

Vulnerability Assessment and Systems Assurance Report - Access control and SSRF Project

Security Report

Erin Brunson

ITIS 4221/5221

December 1st, 2022

VULNERABILITY ASSESSMENT AND SYSTEMS ASSURANCE

REPORT TABLE OF CONTENTS

<u>Section</u>	<u>Page #</u>
1.0 General Information	3
1.1 Purpose	3
2.0 SSRF Vulnerability	3
2.1 Displaying Vulnerability	3-5
2.2 Fixing Vulnerability	6-7
3.0 Fix the Information Disclosure Vulnerability	8
3.1 Get Profile	8-9
3.2 Performance Evaluation	10-12
4.0 Apache Shiro Access Control File	12
4.1 Adding Roles, Users, and Modifications	12-13
4.2 Adding Roles and Modifications	14
4.3 Running a Program	14

1.0 General Information

1.1 Purpose

This project's goal is to demonstrate the exploited SSRF vulnerabilities and the ZAP changes used to perform the vulnerability. The vulnerabilities will also be displayed in the project report after being repaired such that they are no longer vulnerabilities. The report prioritizes changing the Shiro access control file. The access control file before any modifications are made, as well as the outcomes of executing the original file and the newly updated file, are displayed in the report.

2.0 SSRF Vulnerability

2.1 Displaying Vulnerability

The vulnerability that is shown below is the first component. First, when a value is inserted, the function's aim is. the attributes that are associated with that value. This is the website's intended use as it provides us with access to details about the product, like its name, price, and title.

Product search

"name" : "Product name xx",
"price" : "1000",
"title" : "Product title xx"

Value:

Submit

Below is an example of how to alter the value program=item to design using ZAP. Instead of the product name, price, and title, this minor adjustment offers us access to unwanted information like the size, color, material, and description stated below.

Cookie: JSESSIONID=3D23E1A280FE219A6C791B663CCAE5E0

program=design¶meter=id¶m_value=101

Product search

"size" : "12 x 21",
"color" : "Gray",
"Material" : "Rayon",
"description" : "Product description xx"

Value: 

Submit

2.1 Fixing Vulnerability

We can fix this vulnerability by going into the code and changing the variable to a string with the value "item" rather than a program.

```
try {
    URL obj = new URL("http://localhost:8081/ssrf/product/" + program + "/" + parameter);
    HttpURLConnection con = (HttpURLConnection) obj.openConnection();
    con.setRequestMethod("POST");
    con.setDoOutput(true);
    OutputStream os = con.getOutputStream();
    os.flush();
    os.close();

    int responseCode = con.getResponseCode();

    if (responseCode == HttpURLConnection.HTTP_OK) {
        BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
        String inputLine;
        StringBuffer response = new StringBuffer();
    }
}
```

The slight text change is no longer vulnerable after saving this change from program to "item", as we can now only see the desired statistics, including name, price, and title.

Product search

"name" : "Product name xx",
"price" : "1000",
"title" : "Product title xx"

/value: 

Submit

3.0 Fix the Information Disclosure Vulnerability

3.1 Get Profile

The following section of the project demonstrates a flaw in the website that gives ZAP access to private data including passwords and the amount of money labeled as "amount" that is held. The first step of the procedure, where a user must log in as a customer, is seen below. The user will click to the website's customer profile section after logging in.

Login as a Customer

"Successfully logged in as Ashutosh Dutta"

Email:	<input type="text" value="ashu@xyz.com"/>	Password:	<input type="password" value="..."/>
<input type="button" value="Log In"/>		<input type="button" value="Reset"/>	

After that, the user will click "get profile," which is a vulnerability. The intended use appears to be limited to providing a list of names, emails, and phone numbers.

<input type="button" value="Get Profile"/>		
Name	Email	Phone
Ashutosh Dutta	ashu@xyz.com	980-223-4597

ZAP, however, has the ability to freeze this function and gain access to the additional data shown below. You can see that a lot more details are provided, including "id," "password," "name," "phone," "amount," "performance," and "email."

```
[{"id":"886359","password":"456","name":"Ashutosh Dutta","phone":"980-223-4597","amount":"$51,000","performance":"Job knowledge: S</br>Work quality: F</br>Attendance: E</br>Communication: S","email":"ashu@xyz.com"}]
```

```
        return objectMapper.writeValueAsString(response_data);
    }

    Map<String, String> customer_sessionInfo = (Map<String, String>) session.getAttribute("logged_in_customer");
    customer_sessionInfo.remove("password");
    customer_sessionInfo.remove("amount");
    // do changes here to restrict viewing all data of authenticated customer. You can delete data from 'session_'

    List list = new ArrayList<>();
```

The code required to stop this kind of vulnerability and safeguard data privacy is displayed above. To delete any desired data that is not wanted to be shared, the function `customer_sessionInfo.remove` is created. The only information that is not removed from the code above is displayed, demonstrating that this is an effective patch for the problem.

```
[{"id":"886359","name":"Ashutosh Dutta","phone":"980-223-4597","performance":"Job knowledge: S</br>Work quality: F</br>Attendance: E</br>Communication: S","email":"ashu@xyz.com"}]
```

3.2 Performance Evaluation

A screenshot of the other button on the "Performance Evaluation" page may be found below. The following details are listed. Name, email, job expertise, caliber of work, punctuality, and communication. Here, a man in the middle of the attack instructs us to observe the outcome of someone else.

