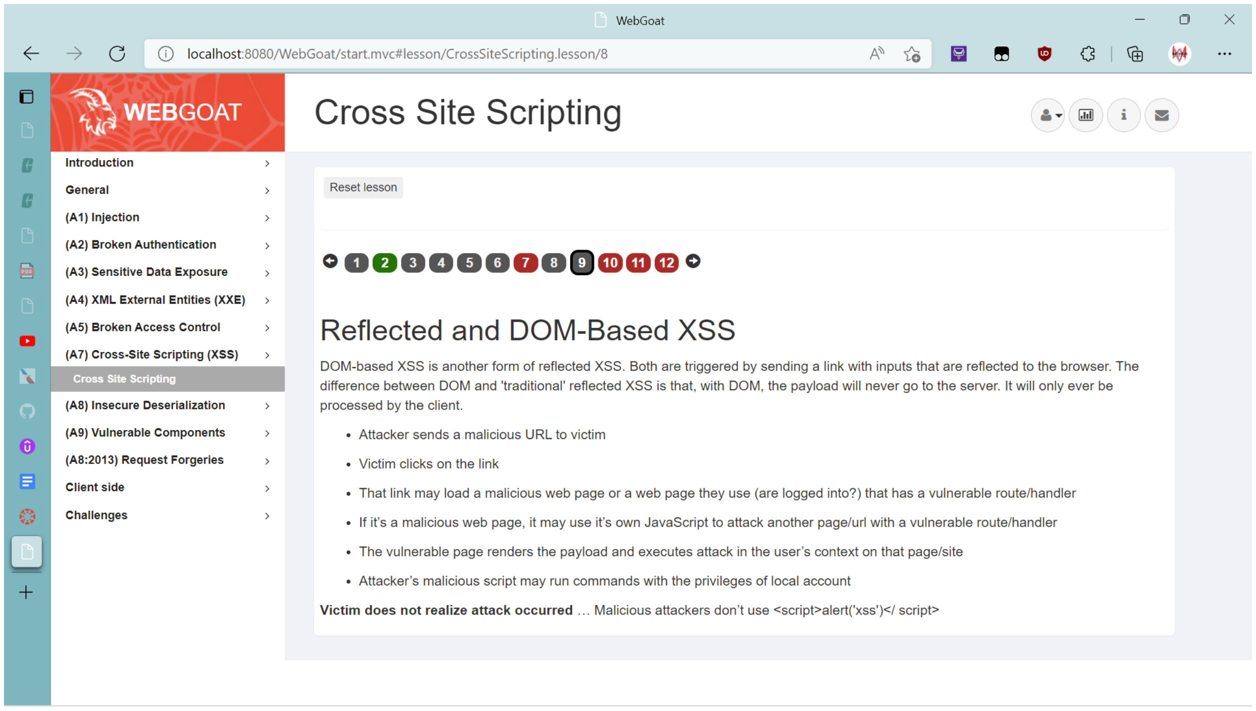
Shashank Mondrati
Connor Caroll
Dominik Gonzales
Will Ngyuyen

Activity 6.2

Screenshots


WebGoat
localhost:8080/WebGoat/start.mvc#lesson/CrossSiteScripting.lesson/8
WEBGOAT
Introduction
General
(A1) Injection
(A2) Broken Authentication
(A3) Sensitive Data Exposure
(A4) XML External Entities (XXE)
(A5) Broken Access Control
(A7) Cross-Site Scripting (XSS)
Cross Site Scripting
(A8) Insecure Deserialization
(A9) Vulnerable Components
(A8:2013) Request Forgeries
Client side
Challenges
Reset lesson
1 2 3 4 5 6 7 8 9 10 11 12

