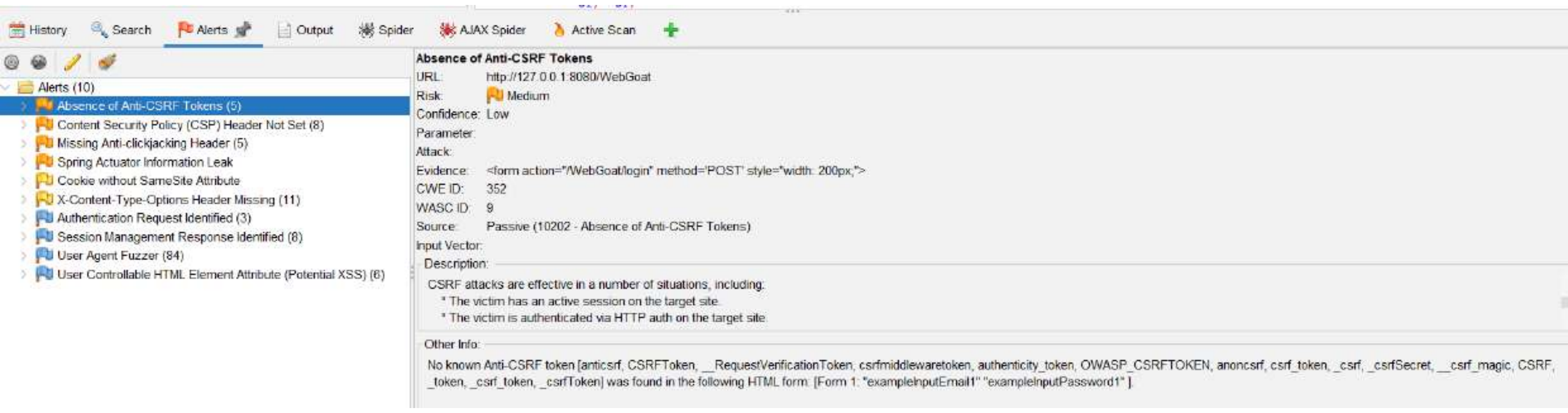


Alerts (10)
- Absence of Anti-CSRF Tokens (5)
- Content Security Policy (CSP) Header Not Set (8)
- Missing Anti-clickjacking Header (5)
- Spring Actuator Information Leak
- Cookie without SameSite Attribute
- X-Content-Type-Options Header Missing (11)
- Authentication Request Identified (3)
- Session Management Response Identified (8)
- User Agent Fuzzer (84)
- User Controllable HTML Element Attribute (Potential XSS) (6)

