**Portfolio Component**

**Programming Language Concepts**

Read Larson (2018) and Weidman (N.D.), then answer the questions below, adding them as evidence to your **e-portfolio**. You may want to complete this activity with or after Seminar 2 preparation.

1. What is ReDOS, and what part does ‘Evil Regex’ play?
2. What are the common problems associated with the use of regex? How can these be mitigated?
3. How and why could regex be used as a security solution?

You can share your responses with a tutor for formative feedback or discuss them in this week’s seminar.

**My answers:**

1. According to Weidman (N.D.), ReDoS, or Regular Expression Denial of Service, is a type of attack that preys on the vulnerability of many Regular Expression implementations. This type of attack can cause significant slowdowns in a program's operations, with the severity of the decelerations being directly linked to the size of the input. When a program utilising Regex encounters these conditions, an attacker can exploit them to cause the program to hang for an extended period.

   For instance, Weidman (N.D.) describes that using Regular Expressions can pose a risk if it encounters manipulated input. Threat actors can exploit this by sending precisely crafted information to trigger system failure or by introducing an Evil Regex to exploit its vulnerabilities. Regular Expressions are ubiquitous in all web layers and can be used to attack a web browser, firewall, database, or a vulnerable web server. If an Evil Regex is present in a Regex, it can be assumed that it is used on the server side, leading to server overload caused by the attacker.

   For example, Contrastsecurity (N.D.) states that an exponential worst-case complexity in regular expressions can result in lengthy evaluation times, leaving applications vulnerable to exploitation by malicious actors who craft corrupted or evil regex inputs. This phenomenon, known as catastrophic backtracking, can cause an application to become unusable due to an infinite search.

2. Regular expressions are a brief language used to manipulate strings in particular formats, such as dates, emails, and phone numbers. These expressions are frequently employed to search for and validate data and process information. Nonetheless, regular expression languages entail intricacies that may result in errors, such as unbalanced parentheses and some symbols with different meanings in various contexts. Verifying that regular expressions are accurate is imperative, as faulty ones could lead to mistaken behaviour or server crashes (Larson, 2018).

3. As Larson (2018) explains, regular expressions offer a type system that aids programs that recognise syntax errors in everyday words and invalid group numbers when extracting. This is advantageous in identifying and rectifying grammatical errors present in text-based sources such as web pages. Metacharacters, such as parentheses and vertical bars, may sometimes be inaccurately specified. Additionally, regular expressions are constraints for program analysis and bug detection tools.

**References:**

Weidman, A. (N.D.). *Regular expression Denial of Service - ReDoS* | *OWASP*.

[online] Available at: https://owasp.org/www-

community/attacks/Regular_expression_Denial_of_Service_-_ReDoS.

Contrast Security (N.D.). *ReDoS Attack*. [online] Available at:

https://www.contrastsecurity.com/glossary/redos-attack# [Accessed 10 Sep. 2023].

Larson, E. (2018). *Automatic Checking of Regular Expressions*. [online] Available at:

http://fac-staff.seattleu.edu/elarson/web/Research/acre.pdf.