MPTEE: Bringing Flexible and Efficient Memory Protection to Intel SGX

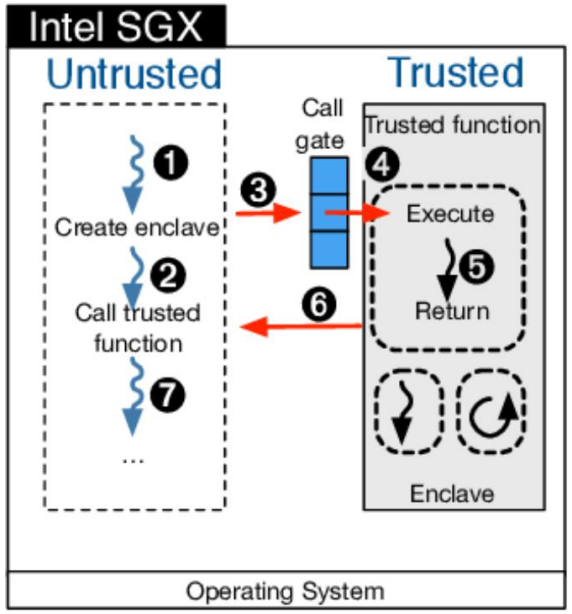
Wenjia Zhao Xi‘an Jiaotong University University of Minnesota
Kangjie Lu University of Minnesota
Yong Qi Xi’an Jiaotong University
Saiyu Qi Xidian University

slide made by wgl

Background

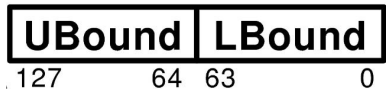
➢ Inter SGX

- Software Guard Extension
- Enclave Cache Page



➢ Intel MPX

- Memory Protection Extensions
- Four bound registers (BND0 ~ BND3)

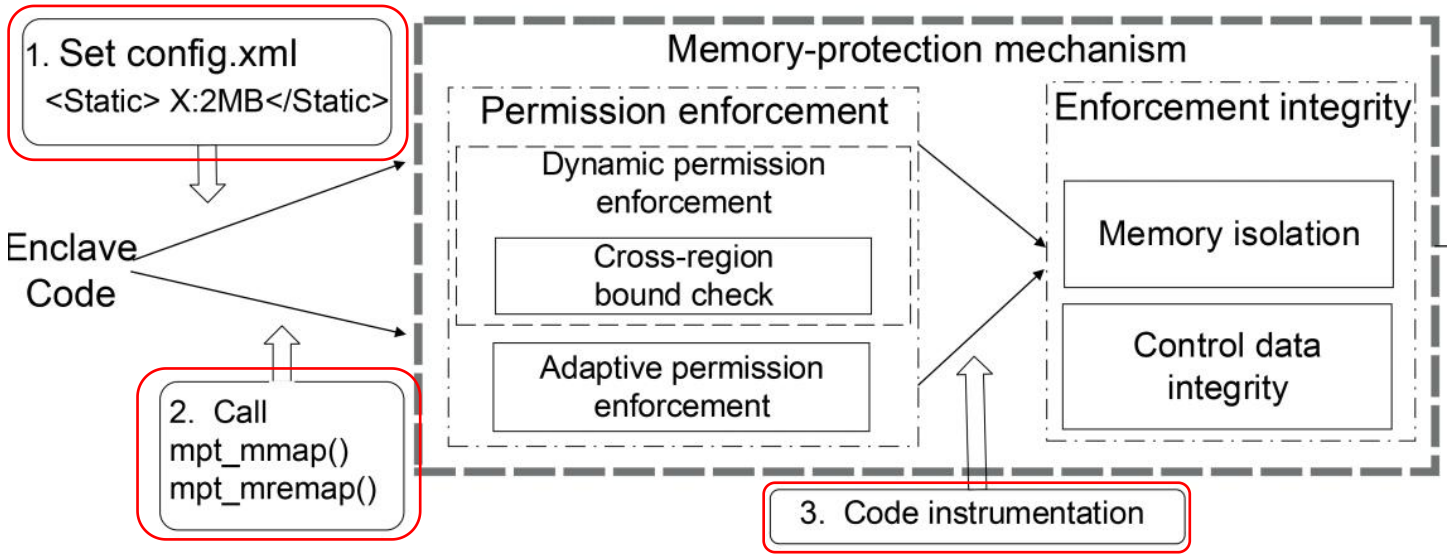


Motivation

- **Need of Flexible memory protection**
  - Dynamically update memory-page permissions
  - Least-privilege principle & Security
- **Existing System**
  - SGX: SECS & Signature-Verifcation mechanism
  - SGX-Shield: R15 Register & SGXCrypter: OS page table
  - MPX: Just four registers
- **Challenge**
  - Limited hardware support
  - Strong adversary

