# XSS VULNERABILITY

**Project-Based Internship 2020 Report**

**Submitted**

**To**

**DataRitz Technologies**

**Duration 6 weeks**

**By**

## KHUSHI RAUNIYAR

**1803213083**

**ABES Engineering College**

**ABDUL KALAM TECHNICAL UNIVERSITY**

**Under the guidance of**

## KRISHNA VIR SINGH

**DECLARATION**

I hereby declare that the Project Report is an authentic record of my own work as requirements of Internship during the period from 15 May, 2020 to 30 June, 2020 for the award of degree of B.Tech. (Information Technology) ABES Engineering College, Ghaziabad.

**Signature of student**

**(Khushi Rauniyar)**

**(1803213083)**

**Date: 30/06/2020**

**CERTIFICATE**

This is to certify that Project Report entitled “XSS VULNERABILITY” which is submitted by KHUSHI RAUNIYAR in partial fulfillment of the requirement for the summer internship of “CISCO Certified Cyber Ops Associate” in Department of Information Technology of ABES ENGINEERING COLLEGE is a record of the candidate's own work carried out by her under my supervision.

**Signature of Supervisor**

**Date:**

**ACKNOWLEDGEMENT**

It gives us a great sense of pleasure to present the report of the Project Based Internship 2020 undertaken during CISCO Cyber Ops Associate 2020. We owe special debt of gratitude to Krishna Vir Singh, DataRitz Technologies for his constant support and guidance throughout the course of our work. His constant motivation has been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of team members of DataRitz Technologies for their full support and assistance during the development of the project.

We also do not like to miss the opportunity to acknowledge the motivation of Information Technology Department and ABES Engineering College to provide us the opportunity to undergo training at DataRitz Technologies.

Signature:

Name: Khushi Rauniyar

Roll No: 1803213083

Class: IT-B

Date: 30.06.2020

**LIST OF FIGURES**

Page No.

1. Reflected XSS Flowchart 15

2. Stored XSS Flowchart 16

3. Burp Suite for Challenge 1 28

4. Successful completion of Challenge 1 28

5. Burp Suite for Challenge 2 29

6. Successful completion of Challenge 2 29

7. Successful completion of Challenge 3 30

8. Successful completion of Challenge 4 30

9. Successful completion of Challenge 5 31

10. Successful completion of Challenge 6 31

11. Successful completion of Challenge 7 32

12. Successful completion of Challenge 8 32

13. Successful completion of Challenge 8 33

14. Successful completion of Labs 33

15. Challenge 1 (Reflected XSS) 35

16. Challenge 1 (DOM XSS) 36

17. Challenge 1 (Stored XSS) 37

18. Challenge 2 (Reflected XSS) 37

19. Challenge 2 (DOM XSS) 38

20. Challenge 2 (Stored XSS) 39

21. Challenge 3 (Reflected XSS) 40

22. Challenge 3 (DOM XSS) 41

23. Challenge 3 (Stored XSS) 41

# TABLE OF CONTENTS

**Page no.**

**DECLARATION 2**

**CERTIFICATE 3**

**ACKNOWLEDGEMENT 4**

## LIST OF FIGURES 5-6

## ABSTRACT 9

## CHAPTER-1 COURSE DESCRIPTION 10-11

CISCO Certified Cyber Ops Associate

## CHAPTER-2 INTRODUCTION TO PROJECT 12

* 1. Aim of the Project
  2. Objective of the Project
  3. Scope of the Project

## CHAPTER-3 DESCRIPTION 13-20

* 1. OVERVIEW TO XSS
  2. DESCRIPTION OF XSS
  3. TYPES OF XSS
     1. Reflected XSS
     2. Stored XSS

3.2.3 Dom-based XSS

* 1. WORKING OF XSS
  2. IMPACT OF XSS
  3. PREVENTION OF XSS

## CHAPTER-4 TOOL DESCRIPTION 21-24

* 1. INTRODUCTION TO BURP SUITE
  2. DIFFERENT TOOLS IN BURP SUITE
     1. Spider
     2. Proxy
     3. Intruder
     4. Repeater
     5. Sequencer
     6. Decoder
     7. Extender
     8. Scanner

## CHAPTER-5 IMPLEMENTATION AND RESULTS 25-41

**CONCLUSION 42**

**REFERENCES 43**

**ABSTRACT**

Cyber Security plays an important role in the field of information technology. Securing the information has become one of the biggest challenges in the present day. Whenever we think about cyber security the first thing that comes to our mind is “cybercrimes” which are increasing immensely day by day. Various Governments and companies are taking many measures in order to prevent these cybercrimes. Besides various measures cyber security is still a very big concern to many.

With improvement in computing and technological advancements, web-based applications are now ubiquitous on the Internet. However, these web applications are becoming prone to vulnerabilities which have led to theft of confidential information, data loss, and denial of data access in the course of information transmission. Cross-site scripting (XSS) is a form of web security attack which involves the injection of malicious codes into web applications from untrusted sources. The best practice to prevent XSS attacks is to apply encoders to sanitize untrusted data. To balance security and functionality, encoders should be applied to match the web page context, such as HTML body, Java script, and style sheets. A common programming error is the use of a wrong encoder to sanitize untrusted data, leaving the application vulnerable. This project report describes XSS vulnerability, how to exploit it, what its impact and how to prevent it, along with the tool description used in this exploit.

# CHAPTER 1

**COURSE DESCRIPTION**

**CISCO Certified Cyber Ops Associate**

**CCNA Cyber Ops Certification** is a great start to your career in Cyber Security operations. It prepares candidates to begin a career working with associate-level Cyber Security analysts within security operations centers. This certification helps us in acquiring skills and helps in preparation to work with the Security Operations Center (SOC) teams. Today's organizations are challenged with rapidly detecting cyber security breaches and effectively responding to security incidents. Teams of people in Security Operations Centers (SOC’s) keep a vigilant eye on security systems, protecting their organizations by detecting and responding to cyber security threats. The Certifications helps Security people to get equipped with ability to monitor security systems and detect online attacks and helps individuals with the knowledge to identify and respond to security incidents and vulnerabilities.

CCNA Cyber Ops also covers topics that are not specific to Cisco - making it more useful for people which use more than just Cisco technology. It also provides introductory knowledge to many topics which makes us aware of the security landscape, understand security concepts and general networking. We learn topics such as networking concepts and IP addressing, as well as security concepts including access control models, risk assessment, and the CIA triad. We can also review cryptography methods and host-based analysis details, as well as security monitoring tools, and attack methods used by threat actors.

The program has one training course and one exam that covers the foundational skills, processes, and Knowledge you need to prevent, detect, analyze and respond to cybersecurity incidents as per SOC team.

Cisco courses from official Cisco training providers include interactive online labs, providing you with a chance to get hands-on with the technology.

**Main topics covered are:**

* Network Concepts.
* Cryptography.
* Security Concepts.
* Security Monitoring.
* Public Key Infrastructure.
* Attack Methods.
* Incident Handling.
* Incident Response.
* Malware Analysis and Implementation.
* Different OS - Windows and Linux.
* Access Control Model for Digital Assets.
* Host-Based Analysis.
* Network Intrusion Analysis.
* Data and Event Analysis.
* Endpoint Threat Analysis and Computer Forensics.

# INTRODUCTION TO PROJECT

* 1. **Aim of the Project:** To learn about Cross Site Scripting (XSS) vulnerability.
  2. **Objective of the Project:** To inspect, learn and explore about Cross Site Scripting (XSS) Vulnerability, how it is performed and its impacts.
  3. **Scope of the Project: Cross-Site Scripting** or **XSS** is second most prevalent Web Application Vulnerability and is also in the current OWASP Top Ten Critical Web Application Vulnerability. It also exists in two-thirds of all the applications. XSS vulnerabilities are easily discovered by attackers as well as it can also be easily discovered by the defenders.

XSS allows an attacker to destroy the existing project, or take advantage of it. XSS injection can be possible when there is a scope for the attacker to insert unauthorized JavaScript, VBScript, HTML, or other active content into a web page. So, people must check their code and clean web applications and take all the safety measures to prevent it with every possible way and stop the attackers from performing any malicious activities.

# DESCRIPTION

* 1. **Overview to XSS:**

Cross Site Scripting (XSS) is a vulnerability in a web application that allows the attacker to execute a JavaScript payload or in simple words a JavaScript code in the user’s browser on behalf of the web application. Cross-site Scripting is one of the most prevalent vulnerabilities present on the web today. The exploitation of XSS against a user can lead to various consequences such as account compromise, account deletion, privilege escalation, malware infection and many more.

An attacker can use the XSS payload in order to execute a particular set of code at the time user logs into the account or in situations when user reads some comments or stored data. XSS may sometimes be used to retrieve server’s crucial information. By means of XSS an attacker may even retrieve the important information of browser like saved passwords and stuff.

# Description of XSS:

Cross-Site Scripting (XSS) attacks occur when:

* + 1. Some information entered gets reflected back.
    2. The information entered gets stored somewhere without getting sanitized.

A webpage often contains some parameters in which we have to enter some values and those values are processed by the server and the server returns us the output. In case the parameter is not sanitized then someone may send some malicious code that could be executed on the server and may lead to destruction or compromisation of the data of user or maybe the privacy of user. XSS is one of the deadliest attacks if a professional hacker performs it. Hence, it is a necessity for any project to have least XSS vulnerabilities.

# Types of XSS:

* **Reflected XSS**
* **Stored XSS**
* **DOM-Based XSS**

# Reflected XSS:

Reflected XSS is a type of XSS which generally causes harm to client side. This kind of XSS is generally found when any parameter is returned on the page as it was entered by the user. It is the simplest type of cross-site scripting. It arises when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way.

If the user visits the URL constructed by the attacker, then the attacker's script executes in the user's browser, in the context of that user's session with the application. At that point, the script can carry out any action, and retrieve any data, to which the user has access.

Reflected XSS has a type called self XSS in which a victim unknowing runs a malicious code which is sent to the victim by means of a URL.

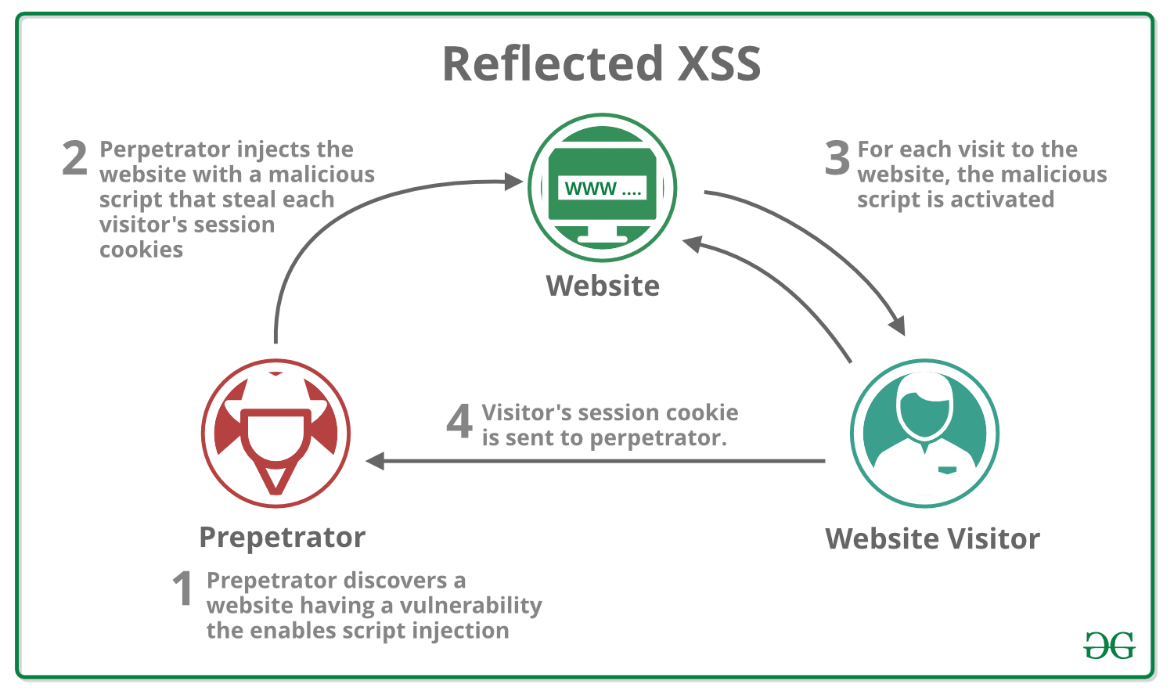


Figure 1 Reflected XSS Flowchart

(Image source: geeksforgeeks.org)

# Stored XSS:

Stored XSS (also known as persistent or second-order XSS) is a type of XSS in which the code or the value of the parameter is store at the server. It generally occurs at the server side. It arises when an application receives data from an untrusted source and includes that data within its later HTTP responses in an unsafe way.

The vulnerable parameters could be the comment boxes, user nicknames in a chat room, or contact details on a customer order or the user descriptions and details. In other cases, the data might arrive from other untrusted sources as well for example, a webmail application, displaying messages received over SMTP, a marketing application displaying social media posts, or a network monitoring application displaying packet data from network traffic.

