JAVA DESERIALIZATION VULNERABILITIES

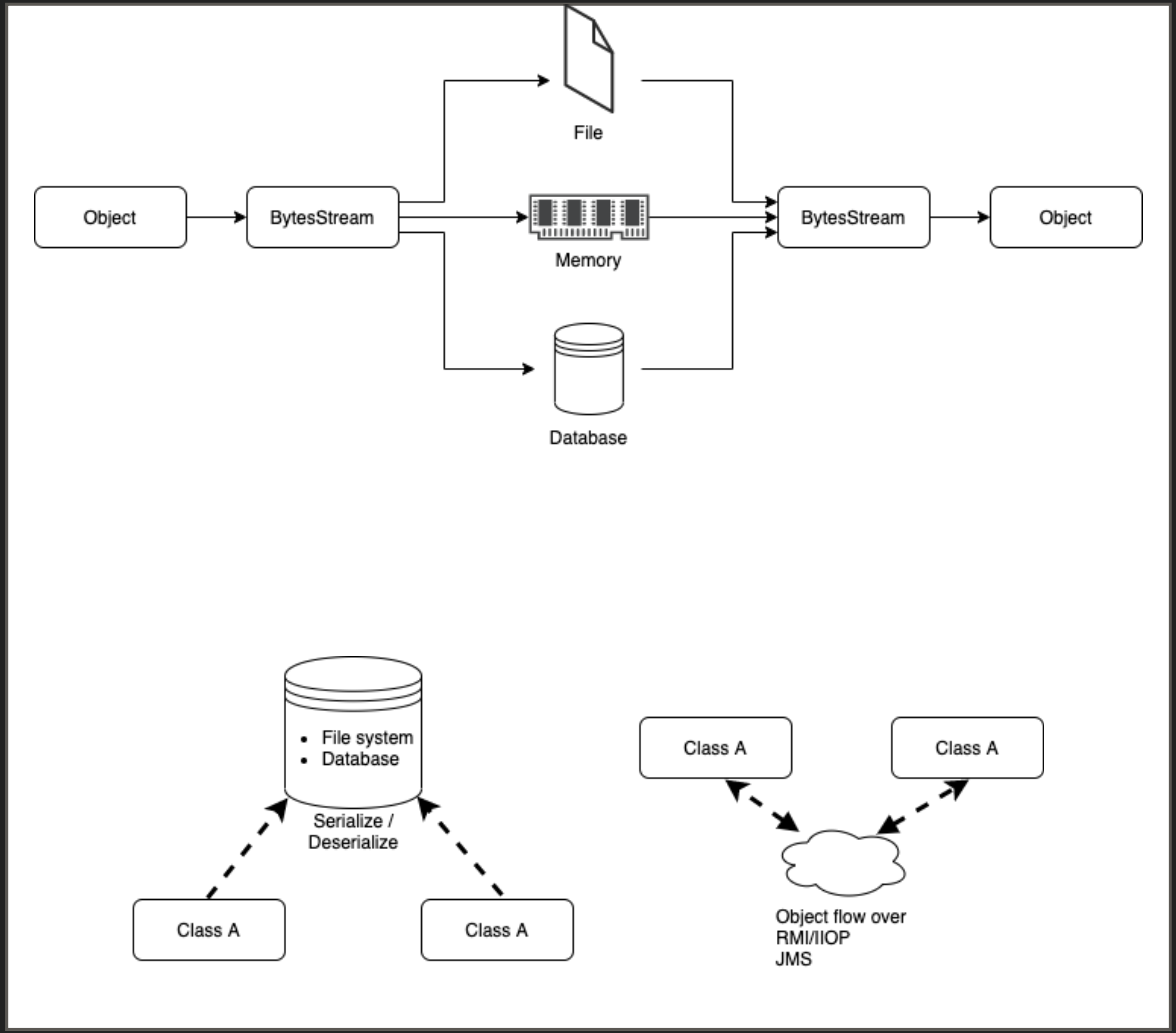
EXPLOITATION TECHNIQUES

TIMELINE

- 2006 - Pentesting J2EE - Marc Schönrnfeld
- 2010 - Beware of Serialized GUI Objects Bearing Data - David Byrne and Rohini Sulatycki
- 2011 - Deserialization Spring RCE - Wouter Coekaerts
- 2015 - Marshaling Pickles - Chris Frohoff and Gabriel Lawrence

