**[GROUP T] IMPROVING THE DEVELOPMENT OF ONLINE API’S WITH MACHINE LEARNING**

* [r0789762] Joris Machon
* [r0822842] Yeska Muhammad Omar

The security of online web applications is an important software development concern to assure privacy of users and security of company data. The many weak security tactics used by many developers are a huge problem in our society and economy causing many data leaks of personal information as well as company sensitive information.

The research in this thesis focusses on detecting and predicting specific API security vulnerabilities in online APIs due to software defects in the application-level access control. The number one security problem in online APIs is broken object-level authorization. APIs tend to expose endpoints to key business data and objects, but often forget to put the correct object-level authorization checks in every online operation or function that involves user input. Moreover, developers tend to expose all object properties without considering sensitivity. As such, application-level access control in online API’s is not enforced or not fine-grained enough.

The quality or state of software applications is often measured in terms of code metrics to assess the size, complexity, and maintainability of the software code. Some of the well-known metrics are lines of code, inheritance depth and cyclomatic complexity. In this thesis you will investigate some of the well-known application-level security vulnerabilities in open-source software on GitHub and assess a set of code metrics on the code base. A first step of the research is to analyze which code metrics contribute to the cause of these security problems or at least show a correlation with a specific security flaw, or a certain type of security flaw.

Moreover, todays applications are developed by agile development teams where many people iterate over the same code with different concerns and expertise. Software code artefacts and software components are developed and maintained by many people. In this thesis you will also investigate some of the well-known application-level security flaws in open source software on GitHub and analyze how the development process and development team contributes or correlates with these application-level security vulnerabilities. More specifically you will assess the development process by analyzing the GitHub repository in terms of number of commits, size of commits, size of the development team, number of people contributing to the problematic part of the software code, etc.

The overall research goal of the thesis is to train and employ machine learning techniques to detect problematic situations in the development of online web applications both at the level of the code, as well as at the level of the development team.

**Access control vulnerability detection:**

Barabanov, Dergunov , D., Makrushin , D., & Teplov , A. (2022). Automatic detection

of access control vulnerabilities via API specification processing

Rennhard, Kushnir , M., Favre, O., Esposito, D., & Zahnd, V. (2022). Automating the Detection of Access Control Vulnerabilities in Web Applications

Sun, F., Xu, L., & Su , Z. (2011, August). Static Detection of Access Control Vulnerabilities in Web Applications

**Further reading AI based approaches:**

N. Nagappan, B. Murphy, and V. Basili. The influence of organizational structure on software quality: an empirical case study. In Proceedings of the 30th International Conference on Software Engineering, ICSE’08, pages 521–530, New York, NY, USA, 2008. ACM.

Nachiappan Nagappan, Andreas Zeller, Thomas Zimmermann, Kim Herzig, and Brendan Murphy. Change Bursts as Defect Predictors. In Proceedings of the International Symposium on Software Reliability Engineering (ISSRE), 309–318. 2010.

Ghaffarian Seyed Mohammad and Hamid Reza Shahriari. "Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey." ACM Computing Surveys (CSUR) 50.4 (2017): 1-36.

Francesco Lomioa , Emanuele Iannone , Andrea De Lucia , Fabio Palomba , Valentina Lenarduzzic. Just-in-time software vulnerability detection: Are we there yet?. Journal of Systems and Software, p.111283. 2022.

Croft, Roland, Yongzheng Xie, and Muhammad Ali Babar. "Data Preparation for Software Vulnerability Prediction: A Systematic Literature Review." IEEE Transactions on Software Engineering (2022).

Triet H. M. Le, Huaming Chen, and M. Ali Babar. 2022. A Survey on Data-driven Software Vulnerability Assessment and Prioritization. ACM Comput. Surv. 55, 5, Article 100 (December 2022), 39 pages. <https://doi.org/10.1145/3529757>

E. Iannone, R. Guadagni, F. Ferrucci, A. De Lucia and F. Palomba, "The Secret Life of Software Vulnerabilities: A Large-Scale Empirical Study," in IEEE Transactions on Software Engineering, vol. 49, no. 1, pp. 44-63, 1 Jan. 2023, doi: 10.1109/TSE.2022.3140868.

R. Croft, M. Babar and M. Kholoosi, "Data Quality for Software Vulnerability Datasets," in 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), Melbourne, Australia, 2023 pp. 121-133.

Approach

1. Literature study and state of the art analysis:

* Literature study:
* Obtain a good overview of the scientific state of the art.
* But also explore important references in the basic reading papers.
* Learn about available data sources and data sets.
* Published vulnerabilities
* Published data sets
* Learn about available tools
* Slides from a summer school by Palomba is a good starting point after the basic reading list.
* Explains the process and the key papers

1. Read research proposal on defect prediction using developer profiles to understand the research goal and find a focus.
2. Formulate a focused research question for your thesis
3. Refine the approach

1. Potential practical experiments:

* A practical comparison of static analysis an dynamic analysis tools for access control problems in online APIs.
  + SAST tools: <https://www.appsecsanta.com/sast-tools>
* Build your own data set:
  + Build advanced automated dataset download tool
  + Git-based, developer-centric, and beyond code itself.
* Train your own model
  + And evaluate on accuracy and recall.