All you need to know about Cross Site Scripting (XSS)

Cross site scripting (XSS), it was introduced back in the year 2000 by Microsoft. One of the famous OWASP top 10 vulnerabilities. So what is it? Where can it occur? Is there a way to prevent it from happening? All these questions will be answered in this article.

XSS is a vulnerability that occurs when the web application allows hackers to execute custom code into the URL path or websites widely used by users such as: MySpace. This vulnerability can be exploit by running malicious JavaScript code on victim browser. Once the malicious code was activated, the user will either re-direct to malicious websites or sensitive data such as: session token and cookies will be hijack by hacker. Take a look on the *figure 1* for a better understand on how XSS works.

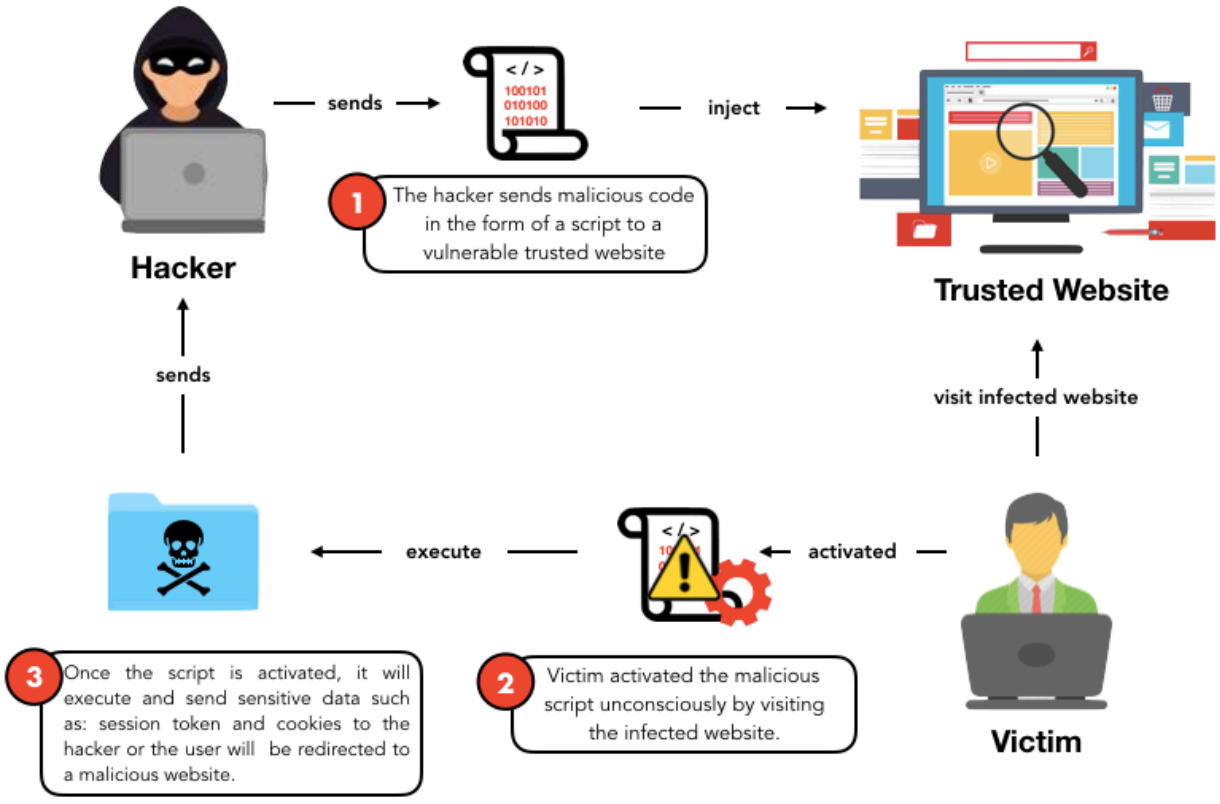

Figure 1 Cross Site Script (XSS) Process

There are 3 common forms of Cross Site Scripting vulnerabilities: reflected, stored & DOM-Based Cross Site Scripting (XSS).

Stored Cross Site Scripting

Stored XSS can happen if the username on a forum is not properly sanitized when it is printed on the page. The hacker can insert malicious code when registering as a new user on the register new account form. The injected script will be stored in the targeted forum database permanently. The malicious JavaScript will trigger when a user (victim) visits the forum. Sensitive data such as: cookies of the user will be sent to the hacker. The hacker can use this information to hijack user's sessions.

Reflected Cross Site Scripting

Reflected XSS can happen when the user input from a URL or POST data is reflected on the page without being stored. This will allow the hacker to inject malicious content. An example scenario: a hacker sends a script injected link to the victim via email. The victim is tricked into clicking a link in the phishing email. Next, the link will redirect to the victim browser and loads into legitimate site and activated the malicious script. The executed script will send victim sensitive data to the hacker.

DOM Based Cross Site Scripting

Document Object Model (DOM) XSS happens in document object model (DOM) instead of HTML. The malicious string is not parsed by the victim's browser until the website legitimate JavaScript is executed. An example scenario: a hacker sends a script injected link to the victim via email. The victim is tricked into clicking a link in the phishing email. The website receives the request but does not include the malicious string in the response. However, the victim browser executes the legitimate script inside the response. This will cause the malicious script to be inserted into the page with client side code. Lastly, victim's sensitive data will send to the hacker server.

Impact of Cross Site Scripting Attacks

The impact of being a victim of XSS attack on a web application can various to: user's session hijacking that leads to social engineering attack by disclosing sensitive data of the victim. This might lead to victim account taken over by the hacker. Cross Site Request Forgery (CSRF) attack and others security vulnerabilities by exploiting a XSS vulnerability.

Countermeasures

To prevent oneself from being a victim of XSS attacks, **trust-no-one**. Do no permit and execute untrusted input such a JavaScript from untrusted sources. Besides that, secure web application coding does prevent XSS attacks by using escape string and data sanitization to prevent malicious code from being inserted into the field. By following the basic security guidelines, you can avoid your web application prone by common vulnerabilities.

In conclusion, majority of the web application on the internet is vulnerable to XSS is due to bad security practices by both users and web developer. Highlights of best security practice by the web developer in his code are: filter input data, filter HTML & XML output and use API for XSS filtering. As for users, be vigilant and do not entertain phishing e-mails or e-mail send from untrusted sources.

References

OWASP , n.d. *OWASP Top 10.* [Online]
Available at: https://owasp.org/www-project-top-ten/

OWASP, n.d. *A7-Cross-Site Scripting (XSS).* [Online]
Available at: https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A7-Cross-Site_Scripting_(XSS)

Makarem, C., 2018. *Medium.* [Online]
Available at: https://medium.com/iocscan/dom-based-cross-site-scripting-dom-xss-3396453364fd