Address-Oblivious Code Reuse: On the Effectiveness of Leakage-Resilient Diversity

Robert Rudd[1], Richard Skowyra[1], David Bigelow[1], Veer Dedhia[1], Thomas Hobson[1], Stephen Crane[2], Christopher Liebchen[4], Per Larsen[3], Lucas Davi[5], Michael Franz[3], Ahmad-Reza Sadeghi[4], Hamed Okhravi[1]

1: MIT Lincoln Laboratory 2: Immunant, Inc. 3: UC Irvine 4: TU Darmstadt 5: University of Duisburg-Essen

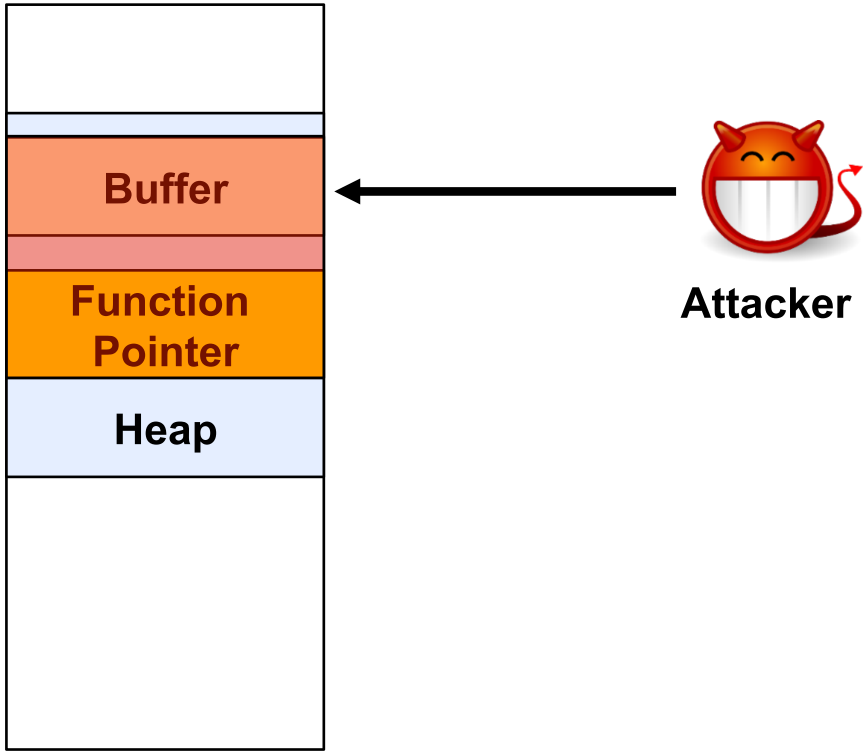
LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

This material is based upon work supported by the Assistant Secretary of Defense for Research and Engineering under Air Force Contract No. FA8721-05-C-0002 and/or FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Assistant Secretary of Defense for Research and Engineering.

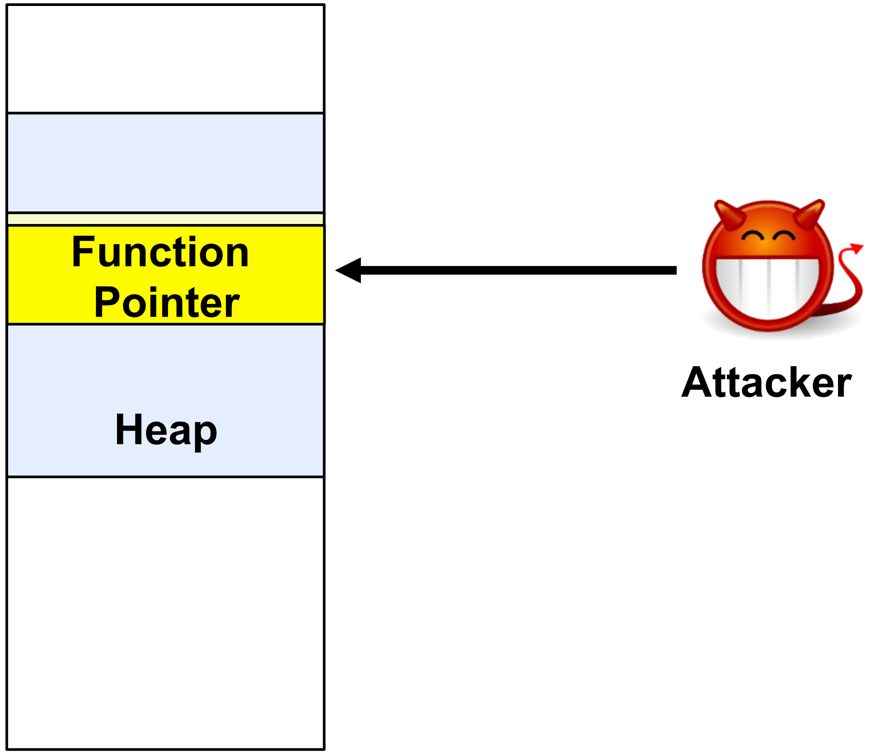
Bottom-Line Upfront

- **Code diversity techniques are vulnerable to information leakage**
- **Recent leakage-resilient techniques employ "execute-only" memory permissions to prevent information leakage**
- **We present a generic type of attack called *Address-Oblivious Code Reuse (AOCR)* that can bypass recent leakage-resilient techniques**
- **We provide 3 real-world exploits**