Absence of Anti-CSRF Tokens

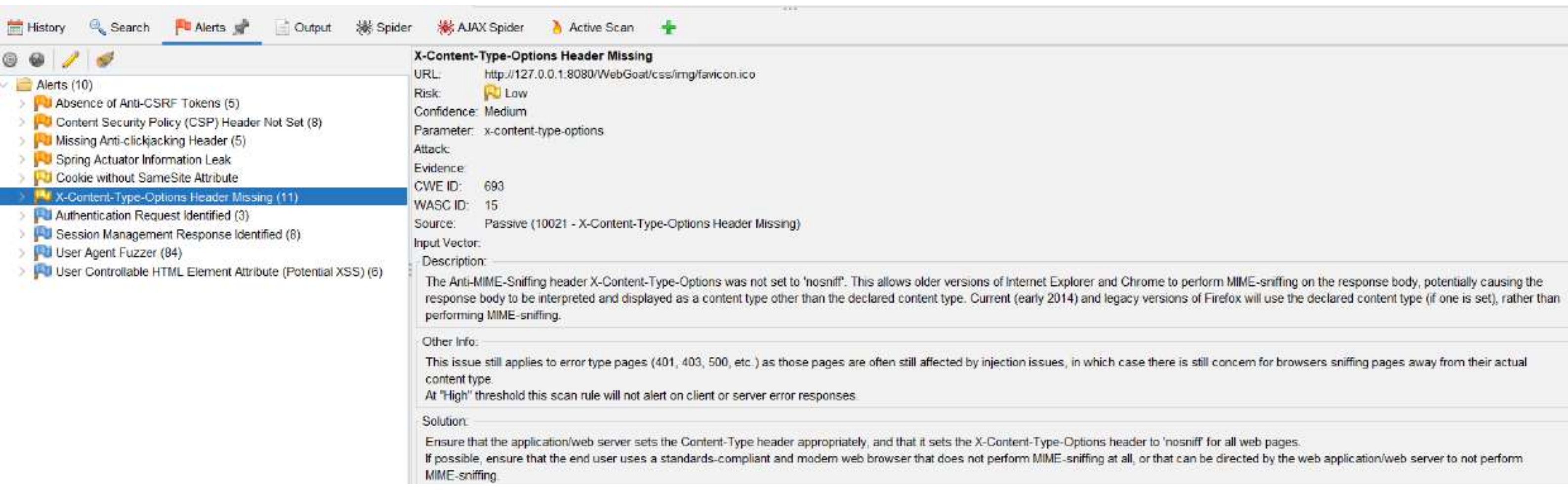
URL:	http://127.0.0.1:8080/WebGoat
Risk:	Medium
Confidence:	Low
Parameter:	
Attack:	
Evidence:	<form action="/WebGoat/login" method='POST' style="width: 200px;">
CWE ID:	352
WASC ID:	9
Source:	Passive (10202 - Absence of Anti-CSRF Tokens)
Input Vector:	

Description:

CSRF attacks are effective in a number of situations, including:
* The victim has an active session on the target site.
* The victim is authenticated via HTTP auth on the target site.

Other Info:

No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token, _csrfToken] was found in the following HTML form: [Form 1: "exampleInputEmail1" "exampleInputPassword1"].



Alerts (10)
- Absence of Anti-CSRF Tokens (5)
- Content Security Policy (CSP) Header Not Set (8)
- Missing Anti-clickjacking Header (5)
- Spring Actuator Information Leak
- Cookie without SameSite Attribute
- X-Content-Type-Options Header Missing (11)
- Authentication Request Identified (3)
- Session Management Response Identified (8)
- User Agent Fuzzer (84)
- User Controllable HTML Element Attribute (Potential XSS) (6)

X-Content-Type-Options Header Missing

URL:	http://127.0.0.1:8080/WebGoat/css/img/favicon.ico
Risk:	Low
Confidence:	Medium
Parameter:	x-content-type-options
Attack:	
Evidence:	
CWE ID:	693
WASC ID:	15
Source:	Passive (10021 - X-Content-Type-Options Header Missing)
Input Vector:	

Description:

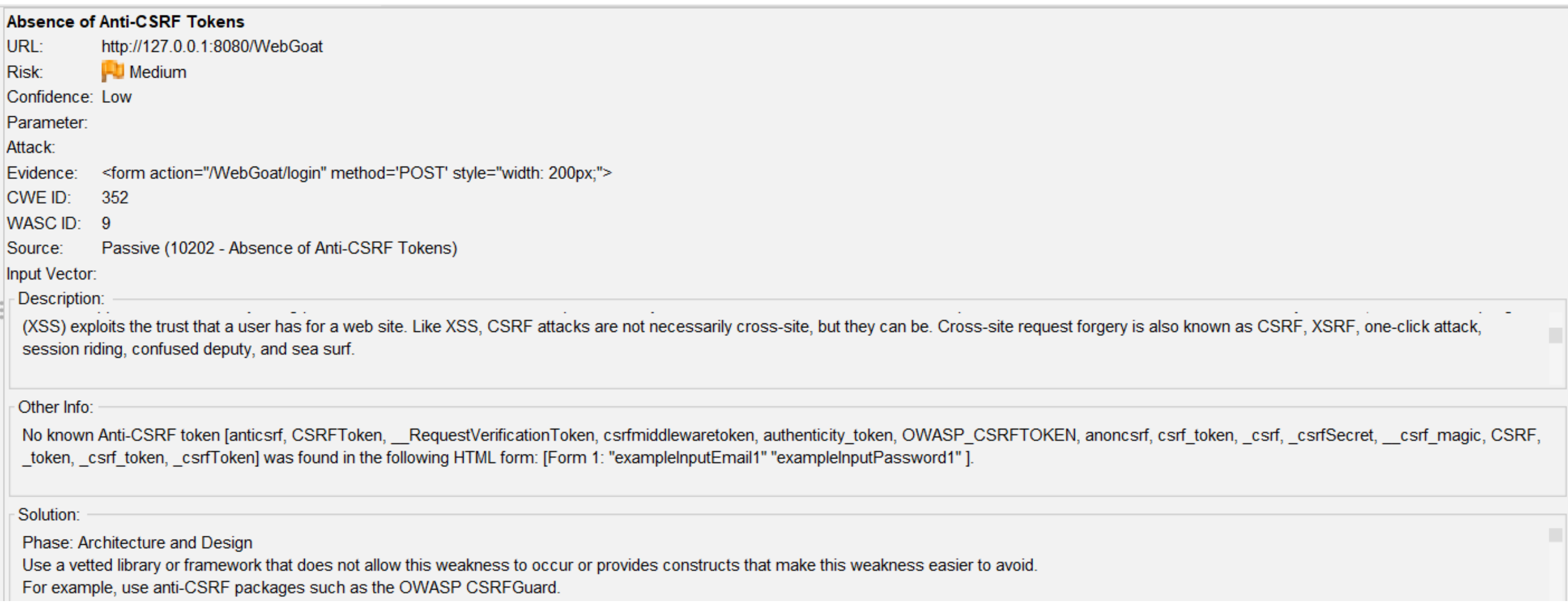
The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

Other Info:

This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type.
At "High" threshold this scan rule will not alert on client or server error responses.

Solution:

Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages.
If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.



Absence of Anti-CSRF Tokens

URL:	http://127.0.0.1:8080/WebGoat
Risk:	Medium
Confidence:	Low
Parameter:	
Attack:	
Evidence:	<form action="/WebGoat/login" method='POST' style="width: 200px;">
CWE ID:	352
WASC ID:	9
Source:	Passive (10202 - Absence of Anti-CSRF Tokens)
Input Vector:	

Description:

