Side-Channel Finder for AVR

Florian Dewald, Heiko Mantel, Alexandra Weber (MAIS, TU Darmstadt)
CHES 2018, Poster Session, Amsterdam, The Netherlands, September 9–12, 2018

Motivation

Side Channels
- unintended communication channels, e.g., running time, power consumption, ...
- can reveal secrets, e.g., cryptos keys
- timing side channels are a dangerous subclass of side channels because they can be exploited remotely [Brumley/Boneh, Computer Networks 2005]

Attacks on Devices in the Internet of Things
- Internet of Things (IoT) devices are attractive targets for attacks
- recently, smart light bulbs were compromised [Ronen/O’Flynn/Shamir/Weingarten, SAP 2017]
- the vulnerable light bulbs contain an AVR processor, a popular processor in IoT devices

Our Focus
- systematic detection of timing side channels in AVR programs

Tool Support

Side-Channel Finder (SCF)
- takes a well-formed AVR program
  - syntactically correct, single return instruction, in supported sublanguage
  - takes a configuration
  - specification of secret and public inputs and outputs
  - checks for absence of timing side channels using the timing-sensitive security type system
  - implemented in roughly 1250 lines of Python code

Reports to the User of Side-Channel Finder
- success report if program is typeable
  - program satisfies TSNI, i.e., is free from timing side channels
  - failure report if program is not typeable
  - shows the reason for failure, e.g., secret-dependent loop
  - points to location of potential vulnerability in the code

Approach

Timing-Sensitive Noninterference (TSNI)
- two program executions that start in attacker-indistinguishable states
take the same time and the final states are attacker-indistinguishable

Timing-Sensitive Security Type System
- assign security types to registers, memory, and stack (secret or public)
- type system only allows to type a program if:
  - the program is free of timing side channels, and
  - there is no direct or indirect information flow
- soundness: all typeable programs satisfy timing-sensitive noninterference
- precision: secret-dependent branches are allowed if the branches take the same time

Case Studies

Crypto Implementations from µNaCl
- µNaCl is a crypto library for AVR microcontrollers [Hutter/Schwabe, Africacrypt 2013]
- Side-Channel Finder was applied to multiple crypto implementations from µNaCl
  - Salsa20 (stream cipher)
  - XSalsa20 (variant of Salsa20 with longer nonces)
  - Poly1305 (message authentication code)

Verifications Results for µNaCl
- Side-Channel Finder successfully verified that the crypto implementations satisfy TSNI, i.e., the implementations are secure against timing side channels

Further Side-Channel Research at MAIS

Our Vision for Side-Channel Security
- traditionally: attack-driven detection and often reactive mitigation of side channels
- our vision: systematize the detection and mitigation of side channels
- complement the attack-driven approach with program analysis

Quantitative Program Analysis
- approach based on reachability and information theory
  - e.g., systematic study of cache side channels across AES implementations [Mantel/Weber/Kap, ESORICS 2017]
  - detection and mitigation of 4 cache side channels in the implementation of the lattice-based signature scheme ring-TESLA [Biebedi/Buchmann/Krämer/Mantel/Schickel/Weber, FPS 2017]

Systematic Distinguishing Experiments
- approach based on statistical methods and information theory
  - e.g., clarification of the security implications of green IT (in the domain of software-based energy side channels) at the example of BouncyCastle RSA and Intel RAFL [Mantel/Schickel/Weber, ESORICS 2018]
  - e.g., clarification of security-performance tradeoff for program transformations against timing side channels [Mantel/Starrstein, ESORICS 2015]