Memory Corruption Attacks

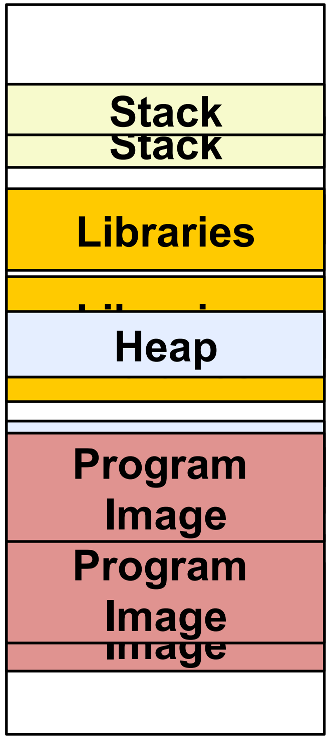


Spatial Memory Violation

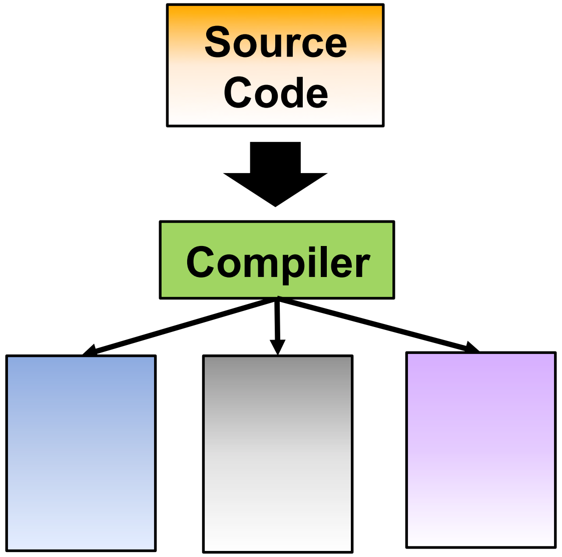


Temporal Memory Violation

Code Diversification Techniques

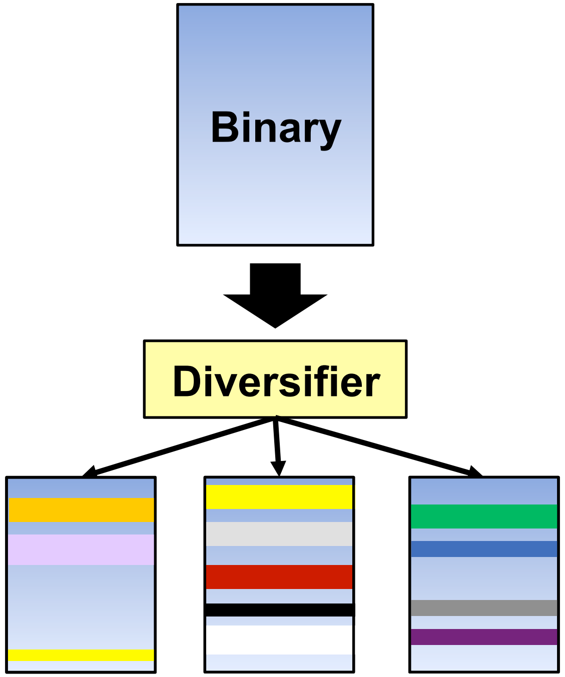


Address Space Layout Randomization (ASLR)



