Access Control and SSRF - Project Pentesting

Access Control and SSRF Project Pentesting

Amanuel Haile

December 3rd, 2024

VULNERABILITY ASSESSMENT AND SYSTEMS ASSURANCE REPORT

TABLE OF CONTENTS

1.0 General Information

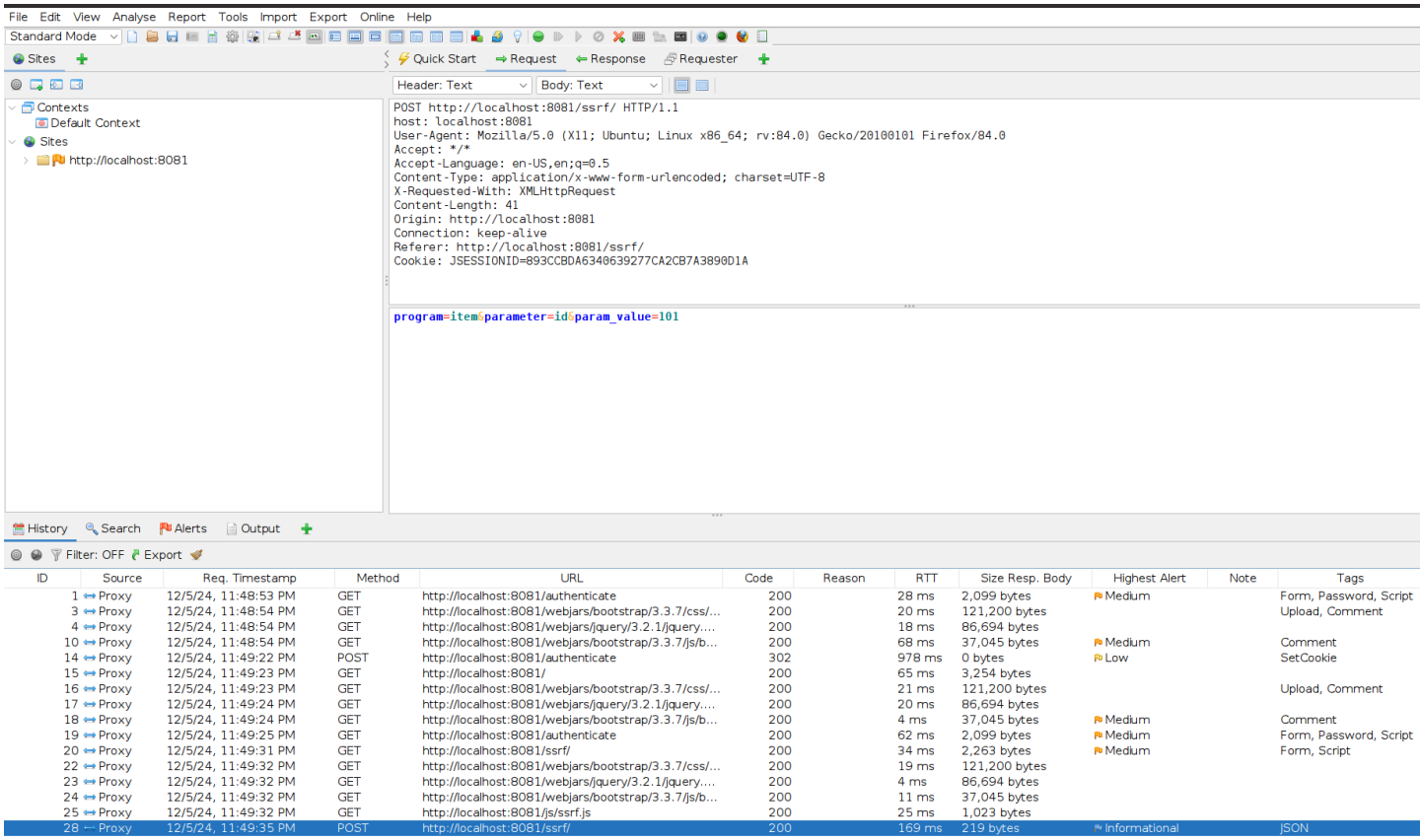
1.1 Purpose

The purpose of this vulnerability assessment and penetration test is to analyze the security of this application. The objective of this report is to discover and demonstrate the exploitation of various security vulnerabilities, specifically SSRF, Insecure Direct Object Reference (Information Disclosure and Access Control vulnerabilities), and improper implementation of security controls.

2.0 SSRF

2.1 SSRF - Vulnerability

Original POST request body:

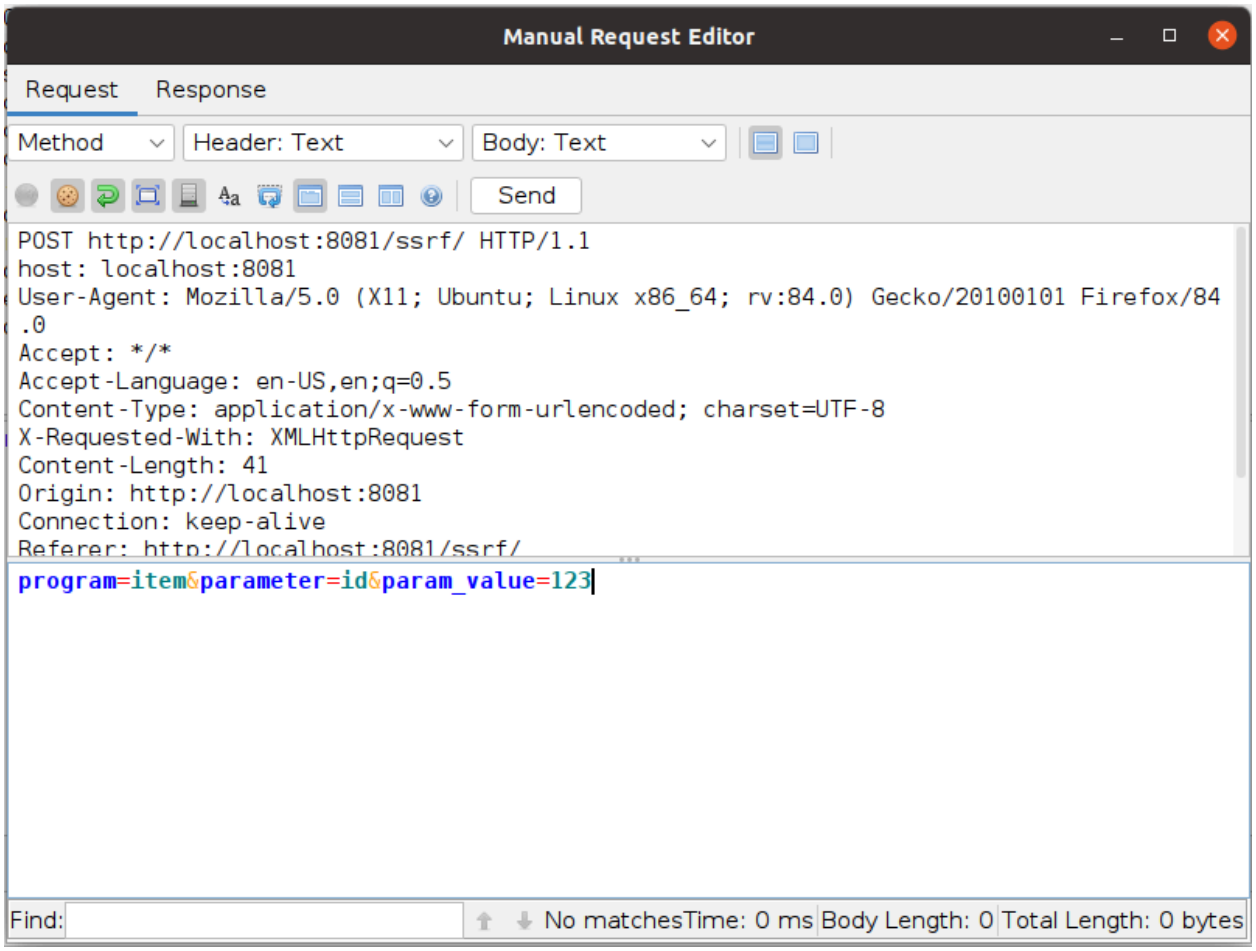


ID	Source	Req. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Body	Highest Alert	Note	Tags
1	Proxy	12/5/24, 11:48:53 PM	GET	http://localhost:8081/authenticate	200		28 ms	2,099 bytes	Medium		Form, Password, Script
3	Proxy	12/5/24, 11:48:54 PM	GET	http://localhost:8081/webjars/bootstrap/3.3.7/css/...	200		20 ms	121,200 bytes			Upload, Comment
4	Proxy	12/5/24, 11:48:54 PM	GET	http://localhost:8081/webjars/jquery/3.2.1/jquery....	200		18 ms	86,694 bytes			
10	Proxy	12/5/24, 11:48:54 PM	GET	http://localhost:8081/webjars/bootstrap/3.3.7/js/b...	200		68 ms	37,045 bytes	Medium		Comment
14	Proxy	12/5/24, 11:49:22 PM	POST	http://localhost:8081/authenticate	302		978 ms	0 bytes	Low		SetCookie
15	Proxy	12/5/24, 11:49:23 PM	GET	http://localhost:8081/	200		65 ms	3,254 bytes			
16	Proxy	12/5/24, 11:49:23 PM	GET	http://localhost:8081/webjars/bootstrap/3.3.7/css/...	200		21 ms	121,200 bytes			Upload, Comment
17	Proxy	12/5/24, 11:49:24 PM	GET	http://localhost:8081/webjars/jquery/3.2.1/jquery....	200		20 ms	86,694 bytes			
18	Proxy	12/5/24, 11:49:24 PM	GET	http://localhost:8081/webjars/bootstrap/3.3.7/js/b...	200		4 ms	37,045 bytes	Medium		Comment
19	Proxy	12/5/24, 11:49:25 PM	GET	http://localhost:8081/authenticate	200		62 ms	2,099 bytes	Medium		Form, Password, Script
20	Proxy	12/5/24, 11:49:31 PM	GET	http://localhost:8081/ssrf/	200		34 ms	2,263 bytes	Medium		Form, Script
22	Proxy	12/5/24, 11:49:32 PM	GET	http://localhost:8081/webjars/bootstrap/3.3.7/css/...	200		19 ms	121,200 bytes			
23	Proxy	12/5/24, 11:49:32 PM	GET	http://localhost:8081/webjars/jquery/3.2.1/jquery....	200		4 ms	86,694 bytes			
24	Proxy	12/5/24, 11:49:32 PM	GET	http://localhost:8081/webjars/bootstrap/3.3.7/js/b...	200		11 ms	37,045 bytes			
25	Proxy	12/5/24, 11:49:32 PM	GET	http://localhost:8081/js/ssrf.js	200		25 ms	1,023 bytes			
28	Proxy	12/5/24, 11:49:35 PM	POST	http://localhost:8081/ssrf/	200		169 ms	219 bytes	Informational		JSON

