# **What is ReDOS and what part do ‘Evil Regex’ play?**

Regular expression Denial of Service Attack Refers to the security issue that is faced when the program encounters undesired behaviour, through the malicious code input. In that particular example, ReDoS causes operating system to hang due to extreme situations. For example, one of the well known consequences of inappropriate use of regex is memory leakage which can cause the entire operating system getting corrupted. All these consequences are caused by the so-called evil Regex, that can be exploited to deliberatly overpass the otherwise protected patterns. In this particular example, the Evil Regex is considered as the one that contains the following characteristics (Weidman, n.d.):

* Grouping with repetition
* Inside the repeated group: Repetition, Alternation with overlapping

# **What are the common problems associated with the use of regex? How can these be mitigated?**

One common problem refers to the fact that they are runtime compiled, which means, that the errors are shown once the expression gets 'triggered'. The next two problems refer to the expression itself, as for example, ^ symbol may have different meanings in different situations. The second problem resides in characters that doesn't return an error. One of that type is brace brackets. In this example, they are not prone to raise an error, while unbalanced parenthesis would raise an error. Nevertheless, as stated by Larson (n.d.) it’s often a human error that results in a potential threat to the system. To mitigate all above issues, it is important to use the tools (for example ACRE) intended to automatically detect potentially harmful expressions (Larson, 2018).

1. **How and why could regex be used as part of a security solution?**

Regex can be used in all types of web communications. Another situation where regex is often used is validating the input data from various actors (it could be user, another system, external actors). All these security vulnerabilities expose viable solutions through the use of regex. To prevent these kinds of vulnerabilities, the regex expressions should be evaluated and constructed with high awareness, as the main thread lies in bad choice of patterns (Weidman, n.d.).

References:

Larson, E. (2018). Automatic Checking of Regular Expressions. IEEE Explore. Available from: <https://ieeexplore.ieee.org/document/8530738> [Accessed 6 June 2023]

Weidman, A. (n.d.). Regular expression Denial of Service – ReDoS. OWASP. Available from: <https://owasp.org/www-community/attacks/Regular_expression_Denial_of_Service_-_ReDoS#> [Accessed 6 June 2023]