Diverse binaries

Compile-Time Diversity †



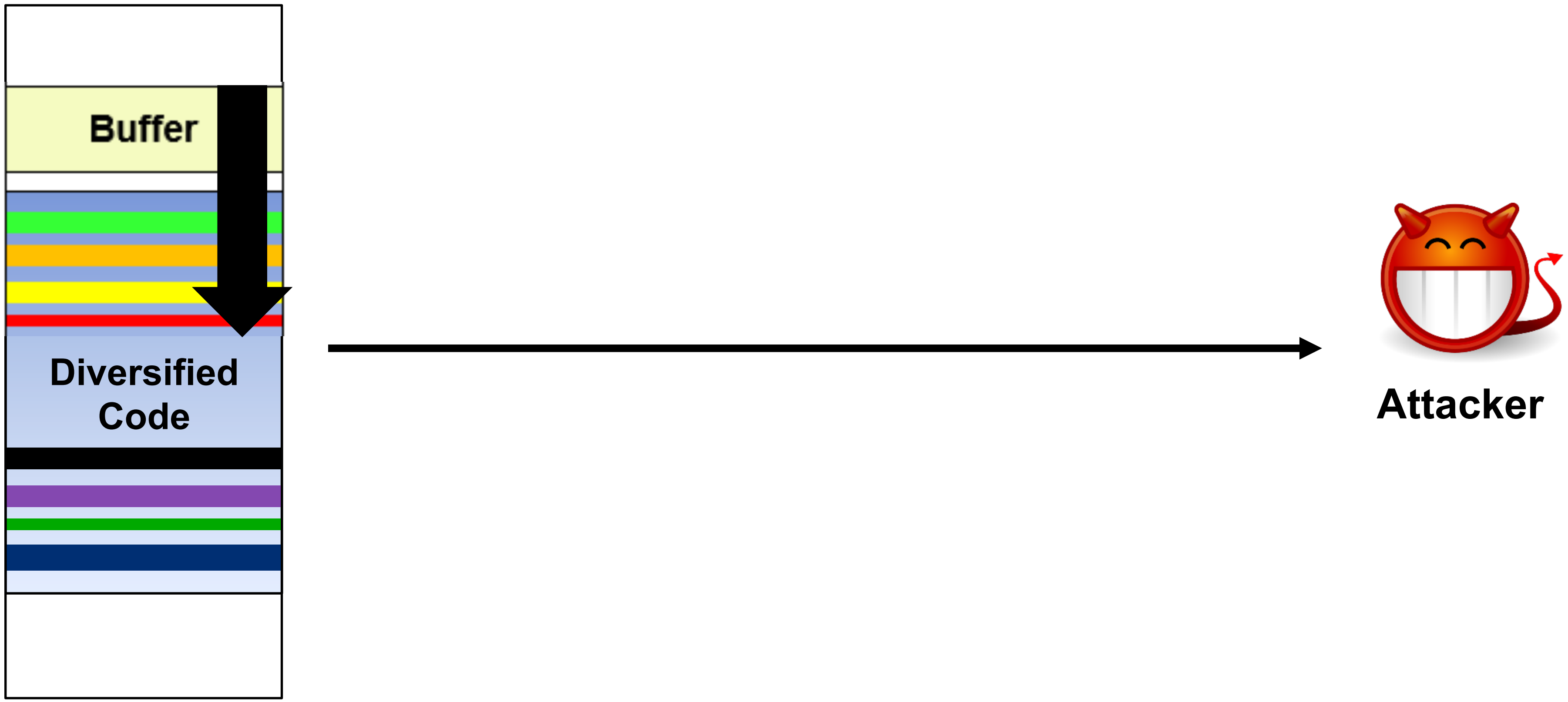
Diverse binaries

Binary Rewriting ‡

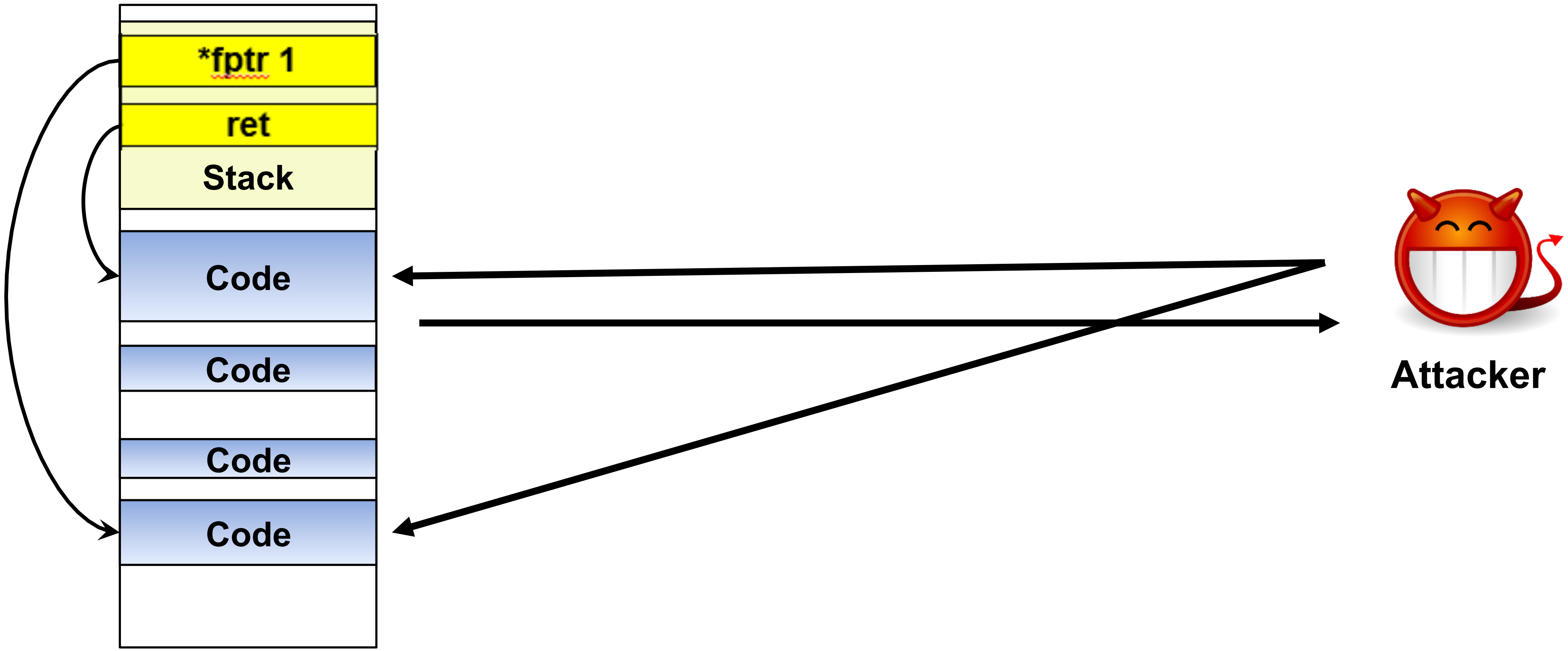
† T. Jackson, et al. "Compiler-generated software diversity." Moving Target Defense. Springer, 2011

