USENIX Security Symposium 2014, San Diego, CA, USA

Stitching the Gadgets

On the Ineffectiveness of Coarse-Grained Control-Flow Integrity Protection

Lucas Davi
Intel Collaborative Research Institute for Secure Computing @ TU Darmstadt, Germany

Joint Work with
Daniel Lehmann, Ahmad-Reza Sadeghi (TU Darmstadt)
Fabian Monrose (UNC Chapel Hill)

Open Question:

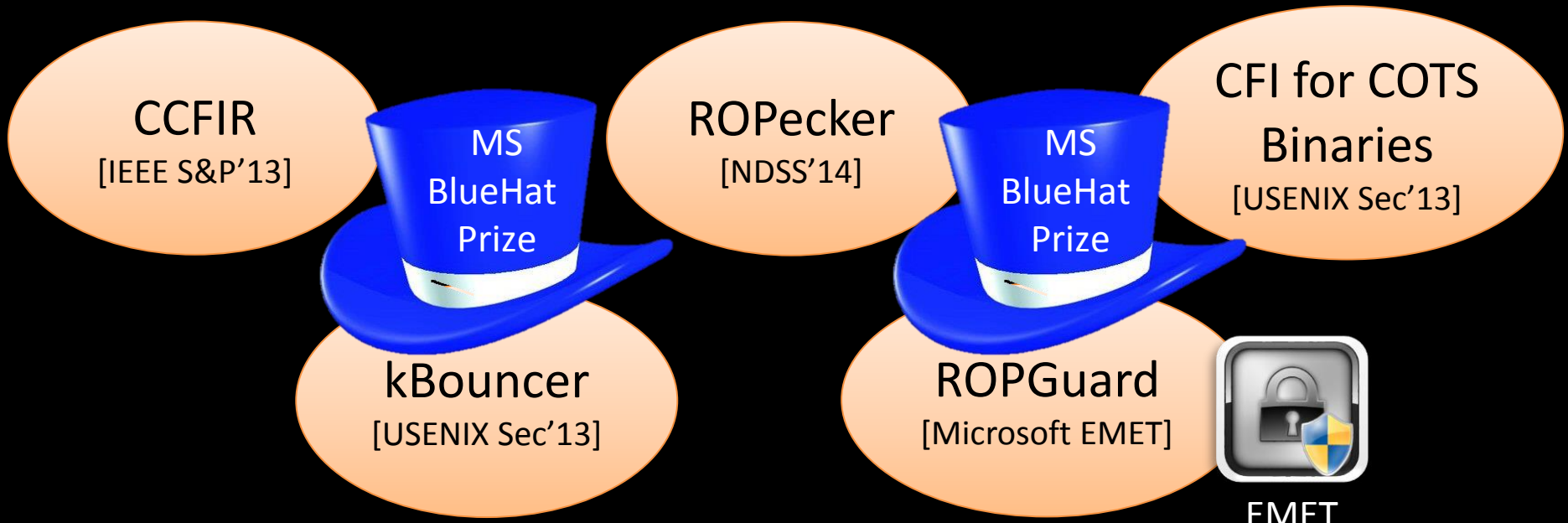
Practical and secure mitigation of code reuse attacks

Turing-completeness of return-oriented programming

Hot Research Topic:

"Practical" (coarse-grained) Control Flow Integrity (CFI)

Recently, many solutions proposed



http://technet.microsoft.com/en-us/security/jj653751

This Talk:

Negative result – all current (published) coarse-grained CFI solutions can be bypassed

Our Contributions

Systematic Security Analysis

Pointing out the conceptual weaknesses of coarse-grained CFI solutions including Microsoft's EMET

Systematic Security Analysis

Generic Attack

Real World Exploits

Our Contributions

Our Attack is Generic

Only CFI policies considered, not specific CFI solutions, bypassing even the Über-CFI that combines all CFI policies

