WASP Software Engineering Course Module 2025

Samuel Kajava

1. Introduction

My research area is, in its broadest sense, language-based security. Therefore, anything including programming languages is on the table. My current research focuses on prototype pollution in JavaScript, a vulnerability that allows attackers to insert arbitrary properties into the global prototype object. We aim to detect such vulnerabilities by utilizing a combination of static and dynamic analysis in order to both identify and verify vulnerabilities that may lead to cross-site scripting (XSS).

Another topic that is relevant to my area is security involving LLMs: *how can LLMs aid us in writing secure code?*, and *how can we secure LLM-based systems from prompt injection vulnerabilities?* are two interesting topics worth exploring.

2. Lecture principles

Confirmation bias

An interesting point brought up during the behavioral software engineering lecture was confirmation bias during debugging – a developer is tasked with tracking down a bug. This process seemed to be guided by the person's bias towards what part of the code base they might deem more likely to be "buggy". This makes sense, you have to start somewhere! However, what is deemed "buggy" is not for sure the culprit, and it may lead to wasted effort, since the problem might be somewhere entirely else (or they introduced it themselves).

Since my research involves a lot of bug-finding, it is only natural that this applies to me. My analysis is often inspired by previous contributions, which makes me look for certain patterns. However, we are also trying to find novel buggy patterns, and my bias towards what seems bugprone might affect this effort negatively. Doing security research requires some thinking out-of-the-box, and confirmation bias might slow down that process at times. However, I do not think it is entirely negative, I think this bias represents my "hunch" as well, which has proven to be correct more than once.

QA - Testing

The QA lecture provided an overview of why QA and testing is vital for software engineering. Testing is commonly used, and companies tend to ensure all software artefacts are thoroughly tested from unit tests to large scale system tests. While verification and validation of results are both very important aspects of my research, I feel like the value of testing throughout the whole process is often overlooked. While the projects are not as long-lived as the software in e.g., Ericsson's base stations, I still think there is value to ensuring quality throughout the development of a research project. Writing useful tests increases the likelihood of identifying mistakes, while also ensuring some kind of correctness (not formally, but your confidence in its functionality increases). In other words, utilizing QA and testing is vital for ensuring high-quality software – I think the same applies for research.

3. Guest-Lecture Principles

Requirements Engineering

On key point that stuck with me after Julian Frattini's lecture was to "understand the problem before you build the solution". By understanding the problem domain, identifying goals and defining what the problem is useful. I have a tendency to jump straight to coding when I start a project. The problem with this, as highlighted during the lecture, is that the idea of what I am trying to achieve is quite fuzzy. The problem here is that key components of the solution might be unclear before sitting down and defining a course of action. The consequences here is that you might have to refactor a lot of the code, or even start from the beginning. I strongly relate to this, and the lecture pinpointed the need to slow down before starting, which is something I am bringing with me for future endeavours.

Problem vs. Solution Space

Another point I found interesting from the lecture on requirements engineering was the contrast between problem space and solution space. When modeling a problem you are trying to solve, it is a good idea to formulate your requirements in terms of solution-space statements. An example from the lecture was "[t]he purpose of the website is to make static information available". There are many ways to provide a solution to this statement, and the one you are most comfortable with might not be the best way. In my research, I would have benefited from modeling my problem in this way, for sure: at the moment, I am building most of our software artefacts in Rust, which in hindsight seems overkill. A lot of the work could simply be done with Lua or Python, as the strong type system is not of too much value. We depend on external processes which may introduce many runtime faults, which makes the overhead of writing Rust code unnecessary. I very much appreciated the lecture, as it provided a solid foundation I can reason about in my research – I probably won't be using Rust for my next project (even if I enjoy the language).

4. Data Scientists versus Software Engineers

Differences between data scientists and software engineers

I agree that there are differences between data scientists and software engineers. The most important distinction, in my opinion, stems from the different goals that the two roles have. The introduction mentions that data scientists commonly aim for pushing the state of the art in controlled scenarios. For instance, training a model to beat previous ones in various performance metrics is interesting from an academic point of view – but it might be relatively expensive computationally; this does not directly translate to the goals of a software engineer. The goal for a software engineer is to, among other things, consider scalability and maintainability. Iterating on a massive model requiring re-training and large-scale deployment introduces additional challenges that might not be as relevant in data science.

Will the roles merge?

Regarding the question of whether the roles of software engineers and data scientists will merge or specialise further, I think the concept of "unicorn"-engineers is a good indication of what the future might look like. People with skills in both data science and software engineering is rare at the moment, and I believe it will stay that way. Data science requires a different skill set than software engineering, and it will attract different people for this reason. Rather than teaching data science to a software engineer (or vice-versa), I agree with the book that it is important to create interdisciplinary teams. So no, the "other side" will not have to learn the skills of the other. What needs to change, however, is how data scientists and software engineers collaborate in order to build good teams and, consequently, good products.

5. Paper analysis

Investigating the Impact of Solid Design Principles on Machine Learning Code Understanding

Core ideas and SE importance The core idea of the paper is to see if the source code of ML projects becomes more comprehensible when applying the SOLID principles [1]. They investigated this by performing a qualitative study where participants review the code of an ML project in two groups - the control group looked at the original source code of the project, and the experimental group observed a refactored version adhering to the SOLID principles. The results show that, indeed, the ability to understand the code increases when written with the design principles in mind. The authors point out that while the benefit of SOLID and other design principles are well studied in SE, no such work has been made in the realm of ML. Filling this gap is important, given the increasing amount of ML in production - long-lived projects need to be maintainable, after all, and is therefore important to study.

