Ethical Hacker 7 – XSS and CSRF

# Part 1: XSS

* **What is cross-site scripting and how does it work?**

Cross site scripting is a security vulnerability found in some web applications. Cross site scripting attacks are a way for the attackers to inject client-side scripts into the web pages. Such a cross site scripting vulnerability can be used by attackers for a variety of purposes, such as bypassing the access controls. However, given the large amount of possibilities, there is a great variety in risk of these attacks depending of the sensitivity of the data handled.

Example of cross site scripting: first doing what I am supposed to do with this form to see the result:

Reflected Cross site scripting:

Graphical user interface, text, application

Description automatically generated

Typing <script>alert("The script in the user input works!")</script> in the textbox:

Graphical user interface, text

Description automatically generated

Typing <script>alert(document.cookie)</script> in the textbox:

Graphical user interface, text, website

Description automatically generated

These above are examples of injection via the input form.

Stored XSS: initial completion according to the purpose of the form:

Graphical user interface, text, application, email

Description automatically generated

Injecting script code in the message field:

Graphical user interface, text, application, email

Description automatically generated

Typing the text above will lead to this:

Rectangle

Description automatically generated with medium confidence

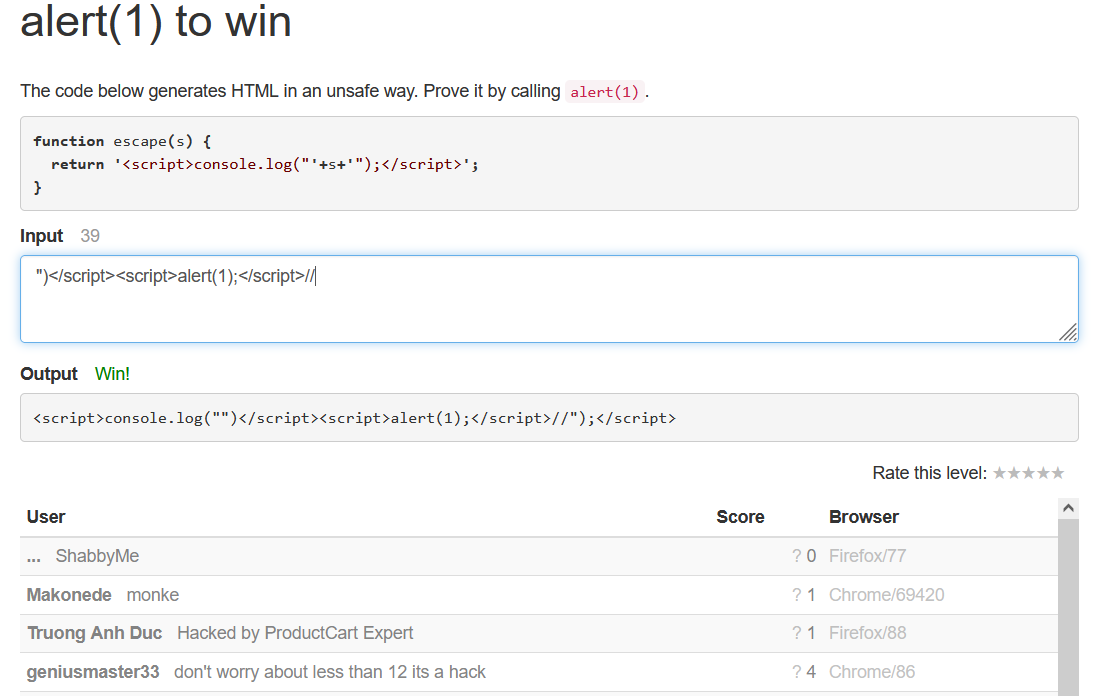
* **Explain the different types of XSS attacks: persistent versus non-persistent.**

The screenshots above are made for stored and reflected cross site scripting. As a matter of fact, persistent XSS is another name for stored XSS. This a more devastating sort of XSS vulnerability. It occurs when the data provided by the attacker is saved by the server and becomes a regular part of the web page, for every user to see. The screenshot above, showing the message “It works!” is displayed every time I am opening the respective DVWA page. It has been saved and it is an integral part of that page.

Non-persistent XSS or reflected XSS is the most common type of web vulnerability. It can be noticed when the data provided by the user, usually in a form, is taken and used by the server and displays the results immediately without properly sanitizing the content.

* **Give some other examples of XSS from Escape Alf.**

This example is very similar to an SQL injection attack. In the input field, the first characters to be typed in represent the characters in the actual code that mark the end of the input text. What is written after these characters is considered actual code and will be treated accordingly.



* **XSS exploitation possibilities conclusion: what could an attacker do if XSS is possible?**

There is a wide range of possibilities when it comes to XSS attacks. They could use persistent XSS to permanently modify the contents of a web page, at least until the code is cleaned up. The attacker could also use different XSS types in order to steal cookies, harvest credentials, force downloads, record audio or take pictures, and many more.

* **How to prevent XSS attacks?**

XSS filters need to be created in order to sanitize all the data the website receives. The main points of data entry that require sanitizing are the URL, the cookie data, the headers data and the database data, if not properly validated on user input.

# Part 2: CSRF

* **What is CSRF and how does it work?**

Cross site request forgery is a type of a malicious exploit of a web page in which unauthorized commands are submitted by a user that the web application trusts. It can be transmitted through hidden forms, specially crafted image tags, or javascript XMLhttp requests. This makes it possible to work without the user acknowledging it. In a CSRF attack the user is tricked by an attacker into submitting a web request they did not intend. This may cause actions to be made on the website that can include even client data leakage, change of session state or manipulation of an end user’s account.

The input data can be seen in the URL. This is a severe vulnerability, since the information is taken from that URL, so whatever is written there will be registered as valid.

Graphical user interface, text, application, email

Description automatically generated

If I modify the URL, I can change the password. Sensitive information such as the password in this case can be easily changed without authentication being required.

Graphical user interface, text, website

Description automatically generated

If I log in using the new credentials in the URL, it will work. The old password is no longer available. I have changed the user password using a CSRF attack.

A picture containing graphical user interface

Description automatically generated

* **Explain the different way of cross site request forgery attacks.**

There are several ways of CSRF attacks. The most common ones are modifying GET or POST statements. Before the attack, the attacker studies the website so the attack can blend in the code without causing any suspicions. Usually, for a GET statement, a <a> tag is modified in the HTML code, by embedding a bad request in a hyperlink. For a POST statement, a CSRF attack can be delivered by modifying code inside a <form> tag.

* **Resources:**
* [**https://resources.infosecinstitute.com/topic/how-to-prevent-cross-site-scripting-attacks/**](https://resources.infosecinstitute.com/topic/how-to-prevent-cross-site-scripting-attacks/)
* [**https://en.wikipedia.org/wiki/Cross-site\_scripting#Related\_vulnerabilities**](https://en.wikipedia.org/wiki/Cross-site_scripting#Related_vulnerabilities)
* [**https://en.wikipedia.org/wiki/Cross-site\_request\_forgery**](https://en.wikipedia.org/wiki/Cross-site_request_forgery)
* [**https://www.imperva.com/learn/application-security/csrf-cross-site-request-forgery/**](https://www.imperva.com/learn/application-security/csrf-cross-site-request-forgery/)