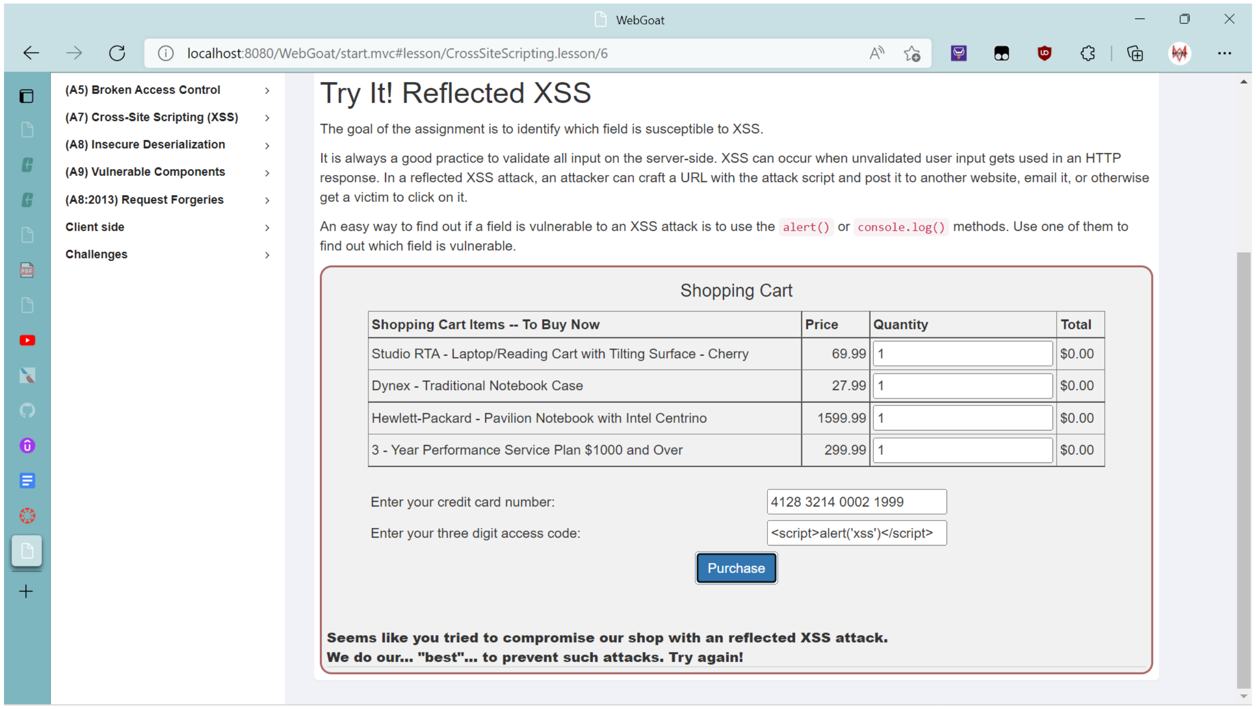
Cross Site Scripting

Reflected and DOM-Based XSS

DOM-based XSS is another form of reflected XSS. Both are triggered by sending a link with inputs that are reflected to the browser. The difference between DOM and 'traditional' reflected XSS is that, with DOM, the payload will never go to the server. It will only ever be processed by the client.

- Attacker sends a malicious URL to victim
- Victim clicks on the link
- That link may load a malicious web page or a web page they use (are logged into?) that has a vulnerable route/handler
- If it's a malicious web page, it may use it's own JavaScript to attack another page/url with a vulnerable route/handler
- The vulnerable page renders the payload and executes attack in the user's context on that page/site
- Attacker's malicious script may run commands with the privileges of local account

Victim does not realize attack occurred … Malicious attackers don't use <script>alert('xss')</ script>


localhost:8080/WebGoat/start.mvc#lesson/CrossSiteScripting.lesson/6

(A5) Broken Access Control
(A7) Cross-Site Scripting (XSS)
(A8) Insecure Deserialization
(A9) Vulnerable Components
(A8:2013) Request Forgeries
Client side
Challenges

Try It! Reflected XSS

The goal of the assignment is to 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 gets 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.

