Control-Flow Integrity for Smartphones

Lucas Davi[1], Alexandra Dmitrienko[2], Manuel Egele[3], Thomas Fischer[4], Thorsten Holz[4], Ralf Hund[4], Stefan Nürnberger[1], Ahmad-Reza Sadeghi[1,2]

[1] CASED, Technische Universität Darmstadt, Germany
[2] Fraunhofer SIT, Darmstadt, Germany
[3] University of California, Santa Barbara, United States
[4] Horst Görtz Institute for IT-Security, Ruhr-Universität Bochum, Germany

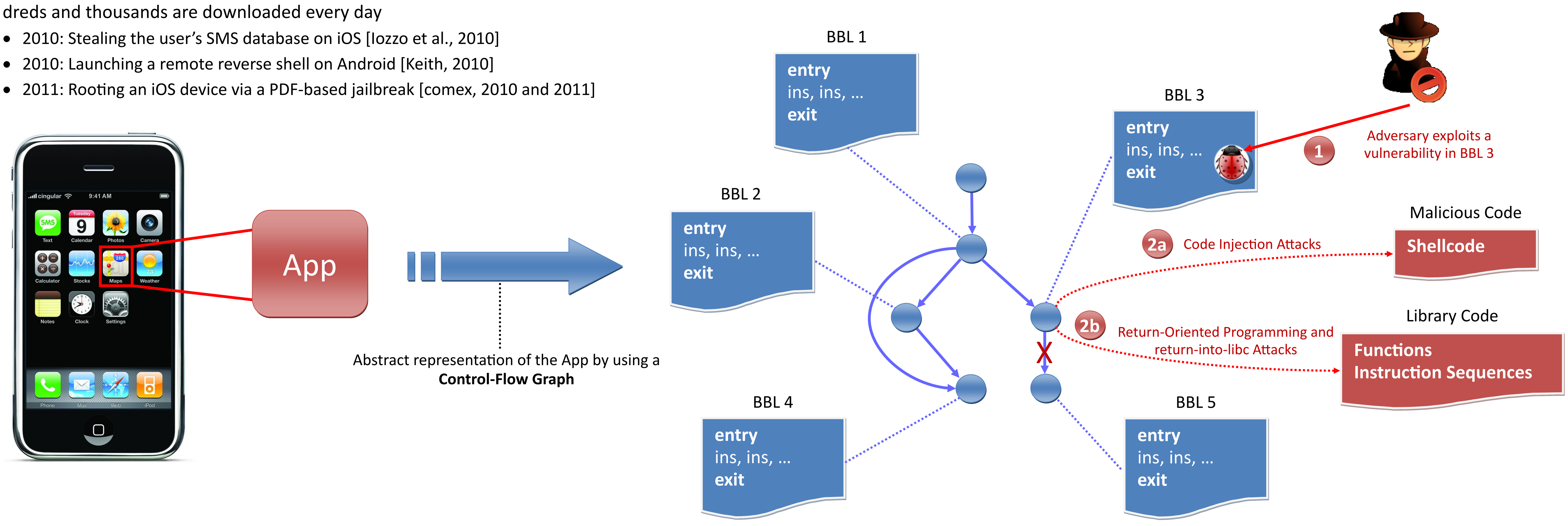
Control-Flow Attacks: A Major Threat to Software Applications (on Desktop PCs and Smartphones)

The Problem of Control-Flow Attacks (Runtime Attacks)

- Control-flow attacks are possible because applications still suffer from a variety of (memory-related) vulnerabilities allowing an adversary to compromise the application flow via diverse techniques, e.g., stack overflows, heap overflows, integer overflows, pointer subterfuges or format strings.
- **Recently these attacks have been applied to smartphones applications as well** of which hundreds and thousands are downloaded every day
  - 2010: Stealing the user's SMS database on iOS [Iozzo et al., 2010]
  - 2010: Launching a remote reverse shell on Android [Keith, 2010]
  - 2011: Rooting an iOS device via a PDF-based jailbreak [comex, 2010 and 2011]

Basic Principle of Control-Flow Attacks

The adversary exploits a vulnerability of a benign application at runtime (*step 1*) and afterwards redirects the control-flow either to injected (malicious) code (*step 2a*) or to existing code pieces residing in shared libraries (*step 2b*)



BBL: A basic block is a sequence of assembler instructions (**ins**) with a single **entry** and **exit** instruction
entry: Any instruction that is target of a branch (e.g., the first instruction of a function)
exit: Any branch instruction (e.g., indirect or direct jump/call, function return)

A General Solution: Control-Flow Integrity

Basic Principle of Control-Flow Integrity (CFI)