Every time any user opens the page containing the malicious comment or description will be a prey of XSS attack.

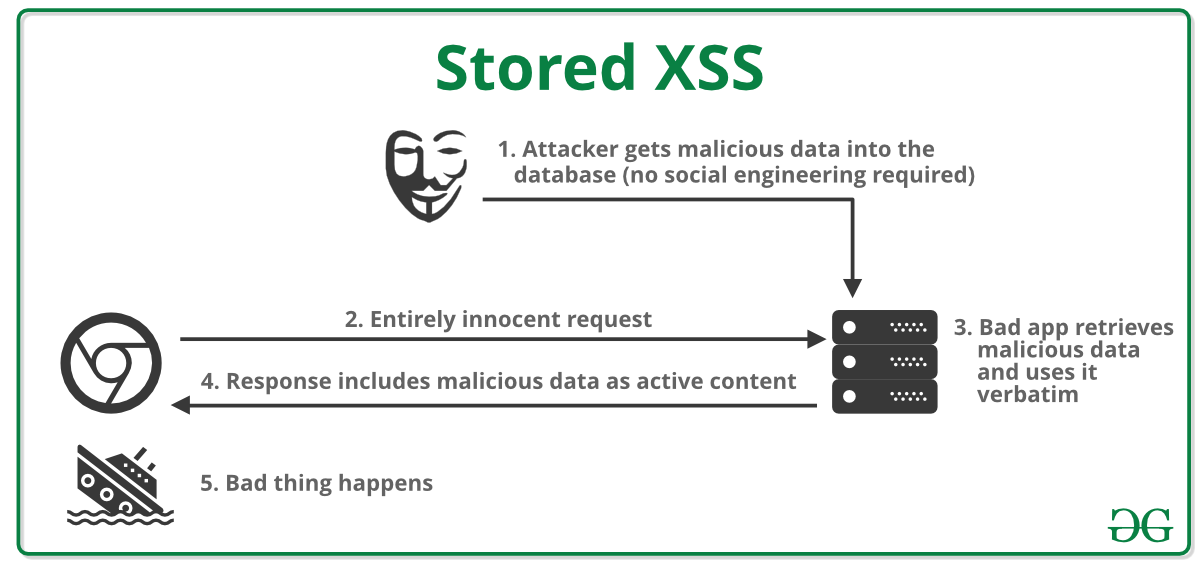


Figure 2 Stored XSS Flowchart

(Image source: geeksforgeeks.org)

# DOM-Based XSS:

DOM-based XSS arises when an application contains some client-side JavaScript that processes data from an untrusted source in an unsafe way, usually by writing the data back to the DOM. In a typical case, the input field would be populated from part of the HTTP request, such as a URL query string parameter, allowing the attacker to deliver an attack using a malicious URL, in the same manner as reflected XSS.

In this the vulnerable parameters are DOM objects like headers, user-agent etc. When a website uses these parameters without sanitizing them then it have the chances of being vulnerable to DOM based XSS.

**Example of XSS vulnerable PHP Code:-**

<?php

if (isset($\_GET["contact"]) && isset($\_GET["email"]))

{

$contact = $\_GET["contact"];

$email = $\_GET["email"];

echo "Entered details are: " . $contact. " " . $email;

}

?>

# Working of XSS:

**Cross Site Scripting** attack is an easy to find vulnerability. The way of finding and exploiting it depends on the type of XSS it is. It means sending and injecting malicious code or script. Malicious code is usually written with client-side programming languages such as JavaScript, HTML, VBScript Flash, etc. However, JavaScript and HTML are mostly used to perform this attack.

This attack can be performed in different ways. Depending upon the type of XSS attack, the malicious script may be reflected on the victim’s browser or stored in the database and executed every time, when the user calls the appropriate function

In order to look for a reflected XSS look for a parameter that reflects itself on the page. And after that try to check whether the parameter is sanitized properly or not, in case the parameter is sanitized properly it is a bit harder to apply XSS but keep on trying.

In order to look for a Stored XSS look for a parameter that gets stored at the server side and is reflected to either admin or other users. Check whether the parameter is sanitized or not and in case the parameter is not sanitized properly, Bravo you got a XSS.

In order to look for a DOM based XSS look for the DOM object which is being processed by the server and then check whether the parameter is sanitized or not in case it is not, try to break through it.

The main reason for this attack is inappropriate user’s input validation, where malicious input can get into the output. A malicious user can enter a script, which will be injected into the website’s code. Then the browser is not able to know if the executed code is malicious or not. Therefore malicious script is being executed on the victim's browser or any faked form is being displayed for the users.

# Impact of XSS:

The Impact of XSS is not fixed, it varies from application to application and page to page. In order to understand the impact of XSS, here I have listed a few famous case studies/reports examples:

* <https://hackerone.com/reports/868934> . In this report the attacker found a XSS vulnerability in the very famous search engine **“Duckduckgo”** .
* <https://hackerone.com/reports/803934> . In this report the attacker found a DOM based XSS vulnerability in the popular gaming website **“Rockstar Games”.**
* In an application holding sensitive data, such as banking transactions, emails, or healthcare records, the impact will usually be serious.
* If the compromised user has elevated privileges within the application, then the impact will generally be critical, allowing the attacker to take full control of the vulnerable application and compromise all users and their data.

# Prevention of XSS:

Preventing cross-site scripting is trivial and bit complex in some cases but can be much harder depending on the complexity of the application and the ways it handles user-controllable data. Here is some list of measures to prevent XSS:

* Using htmlspecialchars() function.
* Using htmlentities() function
* Using strip\_tags() function
* Using addslashes() function
* By using Content Security Policy (CSP)

Even the prevention of XSS depends on the size of the application as well the larger the application is, the more the prevention measures are required. In general, effectively preventing XSS vulnerabilities is likely to involve a combination of the following measures:

* 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.
* Content Security Policy. As a last line of defense, you can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occurs.
* Try properly sanitizing the user input as much as you can .Try to take only that input form user which is of your need and sanitize the rest of them and try to avoid unnecessary parameters as well.

# CHAPTER 4 TOOL DESCRIPTION

**Tool Name: Burp Suite**

* 1. **Introduction to Burp Suite:**

Burp Suite is a set of tools used for penetration testing of web applications. It is developed by the company named Portswigger, which is also the alias of its founder Dafydd Stuttard. Burp Suite aims to be an all in one set of tools and its capabilities can be enhanced by installing add-ons that are called BApps.

It is the most popular tool among professional web app security researchers and bug bounty hunters. Its ease of use makes it a more suitable choice over free alternatives like OWASP ZAP. Burp Suite is available as a community edition which is a free, professional edition that costs $399/year and an enterprise edition that costs $3999/Year.

# Different tools in Burp Suite:

* + 1. **Spider:**

It is a web spider/crawler that is used to map the target web application. The objective of the mapping is to get a list of endpoints so that their functionality can be observed and potential vulnerabilities can be found.

Spidering is done for a simple reason that the more endpoints you gather during your recon process, the more attack surfaces you possess during your actual testing.

# Proxy:

BurpSuite contains an intercepting proxy that lets the user see and modify the contents of requests and responses while they are in transit. It also lets the user send the request/response under monitoring to another relevant tool in BurpSuite, removing the burden of copy-paste. The proxy server can be adjusted to run on a specific loop-back ip and a port. The proxy can also be configured to filter out specific types of request-response pairs.

# Intruder:

It is a fuzzer. This is used to run a set of values through an input point. The values are run and the output is observed for success/failure and content length. Usually, an anomaly results in a change in response code or content length of the response. BurpSuite allows brute-force, dictionary file and single values for its payload position. The intruder is used for:

* + - * Brute-force attacks on password forms, pin forms, and other such forms.
      * The dictionary attack on password forms, fields that are suspected of being vulnerable to XSS or SQL injection.
      * Testing and attacking rate limiting on the web-app.

# Repeater:

Repeater lets a user send requests repeatedly with manual modifications. It is used for:

* + - * If user-supplied values are being verified, how well is it being done?
      * What values is the server expecting in an input parameter/request header?
      * How does the server handle unexpected values?
      * Is input sanitation being applied by the server?
      * How well the server sanitizes the user-supplied inputs?
      * What is the sanitation style being used by the server?

# Sequencer:

The sequencer is an entropy checker that checks for the randomness of tokens generated by the webserver. These tokens are generally used for authentication in sensitive operations: cookies and anti-CSRF tokens are examples of such tokens. Ideally, these tokens must be generated in a fully random manner so that the probability of appearance of each possible character at a position is distributed uniformly. This should be achieved both bit-wise and character-wise. An entropy analyzer tests this hypothesis for being true. It works like this: initially, it is assumed that the tokens are random. Then the tokens are tested on certain parameters for certain characteristics.

# Decoder:

Decoder lists the common encoding methods like URL, HTML, Base64, Hex, etc. This tool comes handy when looking for chunks of data in values of parameters or headers. It is also used for payload construction for various vulnerabilities.

# Extender:

Burp Suite supports external components to be integrated into the tools suite to enhance its capabilities. These external components are called BApps. These work just like browser extensions. These can be viewed, modified, installed, uninstalled in the Extender window. Some of them are supported on the community version, but some require the paid professional version.

# Scanner:

The scanner is not available in the community edition. It scans the website automatically for many common vulnerabilities and lists them with information on confidence over each finding and their complexity of exploitation. It is updated regularly to include new and less known vulnerabilities.

**CHAPTER 5**

**IMPLEMENTATION**

**Vulnerability – Cross Site Scripting (XSS).**

**Description –** I have exploited XSS vulnerability in Leettime Lab and DVWA Lab and completed all its challenges.

**Vulnerable URL –**

* <http://leettime.net/xsslab1/chalg1.php>
* localhost://dvwa

**Leettime Lab :**

**Payloads –**

* <script>alert(document.URL)</script>
* ><script>alert(document.URL)</script>
* “><script>alert(document.URL)</script>
* ‘><script>alert(document.URL)</script>
* “;</script><script>alert(document.URL)</script>
* ‘;</script><script>alert(document.URL)</script>
* ’onmouseover=’alert(document.URL);
* ”onmouseover=”alert(document.URL)

**How to reproduce this Vulnerability** **–**

**Challenge 1:**

* Visit the URL <http://leettime.net/xsslab1/chalg1.php>
* Type any word in the name field provided “Enter Your Name here “and intercept the request with Burp suite.
* Open burp suite and intercept the request.
* Go to Proxy tab in Burp suite and change the “name” parameter with the payload *<script>alert(document.URL)</script>*
* Forward the request and the alert will trigger.

**Challenge 2:**

* Enter payload *><script>alert(document.URL)</script>* in the search box and press enter.
* Open burp suite and intercept the request.
* Go to Proxy tab in Burp suite and change the “name” parameter with the payload *<script>alert(document.URL)</script>*
* The alert will trigger.

**Challenge 3:**

* Enter payload *“><script>alert(document.URL)</script>* in the search box and press enter.
* The alert will trigger.

**Challenge 4:**

* Enter payload *‘><script>alert(document.URL)</script>* in the search box and press enter.
* The alert will trigger.

**Challenge 5:**

* Enter payload *";</script><script>alert(document.URL)</script>* in the search box and press enter.
* The alert will trigger.

**Challenge 6:**

* Enter payload *‘;</script><script>alert(document.URL)</script>* in the search box and press enter.
* The alert will trigger.

**Challenge 7:**

* Enter payload *‘onmouseover=’alert(document.URL);* in the search box and press enter.
* Hover the mouse on search box and the alert will trigger.

**Challenge 8:**

* Type any name in the field provided “ Enter Your Name here” and click on submit button.
* The request has 2 parameters this time one is name other is search and name appears to be sanitized while search is not.
* So enter the payload in the search value and then hit enter.
* The alert will trigger.

**Impact –** The attacker can steal sensitive information by injecting JavaScript

Code and can take full control of the vulnerable application and

compromise all users and their data.

