

Forum:[Hack Forums](#)

Thread: [\[Tutorial\] How to start with Web Pentesting Part 3: XSS \(Cross-site scripting\)](#)

Bodlux - July 12, 2020, 1:12 pm

Hey everyone

In this tutorial on web pentesting I will explain more about reflected and stored XSS (Cross-site scripting).

What is reflected XSS

At XSS we try to have our own client side script executed by the web server.

We do this by entering certain payloads in an input field or in the url.

These payloads usually consist of an HTML tag, an event and finally the script. In this tutorial we mainly discuss these kinds of payloads and as script we use "alert('XSS')". this script shows a popup message with the text "XSS".

What is stored XSS

The principle remains the same, trying to run our client side code through the web server, but there is a big difference.

With stored xss, the web server will remember the payload. This means that your script is executed by everyone who visits that web page.

An example where stored XSS would be possible is at a review page. You write "a review" for a product (it contains / is an XSS payload). The review is stored in a database and then all reviews of the database are loaded onto the page. In this way, your XSS payload is executed by everyone who reads that page.

At first sight this may not seem that special compared to a reflected XSS, but I will explain why this can be interesting.

In this way you can write a script that sends the cookie value of everyone who visits the page to you. This means that you can log in to others' account using their cookie. You could also capture keystrokes by using a keylogger. and so on ...

XSS payloads

As mentioned earlier, the payload consists of an html tag, an event and a script. This means that a basic knowledge of HTML and Javascript is required.

The most standard payload for XSS is as follows:

[XSS Img](#)

This is basically a simple html tag that execute the code between the tags.

You will rarely be successful with this payload because filters are often used. Therefore we have to use other tags and events to bypass these filters. below is an example of some "better payloads".

[Payloads \(Pastebin Link\)](#)

These are just a few examples with which you could bypass the filters. Note that sometimes (radom) capital letters are used, with this you can also bypass some filters.

Below you will find a link with a list of tags, events and even payloads.

[Cheat-Sheet Link](#)

XSS tools

I personally do not often use tools for XSS because I prefer to do it manually but below you will find some tools that you can use. Of course there are many more tools but these seemed the most interesting to me for the time being.

- XSSer
- Burp Suite

Burp suite itself is actually not intended for XSS alone. You can do a lot more interesting things with this tool, but I might discuss this in another tutorial.

The principle of the 2 tools is the same, you provide a URL and a list of XSS payloads. The tool will then send a request for each payload in the list to that specified web page (URL) with the payload as data. This is what you can do in the burp suite with the "Intruder".

For example:

URL = `http://localhost:8080/search?q=mySearchItem`

payload: `Img payload from pastebin link for example.`

In this case a request will be sent to `http://localhost:8080/search?q=Payload`. If the request is successful, it is checked whether the payload is present in the response. If this is the case, it means that the payload was successful. Otherwise the payload is not successful.

These kinds of tools are also very cool and educational to make yourself.

Thank you for reading my tutorial on XSS, if you have any questions feel free to ask and I will try to answer as best I can.