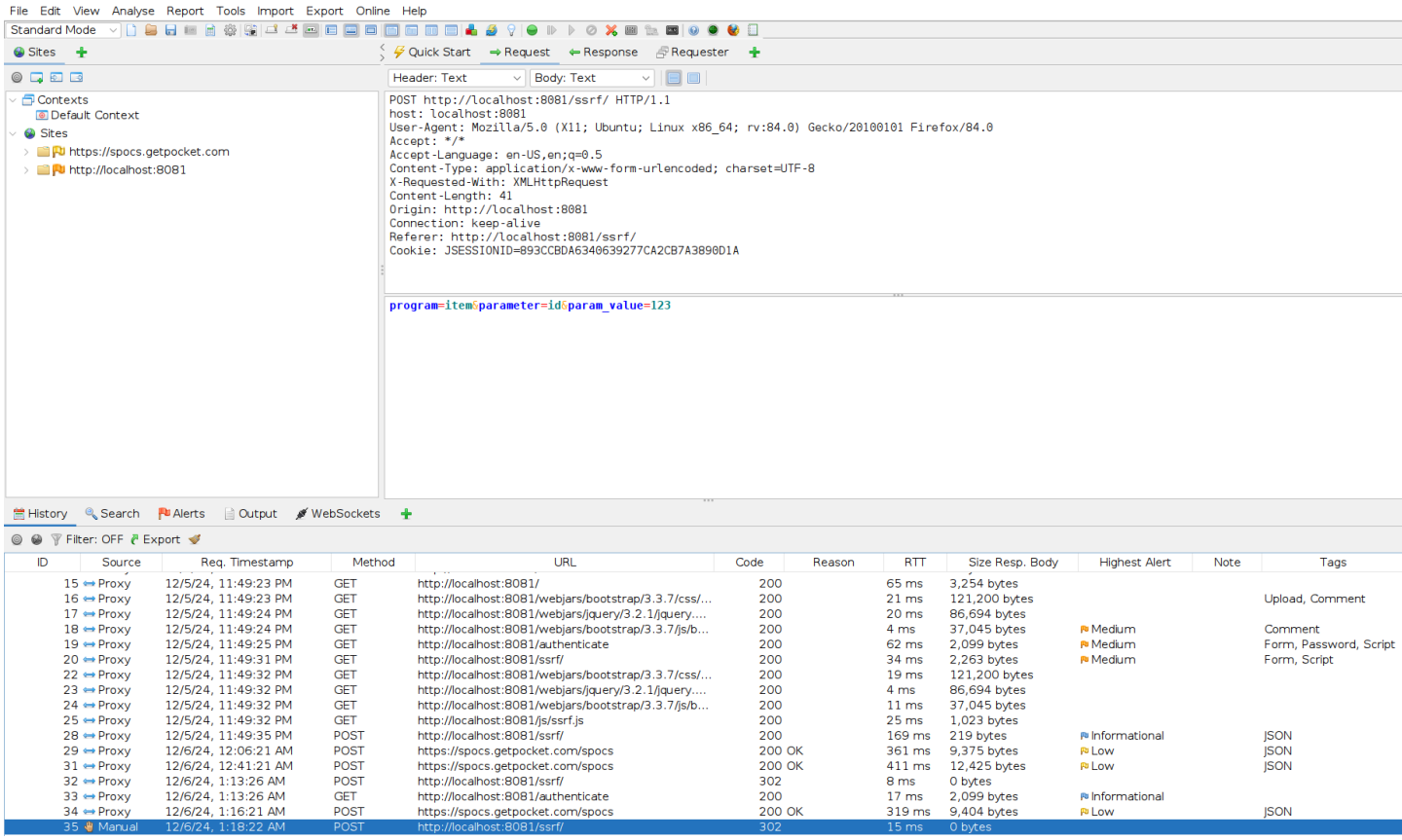
Vulnerable Code (the highlighted line):

```
public Object product_search_post(@RequestParam String program, @RequestParam String parameter, @RequestParam String param_value) {
        Map<String, String> response_data = new HashMap<String, String>();
        String msg = "";
        try {
                URL obj = new URL("http://localhost:8081/ssrf/product/" + program + "/?" + parameter + "=" + param_value);
                HttpURLConnection con = (HttpURLConnection) obj.openConnection();
                con.setRequestMethod("POST");
                con.setDoOutput(true);
                OutputStream os = con.getOutputStream();
```

This code can be exploited by sending a request with changed parameters as seen below:



Here is the screenshot showing that the request was successfully sent:



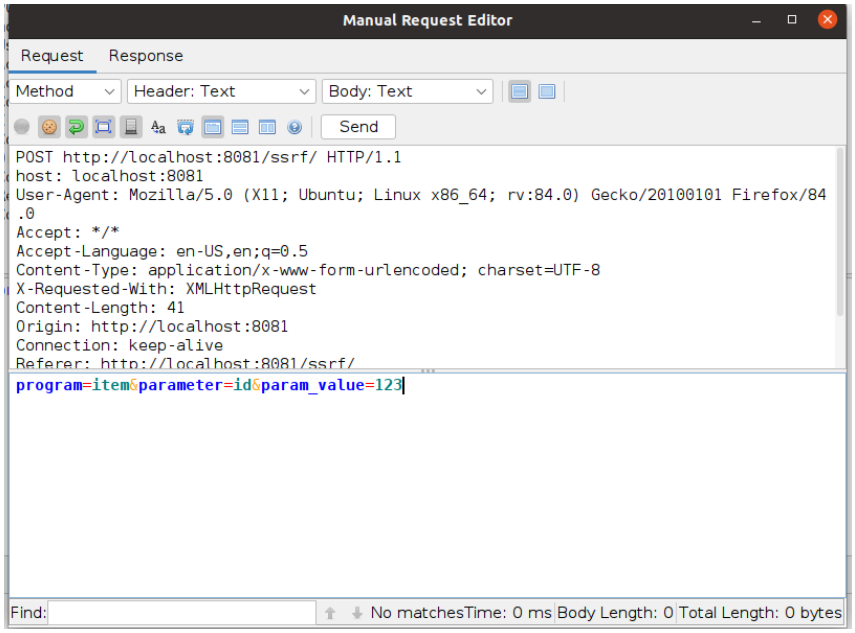
2.2 SSRF - Fix

Here is the fixed code (in the highlighted area):