Architecture

➢ MPTEE



➢ Permission Enforcement
- Dynamic
- Adaptive

➢ Enforcement Integrity (EI)
- Memory isolation
- Control-data integrity (CDI)

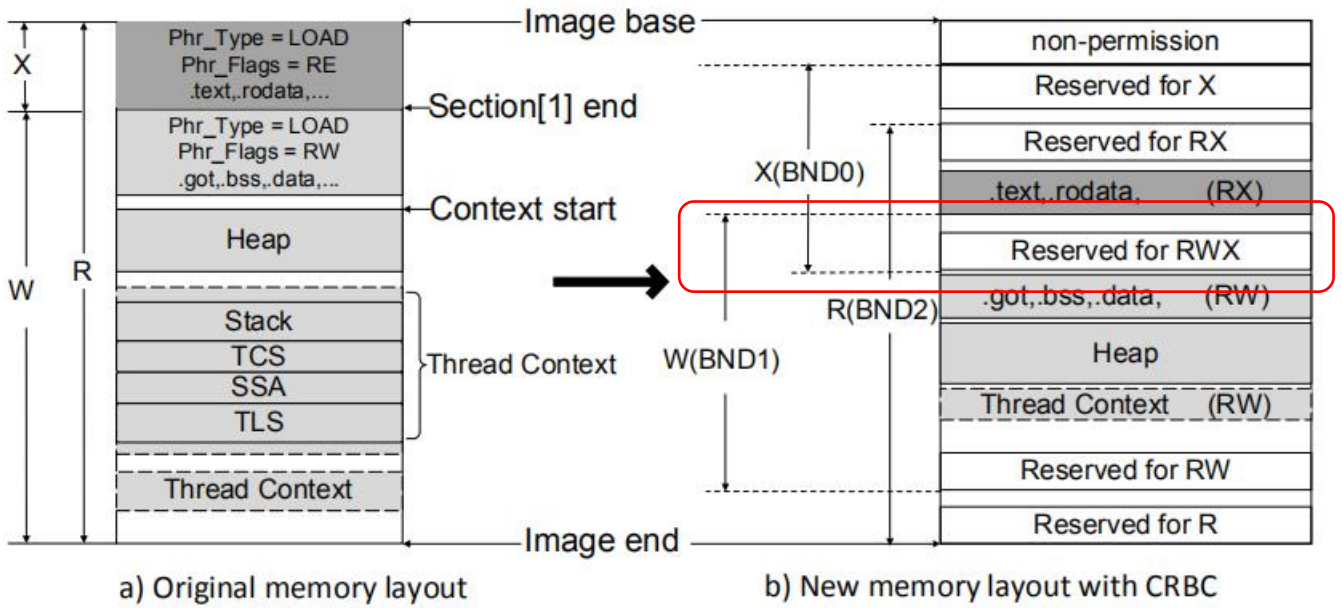
A memory-protection mechanism
Flexible and Efficient enforcement of memory-page permissions in SGX.

Permission Enforcement

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)

Overlapping

➢ Cross-Region Bound Check*



a) Original memory layout

b) New memory layout with CRBC

➢ Unique memory layout of enclaves

- Memory sections that have same permissions are adjacent

➢ Cross-region bounds

❶ BND0 ~ BND2 (Three Registers)

❷ X/R/RX/RW/RWX/No (Six permissions)