**Screenshots are attached:-**

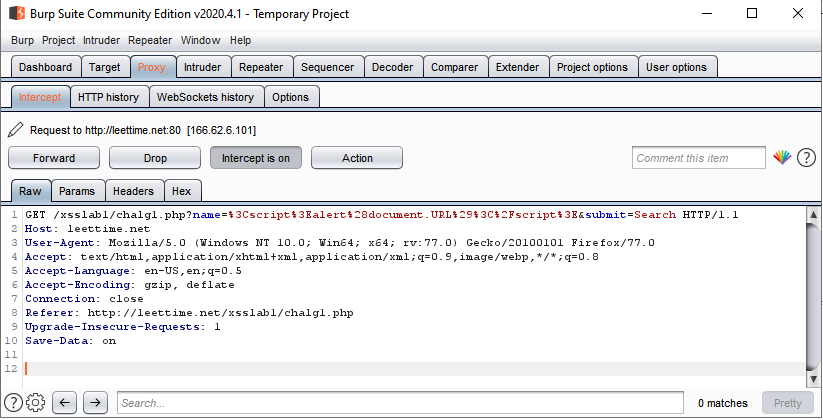
****

Figure 3

* + 1. *Burp Suite for challenge 1*

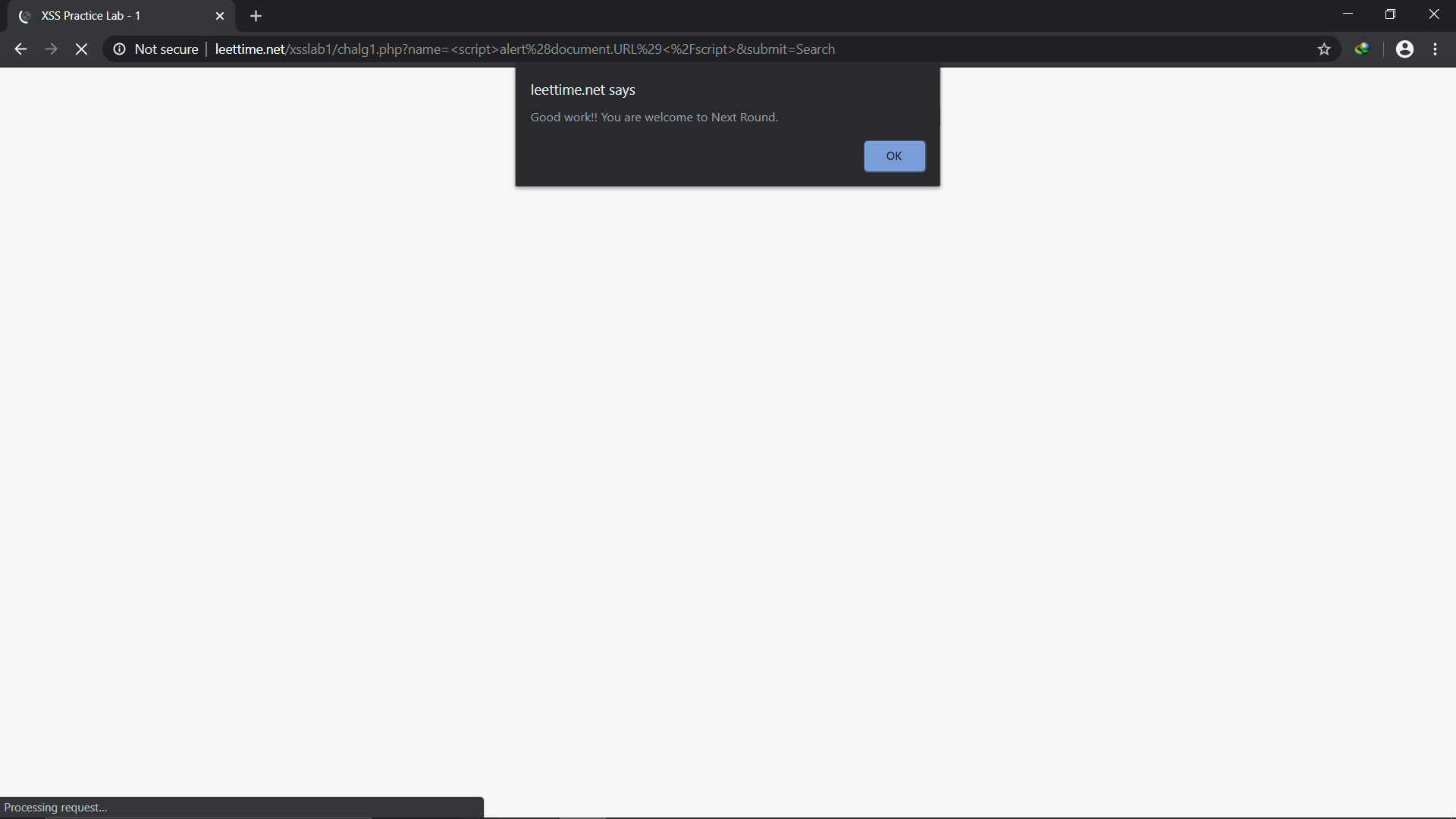


Figure 4

* + 1. *Successful completion of challenge 1*

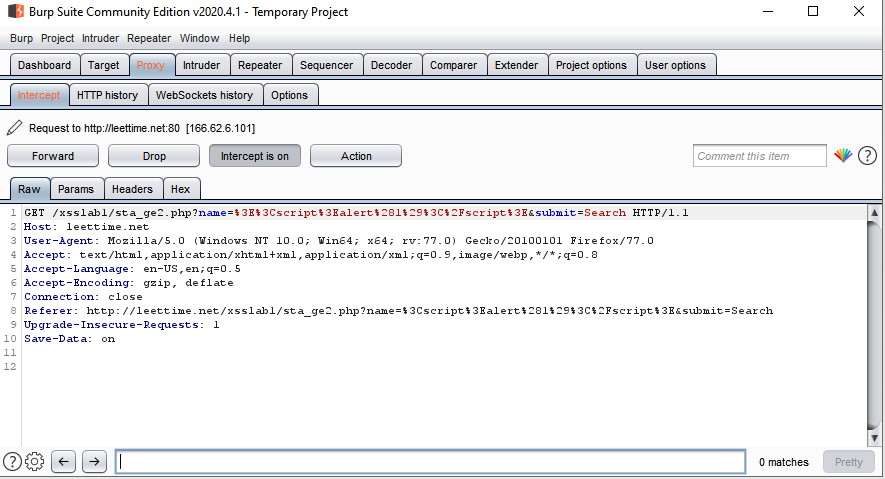
****

Figure 5

* + 1. *Burp Suite for challenge 2*

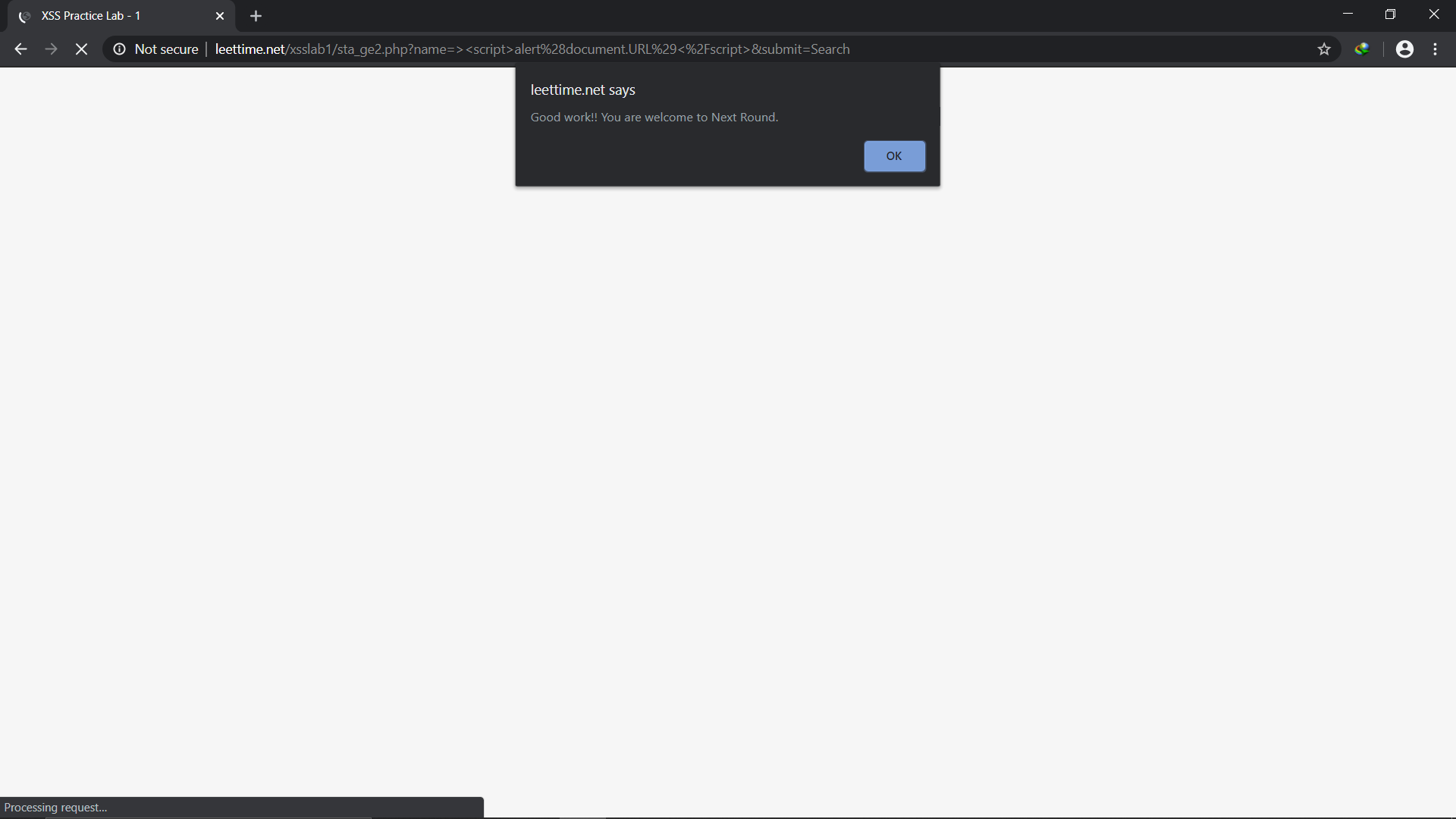


Figure 6

* + 1. *Successful completion of challenge 2*

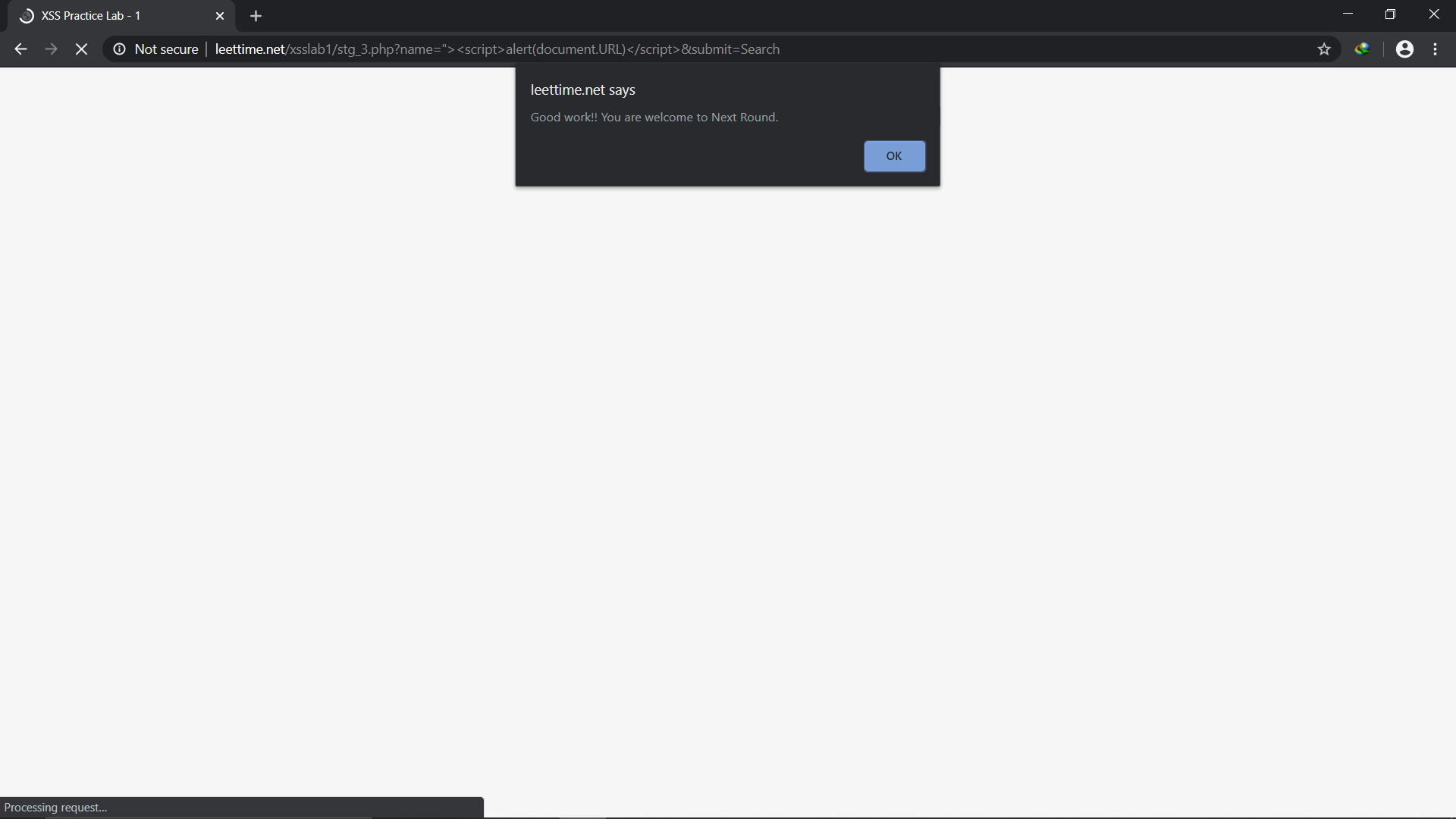


Figure 7

* + 1. *Successful completion of challenge 3*

