A green chameleon logo

Description automatically generated

**CROSS -SITE SCRIPTING ATTACK**

​​

**TABLE OF CONTENTS**

1-SUMMARY……………………………………………………………………………………..

2-INTRODUCTION……………………………………………………………………………..

3- TOOLS USED………………………………………………………………………………..

4-SCOPE OF TESTING………………………………………………………………………….

5-RESULTS…………………………………………………………………………………………

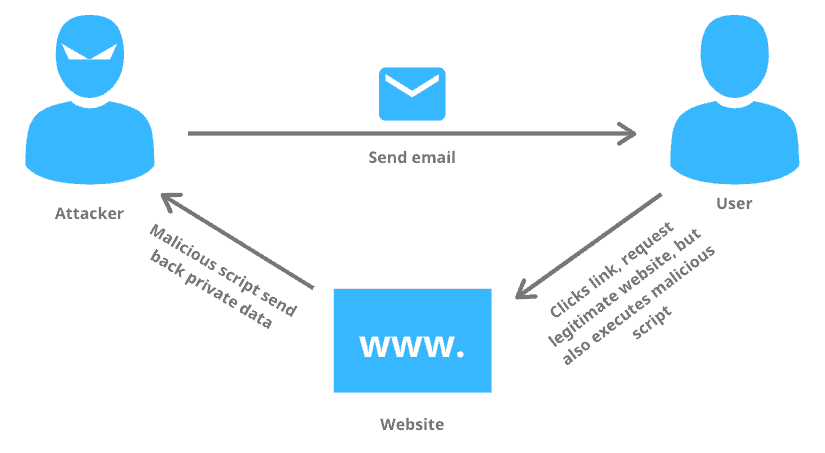
6-CONCLUSIONS……………………………………………………………………………….

7-REFERENCES……………………………………………………………………………………

**EXECUTIVE SUMMARY**

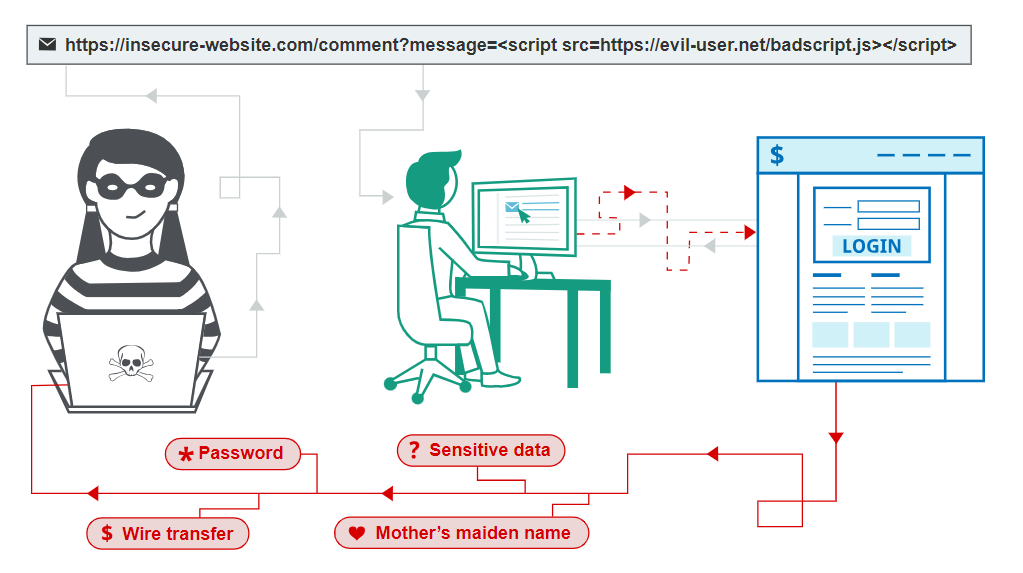
A web security flaw called cross-site scripting, or XSS, makes it possible for an attacker to influence how users interact with a susceptible application. It makes it possible for an attacker to get around the same origin policy, which separates various websites from one another. Via cross-site scripting vulnerabilities, an attacker can typically assume the identity of the victim user, perform any actions that the user can perform, and access any data that the user has access to. The attacker may be able to fully control all of the functionality and data of the application if the victim user has privileged access within it.

**INTRODUCTION**



An attack known as cross-site scripting (XSS) occurs when an attacker inserts malicious executable scripts into the code of a website or application that is supposed to be reliable. An XSS attack is frequently started by an attacker tempting a user to click on a malicious link that they send to them. The malicious link runs the attacker's selected code on the user's machine if the application or website does not perform adequate data sanitization. Thus, the attacker is able to take control of the user's active session cookie.

**WORKING OF CROSS-SITE SCRIPTING**

****

The method by which malicious JavaScript is returned to users via manipulation of a susceptible website is known as "cross-site scripting." An attacker can completely undermine a victim's ability to interact with an application when malicious code runs inside the victim's browser. Most XSS vulnerabilities can be verified by injecting a payload that triggers random JavaScript execution in your own browser. Using the alert() function for this purpose has long been standard procedure because it is brief, safe, and difficult to overlook when called successfully. In fact, using alert() in the browser of a fictitious victim solves most of our XSS labs. Sadly, there's a small issue if you use Chrome. Cross-origin iframes are not allowed to call alert() starting with version 92 (July 20, 2021). You may occasionally need to use a different PoC payload because these are used to build some of the more sophisticated XSS attacks. We advise using the print() function in this case. If you would like additional information.