- CFI is a general countermeasure against control-flow attacks
- Originally proposed and implemented for Intel x86 by Microsoft [Abadi et al., CCS 2005]
- This technique asserts the basic safety property that the control-flow of an application follows only the legitimate paths determined in advance. If an adversary hijacks the control-flow, CFI enforcement can detect this divagation and prevent the attack.
- **However, there exists no CFI solution for smartphone platforms!**

Problems of the Existing Intel x86 CFI Approach

- Requires a sophisticated binary instrumentation framework (Vulcan) that is not publicly available and only supports x86 and Windows operating systems
- Moreover, the binary rewriting approach requires debugging information that are typically not included in third-party applications

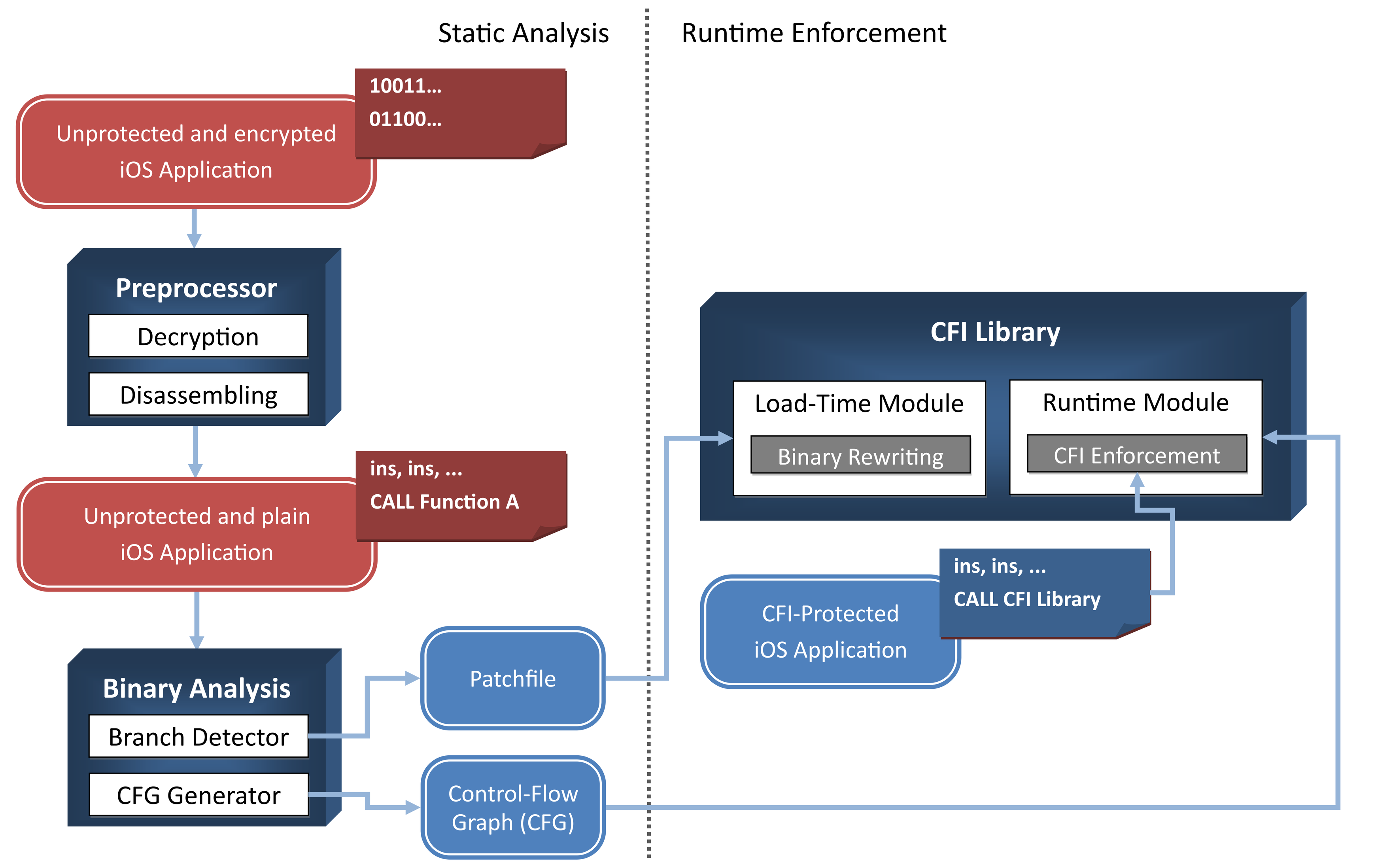
Challenges for a CFI Solution on Smartphones