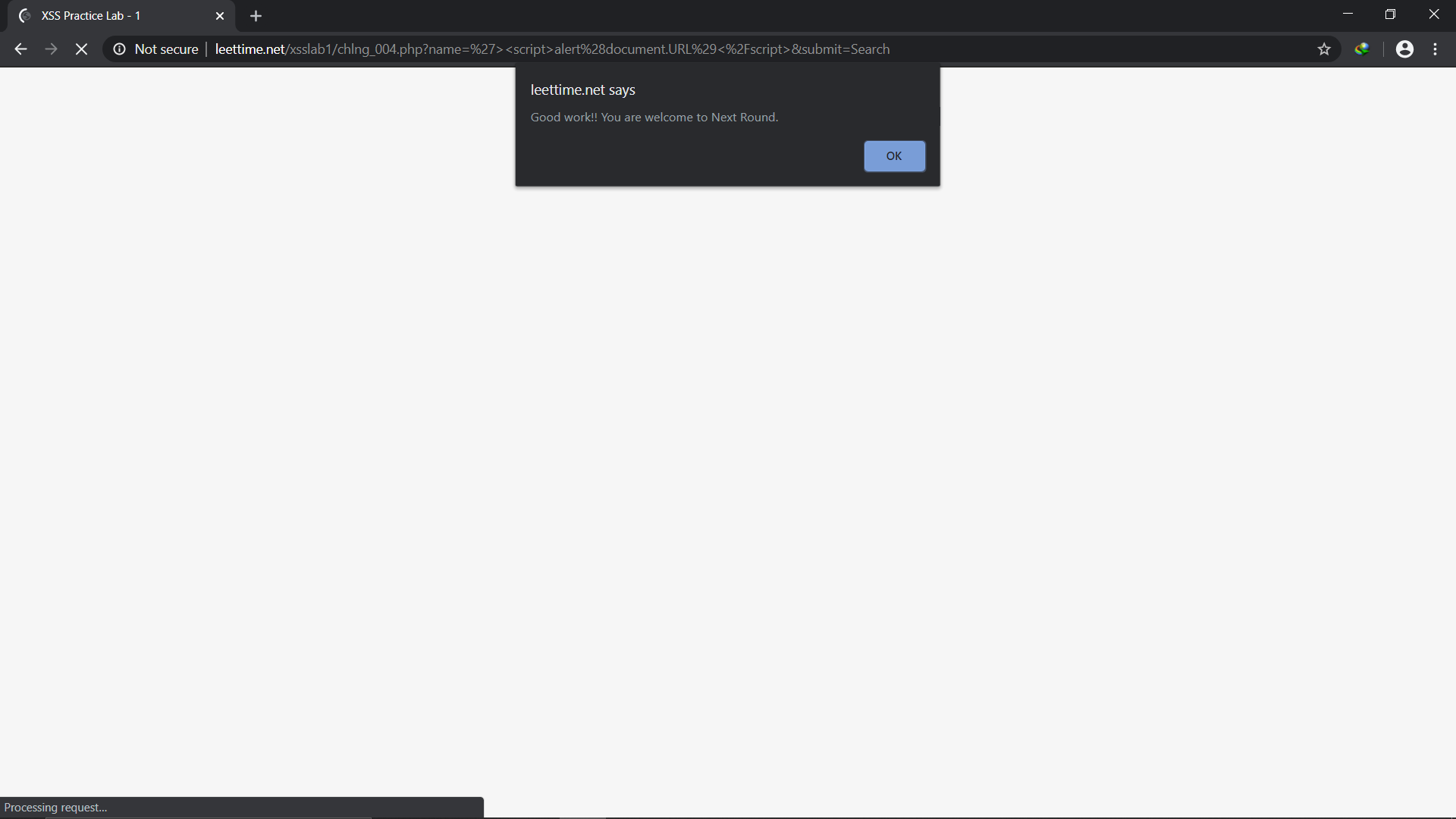


Figure 8

* + 1. *Successful completion of challenge 4*

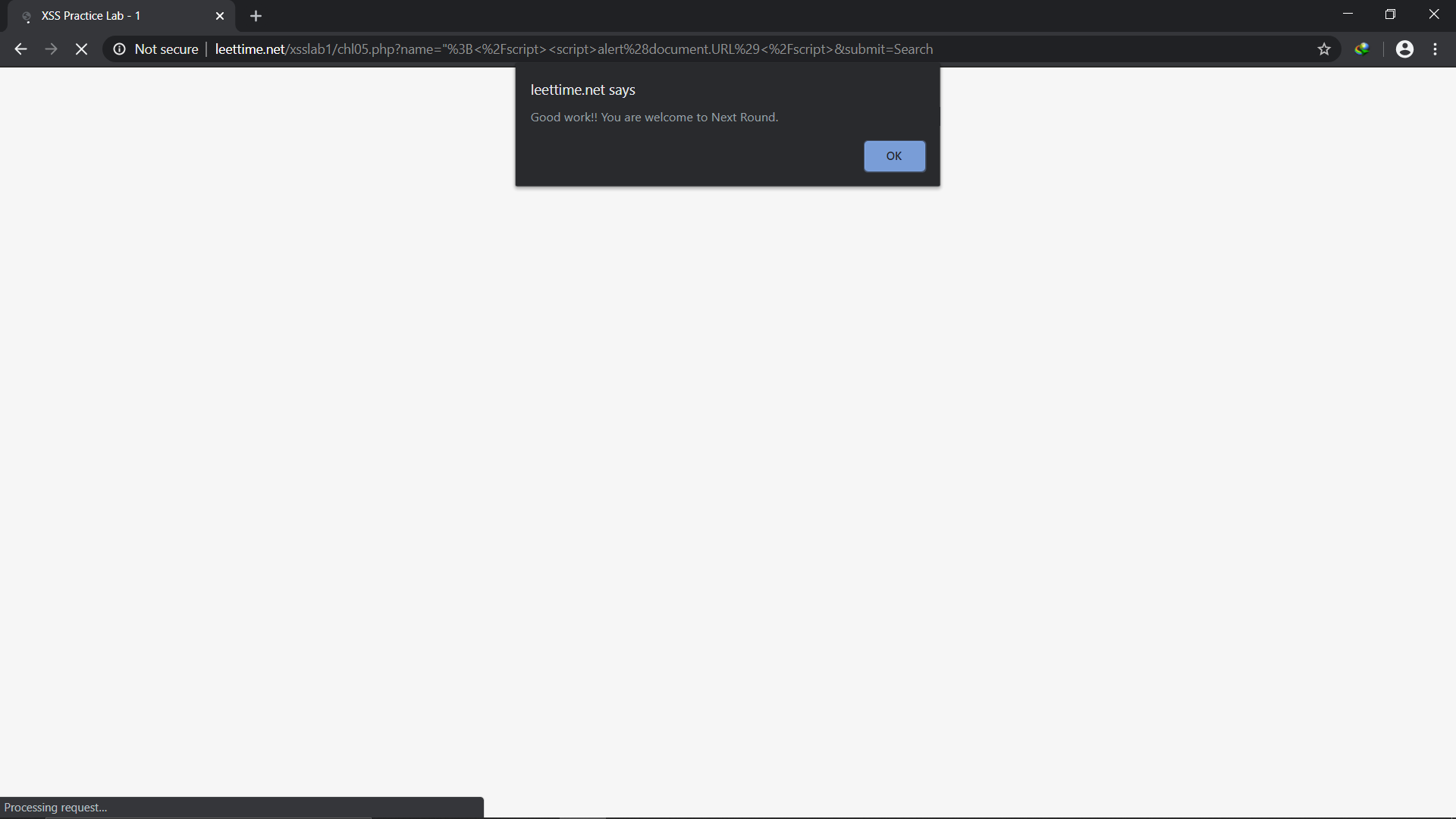


Figure 9

* + 1. *Successful completion of challenge 5*

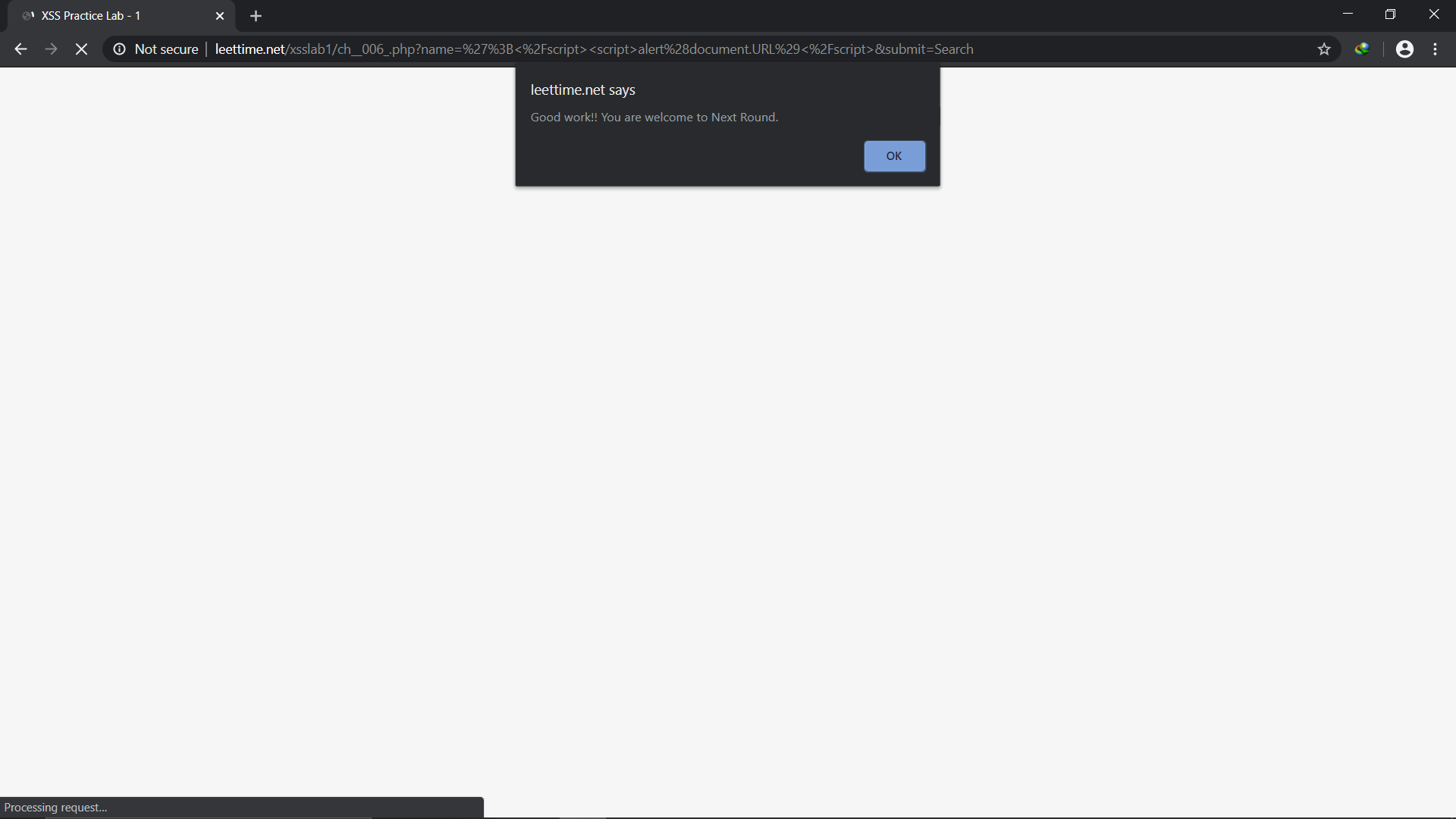


Figure 10

* + 1. *Successful completion of challenge 6*

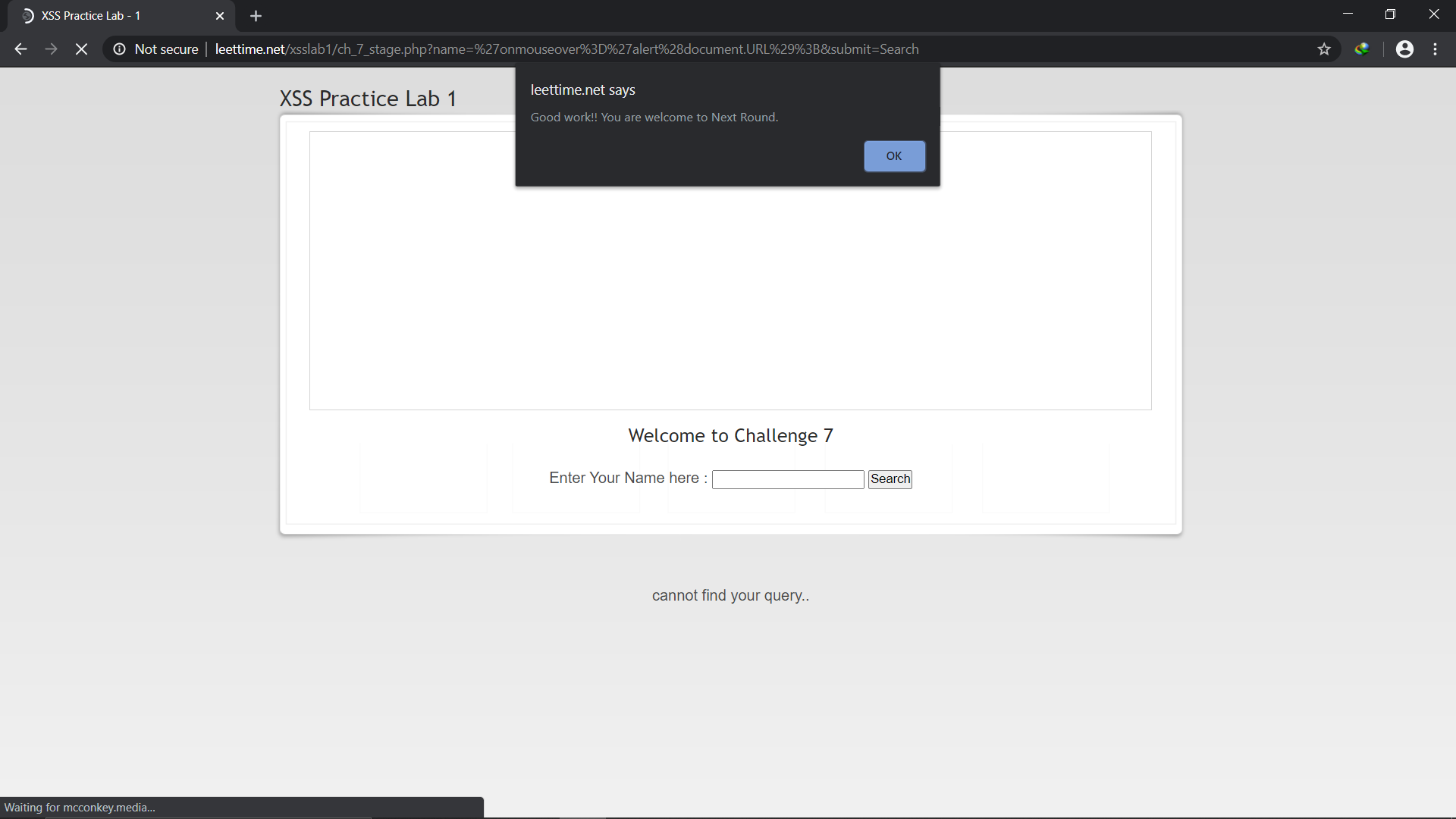


Figure 11

* + 1. *Successful completion of challenge 7*

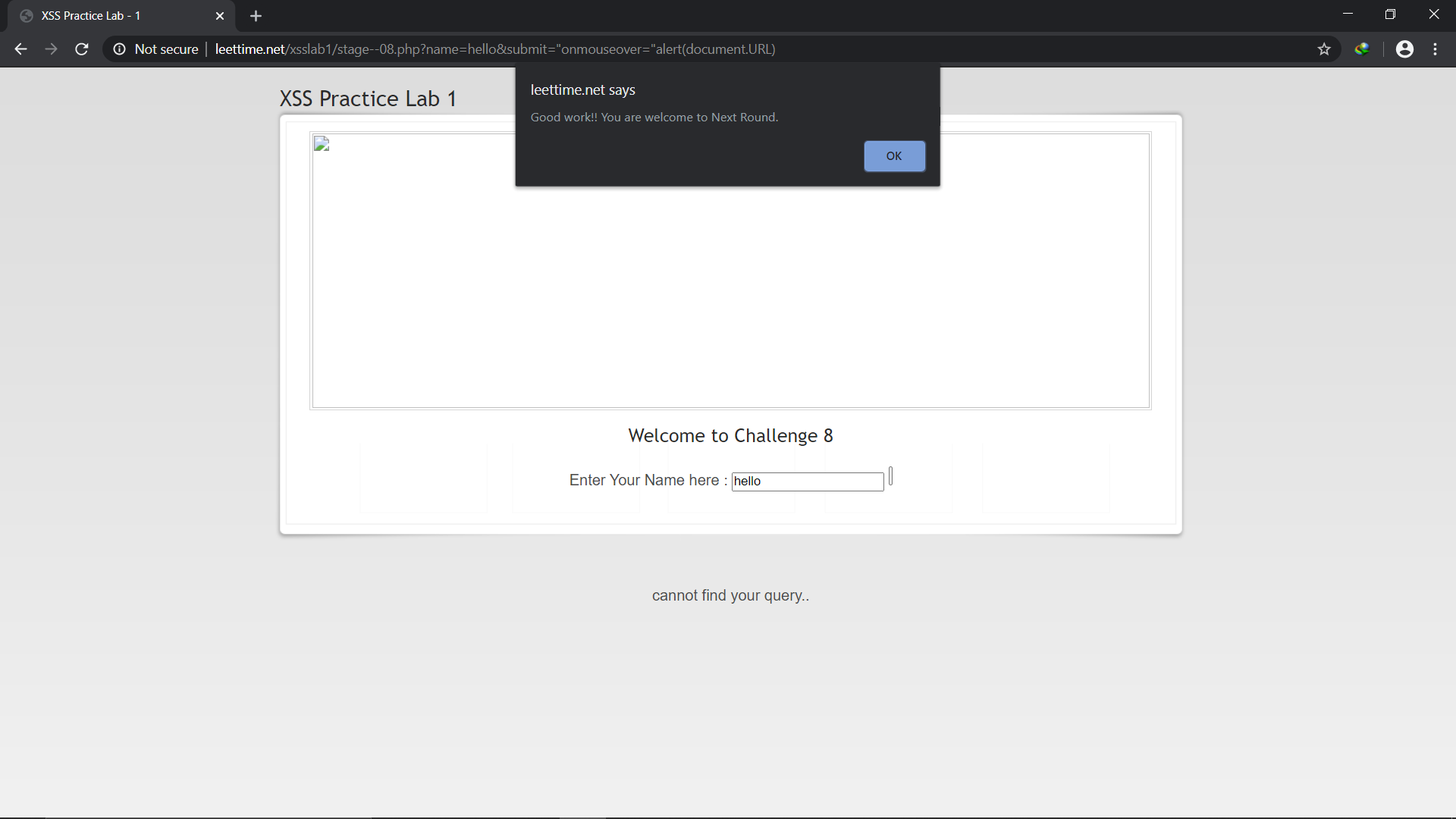


Figure 12

* + 1. *Successful completion of challenge 8*

