Projekt: Access Control and SSRF

Shashank Mondrati

ITIS: 4221

Project: Access Control SSRF

April , 2022

Table of Contents

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

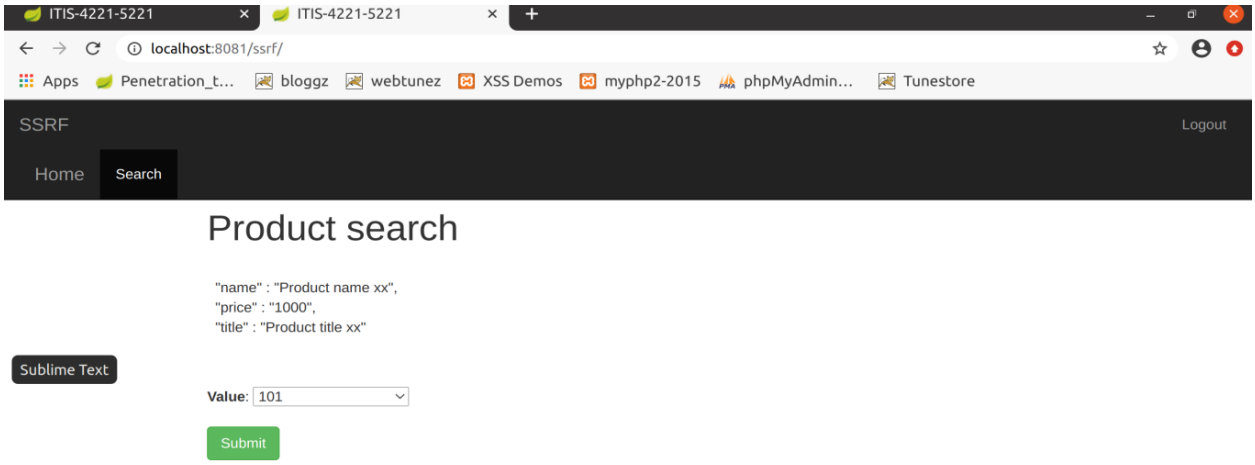
2.0 SSRF Vulnerability

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.

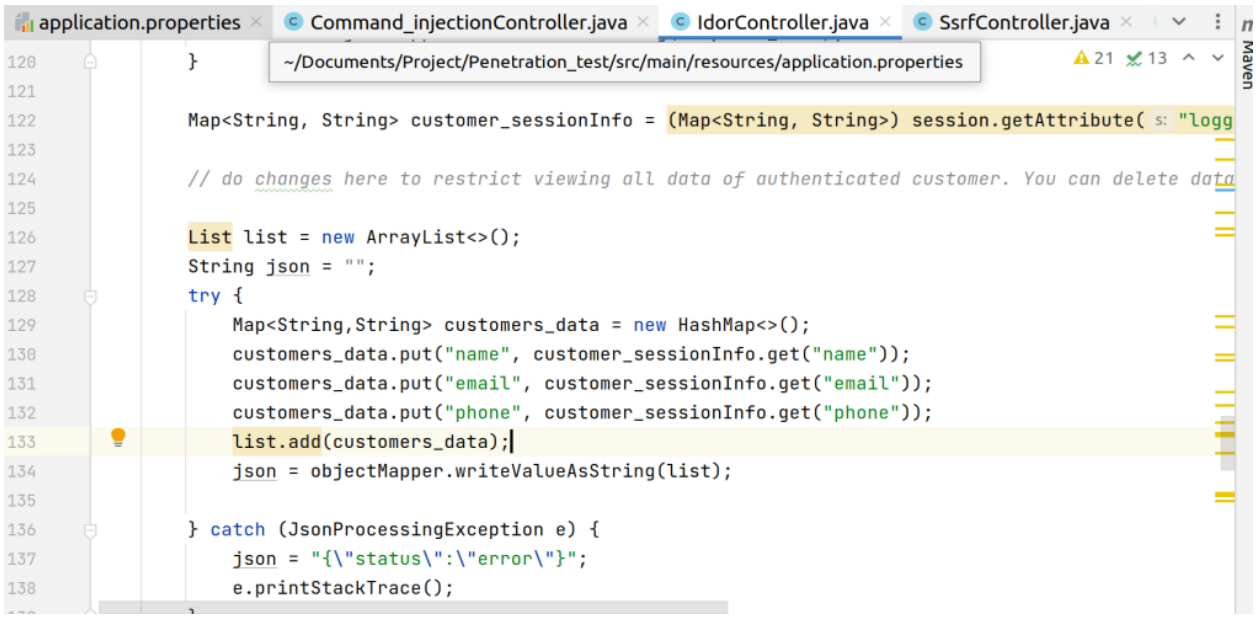
```java
@GetMapping("/")
public String product_search() { return "ssrf/index"; }

@PostMapping("/")
@ResponseBody
public Object product_search_post(@RequestParam String program, @RequestParam String parameter, @RequestParam Str
    Map<String, String> response_data = new HashMap<~>();
    String msg = "";
    try {
        URL obj = new URL( spec: "http://localhost:8081/ssrf/product/" + "item" + "/?" + parameter + "=" + param_va
        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();

            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }
```

