MPTEE: Bringing Flexible and Efficient Memory Protection to Intel SGX

Wenjia Zhao[1,2], Kangjie Lu[2], Yong Qi[1], Sqiyu Qi[3]

[1]Xi'an Jiaotong University, China

[2]University of Minnesota, USA

[3]Xidian University, China

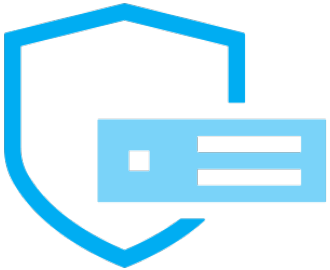
EuroSys'20, April 27–30, 2020

Intel Software eXecute Guard (SGX)

- Hardware-based trusted execution environment
- Provide secure region, namely enclave
- Enhance Application Security

Secure Cloud Services

Blockchain



Edge Computing

Digital Wallet

Two types of SGX Research

Applications (to protect data/code)

- VC3 [OAKLAND'15]
- SCONE [OSDI'16]
- JITGuard [CCS'17]
- SGXCrypter [ASP-DAC'17]

Protection/attack to SGX itself

- Page Fault [OAKLAND'15]
- SGX-Shield [NDSS'17]
- SGXBOUNDS [EUROSYS'17]
- Side-channel [OAKLAND'18, SECURITY'17]

Examples and current disadvantages

SGXCrypter protects code by unpacking the packed code in enclave.

- relies on the OS page table to remove the *W* perm of unpacked code
- is incompatible with the SGX security model

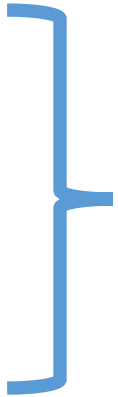
SGX-Shield protects SGX code itself through randomization

- uses software-based DEP to create an Non-RW boundary(R15)
- wastes the R15 register
- NRW boundary using a general register can be shifted[security'18]

Two types of SGX Research

Applications (to protect data/code)

- VC3 [OAKLAND'15]
- SCONE [OSDI'16]
- JITGuard [CCS'17]
- SGXCrypter [ASP-DAC'17]



flexibly and securely enforcing memory-page permissions

Protection/attack to SGX itself

- Page Fault [OAKLAND'15]
- SGX-Shield [NDSS'17]
- SGXBOUNDS [EUROSYS'17]
- Side-channel [OAKLAND'18, SECURITY'17]

Unfortunately, the feature is missing

Unfortunately, the feature is missing

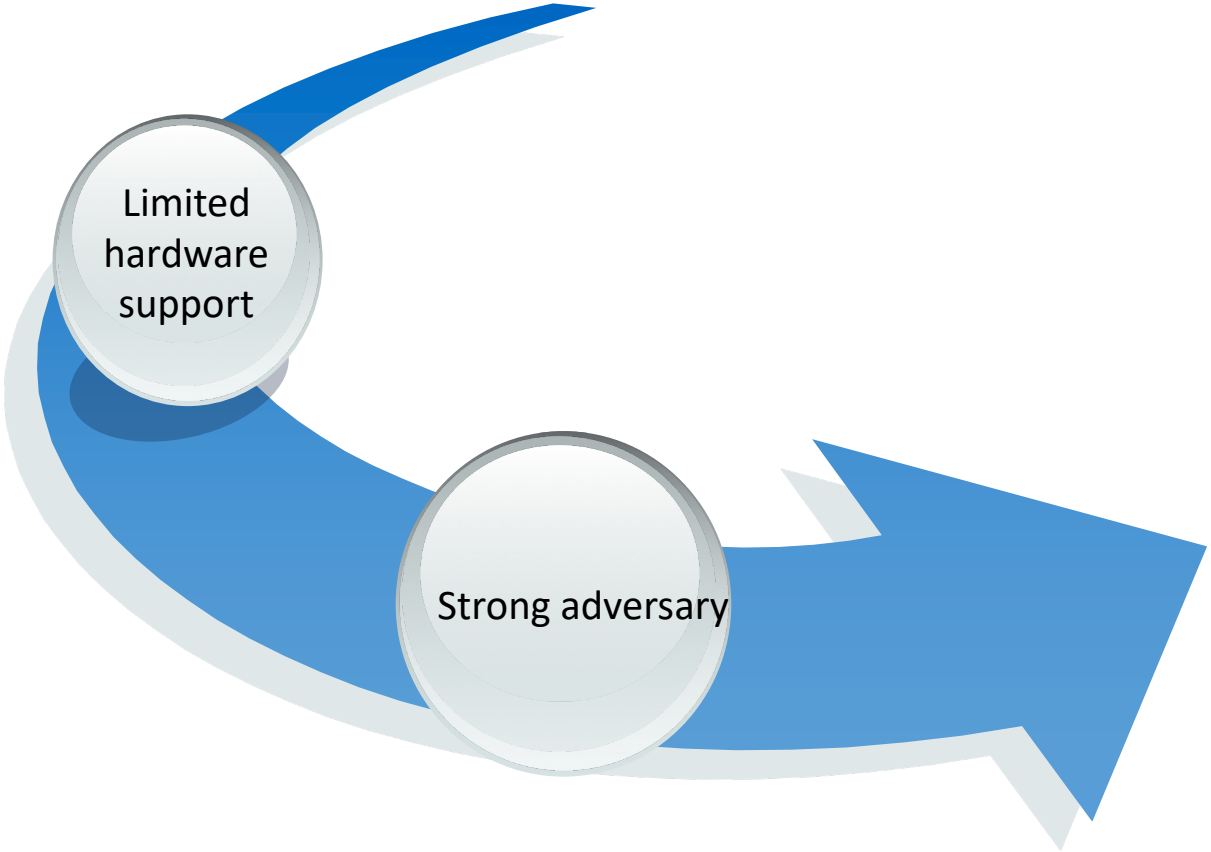
Why?

Unfortunately, the feature is missing
Why?

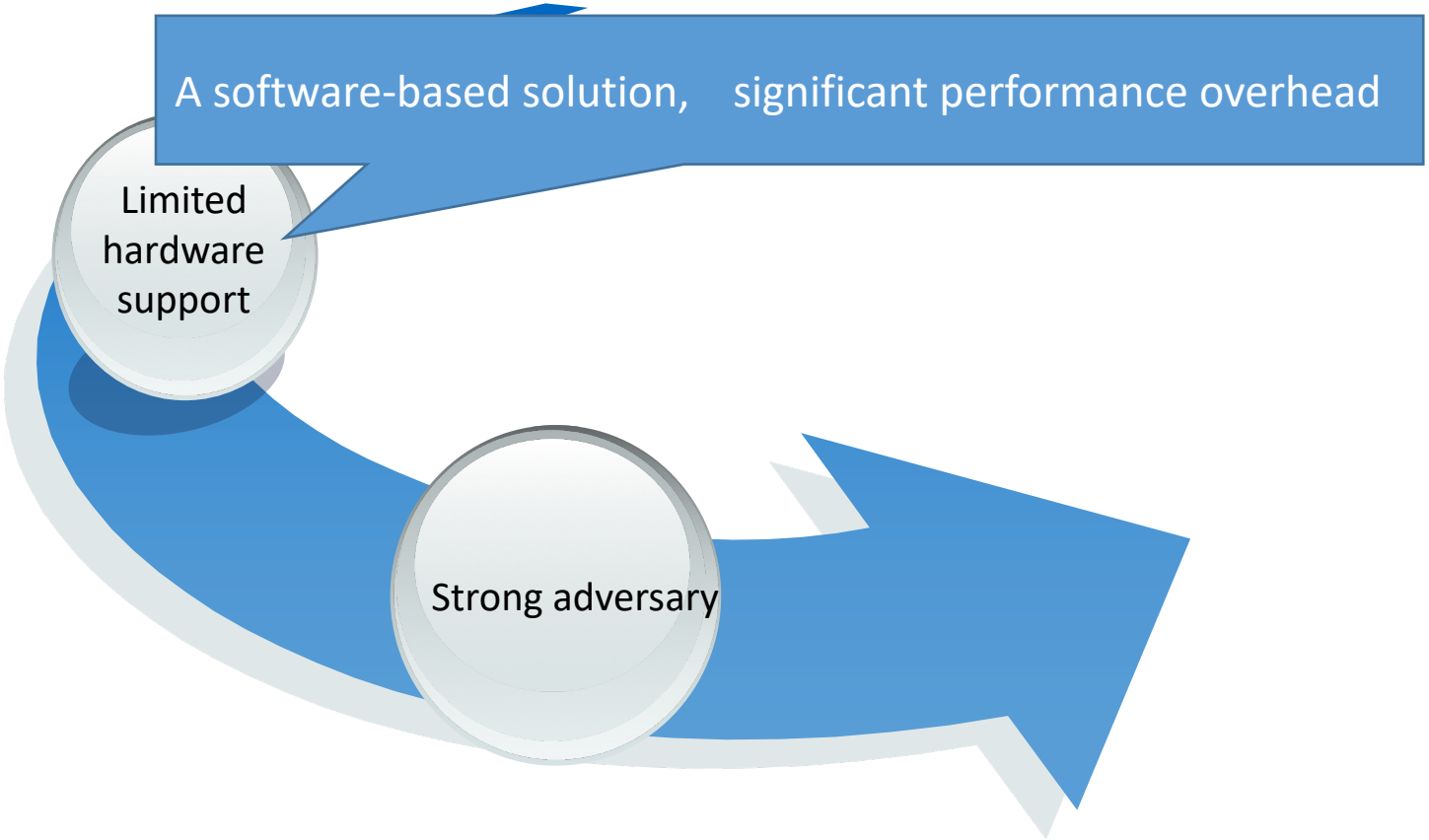
Security considerations (untrusted os)

Permissions are statically decided (sign-verify)

