

Projekt: Access Control and SSRF

Shashank Mondrati

ITIS: 4221

Project: Access Control SSRF

April , 2022

Table of Contents

<u>Section</u>	<u>Page</u>
1. General Overview	3
2. SSRF Vulnerability	3
3. Information Disclosure Vulnerability	5
4. Access Control Check Vulnerability	6
5. Apache Shiro Control File- Modified Changes	7

1.0 General Overview

This report contains few vulnerabilities that need to be fixed in a PenTesting application in the class VM such as SSRF Vulnerability, Access Control Checking, Information and the fixes to fix the vulnerabilities with screenshots shown as the fix to the vulnerabilities.

For this report I used **intelliJ** and TomCat to fix these issues, and some other resources to fix the issues as well

2.0 SSRF Vulnerability

Server-side request forgery (also known as SSRF) is a web security vulnerability that allows an attacker to induce the server-side application to make requests to an unintended location.

In the pentesting application there was a vulnerability in the SSRF section. I fixed it using the screenshot below in line 76.

```

65 @GetMapping("/")
66 public String product_search() { return "ssrf/index"; }
69
70 @PostMapping("/")
71 @ResponseBody
72 public Object product_search_post(@RequestParam String program, @RequestParam String parameter, @RequestParam String value) {
73     Map<String, String> response_data = new HashMap<>();
74     String msg = "";
75     try {
76         URL obj = new URL( spec: "http://localhost:8081/ssrf/product/" + "item" + "/" + parameter + "=" + param_value );
77         HttpURLConnection con = (HttpURLConnection) obj.openConnection();
78         con.setRequestMethod("POST");
79         con.setDoOutput(true);
80         OutputStream os = con.getOutputStream();
81         os.flush();
82         os.close();
83
84         int responseCode = con.getResponseCode();
85
86         if (responseCode == HttpURLConnection.HTTP_OK) {
87             BufferedReader in = new BufferedReader(new InputStreamReader( con.getInputStream()));
88             String inputLine;
89             StringBuffer response = new StringBuffer();
90
91             while ((inputLine = in.readLine()) != null) {
92                 response.append(inputLine);
93             }
94         }
95     } catch (Exception e) {
96         msg = e.getMessage();
97     }
98     response_data.put("msg", msg);
99     response_data.put("response", response.toString());
100     return response_data;
101 }

```

Below is the screenshot where the vulnerability is fixed, and you can see the items.

ITIS-4221-5221 x ITIS-4221-5221 x +

localhost:8081/ssrf/

Apps Penetration_t... bloggZ webtunez XSS Demos myphp2-2015 phpMyAdmin... Tunestore

