UCF DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCEINCE

Spring 2015 Seminar Series

Presented by the CS Division

BINARY PROGRAM INTEGRITY MODELS FOR DEFEATING CODE-REUSE ATTACKS

THURSDAY APRIL 16, 2015

10:00 AM – HEC 450

During a proverbial 'hack', an attacker often exploits a vulnerability in a program, hijacks control-flow, and executes malicous code. Data Execution Prevention (DEP), a hardware-enforced security feature, prevents an attacker from directly executing the injected malicious code. Therefore, attackers have resorted to code-reuse attacks, wherein carefully chosen fragments of code within existing code sections of a program are sequentially executed to accomplish malicious logic. Code-reuse attacks are ubiquitous and account for majority of the attacks in the wild. On one hand, due to the wide use of closed-source software, binary-level solutions are essential. On the other hand, without access to source-code and debug-information, defending raw binaries is hard.

A majority of defenses against code-reuse attacks enforce "control-flow integrity", a program property that requires the runtime execution of a program to adhere to a statically determined control-flow graph (CFG) — a graph that captures the intended flow of control within a program. State-of-the-art binary-level defenses lack in two areas. (1) Precision: Without source-code, binary-level defenses recover a conservative and approximate CFG that accommodates several illegitimate edges along with all the legitimate edges. By launching practical attacks that leverage the illegitimate edges within the approximate CFG, attackers have highlighted the need for more precise CFG. (2) Incremental deploy-ability: A complete CFG includes inter-module control flows, which are unknown until the load time. Therefore, such defenses can either protect all the modules used by a program, or none of them. Partial protection leads to unaffordable false alarms.

In this talk, I will first provide an overview of state-of-the-art in code-reuse attacks and binary-level defenses. Then, I will present two of my works that address precision and deploy-ability of defenses: The first work improves precision of CFI in C++ binaries, and the second work introduces Stack-Pointer Integrity (SPI), a program integrity model that defends against code-reuse attacks by enforcing integrity of stack pointer.

ARAVIND PRAKASH

Syracuse University

Aravind Prakash is a PhD candidate in Dept of Electrical Engineering and Computer Science at Syracuse University. His speciality is system security with emphasis on program analysis. He has authored multiple papers in top-tier security conferences. He holds a Master of Science degree from University of Miami, FL, and a Bachelor of Engineering from VTU, India.

Hosted by: Dr. Sheau-Dong Lang