JAVA SERIALISATION

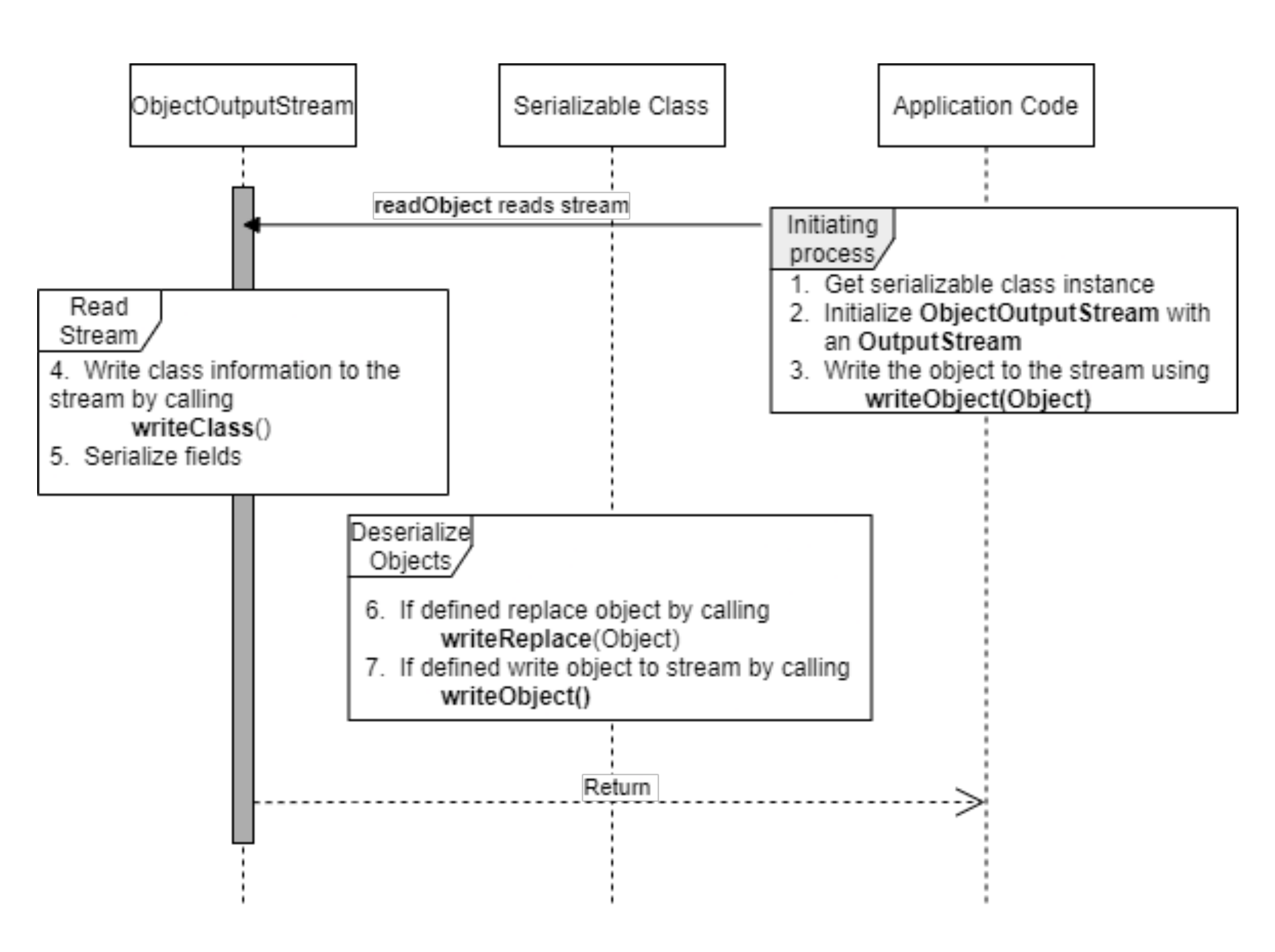
- What is it used for?
  Store and Retrieve
  Producer-consumer relationship
- Release Compatibility