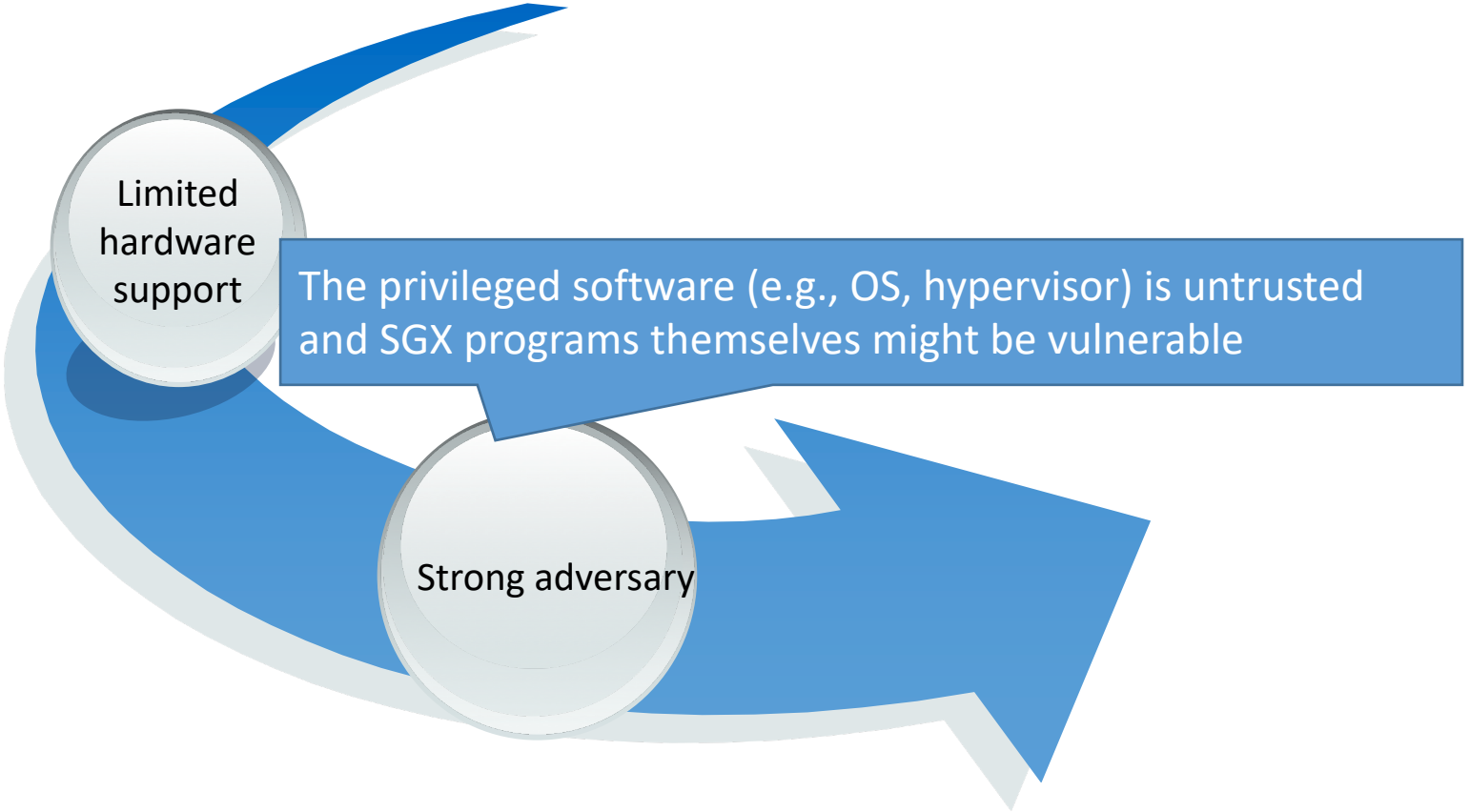
Challenges



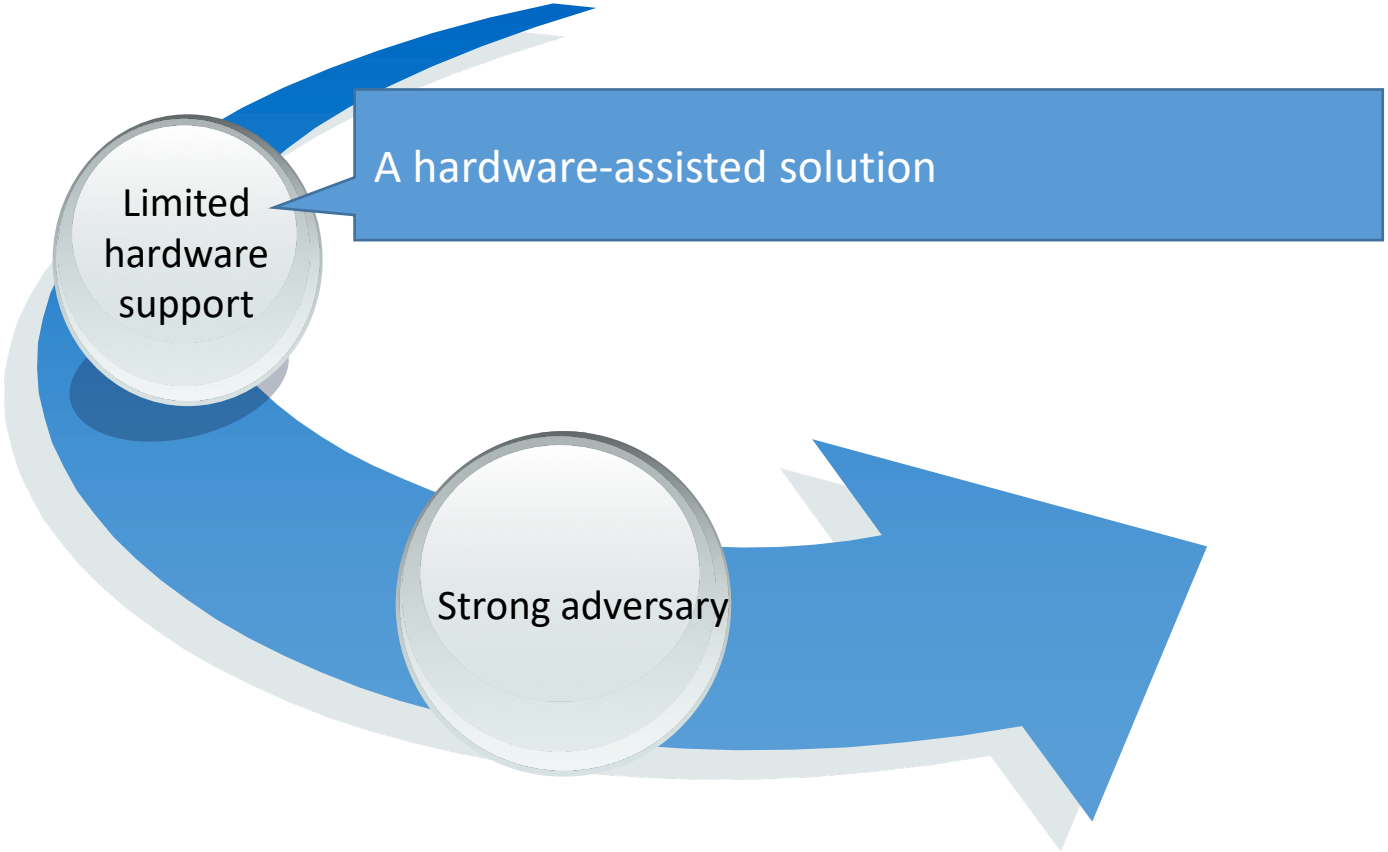
Challenges


A software-based solution, significant performance overhead
Limited hardware support
Strong adversary

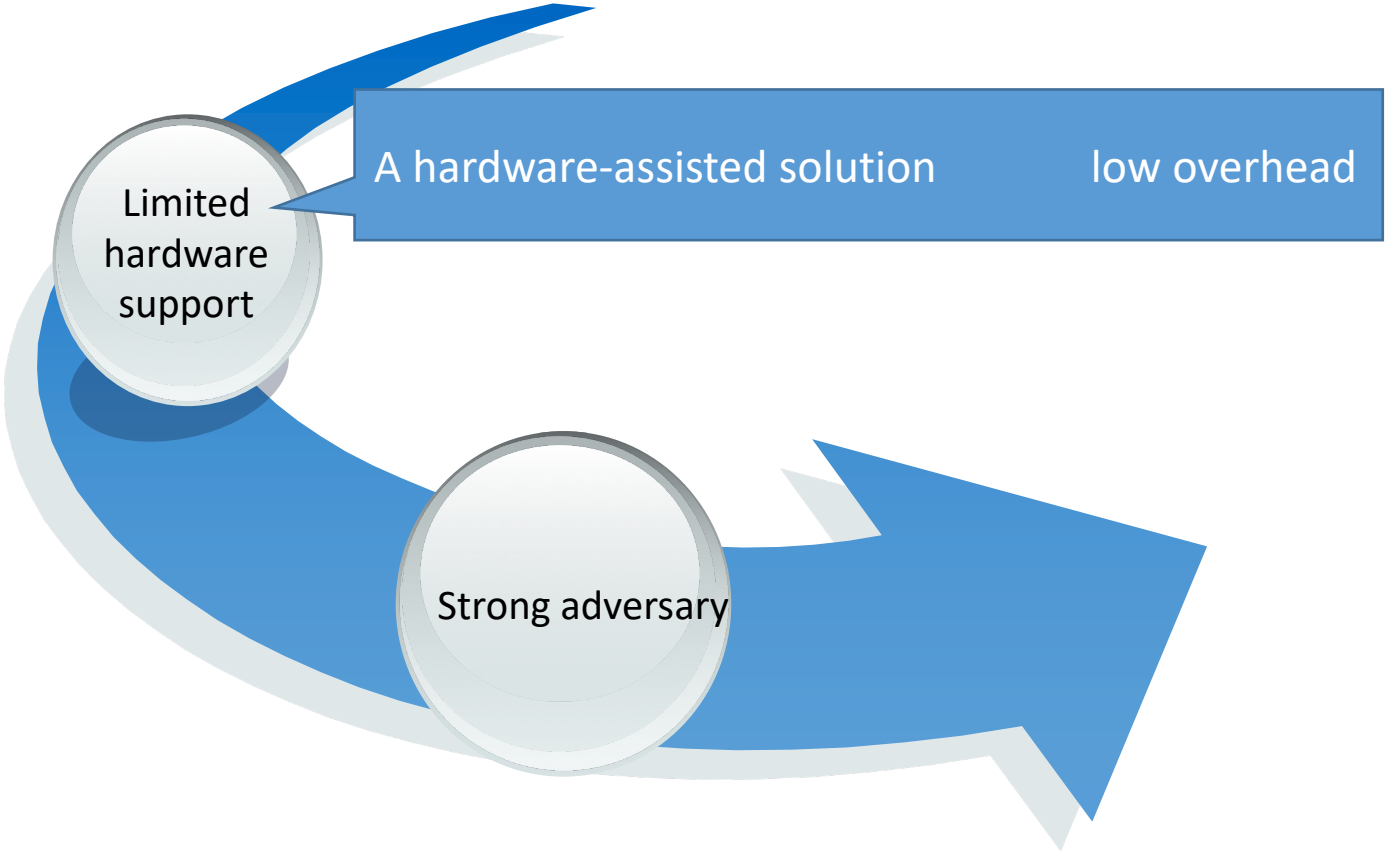
Challenges


Limited hardware support
The privileged software (e.g., OS, hypervisor) is untrusted and SGX programs themselves might be vulnerable
Strong adversary

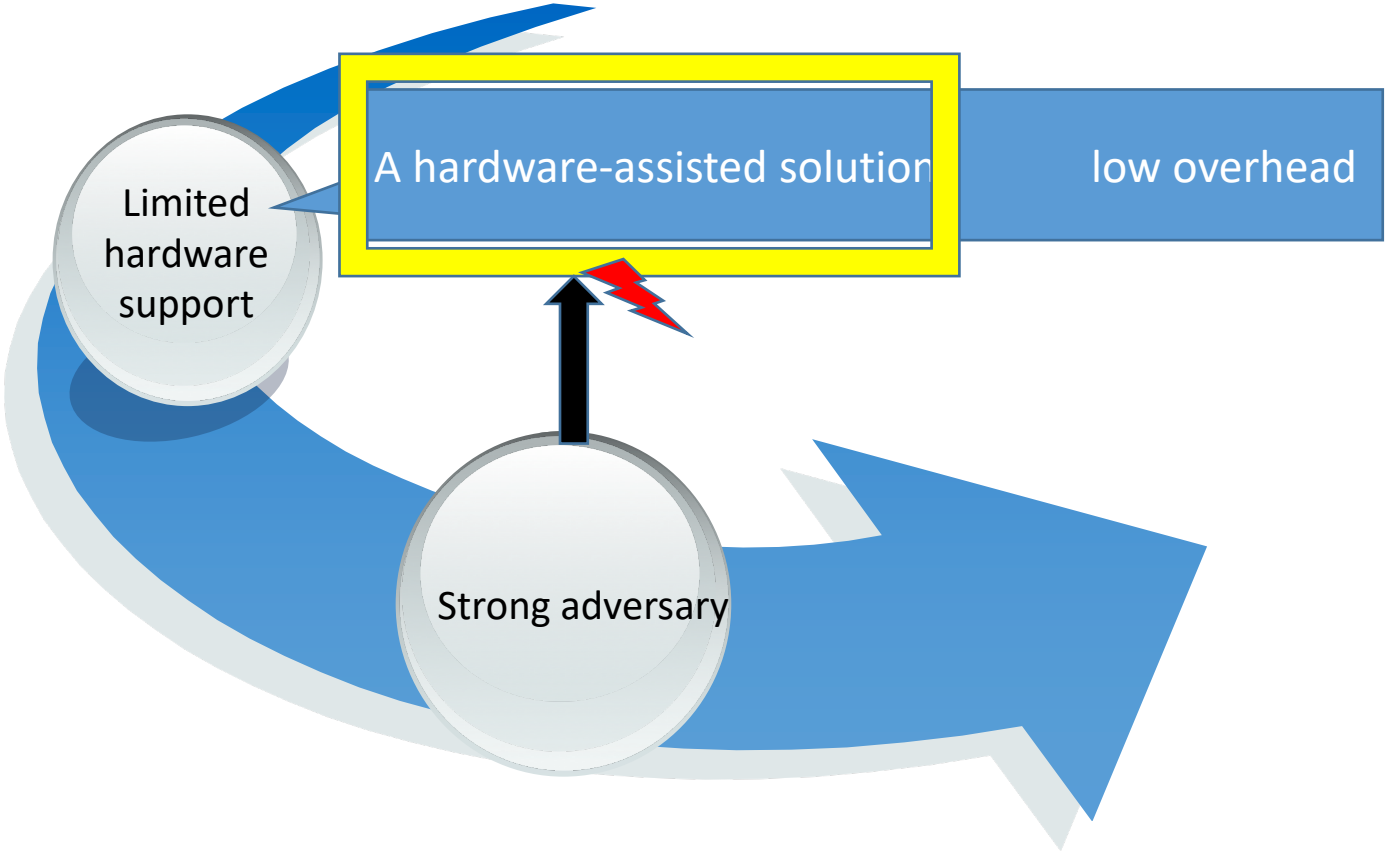
Challenges


Limited hardware support
A hardware-assisted solution
Strong adversary

Challenges


A hardware-assisted solution
low overhead
Limited hardware support
Strong adversary