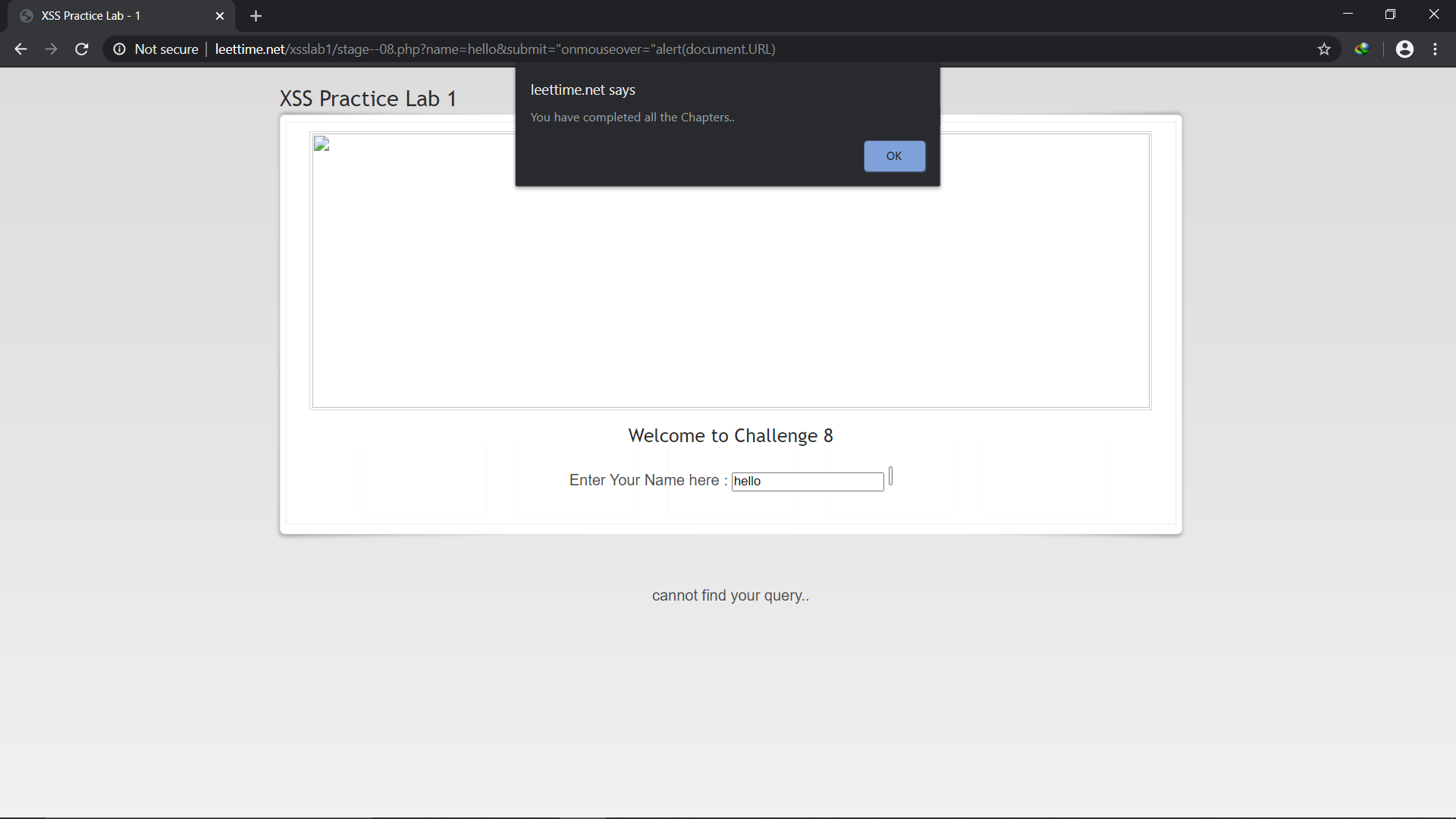


Figure 13

*11. Successful completion of challenge 8*

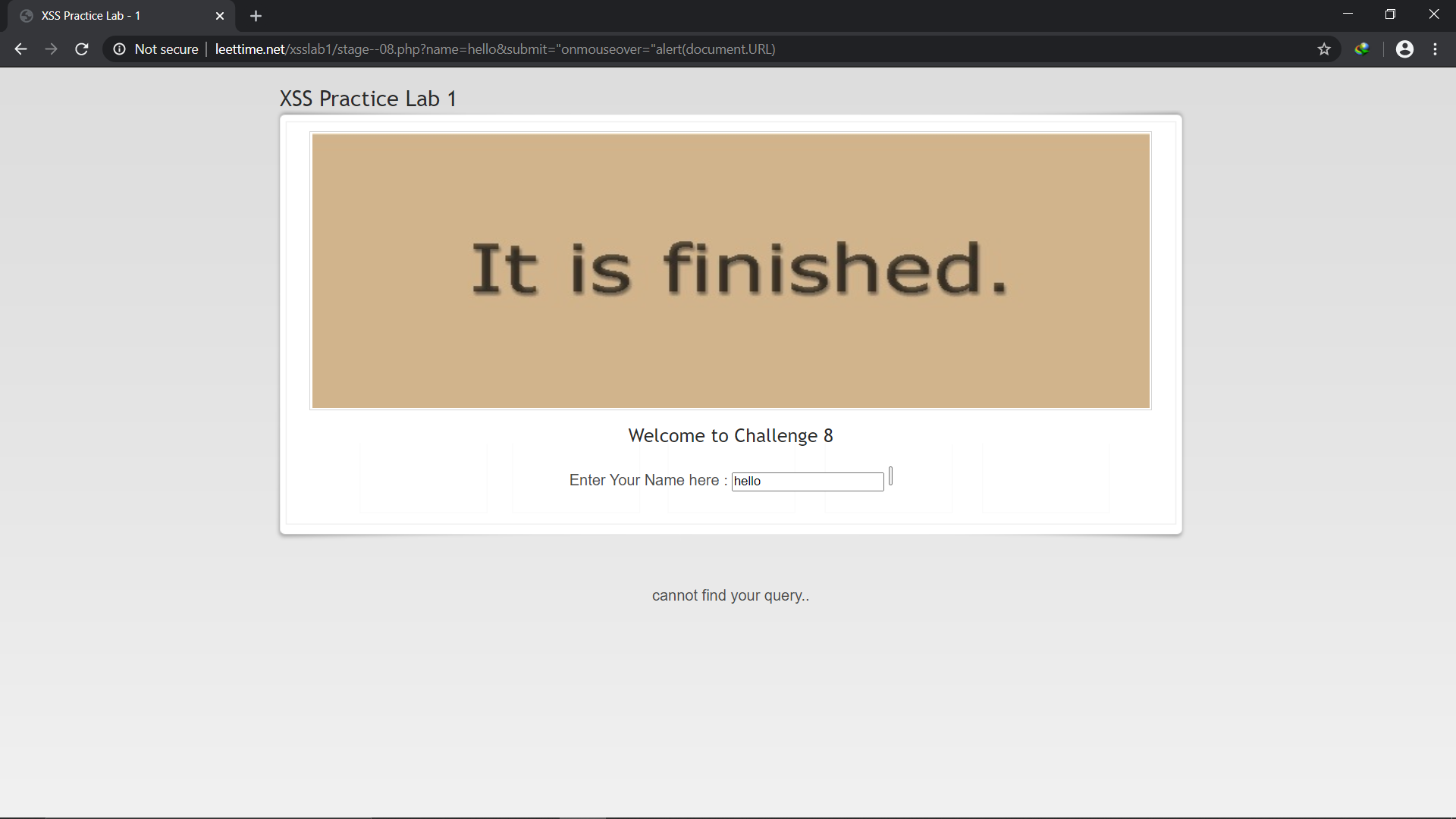


Figure 14

1. *Successful completion of Labs*

**DVWA Lab:**

**Payloads :**

1. <script>alert(1)</script>
2. <body onload=alert(1)>
3. hello>/opttion></select><body onload=alert("XSS")>
4. <scr<script>ipt>alert(document.cookie)</script>
5. <img src=”#” onerror=alert(1) >
6. #</select><img src=1 onerror=alert(1)>
7. <img src=1 onerror=alert(document.cookie)>

