Module 10 - Attacking Serialization

- Serialized data types:
    - i ⇒ integer ⇒ `i:42;`
    - d ⇒ double/float ⇒ `d:3.14;`
    - b ⇒ boolean ⇒ `b:1;`
    - a ⇒ array ⇒ `a:3:{i:0;i:1;i:1;i:2;i:2;i:3;}`
    - N ⇒ null ⇒ `N;`
    - s ⇒ string ⇒ `s:5:"apple";` ⇒ `O:5:"Class":1:{s:5:"prop1";s:3:"val";}`
    - O ⇒ object

▼ What's serialization?

- serialized data itself is not encrypted or signed in anyway.
- **be aware that serialization might not only be present on the web application layer.**

▼ Serialization in Java

```
//Item.java
package firstApp;
import java.io.Serializable;
public class Item implements Serializable{
    int id;
    String name;
    public Item(int id, String name) {
        this.id = id;
        this.name = name;
```

```
      }
 }
```

- standard java serialized format signature
    - 00000000 ac ed 00 05
- you should look for base64 strings starting with "rO0AB"

Insecure Deserialization Conditions

- a potentially exploitable condition in Java occurs when readObject() or a similar function is called on user-controlled object and later, a method on that object is called.
- **search: dynamic proxy & invocation handler**

Gadgets

- every property or method that is part of a nested exploit object is called a gadget
- **tool: ysoserial**

Introduction to Ysoserial

- its used to generate malicious payloads to test java insecure deserialization.

Usage

- `java -jar ysoserial.jar` ← displays help message
- output is often in binary format so I need to convert it to base64 then send it to the web application.
- **##### Google it #####**

Some Burp Suite Extentions

- Freddy, Deserialization Bug Finder

- Java Deserialization Scanner

BruteForce Attack with Ysoserial

- This also **Google it**

Studies from PortSwigger

Identifying Insecure Deserialization

- During auditing, you should look at all data being passed into the website and try to identify anything that looks like serialized data.
- Once you identify serialized data, you can test whether you are able to control it.

PHP serialization format

```
$user->name = "carlos";
$user->isLoggedIn = true;
```

This object can be serialized as

```
O:4:"User":2:{s:4:"name":s:6:"carlos"; s:10:"isLoggedIn":b
```

This can be interpreted as follows:

```
O:4:"User" - An object with the 4-character class name
"User"
2 - the object has 2 attributes
s:4:"name" - The key of the first attribute is the 4-cha
racter string "name"
s:6:"carlos" - The value of the first attribute is the 6
-character string "carlos"
s:10:"isLoggedIn" - The key of the second attribute is t
he 10-character string "isLoggedIn"
```

```
b:1 - The value of the second attribute is the boolean v
alue true
```

- **The native methods for PHP serialization are serialize() and unserialize(). If you have source code access, you should start by looking for unserialize() anywhere in the code and investigating further.**

▼ Java serialization format

- Some languages, such as Java, use binary serialization formats.
- This is more difficult to read, but you can still identify serialized data if you know how to recognize a few tell-tale signs.
- For example, serialized Java objects always begin with the same bytes, which are encoded as **ac ed** in hexadecimal and **rO0** in Base64.
- Any class that implements the interface **java.io.Serializable** can be serialized and deserialized.
- If you have source code access, take note of any code that uses the **readObject()** method, which is used to read and deserialize data from an InputStream.

▼ Manipulating serialized objects

- There are two approaches you can take when manipulating serialized objects.
    - You can either edit the object directly in its byte stream form
    - Or you can write a short script in the corresponding language to create and serialize the new object yourself.
        - The latter approach is often easier when working with binary serialization formats.

▼ Scenarios

[1] Modify serialized data to gain privilege escalation

[2] Modify serialized data (data type) in login cookie to gain unauthorized access

[+] example: php loose comparizon operator `==` returns true when comparing a string with `0` . Therefore when the login functionality is comparing password with user input, attacker might insert `0` and change its data type in the serialized cookie as login functionality compares password with cookie data.

▼ Magic Methods

- For example, PHP's unserialize() method looks for and invokes an object's __wakeup() magic method.
- In Java deserialization, the same applies to the ObjectInputStream.readObject() method, which is used to read data from the initial byte stream and essentially acts like a constructor for "re-initializing" a serialized object. However, Serializable classes can also declare their own readObject() method as follows:

```
private void readObject(ObjectInputStream in) throws IOExce
{
    // implementation
}
```

▼ Gadget chains

- In the wild, many insecure deserialization vulnerabilities will only be exploitable through the use of gadget chains.
- This can sometimes be a simple one or two-step chain, but constructing high-severity attacks will likely require a more elaborate sequence of object instantiations and method invocations.
- Therefore, being able to construct gadget chains is one of the key aspects of successfully exploiting insecure deserialization.

▼ Working with pre-built gadget chains

→ Ysoserial tool ← Java

```
 In Java versions 16 and above, you need to set a series
java -jar ysoserial-all.jar \
   --add-opens=java.xml/com.sun.org.apache.xalan.interna
   --add-opens=java.xml/com.sun.org.apache.xalan.interna
   --add-opens=java.base/java.net=ALL-UNNAMED \
   --add-opens=java.base/java.util=ALL-UNNAMED \
   [payload] '[command]'
```