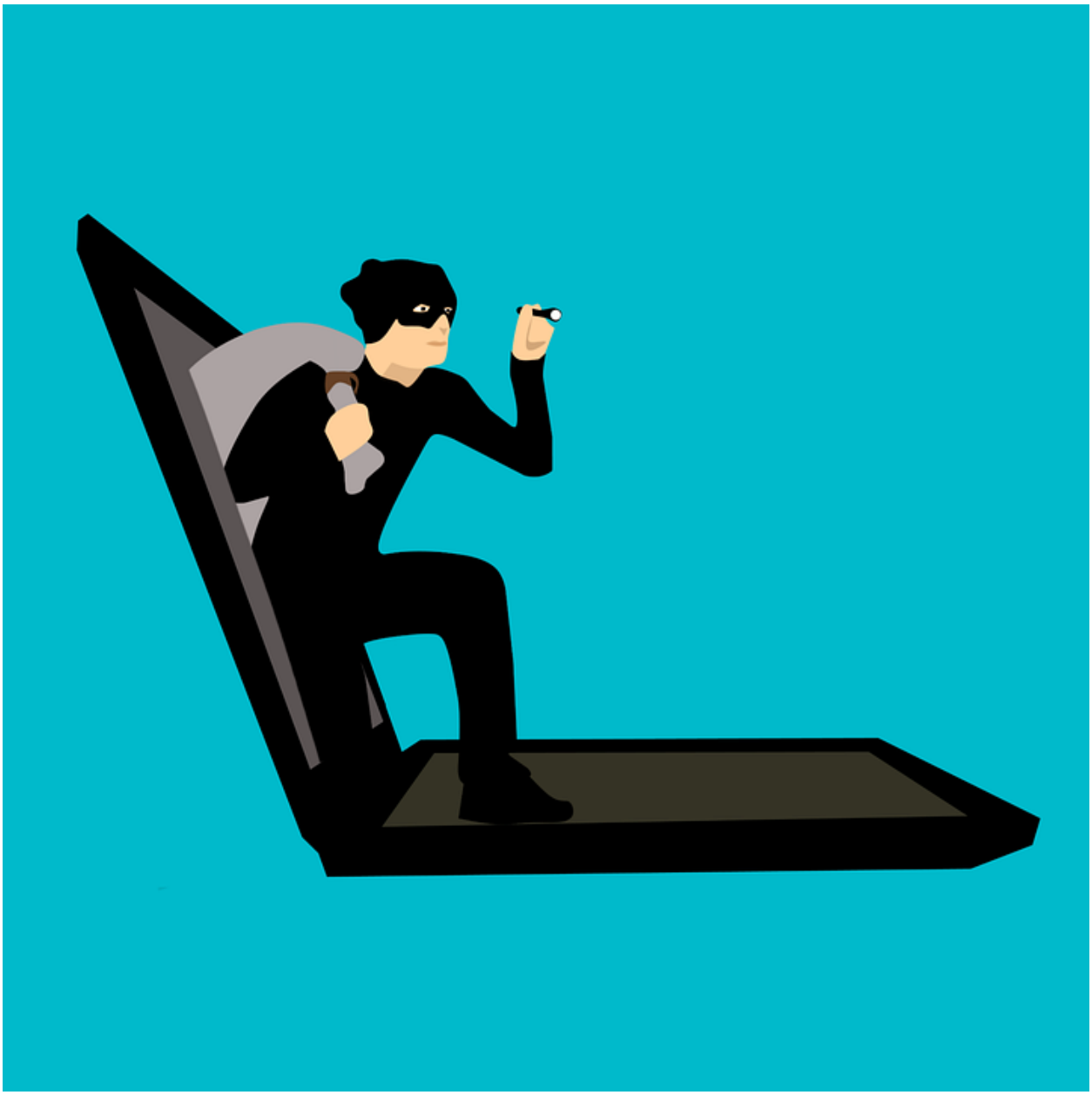
Basics of Cross Site Scripting(XSS) attack

Amit Cheke Follow
Apr 8 · 5 min read

This article has a very basic introduction to Cross Site Scripting(XSS) with some simple examples. I hope after reading this article you will start to take javascript injection seriously and will take necessary precautions while developing a web application.

What is Cross Site Scripting(XSS) & how it can impact the end user?



Cross site scripting(XSS) is a security vulnerability found in a web application which allows the attacker to inject client-side script(say javascript) to a webpage through input receiving areas like a search box, forms, file upload and execute it at user's end without any consent.

The severity of cross site scripting attack can range from showing the useless **alert box** to stealing **cookies, user's session ids & take over the account**.

. . .

Till now you might have understood that one can get attacked using javascript as well. Now let's dig into it, I will try to explain the things in a simple manner, with examples, instead of going in technical details of it.

XSS attack can be of two types **Stored and Reflected XSS Attacks.**

> ***Stored XSS Attack:*** *By taking advantage of flaws present in web-application, an attacker can store malicious javascript code into the target application. When a victim visits the affected web page in a browser, the XSS attack payload is served to the victim's browser as part of the HTML code (just like a legitimate comment would). This means that victims will end up executing the malicious script once the page is viewed in their browser. I have explained it using e-commerce example in later part.*

. . .

