Assignment 1

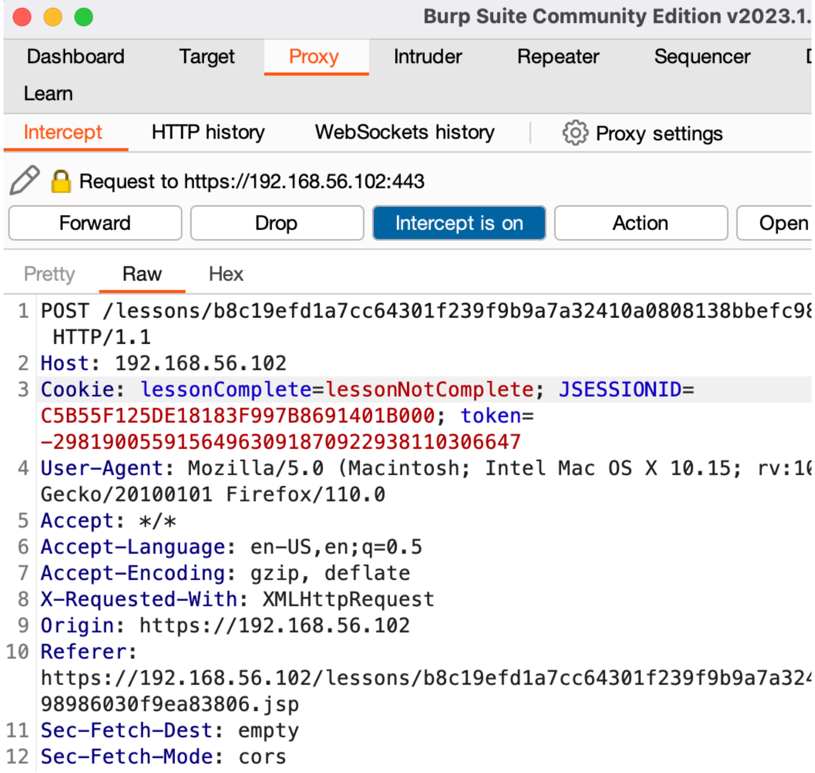
OWASP Security Shepherd Responsible Disclosure Program

High: Broken Authentication and Session Management [CWE-287]

Broken Authentication and Session Management is where authentication and session management has flaws that can be attacked to retrieve other users' session token by guessing their secret questions or through parameter abuse. In this lesson, one can change the session status by abusing parameter.

Steps to reproduce[1]

1. Download and run Burp Suite https://portswigger.net/burp/communitydownload (making sure you have Oracle Java Installed)
2. Utilising Firefox set the system proxy to route traffic through Burp Suite - "Open Menu" button in the right hand corner -> Advanced -> Network (tab) -> Connection "Settings Button" -> Manual proxy configuration. The default for Burp is 127.0.0.1 with a port of 8080
3. Got to Security Shepherd https://192.168.56.102
4. Confirm that Burp can see and capture requests and turn off intercept in Burp
5. Go to Lessons -> Broken Session Management
6. Turn on intercept in Burp
7. Press the "Refresh your Profile" button in the lesson
8. You should see the request caught in Burp



Request to https://192.168.56.102:443

```
POST /lessons/b8c19efd1a7cc64301f239f9b9a7a32410a0808138bbefc98
 HTTP/1.1
Host: 192.168.56.102
Cookie: lessonComplete=lessonNotComplete; JSESSIONID=
C5B55F125DE18183F997B8691401B000; token=
-2981900559156496309187092293811030664 7
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:10
Gecko/20100101 Firefox/110.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Origin: https://192.168.56.102
Referer:
https://192.168.56.102/lessons/b8c19efd1a7cc64301f239f9b9a7a324
98986030f9ea83806.jsp
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
```

9. Modify the value for the parameter "lessonComplete" in the body of the request from "lessonNotComplete" to "lessonComplete"
10. Press forward in Burp
11. Go back to the lesson and you will see that you have completed the lesson