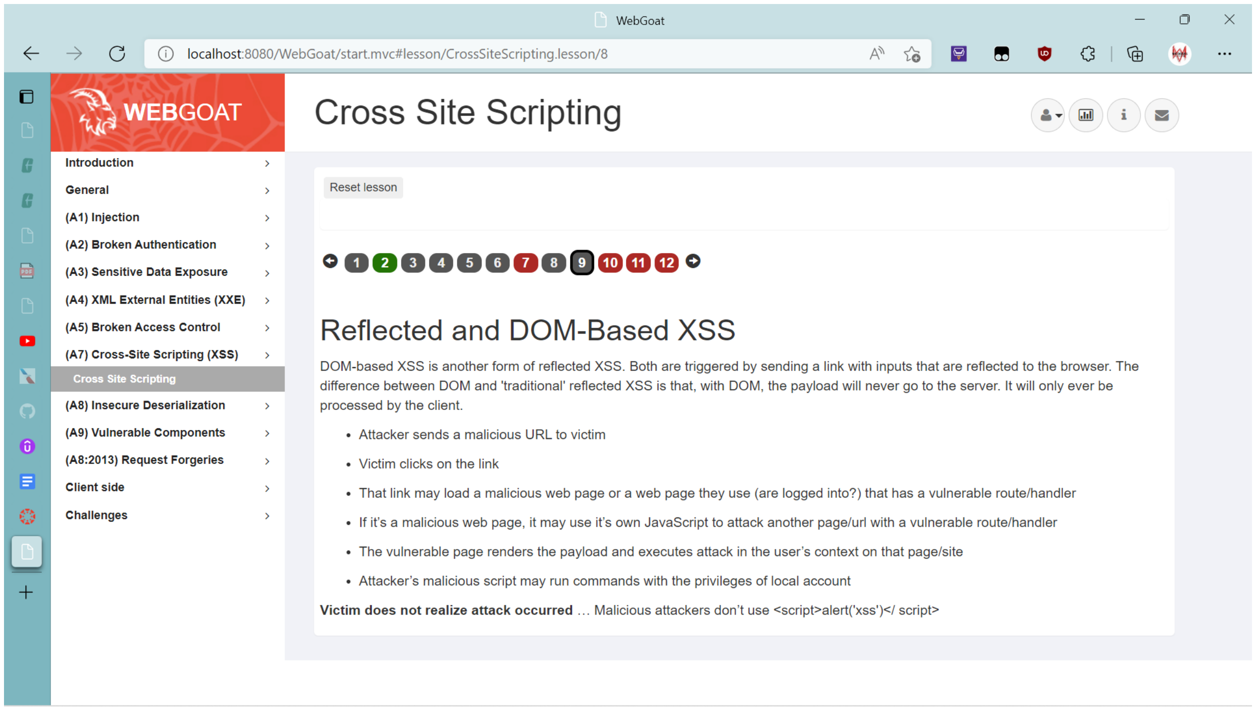
Shopping Cart

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

Enter your credit card number:	4128 3214 0002 1999
Enter your three digit access code:	<script>alert('xss')</script>

Purchase

Seems like you tried to compromise our shop with an reflected XSS attack.
We do our... "best"... to prevent such attacks. Try again!


localhost:8080/WebGoat/start.mvc#lesson/CrossSiteScripting.lesson/8

WEBGOAT
Introduction
General
(A1) Injection
(A2) Broken Authentication
(A3) Sensitive Data Exposure
(A4) XML External Entities (XXE)
(A5) Broken Access Control
(A7) Cross-Site Scripting (XSS)
Cross Site Scripting
(A8) Insecure Deserialization
(A9) Vulnerable Components
(A8:2013) Request Forgeries
Client side
Challenges
Reset lesson
1 2 3 4 5 6 7 8 9 10 11 12

Cross Site Scripting

Reflected and DOM-Based XSS

DOM-based XSS is another form of reflected XSS. Both are triggered by sending a link with inputs that are reflected to the browser. The difference between DOM and 'traditional' reflected XSS is that, with DOM, the payload will never go to the server. It will only ever be processed by the client.

- Attacker sends a malicious URL to victim
- Victim clicks on the link
- That link may load a malicious web page or a web page they use (are logged into?) that has a vulnerable route/handler
- If it's a malicious web page, it may use it's own JavaScript to attack another page/url with a vulnerable route/handler
- The vulnerable page renders the payload and executes attack in the user's context on that page/site
- Attacker's malicious script may run commands with the privileges of local account

Victim does not realize attack occurred … Malicious attackers don't use <script>alert('xss')</ script>

And as the above screenshot shows I cannot see Cross Site Scripting (Stored) menu in my webgoat

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

From what we see is when you enter a XSS command the webgoat recognized it as an attack command and countered it with the message saying it was recognized.

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

- Filter input on arrival. At the point where user input is received, filter as strictly as possible based on what is expected or valid input.
- Encode data on output. At the point where user-controllable data is output in HTTP responses, encode the output to prevent it from being interpreted as active content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding.
- Use appropriate response headers. To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, you can use the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend.

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

- Session tokens, user credentials or commercially valuable data, as well as to perform sensitive operations

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

- Scripting attacks are forums, message boards, and web pages that allow comments.

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

- Code injected into a vulnerable application can exfiltrate data or install malware on the user's machine. Attackers can masquerade as authorized users via session cookies, allowing them to perform any action allowed by the user account.