Introduction

Cross-site Scripting (XSS) is a client-side injection attack where the attacker tries to execute the malicious scripts in the victim's browser by injecting malicious payload in the legitimate web application. Every time the users accessing the web pages that are injected with the malicious script then the real attack will happen.

The web application becomes a means for delivering the malicious script to the user browser. Usually, the attacker will target the web application with forums, message boards, and web pages that allow comments, search boxes, input fields will be targeted by attackers to inject malicious payload for performing cross-site scripting.

At first, the attacker tries to find web pages that are vulnerable to cross-site scripting and tries to inject the malicious payload in the vulnerable pages whenever the user tries to load that page then the malicious payload will be executed in victim browser and JavaScript will access the cookies and sends to attacker and by using these cookies the attacker can impersonate the victim by using session hijacking attack as shown in figure 1

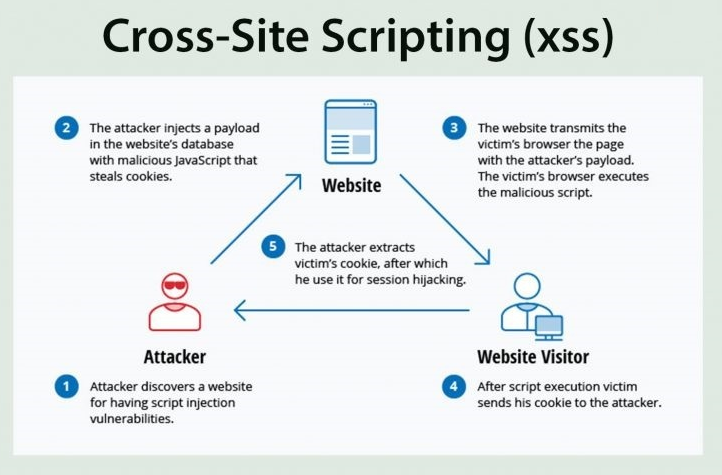


 Figure 1. Demonstration of XSS attack

By using the Cross-site Scripting the attacker may damage the website instead of targeting the user and the attacker can also use injected malicious scripts to change the content of the website and may even redirect to other web site or website with malicious contents Vulnerability is regarded to have less impact than SQL injection vulnerability. At first, the consequences of the ability to run JavaScript on a web page might not seem severe. Because most modern web browsers run java scripts in a controlled environment and have limited access to the user's OS and files.

But if JavaScript is used as part of malicious content, it can still be dangerous as Malicious JavaScript can also access to all objects that remaining web pages can access. Which includes access to the User cookies often used to store data related to the session? If an intruder succeeds in obtaining the session cookie of a user, the attacker can get access the user account and perform malicious action on behalf of the user and can access the sensitive information and JavaScript can use the XMLHttpRequest object to send arbitrary HTTP requests to destinations which can also use HTML5 APIs in modern browsers. For instance, gaining access to specific files from the user's file system to the geolocation, webcam, and microphone. Most of these APIs require granting permission from the user to execute in the browser, but the attacker may use social engineering to address that restriction.

the different types of cross-site scripting

**Reflected Cross-Site Scripting:**

In Reflected cross-site scripting, the attacker will inject the malicious script in the request and pass that request to the server then the server will process that request and sent back the response with injected payload in an unsafe way which will lead to the execution of script code in the browser. Reflected XSS is also known as non-persistent or TYPE II XSS. In reflected XSS the malicious script will be executed on the same place or page where it is injected so in generally the will be delivered xss to the victim by the other means like email messages with malicious links etc.when ever the malicious links are visited by the victim then the original attack will happen.

**Stored Cross-Site Scripting:**

In stored XSS the injected payloads will be saved in the effected servers such as database, message forum, server logs, comment field, etc stored XSS is also known as persistent XSS or type I XSS. In stored XSS all the users whoever visits the infected page will be effected sometimes admin can also be infected. the payload injected on infected pages will be executed continuously until it has been deleted from its source.

**DOM-based Cross-Site Scripting:**

DOM-based XSS is an advanced XSS attack. It is possible if the web application’s client-side scripts write data provided by the user to the Document Object Model (DOM). The data is subsequently read from the DOM by the web application and outputted to the browser. If the data is incorrectly handled, an attacker can inject a payload, which will be stored as part of the DOM and executed when the data is read back from the DOM [accuentix citation].

Dom based XSS has two main components they are sources and sinks, the sources are the location where the payload is being injected and the sinks are the location where the payloads are being injected.the most popular sources and sinks are given in the image below.

|  |  |
| --- | --- |
| sources | sinks |
| * document.URL * document.documentURI * location * location.href * location.Search * Location. Hash * Document.referrer * Window .name | * eval * setTimeout * setInterval * document.write * document.writeIn * innerHTML * outerHTML * location * location.href |

 A DOM-based XSS attack is a client-side attack because the malicious payload injected will be injected in client-side dom and shows response immediately so payload will not be sent to the server. which

makes dom based XSS more difficult to detect by the Web Application Firewalls (WAFs) or at the server-side.

 Impact of Cross-Site Scripting (XSS):

The impact of XSS on web applications will be minimal if there is no confidential information and no dynamic content change based on the user.

The impact will be critical on the web application that containing sensitive data, such as banking transactions, emails and health records, the impact will typically be significant and based on rights the compromised has for example If the compromised user has admin privileges then attacker can take full control over application and can get any kind of data

Session Hijacking

The most popular XSS attack vectors are stealing the victim's session cookies to hijack the victim's accounts. This enables attackers to impersonate victim account and access any sensitive data or features on behalf of victims.

Stealing credentials

The attacker will use HTML pages and JavaScript to steal customer credentials, instead of obtaining their cookies cloning the login page of the web application and then using XSS attacker steal credentials from the user. This situation is even more useful from an attacker's view, as they ultimately acquire plaintext credentials instead of expiring ephemeral session cookies.

Targeting Sensitive Data

Another strong XSS attack vector is to use it to exfiltrate sensitive data (e.g. private identifiable data or cardholder data) or to conduct unauthorized activities.

1. Key logger:

Using JavaScript, all keystrokes entered by a user on a vulnerable site can be logged. For this purpose, Metasploit involves an off-the-shelf payload. There are also some commercial websites offering JavaScript software that records all visitor motions, clicks, mobile gestures, or input forms that can be used for malicious reasons.

2. Port scan:

XSS is also an unexpected source for port scans to be initiated against a victim's internal network by accessing a vulnerable website.

3. Web site defacement:

Changing the visual appearance of a website vulnerable to XSS is one of the easiest and yet most efficient ways for attackers to target companies or public organizations. Either this can take organizations to the spotlight for the improper reasons by using embarrassing pictures or hacktivism messages.

Mitigations

the best way to mitigate the cross-site scripting is to properly sanitize the untrusted input is taken from input field, ids, URL parameters given by user, properly escaping the special characters in input and also output encoding the special characters while displaying in the HTML pages, the content security headers,httponly flag, **X-XSS-Protection Response Header can also be used are the features in modern js frameworks can also be to prevent XSS**