jamk.fi

Assignment 4

Web Application Security

Joni Korpihalkola
K1625

Report
02/2018
Information and communication technology
ICT-field

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

Contents

Figures

1 Introduction

In this assignment the task is to penetration test a vulnerable java web application that uses serialized classes. The goal is to gain access to the admin view page with a deserialization attack and figure out a way execute shell commands in a separate attack. I have the java application running on a separate victim machine, that is a Kali Linux virtual machine. My attacker machine is also a Kali Linux. The application is running on address 192.168.82.4, port 8080.

"*To serialize an object means to convert its state to a byte stream so that the byte stream can be reverted back into a copy of the object. A Java object is serializable if its class or any of its superclasses implements either the java.io.Serializable interface or its subinterface, java.io.Externalizable. Deserialization is the process of converting the serialized form of an object back into a copy of the object.*" (Oracle. Serializable Objects. n.d)

If an attacker can control the serialized data that is sent and going to be deserialized by an application, he can potentially influence variables and objects used in the app.

2 Used tools

Name	Link	Description
Burpsuite	https://portswigger.net/burp	Used in this test to intercept HTTP requests sent to the application and to decode base64 data
JavaSerialKiller	https://github.com/NetSPI/JavaSerialKiller	An extension to burpsuite that generates payloads from ysoserial

JD Project	http://jd.benow.ca/	Java Decompiler
OWASP Dependency Check	https://www.owasp.org/index.php/OWASP_Dependency_Check	Identifies dependencies and checks for vulnerabilities in Java and .NET applications
Serialization Dumper	https://github.com/NickstaDB/SerializationDumper	Shows fields, classes and values in serialized Java streams.
ysoserial	https://github.com/frohoff/ysoserial	Generates payloads that exploit Java deserialization

3 Changing the admin bit

The application creates new users on login page, if the username doesn't already exist. New users are created with an 'admin bit' set to false as default. When we decompile the applications jar file and look at the classes, we can see that the logincookie has an isAdmin attribute that has a boolean value. So the application stores the admin bit inside the users cookie data. (Figure 1)

```java
public class LoginCookie
  implements Serializable
{
  private static final long serialVersionUID = 666L;
  @Id
  @GeneratedValue(generator="uuid2")
  @GenericGenerator(name="uuid2", strategy="uuid2")
  @Column(name="id", columnDefinition="UUID")
  private UUID id;
  @Column
  public boolean isAdmin = false;
  @Column(unique=true)
  private String userName = "";
  @Column
  private String secretKey;
  @Transient
  private Bag bag = new HashBag();

  public LoginCookie() {}

  public LoginCookie(String userName, String secretKey)
  {
    this.userName = userName;
    this.secretKey = secretKey;
    this.bag.add(userName);
    this.bag.add(secretKey);
  }
```

Figure 1. LoginCookie class code

The userinfo and adminview pages have an if-check with Thymeleaf to check if the user is an admin and display information accordingly. (Figure 2)

```
1  <!DOCTYPE HTML>
2  <html xmlns:th="http://www.thymeleaf.org">
3  <head>
4      <title>SECURE AUTHENTICATION PORTAL</title>
5      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6  </head>
7  <body>
8  <div th:if="${cookie.isAdmin}">
9      <h1>Users [ID Username AdminBit Password]</h1>
10     <ul>
11         <li th:each="item : ${users}" th:text="${item}">Item description here...</li>
12     </ul>
13 </div>
14 <div th:unless="${cookie.isAdmin}"><p>Admin bit is false. ACCESS DENIED.</p></div>
15 </body>
16 </html>
```

Figure 2. Thymeleaf if-check

I setup burpsuite to intercept my HTTP requests and get the cookie data that is sent to the application. The cookie data is encoded with base64, so I decoded it first to a file. Then I used the serializerdumper tool to simplify the cookie data. We can see that the value for isAdmin is 00 in hexadecimals. (Figure 3)