‡ J. Hiser, et al. "ILR: Where'd My Gadgets Go?." IEEE S&P, 2012

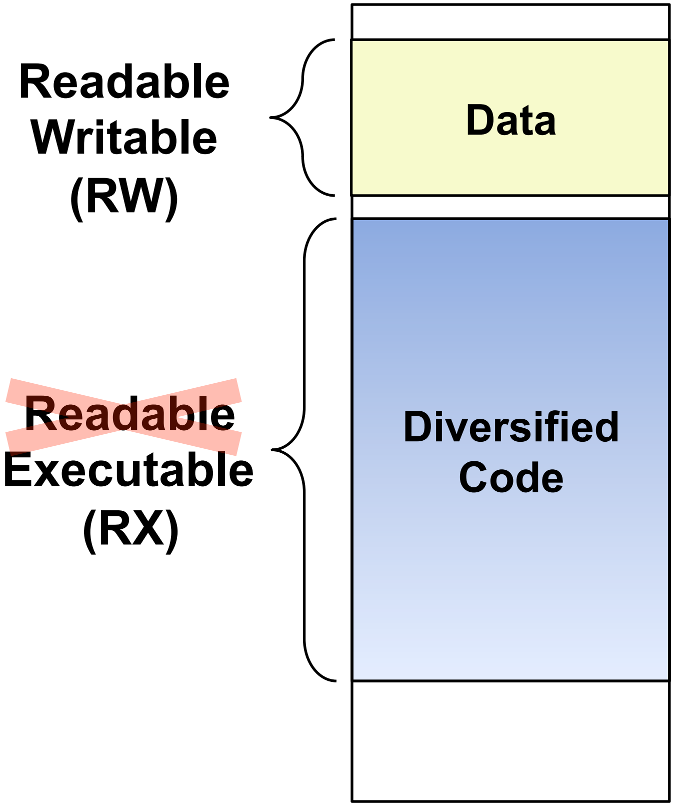
Information Leakage (Direct Memory Disclosure)



Information Leakage (Indirect Memory Disclosure)



Memory Permissions



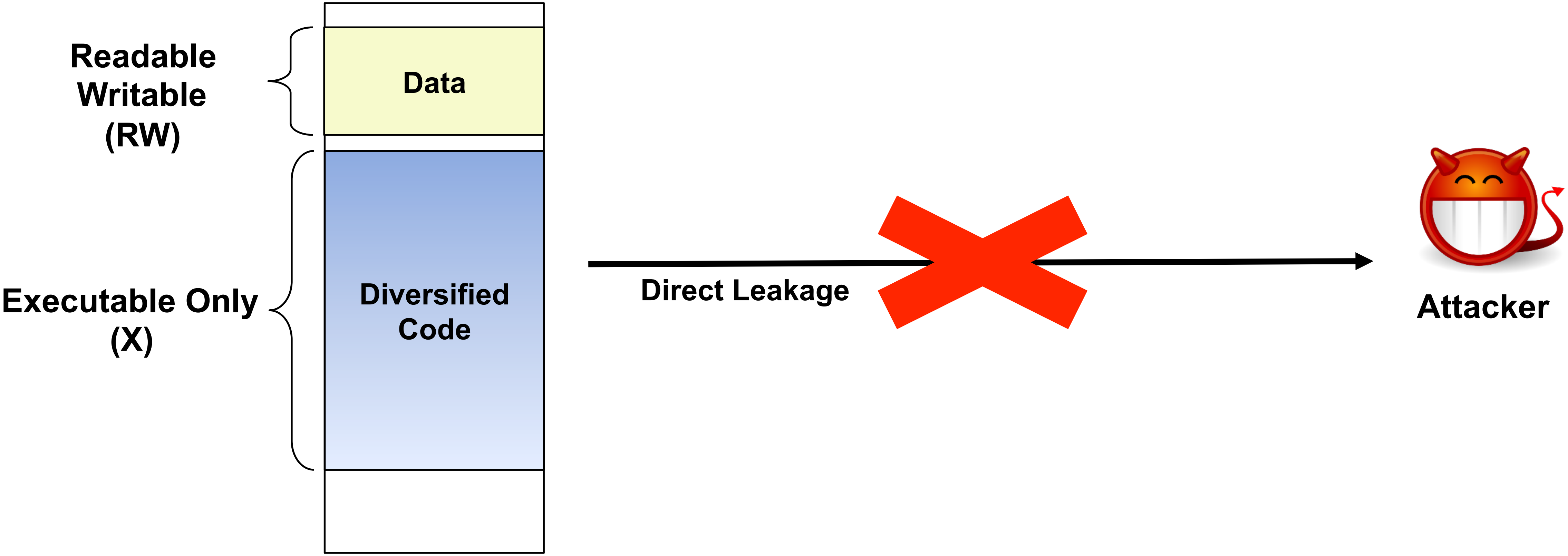
Readable
Writable
(RW)