The only work showing Turing-completeness

Systematic Security Analysis

Generic Attack

Real World Exploits

Our Contributions

Concrete Attack Instantiations under Strong Adversary Model

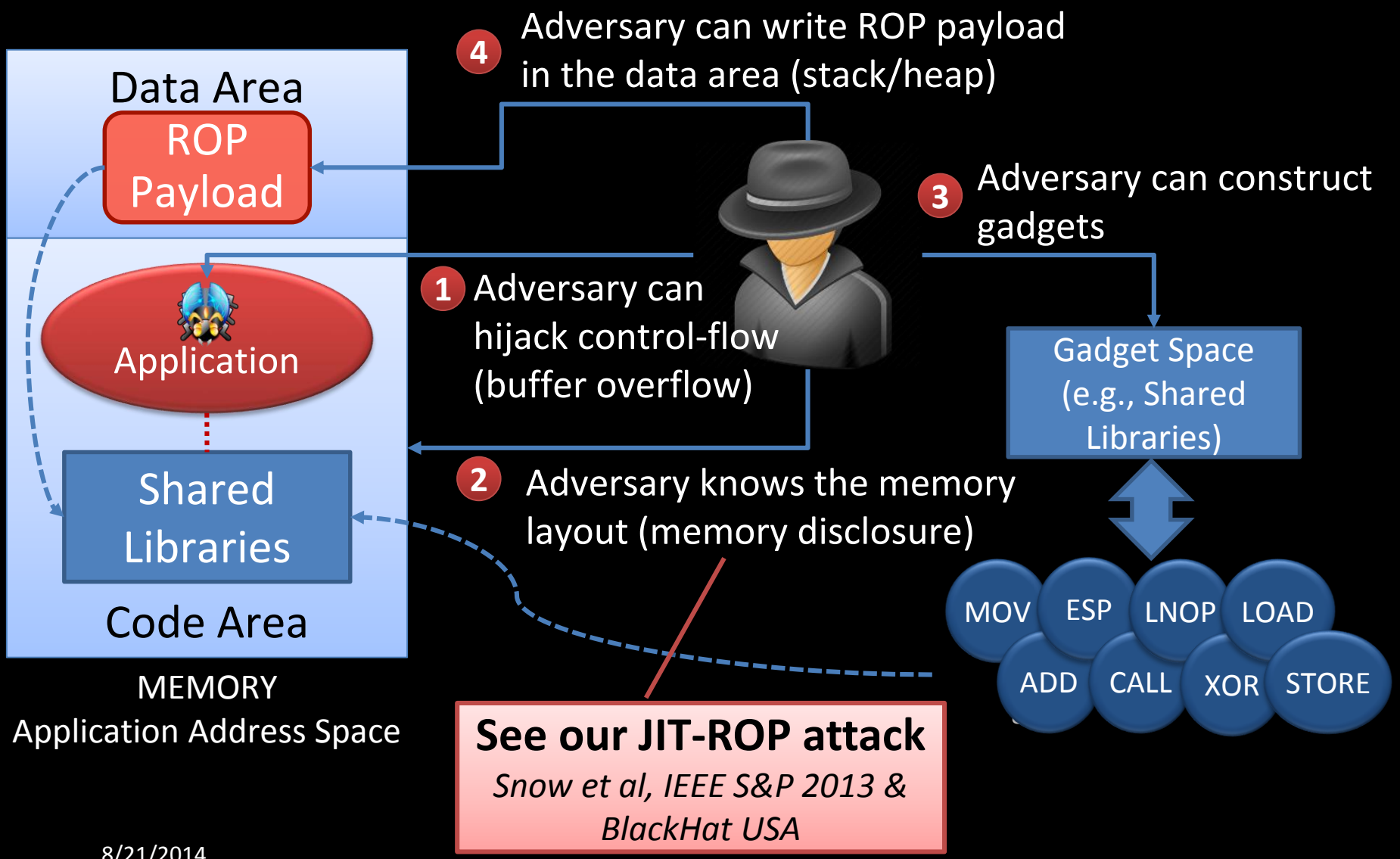
Very small code base: 840kb

Systematic Security Analysis