CVSS Score 7.2

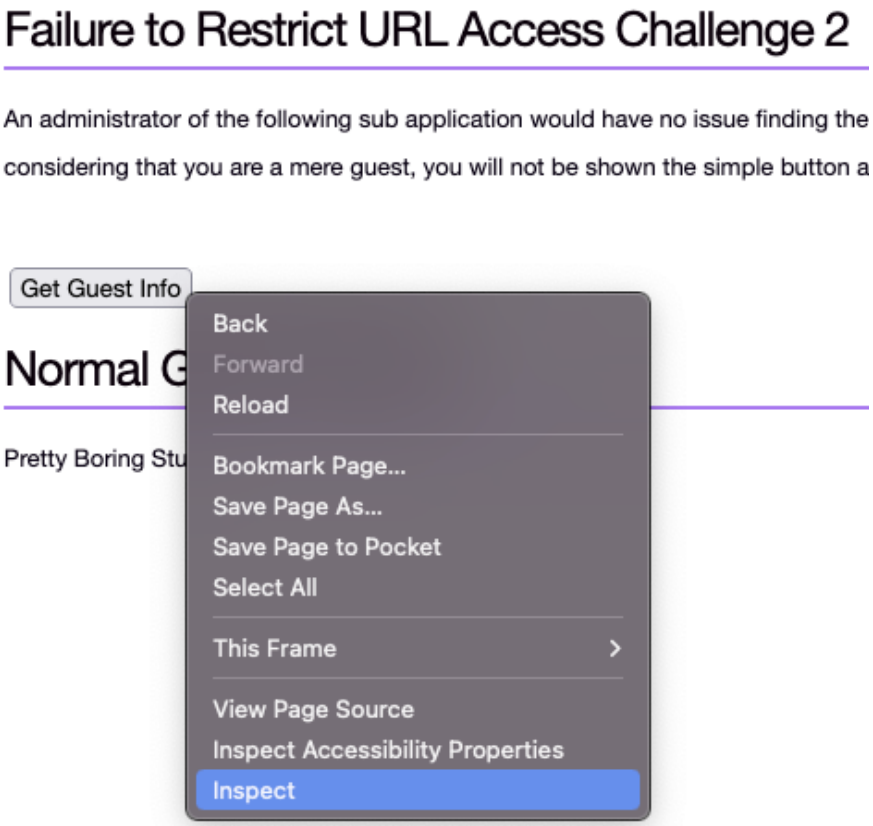
Attack Vector	**Network**
Attack Complexity	**Low**
Privileges Required	**None**
User Interaction	**None**
Scope	**Changed**
Confidentiality	**Low**
Integrity	**Low**
Availability	**None**

High: Failure to Restrict URL Access Challenge 2 [CWE-285]

Failure to restrict URL Access is an application where improper users can gain access to functions that should be hidden from. Under this circumstance, normal users can trigger administers' functions by URL access. In this example, guest users can click on the admin-only button by looking into snippets and gain access to the private key via normal button.

Steps to reproduce[1]

1. Download and run Burp Suite https://portswigger.net/burp/communitydownload (making sure you have Oracle Java Installed)
2. Utilising Firefox set the system proxy to route traffic through Burp Suite - "Open Menu" button in the right hand corner -> Advanced -> Network (tab) -> Connection "Settings Button" -> Manual proxy configuration. The default for Burp is 127.0.0.1 with a port of 8080
3. Got to Security Shepherd https://192.168.56.102
4. Confirm that Burp can see and capture requests and turn off intercept in Burp
5. Go to Challenges -> Failure to Restrict URL Access Challenge 2
6. Right click on button "Get Guest Info", then click "Inspect"



Failure to Restrict URL Access Challenge 2

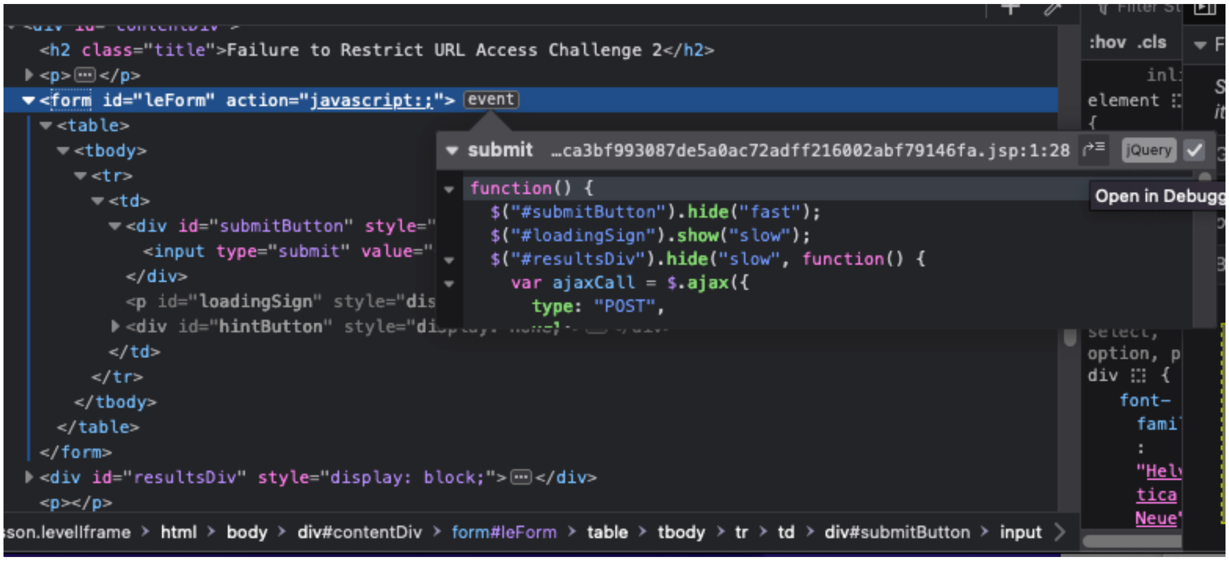
An administrator of the following sub application would have no issue finding the