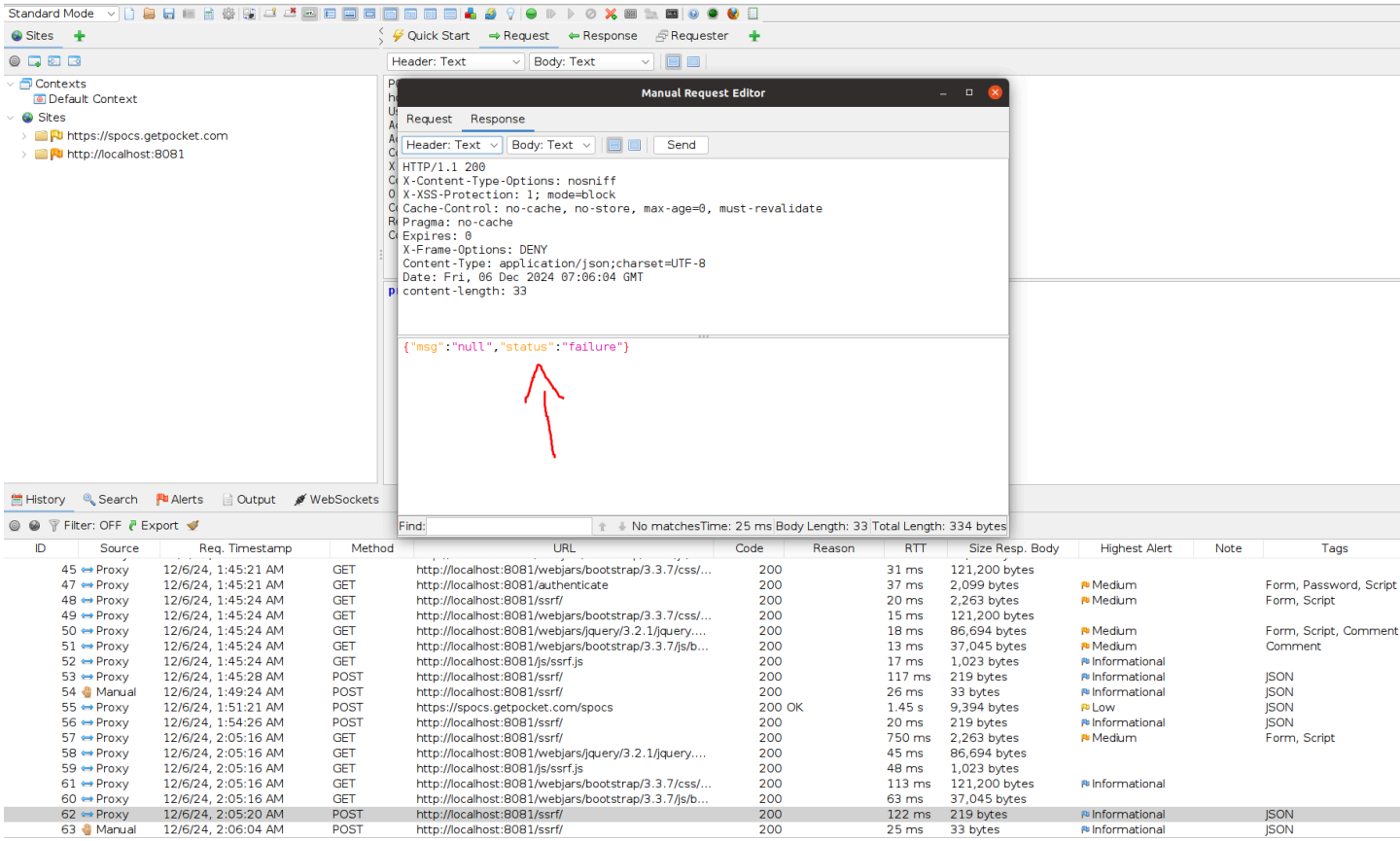
```
@PostMapping("/")
@ResponseBody
public Object product_search_post(@RequestParam String program, @RequestParam String parameter, @RequestParam String param_value) {
    Map<String, String> response_data = new HashMap<>();
    String msg = "";
    final List<String> allowedPrograms = Arrays.asList("item", "design");
    final List<String> allowedParameters = Arrays.asList("id", "type");

    try {
        if (!allowedPrograms.contains(program)) {
            throw new IllegalArgumentException("Invalid program value: " + program);
        }
        if (!allowedParameters.contains(parameter)) {
            throw new IllegalArgumentException("Invalid parameter value: " + parameter);
        }
        String baseUrl = "http://localhost:8081/ssrf/product/";
        String safeUrl = String.format("%s%s/?%s=%s", baseUrl, program, parameter, param_value);

        URL obj = new URL(safeUrl);
        HttpURLConnection con = (HttpURLConnection) obj.openConnection();
        con.setRequestMethod("POST");
        con.setDoOutput(true);
        try (OutputStream os = con.getOutputStream()) {
            os.flush();
        }

        int responseCode = con.getResponseCode();
```

The fix involves adding validation for the program and parameter values by whitelisting only trusted options in the drop down menu. This ensures that the application only sends requests to the correct endpoints, which fixes the SSRF vulnerability.

Here is the request body again with the changed parameters:



And here is the result of sending the new request with the updated code:



Vulnerability assessment and System Assurance Report 8

3.0 IDOR Information Disclosure

3.1 IDOR Information Disclosure - Vulnerability

There is an information disclosure vulnerability in the “Get Profile” Button on this app. When a user logs into the “idor” page and goes to the “Customer Profile” section, a button with the name “Get Profile” appears. If the victim decides to click on the “Get Profile” button, this information is shown:

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

The issue is that this information is shown in ZAP when the same GET request is sent again through the ZAP app, which shows that the server is sending more information than what is necessary to the user:

```
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: 214
Date: Sat, 07 Dec 2024 10:06:45 GMT
```


```
[{"id":"886459","password":"123","name":"Moumita Das","phone":"980-126-5874","amount":
"$52,000","performance":
"Job knowledge: F</br>Work quality: S</br>Attendance: E</br>Communication: S","email":
"mpurba@xyz.com"}]
```

3.2 IDOR Information Disclosure - Fix

Here is the vulnerable code for the Information Disclosure vulnerability:

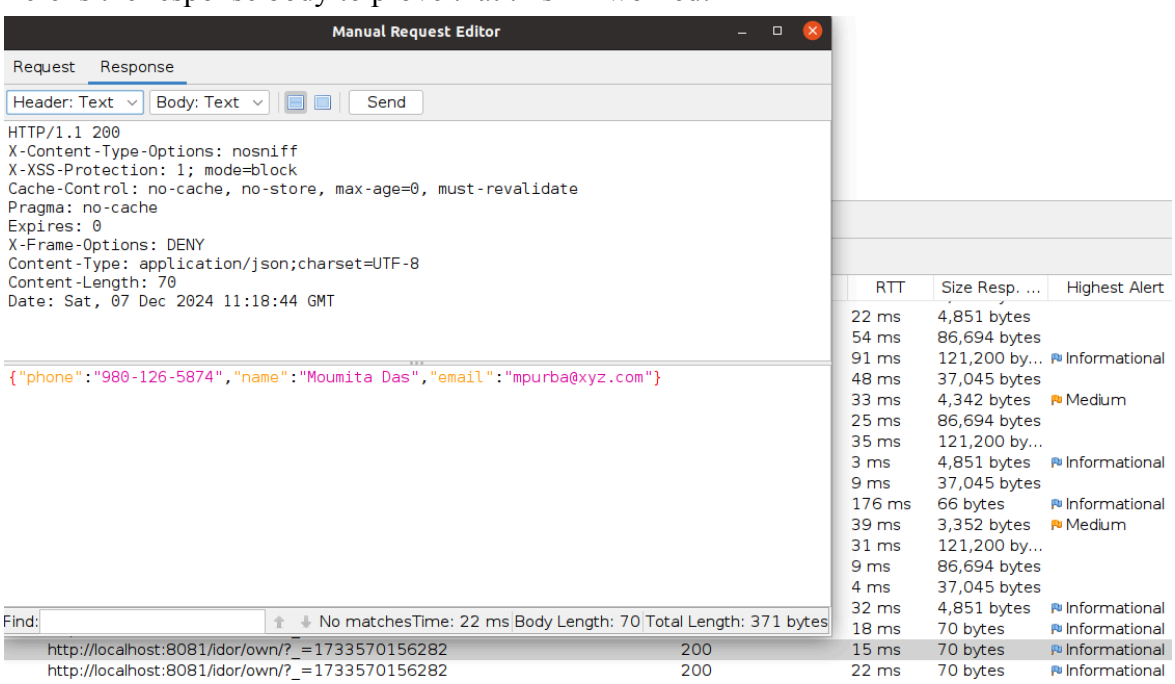
```
List list = new ArrayList<>();
String json = "";
try {
        list.add(customer_sessionInfo);
        json = objectMapper.writeValueAsString(list);

} catch (JsonProcessingException e) {
        json = "{\"status\":\"error\"}";
    e.printStackTrace();
}
return json;
```

The issue is that the password is included in the JSON response, which exposes it to the client.