Generic Attack

Real World Exploits

ROP Adversary Model/Assumption

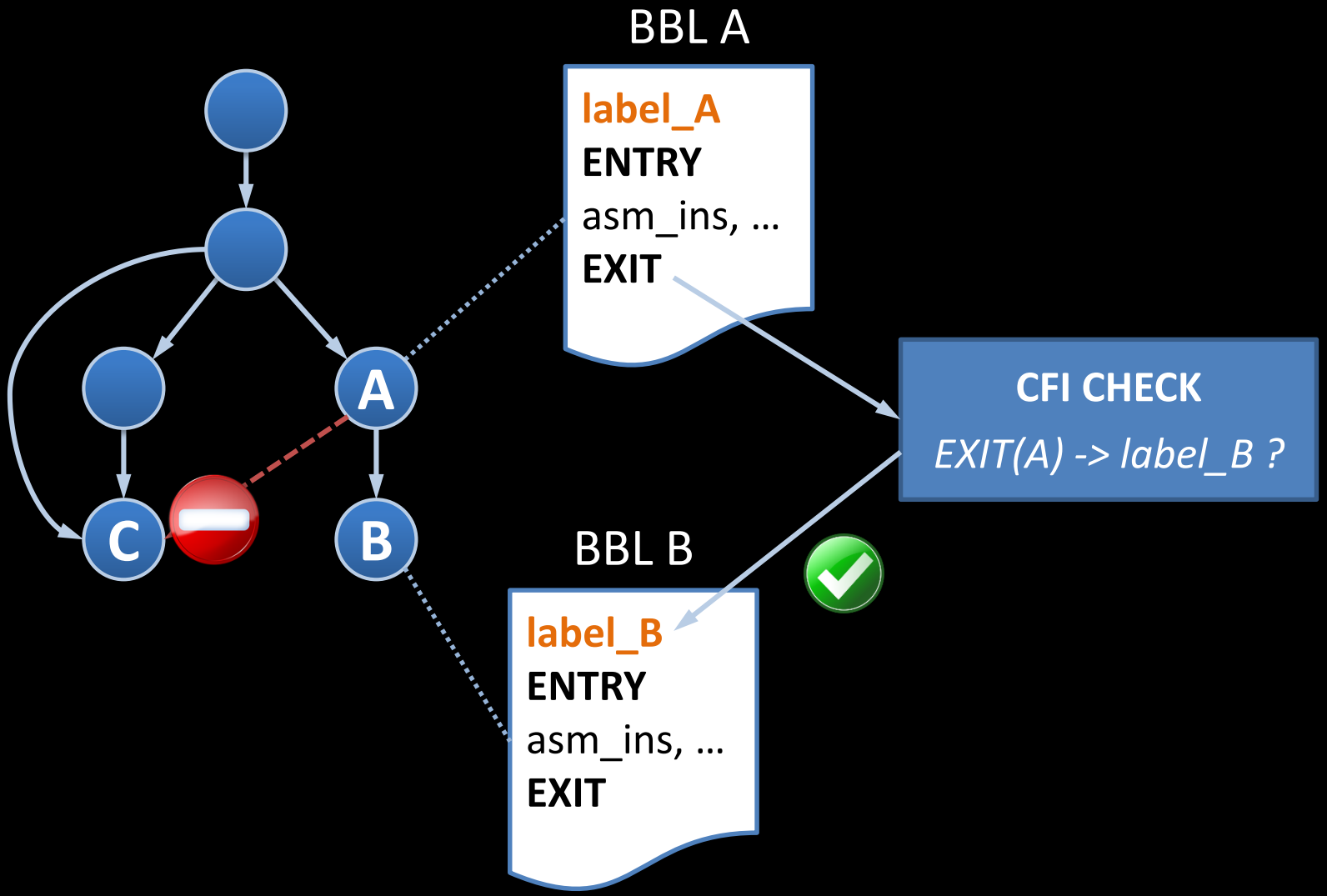


Generic ROP Defense: Control-Flow Integrity (CFI)

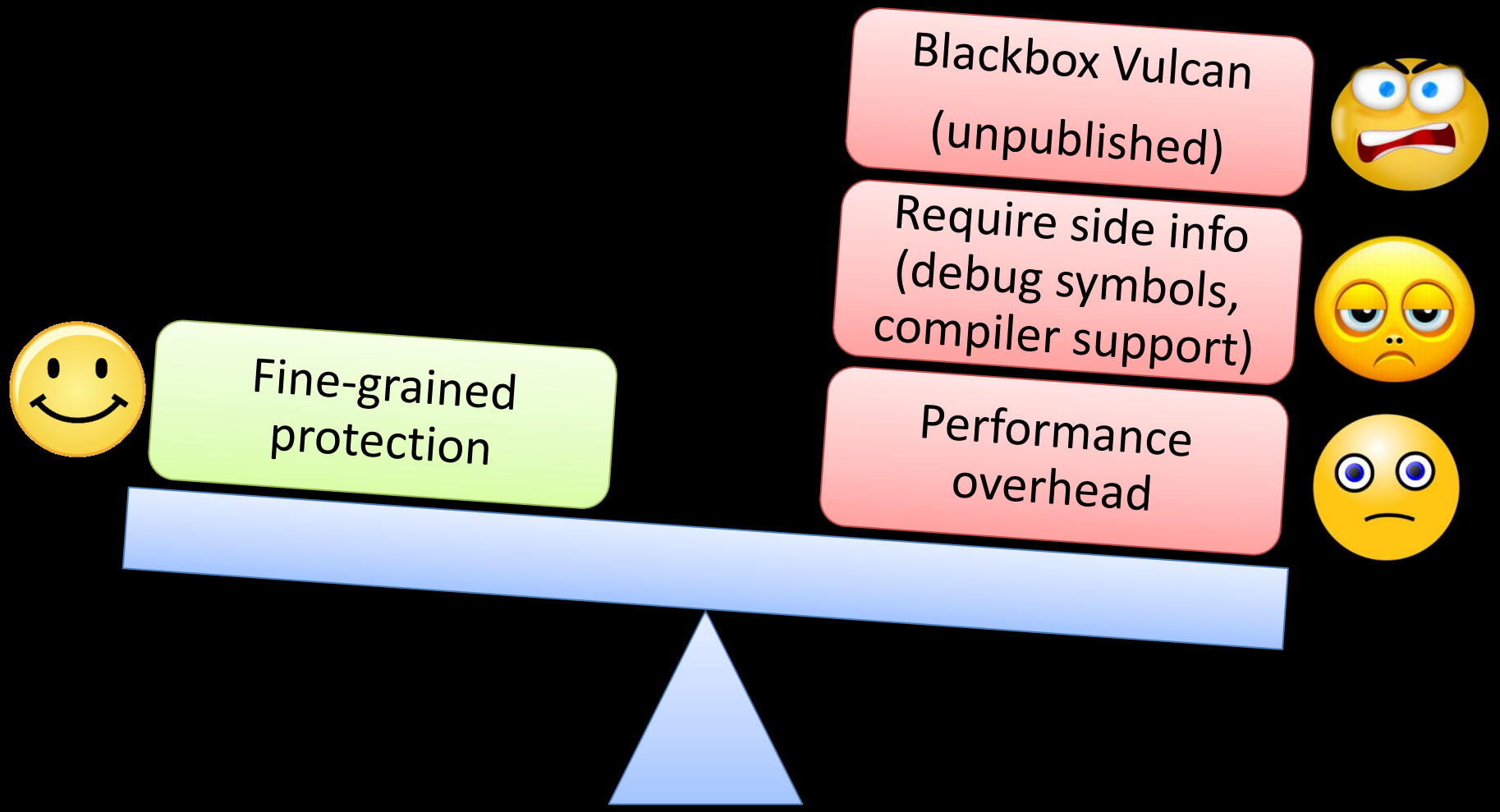
Restricting indirect control-flow targets to a pre-defined control-flow graph