❸ RWX - RW: bnd0.ub decrease

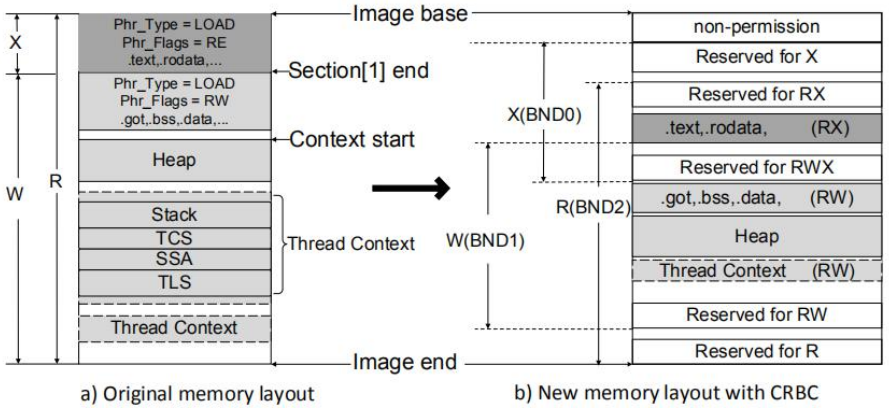
Permission Enforcement

➢ Cross-Region Bound Check*

API	Description
void *mpt_mmap(size, flags)	Acquires a memory buffer, which is at least *size* bytes. It is restricted as specified in *flags*.
void *mpt_mremap(pages, size, flags)	Changes *old_pages* to *flags*. If the *pages* are located at a region boundary, it changes the permissions; otherwise, it allocates new memory buffer and copy content over.
void *mpt_munmap(pages)	Frees an acquired memory buffer, the freed memory will be sealed and swapped out.
void *mpt_write(pages, size, content)	Writes *content* to the mapped region. The function will not be bound-checked.

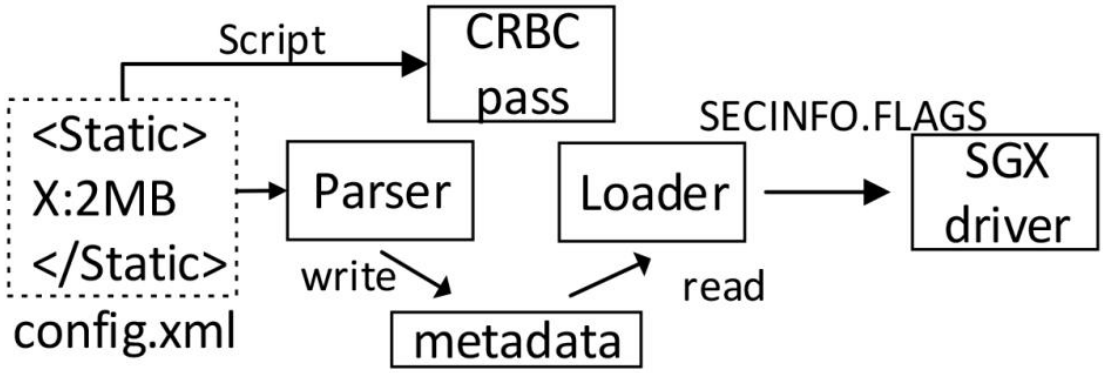
➢ Four APIs

- mpt_mmap() for accquire memory
- mpt_mremap() for change permission



Permission Enforcement

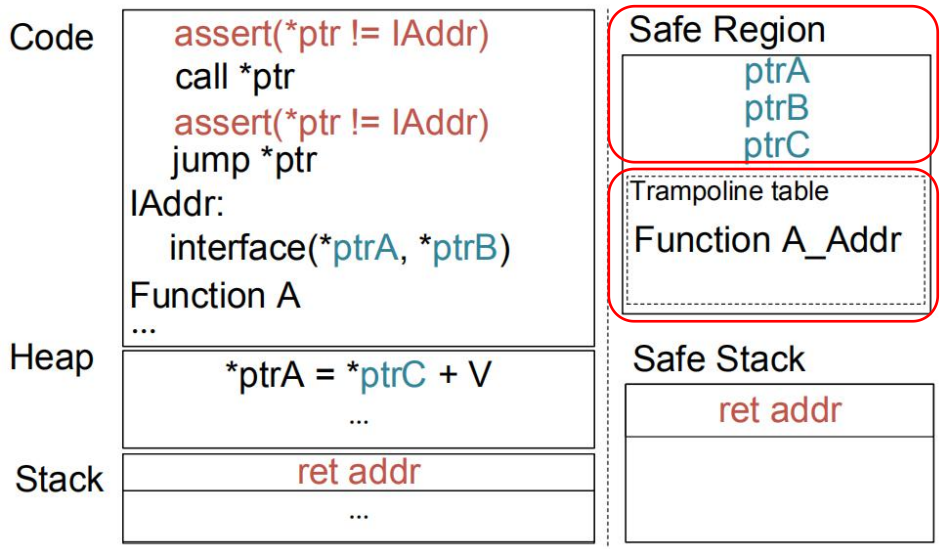
➢ Adaptive Permission Enforcement



- ➢Setting the static region in config.xml
- ➢<Permission>:<Region size>
- ➢CRBC pass: Reduce the performance overhead

