Collaborative Discussion 1: UML flowchart

Summary Post:

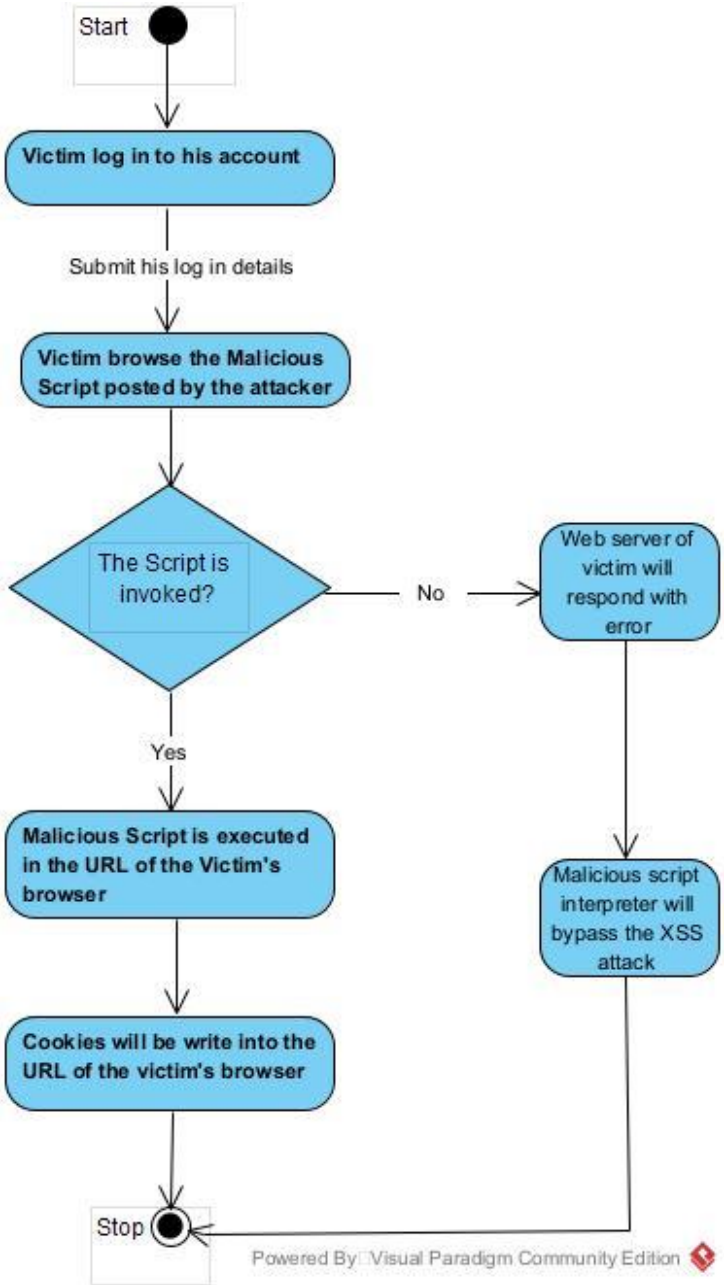
My previous post discussed Cross-Site Scripting (XSS) and its potential security risks. XSS is a type of vulnerability that occurs when malicious scripts are inserted into a trusted website or app, enabling hackers to steal data, take control of accounts, and execute harmful code on the user's browser (OWASP, N.D.). This issue arises when applications fail to validate, filter, or sanitise user-supplied data. To illustrate the weaknesses of XSS, I created a Sequence Diagram. However, my tutor suggested using an Activity Diagram, which visually represents a system's actions or control flow more effectively (Smartdraw, N.D.). I agree and have included an alternative activity diagram to demonstrate a malicious XSS attack based on the ideas of Gupta & Sharma (2012).

During a recent discussion, my colleague Sebastian raised an essential point regarding XSS vulnerabilities. Selvamani et al. (2010) found that careless use of the Document Object Model in JavaScript can create vulnerabilities where malicious code from another page can alter the principle of the first page on a local system. It is essential to understand these distinctions, as each one requires specific countermeasures. Sebastian ended his point with a relevant question: How can developers protect their applications against various XSS attacks? IBM (N.D.) suggests that to prevent XSS attacks, the application must validate all input data, only allow listed data, and encode all variable output on pages before returning it to the user.

I have commented on my colleagues' posts regarding cryptographic flaws and insecure design and implementation of APIs.

After analysing colleague Adesola's post on Cryptographic Flaws, I realised the importance of web application security. Further research revealed potential implications such as data breaches, loss of trust, legal and compliance issues, and intellectual property theft, as explained in Ali's research (2023).

My colleague Sebastien provided insightful information that helped me understand the importance of security in API projects. I learned more about RESTful, BOLA, and UBER API security incidents, illustrating the need for in-API security.



References:

OWASP (N.D.). *Cross-Site Scripting (XSS) Software Attack | OWASP Foundation*. [online] Available at: **https://owasp.org/www-community/attacks/xss/#:~:text=Cross%2DSite%20Scripting%20(XSS)**.

Smartdraw (N.D.). *Activity Diagram - Activity Diagram Symbols, Examples, and More*. [online] Available at: **https://www.smartdraw.com/activity-diagram/#:~:text=Learn%20More-**.

Gupta, S. & Sharma, L. (2012). *Exploitation of Cross-Site Scripting (XSS) Vulnerability on Real World Web Applications and its Defense.* Available at: **https://research.ijcaonline.org/volume60/number14/pxc3883594.pdf**

Selvamani, K., Duraisamy, A. & Kannan, A. (2010). *Protection of Web Applications from Cross-Site Scripting Attacks in Browser Side*. [online] arXiv.org. doi:https://doi.org/10.48550/arXiv.1004.1769.

IBM (N.D.). *Protect from cross-site scripting attacks*. Available at: **https://www.ibm.com/garage/method/practices/code/protect-from-cross-site-scripting/**.

Ali, Z. (2023). *Cryptographic Failures: Understanding the Pitfalls and Impact.* [online] Available at: **https://www.linkedin.com/pulse/cryptographic-failures-understanding-pitfalls-impact-zahid-ali/**.