- ARM and Intel x86 differ substantially
  - No dedicated return instruction
  - The program counter is directly accessible
  - Side-Effects: Control-Flow changes may involve the loading of several other registers
  - ARM supports two instructions sets (32 Bit ARM and 16 Bit THUMB) and the processor can switch among these at runtime
- More problems on iPhone:
  - Applications are encrypted and signed
  - iOS is closed-source and cannot be changed

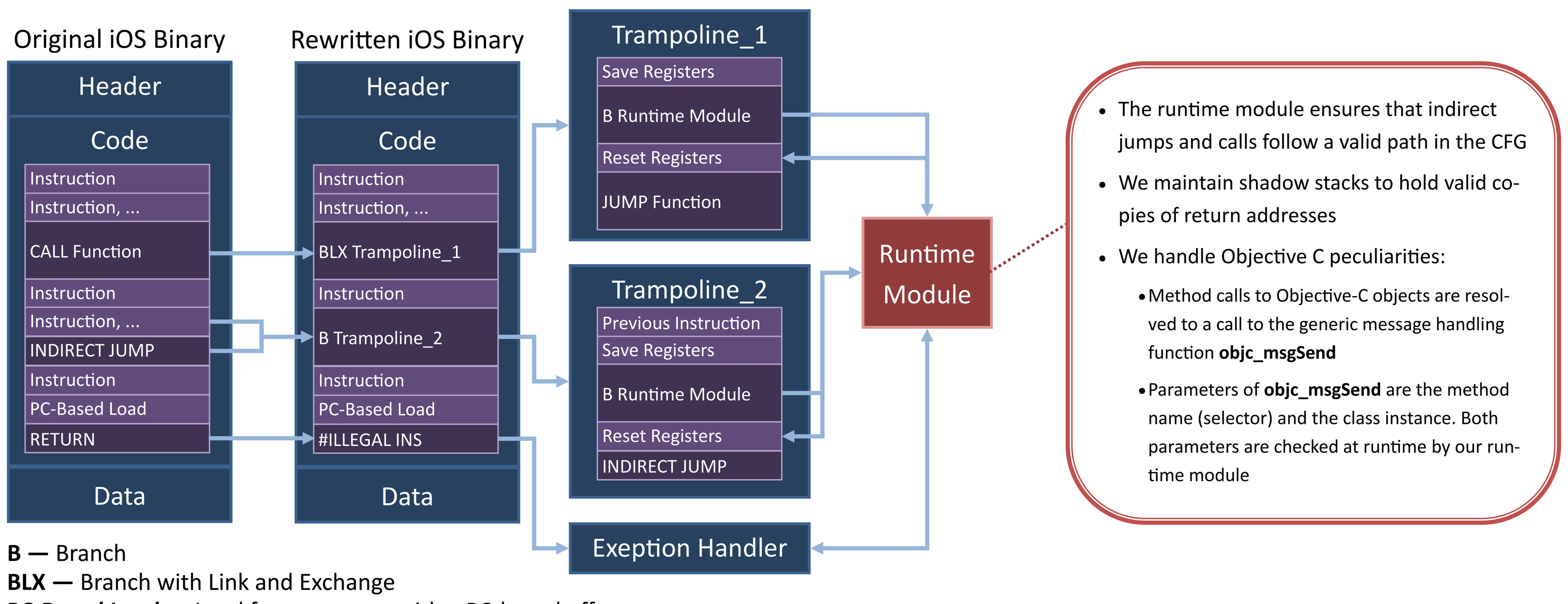
Control-Flow Integrity Framework for Smartphones

We present the design and implementation of the first CFI enforcement framework for iOS. Our framework can be divided into two phases:

1. **Static Analysis:** We developed new tools and extended the PiOS framework [Egele et al., NDSS 2011] to recognize all branch instructions and derive the CFG of an iOS application
2. **Runtime Enforcement:** We developed a new CFI library that consist of a load-time module which rewrites the application on-the-fly, and a runtime module which performs the control-flow checks



Implementation Details of Our CFI Library



B — Branch
BLX — Branch with Link and Exchange
PC-Based Load — Load from memory with a PC-based offset

Conclusion

First CFI Enforcement Framework for Smartphones

- Requires no access to source code
- Performs binary rewriting on-the-fly and is therefore compatible to application signing/encryption and memory randomization (e.g., ASLR)
- We addressed unique challenges of smartphone platforms and operating systems
- Our CFI enforcement is efficient and induces acceptable performance overhead

Contact:
Lucas.Davi@trust.cased.de
Ralf.Hund@rub.de

18th ACM Conference on Computer and Communications Security (CCS 2011)
OCT 17-21 2011. Chicago, IL, USA
POSTER: Control-Flow Integrity for Smartphones