Address Space Isolation for Container Security

Mike Rapoport, James Bottomley
<{rppt,jejb}@linux.ibm.com>

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 825377

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:

```
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

```
@@ -52,6 +52,7 @@ struct bpf_prog;
 #define NETDEV_HASHENTRIES (1 << NETDEV_HASHBITS)

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

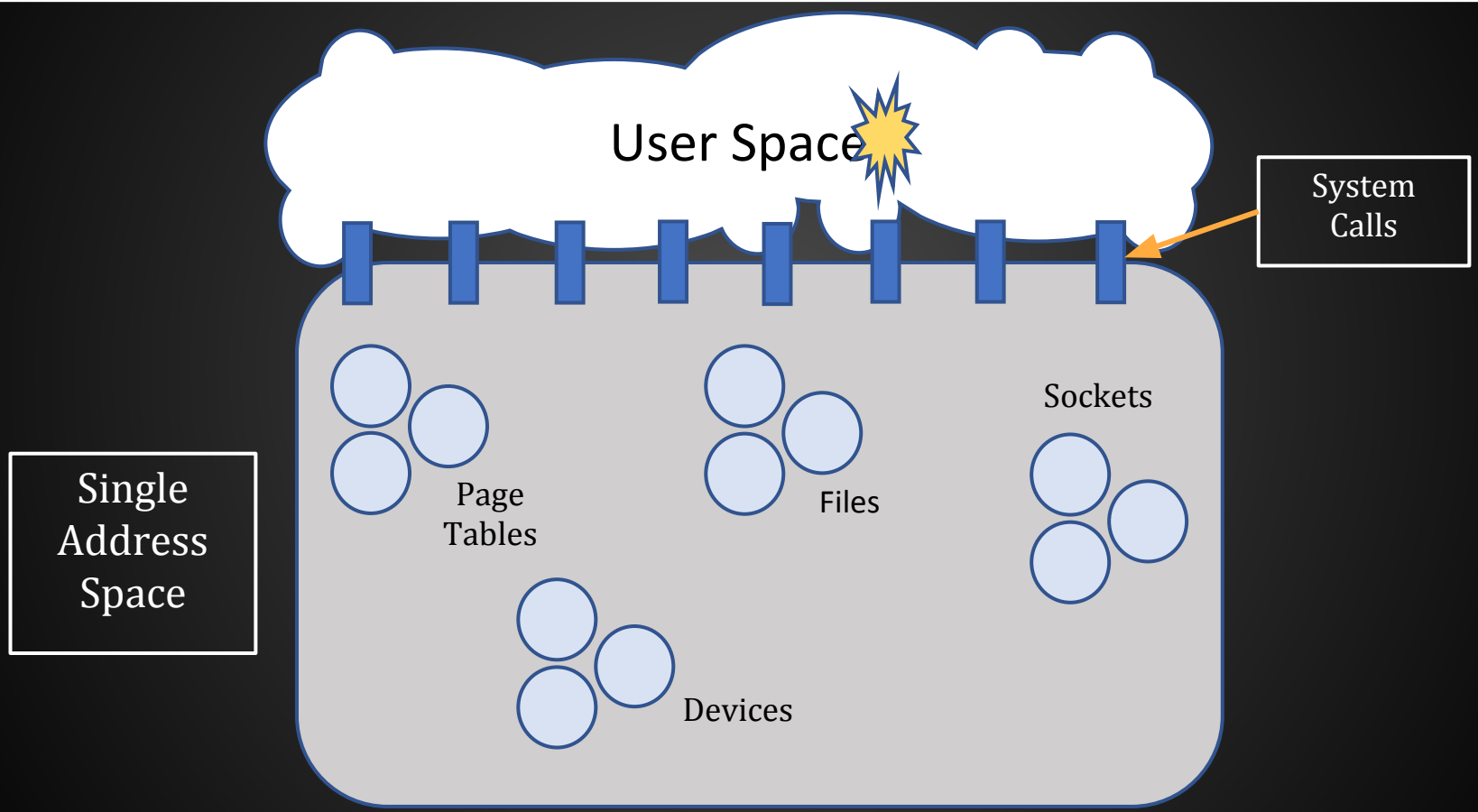
- Processes in a namespace share view of the kernel mappings
  - Switch page table at `clone()`,`unshare()`,`setns()` time.
- Private kernel objects are mapped only in the namespace PGD

Suppose it works, now what?

