6360622035014053461005076937_joker.png