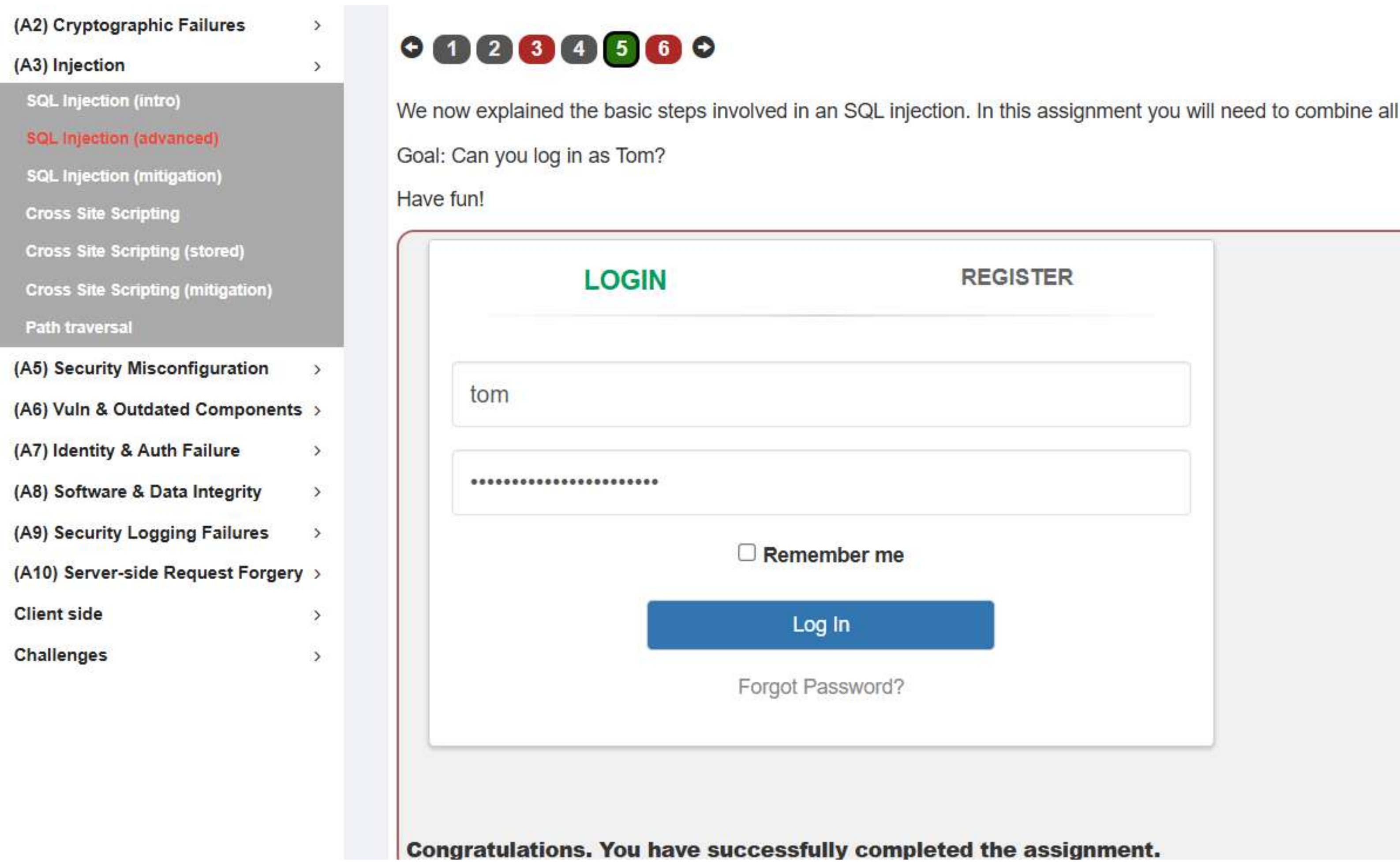
(XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.

Other Info:

No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token, _csrfToken] was found in the following HTML form: [Form 1: "exampleInputEmail1" "exampleInputPassword1"].

Solution:

Phase: Architecture and Design
Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.
For example, use anti-CSRF packages such as the OWASP CSRFGuard.


(A2) Cryptographic Failures
(A3) Injection
SQL Injection (intro)
SQL Injection (advanced)
SQL Injection (mitigation)
Cross Site Scripting
Cross Site Scripting (stored)
Cross Site Scripting (mitigation)
Path traversal
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
1
2
3
4
5
6

We now explained the basic steps involved in an SQL injection. In this assignment you will need to combine all

Goal: Can you log in as Tom?

Have fun!

