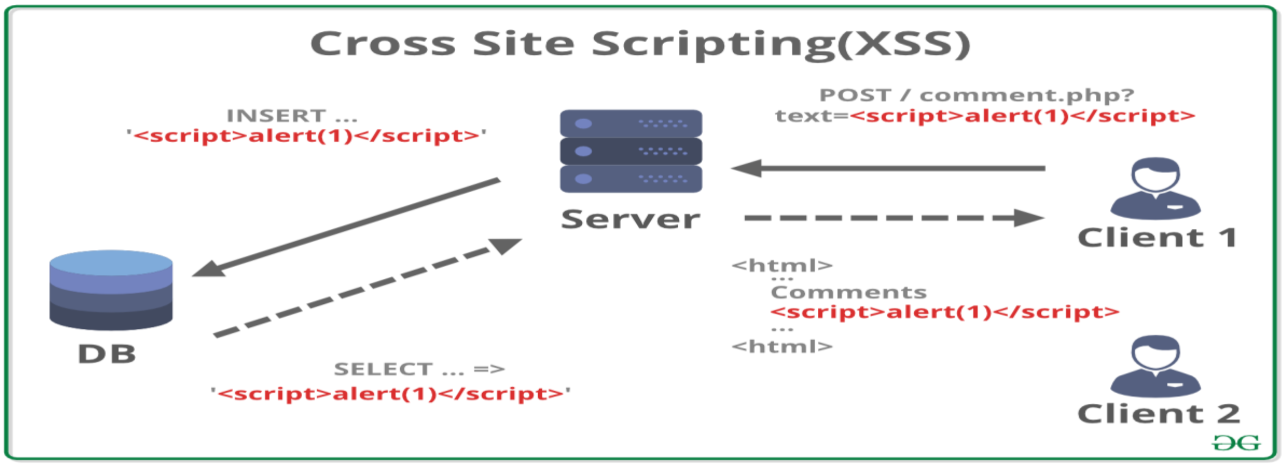
Cross-Site Scripting (XSS)

What is cross-site scripting (xss)?


Cross Site Scripting(XSS)
INSERT ...
'<script>alert(1)</script>'
Server
POST / comment.php?
text=<script>alert(1)</script>
Client 1
DB
SELECT ... =>
'<script>alert(1)</script>'
<html>
...
Comments
<script>alert(1)</script>
...
<html>
Client 2

Cross-site scripting (also known as XSS) is a web security vulnerability that enables an attacker to compromise users' interactions with a vulnerable application. It allows an attacker to circumvent the same origin policy, which segregates different websites from each other.

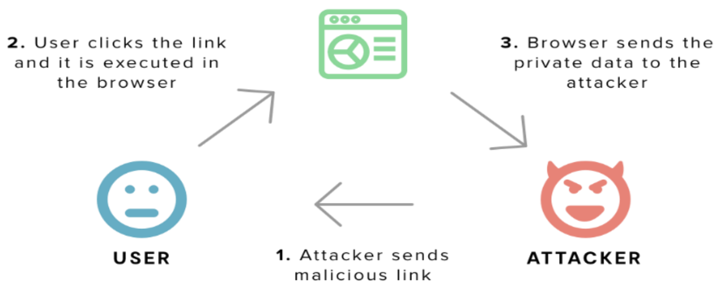
Cross-site scripting vulnerabilities normally allow an attacker to masquerade as a victim user, carry out any actions the user can perform, and access any of the user's data. If the victim user has privileged access to the application, the attacker might be able to gain full control over all of the application's functionality and data.

Types of cross-site scripting attacks (xss)?

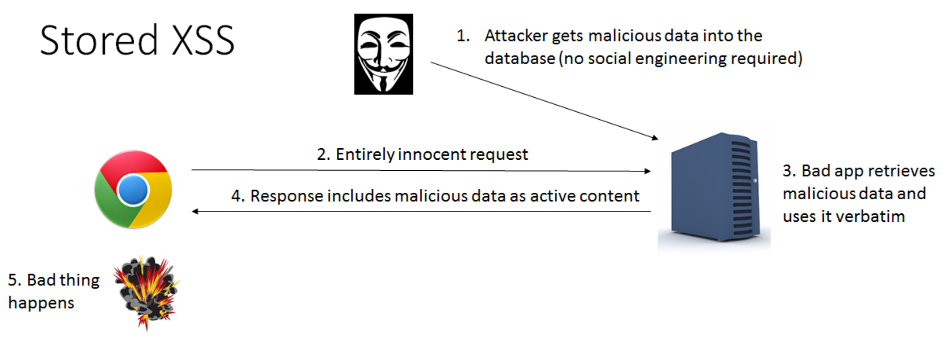
There are four types of cross-site scripting attacks.

1. Reflected XSS
2. Stored XSS
3. DOM-based XSS
4. Blind XSS

1. **Reflected XSS:** Reflected XSS (Non-persistent XSS) is the most common type of XSS, where the attacker's payload is part of the HTTP request sent to the web server and reflected back. Attackers use malicious links, phishing emails, and social engineering techniques to lure victims into making a request. The reflected XSS payload is executed in the user's browser, and it is not a persistent attack, requiring the attacker to deliver the payload to each victim. These attacks are often made using social networks.