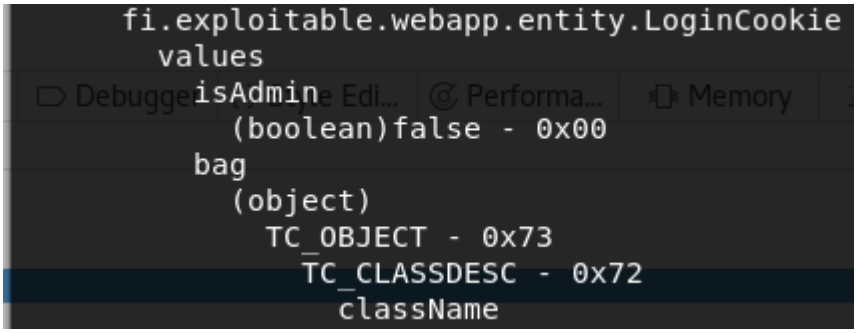

Figure 3. isAdmin value

I pasted the cookiedata to Burpsuites decoder to find the hexadecimal values. After decoding the data from base64, the isAdmin value is found on row b. (Figure 4)

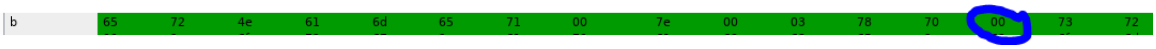

Figure 4. isAdmin hexadecimal value

When the 00-hexadecimal value is changed, a character is added to the empty space between “xp” and “sr”. (Figure 5)

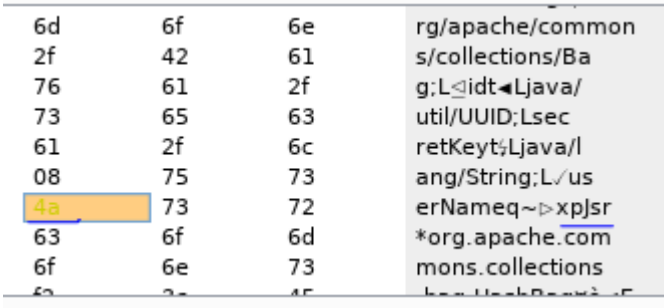

Figure 5. Changing the isAdmin value

When I changed the isAdmin value, encoded the modified cookie data back to base64 and send it forward to the application, I was able to gain access to the admin page of the application, that lists all created users. (Figure 6)

