Context

In Man-at-the-end attack setting a malicious user physically owns the software under protection and the system it executes in. Thus he/she can tamper both with the runtime environment and the software. One way to raise the bar against such attackers is to resort to trusted computing. In this model, sensitive code or data can be placed in a tamper-resistant hardware module, e.g. Intel SGX places such code or data in protected memory pages so called enclaves [1]. An enclave code resides in a protected memory region protected by a mandatory access control enforced at the CPU level, ensuring that enclave data and code are protected from other processes, regardless of their permission level.

Utilizing trusted hardware requires modifications in the existing applications, particularly one has to decide which parts of a program shall be executed or stored inside secure hardware. This not only impacts the security but also severely affects the performance of protected programs. Large slowdowns stem from expensive CPU context switches between enclave code and normal code, and vice versa. This requires adequate techniques in program partitioning, given that enclaves by design cannot contain system calls. Lind et al [2] proposed a technique in which a function’s access to sensitive data (previously marked by users) indicates whether it should be placed in enclaves or not.

Goal

In this thesis, we aim at extending the mentioned partitioning technique [2] to sensitive behaviors, i.e. the logic carrying out sensitive operations shall be placed in enclaves. To further optimize the partitioning, we apply our technique at the granularity of instructions, instead of functions. Information flow analysis techniques will be used to group related logic into an enclave unit to minimize context switches. The student will implement the technique as a compiler pass in LLVM. This will give the tool independence from the original programming language of the application. To build a completely hardware-agnostic tool, Asylo framework [3] will be utilized to compile the generated enclave code, which will work not only for Intel SGX but also for other enclave technologies. Finally, the student will evaluate the solution in terms of the provided security guarantees, memory and performance overhead, and Trusted Code Base (TCB) size using a dataset of programs.

Working Plan

- Study existing application partitioning tools
- Evaluate existing frameworks for enclave applications - Asylo
- Implement automatic application division tool
- Evaluate the solution
  - Security by attack defense trees
  - Performance and memory overhead
  - TCB size
- Write the final thesis document

Deliverables

- Docker container able to run a demo of the implementation, including instructions on how to run the demo
- The container should also include the source code of the implementation
- Technical report with comprehensive documentation of the implementation, i.e. design decision, architecture description, API description and usage instructions
- Final thesis report written in conformance with TUM guidelines
References

