Intro and Project Goal

The principle of least privilege is a concept used in the design of secure computing systems, one of the existing security models. However, implementing this in real-world scenarios through various programming languages still allows for various vulnerabilities to be found allowing for privilege escalation.

To counter this, one of the solutions is having capability-based security. In a capability model, programs/objects must have a special token that gives them the authority to perform only a specific set of actions (such as reading or writing to certain files). [1] We can also design programming languages that use capabilities from the ground up to design secure programs, such as Wyvern [2].

The goal of the project is to compare programmer productivity, security of the design, extensibility of packages, and other factors to find out whether modules systems/packages having capabilities provide advantages when compared with their absence.

To solve this, we would be designing a set of tasks that can be given to either software architects or experienced software engineers that will involve designing a small product architecture. We will ask one group to solve these using a language with support for modules and object capabilities (e.g. Wyvern) and the other group to use a more traditional language with module or package imports (e.g.Java).[3]

Extensions

1. Analyse some potential security bugs from CVE that the Wyvern module system with capabilities might address.
2. Design the specific module design exercises for the participants to do and see which of them obtain a solution that contains a security flaw and which does not. Compare the two module systems. [1] https://archive.ph/20130112225523/http://www.eros-os.org/essays/capintro.html [2] https://drops.dagstuhl.de/opus/volltexte/2017/7270/pdf/LIPIcs-ECOOP-2017-20.pdf [3] Email Communication

Artefact

The artifact would consist of:

1. Collected data - The source code of programmers solving the designed case studies in various programming languages
2. Experimental results - Comparing traditional and capability-based module Systems through collected data based on factors such as programmer productivity, security of the design, extensibility of packages, etc

Milestones

1. Read about different types of existing Module systems in various programming languages (Wyvern, OCaml, Rust, Java)
2. Develop examples of solving various modules (editors, networking) in languages not having capability by design (Rust for example) compared with languages having capabilities (Wyvern)

3. Week 9: Compete designing case studies for comparing extensions of capable modules of a language versus languages having no extensions of capabilities. The overall results should also be compared with the results of Wyvern modules.
4. Week 11: Design the human study with experienced programmers and obtain the required Human Ethics approval.
5. After approval: Running the study and comparing the results primarily based on usability, productivity, and security