SERIALIZABLE INTERFACE

- writeObject() - writes an object to a serialized format
- **readObject**() - read an object from a serialized format
- **readObjectNoData**() - control the initialization of its own fields and superclass
- **readResolve**() - replace the the object that has been read
- writeReplace() - replace the object being serialized with another object
- **readExternal**() - responsible for reading an objects state
- writeExternal() - responsible for saving an objects state

JAVA SERIALIZATION



Initiating process

1. Get serializable class instance
2. Initialize **ObjectOutputStream** with an **OutputStream**
3. Write the object to the stream using **writeObject(Object)**

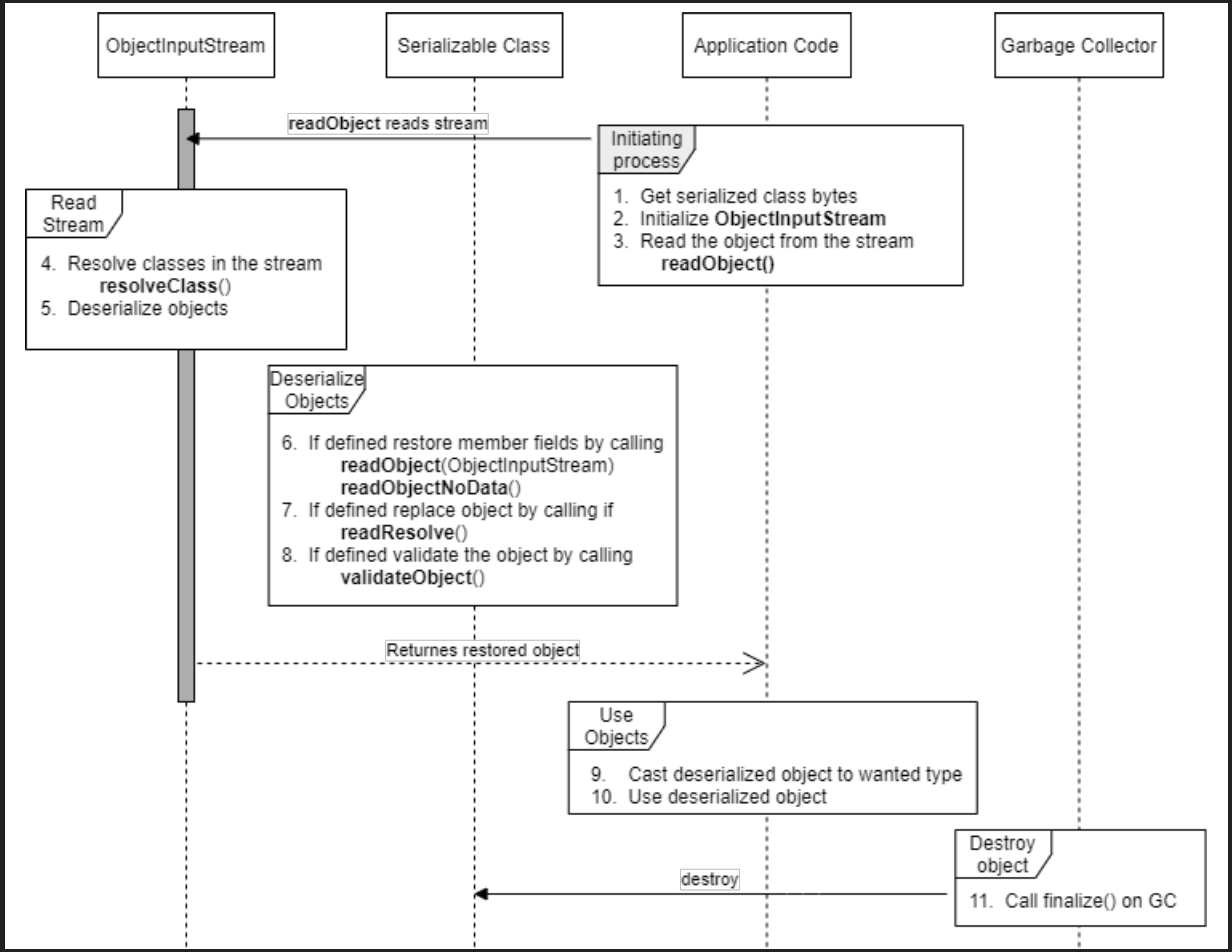
Read Stream

4. Write class information to the stream by calling **writeClass**()