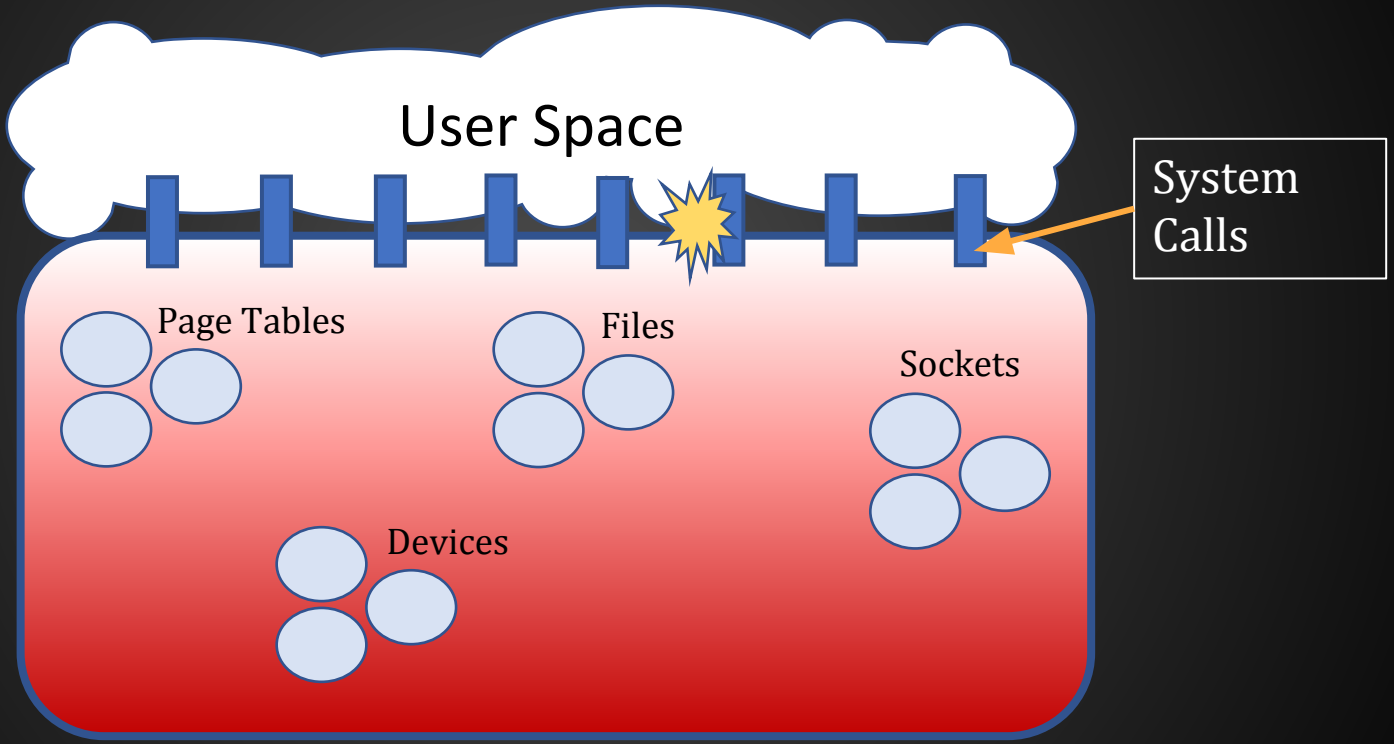
- Makes sense for netns, what about others?
- How to handle nested namespaces?
- What userspace ABIs are needed?
  - On/off command line parameter?
  - `proc` or `sysfs` knobs?
  - Address space namespace?
- What is the actual security benefit?