Data

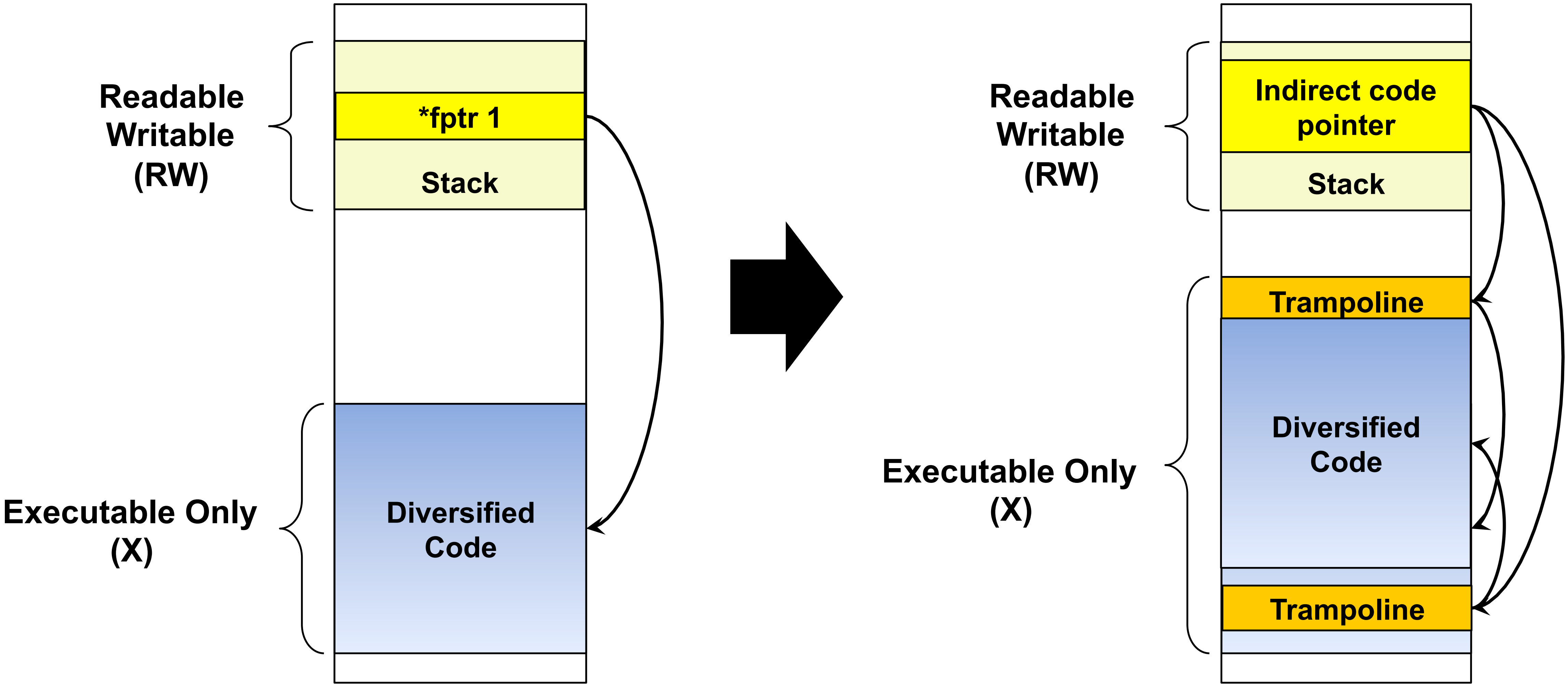
~~Readable~~
Executable
(RX)