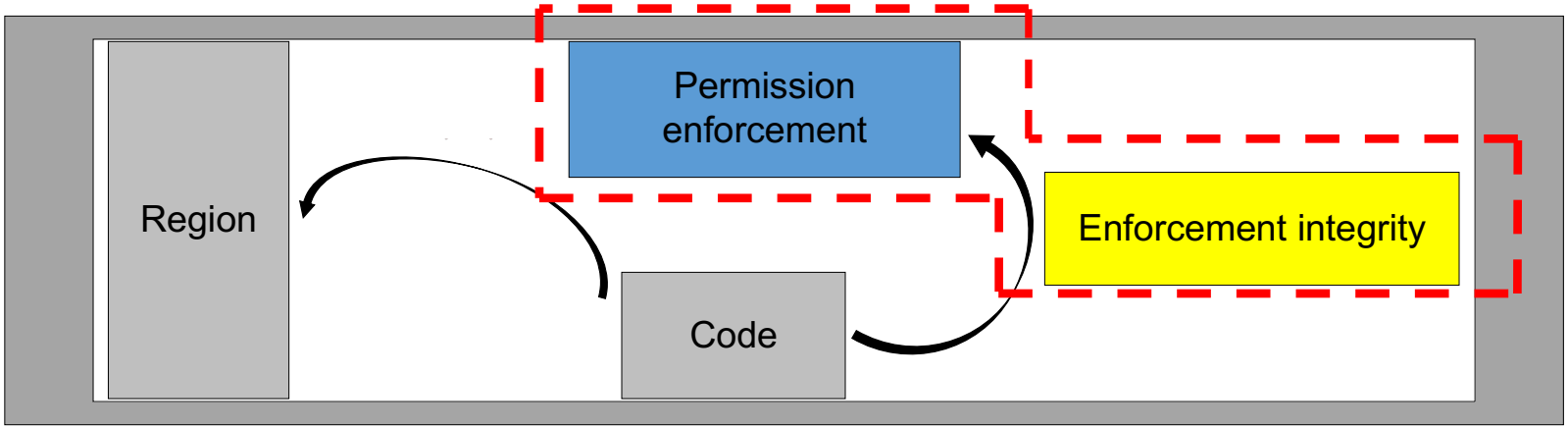
Challenges


A hardware-assisted solution
low overhead
Limited hardware support
Strong adversary

MPTEE: memory permission protection

Flexible, efficient, and isolated memory permission enforcement for SGX.

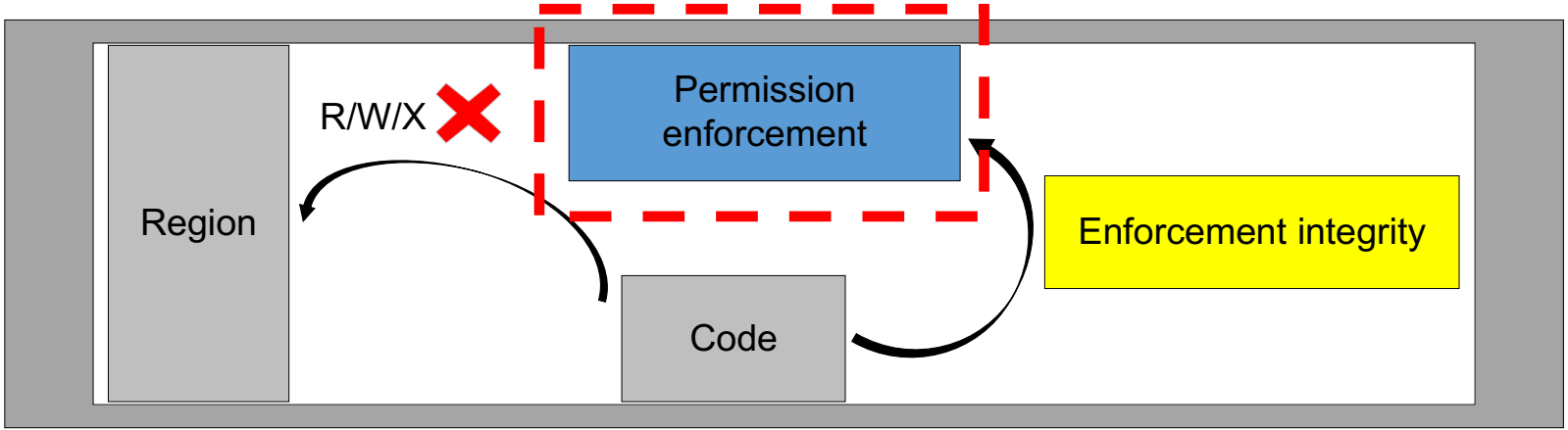
- Flexible and Efficient Memory-Permission Enforcement
- Enforcement Integrity



MPTEE: memory permission protection

Flexible, efficient, and isolated memory permission enforcement for SGX.

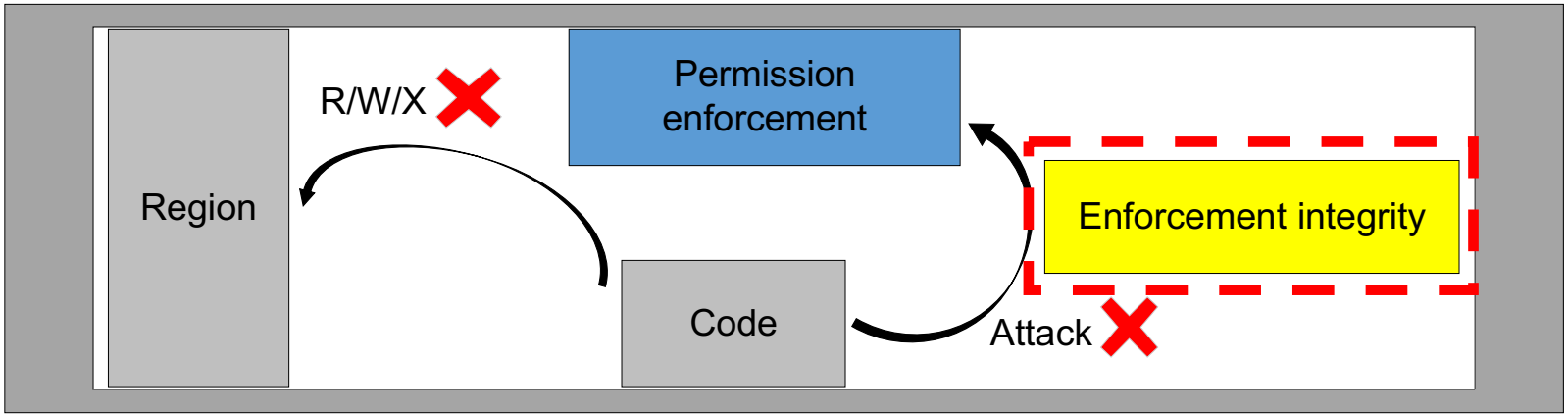
- Flexible and Efficient Memory-Permission Enforcement
- Enforcement Integrity



MPTEE: memory permission protection

Flexible, efficient, and isolated memory permission enforcement for SGX.

- Flexible and Efficient Memory-Permission Enforcement
- Enforcement Integrity



Memory-Permission Enforcement

Elastic Cross-Region Bound Check(CRBC)

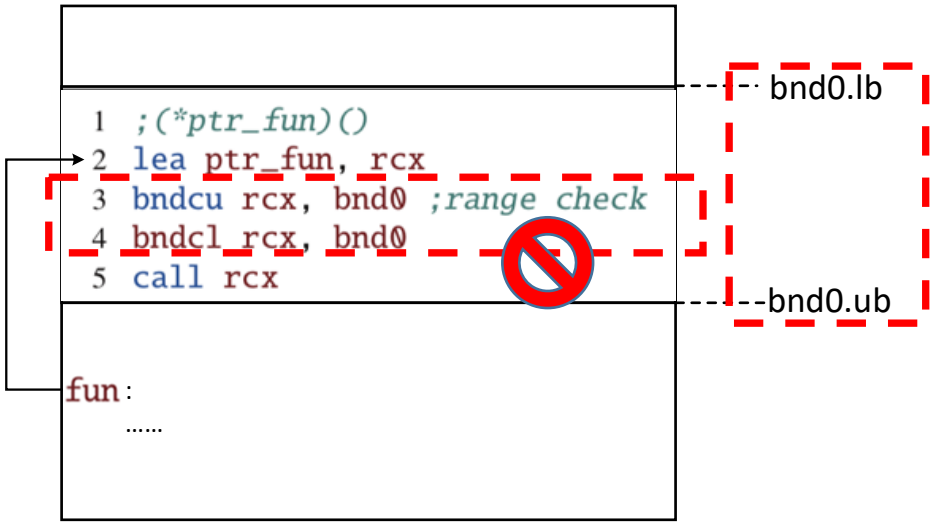
Basic idea

Use hardware-assisted technique(MPX) to bound-check access

Elastic Cross-Region Bound Check(CRBC)

Memory Protection Extension(MPX)

- New instructions, bndcu, bndcl, bndmk…
- Four dedicated bound registers (BND0~BND3)