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

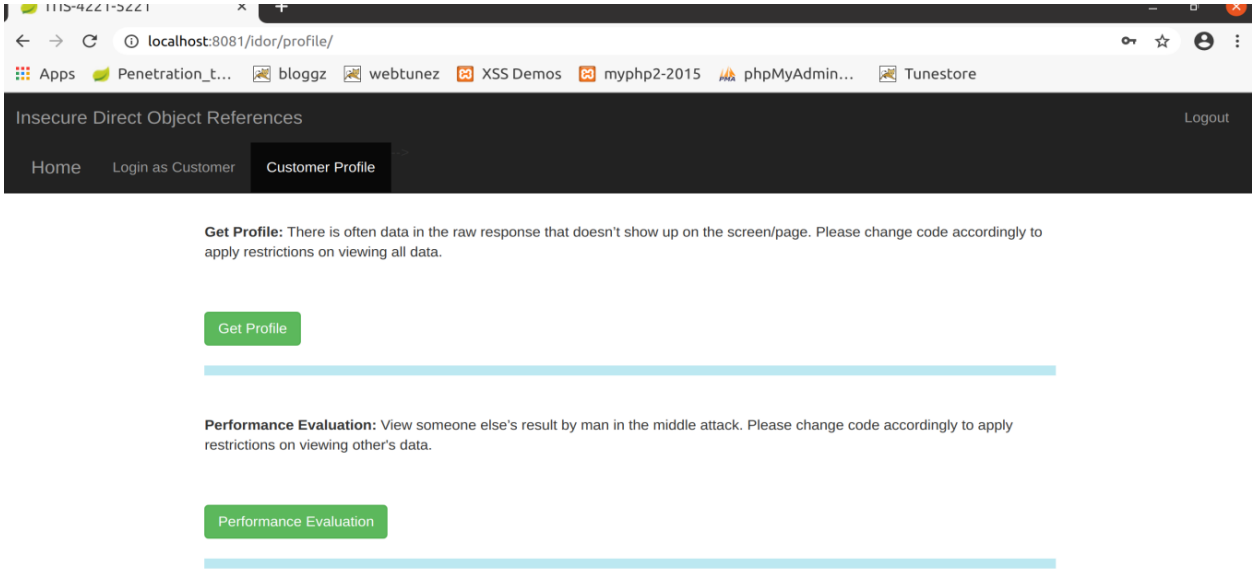


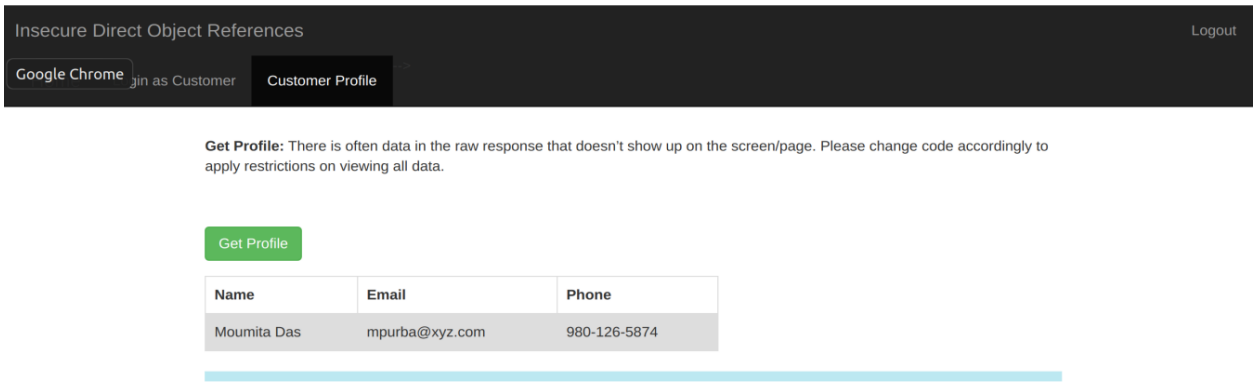
3.0 Information Disclosure Vulnerability

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.



