

Web Security Activity: Part 2

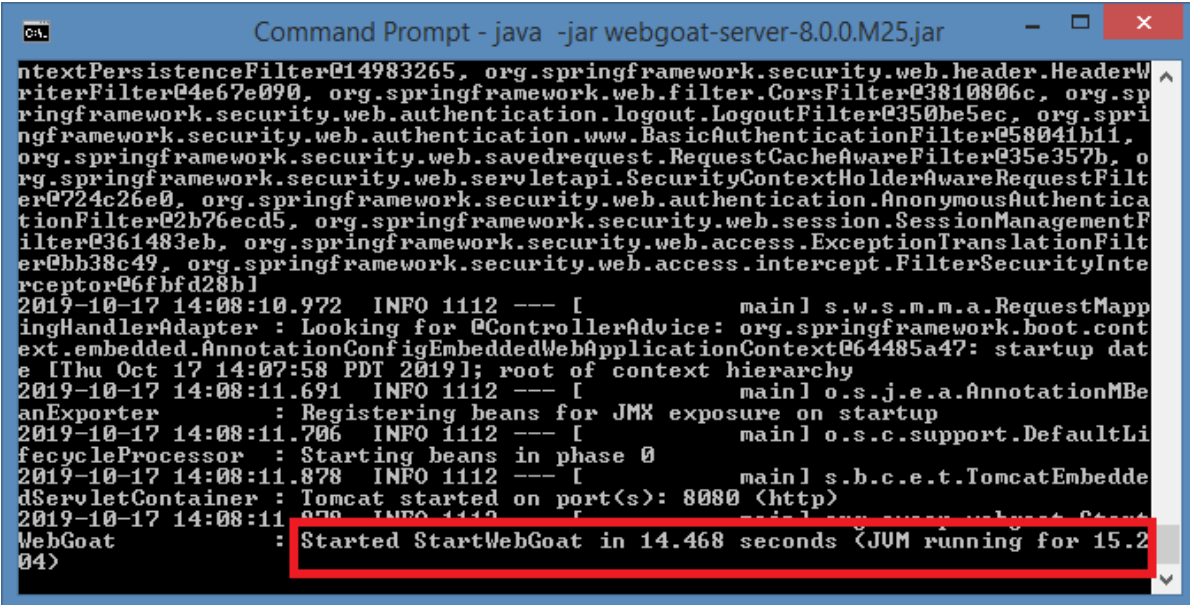
Part 2: individual work

In this part, you will be able to describe an XSS attack and the risks it poses to an organization. You will also be able to do a basic XSS evaluation of web applications. (If you are doing this part right after part 1, you can skip next step 1, 2, and 3 in this document)

1. Open 'Command Prompt' and navigate to the folder where you downloaded the WebGoat jar file (hint, type the command: 'cd Downloads'). Then type the following command to start WebGoat server:

```
java -jar webgoat-server-8.0.0.M25.jar
```

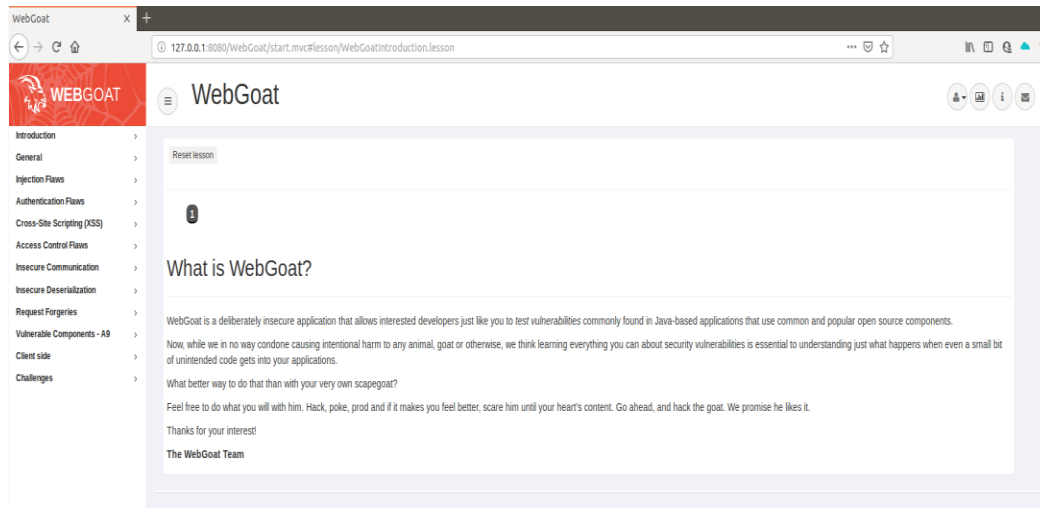
Wait until the server is up and running (You will see the following message)



