BINARY HARDENING CHALLENGES: MODULARITY & IMMUTABILITY

Dr. Kevin Hamlen
Associate Professor
Computer Science Department
Cyber Security Research and Education Institute
The University of Texas at Dallas

Supported in part by:
ONR Awards N00014-14-1-0030 & N00014-17-1-2995,
NSF CAREER Award #1054629,
AFOSR YIP (Career) Award FA9550-08-1-0044,
NSF Award 1513704
and an NSF I/UCRC Award from Raytheon Company

Any opinions, findings, conclusions, or recommendations expressed in this presentation are those of the author(s) and do not necessarily reflect the views of the AFOSR, ONR, NSA, or Raytheon.
Modularity Challenge: Diverse, Incompatible Protections

**XFI [Erlingsson et al., OSDI’06]**
- Instrumentation Strategy:
  - Insert labels at jump targets
  - Insert label-checking guards at control-flow transfer instructions
- Policy: a CFG

**Microsoft Control-Flow Guard**
- Instrumentation Strategy:
  - Enumerate jump targets in a table
  - Insert table-checking guards at indirect calls
- Policy: jump target whitelist
Modularity Challenge: Diverse, Incompatible Protections

XFI [Erlingsson et al., OSDI'06]
- Instrumentation Strategy:
  - Insert labels at jump targets
  - Insert label-checking guards at control-flow transfer instructions
- Policy: a CFG

Microsoft Control-Flow Guard
- Instrumentation Strategy:
  - Enumerate jump targets in a table
  - Insert table-checking guards at indirect calls
- Policy: jump target whitelist

CO nfused D Ep uty- assisted Object Oriented Programming (CODE-COOP) Attack
Immutability Challenge: System Modules and Runtime APIs

- Trusted but Immutable modules
  - Integrated into closed-source OS/VM
  - Digitally signed
  - Hard to disassemble (e.g., obfuscated)
  - Dynamically downloaded (e.g., clouds)
  - Anti-tampering / anti-piracy technology
  - Nevertheless “secure” (according to some policy)

- Modularity question: Can we enforce strong CFI policies if we can’t modify some “secure” modules?

  Our Answer: Yes! … sometimes … mostly …
Object Flow Integrity

[Wang, Xu, Hamlen, CCS’17]
OFI Outcomes

- First source-free CFI that can protect large-scale, commercial, event-driven Windows apps
  - finally support for Component Object Model (COM) apps!
- Low overhead (~1%)
- No modification of Windows system libraries
- Embeds app-enforced CFI protections into shared data (objects and code pointers)

Research Challenges/Limitations

- How to know what security invariants trusted modules obey/enforce?
  - What memory safety policy? (stack? heap?)
  - What control-flow policy? (threading? indirect calls? returns?)
  - What API call policy? (needed API accesses? arguments?)

- Missing Research
  - Source-free analyses to answer such questions
  - Source-aware technologies for proving answers to such questions
  - More modular hardening algorithms that anticipate diversity of defenses
Beyond Linux

Desktop OS Market Share

- Windows 91%
- macOS 6%
- Linux 3%
Thank you!