**Faculteit Bedrijf en Organisatie**
Valentin Vaerwyckweg 1
9000 GENT

ACADEMIEJAAR 2019-2020
OPLEIDING TOEGEPASTE INFORMATICA

# Opdracht 5

Opleidingsonderdeel 'Projecten-workshops II: Systeembeheer'

# *What is Cross-Site Scripting, and how do we prevent it?*

**GROEP:** 09
**STUDENTEN:**
*Michiel Vanreybrouck*
*Dries Melkebeke*
*Nick Heymans*
*Quinten De Bruyne*

1 Table of contents

## 2 What is Cross-site Scripting

Cross-site Scripting, also known as XSS, is a client-side code injection attack. The attacker aims to execute malicious scripts in a web browser of the victim by including malicious code in a legitimate web page or web application. The actual attack occurs when the victim visits the web page or web application that executes the malicious code. The web page or web application becomes a medium to deliver the malicious script to the user's browser. Vulnerable mediums that are commonly used for Cross-site Scripting are forums, message boards, and web pages that allow comments.

A web page or web application is vulnerable to XSS if it uses unfiltered user input in the output that it generates. This unfiltered user input must then be parsed by the victim's browser. Cross-site Scripting attacks are possible in VBScript, ActiveX, Flash, and even CSS. However, they are typically used in JavaScript, primarily because JavaScript is fundamental to most browsing sessions.

## 3 How does Cross-site Scripting work

There are two stages to a typical Cross-site Scripting attack:

1. To run malicious JavaScript code in a victim's browser, an attacker must first find a way to inject malicious code, also known as the payload, into a web page that the victim visits.
2. After that, the victim must visit the web page with the malicious code. If the attack is directed at particular victims, the attacker can use social engineering and/or phishing to send a malicious URL to the victim.

For step one to be possible, the vulnerable website needs to directly include user input in its pages. An attacker can then insert a malicious string that will be used within the web page and treated as source code by the victim's browser. There are also variants of XSS attacks where the attacker lures the user to visit a URL using social engineering and the payload is part of the link that the user clicks.

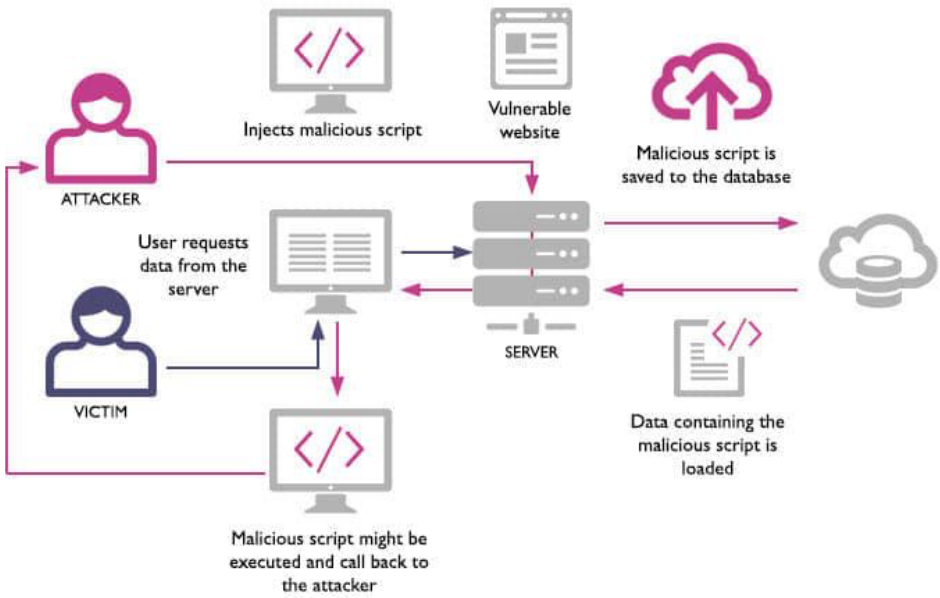


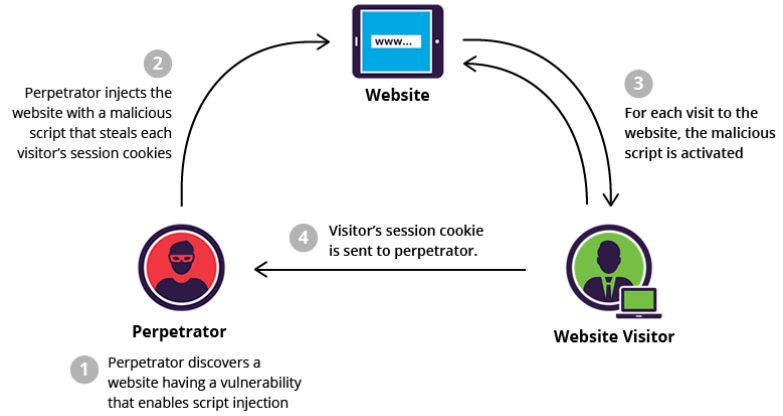

*Image source: https://snyk.io/blog/xss-attacks-the-next-wave/*

# 4 Different types of Cross-site Scripting vulnerability

XSS attack can generally be categorized into two categories: stored and reflected. There is a third, much less well-known sort of XSS attack called DOM Based XSS.