**How to reproduce this Vulnerability** **:**

**Challenge 1(Reflected XSS):**

* Visit localhost/dvwa/vulnerabilities/xss\_r
* Type any word in the name field and as it can be seen that it reflects the word, hence it has a vulnerability of reflected XSS.
* Now enter the payload 1 there and as it can be seen the alert box confirms the XSS.

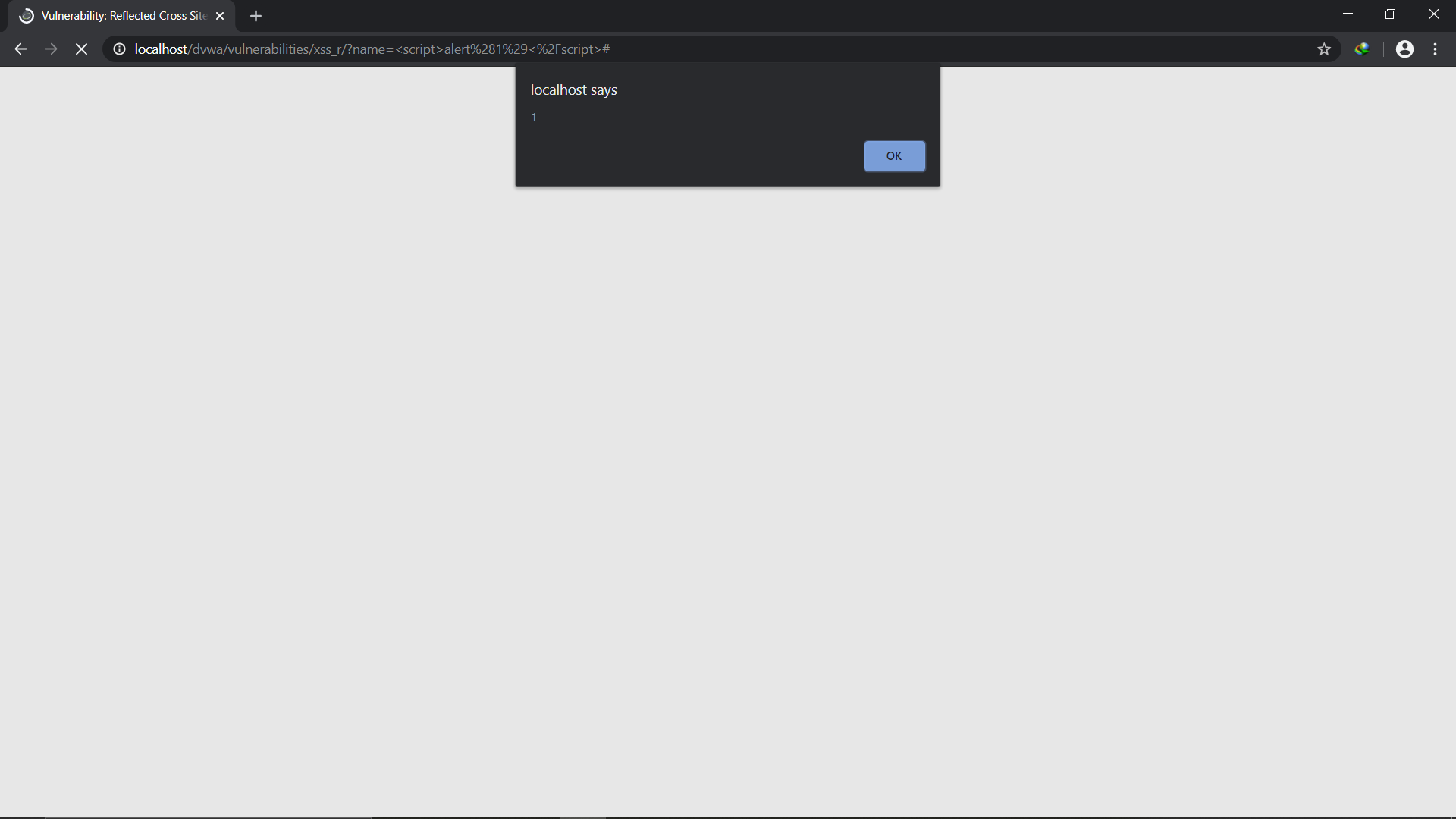


Figure 15

**Challenge 1(DOM XSS):**

* Visit [localhost/dvwa/vulnerabilities/xss\_d](http://leettime.net/xsslab1/chalg1.php)
* Press submit button.
* Change the default parameter to payload 1 and it will execute an alert confirming XSS.

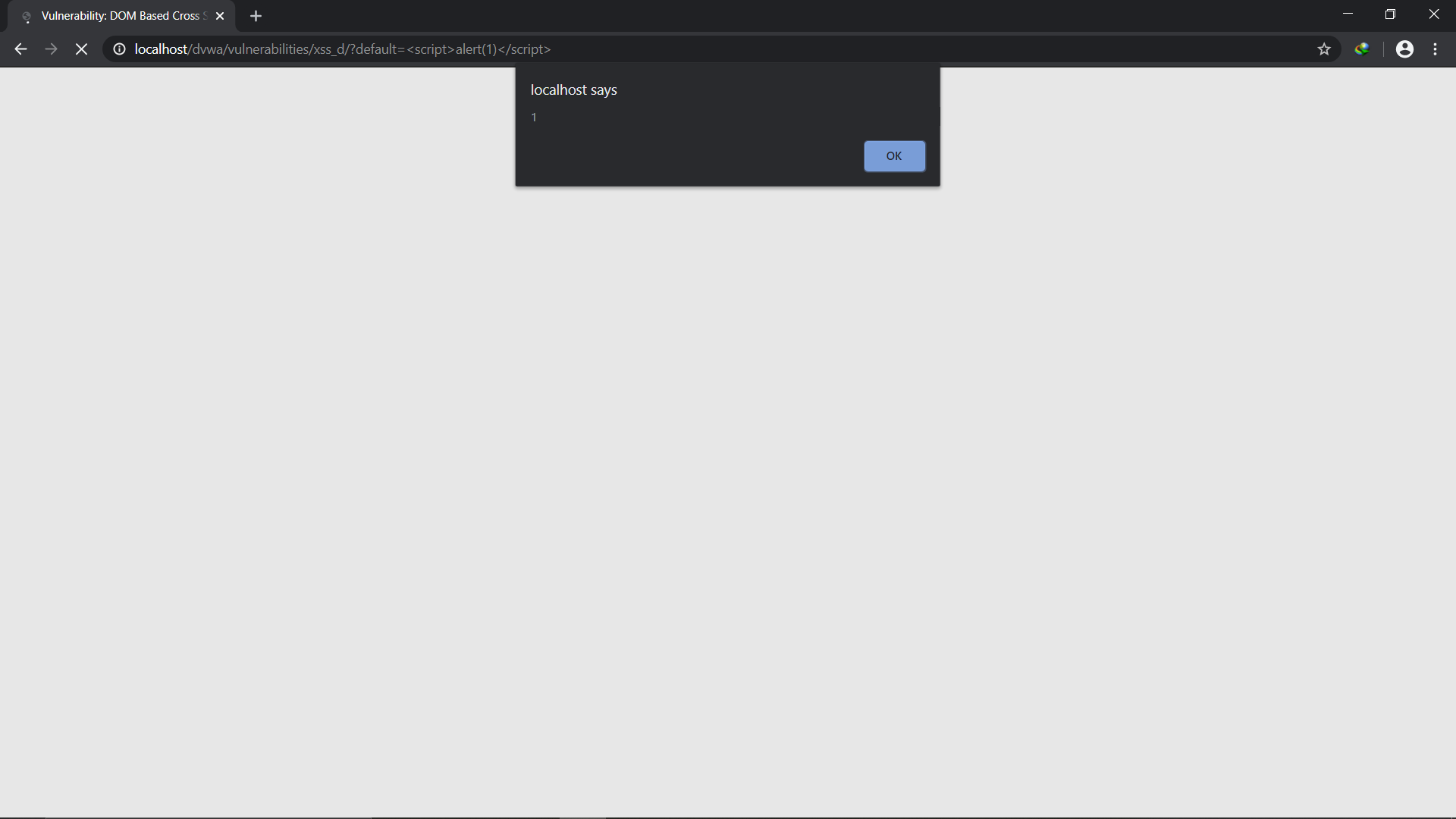


Figure 16

**Challenge 1(Stored XSS):**

* Visit localhost/dvwa/vulnerabilities/xss\_s
* Type any word in the name field and enter the payload 1 in the message field.
* As it can be seen the alert box confirms the XSS.

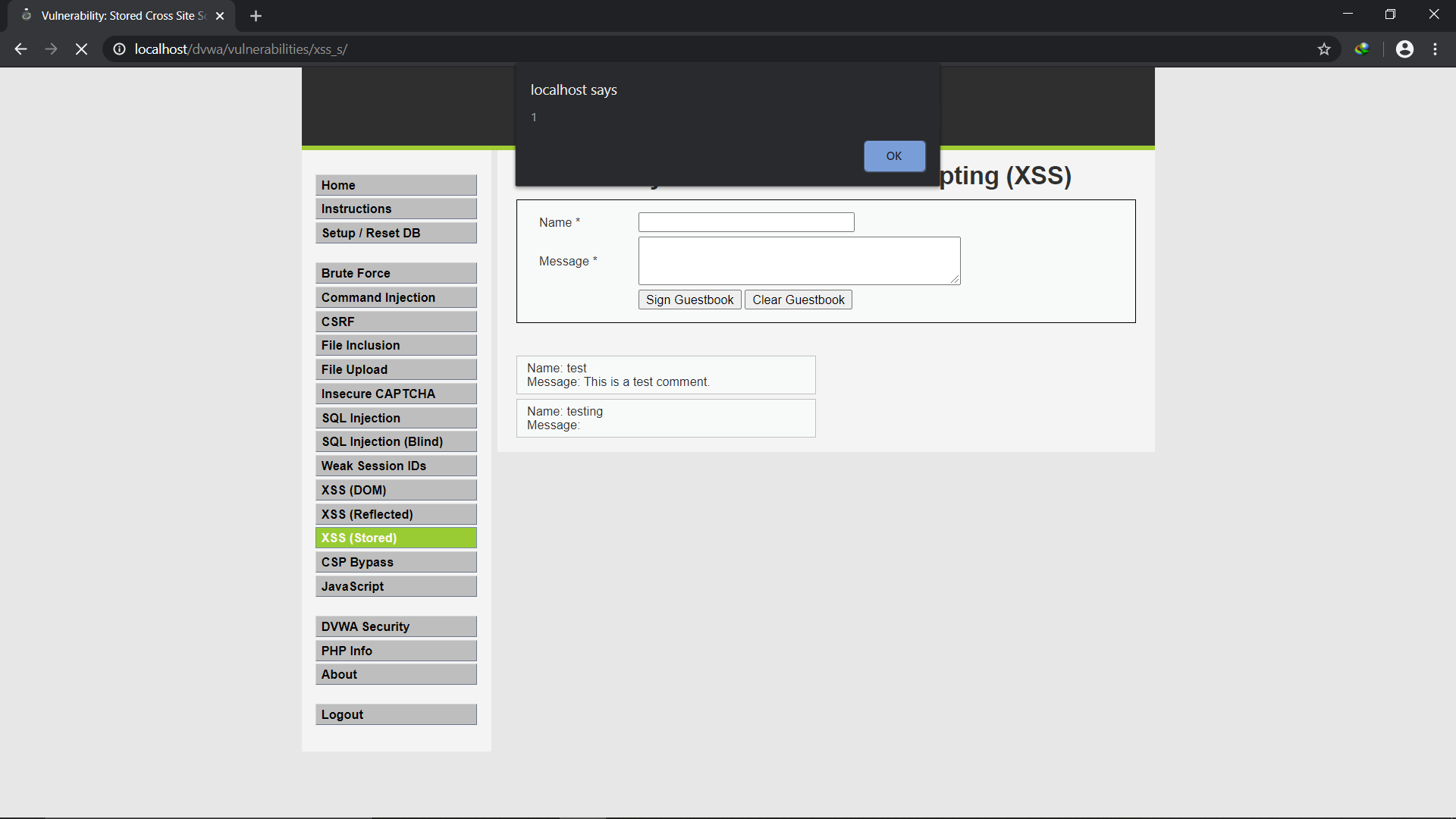


Figure 17

**Challenge 2(Reflected XSS):**

* Enter payload *2* in the box and press enter.
* The alert will trigger.