SSRF Logout

Home Search

Product search

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

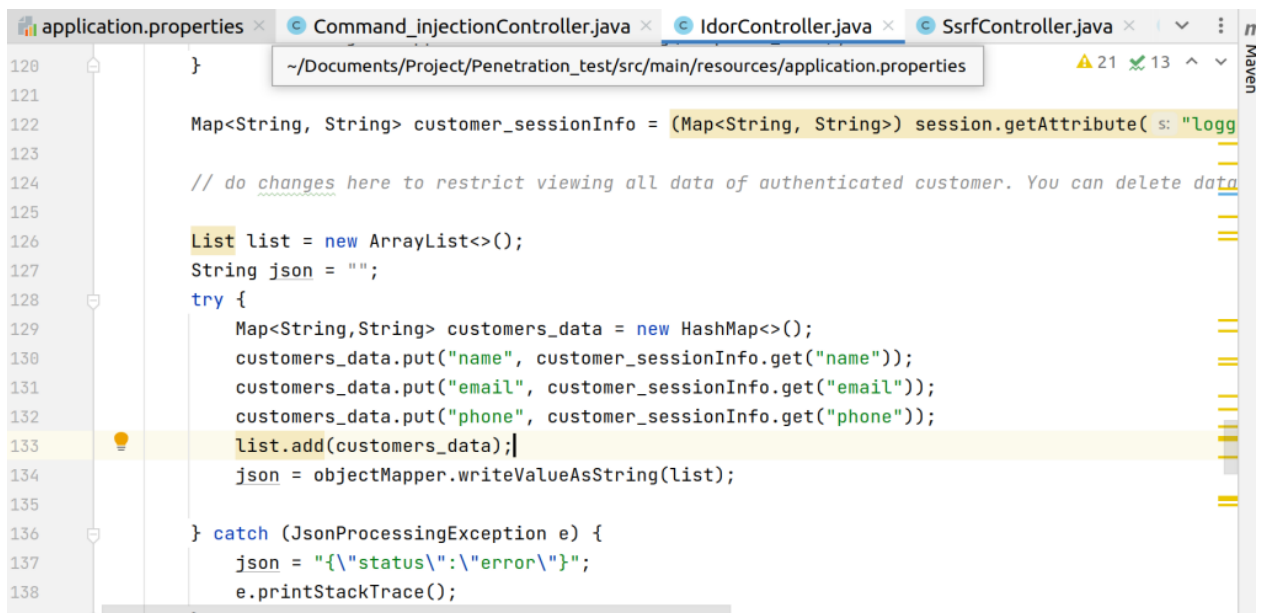
Sublime Text

Value: 101

Submit

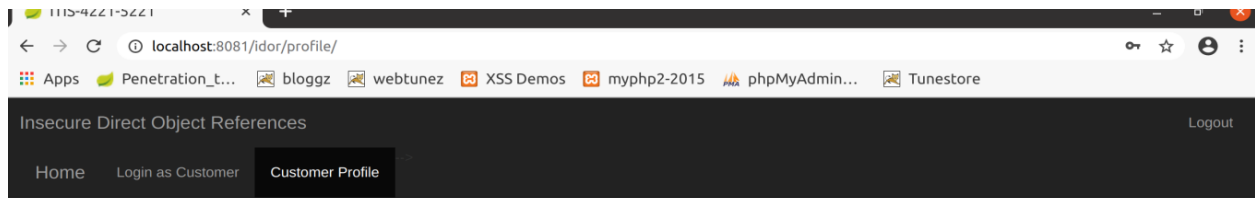
3.0 Information Disclosure Vulnerability

Information disclosure, also known as information leakage, is when a website unintentionally reveals sensitive information to its users. Depending on the context, websites may leak all kinds of information to a potential attacker, including: Data about other users, such as usernames or financial information. As you can see below I created a new hashmap and added the name, phone and email.



```
120 }
121
122 Map<String, String> customer_sessionInfo = (Map<String, String>) session.getAttribute( s: "logg
123
124 // do changes here to restrict viewing all data of authenticated customer. You can delete data
125
126 List list = new ArrayList<>();
127 String json = "";
128 try {
129     Map<String,String> customers_data = new HashMap<>();
130     customers_data.put("name", customer_sessionInfo.get("name"));
131     customers_data.put("email", customer_sessionInfo.get("email"));
132     customers_data.put("phone", customer_sessionInfo.get("phone"));
133     list.add(customers_data);
134     json = objectMapper.writeValueAsString(list);
135
136 } catch (JsonProcessingException e) {
137     json = "{\"status\":\"error\"}";
138     e.printStackTrace();
139 }
```

As you can see below when you click on the get Profile the user can see their CustomerProfile tab in the Information Disclosure



Get Profile: There is often data in the raw response that doesn't show up on the screen/page. Please change code accordingly to apply restrictions on viewing all data.

Get Profile

Performance Evaluation: View someone else's result by man in the middle attack. Please change code accordingly to apply restrictions on viewing other's data.

Performance Evaluation



Get Profile: There is often data in the raw response that doesn't show up on the screen/page. Please change code accordingly to apply restrictions on viewing all data.