5. Serialize fields

Deserialize Objects

6. If defined replace object by calling **writeReplace**(Object)
7. If defined write object to stream by calling **writeObject()**

JAVA DESERIALIZATION



Initiating process

1. Get serialized class bytes
2. Initialize **ObjectInputStream**
3. Read the object from the stream
   readObject()

Read Stream

4. Resolve classes in the stream
   resolveClass()
5. Deserialize objects

Deserialize Objects

6. If defined restore member fields by calling
   readObject(ObjectInputStream)
   readObjectNoData()
7. If defined replace object by calling if
   readResolve()
8. If defined validate the object by calling
   validateObject()

Use Objects

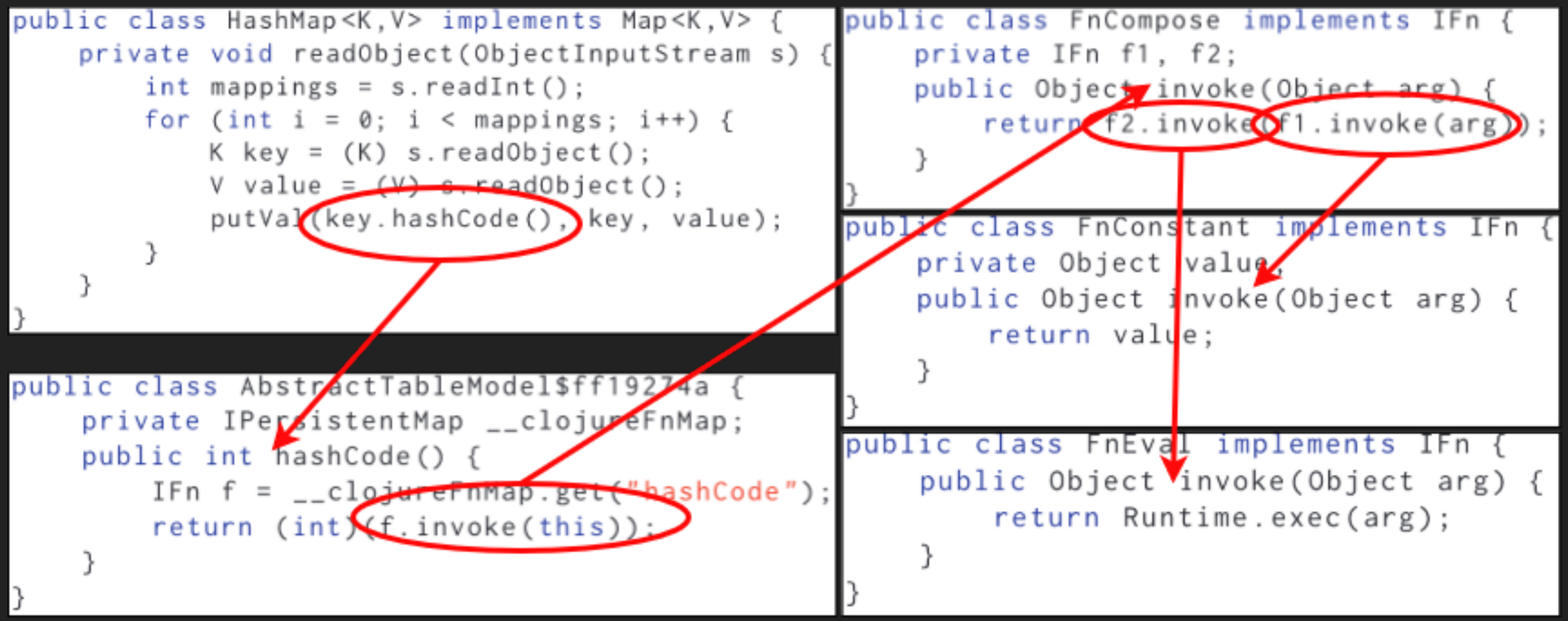
9. Cast deserialized object to wanted type
10. Use deserialized object