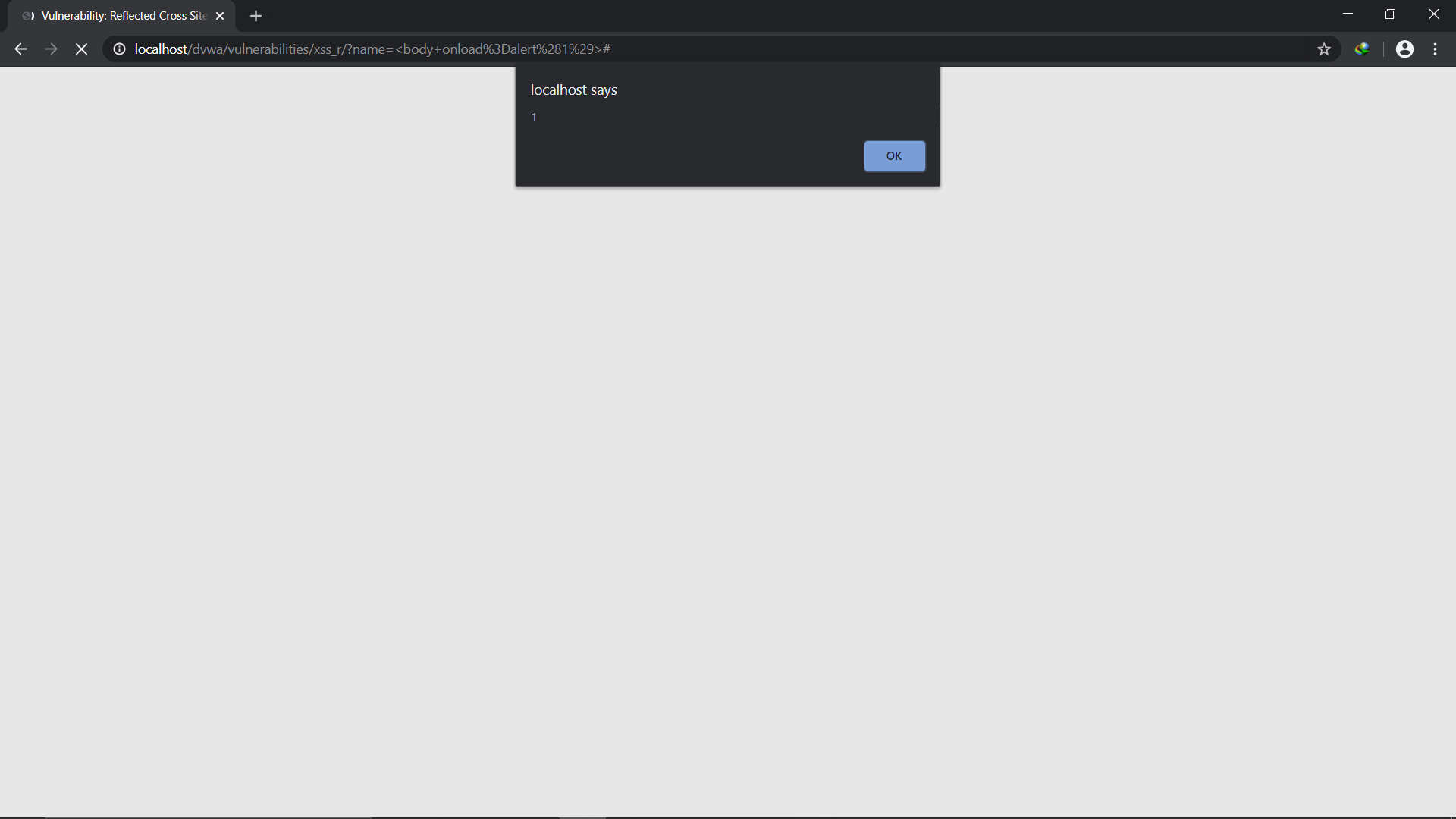


Figure 18

**Challenge 2(DOM XSS):**

* Enter payload *3* in the box and press enter.
* The alert will trigger.

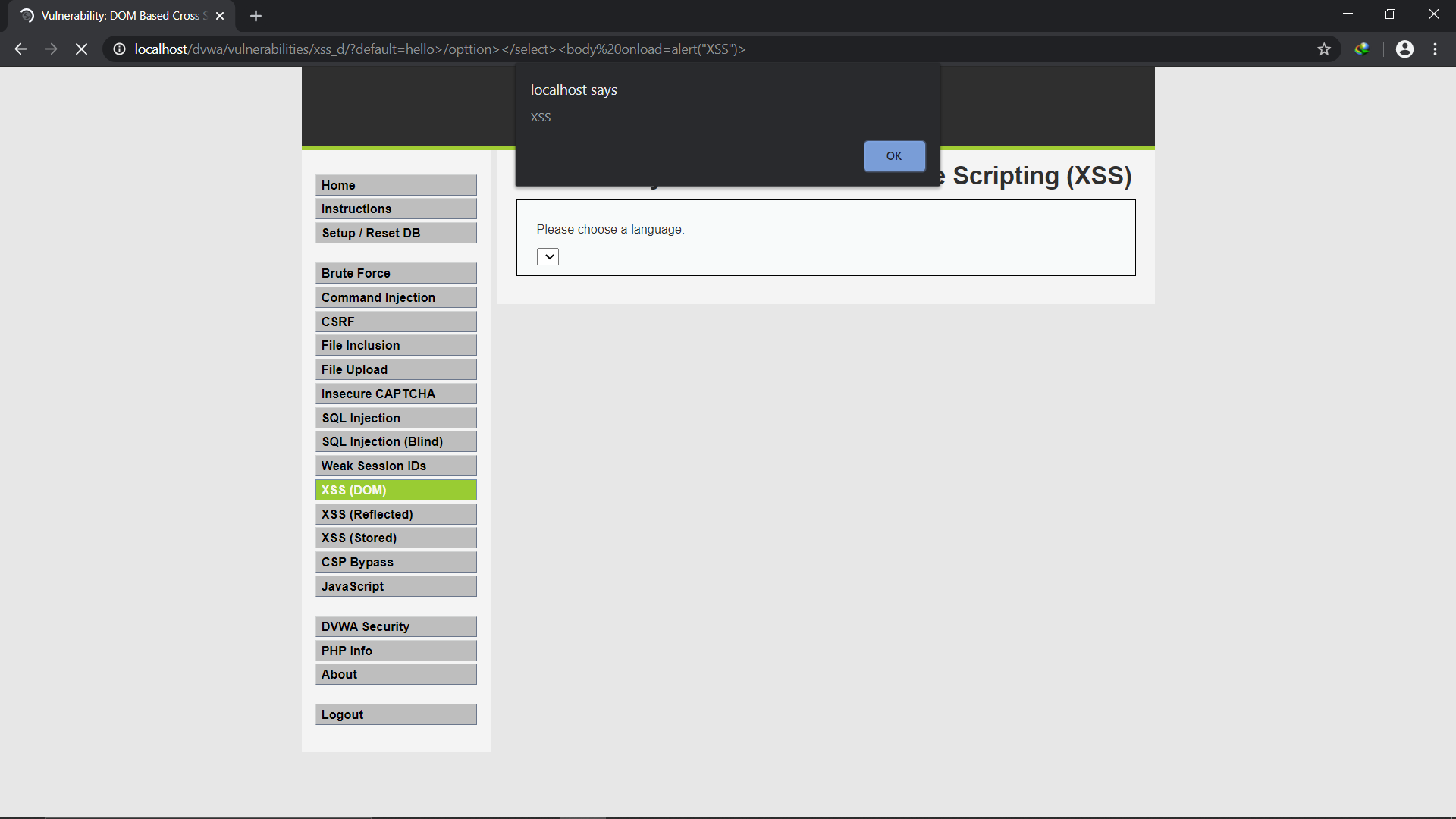


Figure 19

**Challenge 2(Stored XSS):**

* Change the size of name box by inspecting element.
* Enter payload *4* in the box and press enter.
* The alert will trigger.
* <img src=”#” onclick=alert(document.cookie) >

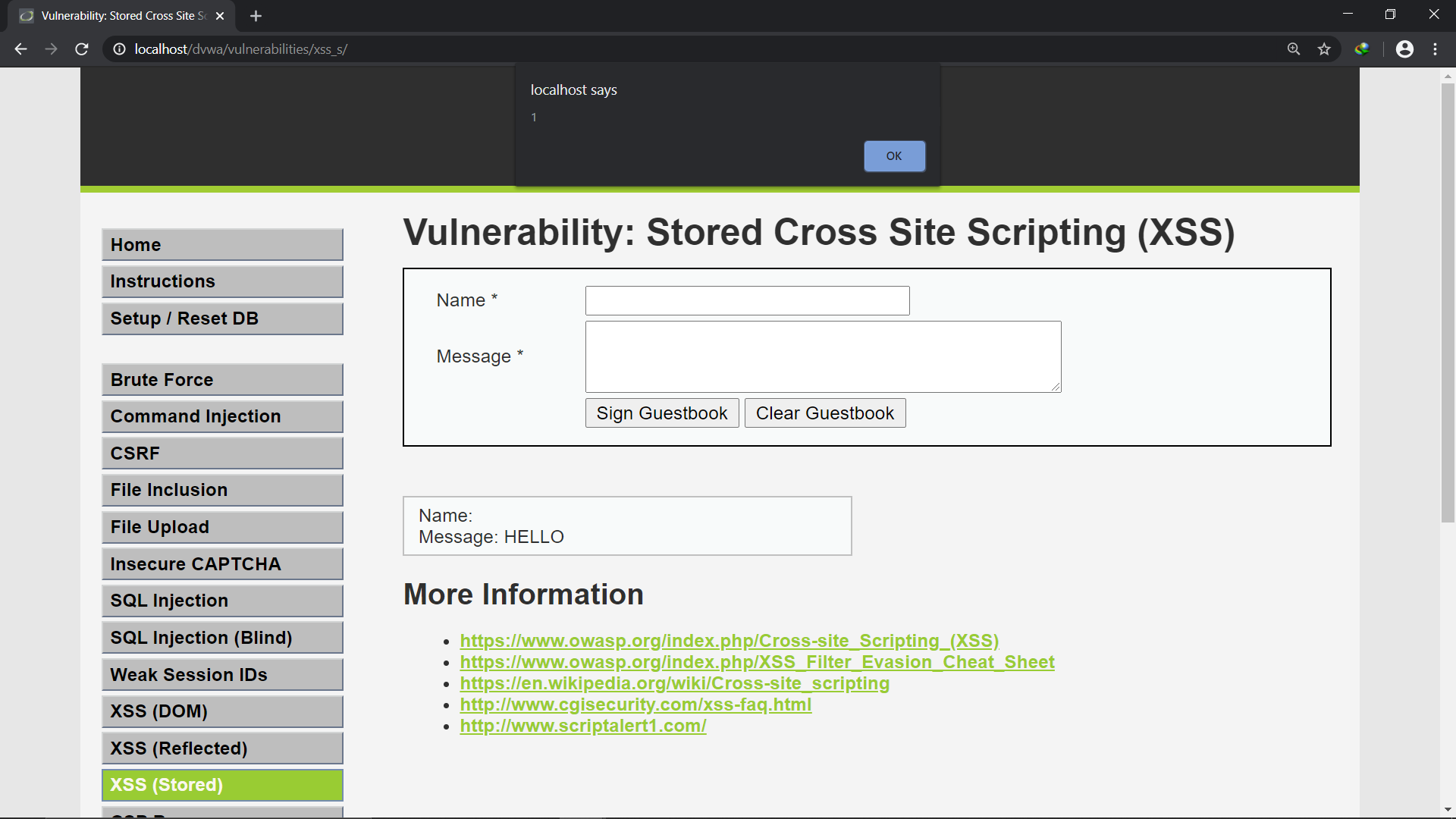


Figure 20

**Challenge 3(Reflected XSS):**

* Enter payload *5* in the box and press enter.
* The alert will trigger.

