What you don't know won't hurt you

- Address space isolation is one of the best protection methods since the invention of the virtual memory.
- Vulnerabilities are inevitable, how can we minimize the damage
- Make parts of the Linux kernel use a restricted address space for better security
Related work

- **Page Table Isolation**
  - Restricted context for kernel-mode code on entry boundary

- **WIP: improve mitigation for HyperThreading leaks**
  - **KVM address space isolation**
    - Restricted context for KVM VMExit handlers
  - **Process local memory**
    - Kernel memory visible only in the context of a specific process
System Call Isolation (SCI)

- Execute system calls in a dedicated address space
  - System calls run with very limited page tables
  - Accesses to most of the kernel code and data cause page faults

- Ability to inspect and verify memory accesses
  - For code: only allow calls and jumps to known symbols to prevent ROP attacks
  - For data: TBD?

- Weakness
  - Cannot verify RET targets
  - Performance degradation
  - Page granularity

https://lore.kernel.org/lkml/1556228754-12996-1-git-send-email-rppt@linux.ibm.com/
mmap(MAP_SECRET)

- Memory region in a process is isolated from the rest of the system
- Can be used to store secrets in memory:
  
  ```c
  void *addr = mmap(MAP_SECRET, ...);
  struct iovec iov = {
    .base = addr,
    .len = PAGE_SIZE,
  };

  fd = open_and_decrypt("/path/to/secret.file", O_RDONLY);
  readv(fd, &iov, 1);
  ```

- Assumption: ‘struct page’ metadata is sufficient for block IO
Address space for netns

- Netns is an independent network stack
  - Network devices, sockets, protocol data
- Objects inside the network namespace are private
  - Except skb’s that cross namespace boundaries
- Let’s enforce privacy with page tables
Address space for netns

- Kernel page table per namespace
  ```c
  struct net {
      pgd_t *pgd; /* namespace private page table */
      refcount_t passive; /* To decided when the network */
      /* namespace should be freed. */
  }
  ```

- Processes in a namespace share view of the kernel mappings
  - Switch page table at `clone()`, `unshare()`, `setns()` time.
  - Private kernel objects are mapped only in the namespace PGD
Suppose it works, now what?

- Makes sense for netns, what about others?
- How to handle nested namespaces?
- What userspace ABIs are needed?
  - On/off command line parameter?
  - `proc` or `sysfs` knobs?
  - Address space namespace?
- What is the actual security benefit?
Thank You
Kernel address space

User Space

System Calls

Single Address Space

Page Tables

Files

Devices

Sockets
Syscall vulnerability

The entire kernel is compromised

User Space

System Calls

Page Tables
Files
Sockets
Devices
SCI flow

- System call
- Switch address space
- Access unmapped code
- Page fault
- Switch address space
- Map the page
- Is access safe?
  - Yes: Map the page
  - No: Kill process
Netns isolation overview

User Space

- Page Tables
- Files
- System Calls
- Generic Address Space
- Restricted Address Space

NAMESPACE
- Buffers
- Devices
- Sockets

NAMESPACE
- Buffers
- Devices
- Sockets
Netns isolation - page tables

Network Namespace

Kernel mappings for namespace

User Space

PTI pair

Kernel Space