And here is the updated code:

```
@GetMapping("/own")
@ResponseBody
public String idor_profile_own(HttpServletRequest request) throws Exception {
    HttpSession session = request.getSession();
    ObjectMapper objectMapper = new ObjectMapper();
    if (session.getAttribute("logged_in_customer") == null) {
        Map<String, String> response_data = new HashMap<>();
        response_data.put("status", "error");
        response_data.put("msg", "Please login as customer first");
        return objectMapper.writeValueAsString(response_data);
    }
    Map<String, String> customer_sessionInfo = (Map<String, String>) session.getAttribute("logged_in_customer");
    Map<String, String> sanitizedResponse = new HashMap<>();
    sanitizedResponse.put("name", customer_sessionInfo.get("name"));
    sanitizedResponse.put("email", customer_sessionInfo.get("email"));
    sanitizedResponse.put("phone", customer_sessionInfo.get("phone"));
    return objectMapper.writeValueAsString(sanitizedResponse);
}
```

The method now only includes the name, email, and phone in the JSON response. Sensitive data such as id, password, and amount are not included.
Here is the response body to prove that this fix worked:



4.0 IDOR Access Control

4.1 IDOR Access Control - Vulnerability

There is also a access control vulnerability in the Performance Evaluation button, and the information that would usually be shown to the user can be seen below:

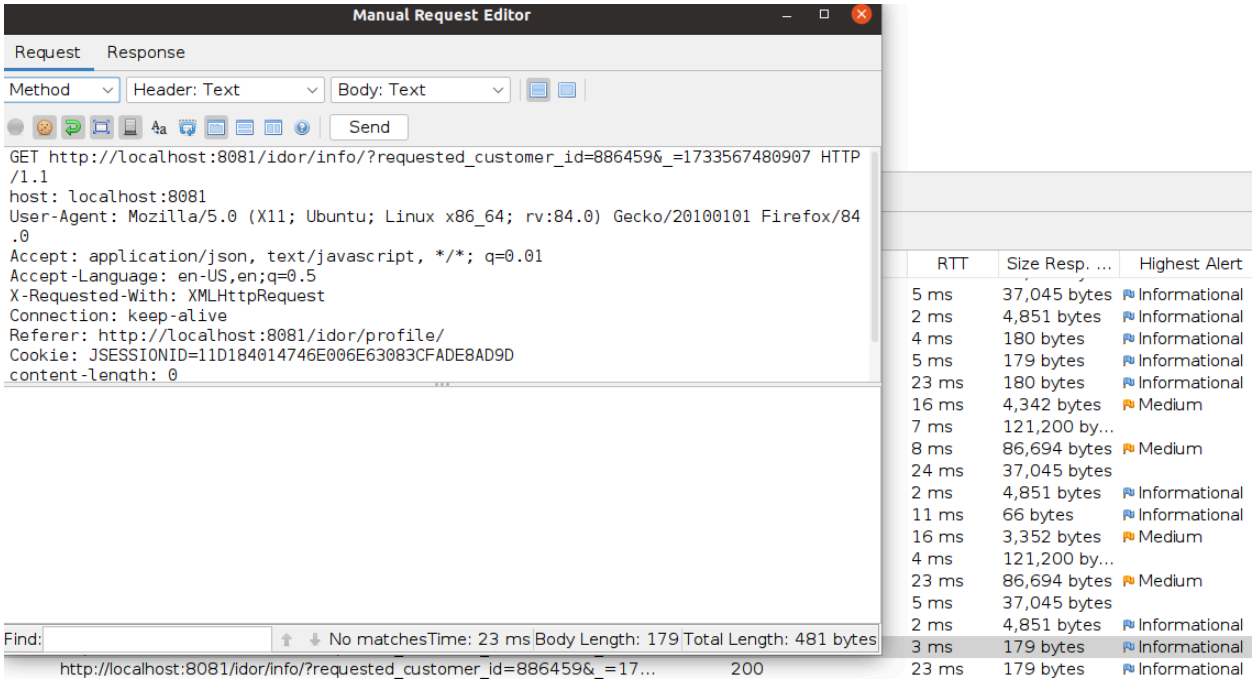
Performance Evaluation: View someone else's result by man in the middle attack. Please change code accordingly to apply restrictions on viewing other's data.

Performance Evaluation

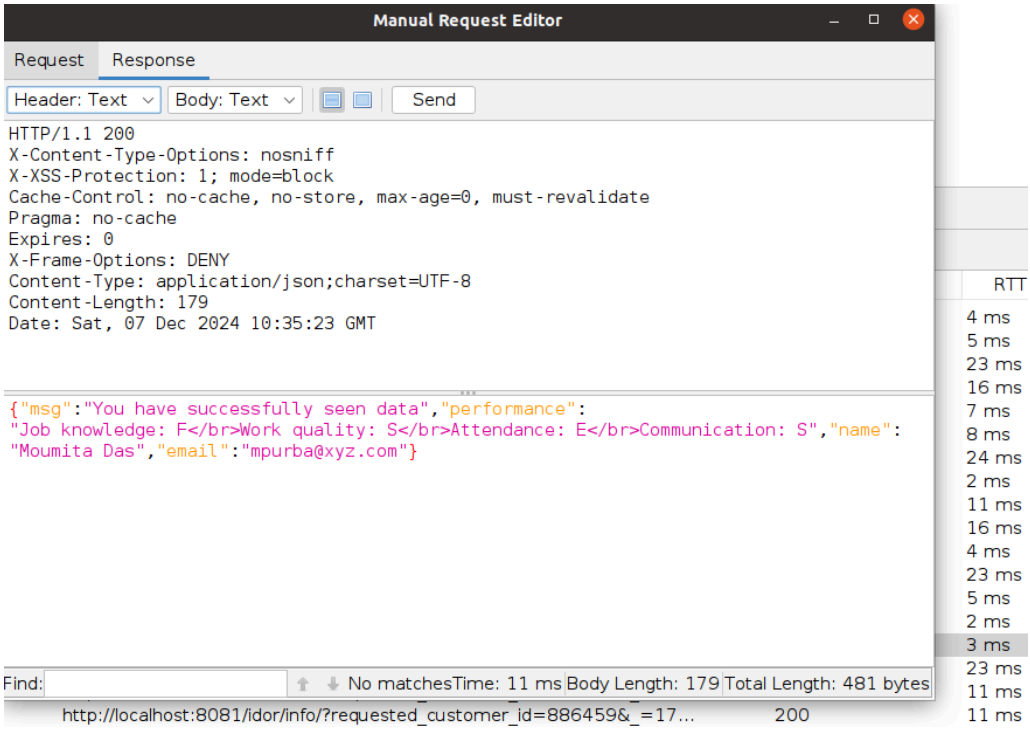
Name	Email	Performance Evaluation(%)
Moumita Das	mpurba@xyz.com	Job knowledge: F Work quality: S Attendance: E Communication: S

