Cross-Site Scripting

Estimated time needed: 10 minutes

In this reading, you will get an overview of cross-site scripting.

Objectives

After completing this reading, you will be able to:

- Describe cross-site scripting
- Recognize the different types of cross-site scripting attacks
- Infer the impact of cross-site scripting attacks

In the contemporary digital landscape, the term 'cyberattack' reverberates through a wide spectrum of devices from phones, laptops, and servers to cloud infrastructure. There are many ways in which cyberattacks can happen. In this reading, you will look at a client-side injection attack. An injection attack is when an attacker injects or interpolates an external script or code into an application to gain data control, breach privacy, or perform other malicious actions. Hackers can do this in various ways.

Cross-site scripting

Cross-site scripting is one of the top ten injection hacking techniques. A client-side script-based attack happens when the attacker finds a way to let the browser execute javascript's code, allowing the attacker to circumvent the same origin policy. Cross-site scripting manipulates a susceptible website, causing it to deliver harmful JavaScript to its users. Once this malicious code runs within the target's browser, the attacker can completely compromise the user's interaction with the application.

Suppose an HTML page accepts input that is not validated, and takes HTML code instead of regular text input. In that case, an attacker can abuse that to send a script payload through this vulnerable input, and then modify the application to do something it wasn't intended to do. In other words, the attacker could inject a malicious script, which would then get loaded by the browser, since the browser thinks that it's a necessary part of the webpage.

Let's say that you have a static website that contains only HTML and CSS. Upon accessing the website, the browser processes the HTML and CSS, resulting in the visual representation displayed on your screen. In the present day, the majority of websites incorporate interactive components. For instance, these sites retrieve data from a database, soliciting user input for various functions. One might encounter a form where users input information, and the application subsequently transmits this data to another page through the URL. This receiving page then extracts and processes the data embedded in the URL, capturing the user's submitted content.

Consider a scenario where this data fills a designated HTML field within the page. In this case, the data is extracted from the URL, undergoes parsing, and is subsequently displayed within the HTML content of the page. This effectively grants the user the capability to make modifications to the page. A potential attacker can exploit this situation by transmitting a script payload through the vulnerable input. Subsequently, they could manipulate the application to execute actions beyond its intended scope. To elaborate, the attacker could introduce a harmful script that the browser, interpreting it as integral to the webpage, would load to ensure the webpage's proper functionality.

Typically, malicious code adopts the disguise of relatively simple browser-oriented scripts. An attacker identifies a method to insert their evil code into the HTML document, while not infecting the actual web server. Instead, they employ the server as a vector to deliver harmful content to the user.

There are three types of cross-site scripting attacks:

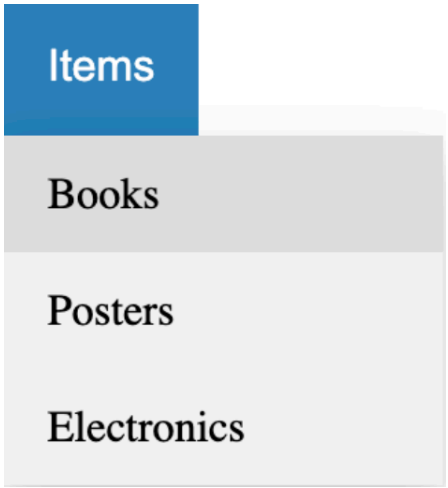
1. Reflected cross-site scripting attack
2. Stored cross-site scripting attack
3. DOM-based cross-site scripting attack

Reflected cross-site scripting attack:

Reflected cross-site scripting (XSS) occurs when an application takes data from an HTTP request and incorporates it into the immediate response through an insecure manner. The malicious script is delivered immediately upon the user's request.

An example is seen below:

Imagine there is a site that allows you to search for items based on categories.

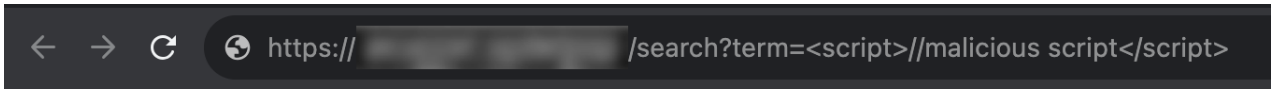


When you choose an item, for example, `Books`, the request is sent through the URL, which you may notice in the address bar.

A coded response looks like this:

It has been coded to display the items with the input you provided.

If the hacker were to enter any malicious script in the search text, it will be executed almost immediately in this case.



Stored cross-site scripting attack

A stored cross-site scripting attack uses storage and subsequent retrieval to postpone the malicious code attack. A stored cross-site scripting attack is also known as a persisted attack. Commonly, these attacks hinge on unfiltered user inputs that serve as entry points for scripts permanently residing on the target servers. Given that these exploits grant malevolent actors control over script execution within the browser, they often pave the way for a comprehensive takeover of user accounts.

Within reflected XSS, the server triggers the execution of hostile content, incorporating it solely within the immediate HTTP response. On the other hand, in stored XSS, the arbitrary code finds its storage place.

DOM-based cross-site scripting attack

DOM stands for Document Object Model. In this context, an HTML page is a document represented by an object comprising of one or more objects. In this attack, attackers inject a malicious payload into a web page by manipulating the client's browser environment. The point of attack, in this case, is targeting applications that provide a route for data to move from a source to a destination point. These sources encompass JavaScript properties capable of hosting malicious input, such as document.URL, document.referrer, location.search, and location.hash, among others.

Cross-site scripting today

The positive development is that modern web browsers continuously improve and incorporate numerous built-in safeguards to thwart successful XSS attacks. However, the drawback is that these browser security measures are not all-encompassing and cannot prevent every XSS attack. Although the prevalence of such attacks has diminished over time, they are still in the top ten cyber attacks, as per the OWASP - https://owasp.org/www-project-top-ten/.

Author

Lavanya T S