Context

Man-At-The-End (MATE) attacks are a threat to software security and intellectual property. Tampering with the software code (also called code manipulation) is an attack that enables adversaries to achieve such goals. Attackers can tamper with the code statically (i.e., binary representation), or dynamically (i.e., in memory). One way to raise the bar against attacks of such is to employ code integrity guards, e.g., self-checksumming [1].

Self-checksumming guards check that program's code hashes to expected values and respond, in a punishing way to users, upon detection of inconsistencies. Clearly, all the expected values have to be precomputed and fed into guards in advance. Bear in mind that checks take place at runtime, and hence hashes are computed over the machine code representation of the software. Therefore, adjusting guards (precomputing expected hashes and subsequently feeding them into software) requires an after compilation patching routine. The post-patching process is heavily architecture (OS, CPU) dependent and extremely error-prone [2].

In order to facilitate the usage of self-checksumming techniques, it is highly desirable to eliminate the necessity for the post-patching process. Applying guards at a higher level of abstraction instead of the machine code can aid us to achieve this goal. One possibility is to employ guards on top of virtualized instructions [3]. The virtualization obfuscation technique [4] can readily provide this abstraction of interest. Therefore, combining self-checksumming with virtualization obfuscation is particularly compelling.

Goal

The goal of this thesis is to implement a prototype of self-checksumming protection on top of the virtualization obfuscation in LLVM. To achieve this goal, the student first adopts/implements a virtualization obfuscation technique in LLVM. Subsequently, the self-checksumming protection, with the ability to have circular checks across function boundaries, is implemented. Another goal is to perform thorough evaluations (performance and security) on a given dataset of programs.

Working Plan

1. Write a state-of-the-art survey on Virtualization and Self-checksumming
2. Implement/adopt virtualization obfuscation technique in LLVM
3. Implement self-checksumming guards on virtualized instructions
4. Carry out security and performance evaluations

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