considering that you are a mere guest, you will not be shown the simple button a

Normal G

Pretty Boring Stu

7. Looking into the source javascript code of the form containing the button

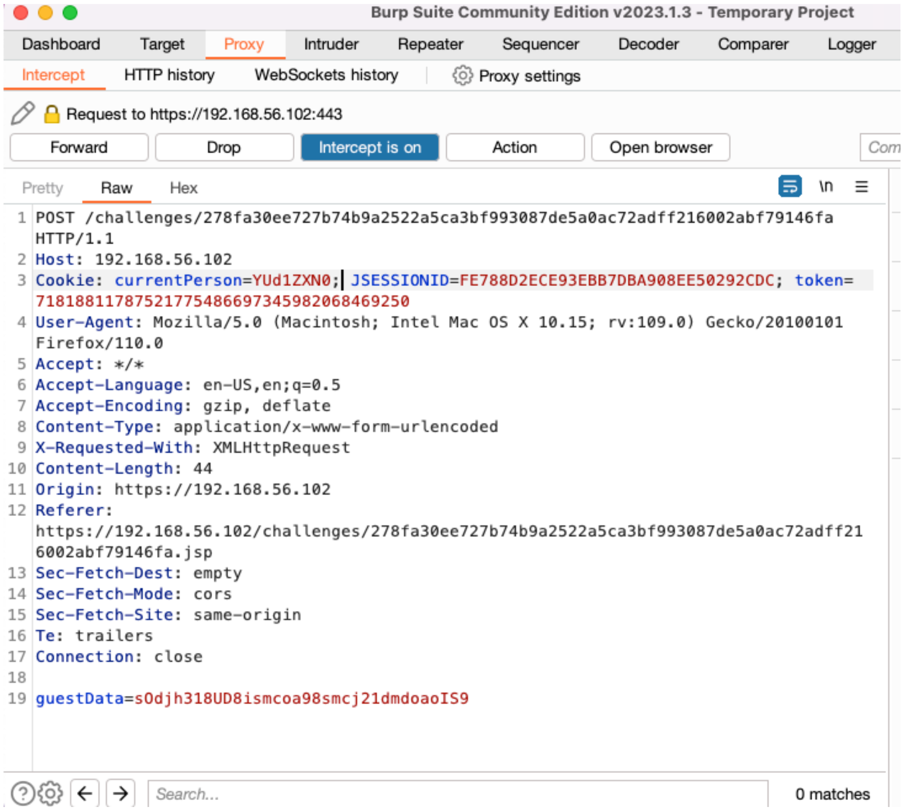


```
<h2 class="title">Failure to Restrict URL Access Challenge 2</h2>
▶ <p>⋯</p>
▼ <form id="leForm" action="javascript:;"> event
  ▼ <table>
    ▼ <tbody>
      ▼ <tr>
        ▼ <td>
          ▼ <div id="submitButton" style="
              <input type="submit" value="
            </div>
            <p id="loadingSign" style="dis
          ▶ <div id="hintButton" style="di
          </td>
        </tr>
      </tbody>
    </table>
  </form>
▶ <div id="resultsDiv" style="display: block;">⋯</div>
  <p></p>
```

```
▼ submit  …ca3bf993087de5a0ac72adff216002abf79146fa.jsp:1:28
function() {
  $("#submitButton").hide("fast");
  $("#loadingSign").show("slow");
  $("#resultsDiv").hide("slow", function() {
    var ajaxCall = $.ajax({
      type: "POST",
```