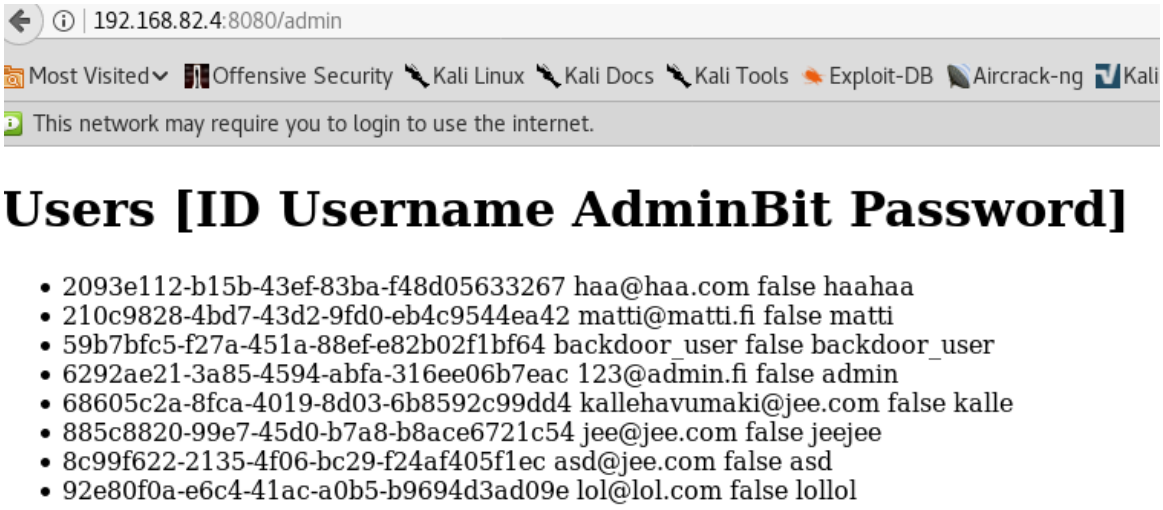

Figure 6. Applications admin view

The admin check when entering the page seems to only check if the admin bit is null, rather than have a separate true value, because changing the admin bit to a few different characters worked for me. Changing the application to check for a specific true value for the admin bit would slow down this attack, because an attacker would have to find a way to get an admins serialized cookie data and then find the admin bit there and compare it to a regular users cookie data.

4 Trying to gain shell access

Running the OWASP dependency check on the application reveals vulnerabilities in commonscollections 3.1, tomcat 8.5.27, ogni 3.0.8 and fasterxml jackson 2.8.10. (Figure 7)

Dependency	CPE	Coordinates	Highest Severity	CVE Count	CPE Confidence	Evidence Count
web-app-BEST-SNAPSHOT.jar	cpe:/a:in-portal:in-portal:-	fi.exploitable:web-app:BEST-SNAPSHOT	Medium	1	Low	23
web-app-BEST-SNAPSHOT.jar: commons-collections-3.1.jar	cpe:/a:apache:commons_collections:3.1	commons-collections:commons-collections:3.1 ✓	High	2	Low	29
web-app-BEST-SNAPSHOT.jar: tomcat-annotations-api-8.5.27.jar	cpe:/a:apache_tomcat:apache_tomcat:8.5.27 cpe:/a:apache:tomcat:8.5.27 cpe:/a:apache_software_foundation:tomcat:8.5.27	org.apache.tomcat:tomcat-annotations-api:8.5.27 ✓	High	3	Low	21
web-app-BEST-SNAPSHOT.jar: ognl-3.0.8.jar	cpe:/a:ognl_project:ognl:3.0.8	ognl:ognl:3.0.8 ✓	Medium	1	Low	22
web-app-BEST-SNAPSHOT.jar: jackson-databind-2.8.10.jar	cpe:/a:fasterxml:jackson-databind:2.8.10 cpe:/a:fasterxml:jackson:2.8.10	com.fasterxml.jackson.core:jackson-databind:2.8.10 ✓	High	2	Highest	38

Figure 7. OWASP dependency chack scan results

The highest severity vulnerabilities is listed by MITRE as CVE-2015-6420 . It is a vulnerability in Apache Commons Collections library that allows a remote attacker to

execute arbitrary code. An attacker could craft an input to the application that when deserialized executes code, which could for example open a reverse shell. (Vulnerability in Java Deserialization Affecting Cisco Products. Cisco Security Advisory. 25.2.2016)

To target the vulnerability I setup burpsuite again to intercept HTTP requests to the application. I generated a payload with ysoserial that targets commonscollections 3.1 version, the payload tries to create a file to the servers tmp directory. I then base64 encoded the payload and tried to insert it into cookie data and changed the HTTP method to POST. (Figure 8)