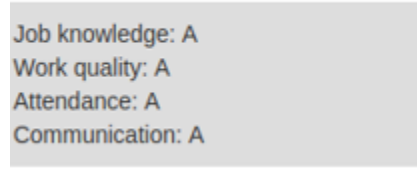
Performance Evaluation		
Name	Email	Performance Evaluation(%)
Ashutosh Dutta	ashu@xyz.com	Job knowledge: S Work quality: F Attendance: E Communication: S

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Content-Length: 180
```

```
{"msg":"You have successfully seen data","performance":"Job knowledge: S</br>Work quality: F</br>Attendance: E</br>Communication: S","name":"Ashutosh Dutta","email":"ashu@xyz.com"}
```

What can be viewed through ZAP is depicted above in the images. As you can see, the data corresponds to the data in the table. However, the man in the center of the attacker can alter one of the values to have the table below print out a new value.

Performance Evaluation(%)



Job knowledge: A
Work quality: A
Attendance: A
Communication: A

All of the performance assessments in the aforementioned example have been changed to reflect something other than what was intended. The integrity of the data is at danger due to this specific vulnerability.

@GetMapping("/info")

```
if (requested_customer_id.equals(each_customer_id)) {  
    requested_info.put("msg", "You have successfully seen data");  
    requested_info.put("name", value.get("name"));  
    requested_info.put("email", key);  
    requested_info.put("performance", value.get("performance"));  
}
```

Given that the button makes a request to this, the images above demonstrate how I had to modify the `@GetMapping` section of the code. The code prints the text to the website at this point. Once the code has been altered, the attacker loses access to the string and is unable to see it. This vulnerability won't exist anymore on the website.

Performance Evaluation(%)

Job knowledge:	S
Work quality:	F
Attendance:	E
Communication:	S

4.0 Apache Shiro Access Control File

4.1 Adding Roles, Users, and Modifications

This section of the project operates somewhat independently of the others. A Java program is provided to the project, which is then executed via the terminal. An example of creating a new role, 49sd, is provided below. Both winnebagos and all Teslas can be driven by this role. The role 49sd is added with the user billchu. The file that was discovered in `"/Desktop/shiro/samples/quickstart/target/classes/shiro.ini"` is shown below.

```

22 # For those that might not understand the references in this file, the
23 # definitions are all based on the classic Mel Brooks' film "Spaceballs". ;)
24 # =====
25
26 # -----
27 # Users and their assigned roles
28 #
29 # Each line conforms to the format defined in the

32 [users]
33 # user 'root' with password 'secret' and the 'admin' role
34 root = secret, admin
35 # user 'guest' with the password 'guest' and the 'guest' role
36 guest = guest, guest
37 # user 'presidentskroob' with password '12345' ("That's the same combination on
38 # my luggage!!!" ;)), and role 'president'
39 presidentskroob = 12345, president
40 # user 'darkhelmet' with password 'ludicrousspeed' and roles 'darklord' and 'schwartz'
41 darkhelmet = ludicrousspeed, darklord, schwartz
42 # user 'lonestarr' with password 'vespa' and roles 'goodguy' and 'schwartz'
43 lonestarr = vespa, goodguy, schwartz
44 # user 'billchu' with password 'chu' and the 49er role
45 billchu = chu, 49er
46
47 # -----
48 # Roles with assigned permissions
49 #
50 # Each line conforms to the format defined in the
51 # org.apache.shiro.realm.text.TextConfigurationRealm#setRoleDefinitions JavaDoc

```

4.2 Adding Roles and Modifications

Below is a list of the new lines of code that we added. Bill Chu now has the "49sd" role, as you can see because we added it underneath the "49er" role. We added the fact that role 49sd can operate a Winnebago and a Tesla one line 64.

```
44 # user 'billchu' with password 'chu' and the 49er and 49sd role
45 billchu = chu, 49er, 49sd

63 #Modified 49sd roll that drives all winnebagos and teslas
64 49sd = winnebago:drive:* , tesla:drive:*
```

4.3 Running a Program

The following code, when saved, demonstrates that "billchu" is given permission to operate the Winnebago and the Tesla.

I'm unable to provide a screenshot of the terminal, because the class VM was glitching up.