Diversified
Code

Leakage Resilient Diversity



Indirect Leakage Prevention

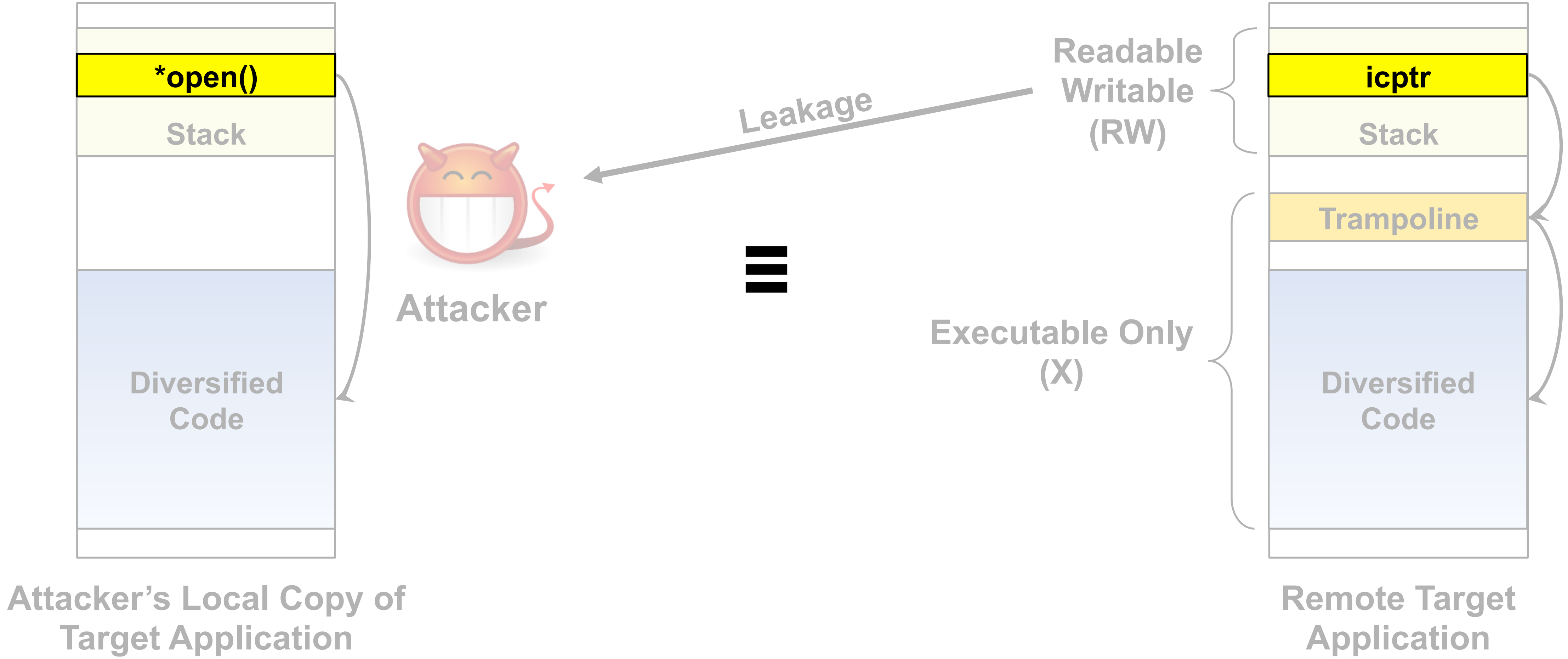


Research Questions

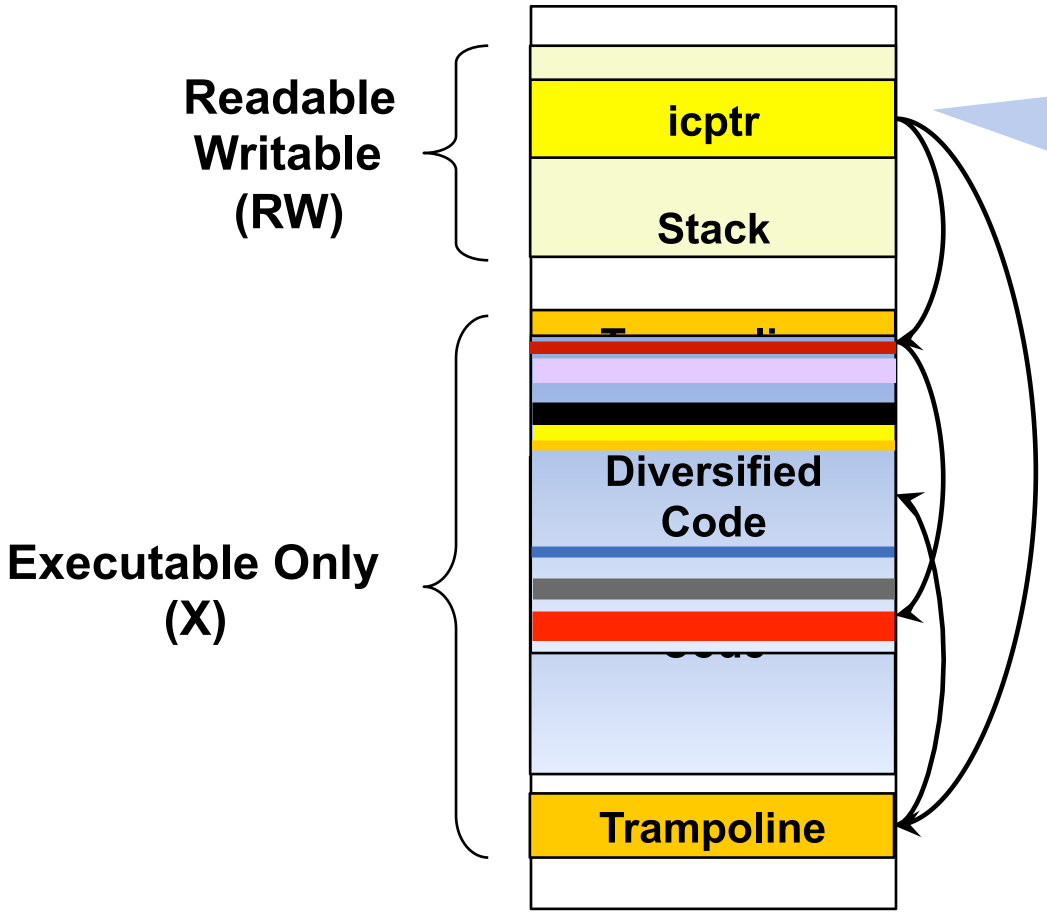
- Indirect code pointers create a surrogate for code
- Can attackers reuse code at the granularity of indirect code pointers?
- Can they accurately identify the corresponding functions?
- Can they chain indirect code pointers together?