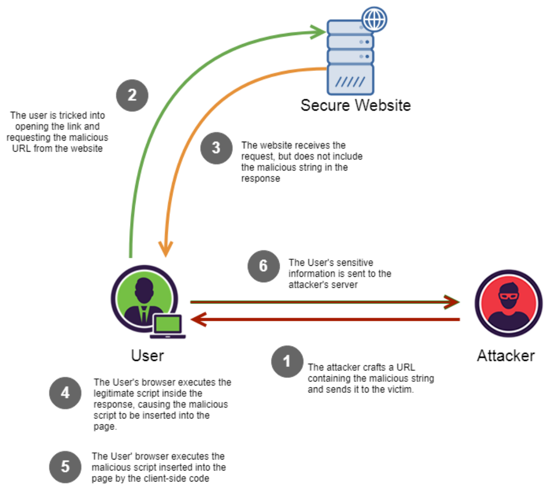

2. User clicks the link and it is executed in the browser
3. Browser sends the private data to the attacker
USER
1. Attacker sends malicious link
ATTACKER

2. **Stored XSS:** The Stored XSS (Persistent XSS) is the most damaging type of XSS, where an attacker injects malicious content, typically JavaScript code, into the target application. If no input validation is provided, the code is permanently stored within the target

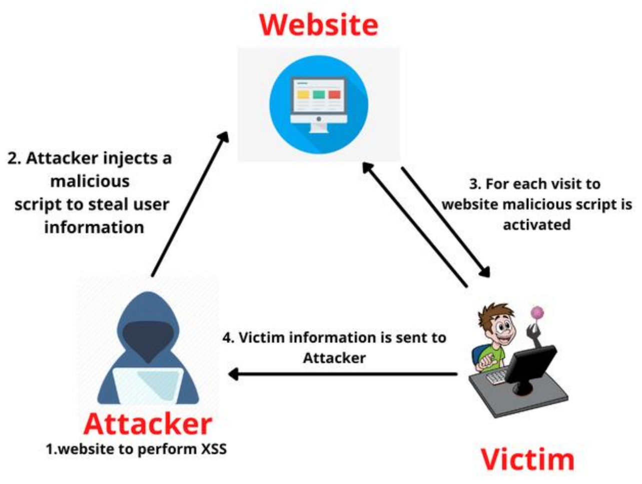

Stored XSS
1. Attacker gets malicious data into the database (no social engineering required)
2. Entirely innocent request
4. Response includes malicious data as active content
3. Bad app retrieves malicious data and uses it verbatim
5. Bad thing happens

application, such as in a database. When a victim opens the affected web page, the XSS attack payload is served as part of the HTML code, causing them to execute the malicious script. Reflected XSS (Non-persistent XSS) is another type of XSS that can be more damaging.

3. **DOM-based XSS:** DOM-based XSS is an advanced attack where a web application's client-side scripts write user-provided data to the Document Object Model (DOM), which is then read by the application and outputted to the browser. If the data is incorrectly handled, an attacker can inject a payload, which is stored as part of the DOM and executed when read back. This type of attack is often client-side and is difficult to detect for Web Application Firewalls (WAFs) and security engineers who analyze server logs. DOM objects most often manipulated include URL, anchor part, and referrer. Reflected XSS (Non-persistent XSS) is another form of XSS attack.


Secure Website
2 The user is tricked into opening the link and requesting the malicious URL from the website
3 The website receives the request, but does not include the malicious string in the response
6 The User's sensitive information is sent to the attacker's server
User
Attacker
1 The attacker crafts a URL containing the malicious string and sends it to the victim.
4 The User's browser executes the legitimate script inside the response, causing the malicious script to be inserted into the page.
5 The User' browser executes the malicious script inserted into the page by the client-side code

4. **Blind XSS:** Blind XSS is a flavor of cross-site scripting (XSS), where the attacker "blindly" deploys a series of malicious payloads on web pages that are likely to save them to a persistent state (like in a database, or in a log file). Blind XSS is a special type of stored XSS in which the data retrieval point is not


Website
2. Attacker injects a malicious script to steal user information
3. For each visit to website malicious script is activated
4. Victim information is sent to Attacker
Attacker
1.website to perform XSS
Victim

accessible by the attacker – for example, due to lack of privileges. This makes the vulnerability very difficult to test for using conventional techniques.

Impact of cross-site scripting?

Cross-site scripting attacks can have devastating consequences. Code injected into a vulnerable application can exfiltrate data or install malware on the user's machine. Attackers can masquerade as authorized users via session cookies, allowing them to perform any action allowed by the user account.

The reputation of a business can also be affected by XSS. The content of a corporate website can be defaced, the company's image damaged, or misinformation spread. The instructions given to users visiting the target website can also be altered by hackers, misdirecting their behavior. This scenario is particularly dangerous if the target is a government website that provides vital resources in times of crisis.

How to prevent cross-site scripting?

Preventing cross-site scripting is trivial in some cases but can be much harder depending on the complexity of the application and the ways it handles user-controllable data.

- **Filter input on arrival:** At the point where user input is received, filter as strictly as possible based on what is expected or valid input.

- **Encode data on output:** At the point where user-controllable data is output in HTTP responses, encode the output to prevent it from being interpreted as active content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding.

- **Use appropriate response headers:** To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, you can use the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend.\

- **Content Security Policy:** As a last line of defense, you can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occur.

Reference and Research

- https://portswigger.net/web-security/cross-site-scripting
- https://www.acunetix.com/websitesecurity/cross-site-scripting/
- https://brightsec.com/