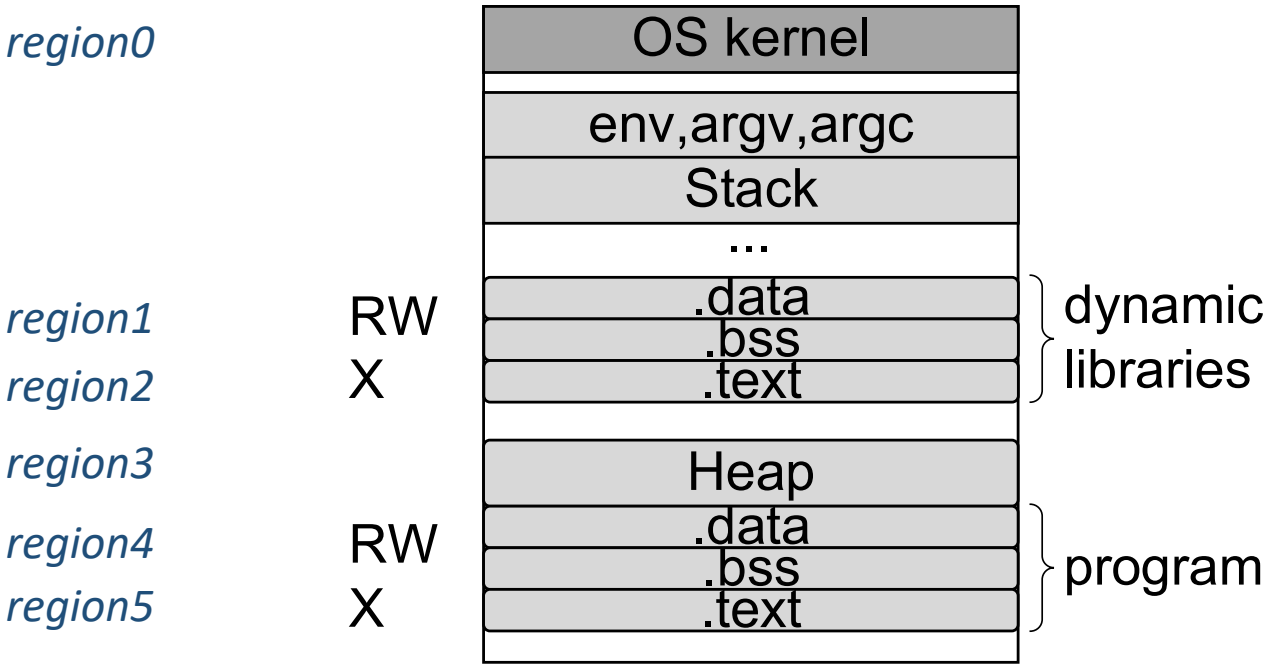
Elastic Cross-Region Bound Check(CRBC)

Memory Protection Extension(MPX)

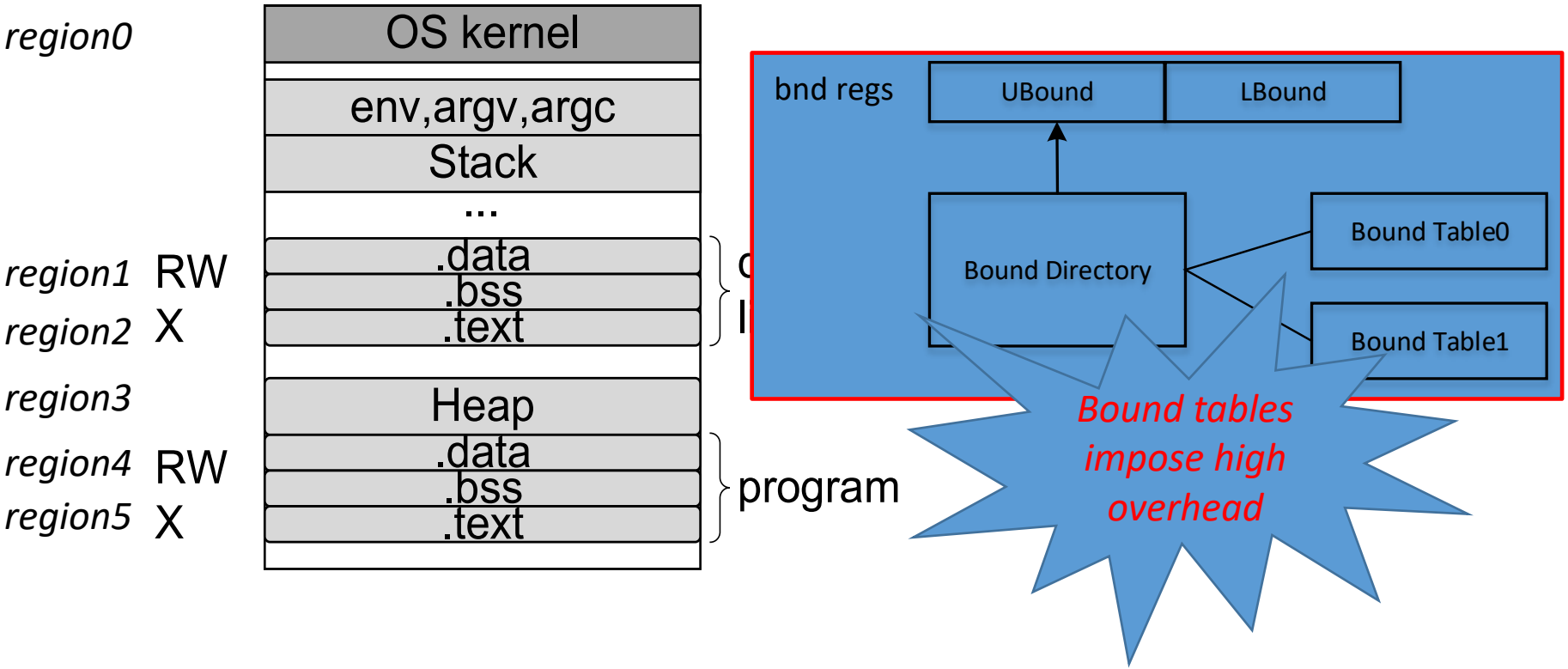
- New instructions, bndcu, bndcl, bndmk…
- Four dedicated bound registers (BND0~BND3)
- *More bounds will be stored in a bound table in memory*

Significant performance overhead (over 60%)

Elastic Cross-Region Bound Check(CRBC)



Elastic Cross-Region Bound Check(CRBC)



Elastic Cross-Region Bound Check(CRBC)

Elastic Cross-Region Bound Check(CRBC)

Key observation

The same permission memory range is continuous in an enclave

Elastic Cross-Region Bound Check(CRBC)

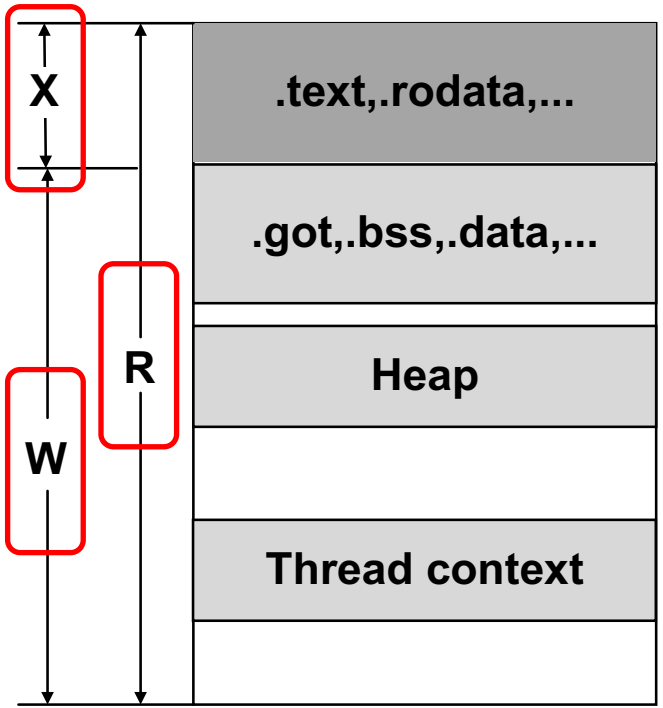
Key observation

The same permission memory range is continuous in an enclave

Because

All required libraries must be statically linked in the target enclave program

Elastic Cross-Region Bound Check(CRBC)



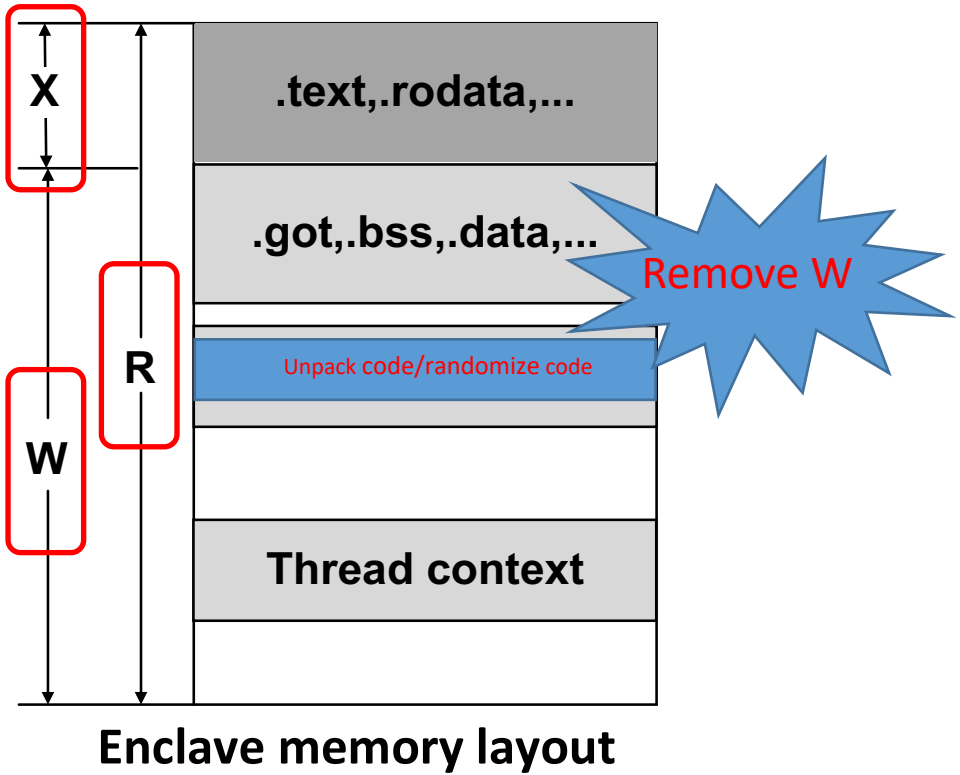
Enclave memory layout

Permission change

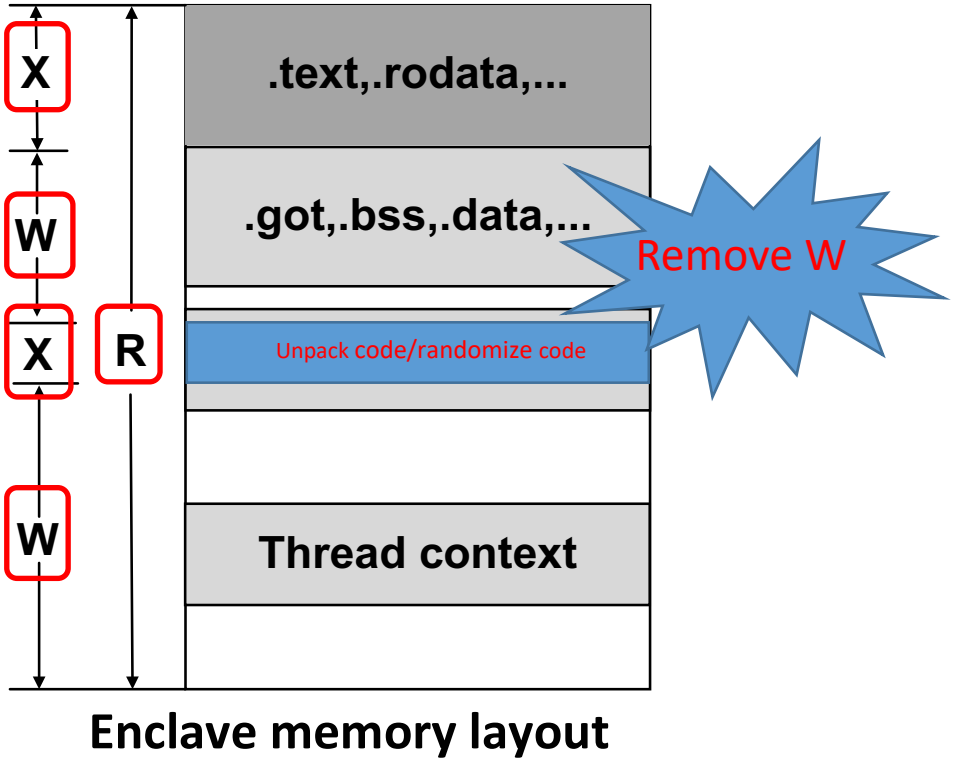
Continuous → Non-continuous

Exceeded the number of MPX registers

Elastic Cross-Region Bound Check(CRBC)