Thank You

Kernel address space

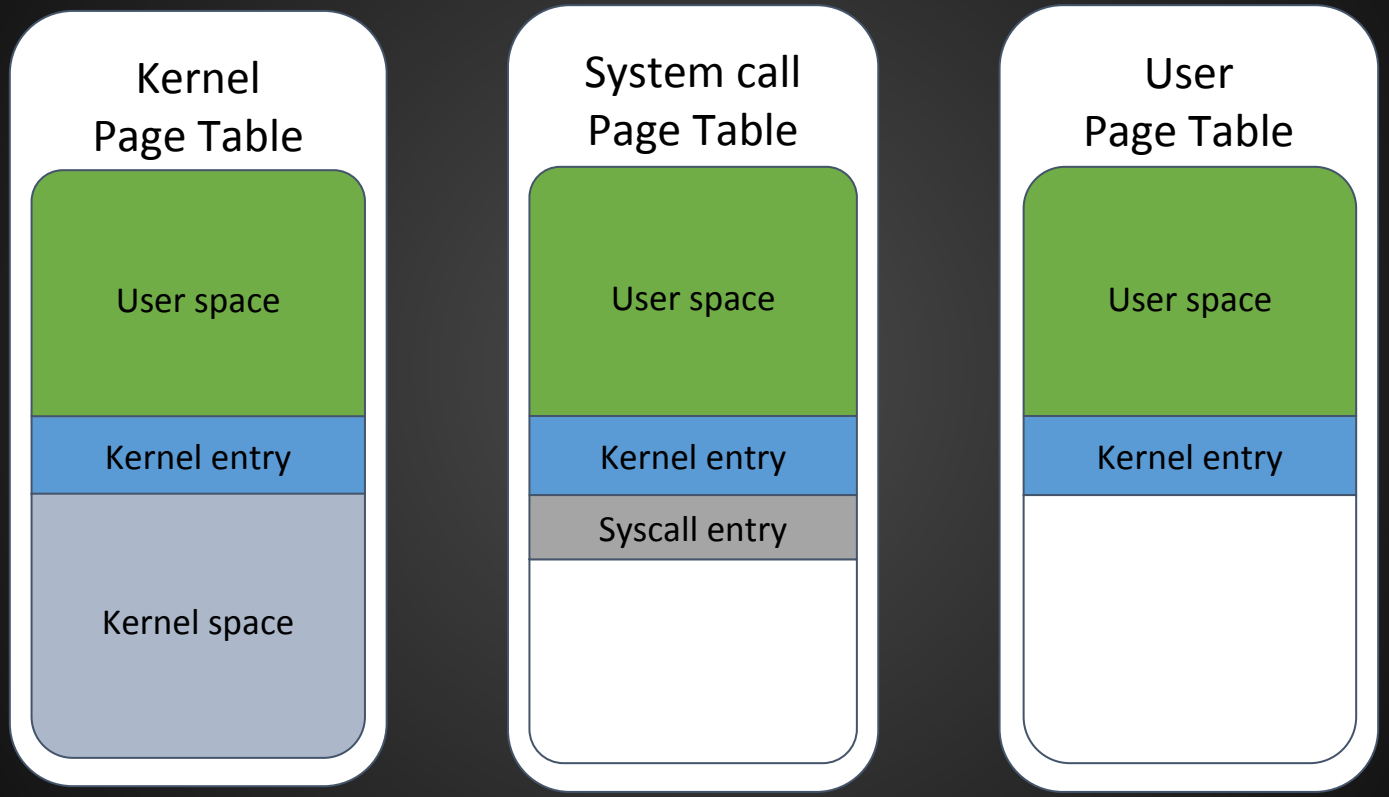


Syscall vulnerability