Enforcement Integrity

➢Memory Isolation



➢Safe Region

- Protect code and data for permission enforcement
- No-permission region

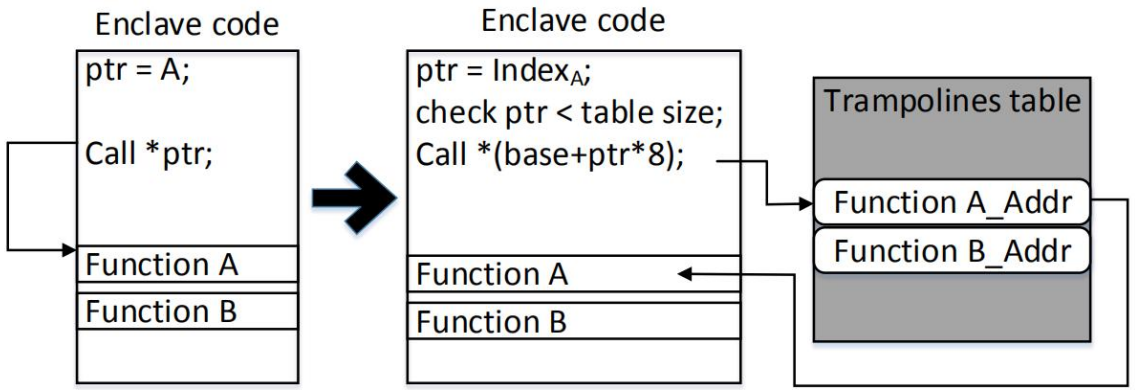
➢Safe Stack & Trampoline table

- Ensure the integrity of return addresses and function pointers

➢Safe Region & Trampoline table

Enforcement Integrity

➢Control-Data Integrity



➢Trampoline table

- protect pointers for indirect calls and jumps

➢Static analysis: All functions whose addresses are ever taken

Evaluation

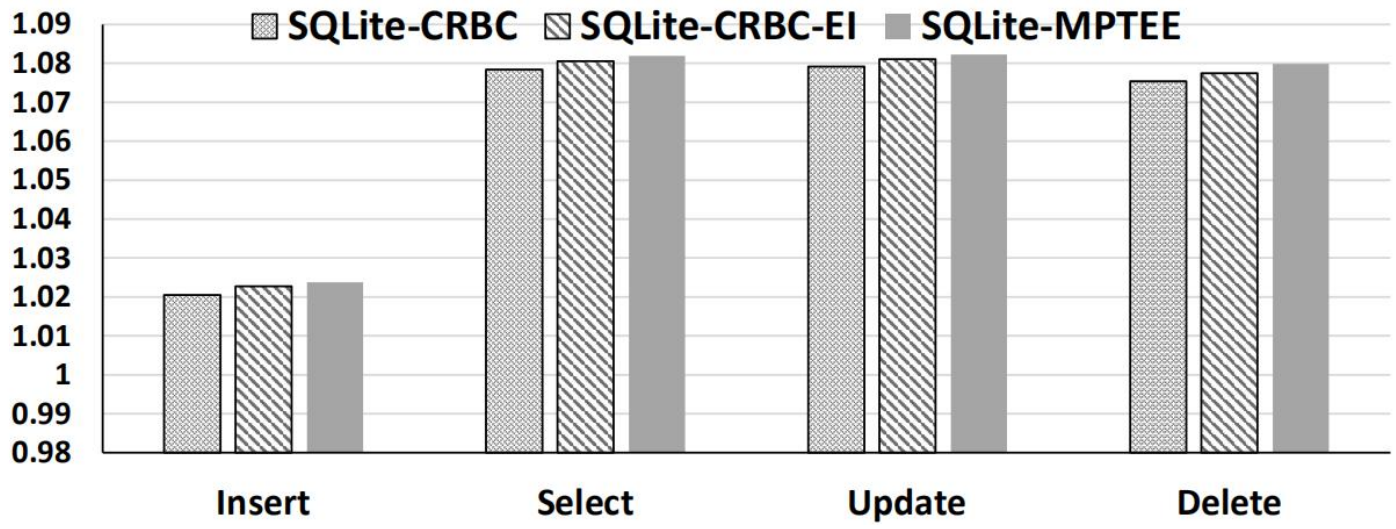
- **Experiments**
  - Dell workstation
  - Intel Xeon E3-1225v5 CPU & 8G RAM
  - Ubuntu 16.04 Server

- **Evaluation**
  - Performance
  - Utility Analysis
  - Security Analysis

Evaluation

SQLite is a C-language library
Implements small and fast SQL database engine.