**XSS DIFFERENT TYPES OF THE APPROACHES**

When a database contains the malicious payload, stored cross-site scripting (XSS) occurs. If there is no output encoding or sanitization, it renders to other users upon data request.

Reflected cross-site scripting attacks happen when a web application sends strings supplied by the attacker to the victim's browser, causing the browser to run a portion of the string as code. Since the payload lacks server-side output encoding, it echoes back in response.

DOM-oriented When a script is injected into a response, XSS occurs. To create a malicious URL, the attacker can read and modify the document object model (DOM) data. This URL is used by the attacker to deceive a user into clicking on it. The attacker can take advantage of the user's active session if they click on the link.

**STEPS TO AVOID CROSS SITE SRCITPING ATTACK**



Early on in the development life cycle of an application, security measures should be put in place. For instance, carry out security tasks like threat modelling and architecture risk analysis during the software design phase. After application development is finished, security testing should be carried out as well.

**XSS attack prevention techniques include-**

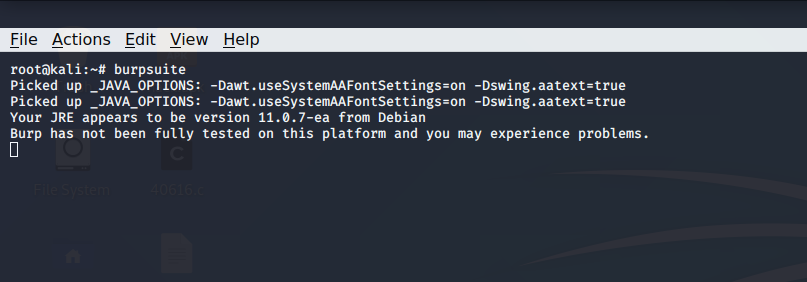
Never rely on what users have to say. As soon as you receive input from sources you don't trust, you should always validate and sanitise it. It is important to take into account both inbound and outbound input handling in order to offer thorough coverage.

Put output encoding into practice. This is the last step before writing data that can be controlled by the user. In order to ensure that the browser interprets output encoding as benign data rather than code, it escapes user input.

Adhere to the defense-in-depth strategy. To protect your most valuable assets, this strategy makes use of a range of security controls. Having several walls of defence (controls) guarantees that, in the event of a breach in one, there will still be other barriers preventing malevolent attacks.

Make sure that the OWASP XSS Prevention Cheat Sheet is followed when developing web applications. Here is a list of methods that have been proven to work in preventing XSS. As an XSS defence mechanism, OWASP suggests using a variety of techniques that can be tailored to your particular application. Conduct penetration testing in order to ensure that your high-risk XSS vulnerabilities are fortified.

**RESULTS**

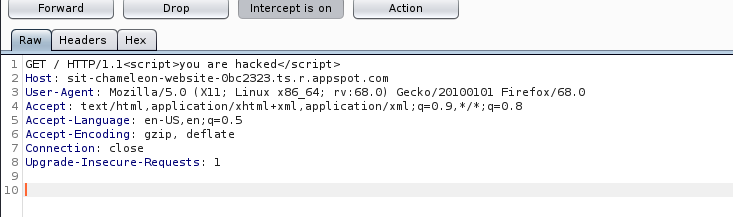


A screenshot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generated**

****

**A screenshot of a computer

Description automatically generated**

For the cross-site scripting attack I watched a lot of videos and did a lot research but didn’t succeed in the chameleon website, the steps which I took for this are as follows and I know what is cross site scripting which is by definition as Cross-site scripting (XSS) is an attack in which an attacker injects malicious executable scripts into the code of a trusted application or website. Attackers often initiate an XSS attack by sending a malicious link to a user and enticing the user to click it.

For this attack I used the burpsuite tool where I can monitor the traffic of the chameleon site as shown on the screenshots above, when I started monitoring the payload of the chameleon website in the proxy tab in the burpsuite I found many things as shown above which is get, host, user agent and many more things. Since we know that in the cross site scripting attack we have to inject the script in the JS so that the website is hacked so same I did for the chameleon website but I did not able to figure it out the correct script to enter and ended in the failure for this attack.

**CONCLUSION**

In summary, addressing XSS requires a multifaceted approach, combining secure coding practices, education, awareness, and the use of security tools. By prioritizing web security and adopting a proactive stance, organizations can significantly reduce the risk of XSS and enhance the overall resilience of their web applications.

References:

1- PortSwigger (2023). *What is cross-site scripting (XSS) and how to prevent it?* [online] Portswigger.net. Available at: https://portswigger.net/web-security/cross-site-scripting.

2-TCP FIN, NULL, and Xmas Scans (-sF, -sN, -sX) | Nmap Network Scanning

3- www.youtube.com. (n.d.). *Cross Site Scripting (XSS) tutorial for Beginners*. [online] Available at: https://youtu.be/bCP8\_WYsvP4 [Accessed 10 Dec. 2023].

4- www.youtube.com. (n.d.). *Cross-Site Scripting (XSS) Explained And Demonstrated By A Pro Hacker!* [online] Available at: https://youtu.be/PPzn4K2ZjfY [Accessed 10 Dec. 2023].