Control Flow Integrity: Security Precision and Performance - A Summary

Boakye Dankwa

January 24, 2018

Control-Flow Integrity (CFI) shows promise to defeat control flow modification attacks such as remote code injection, Return-Oriented Programing (ROP) and code-reuse, that expoit memory corruption vulnerabilities in C/C++ programs. The technique has been researched and improved upon by researchers and has been integrated into products such as LLVM and some Microsoft products. Current CFI evaluations usually use the SPEC2006 benchmark and Average Indirect-target Reduction (AIR) to measure performance and security precision respectively. Control-Flow Integrity: Security Precision and Performance [1] systematize various CFI implementations and their trade-offs, and propose a novel way of evaluating these implementations against security precision and performance.

The authors first described CFI mechanisms as consisting of an analysis and a runtime component. The analysis component constructs a Control-Flow Graph(CFG), which approximates the set of control-flow transfers in the program, from the source code or binary. The CFG is then used by the runtime to restrict program execution flow by validating indirect control-flow transfer against the CFG edges. They classified control-flow transfer into nine categories from CF.1 (backward control flow such as return instruction) to CF.9 (control flow that supports features such as dynamic linking) based on the kind of control flow (*direct, indirect, jump, return* or *call*) and supported language constructs (*such as function pointers and vtables*). They further explained that CFG is subject to over-approximation, making the precision of the CFG analysis crucial in determining the security precision of a given CFI mechanism. They classified prior work on precision of the static analysis into increasing order of static analysis precision (SAP), from SAP.F.0 (No forward branch validation) to SAP.F.6(supports dynamic analysis) for forward control-flow analysis precision. For backward control-flow precision, the classification are SAP.B.0 (No backward branch validation), SAP.B.1(Labeling equivalent classes) and SAP.B.2 (Shadow stack). They hope to achieve a more precise and consistent taxonomy this way as opposed to the fine-gained/coarse-grained classification used in prior work within the CFI research community.

Using the proposed classifications on control-flow transfers and static analysis precision, they qualitatively analyzed the theoretical security guarantees of different existing CFI mechanisms in publications. They also gave qualitative evaluation of the security precision of several CFI implementations. The CFI mechanisms were evaluated qualitatively on four dimensions namely: supported control-flow transfers (CF), reported implementation overhead (RP), static analysis precision on forward control-flows (SAP.F) and static analysis precision of backward control-flows (SAP.B). Each of the mechanisms were evaluated on four axes on a spider plot, each exes corresponding to one of the dimensions. They stated that the area under the spider plot gives a rough estimate of the precision of a given mechanism.

References

[1] Nathan Burow, Scott A. Carr, Stefan Brunthaler, Mathias Payer, Joseph Nash, Per Larsen, and Michael Franz. Control-flow integrity: Precision, security, and performance. *CoRR*, abs/1602.04056, 2016.