**Multiple interpretari ale unei expresii regulate in diverse contexte**

Cuprins(schita):

1. Introducere:

* Ce reprezinta regex urile, scurta descriere a modului in care sunt folosite si cum sunt interpretate.

1. Regex Engines si algoritmi de potrivire

* Ce sunt Regex engine urile, care sunt algoritmii de protrivire ce stau la baza regex urile, o mica descriere a implementarii/proiectarii regex engine urilor, exemple de regex engines, de ce sunt ele utile.

1. Impacte negative sau efecte neasteptate ale modului actual de interpretare al expresiilor regulare

* Care sunt impactele negative in proiectarea regex engine urilor actuale, ReDoS si alte impacte de performanta

1. Alternative

* Care sunt alternativele care pot rezolva problemele evidentiale la capitolul precedent.

1. Concluzie

* De ce sunt mutiple interpretari, cum sunt exploatate aceste interpretari si cum se pot rezolva

+ Referinte

Ipoteza de lucru:

Expresiile regulare sau regex urile devin din ce in ce mai des folosite in diverse contexte/ platforme, iar corectitudinea lor este cruciala dpdv al functionalitatii si securitatii. Pentru aceasta regex urile scrise de programatori trebuie testate/ verificare iar cea mai buna optiune la ora actuala este <https://regex101.com/>, dar nu se ofera nicio garantie ca interpretarea va fi aceeasi si in Python, Java, C++ etc.

Posibile Experimente si Metodologie

Scrierea unor expresii regulate si testarea acestora folosind multiple limbaje de nivel inalt(Python, Java, C++ etc.) si/sau multiple platforme si compararea rezultatelor acestora.

Research questions:

De ce expresiile regulare pot avea multiple interpretari in functie de context/platofrme?

Care sunt efectele negative ale acestei multiple interpretari?

Care sunt alternativele pentru a preveni interpretarea multipla.

Referinte:

1. Xi Ye, Qiaochu Chen, Xinyu Wang, Isil Dillig and Greg Durrett (2020). Sketch-Driven Regular Expression Generation from Natural Language and Examples.
2. Ye, X., Chen, Q., Dillig, I., & Durrett, G. (2020). Benchmarking Multimodal Regex Synthesis with Complex Structures. *ArXiv, abs/2005.00663*.
3. Prakash M Nadkarni, Lucila Ohno-Machado, Wendy W Chapman . Natural language processing: an introduction .
4. Jun-U Park, Sang-Ki Ko, Marco Cognetta, and Yo-Sub Han. 2019. [SoftRegex: Generating Regex from Natural Language Descriptions using Softened Regex Equivalence](https://aclanthology.org/D19-1677). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6425–6431, Hong Kong, China. Association for Computational Linguistics
5. Nate Kushman & Regina Barzilay. Using Semantic Unification to Generate Regular Expressions from Natural Language.
6. D. Regéciová, D. Kolář and M. Milkovič, "Pattern Matching in YARA: Improved Aho-Corasick Algorithm," in IEEE Access, vol. 9, pp. 62857-62866, 2021, doi: 10.1109/ACCESS.2021.3074801.
7. P. Dlugosch, D. Brown, P. Glendenning, M. Leventhal and H. Noyes, "An Efficient and Scalable Semiconductor Architecture for Parallel Automata Processing," in IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 12, pp. 3088-3098, Dec. 2014, doi: 10.1109/TPDS.2014.8.
8. F. Yi, B. Jiang, L. Wang and J. Wu, "Cybersecurity Named Entity Recognition Using Multi-Modal Ensemble Learning," in IEEE Access, vol. 8, pp. 63214-63224, 2020, doi: 10.1109/ACCESS.2020.2984582.
9. E. Barlas, X. Du and J. C. Davis, "Exploiting Input Sanitization for Regex Denial of Service," 2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE), 2022, pp. 883-895, doi: 10.1145/3510003.3510047.
10. Shuyi Pei, Jing Yang, and Qing Yang. 2019. „REGISTOR: A Platform for Unstructured Data Processing Inside SSD Storage”. ACM Trans. Storage 15, 1, Article 7 (February 2019), 24 pages.
11. J. C. Davis, D. Moyer, A. M. Kazerouni and D. Lee, "Testing Regex Generalizability And Its Implications: A Large-Scale Many-Language Measurement Study," 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2019, pp. 427-439, doi: 10.1109/ASE.2019.00048.
12. Ehrenfeucht, Andrzej, and Paul Zeiger. "Complexity measures for regular expressions." *Proceedings of the sixth annual ACM symposium on Theory of computing*. 1974.
13. Chapman, Carl, Peipei Wang, and Kathryn T. Stolee. "Exploring regular expression comprehension." *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2017.
14. Li, Yeting, et al. "{ReDoSHunter}: A Combined Static and Dynamic Approach for Regular Expression {DoS} Detection." *30th USENIX Security Symposium (USENIX Security 21)*. 2021.
15. Berglund, Martin, Frank Drewes, and Brink Van Der Merwe. "Analyzing catastrophic backtracking behavior in practical regular expression matching." *arXiv preprint arXiv:1405.5599* (2014).