:hov .cls
jQuery
Open in Debugg
son.levellframe › html › body › div#contentDiv › form#leForm › table › tbody › tr › td › div#submitButton › input

8. You will find that the url and data properties sent from normal user and admin are slightly different
9. Turn on intercept in Burp
10. Click on button "Get Guest Info"
11. You should see the request caught in Burp



```
POST /challenges/278fa30ee727b74b9a2522a5ca3bf993087de5a0ac72adff216002abf79146fa HTTP/1.1
Host: 192.168.56.102
Cookie: currentPerson=YUd1ZXN0; JSESSIONID=FE788D2ECE93EBB7DBA908EE50292CDC; token=71818811787521775486697345982068469250
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/110.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
X-Requested-With: XMLHttpRequest
Content-Length: 44
Origin: https://192.168.56.102
Referer: https://192.168.56.102/challenges/278fa30ee727b74b9a2522a5ca3bf993087de5a0ac72adff216002abf79146fa.jsp
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: close

guestData=sOdjh318UD8ismcoa98smcj21dmdoaoIS9
```

12. Modify the POST url from “/challenges/278fa30ee727b74b9a2522a5ca3bf993087de5a0ac72adff216002abf79146fa” to “/challenges/278fa30ee727b74b9a2522a5ca3bf993087de5a0ac72adff216002abf79146fahghghmin”; the Referer property from “https://192.168.56.102/challenges/278fa30ee727b74b9a2522a5ca3bf993087de5a0ac72adff216002abf79146fa.jsp” to “https://192.168.56.102/challenges/278fa30ee727b74b9a2522a5ca3bf993087de5a0ac72adff216002abf79146fahghghmin.jsp”; Line “guestData=sOdjh318UD8ismcoa98smcj21dmdoaoIS9” to “adminData=youAreAnAdminOfAwesomenessWoopWoop”
13. Press forward in Burp
14. Go back to the challenge and you will see that you have completed the challenge

CVSS Score 8.3

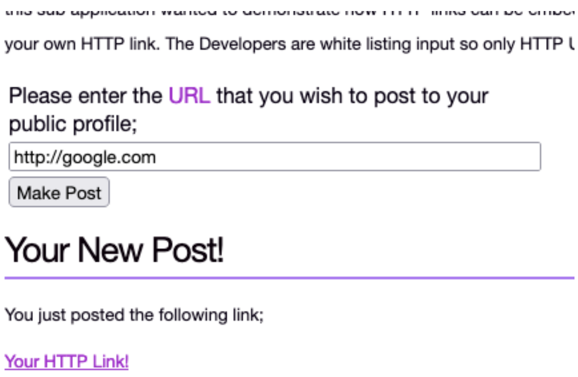
Attack Vector	**Network**
Attack Complexity	**Low**
Privileges Required	**None**
User Interaction	**None**
Scope	**Changed**
Confidentiality	**Low**
Integrity	**Low**
Availability	**Low**

High: Cross Site Scripting Challenge Five [CWE-79]

When an web application fails to validate user input and didn't prevent user input into the web applications, the web was generated dynamically, it would results in malicious script or code being executed on web servers' domain. In this case, the user input was not validated, thus injecting into the generated a tag and ran a cross site alert script.

Steps to reproduce[1]

1. Download and run Burp Suite https://portswigger.net/burp/communitydownload (making sure you have Oracle Java Installed)
2. Utilising Firefox set the system proxy to route traffic through Burp Suite - "Open Menu" button in the right hand corner -> Advanced -> Network (tab) -> Connection "Settings Button" -> Manual proxy configuration. The default for Burp is 127.0.0.1 with a port of 8080
3. Got to Security Shepherd https://192.168.56.102
4. Confirm that Burp can see and capture requests and turn off intercept in Burp
5. Go to Challenges -> Cross Site Scripting Challenge Five
6. First try to input some normal URL, http://google.com, and click "Make Post" button



this sub application wanted to demonstrate how HTTP links can be embe

your own HTTP link. The Developers are white listing input so only HTTP U

Please enter the URL that you wish to post to your public profile;

http://google.com

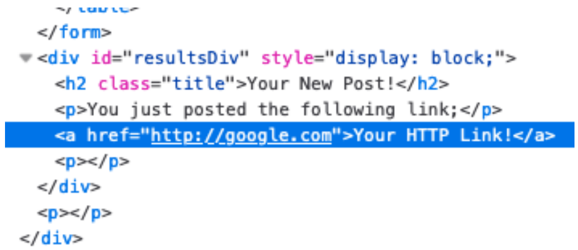
Make Post

Your New Post!