## 4.1 Stored XSS attacks

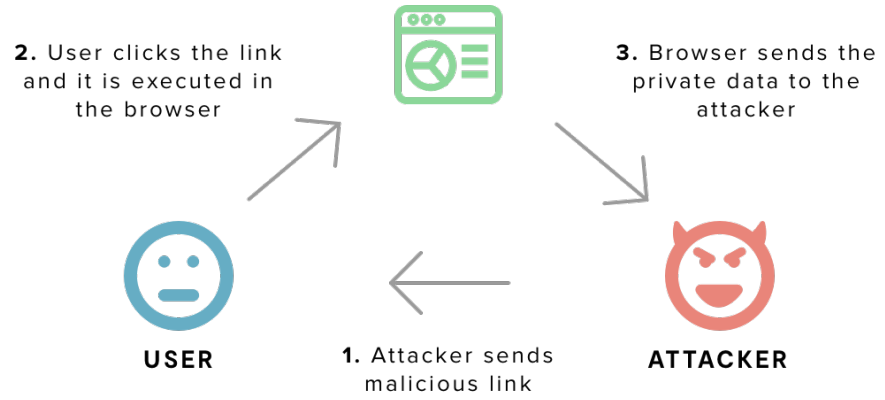
Stored cross-site scripting attacks occur when attackers stores their payload on a compromised server, causing the website to deliver malicious code to other visitors.
Since this method only requires an initial action from the attacker and can compromise many visitors afterwards, this is the most dangerous and most commonly employed type of cross-site scripting.
Examples of stored cross-site scripting attacks include the profile fields such as your username or email, which are saved on the server and displayed on your account page.
Stored XSS is also sometimes referred to as Persistent or Type-I XSS.




*Execution flow of Stored XSS attacks*

## 4.2 Reflected Cross-site Scripting

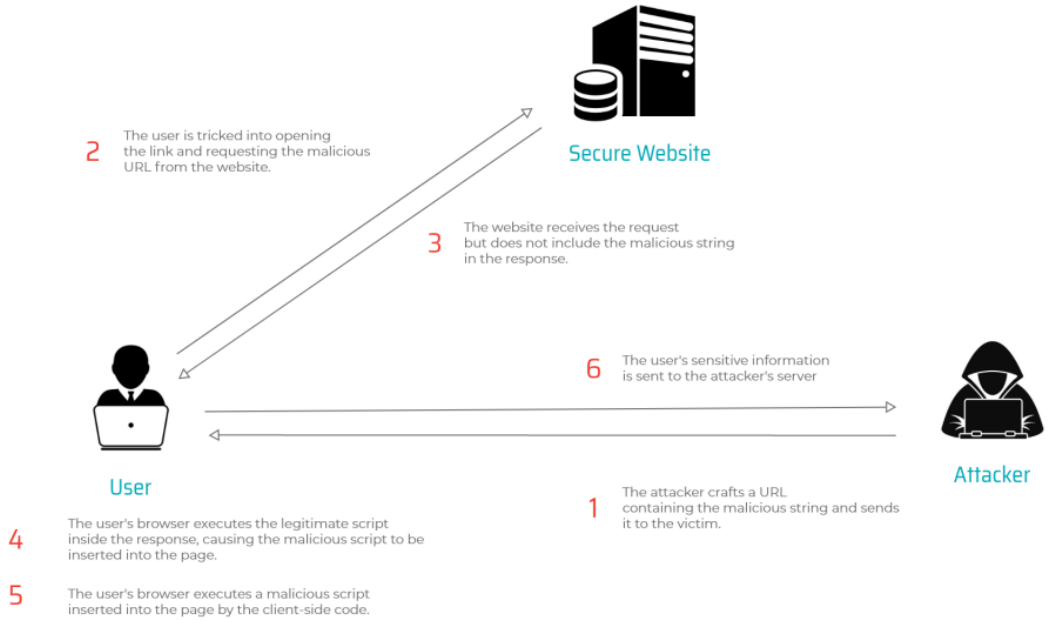
Reflected attacks are those where the injected script is reflected off the web server, such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request.
Reflected attacks are delivered to victims via another route, such as in an e-mail message, or on some other website. When a user is tricked into clicking on a malicious link, submitting a specially crafted form, or even just browsing to a malicious site, the injected code travels to the vulnerable web site, which reflects the attack back to the user's browser. The browser then executes the code because it came from a "trusted" server.
Reflected XSS is also sometimes referred to as Non-Persistent or Type-II XSS.