Original CFI Label Checking

[Abadi et al., CCS 2005 & TISSEC 2009]



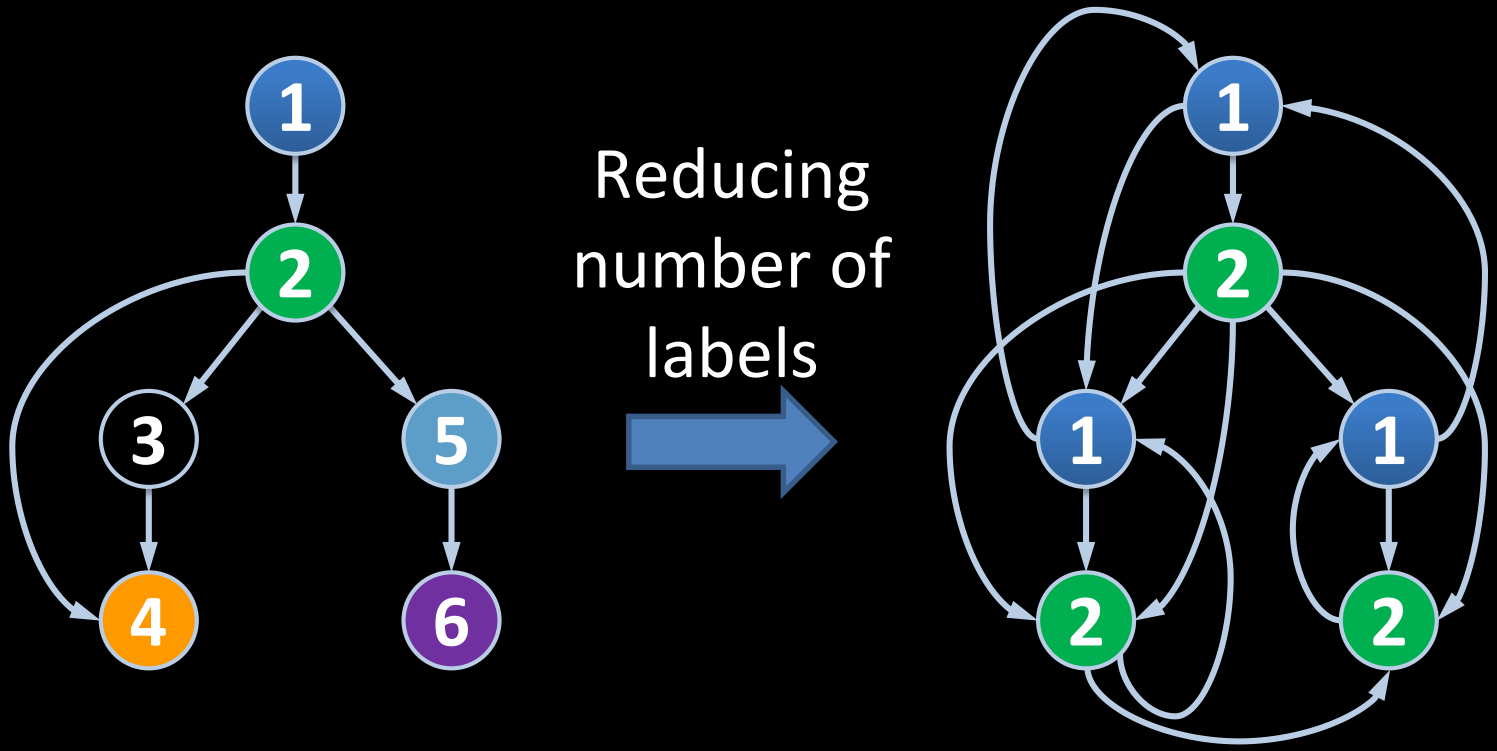