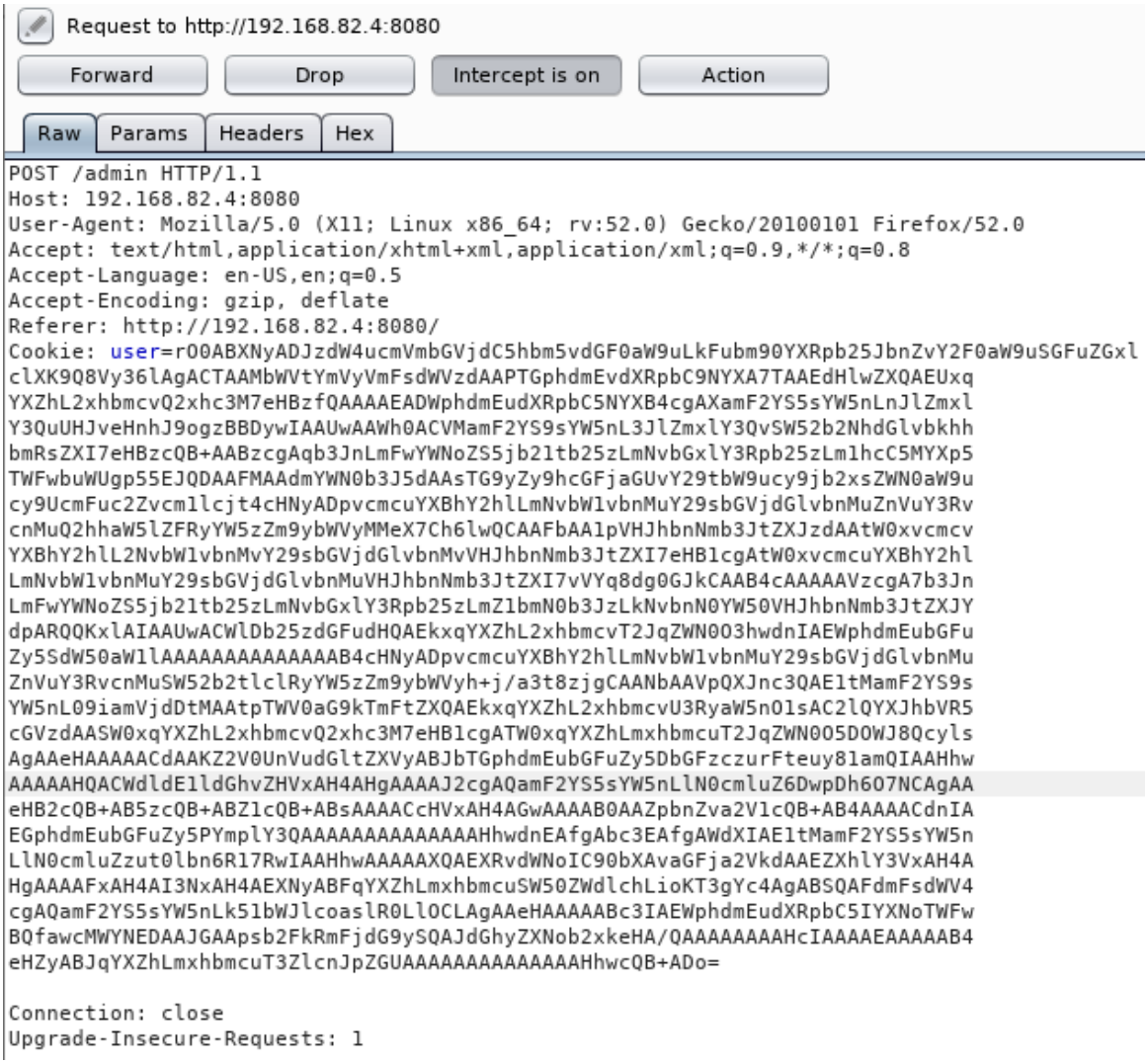

Figure 8. Payload inserted into cookie data

There was no luck with the payload. I tried ysoserials other payloads, like CommonsCollections3 et cetera. I also tried to insert the payload at the end of the HTTP

request, but nothing worked. I tried to find other entry points from jackson databind and tomcat, but I was unable to find anything that could give me shell access to the application.

Sources

Cisco Security Advisory. Vulnerability in Java Deserialization Affecting Cisco Products. 25.2.2016 https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20151209-java-deserialization

Oracle. Serializable Objects. n.d https://docs.oracle.com/javase/tutorial/jndi/objects/serial.html