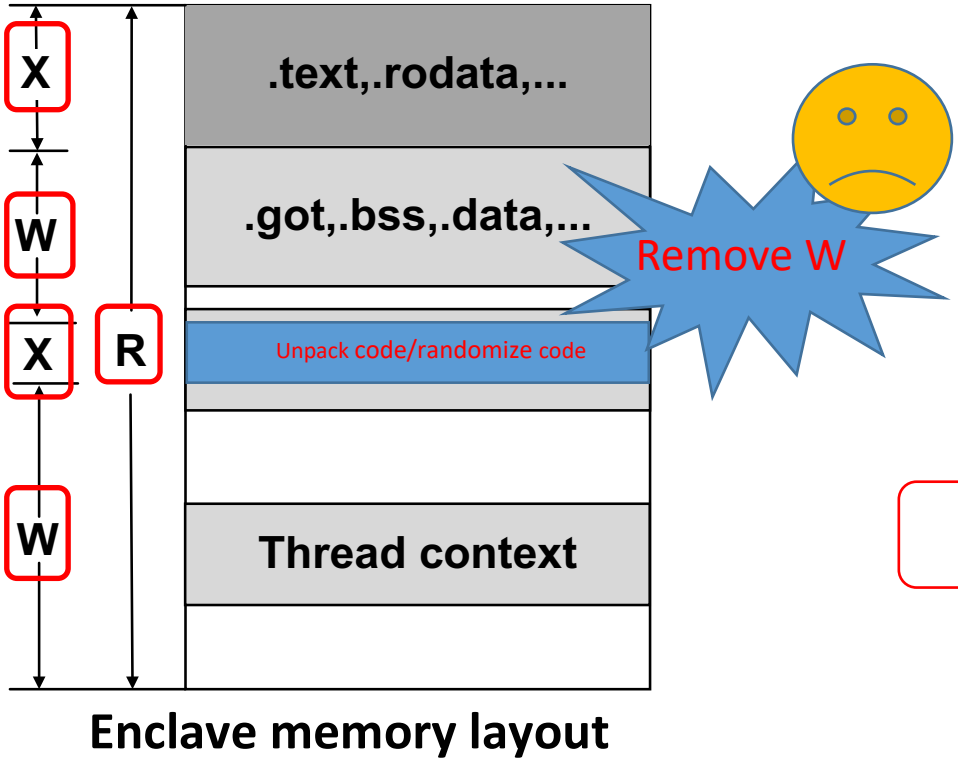


Elastic Cross-Region Bound Check(CRBC)



3 regions → 5 regions

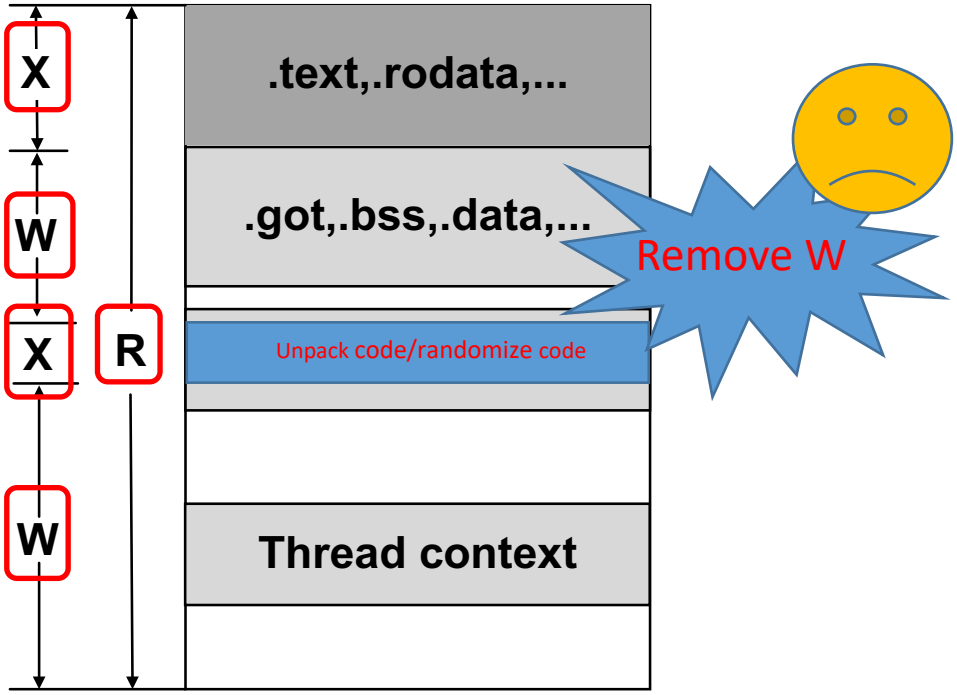
Elastic Cross-Region Bound Check(CRBC)



3 regions → 5 regions

4 MPX registers are not enough

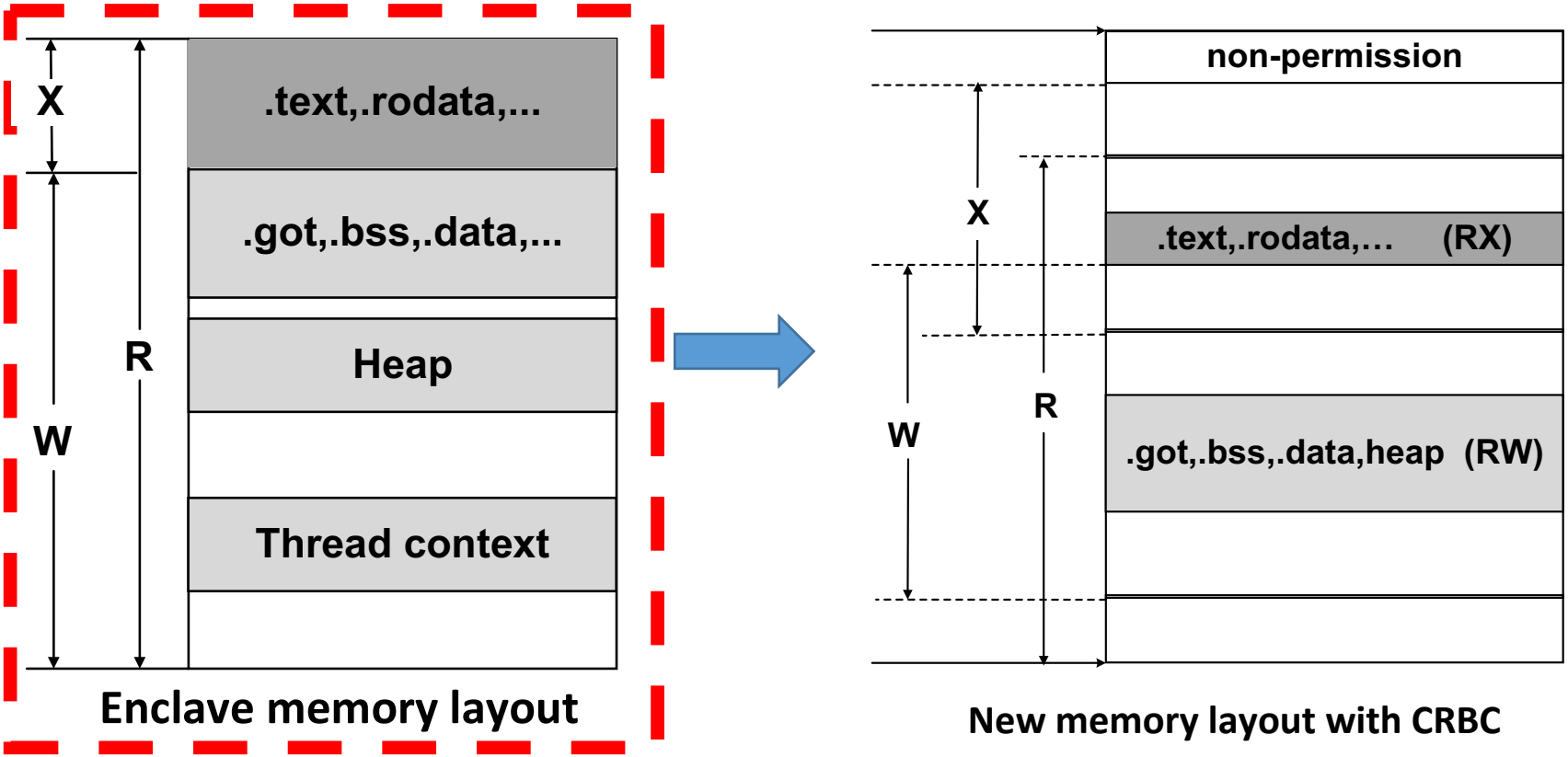
Elastic Cross-Region Bound Check(CRBC)



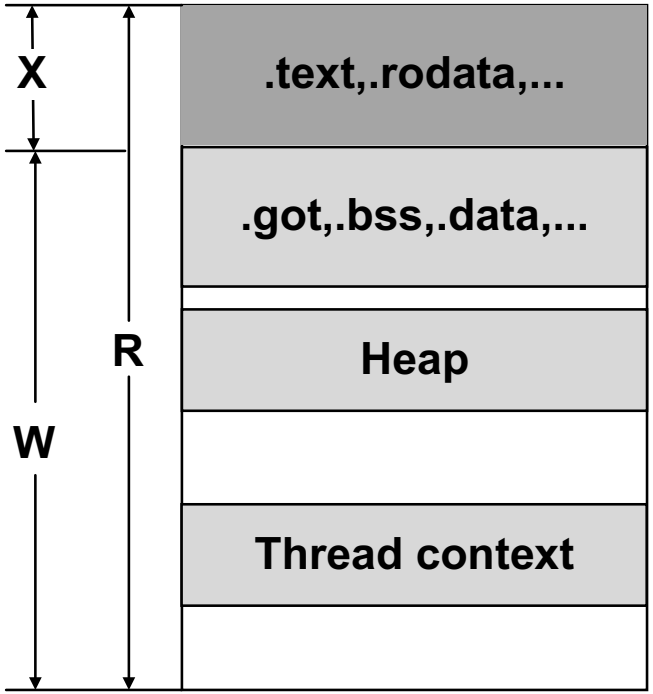
Enclave memory layout

We design a new layout

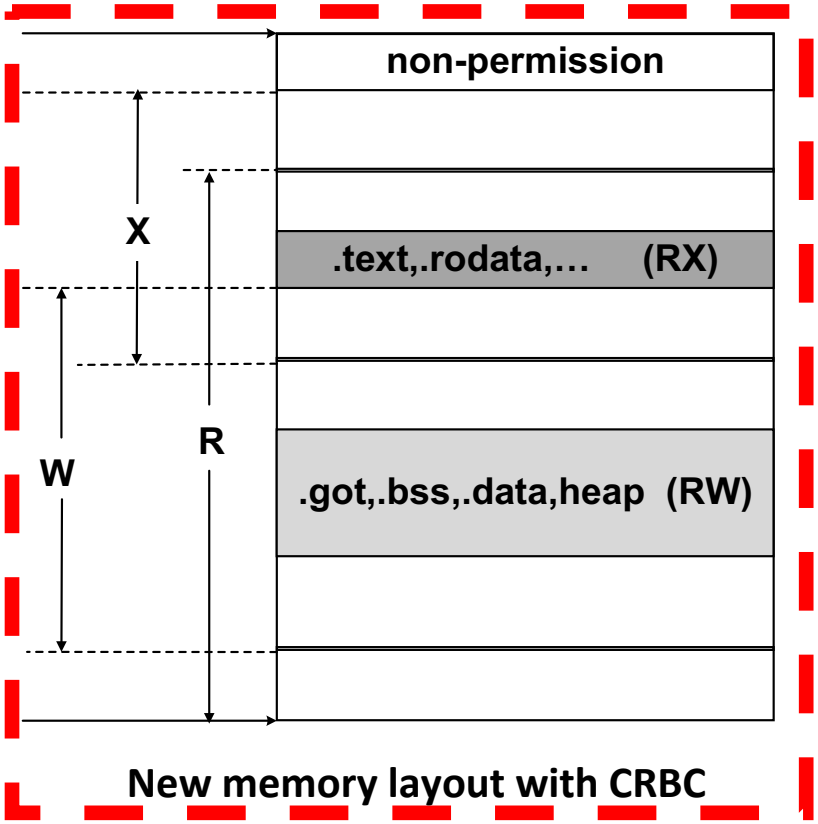
Elastic Cross-Region Bound Check(CRBC)