Original CFI: Benefits and Limitations



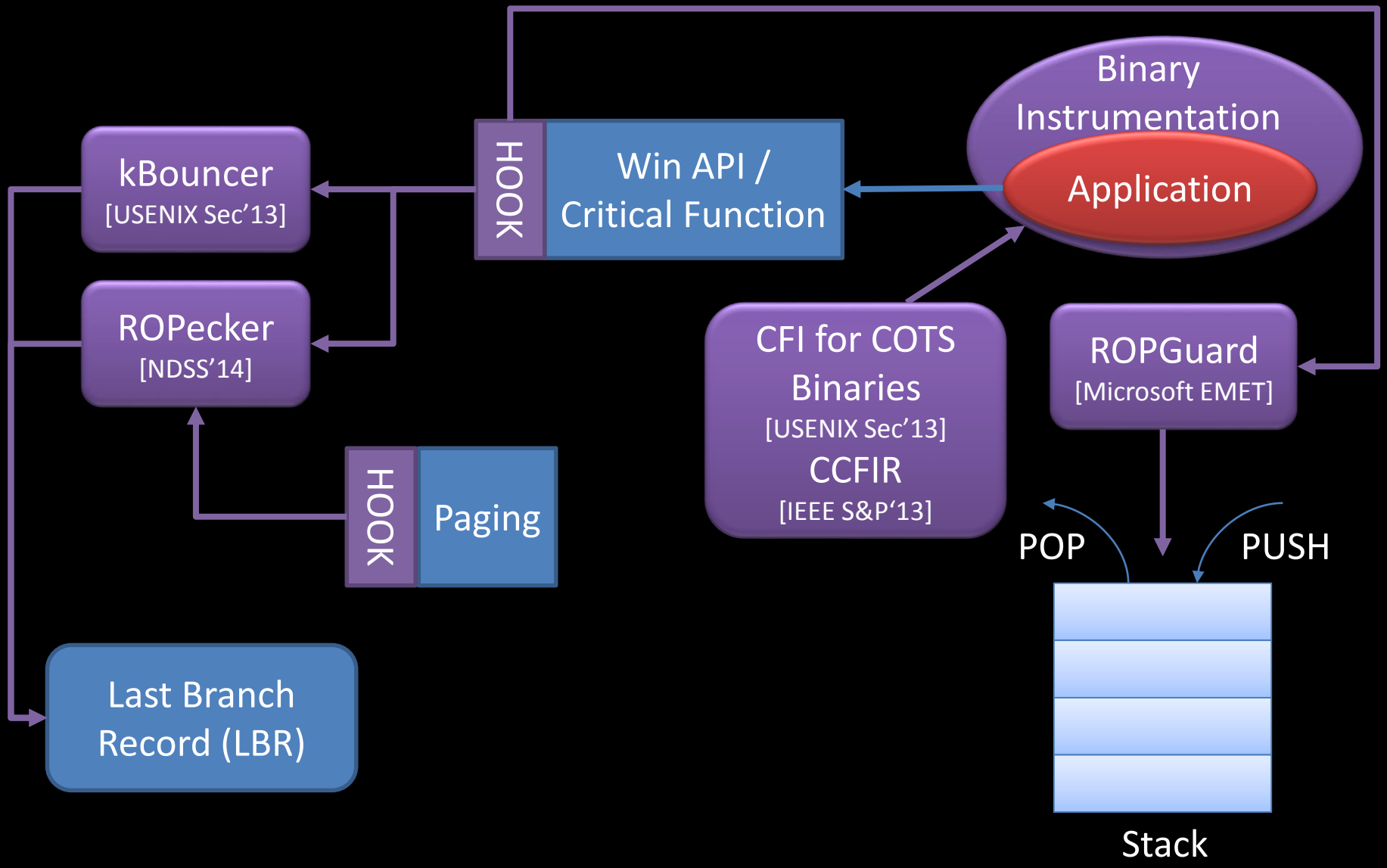
Coarse-Grained CFI:

Aims at practical CFI for real-world deployment

General Idea

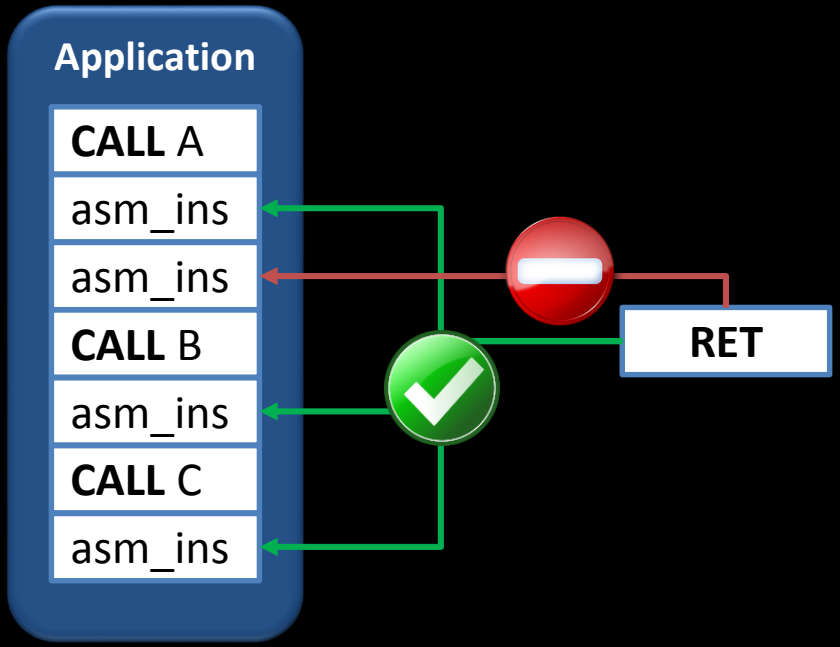


Coarse-Grained CFI Proposals



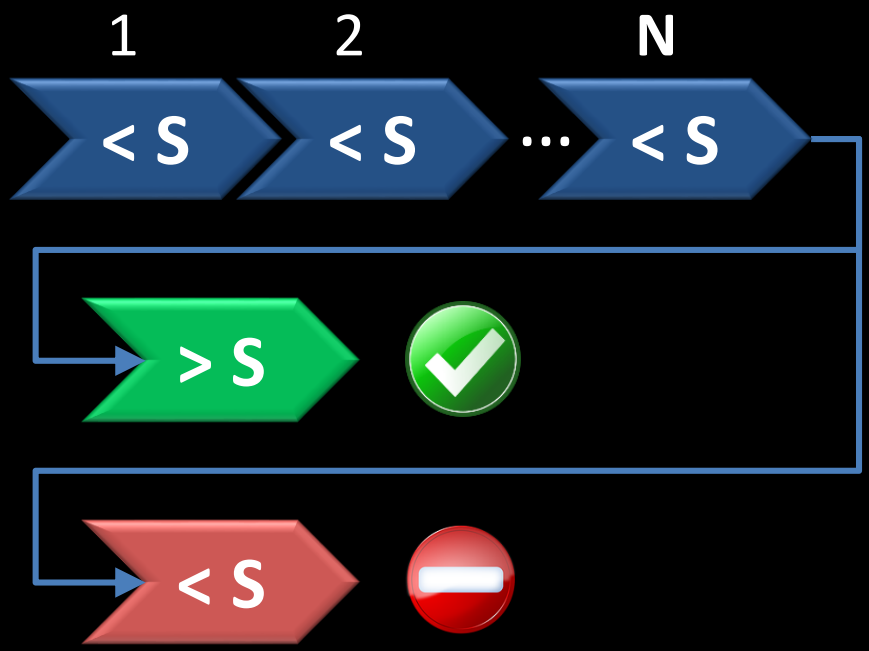
Main Coarse-Grained CFI Policies

- **CFI Policy 1: Call-Preceded Sequences**
  - Returns need to target a call-preceded instruction



- **CFI Policy 2: Behavioral-Based Heuristics**

Threshold Setting
kBouncer: (N=8; S<=20)
ROPecker: (N=11; S<=6)