Profiling Indirect Code Pointers

- Goal: identify the function corresponding to each indirect code pointer



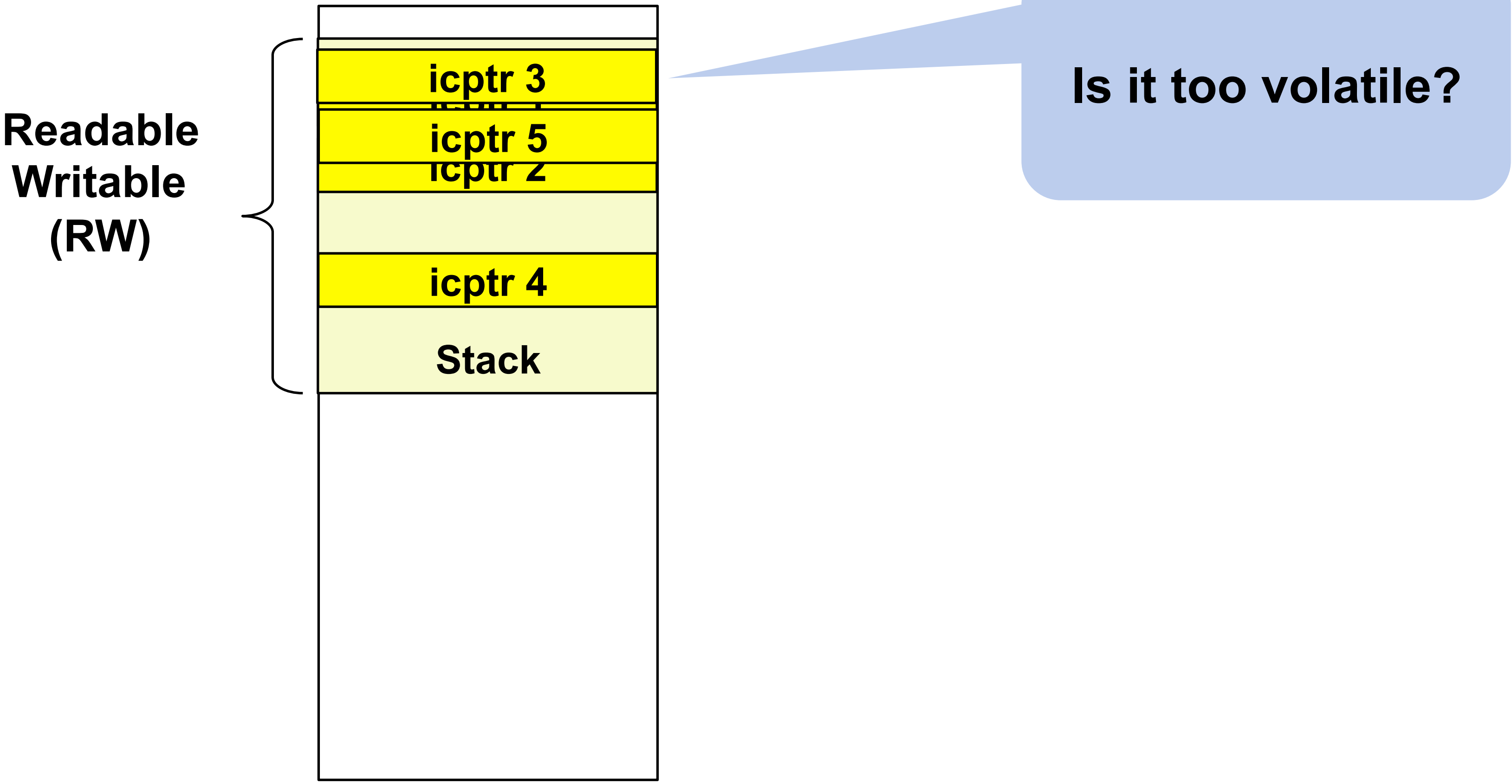
Address-Oblivious Code Reuse



Remote Target Application

Address-Oblivious Code Reuse (AOCR) Gadget

Accurate Profiling

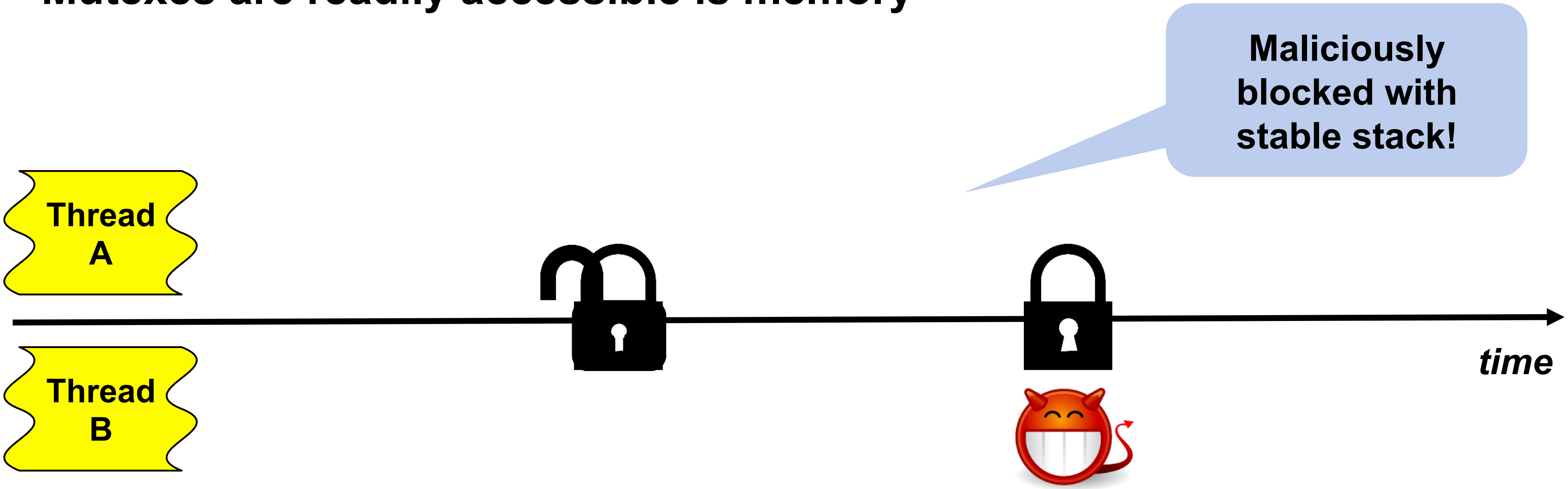


Is it too volatile?