> ***Reflected XSS Attack:*** *In this type of attack, attackers use malicious links, phishing emails, and other social engineering techniques and force*

> *the victim to make a request to the server for example attacker may send the malicious link like:*

```
<a href="http://attack.co/steal.php">Click here to win a prize
</a>
```

> *When user click on such link and he reflected XSS payload is then executed in the user's browser.*

Here is one simple application, where the background color of the webpage will get changed as per the user's input. You can see that there is one input field of type text inside the form, which accepts user's input as a color name(Ex. red, black) and once a user submits the form, background color of the webpage will change to color entered by the user.

```
<h2>Update Background Color</h2>
<form onsubmit=" return changeColor()">
  Color:<br />
  <input type="text" id="bgColor" value="" />
  <input type="submit" value="Submit" />
</form>
<h3 class="search-query">
  Your selected color: <span id="query-output" class="query"></span>
</h3>
```

index.html hosted with ❤ by **GitHub** **view raw**

```
function changeColor() {
  var bgColor = document.getElementById("bgColor").value;
  document.body.style.background = bgColor;
  document.getElementById("query-output").innerHTML = bgColor;
  return false;
}
```

One can easily inject javascript into the above web page(*index.html*) by providing the following HTML snippet as input:

```
<img src onerror="alert('Your website is under XSS attack')">
```

Just copy above HTML snippet and paste it to the color input field and submit the form. You will be prompt with an alert box saying ***Your website is under XSS attack***

. . .

what extent it can cost the end user?

Now one can argue that how can it can have an impact on the user's security as it is injected by the user to its own web page?

So let's take an example of an e-commerce website. When you search for any product on the website, how you take a decision if that product is worth to buy or not? by going through the other buyer's review, right? now what if the attacker has submitted a review with javascript injection snippet, that review will be stored in website's database and when other users will visit that webpage of that product, all review along with attacker's review will get fetched & without your consent, the injected script will get executed and this way you will become the victim without doing any stupid activity(like clicking on link from spam mail, answer to untrusted alert box) .

This way an attacker can have access to cookies, session id. An attacker can inject a script that can send an email on his/her email account with the above details. This kind of attack is called **stored XSS Attack**

With the technical aspect, this attack could be described as:

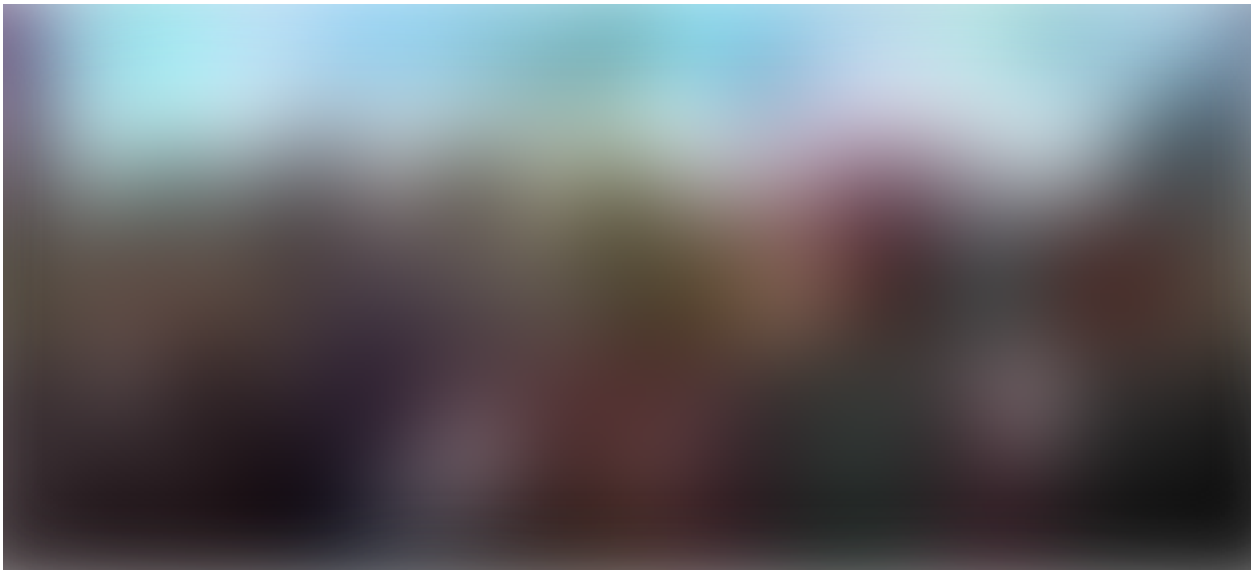
. . .

How to protect yourself?

It is difficult to find and remove XSS flaws from the web application, the best way is to keep eye on all places where the response of HTTP request could possibly make its way into the HTML output. For example, if your application is creating **href** dynamically as below:

```
<a href=”http://www.somesite.com?test=UNTRUSTED DATA FROM
RESPONSE OF HTTP REQUEST…”>link</a >
```

We are living in a world, which has more good people than bad



Such people always try to protect the world (web-world), few of them have developed open source tools like **Grabber, Nikto, Vega, Grendel-Scan, Wapiti, etc.** and you can use these tools to scan your web application and find cross site scripting vulnerabilities.

The Open Web Application Security Project (OWASP) is a worldwide not-for-profit charitable organization focused on improving the security of software. OWASP has developed the cheat sheet that you can consider while developing the web application or to fix the XSS vulnerabilities and you can find it at https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.md

Reference:

OWASP

There are thousands of active wiki users around the globe who review the changes to the site to help ensure quality.

www.owasp.org

Nurmuhammed3.png

From Wikimedia Commons, the free media repository

commons.wikimedia.org