➢Performance



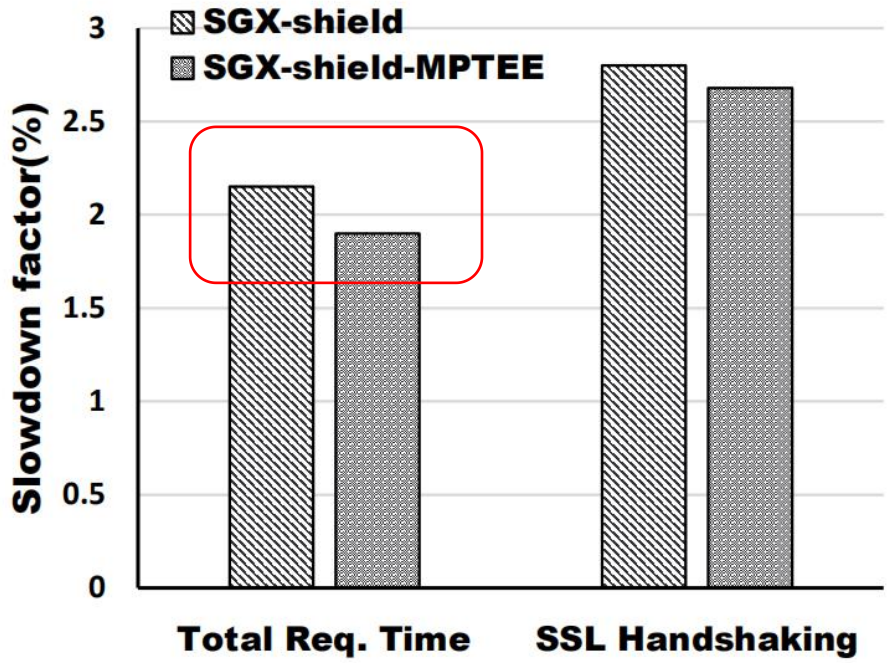
Low overhead imposed by MPTEE

- ➢CRBC imposes 2%–7% (avg 6.6%) overhead
- ➢EI imposes smaller than 1%
- ➢More frequent memory accesses leading to more bound checks.

Evaluation

Requesting HTML fles from the HTTPS server

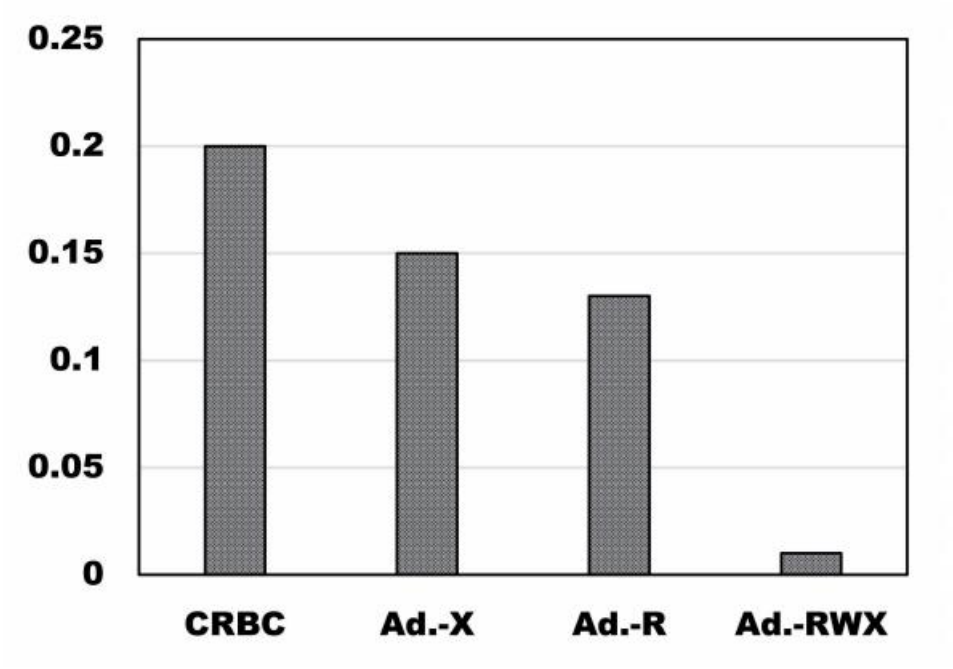
➢Performance



- ➢SGX-Shield-MPTEE has lower overhead 1.9%
  (with no adaptive permission)
- ➢SGX-Shield is 2.2% & Only provide one permission