On ZAP, if the customer_id is changed to a different user's, you will be able to see their information instead, which can be seen below:

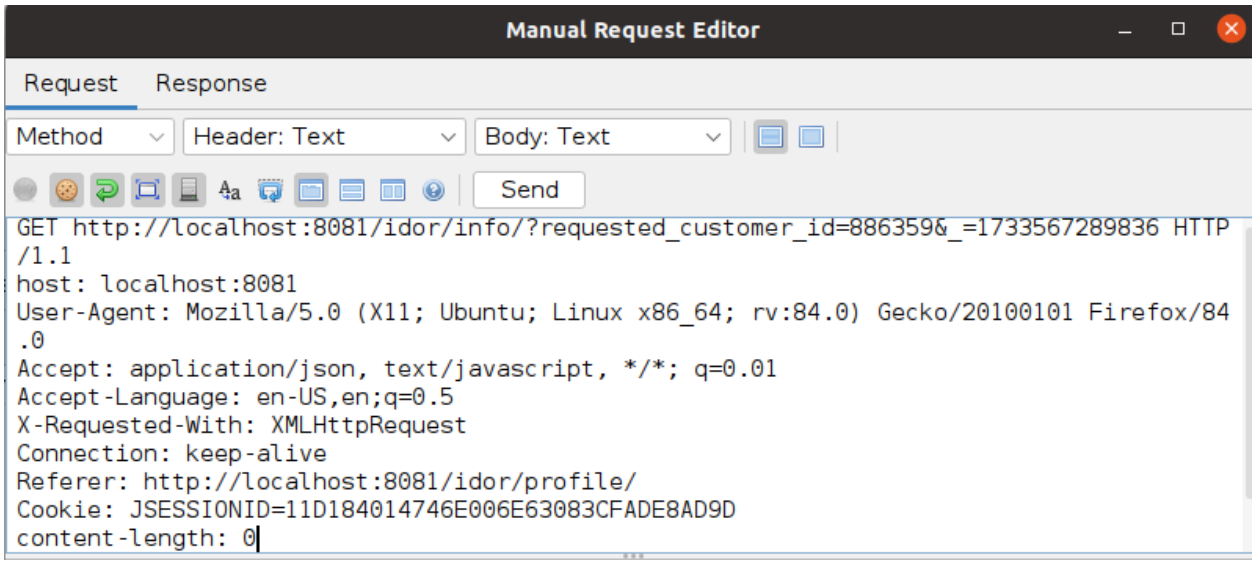
Original GET request body:



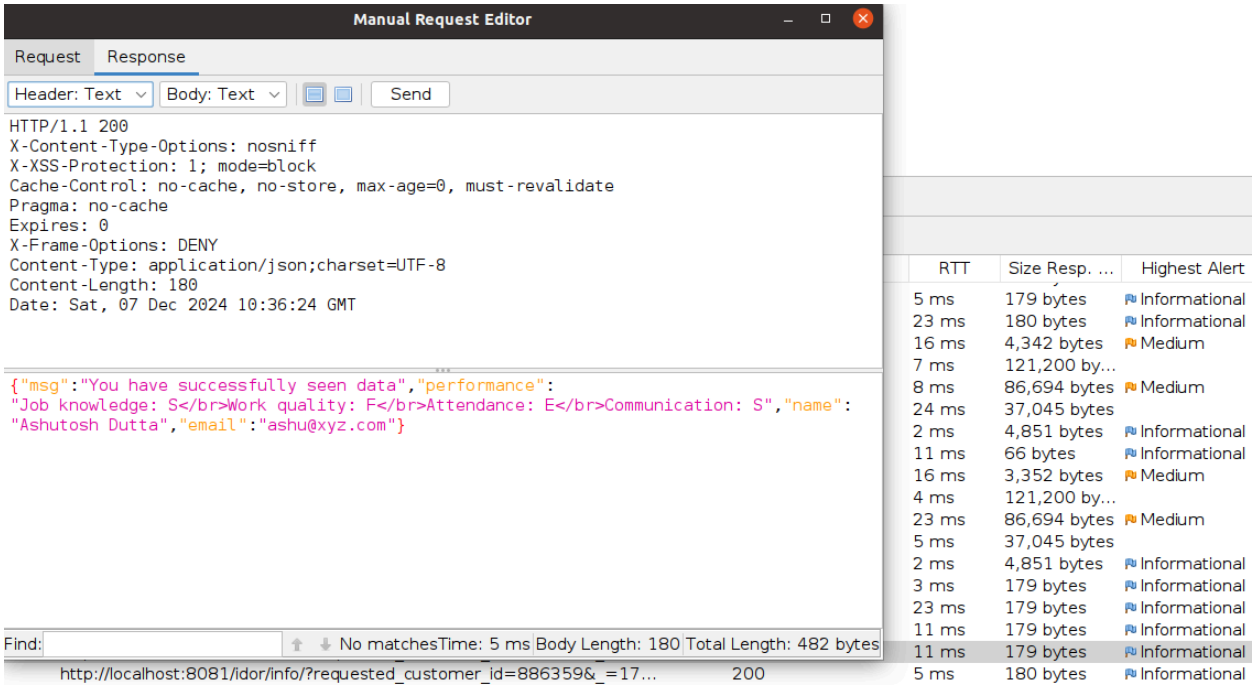
Original Response Body:



Changed customer_id:



Response body:



This shows that the server does not validate whether the logged-in user is authorized to view the requested customer's information, resulting in unauthorized data exposure.

