Summary

Disassembly is the process of identifying code in binary programs, and translating it into a form fit for human analysis or further processing. It is a crucial step in virtually all forms of binary analysis, including malware analysis and security techniques for binaries. The need for securing legacy and proprietary binaries becomes ever more pressing as attackers develop new exploitation techniques which threaten unhardened binaries. Often, recompilation is not an option, as source code (or even symbols) may not be available; this leaves us with no choice but to use binary-level techniques. Unfortunately, disassembly is an undecidable problem, meaning that any system that operates on non-trivial binaries is bound to run into incomplete or erroneous disassembly.

This thesis explores methods for safely implementing binary-level hardening techniques using imperfect disassembly as a basis. We develop novel defenses against several forms of advanced attacks, including stack-based attacks, control-flow hijacking attacks, and tampering attacks on a hostile host. These defenses implement several strategies which allow us to make a balanced tradeoff between the level of security, overhead, and crash-safety in protected binaries.

Moreover, we perform an in-depth analysis of the disassembly process itself on the x86/x86-64 platform, highlighting the most error-prone cases and uncovering discrepancies between disassembly accuracy in practice, and widespread expectations on disassembly accuracy in the literature. In doing so, we provide a more stable foundation for future disassembly-based research, clarifying where problems are most likely to occur and special measures for ensuring correctness are thus the most necessary. Based on our analysis, we also implement improved methods for function detection; the most error-prone disassembly primitive at the time of writing.