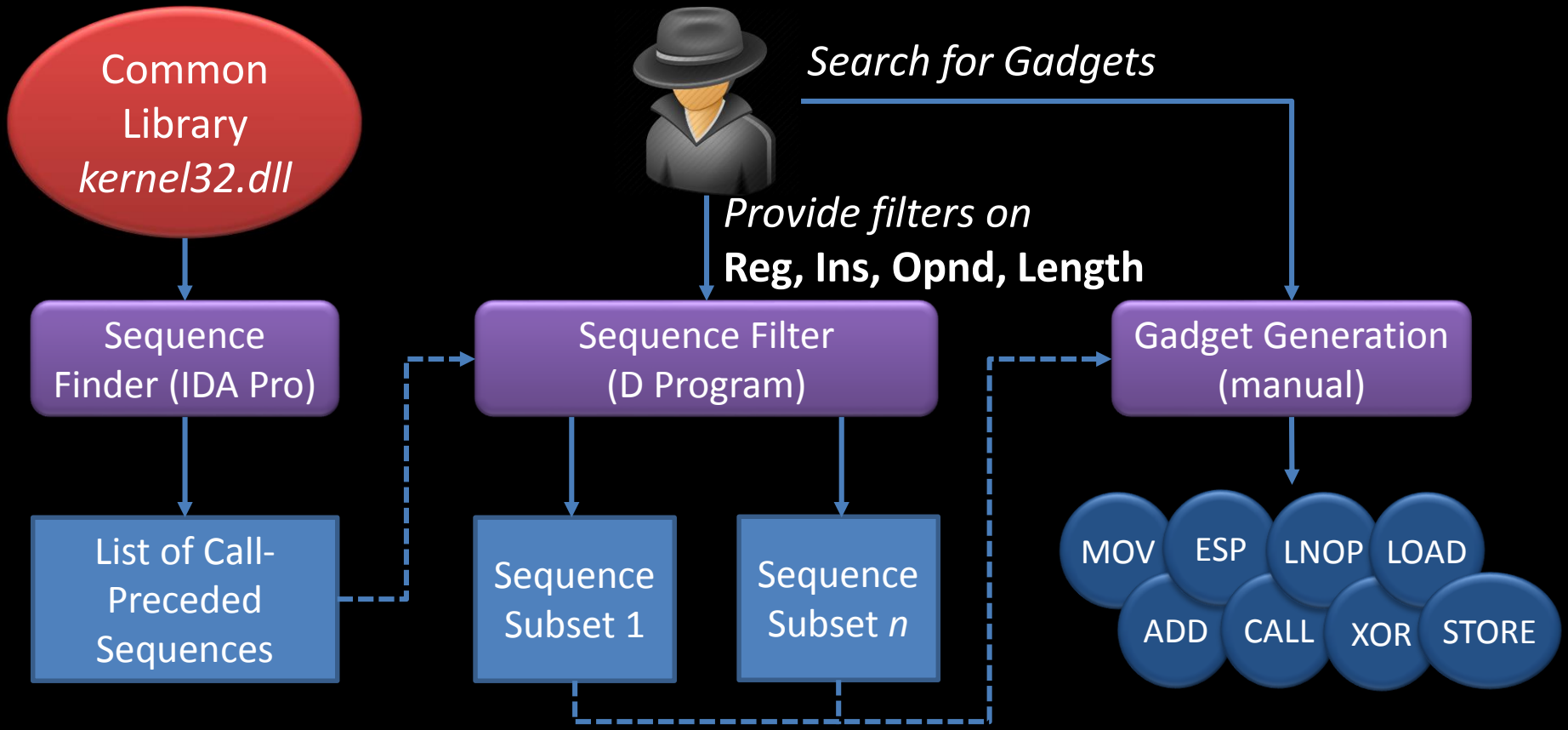
Most Restrictive Coarse-Grained CFI

CFI Policy	kBouncer	ROPecker	ROPGuard EMET	CFI for COTS Binaries	*Über-CFI*
CFI Policy 1 *Call-Preceded Sequences*	✓	⊖	✓	✓	✓
CFI Policy 2 *Behavioral-Based Heuristics*	✓	✓	⊖	⊖	✓
Time of CFI Check	WinAPI	2 Pages Sliding Window / Critical Functions	WinAPI/ Critical Functions	Indirect Branch	**Any Time**

⊖ No Restriction ✓ CFI Policy

Here only the core policies shown. However, we consider all other deployed policies in our analysis.