Remote Target Application

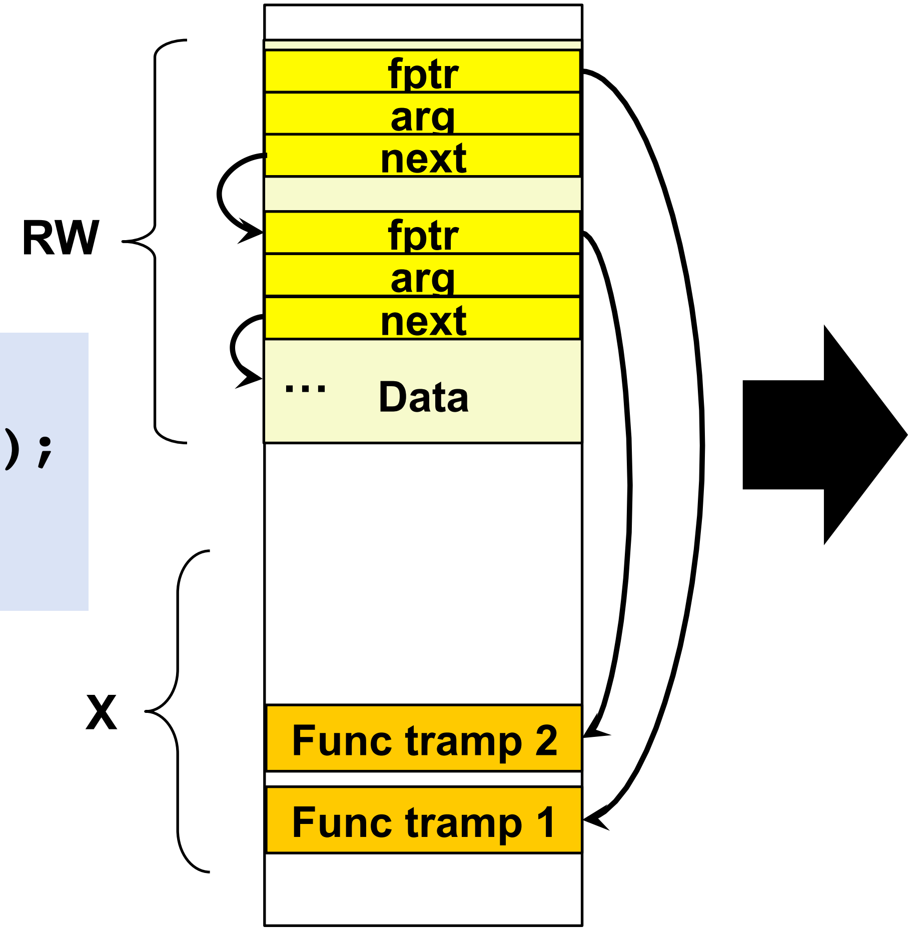
Accurate Profiling using Malicious Thread Blocking (MTB)

- A thread can force another threat to halt by maliciously setting a mutex
- Mutexes are readily accessible is memory



Chaining Gadgets Together using Malicious Loop Redirection (MLR)

```
while (task) {
   task->fptr(task->arg);
   task = task->next;
}
```



Normal Loop

Nginx Proof-of-Concept Exploit

1. **Locate a mutex for MTB**

2. **Profile an indirect code pointer for open (1st AOCR gadget)**

3. **Profile an indirect code pointer for _IO_new_file_overflow (2nd AOCR gadget)**

4. **Corrupt Nginx’s task queue to call our profiled trampolines using MLR**

Implementation Challenges of Execute-Only Permissions

- **Forged Direct Memory Access (FDMA)**
  - A malicious application forges a software-based DMA call to kernel
  - Uses O_DIRECT flag in Linux
  - DMA request bypasses memory permissions

- **Procfs**
  - Ubiquitous facility in Linux
  - Provides memory maps and addresses
  - Blocking it breaks many benign applications
  - Protections such as GRSecurity's permissions will not block it

Impact on Leakage-Resilient Diversity Techniques

	Direct Leak		Indirect Leak	
	TLB-mediated (Buffer Overread)	Non-TLB-mediated (Forged DMA)	Code Pointer Leak (Ret address leak)	Indirect Code Pointer Leak (AOCR)
PointGuard			●	
Oxymoron			●	
Isomeron			●	
XnR	●			
HideM	●			
Readactor	●		●	
Heisenbyte	●			
NEAR	●			
ASLR-Guard			●	
TASR	◐	◐	◐	◐

Possible Countermeasures

- Complete memory safety
- Data randomization
- Authentication of indirect calls and returns
  - Use HMAC tokens to disallow redirection of indirect code pointers
  - Similar to cryptographically-enforced CFI (CCFI)

Conclusion

- **Code pointers pose a major challenge to leakage-resilient diversity**
- **AOCR attacks bypass code pointer obfuscation by profiling indirect code pointers**
- **Malicious threat blocking (MTB) allows accurate profiling**
- **Malicious loop redirection (MLR) allows chaining AOCR gadgets**
- **Effective defenses should incorporate aspects of *diversification* and *enforcement***

Questions?