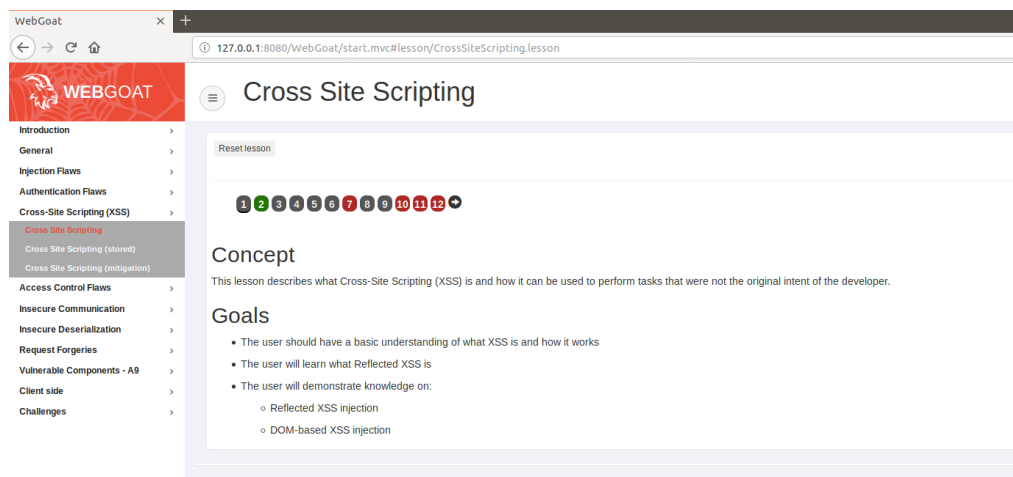
```
ntextPersistenceFilter@14983265, org.springframework.security.web.header.HeaderW
riterFilter@4e67e090, org.springframework.web.filter.CorsFilter@3810806c, org.sp
ringframework.security.web.authentication.logout.LogoutFilter@350be5ec, org.spri
ngframework.security.web.authentication.www.BasicAuthenticationFilter@58041b11,
org.springframework.security.web.savedrequest.RequestCacheAwareFilter@35e357b, o
rg.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilt
er@724c26e0, org.springframework.security.web.authentication.AnonymousAuthentic
ationFilter@2b76ecd5, org.springframework.security.web.session.SessionManagementF
ilter@361483eb, org.springframework.security.web.access.ExceptionTranslationFilt
er@bb38c49, org.springframework.security.web.access.intercept.FilterSecurityInte
rceptor@6fbfd28b]
2019-10-17 14:08:10.972 INFO 1112 --- [          main] s.w.s.m.m.a.RequestMapp
ingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.cont
ext.embedded.AnnotationConfigEmbeddedWebApplicationContext@64485a47: startup dat
e [Thu Oct 17 14:07:58 PDT 2019]; root of context hierarchy
2019-10-17 14:08:11.691 INFO 1112 --- [          main] o.s.j.e.a.AnnotationMBe
anExporter       : Registering beans for JMX exposure on startup
2019-10-17 14:08:11.706 INFO 1112 --- [          main] o.s.c.support.DefaultLi
fecycleProcessor : Starting beans in phase 0
2019-10-17 14:08:11.878 INFO 1112 --- [          main] s.b.c.e.t.TomcatEmbedde
dServletContainer : Tomcat started on port(s): 8080 (http)
2019-10-17 14:08:11.878 INFO 1112 --- [          main] s.b.c.e.t.TomcatEmbedde
dServletContainer : Tomcat started on port(s): 8080 (http)
WebGoat         : Started StartWebGoat in 14.468 seconds (JVM running for 15.2
04)
```

2. Open a browser window and type 'http://localhost:8080/WebGoat/' to start WebGoat in the browser.
3. Use the credentials that you created in Part 1 to login to the "home page" of WebGoat. Once you log in you will land on WebGoat's home page.

Web-Based Application Design and Development (ITIS 3135)



4. Navigate to Cross-Site Scripting (XSS) → “Cross-Site Scripting” to get on the following page.



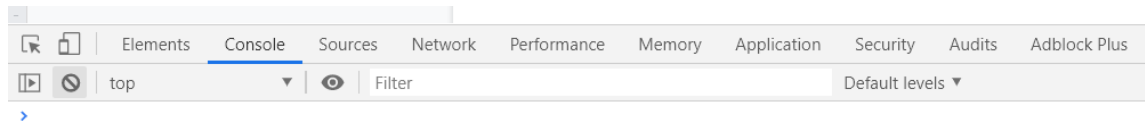
The numbers 1 to 12 indicate the lesson number. The lessons marked in red contain challenges. Once you complete a challenge the lesson number changes color to green. The lessons without challenges contain important concepts that will help you in solving the challenges. It is advisable to go through the lessons in order to have a better understanding to solve the challenges. **We will go up to lesson 7 in this section.**

5. To start, complete Lessons 1-6 including the trivial challenge in Lesson 2. For Lesson 2, instead of using the address bar to execute the JavaScript, use the Console in the browser developer tools (access instructions provided below).

