From Fine Grained Code Diversity to JIT-ROP to Execute-Only Memory: The Cat and Mouse Game Between Attackers and Defenders Continues

Michael Franz
University of California, Irvine

Abstract

Today's software monoculture creates asymmetric threats. An attacker needs to find only one way in, while defenders need to guard a lot of ground. Adversaries can fully debug and perfect their attacks on their own computers, exactly replicating the environment that they will later be targeting.

One possible defense is software diversity, which raises the bar to attackers. A diversification engine automatically generates a large number of different versions of the same program, potentially one unique version for every computer. These all behave in exactly the same way from the perspective of the end-user, but they implement their functionality in subtly different ways. As a result, a specific attack will succeed on only a small fraction of targets and a large number of different attack vectors would be needed to take over a significant percentage of them. Because an attacker has no way of knowing a priori which specific attack will succeed on which specific target, this method also very significantly increases the cost of attacks directed at specific targets.

Unfortunately, attackers have now started assembling their attacks on the target itself, circumventing diversity. In order to prevent this, we need to make all executable code on the target platform unreadable by the attacker. We present a solution that keeps randomized executable code completely hidden from the attacker, preventing even the latest class of dynamically assembled code reuse attacks ("JIT-ROP").

We will also report on a set of new software diversity techniques that can additionally also defend against side-channel attacks by dynamically and systematically randomizing the control flow of programs. Previous software diversity techniques transform each program trace identically. Our new technique instead transforms programs to make each program trace unique. This approach offers probabilistic protection against both online and off-line side-channel attacks, including timing and cache-based attacks.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
Copyright is held by the owner/author(s).
MTD'15, October 12, 2015, Denver, Colorado, USA.
ACM 978-1-4503-3823-3/15/10.
DOI: http://dx.doi.org/10.1145/2808475.2808488

In particular, we create a large number of unique program execution paths by automatically generating diversified replicas for parts of an input program. At runtime we then randomly and frequently switch between these replicas. As a consequence, no two executions of the same program are ever alike, even when the same inputs are used. Our method requires no manual effort or hardware changes, has a reasonable performance impact, and reduces side-channel information leakage significantly when applied to known attacks on AES.

Acknowledgement:
The author's research has been partially supported by the National Science Foundation (NSF) under grants No. CCF-1117162 and CNS-1513837, by the Defense Advanced Research Projects Agency (DARPA) under contracts D11PC20024, N660001-1-2-4014, N66001-13-C-4057, FA8750-15-C-0124, and FA8750-15-C-0085, as well as by gifts from Adobe, Google, Mozilla, Oracle, and Qualcomm.

Short Bio

Michael Franz is the director of the Secure Systems and Software Laboratory at the University of California, Irvine (UCI). He is a Full Professor of Computer Science in UCI's Donald Bren School of Information and Computer Sciences and a Full Professor of Electrical Engineering and Computer Science (by courtesy) in UCI's Henry Samueli School of Engineering. Prof. Franz was an early pioneer in the areas of mobile code and dynamic compilation. He created an early just-in-time compilation system, contributed to the theory and practice of continuous compilation and optimization, and co-invented the trace compilation technology that eventually became the JavaScript engine in Mozilla's Firefox browser. Franz received a Dr. sc. techn. degree in Computer Science and a Dipl. Informatik-Ing. ETH degree, both from the Swiss Federal Institute of Technology, ETH Zurich.