Elastic Cross-Region Bound Check(CRBC)

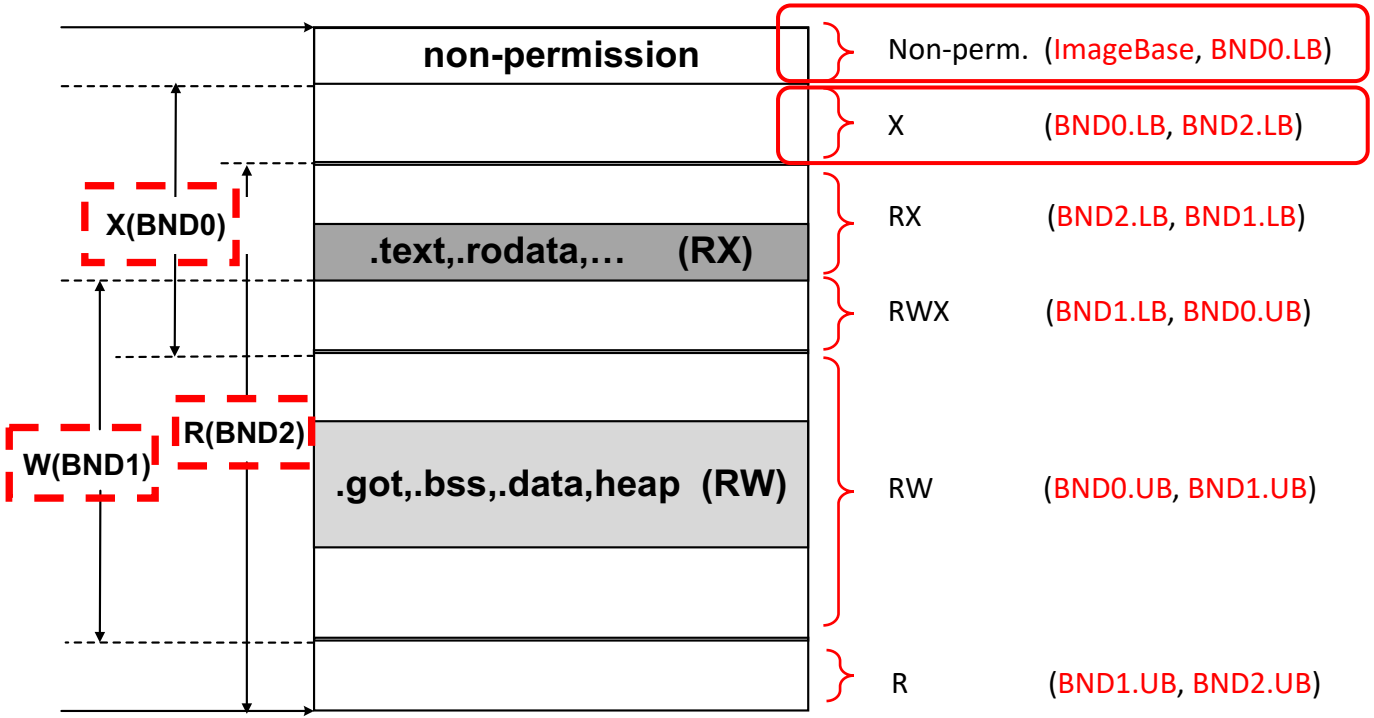


Enclave memory layout



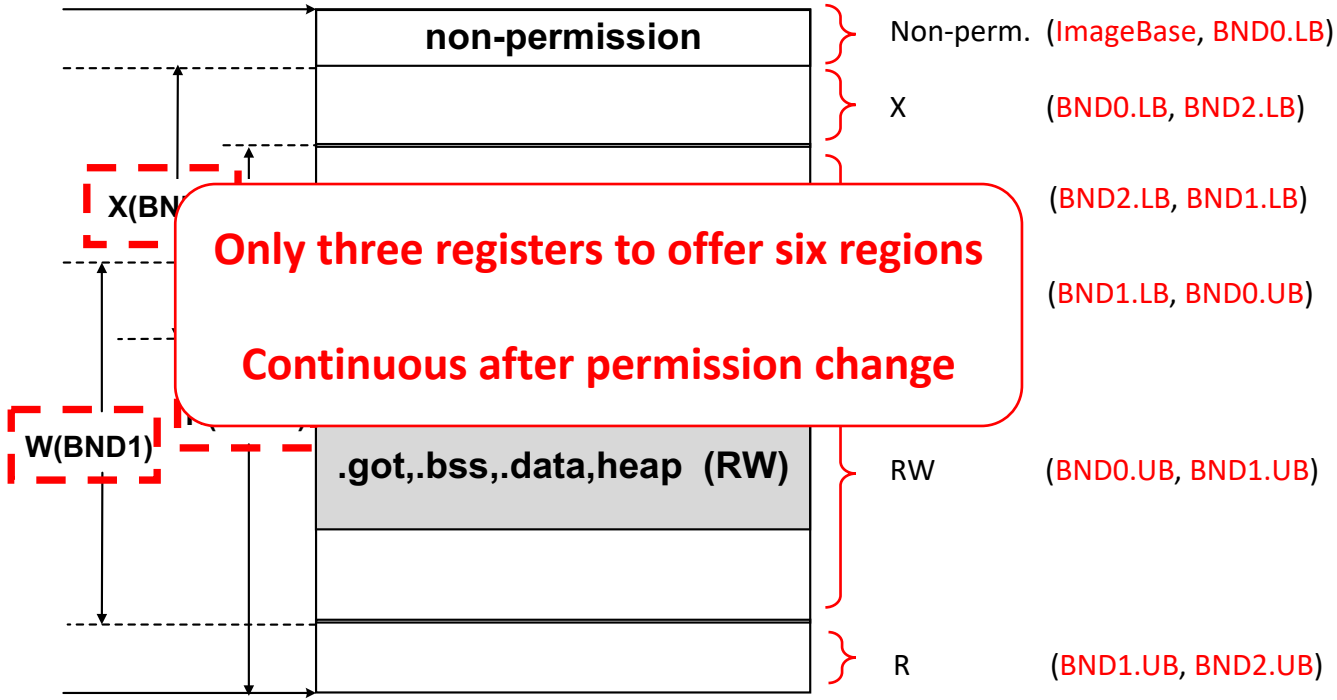
New memory layout with CRBC

Elastic Cross-Region Bound Check(CRBC)



New memory layout with CRBC

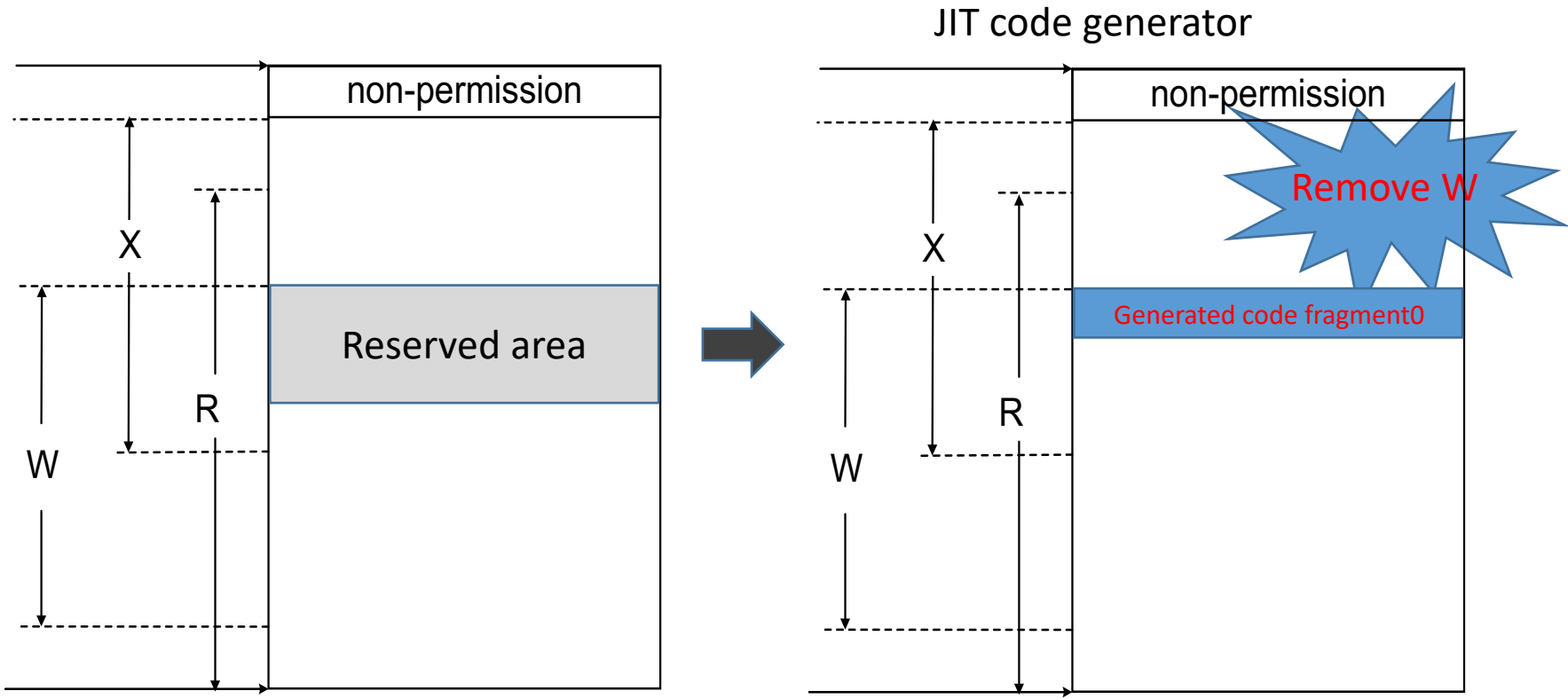
Elastic Cross-Region Bound Check(CRBC)