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





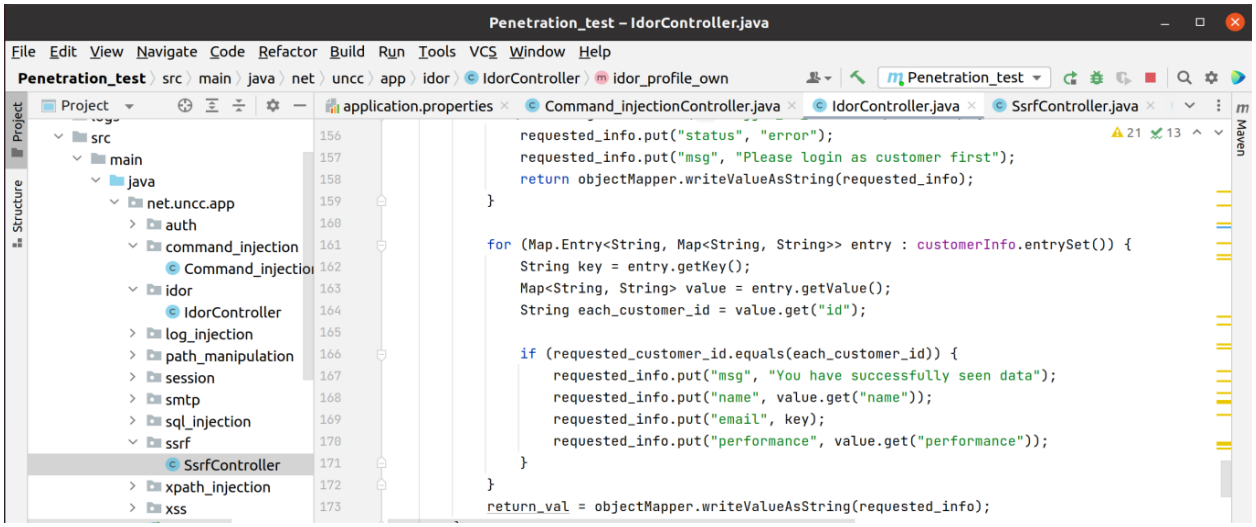
As you can see above the users can see the person who logins name, email and phone number.

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.