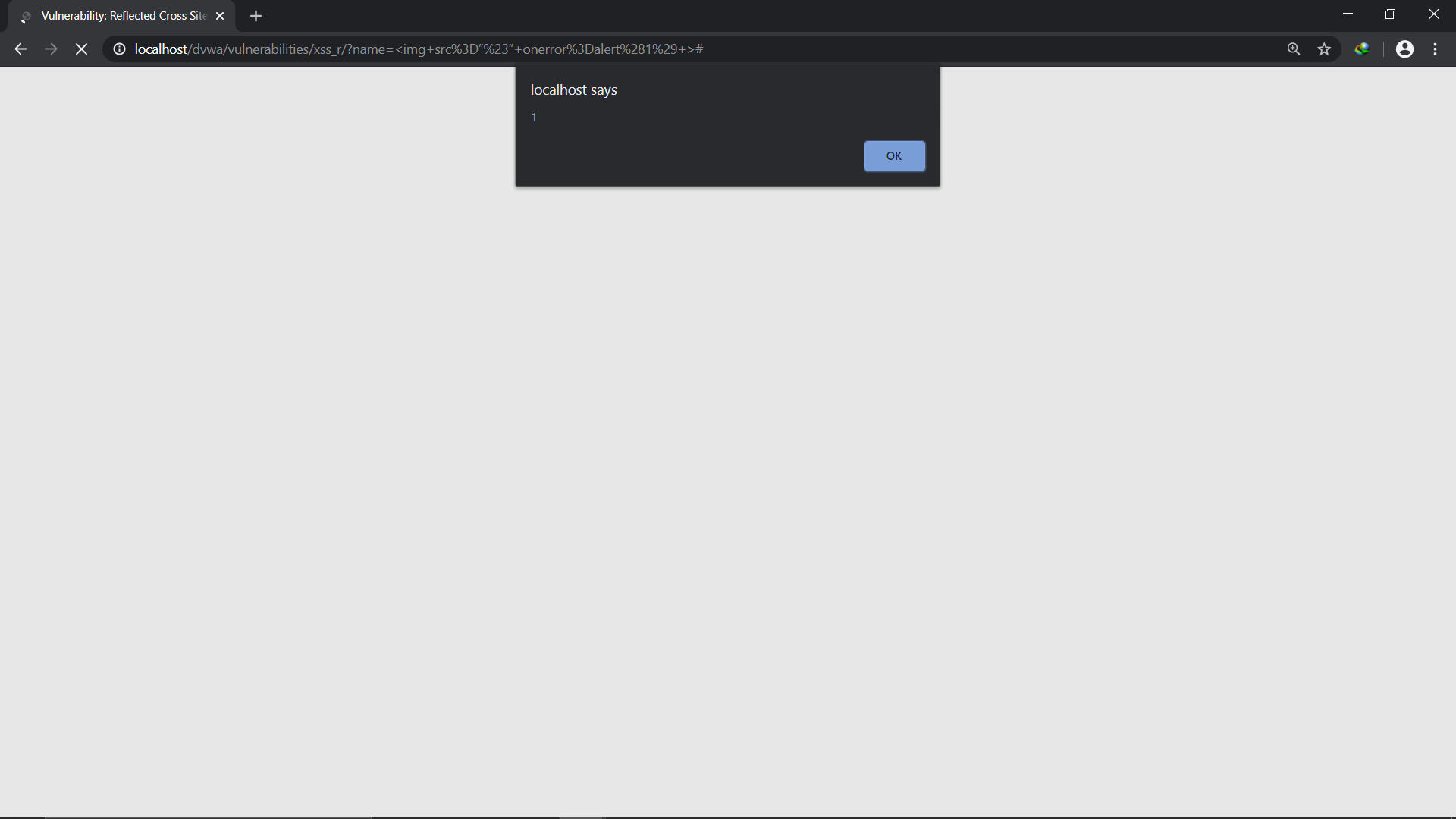


Figure 21

**Challenge 3(DOM XSS):**

* Enter payload *6* in the default parameter and press enter.
* The alert will trigger.

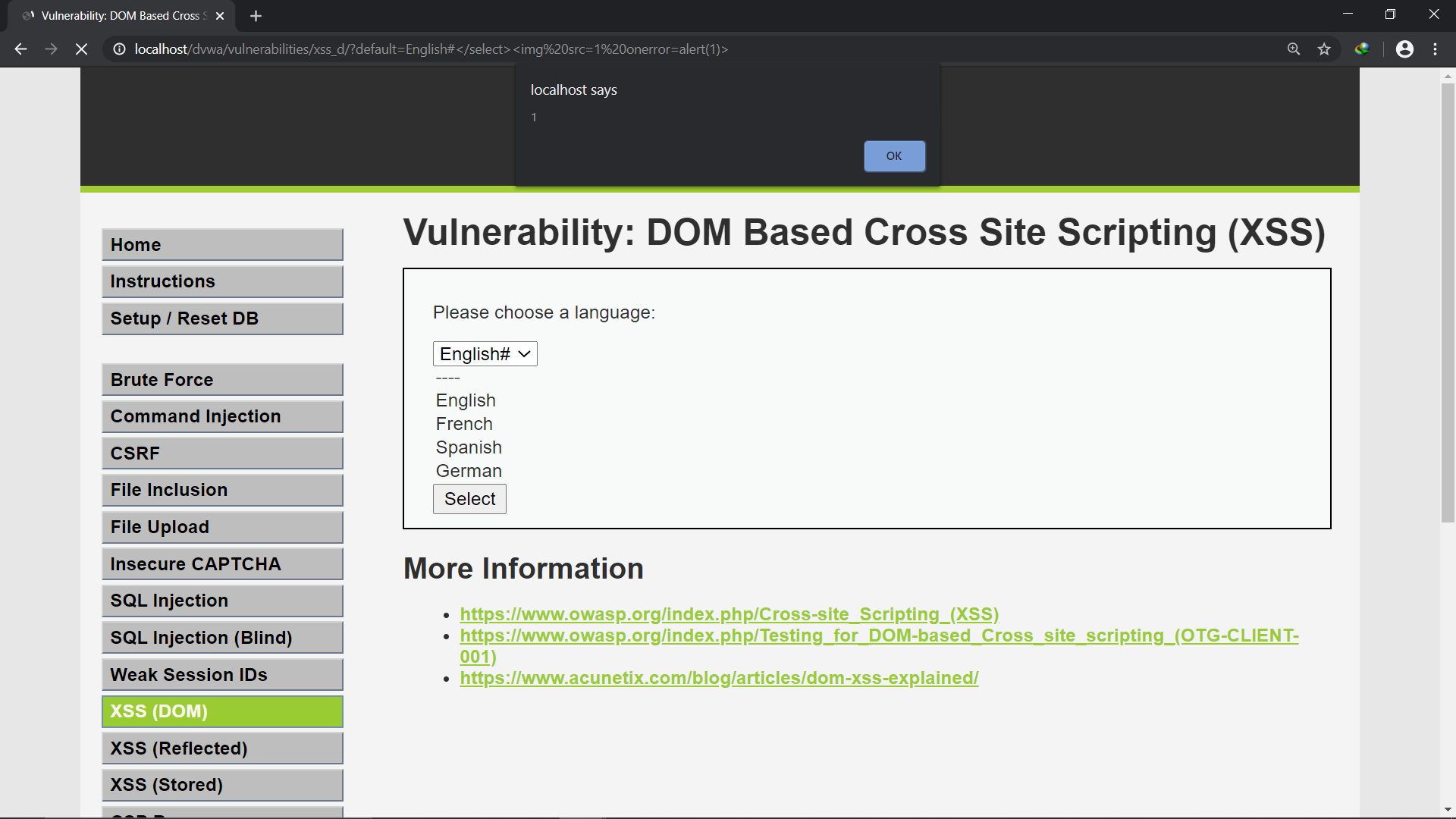


Figure 22

**Challenge 3(Stored XSS):**

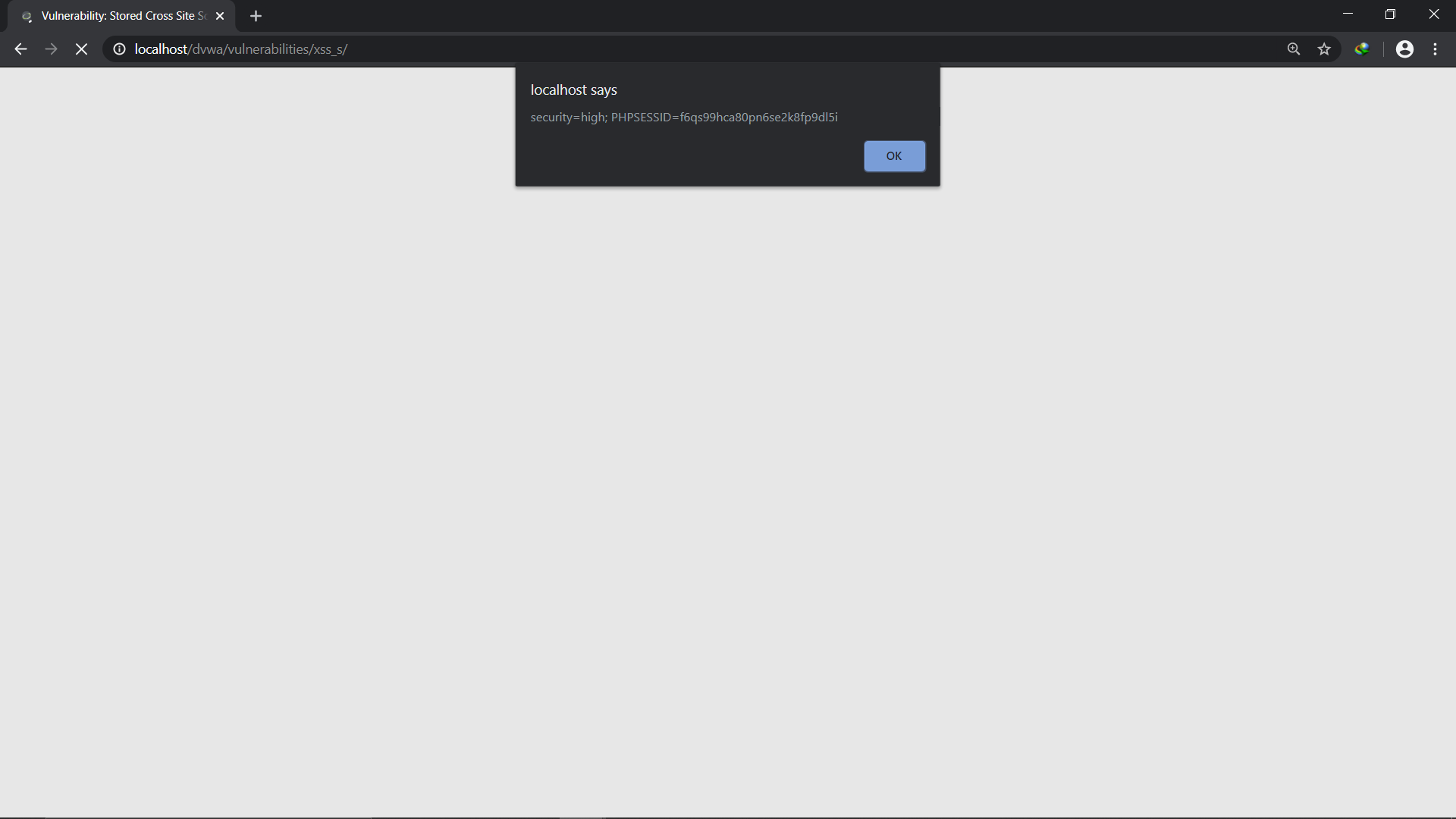
* Enter payload *7* in the name box by increasing its size with inspect element and press enter.
* The alert will trigger.

Figure 23

**CONCLUSION**

So, in the end I would like to conclude that every vulnerability over internet affects a lot to the respective application and XSS is one of those vulnerabilities which have affected internet the most. We shall use the minimum number of parameters as increasing the number of parameter means increasing the chances of vulnerability. The effect of XSS though also depends on what type of XSS it is and on what application it is getting tested. XSS is a versatile attack vector which opens the door to a large number of social-engineering and client-side attacks. After exploiting XSS vulnerability, it shows a clear result that we must analyze the content of web pages to ﬁnd patterns that allow identifying if their programming contains XSS type scripts. As a result of this, the executable content, the text ﬁltering rules, the string analysis and the web query analysis, are some of the proposals that we have found among the most common to mitigate attacks of XSS vulnerability. It can further be exploited more on live websites. As security consultants, we should do our best to explain the risks.

.

**REFERENCES**

* CCNA Cyber ops Course on Netacad.com website
* <https://portswigger.net/web-security/cross-site-scripting>
* ​<https://www.udemy.com/course/burp-suite/learn>
* https://www.geeksforgeeks.org/what-is-cross-site- scripting-xss/
* <https://owasp.org/www-community/attacks/xss/>
* <https://en.wikipedia.org/wiki/Cross-site_scripting>
* <https://www.geeksforgeeks.org/what-is-burp-suite/>