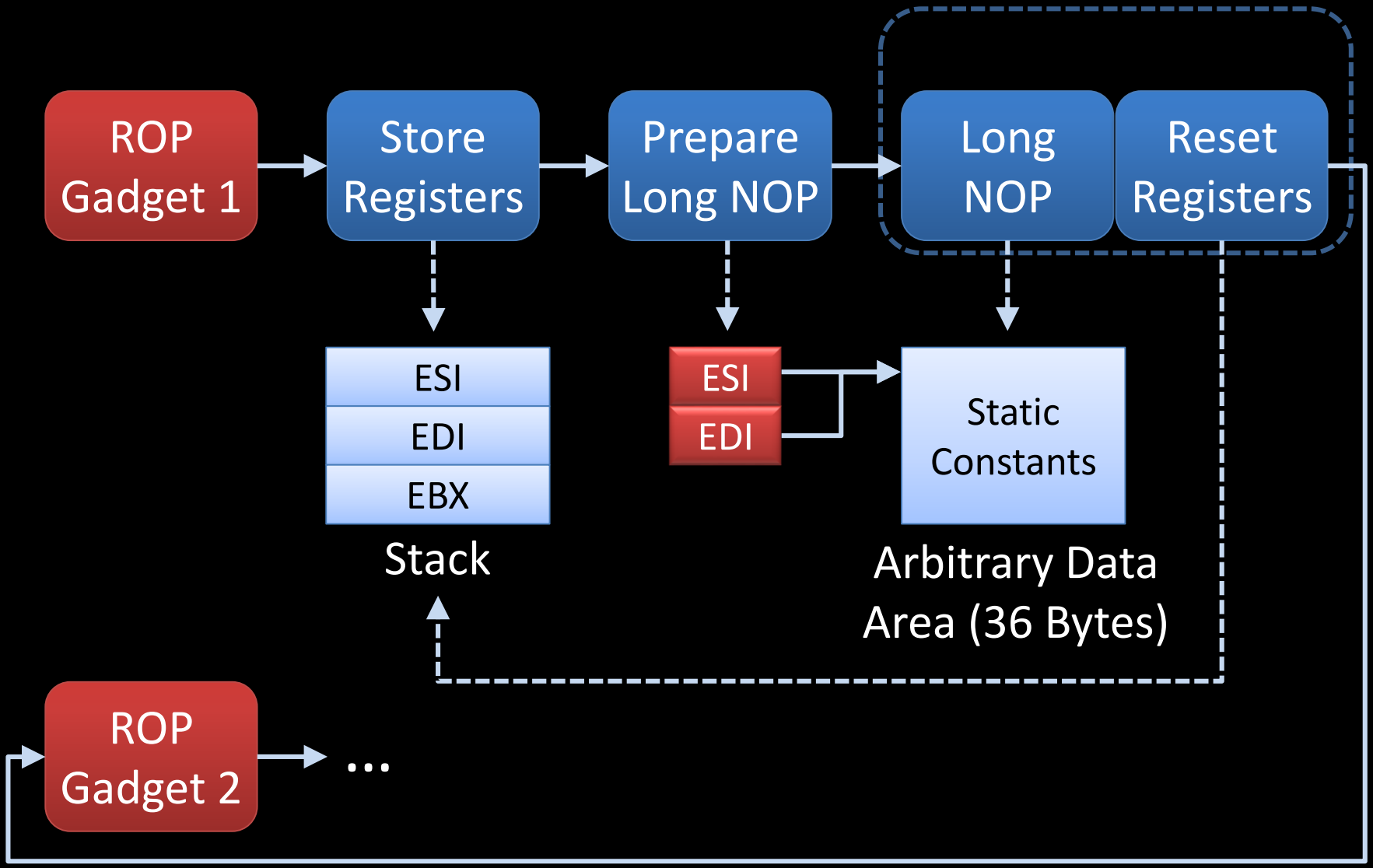
Our Methodology



(Excerpt of) Turing-Complete Gadget Set in CFI-Protected *kernel32.dll*

Gadget Type	CALL-Preceded Sequence *ending in a RET instruction*
LOAD Register	**EBP** := pop ebp **ESI** := pop esi; pop ebp **EDI** := pop edi; leave **ECX** := pop ecx; leave **EBX** := pop edi; pop esi; pop ebx; pop ebp **EAX** := mov eax,edi; pop edi; leave **EDX** := mov eax,[ebp-8]; mov edx,[ebp-4]; pop edi; leave
LOAD/STORE Memory	**LD(EAX)** := mov eax,[ebp+8]; pop ebp **ST(EAX)** := mov [esi],eax; xor eax,eax; pop esi; pop ebp **ST(ESI)** := mov [ebp-20h],esi **ST(EDI)** := mov [ebp-20h],edi
Arithmetic/ Logical	**ADD/SUB** := sub eax,esi; pop esi; pop ebp **XOR** := xor eax,edi; pop edi; pop esi; pop ebp
Branches	**unconditional branch 1** := leave **unconditional branch 2** := add esp,0Ch; pop ebp **conditional LD(EAX)** := neg eax; sbb eax,eax; and eax,[ebp-4]; leave

Long NOP Gadget



Real-World Exploitation

Adobe Reader 9.1
CVE-2010-0188



MPlayer Lite r33064 m3u
Buffer Overflow Exploit



Both detected by Microsoft EMET and coarse-grained CFI



Common Library *kernel32.dll* → Exploit Transformation → Successful Exploit *Details in Paper*

Lessons Learned

- **Fundamental problems of coarse-grained CFI**
  1. **Too many call sites available**
     - Turing-complete gadget set possible on small code base

     → Restrict returns to valid/hot call sites (shadow stack)
  2. **Heuristics are ad-hoc and ineffective**
     - Our generic Long NOP evades heuristics and handles all side effects

     → Adjusted sequence length leads to high false positive
  3. **Too many indirect jump and call targets**
     - Resolving indirect jumps and calls is non-trivial
     - Original CFI: Vulcan framework remains blackbox

     → Compromise: Compiler support

Summary and Future Work

- **Summary**
  - We systematically analyze the security of all recently proposed “practical” defenses against code reuse attacks
  - We show Turing-completeness of ROP even over less than 1MB code base
  - And, we provide real-world exploits
- **Our ongoing work: fine-grained CFI**
  - Hardware/Software Co-design [DAC 2014]; collaboration with Intel
  - Software Solutions; collaboration with Microsoft
- **Future work**
  - CFI for dynamic code or JIT-enabled applications
  - Improved code randomization schemes

Thank you!

For more information see

www.trust.cased.de