You just posted the following link;

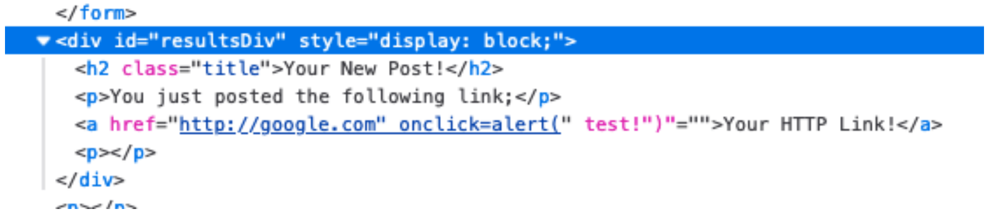
Your HTTP Link!

7. Right click on the generated hyperlink "Your HTTP Link!" and look into the a tag



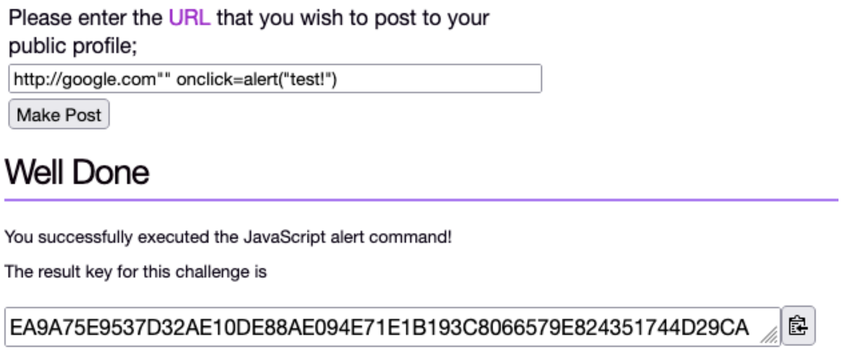
```
  </table>
  </form>
▾ <div id="resultsDiv" style="display: block;">
    <h2 class="title">Your New Post!</h2>
    <p>You just posted the following link;</p>
    <a href="http://google.com">Your HTTP Link!</a>
    <p></p>
  </div>
  <p></p>
</div>
```

8. Input http://google.com" onclick=alert("test!") and repeat step 7



```
  </form>
▾ <div id="resultsDiv" style="display: block;">
    <h2 class="title">Your New Post!</h2>
    <p>You just posted the following link;</p>
    <a href="http://google.com" onclick=alert(" test!")"="">Your HTTP Link!</a>
    <p></p>
  </div>
  <p></p>
```

9. The second " was emitted, so add another " after the second one and try again, this time the challenge was completed



Please enter the URL that you wish to post to your public profile;

http://google.com"" onclick=alert("test!")

Well Done

You successfully executed the JavaScript alert command!

The result key for this challenge is

EA9A75E9537D32AE10DE88AE094E71E1B193C8066579E824351744D29CA

CVSS Score 7.9

Attack Vector	**Network**
Attack Complexity	**High**
Privileges Required	**Low**
User Interaction	**Required**
Scope	**Changed**
Confidentiality	**High**
Integrity	**High**
Availability	**Low**

Medium: Insecure Cryptographic Storage Challenge 1 [CWE-326]

The application contains under encrypted data or non-encrypted sensitive information. Attackers can easily decrypt the sensitive information using brute force. In this case, Roman cipher is a simple encryption and is very vulnerable.

Steps to reproduce[1]

1. Got to Security Shepherd https://192.168.56.102
2. Go to Challenges -> SQL Injection Challenge Two
3. Open another tab in the browser, go to https://cryptii.com/pipes/caesar-cipher, copy the encrypted text "Ymj wjxzqy pjd ktw ymnx qjxxts nx ymj ktqqtbnsl xywnsl; rdqtajqdmtwxjwzssnslymwtzlmymjknjqibmjwjfwjdtzltnslbnymdtzwgnlf" into the Roman cipher decoder
4. Brute force shift value from 1 through 25 and look at the decoded text, you should find that when shift equals 5, the decoded text are plain easy English, the challenge is completed

CVSS Score 6.5

Attack Vector	**Network**
Attack Complexity	**Low**
Privileges Required	**None**
User Interaction	**None**
Scope	**Unchanged**
Confidentiality	**Low**
Integrity	**None**
Availability	**Low**

Reference

[1] The OWASP Security Shepherd team. High: Insecure Direct Object Reference in Lessons [CWE-639].