Instructions to access the browser developer tools:

- If you are using Google Chrome or Mozilla Firefox, pressing Ctrl+Shift+I (on Windows) will open the browser developer tools.

Web-Based Application Design and Development (ITIS 3135)



Take a screenshot (Screenshot 2) of the message received after completing the challenge in Lesson 2 and save it in the YourLastName_WebSecurityActivity.docx indicating the challenge completion. [1 point]

- After completing Lessons 1-6, proceed to Lesson 7 that will test your ability to identify reflected XSS. You should see the following form once you navigate to Lesson 7.

Try It! Reflected XSS

Identify which field is susceptible to XSS

It is always a good practice to validate all input on the server side. XSS can occur when unvalidated user input is used in an HTTP response. In a reflected XSS attack, an attacker can craft a URL with the attack script and post it to another website, email it, or otherwise get a victim to click on it.

An easy way to find out if a field is vulnerable to an XSS attack is to use the `alert()` or `console.log()` methods. Use one of them to find out which field is vulnerable.

Shopping Cart

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry	69.99	<input type="text" value="1"/>	\$0.00
Dynex - Traditional Notebook Case	27.99	<input type="text" value="1"/>	\$0.00
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	<input type="text" value="1"/>	\$0.00
3 - Year Performance Service Plan \$1000 and Over	299.99	<input type="text" value="1"/>	\$0.00

The total charged to your credit card: \$0.00

Enter your credit card number:

Enter your three digit access code:

Here, the goal is to identify which field is susceptible to XSS. An easy way to find out if a field is vulnerable to an XSS attack is to use the `alert()` method. To start, click the 'Purchase' button and observe what happens. This will help you in identifying the text field that is susceptible to the XSS attack. After identifying the text field use the following input to trigger the XSS vulnerability:

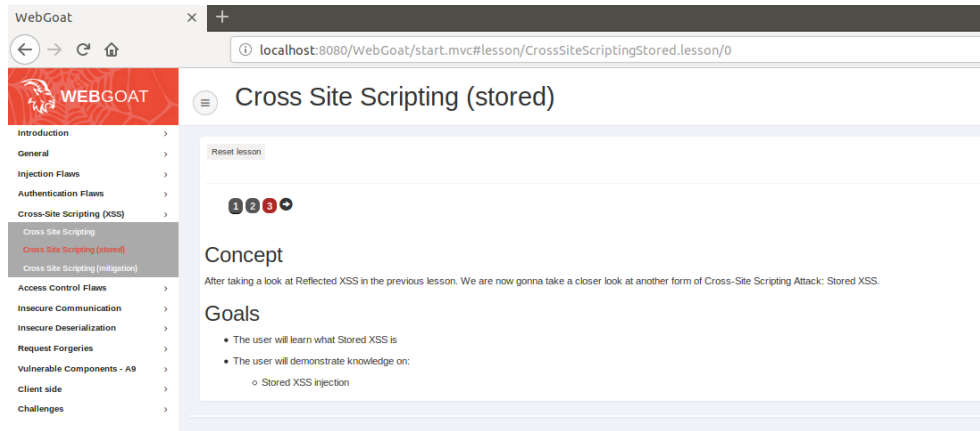
```
<script>alert('xss')</script>
```

Take a screenshot (Screenshot 3) of the alert message received after completing the challenge in Lesson 7 and save it in the YourLastName_WebSecurityActivity.docx indicating the challenge completion. [1 point]