Destroy object

11. Call finalize() on GC

LOCATING GADGETS

- What is a gadget?
- What is a gadget chain?
- Trigger, Bypass, Helper, Abuse
- Large gadget space



```
public class HashMap<K,V> implements Map<K,V> {
    private void readObject(ObjectInputStream s) {
        int mappings = s.readInt();
        for (int i = 0; i < mappings; i++) {
            K key = (K) s.readObject();
            V value = (V) s.readObject();
            putVal(key.hashCode(), key, value);
        }
    }
}
```

```
public class AbstractTableModel$ff19274a {
    private IPersistentMap __clojureFnMap;
    public int hashCode() {
        IFn f = __clojureFnMap.get("hashCode");
        return (int)(f.invoke(this));
    }
}
```

```
public class FnCompose implements IFn {
    private IFn f1, f2;
    public Object invoke(Object arg) {
        return f2.invoke(f1.invoke(arg));
    }
}
```

```
public class FnConstant implements IFn {
    private Object value;
    public Object invoke(Object arg) {
        return value;
    }
}
```

```
public class FnEval implements IFn {
    public Object invoke(Object arg) {
        return Runtime.exec(arg);
    }
}
```

DESERIALIZATION ATTACK TECHNIQUES

- Variable Modification Attack
- Polymorphism Attack
- Deferred Execution Attack
- Gadget Chain Attack
- Proxy Attack

MITIGATION TECHNIQUES

- Consequences
- Do not Deserialize Untrusted Data
- Using Alternative Data Formats
- Blacklisting
- Whitelisting
- Java Serialization Filtering
- Web Application Firewall
- Signing Serialized Data
- Ad-hoc Security Manager
- Virtualization

TOOLS

- Ysoserial
- SerializationDumper
- Freddy
- GadgetInspector

DEMOS

DEMO 1 - VARIABLE MODIFICATION ATTACK

- 1. Modify the private variable in the serialized User Class
- 2. Send serialized object to profilePicture REST Endpoint
- 3 Echo “” | base64 -d

DEMO 2 - POLYMORPHISM ATTACK

- 1. Attacker must know about the Admin Class
- 2. Create an serialized byte stream representing the class.
- 3. git diff --no-index User_Serialized_Dumped.txt AdminUser_Serialized_Dumped.txt
- 4. Send the AdminUser byte stream to the listStatistics REST Endpoint

DEMO 3 - DEFERRED EXECUTION ATTACK

- 1. Patch a BoardState class with the bytes that should be saved to Disc
- Send the BoardState byte stream to any REST Endpoint
- Wait until the Garbage collector has run and saved the file to Disc

APACHE COMMON COLLECTION VULNERABILITY

- Can easily be created by ysoserial
- Can run arbitrary commands on any system accepting Untrusted serialised data and has the Apache.Commons.Collections library implemented.
- ysoserial CommonsCollections4 'chmod +x 1001.boardstate’
- ysoserial CommonsCollections4 ‘./1001.boardstate’

SUMMARY

- There is not a lot of research on the subject, but a lot of vulnerabilities
- Consequence for vulnerabilities are very severe
  RCE, Denial of Service, File Download/Upload
- Attack Techniques has been categorised into five categories
- Mitigation Strategies
- Future Work