*Execution flow of Reflected XSS attacks*

## 4.3 DOM based XSS

DOM Based XSS, is an XSS attack wherein the attack payload is executed as a result of modifying the DOM "environment" in the victim's browser used by the original client side script, so that the client side code runs in an "unexpected" manner. That is, the page itself (the HTTP response that is) does not change, but the client-side code contained in the page executes differently due to the malicious modifications that have occurred in the DOM environment.
DOM based XSS is also sometimes referred to as Type-0 XSS.




*Execution flow of DOM-based XSS attacks*

# 5 Cross-site Scripting attack vectors

The following is a list of common XSS attack vectors that an attacker could use to compromise the security of a website or web application through an XSS attack.

## 5.1 <script> tag

The <script> tag is the most straightforward XSS payload. A script tag can reference external JavaScript code or you can embed the code within the script tag itself.

## 5.2 JavaScript events

JavaScript event attributes such as onload and onerror can be used in many different tags. This is a very popular XSS attack vector.

## 5.3 <body> tag

An XSS payload can be delivered inside the <body> by using event attributes (see above) or other more obscure attributes such as the background attribute.

## 5.4 <img> tag

Some browsers execute JavaScript found in the <img> attributes.

## 5.5 <iframe> tag

The <iframe> tag lets you embed another HTML page in the current page. An IFrame may contain JavaScript but JavaScript in the IFrame does not have access to the DOM of the parent page due to the Content Security Policy (CSP) of the browser.

## 5.6 <input> tag

In some browsers, if the type attribute of the <input> tag is set to image, it can be manipulated to embed a script.

## 5.7 <link> tag

The <link> tag, which is often used to link to external style sheets, may contain a script.

## 5.8 <table> tag

The background attribute of the <table> and <td> tags can be exploited to refer to a script instead of an image.

## 5.9 <object> tag

The <object> tag can be used to include a script from an external site.

# 6 XSS attack consequences

The consequence of an Cross-site Scripting attack is the same regardless of whether it is stored or reflected (or DOM Based). The difference is in how the payload arrives at the server. XSS can cause a variety of problems for the end user that range in severity from an annoyance to complete account compromise. The most severe XSS attacks involve disclosure of the user's session cookie, allowing an attacker to hijack the user's session and take over the account. Other damaging attacks include the disclosure of end user files, installation of Trojan horse programs, redirect the user to some other page or site, or modify presentation of content. An XSS vulnerability allowing an attacker to modify a press release or news item could affect a company's stock price or lessen consumer confidence. An XSS vulnerability on a pharmaceutical site could allow an attacker to modify dosage information resulting in an overdose.

# 7 How to prevent an XSS attack

## 7.1 Whitelist values

Restrict user input to a specific whitelist. This practice ensures that only known and safe values are sent to the server. Restricting user input only works if you know what data you will receive, such as the content of a drop-down menu, and is not practical for custom user content.

## 7.2 Avoid and restrict HTML in inputs

While HTML might be needed for rich content, it should be limited to trusted users. If you do allow styling and formatting on an input, you should consider using alternative ways to generate the content such as Markdown.

## 7.3 Sanitize values

When you are using user-generated content to a page, ensure it won't result in HTML content by replacing unsafe characters with their respective entities. Entities have the same appearance as a regular character, but can't be used to generate HTML.

## 7.4 Use HTTPOnly Flags on Cookies

Session cookies are a mechanism that allows a website to recognize a user between requests, and attackers frequently steal admin sessions by exfiltrating their cookies. Once a cookie has been stolen, attackers can then log in to their account without credentials or authorized access.
Use HttpOnly cookies to prevent JavaScript from reading the content of the cookie, making it harder for an attacker to steal the session.

## 7.5 Use a Content Security Policy

To mitigate the consequences of a possible XSS vulnerability, also use a Content Security Policy (CSP). CSP is an HTTP response header that lets you declare the dynamic resources that are allowed to load depending on the request source.

# 8 Bibliography

KirstenS. (2020). *Cross Site Scripting (XSS)*. Retrieved from OWASP: https://owasp.org/www-community/attacks/xss/

Acunetix. (2017). *Cross-site Scripting (XSS)* . Retrieved from Acunetix: https://www.acunetix.com/websitesecurity/cross-site-scripting/

Guides, S. (2019, November 7). *Cross-Site Scripting (XSS) Attacks*. Retrieved from Sucuri: https://sucuri.net/guides/what-is-cross-site-scripting/

Veracode. (2020). *Veracode*. Retrieved from CROSS-SITE SCRIPTING (XSS) TUTORIAL: LEARN ABOUT XSS VULNERABILITIES, INJECTIONS AND HOW TO PREVENT ATTACKS: https://www.veracode.com/security/xss

Team, N. S. (2019, April 18). *The Cross-site Scripting (XSS) Vulnerability: Definition and Prevention*. Retrieved from Netsparker: https://www.netsparker.com/blog/web-security/cross-site-scripting-xss/