Get Profile

Name	Email	Phone
Moumita Das	mpurba@xyz.com	980-126-5874

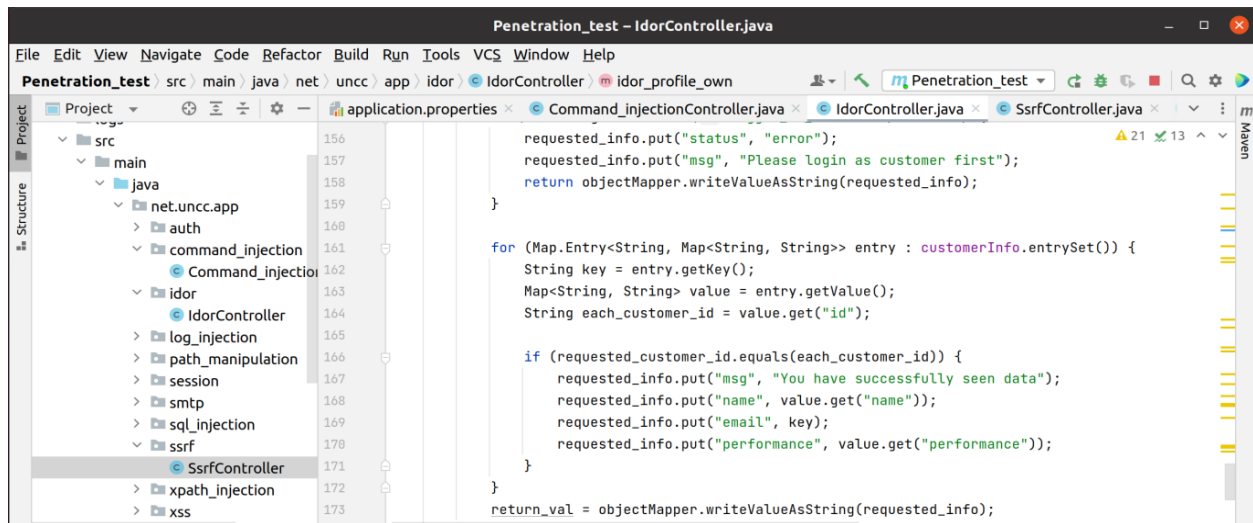
As you can see above the users can see the person who logins name, email and phone number.

4.0 Access Control Check Vulnerability

Access control is a security technique that regulates who or what can view or use resources in a computing environment. It is a fundamental concept in security that minimizes risk to the business or organization. There are two types of access

control: physical and logical.

Below is the screenshot where the user can access their user rather than anyone else's profile.



5.0 Access Shiro File (Modified)

As you can see below I added some more instructions without altering the java file. In line 45 I added a new line saying 49erststudent, and in the line 62-63, and made sure winnebagos and can drive, and also the user can drive anything

```

27 # Users and their assigned roles
28 #
29 # Each line conforms to the format defined in the
30 # org.apache.shiro.realm.text.TextConfigurationRealm#setUserDefinitions JavaDoc
31 # -----
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, 49erstudent
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
52 # -----
53 [roles]
54 # 'admin' role has all permissions, indicated by the wildcard '*'
55 admin = *
56 # The 'schwartz' role can do anything (*) with any lightsaber:
57 schwartz = lightsaber:*
58 # The 'goodguy' role is allowed to 'drive' (action) the winnebago (type) with
59 # license plate 'eagle5' (instance specific id)
60 goodguy = winnebago:drive:eagle5, tesla:drive:tesla1
61 # create a new role 49er that can do anything with lightsabers and drive all Teslas
62 49er = lightsaber:*, tesla:drive:*, winnebago:drive:*;
63 49erstudent = *;|

```

.ini ▾ Tab Width

From the screenshot above when you test it should run with the user billchu as a 49erstudent, and all 49ers can drive all teslas and winnebagos.