# Xss

# 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 be happens.

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 perform cross-site scripting attacks.

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

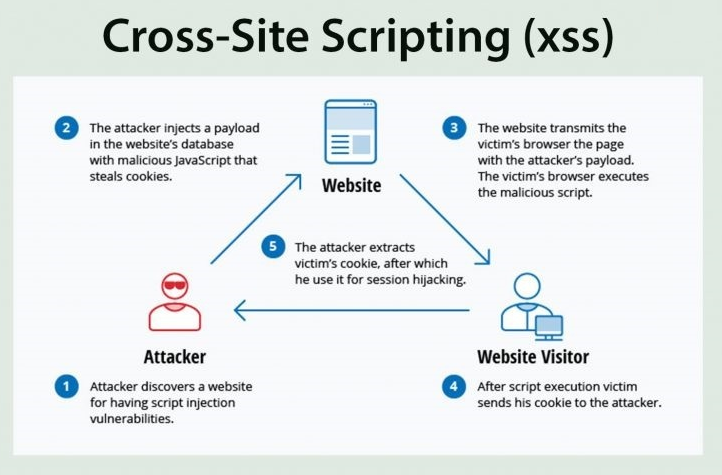


Figure 1. Demonstration of XSS attack

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 very tightly 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, they can impersonate that user and take action on behalf of the legitimate user and gain access to sensitive information of the user and JavaScript can use the XMLHttRequest object to send arbitrary HTTP requests to destinations. it can also use HTML5 APIs in modern browsers. For instance, gain access to specific files from the user's file system to the geo location, webcam, and microphone. Most of these APIs require opt-in from the user, but the attacker may use social engineering to address that restriction.

# Types of XSS attacks

As the main purpose of XSS attack is to execute malicious JavaScript in the victim's browser, and there are few fundamentally different ways of achieving that goal.

# ReflectedXSS:

In reflected XSS, the malicious string is part of the victim's request to the website.It might seem harmless asit requires the victim himself to actually send a request containing a malicious string. But attackers may trick the victims to send the malicious script without informing them. Whenthe attacker targets a specific individual or group theN attacker WILL send the malicious URL to the victim (using e-mail or instant messaging, or social networking link or link) and trick them into visiting link THEN AFTER vitctim visiting the link then attacker will steal confidential information stored in cookies.

# PersistentXSS:

In persistent/stored XSS,the malicious string originates from the website's database.It occurs when the data provided by the attacker is saved to the server, and then displaying on "normal" pages returned where proper HTML escaping which will dislayed to other users. Here, malicious code is inputted by attackersinto vulnerable web pages and is then stored on the web server for later use. The payload may be served back to other users browsing web pages and is executed in their context, at a later stage. Therefore, victims do not need to click on a malicious link to execute the payload (as in the case of Non-Persistent XSS); they just need to access the compromised web page, supplying user input from other web sessions that is not sanitized.

# DOM-basedXSS:

DOM XSS is a type ofcross site scripting attackwhich relies on inappropriate handling, in the HTML page, of the data from its associated DOM. Among the objects in the DOM, there are several which the attacker can manipulate in order to generate the XSS condition, and the most popular, from this perspective, arethedocument.url,document.locationanddocument.referrerobjects.

XSSattacks have been around years now and lot of research in the field has been done by Industry and academic experts.In literature, there are many methodologies, algorithms and techniques proposed in order to prevent XSS attacks. Analysis of XSSattacks reveal that they are caused due to improper coding of web applications and inability to filter or sanitize inputand encode the output. So, here known XSScountermeasures and mitigation techniques from various researchers are classified in phases of SDLC.

# 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 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. If the compromised user has admin privillages within the application, allowing the hacker to take full control of the insecure application and to compromise all users and their 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.

# 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.

# 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.

# 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

In order to minimize the risks associated with XSS, developers should encode all fields when displaying them in the browser. In addition, ensure that user input is filtered properly, particularly in the case of special characters. A common source of XSS are outdated third party libraries integrated in the code, and as such, update these to the latest stable versions. As part of a defence in depth strategy, ensure that cookie properties (such as HttpOnly) and [security headers](https://www.dionach.com/blog/an-overview-of-http-security-headers), especially CSP, are set accordingly.

On a higher level, ensure that security is properly integrated in all phases of the development process and that developers are aware of common web application vulnerabilities. Ultimately, regular penetration tests would help identify such flaws and improve the security stance of the web applications.