4.2 IDOR Access Control - Fix

Here is the vulnerable code for the Access Control vulnerability:

```
for (Map.Entry<String, Map<String, String>> entry : customerInfo.entrySet()) {
        String key = entry.getKey();
        Map<String, String> value = entry.getValue();
        String each_customer_id = value.get("id");

        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"));
        }
}
```

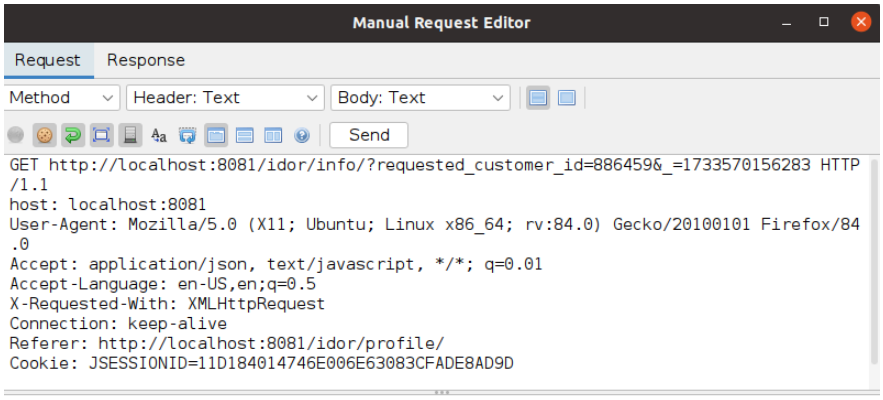
The issue is that the endpoint is allowing querying of any requested_customer_id without verifying if the logged-in customer was authorized to view the requested data.

And here is the updated code (the main fix is highlighted):

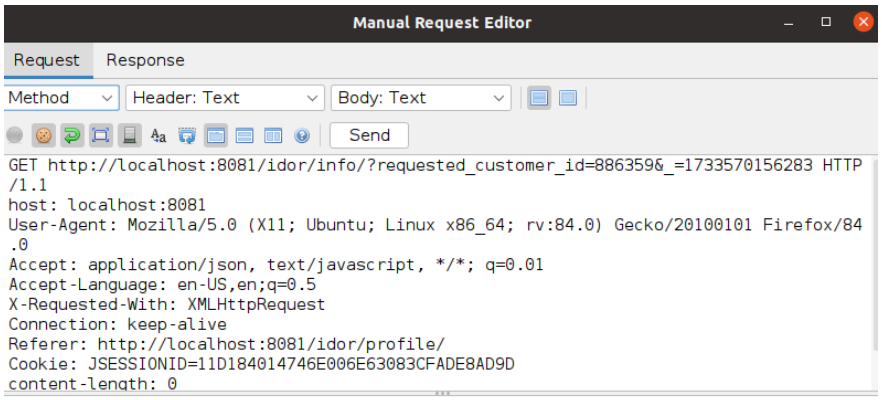
```
@GetMapping("/info")
@ResponseBody
public String idor_performance_info(@RequestParam String requested_customer_id, HttpServletRequest request) throws Exception {
    Map<String, String> requested_info = new HashMap<>();
    HttpSession session = request.getSession();
    String return_val = "";
    try {
        ObjectMapper objectMapper = new ObjectMapper();
        if (session.getAttribute("logged_in_customer") == null) {
            requested_info.put("status", "error");
            requested_info.put("msg", "Please login as customer first");
            return objectMapper.writeValueAsString(requested_info);
        }

        Map<String, String> loggedInCustomer = (Map<String, String>) session.getAttribute("logged_in_customer");
        if (!loggedInCustomer.get("id").equals(requested_customer_id)) {
            requested_info.put("status", "error");
            requested_info.put("msg", "You are not authorized to view this data");
            return objectMapper.writeValueAsString(requested_info);
        }

        for (Map.Entry<String, Map<String, String>> entry : customerInfo.entrySet()) {
            String key = entry.getKey();
            Map<String, String> value = entry.getValue();
            String each_customer_id = value.get("id");

            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"));
            }
        }
        return_val = objectMapper.writeValueAsString(requested_info);
    } catch (JsonProcessingException e) {
        e.printStackTrace();
        return_val = "{\"status\":\"error\"}";
    }
    return return_val;
}
```

This checks if the logged-in customer's ID (loggedInCustomer.get("id")) matches the requested customer ID (requested_customer_id). If the IDs do not match, it returns an error message stating that the user is not authorized to view the requested data.
Proof of it working can be shown on the next page:

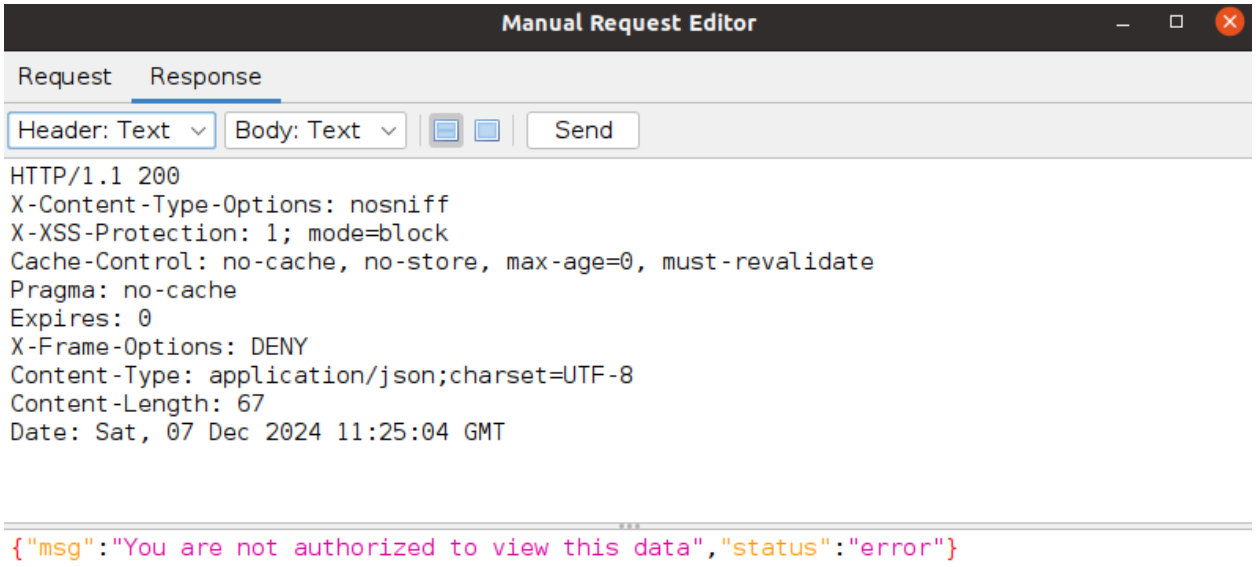
Original GET request body:



Changed GET request body:



Response body after the change:



5.0 Apache Shiro Access Control File

5.1 Apache Shiro Access Control File - Before Fix

Here is the access control code before the fix and the result of running the original code:

```
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing,
# software distributed under the License is distributed on an
# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
# KIND, either express or implied.  See the License for the
# specific language governing permissions and limitations
# under the License.
#
# =============================================================================
# Quickstart INI Realm configuration
#
# For those that might not understand the references in this file, the
# definitions are all based on the classic Mel Brooks' film "Spaceballs". ;)
# =============================================================================

# -----------------------------------------------------------------------------
# Users and their assigned roles
#
# Each line conforms to the format defined in the
# org.apache.shiro.realm.text.TextConfigurationRealm#setUserDefinitions JavaDoc
# -----------------------------------------------------------------------------
[users]
# user 'root' with password 'secret' and the 'admin' role
root = secret, admin
# user 'guest' with the password 'guest' and the 'guest' role
guest = guest, guest
# user 'presidentskroob' with password '12345' ("That's the same combination on
# my luggage!!!" ;)), and role 'president'
presidentskroob = 12345, president
# user 'darkhelmet' with password 'ludicrousspeed' and roles 'darklord' and 'schwartz'
darkhelmet = ludicrousspeed, darklord, schwartz
# user 'lonestarr' with password 'vespa' and roles 'goodguy' and 'schwartz'
lonestarr = vespa, goodguy, schwartz
# user 'billchu' with password 'chu' and the 49er role
billchu = chu, 49er

# -----------------------------------------------------------------------------
# Roles with assigned permissions
#
# Each line conforms to the format defined in the
# org.apache.shiro.realm.text.TextConfigurationRealm#setRoleDefinitions JavaDoc
# -----------------------------------------------------------------------------
[roles]
# 'admin' role has all permissions, indicated by the wildcard '*'
admin = *
# The 'schwartz' role can do anything (*) with any lightsaber:
schwartz = lightsaber:*
# The 'goodguy' role is allowed to 'drive' (action) the winnebago (type) with
# license plate 'eagle5' (instance specific id)
goodguy = winnebago:drive:eagle5 , tesla:drive:tesla1
# create a new role 49er that can do anything with lightsabers and drive all Teslas
49er = lightsaber:*, tesla:drive:*
```

```
itis42215221@itis-4221-5221: ~/Desktop/shiro/samples/quickstart
quickstart ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 2 resources
[INFO] Copying 3 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ samples-quick
start ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ samples-quickstart ---
2024-12-07 08:13:42,444 INFO [org.apache.shiro.session.mgt.AbstractValidatingSes
sionManager] - Enabling session validation scheduler...
2024-12-07 08:13:42,457 INFO [Quickstart] - Retrieved the correct value! [aValue
]
2024-12-07 08:13:42,461 INFO [Quickstart] - User [billchu] logged in successfull
y.
2024-12-07 08:13:42,462 INFO [Quickstart] - Hello, mere mortal.
2024-12-07 08:13:42,463 INFO [Quickstart] - You may use a lightsaber ring.  Use
it wisely.
2024-12-07 08:13:42,463 INFO [Quickstart] - Sorry, you aren't allowed to drive t
he 'eagle5' winnebago!
2024-12-07 08:13:42,463 INFO [Quickstart] - You are permitted to 'drive' the tes
la with license plate (id) 'tesla5'.  Here are the keys - have fun!
itis42215221@itis-4221-5221:~/Desktop/shiro/samples/quickstart$
```

5.2 Apache Shiro Access Control File - After Fix

Here is the access control code and the result of running it after the new roles and permissions were added:

shiro.ini
~/Desktop/shiro/samples/quickstart/target/classes

```
# or more contributor license agreements.  See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing,
# software distributed under the License is distributed on an
# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
# KIND, either express or implied.  See the License for the
# specific language governing permissions and limitations
# under the License.
#
# =============================================================================
# Quickstart INI Realm configuration
#
# For those that might not understand the references in this file, the
# definitions are all based on the classic Mel Brooks' film "Spaceballs". ;)
# =============================================================================

# -----------------------------------------------------------------------------
# Users and their assigned roles
#
# Each line conforms to the format defined in the
# org.apache.shiro.realm.text.TextConfigurationRealm#setUserDefinitions JavaDoc
# -----------------------------------------------------------------------------
[users]
# user 'root' with password 'secret' and the 'admin' role
root = secret, admin
# user 'guest' with the password 'guest' and the 'guest' role
guest = guest, guest
# user 'presidentskroob' with password '12345' ("That's the same combination on
# my luggage!!!" ;)), and role 'president'
presidentskroob = 12345, president
# user 'darkhelmet' with password 'ludicrousspeed' and roles 'darklord' and 'schwartz'
darkhelmet = ludicrousspeed, darklord, schwartz
# user 'lonestarr' with password 'vespa' and roles 'goodguy' and 'schwartz'
lonestarr = vespa, goodguy, schwartz
# user 'billchu' with password 'chu' and the 49er role
billchu = chu, 49er, 49sd

# -----------------------------------------------------------------------------
# Roles with assigned permissions
#
# Each line conforms to the format defined in the
# org.apache.shiro.realm.text.TextConfigurationRealm#setRoleDefinitions JavaDoc
# -----------------------------------------------------------------------------
[roles]
# 'admin' role has all permissions, indicated by the wildcard '*'
admin = *
# The 'schwartz' role can do anything (*) with any lightsaber:
schwartz = lightsaber:*
# The 'goodguy' role is allowed to 'drive' (action) the winnebago (type) with
# license plate 'eagle5' (instance specific id)
goodguy = winnebago:drive:eagle5 , tesla:drive:tesla1
# create a new role 49er that can do anything with lightsabers and drive all Teslas
49er = lightsaber:*, tesla:drive:*
49sd = tesla:drive:*, winnebago:drive:*
```

itis42215221@itis-4221-5221: ~/Desktop/shiro/samples/quickstart

```
quickstart ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 2 resources
[INFO] Copying 3 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ samples-quick
start ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ samples-quickstart ---
2024-12-07 08:25:08,152 INFO [org.apache.shiro.session.mgt.AbstractValidatingSes
sionManager] - Enabling session validation scheduler...
2024-12-07 08:25:08,173 INFO [Quickstart] - Retrieved the correct value! [aValue
]
2024-12-07 08:25:08,173 INFO [Quickstart] - User [billchu] logged in successfull
y.
2024-12-07 08:25:08,174 INFO [Quickstart] - Hello, mere mortal.
2024-12-07 08:25:08,174 INFO [Quickstart] - You may use a lightsaber ring.  Use
it wisely.
2024-12-07 08:25:08,174 INFO [Quickstart] - You are permitted to 'drive' the win
nebago with license plate (id) 'eagle5'.  Here are the keys - have fun!
2024-12-07 08:25:08,175 INFO [Quickstart] - You are permitted to 'drive' the tes
la with license plate (id) 'tesla5'.  Here are the keys - have fun!
itis42215221@itis-4221-5221:~/Desktop/shiro/samples/quickstart$
```