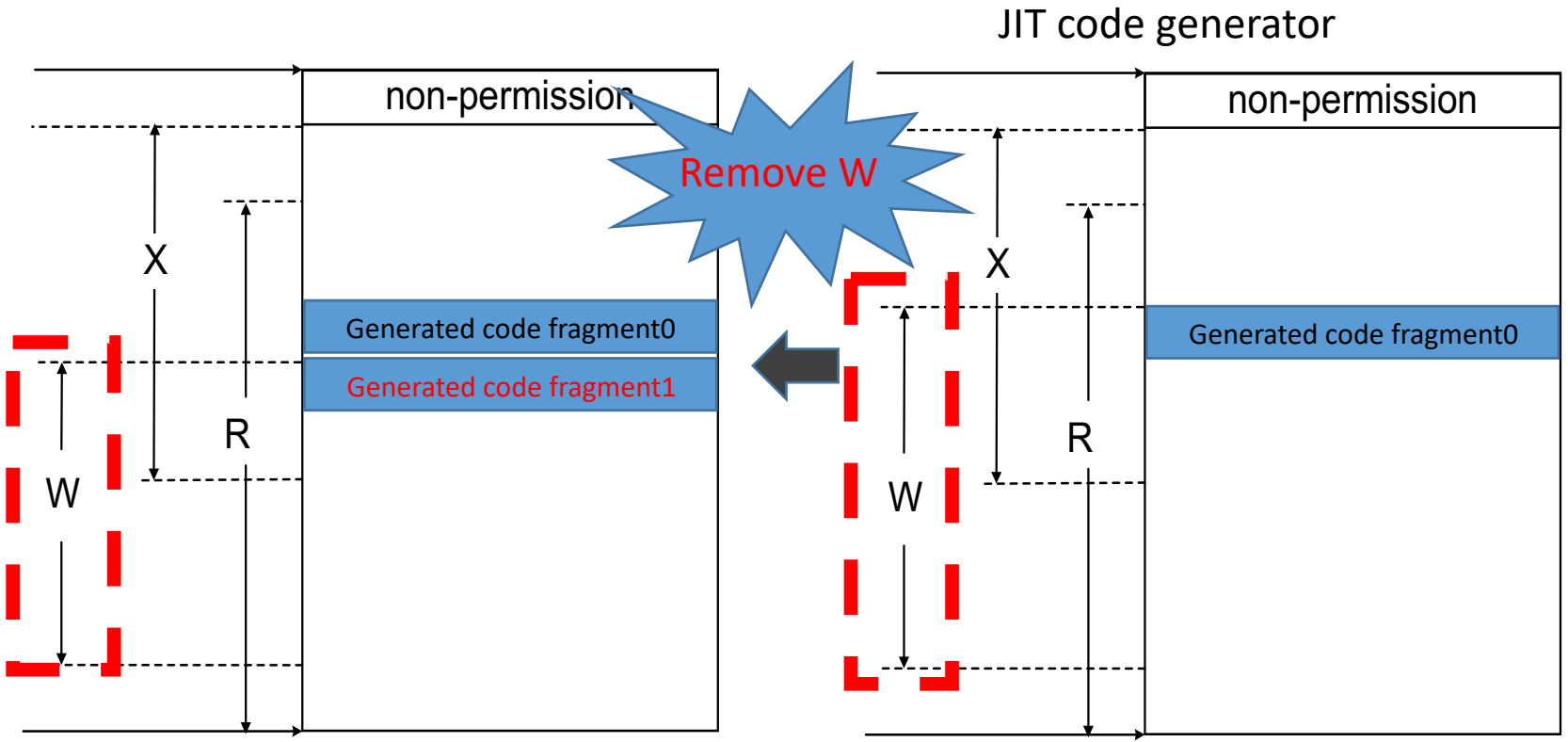
Permission	Bound range
non-perm.	(ImageBase, bnd0.lb)
X	(bnd0.lb, bnd2.lb)
RX	(bnd2.lb, bnd1.lb)
RWX	(bnd1.lb, bnd0.ub)
RW	(bnd0.ub, bnd1.ub)
R	(bnd1.ub, bnd2.ub)

New memory layout with CRBC

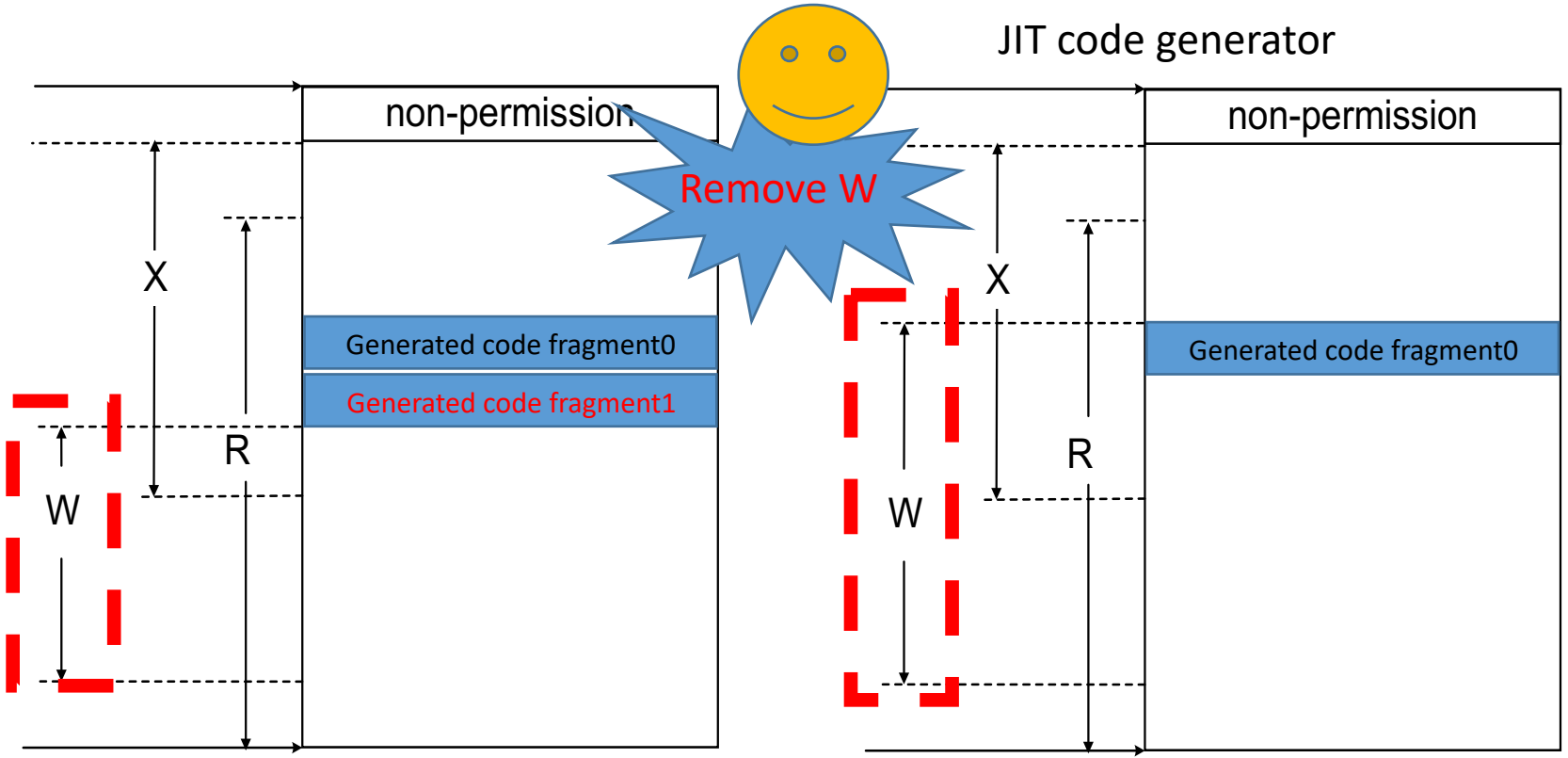
Elastic Cross-Region Bound Check(CRBC)



Elastic Cross-Region Bound Check(CRBC)



Elastic Cross-Region Bound Check(CRBC)



Elastic Cross-Region Bound Check(CRBC)

- Initializing the bounds
- Updating the bounds
- Permission enforcement using CRBC
  - Four APIs, *mpt_mmap, mpt_mremap, mpt_uunmap, mpt_write*
- Improving EPC usage
- Optimizing CRBC: Adaptive Permission Enforcement

More details in the paper

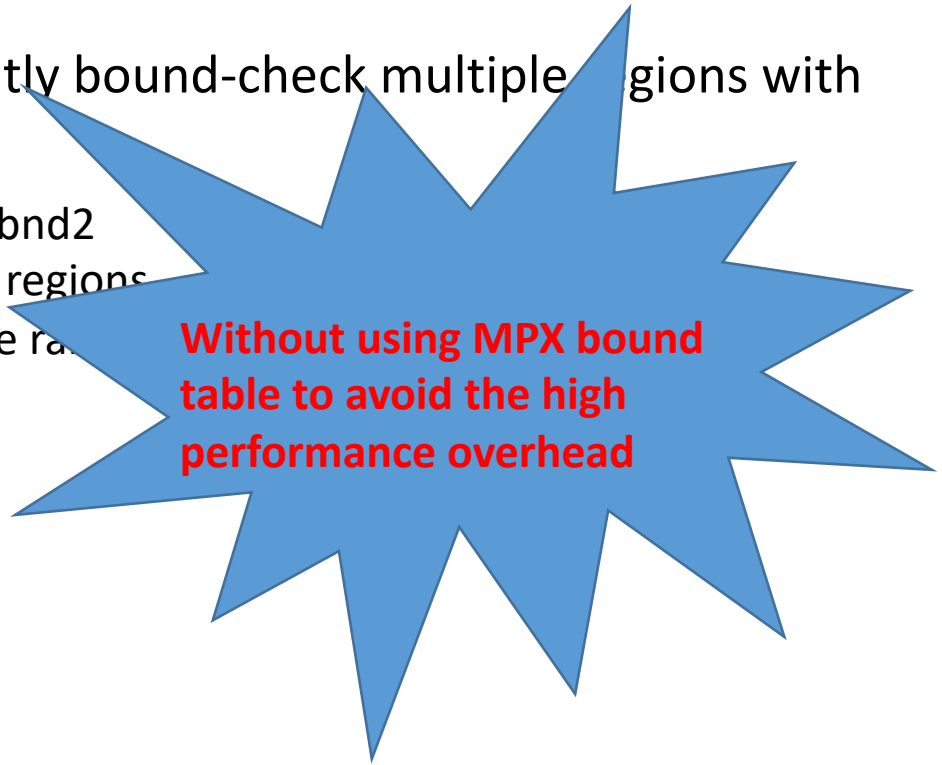
Elastic Cross-Region Bound Check(CRBC)

- CRBC leverages MPX to efficiently bound-check multiple regions with different boundary registers.
  - Use **only regs**, bnd0, bnd1, and bnd2
  - Provide **six** different permission regions
  - Allow the **flexible changes** of the ranges of memory regions at runtime

Permission	Bound range
non-perm.	(ImageBase, bnd0.lb)
X	(bnd0.lb, bnd2.lb)
RX	(bnd2.lb, bnd1.lb)
RWX	(bnd1.lb, bnd0.ub)
RW	(bnd0.ub, bnd1.ub)
R	(bnd1.ub, bnd2.ub)

Elastic Cross-Region Bound Check(CRBC)

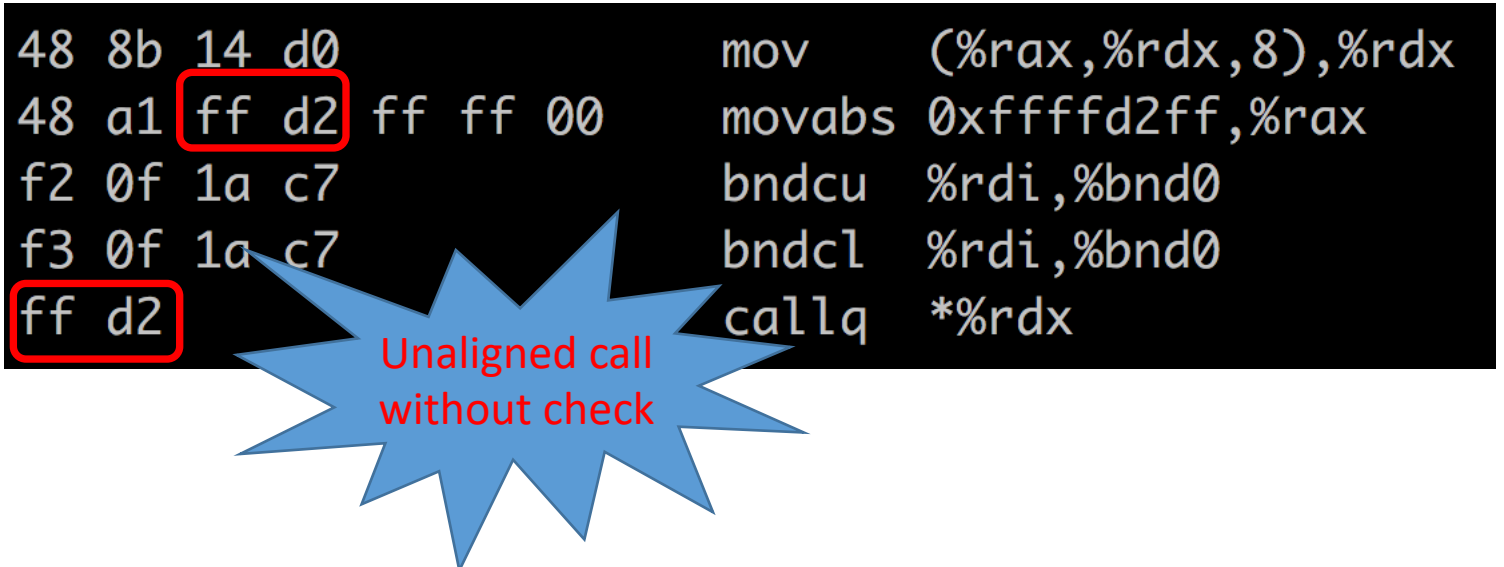
- CRBC leverages MPX to efficiently bound-check multiple gions with different boundary registers
  - Use **only regs**, bnd0, bnd1, and bnd2
  - Provide **six** different permission regions
  - Allow the **flexible changes** of the ra
    memory regions at runtime