Evaluation

- **Performance**



Benefts from adaptive permission enforcement

- CRBC incur up 20% overhead
- Adaptive-RWX incurs almost 0% (hardware)
- Bound-checking of R is more frequent than X

Evaluation

➢Utility Analysis

- Confrm MPTEE is easy-to-use and practical
- SGX-Shield NRW boundary
  - mpt_mmap(sgx_end-sgx_start, X)
- Protecting SGX-Shield code

Evaluation

➢Threat Model

- Trusted: only Intel SGX itself
- Untrusted: All other software and hardware components
- Vulnerabilities of code running inside SGX: buffer overfows

➢Security Analysis

- Check-skipping attacks: CDI
- Bound-manipulating attacks: Memory Isolation

Conclusion

- **MPTEE**
  - Cross-Region Bound Check
  - Adaptive Permission Enforcement
  - Memory Isolation & Control-data Integrity
- **Evaluation**
  - Performance
  - Utility & Effectiveness.

MPTEE can achieve a flexible and secure memory protection while just imposing a small overhead

Comments

➢ Advantages

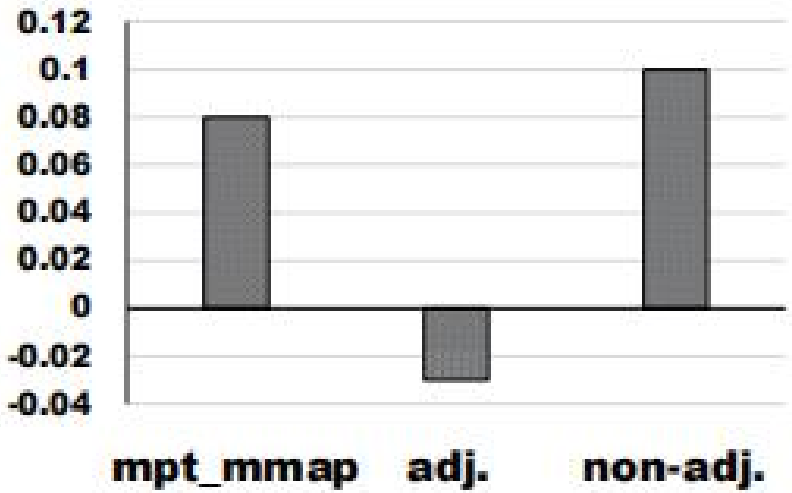
- The MPTEE actually develop a flexible and secure memory-protection mechanism with less hardware support and just impose a small overhead.

➢ Disadvantages

- Intel MPX has been eliminated by Microsoft.
- Argues:
  - As the hardware component, SGX cannot be easily upgraded. Old version SGX will still be the dominating version for a while.
  - For enforcement integrity, MPTEE's design is compatible with new SGX.
- Maintenance burdens

Append

- mpt_mremap()




```
   Input  : addr, size, flags
   Output : new_addr
   Data: Metadata of 5 Reserved region, 3 bnd register
1  Get cur_region, flags_region, neighbor_region from addr and registers
2  if cur_region == flags_region then
3      new_addr = realloc(addr, size)
4      if realloc failed then
5          if neighbor_region is not reserved || neighbor_region has
             allocated then
6              new_add = NULL
7          else
8              Get bndX according to flags_region
9              bndX.lb = bndX.lb + size
10             update metadata of cur_region
11         new_addr = realloc(addr, size)
12 else if cur_region is neighbor with flags_region then
13     Get bndX according to flags_region
14     bndX.lb = bndX.lb - size
15     new_addr = addr
16 else
17     Get the dest_heap according to flags
18     new_addr = malloc in dest_heap
19     mpt_write(new_addr, size, addr)
20     mpt_munmap(addr)
21 return new_addr
```