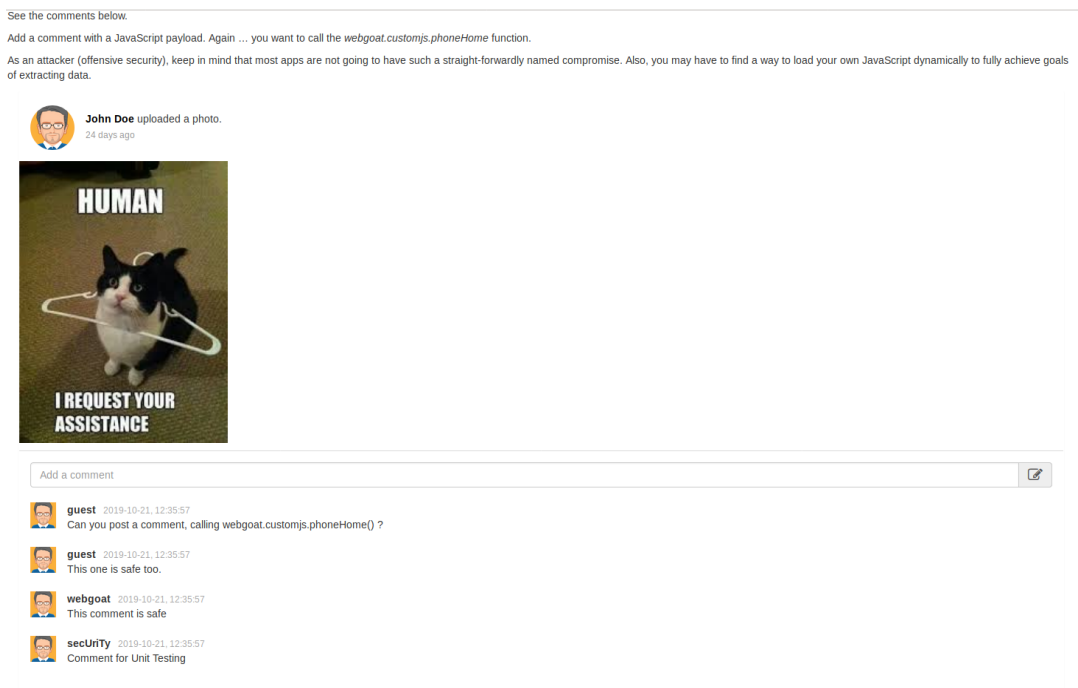
Q1: Explain your observation and understanding of the vulnerability of this challenge in the in the YourLastName_WebSecurityActivity.docx – **Reflected XSS. [3 points]**

Web-Based Application Design and Development (ITIS 3135)

7. Next, navigate to “Cross-Site Scripting (stored)” to navigate to the following page:



8. Complete Lessons 1 and 2 and the challenge in Lesson 3 (pictured below).



For this challenge, add a comment that contains JavaScript to call the `webgoat.customjs.phoneHome()` function.

Use the following input to trigger the stored XSS vulnerability:

```
<script>webgoat.customjs.phoneHome()</script>
```

The output of the script execution can be seen using the browser developer tools (access instructions provided below). An `'alert(webgoat.cusotmjs.phoneHome())'` would give an undefined output as the `phoneHome()` function is defined to output the result using

Web-Based Application Design and Development (ITIS 3135)

the `console.log()` DOM API instead of `alert()`. The output should include a value starting with 'phoneHome Response is'. Put that value in submission text box at the end of the lesson to complete this challenge. Note that, each subsequent call to the `phoneHome()` method will change that value. You may need to ensure you have the most recent one.

The response of the script execution can be seen using the same 'Console' tab in browser developer tools used in Lesson 2. The response will be similar to the screenshot below.

```
phoneHome invoked  
phone home said {"lessonCompleted":true,"feedback":"Congratulations. You have successfully completed the assignment.","output":"phoneHome Response is 556779667"}
```

Take a screenshot (Screenshot 4) of the message received after completing the challenge in Lesson 3 in the YourLastName_WebSecurityActivity.docx indicating the challenge completion. [1 point]

Q2: Explain your observation and understanding of the vulnerability of this challenge in the YourLastName_WebSecurityActivity.docx – **Stored XSS. [3 points]**

9. Answer the following questions in the YourLastName_WebSecurityActivity.docx:

Q3: What are some measures that can be taken to block a stored XSS attack (List 3)? **[3 points]**

Hint: search for 'preventing XSS' and/or use the resources provided in WebGoat itself under the section---Cross-Site Scripting (XSS) → Cross-Site Scripting (mitigation).

Q4: What information can the attacker steal using XSS attacks (List 3). **[3 points]**

Q5: What are some common locations on a web page that can be vulnerable to an XSS attack (List 3). **[3 points]**

Q6: What are some of the consequences of an XSS attack (List 3). **[3 points]**

10. Submission: (31 Points Total including Part 1 and 2)

Record your findings (screenshots + answers to Questions 1- 6) and upload your **YourLastName_WebSecurityActivity.docx** as a pdf in the Web Security Activity submission link on Canvas. One submission per student.

Note: Follow the instructions closely, and organize your answers neatly. Please label your answers with the appropriate question labeling. For example "Screenshot 1, Answer to Q1, etc.". Illegible, unclear answers or answers that do not adhere to instructions will be penalized. Only one submission per student.