LOGIN REGISTER

tom

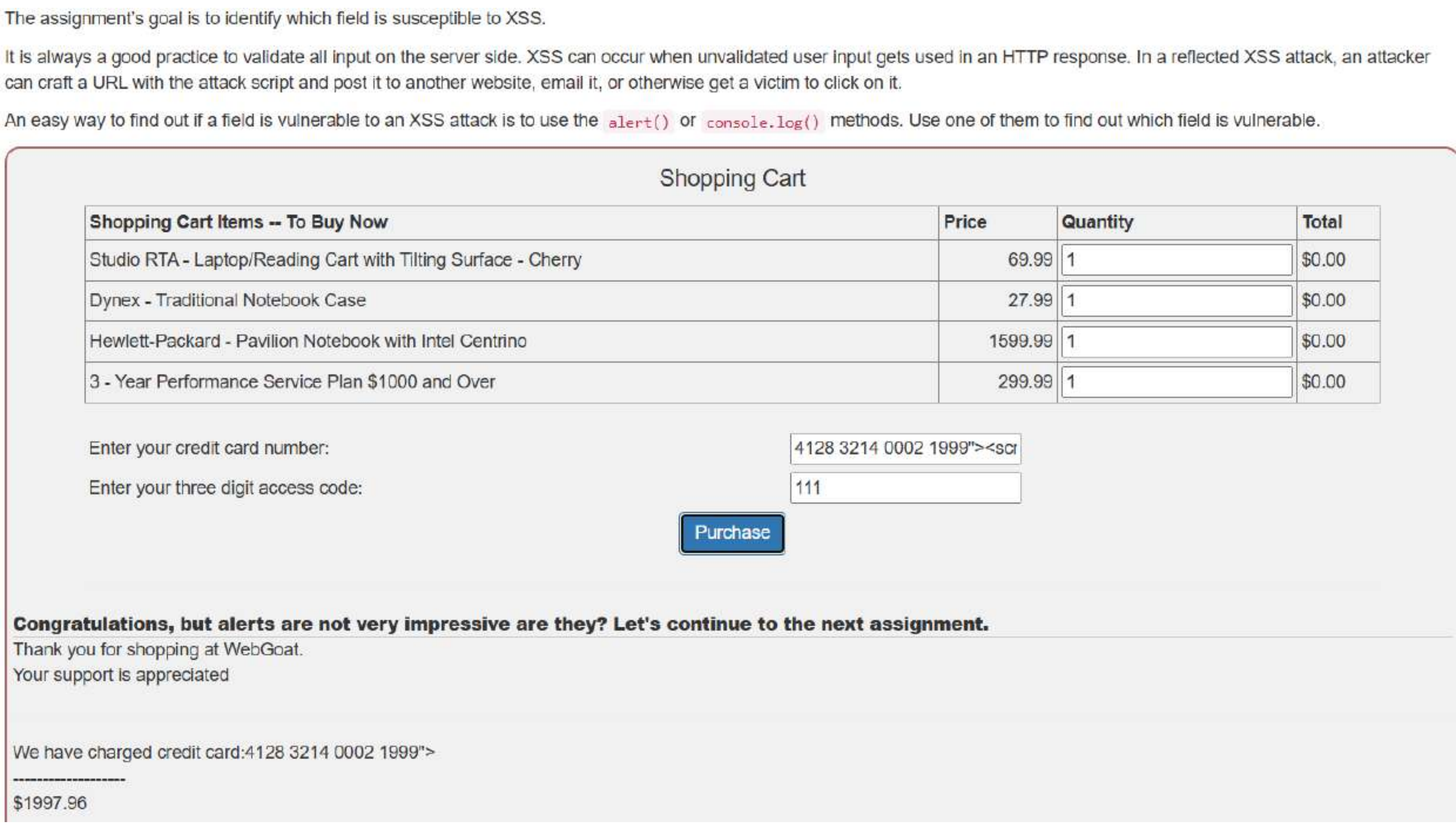
••••••••••••••••••••••

☐ Remember me

Log In

Forgot Password?

Congratulations. You have successfully completed the assignment.



The assignment's goal 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"><scr
Enter your three digit access code:	111

Purchase

Congratulations, but alerts are not very impressive are they? Let's continue to the next assignment.

Thank you for shopping at WebGoat.
Your support is appreciated

We have charged credit card:4128 3214 0002 1999">

$1997.96