Personally, I do not think the results are very surprising, but what I do find important is that they surveyed data scientists from varying academic backgrounds. This is an interesting observation, since it shows how important standarized practices are for data scientists despite the different area of expertise they possess compared to a traditional software engineer.

Relation to my research While I do not research anything ML related, the data science aspects brought up in the paper applies to mine. The authors mention that the code for the control group is written in computational notebooks which they point out tends to "encourage poor coding practices" [1]. I perform data analysis in a similar fashion, and I agree that it quickly gets messy. If I were to be more strict in terms of coding practices, my results could get easier to reproduce.

Integration into a larger AI-intensive project A fictional project utilizes ML to generate proof-of-concept security exploits. It utilizes a RAG based on known vulnerable code snippets. The amount of different vulnerabilities quickly rises and needs to be represented in a maintainable way. By incorporating the idea of using the SOLID principles for these tasks makes it easier to extend in the future, especially considering that all vulnerabilities share a common interface - code snippets. I could incorporate my research into this by supplying the vulnerabilities I find on the web along with the insight I gain from learning how they typically look.

Adaption of my research As briefly mentioned above, adapting the SOLID principles in places where I would otherwise use quick notebooks is something I will keep in mind in the future. I think notebooks are useful, but they often get riddled with hacky solutions; if I apply the SOLID principles, the code will inevitably get more comprehensible, even if it is only me working on it.

Welcome Your New AI Teammate: On Safety Analysis by Leashing Large Language Models

Core ideas and SE importance The core idea of the paper by Nouri et. al [2] is to determine whether it is feasible to apply LLMs in safety engineering in terms of improved automation. To explore their idea, they task GPT-4 with creating "Hazard Analysis & Risk Assessment" (HARA) documents, which is important to complete before creating safety requirements in autonomous vehicles. Through prompt engineering and utilization of GPT-4, they conclude that it, indeed, is useful to apply LLMs in safety critical scenarios, but not without limitations. The outputs need to be verified by experts, and is mostly useful as grounds for brainstorming or as an early version that can be expanded.

The importance for SE is clear. LLMs can save time in SE settings by providing a starting point for engineers to expand upon. In terms of relevance to AI systems in general, providing insight into the use of LLMs in safety related domains is for sure a useful contribution.

Relation to my research In security research, similar to safety engineering, it is important to consider processes that handle challenges with risks and requirements. Like the paper generates HARAs, LLMs could be applied in security related areas as well. For instance, consider threat modelling or security requirements. Much like hazard analysis, tasks like these can be repetitive, and would benefit from automation.

Integration into a larger AI-intensive project A fictional cybersecurity company focusing on helping customers with assessing their security of their projects uses ML at the core of their product. The paper describes the prompt engineering effort used for creating HARAs in a structured manner [2]. It involves identifying the potential hazards of a given item, specifying scenarios were they may occur, expanding on those, evaluating the severity, before finally specifying goals. This effort is domain specific, but can serve as a modular way of solving similar problems. The fictional company can adopt this process on a per-customer basis, or choose to create a general solution. However, this does not resolve the issue of needing experts analysing the output, and this is where my research comes in. Given the wide scope of language-based security, bringing in threat-modelling research into this can help create a good product.

Adaption of my research Adapting the ideas presented in this paper is quite tricky since we are not incorporating anything AI-related into it. However, a similar effort [3] leverages LLMs (see [4]) for exploit payload generation, and could possibly be improved by applying the framework of prompt engineering presented in the paper.

6. Research Ethics & Synthesis Reflection

1. Search and screening process.

- Collect all papers in a list from the CAIN website.
- Read through the titles. Save the ones that suits the question (i.e., does it propose something that might be integrated in a project, and does my research somehow fit in it as well?)
- Read abstracts. Discard ones that do not fit the above criteria.
- If an abstract is deemed interesting, start reading the paper. The first two papers that were deemed interesting where chosen.

2. Pitfalls and mitigations.
    - None of the discarded papers were misleading per se, but e.g., "exploratory studies" all seemed irrelevant based on the criteria mentioned above. I decided to exclude such papers from my search.
3. Ethical considerations.
    - I did not use any LLMs for the contents of this essay.
    - Spell checking was done with the builtin solution for Neovim.
    - I did use an LLM for setting up the LaTeX preamble in the markdown document to get the correct output from pandoc.

References

[1] R. Cabral, M. Kalinowski, M. T. Baldassarre, H. Villamizar, T. Escovedo, and H. Lopes, "Investigating the impact of solid design principles on machine learning code understanding," in *Proceedings of the IEEE/ACM 3rd international conference on AI engineering-software engineering for AI*, 2024, pp. 7–17.

[2] A. Nouri, B. Cabrero-Daniel, F. Torner, H. Sivencrona, and C. Berger, "Welcome your new AI teammate: On safety analysis by leashing large language models," in *Proceedings of the IEEE/ACM 3rd international conference on AI engineering - software engineering for AI*, in CAIN '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 172–177. doi: 10.1145/3644815.3644953. Available: https://doi.org/10.1145/3644815.3644953

[3] Z. Kang *et al.*, "Follow my flow: Unveiling client-side prototype pollution gadgets from one million real-world websites," in *2025 IEEE symposium on security and privacy (SP)*, 2025, pp. 991–1008. doi: 10.1109/SP61157.2025.00016

[4] Z. Kang *et al.*, "analysis/phase3/exploit_gen/llm_config.py." https://github.com/Follow-my-Flow-GaLA/analysis/blob/main/analysis/phase3/exploit_gen/llm_config.py, 2025.