Data Confidentiality and Integrity

Scott A. Carr and Mathias Payer

Motivation:
Protect integrity and confidentiality of select data from memory safety vulnerabilities

Background:
- Vulnerabilities -> Memory errors
- Complete protection expensive
- SoftBound: 112% for SPEC CPU [1]

Insights:
- Not all data critical/sensitive
- Overhead proportional to amount of protected data

Idea:
- Programmer decides what is protected
- Annotations in C/C++
- Enforcement: compiler plugin, runtime

Implementation:
- LLVM Pass
- Runtime library creates and maintains metadata for each protected variable
- Memory regions enforced with SFI

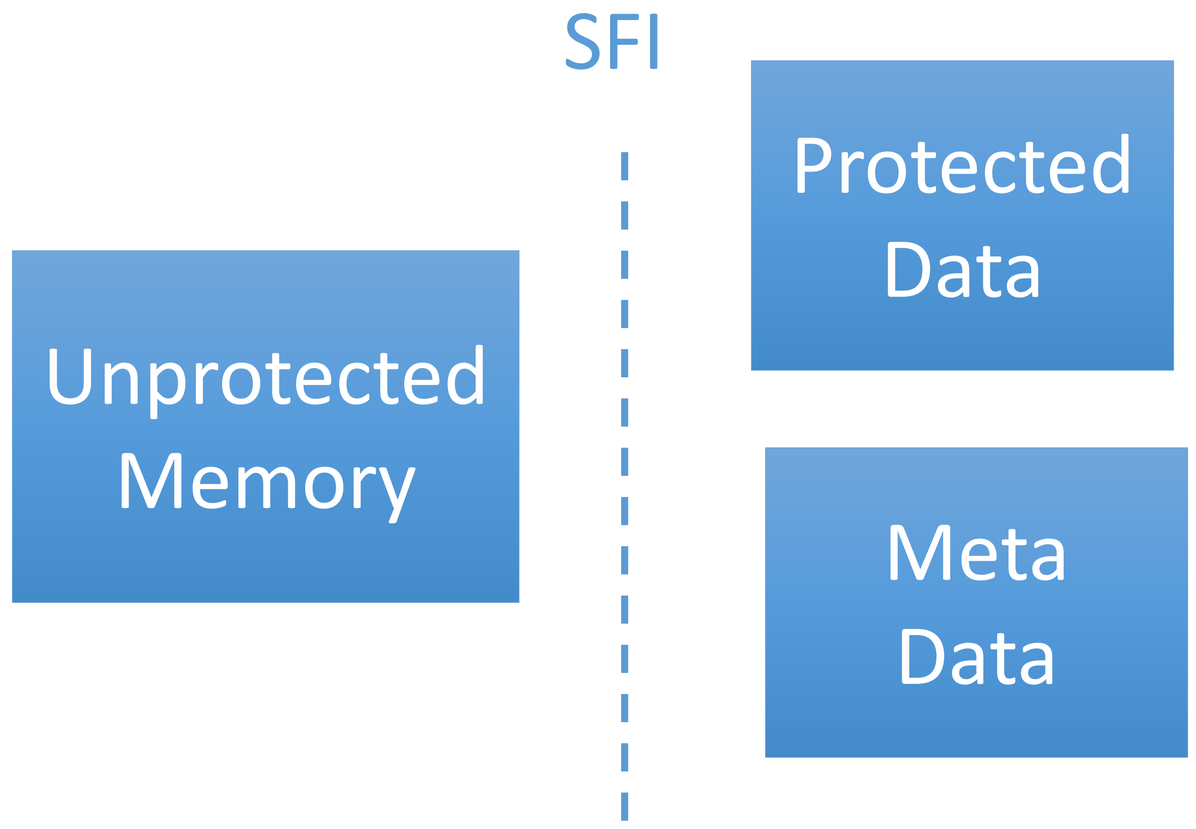
Case Study – PolarSSL:
- Prototype instruments library
- Passes all tests
- Lower overhead than SoftBound

Ongoing Work:
- Aggressive in-lining and optimization of security checks
- Automatically identify sensitive variables

```
void vulnerable() {
  struct key *secret;
  int cmd[5];
  secret = load_key();
  input(cmd); // vulnerability
}
```

sensitive key *secret;

	x Slow Down
DCI	7.28
SoftBound	11.4



1. SoftBound: Highly Compatible and Complete Spatial Memory Safety for C. Santosh Nagarakatte et al. PLDI 2009
2. Code Pointer Integrity. Kuznetsov et. al. OSDI 2014

PURDUE UNIVERSITY

CERIAS Center for Education and Research in Information Assurance and Security