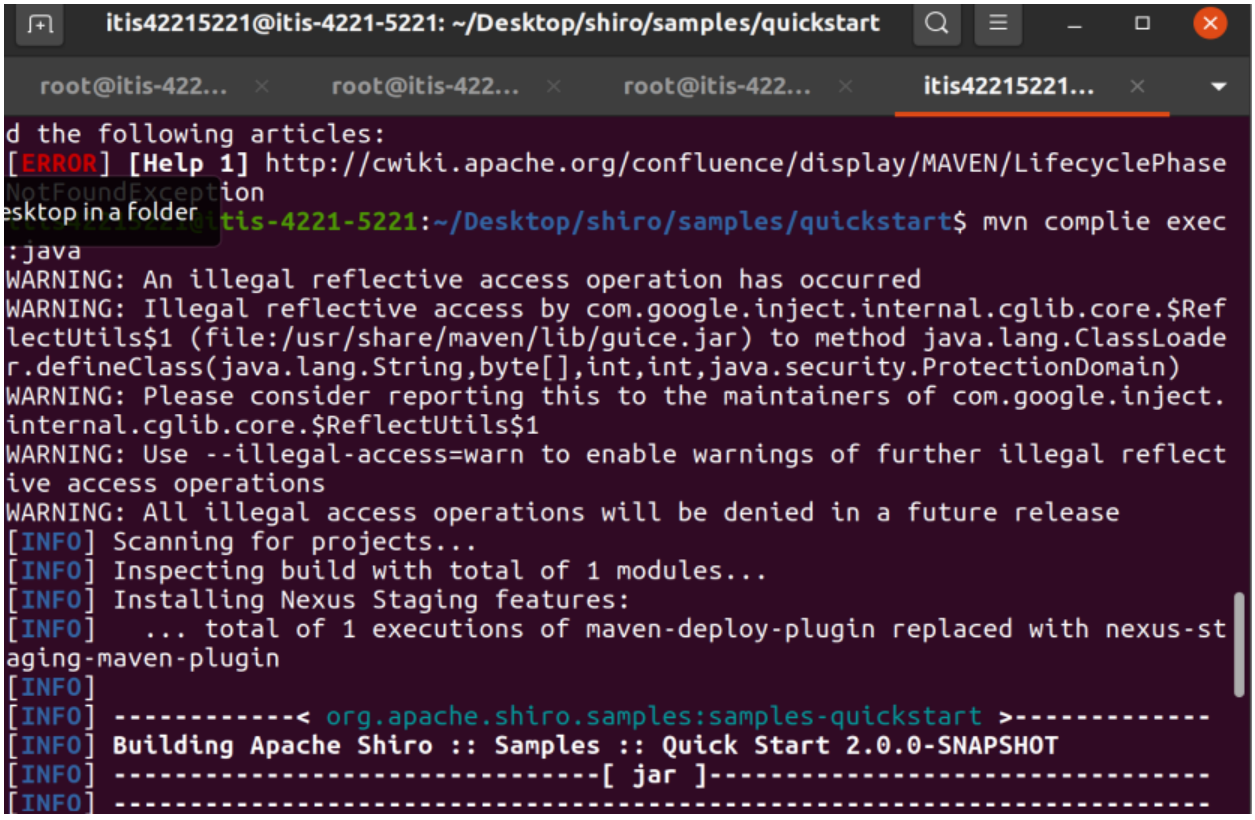
5.0 Access Shiro File (Modified)

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

```
# 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, 49erstudent

# -----------------------------------------------------------------------------
# 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:*, winnebago:drive:*;
49erstudent = *;
```


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.

Below is the screenshot where the exec command in the terminal is executed.


itis42215221@itis-4221-5221: ~/Desktop/shiro/samples/quickstart
root@itis-422...
root@itis-422...
root@itis-422...
itis42215221...
d the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/LifecyclePhase
NotFoundException
esktop in a folder
itis-4221-5221:~/Desktop/shiro/samples/quickstart$ mvn complie exec
:java
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.google.inject.internal.cglib.core.$Ref
lectUtils$1 (file:/usr/share/maven/lib/guice.jar) to method java.lang.ClassLoade
r.defineClass(java.lang.String,byte[],int,int,java.security.ProtectionDomain)
WARNING: Please consider reporting this to the maintainers of com.google.inject.
internal.cglib.core.$ReflectUtils$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflect
ive access operations
WARNING: All illegal access operations will be denied in a future release
[INFO] Scanning for projects...
[INFO] Inspecting build with total of 1 modules...
[INFO] Installing Nexus Staging features:
[INFO]   ... total of 1 executions of maven-deploy-plugin replaced with nexus-st
aging-maven-plugin
[INFO]
[INFO] ------------< org.apache.shiro.samples:samples-quickstart >-------------
[INFO] Building Apache Shiro :: Samples :: Quick Start 2.0.0-SNAPSHOT
[INFO] --------------------------------[jar]---------------------------------
[INFO] --