CRBC may be attacked

Check-skipping attacks

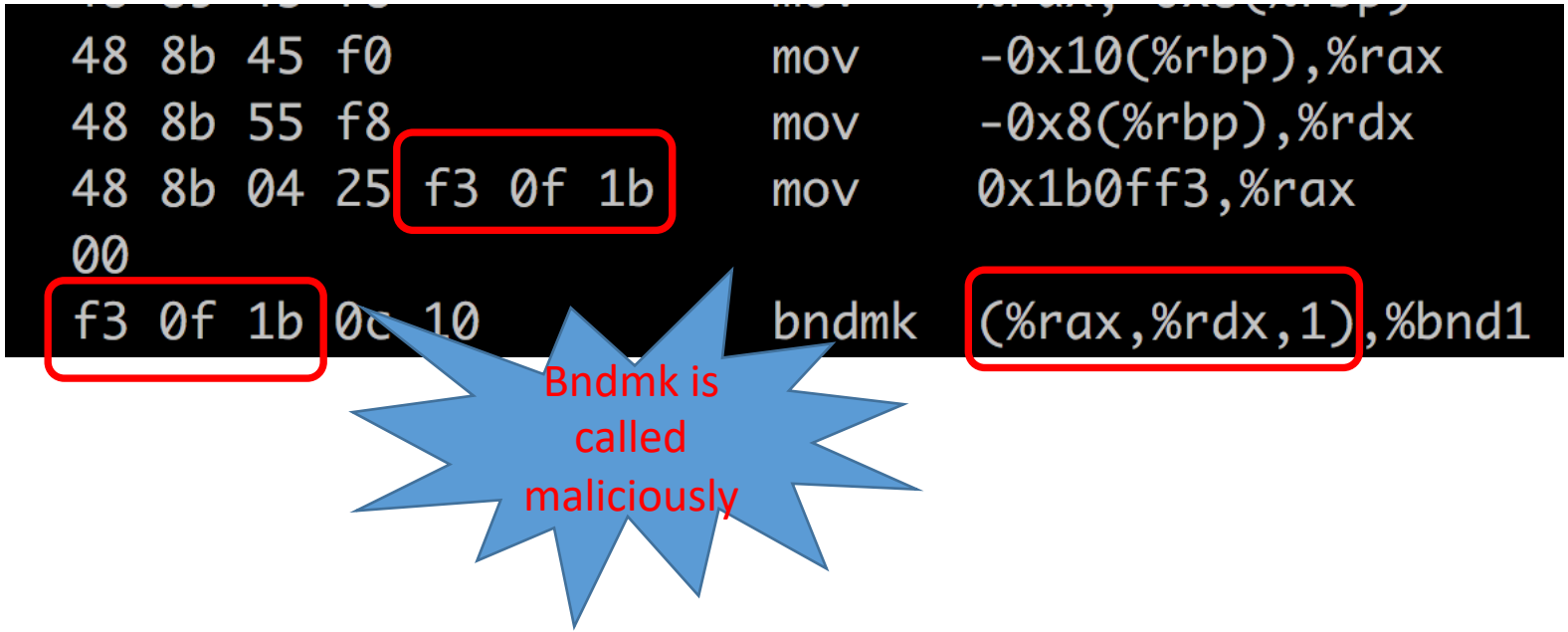
- Control-flow attacks that bypass the bound checks and abuse the permission control



CRBC may be attacked

Bound-manipulating attacks

- Data-flow attacks that manipulate bounds



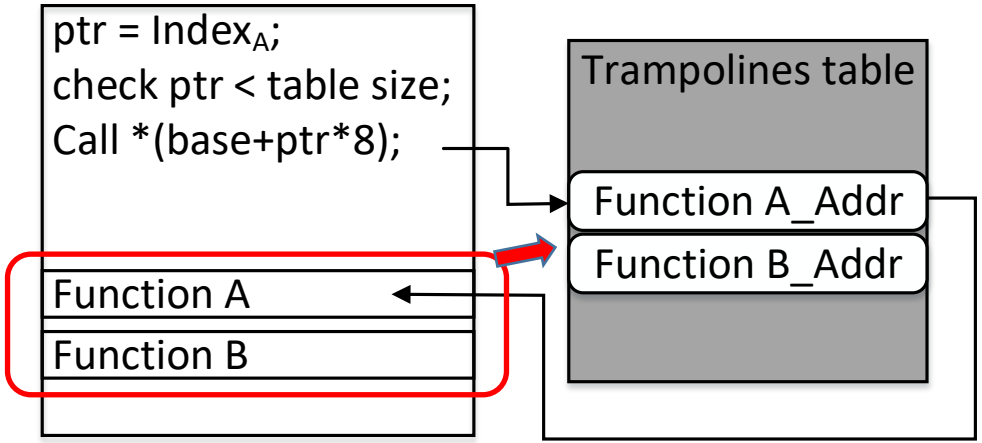
Enforcement Integrity

control-data integrity + memory isolation

Enforcement Integrity

Control-data integrity

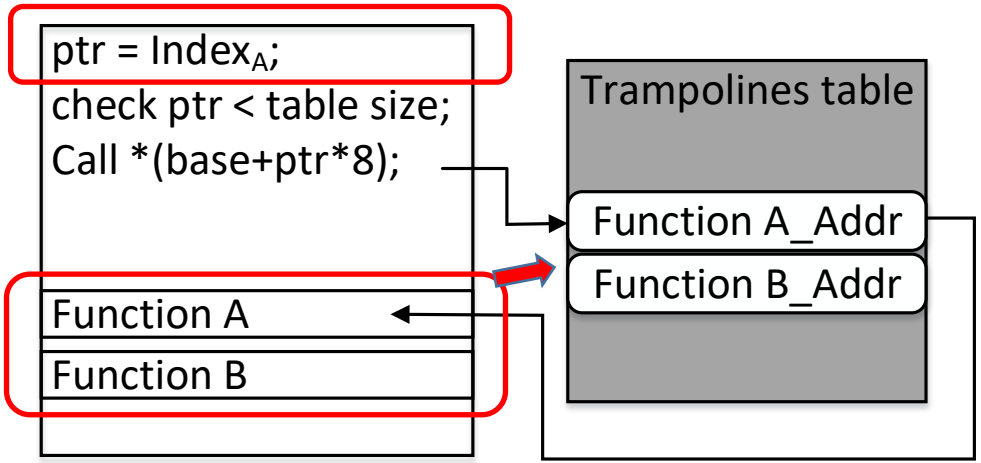
- Indirect calls/jumps



Enforcement Integrity

Control-data integrity

- Indirect calls/jumps



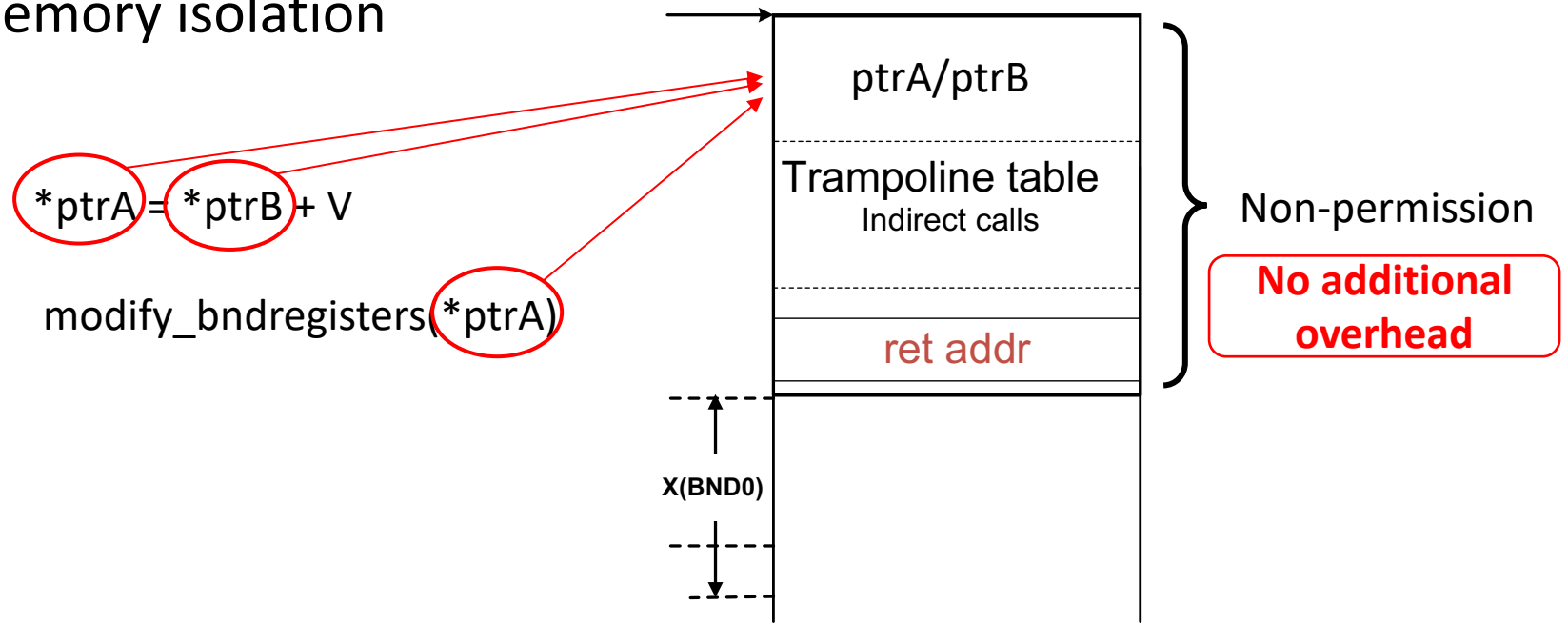
Enforcement Integrity

Control-data integrity

- Indirect calls/jumps
- Return Address
  - SafeStack [OSDI'14]

Enforcement Integrity

Memory isolation



Evaluation

Hardware platform

- Intel Xeon E3-1225v5
- 8GB memory

Software environments

- Ubuntu 16.04 Server
- SGX SDK v2.0
- SGX driver v0.10
- LLVM v6.0

Macro-benchmark

SQLite

- Overhead from 2% to 8%
- Average overhead 6.6%

Memcached

- Average overhead 2.2%

Micro-benchmark and Case Studies

- Nbench
- Protecting SGXELIDE code
- Protecting SGX-Shield code

More details in the paper

Conclusions

MPTEE provides a flexible, isolated, and efficient memory permission protection mechanism

- Three bound registers offer six permission regions
- Efficient enforcement integrity
  - Control data integrity with efficient trampoline table
  - No additional overhead of isolation beyond the CRBC

Flexible and efficient memory permission protection for SGX