Net-Centric & Cloud Software & Systems (NCSS)

University of North Texas, Krishna Kavi, 940.369.7216, kavi@cse.unt.edu

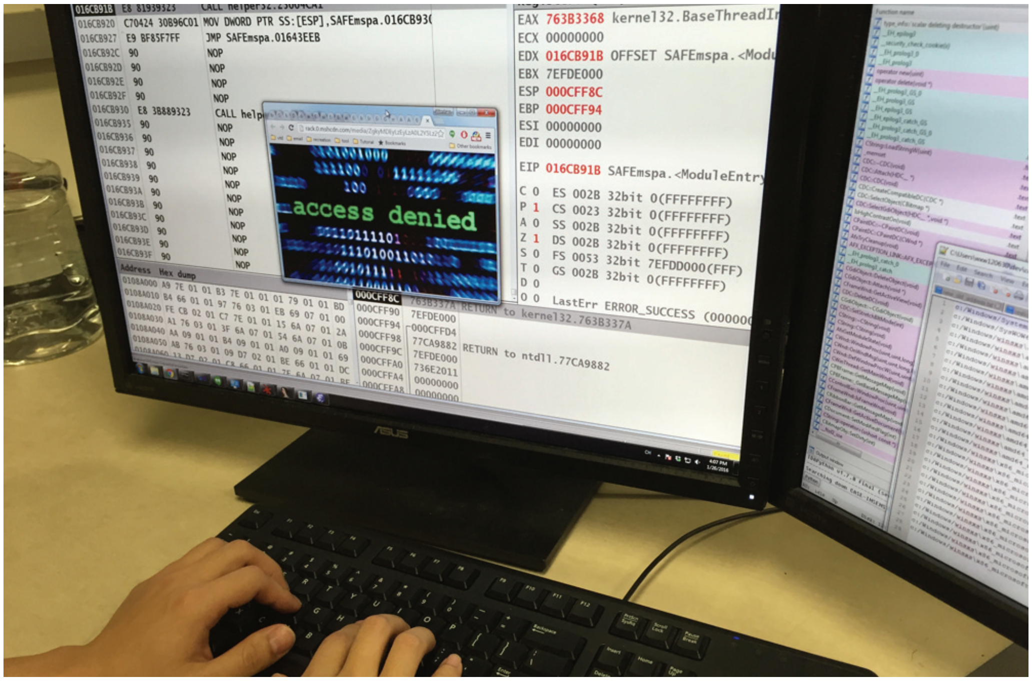
University of Texas at Dallas, Farokh B. Bastani, 972.883.2299, bastani@utdallas.edu

Missouri University of Science and Technology, Sanjay Madria, 573-341-4856, madrias@mst.edu

Arizona State University, Andreas Spanias, 480.965.5311, spanias@asu.edu

Protecting Against State-of-the-Art Cyber Attacks through Opaque Control-Flow Integrity

Opaque Control-Flow Integrity (O-CFI) is the first software security defense that merges binary software randomization with CFI to defeat one of the latest cybersecurity threats: implementation-aware code-reuse (IACR) attacks. IACR attacks hijack software by first exfiltrating in-memory code details of a victim program, and then exploiting those details to corrupt the victim program's control-flow paths. Such attacks defeat most CFI defenses, because CFI must typically enforce an approximation of the control-flow policy to maintain performance, and the attack's exfiltration step leaks the secret approximation to the attacker. Likewise, IACR defeats traditional randomization-based defenses, because the exfiltration leaks the randomized code layout.



Binary software security research being conducted at the University of Texas at Dallas.

Opaque Control-Flow Integrity defeats IACR attacks by randomizing the policy approximation enforced by CFI in such a way that the secret approximation is confined to a protected data region of the software, not its code. This means that even if an IACR attack leaks the complete binary code, stack, and heap memory of the victim program to the attacker, this is not usually enough information for the attacker to reliably determine how the software's control-flows can be safely corrupted without raising an alarm. Attackers are forced to guess effective attack strategies. Such guesses have been shown to fail with astronomically high probabilities. Moreover, randomization ensures that even if one attack succeeds, the same attack fails against other instances of the program (or even the same program after it is restarted) because are all randomized differently.

Protections via O-CFI can be applied to secure software either at compile-time or to the already-compiled binary code. This makes this breakthrough approach extremely flexible; for example, O-CFI can be applied to secure binary software after it has already been shipped and deployed. This can be done without any aid from the software's original developers. In addition, its code transformations are fully automated, allowing it to be deployed almost instantly in response to emerging threats.

Opaque Control-Flow Integrity introduces the only known defense against an important class of zero-day cyberattacks. Moreover, it is extremely practical: It can be effectively implemented as a binary code transformation of existing, commodity software products, avoiding any need to re-develop the software or use any special tools in its creation. Experiments show that it introduces only 4.7% mean performance overhead on current processors, with even faster performance expected in the near future, since O-CFI's integrity checks are implemented using instructions that are expected to be hardware-accelerated on forthcoming x86/x64 processors.

O-CFI can be applied to protect most binary software products that are potential victims of control-flow hijacking attacks. These include web ecommerce systems, military systems, online database systems (e.g., healthcare information databases), industrial control systems, and other computing infrastructures requiring high assurance.

Economic impact: Since O-CFI can be applied to secure binary software that has already been designed, compiled, and deployed, it potentially mitigates the high costs associated with alternative security approaches requiring software re-development, such as penetration testing (e.g., fuzzing), source code review (which is notoriously ineffective at finding IACR vulnerabilities), or formal methods validation. It can also be applied much more quickly than these alternatives, potentially rescuing software from attacks that would otherwise have succeeded before more time-consuming approaches could discover and eliminate vulnerabilities.

For more information, contact Kevin Hamlen, hamlen@utdallas.edu, Bio http://www.utdallas.edu/~hamlen/.