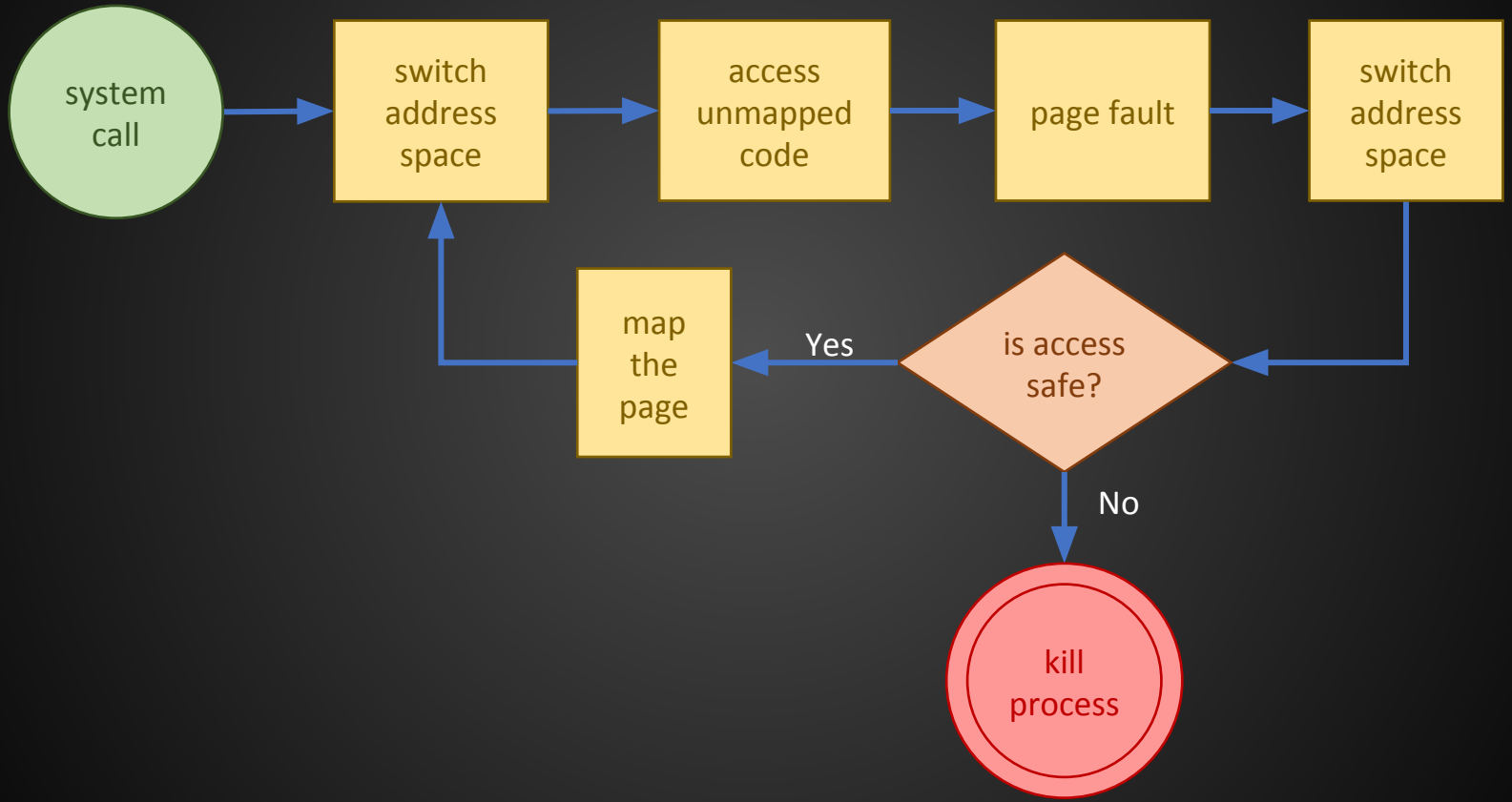


The entire kernel is compromised

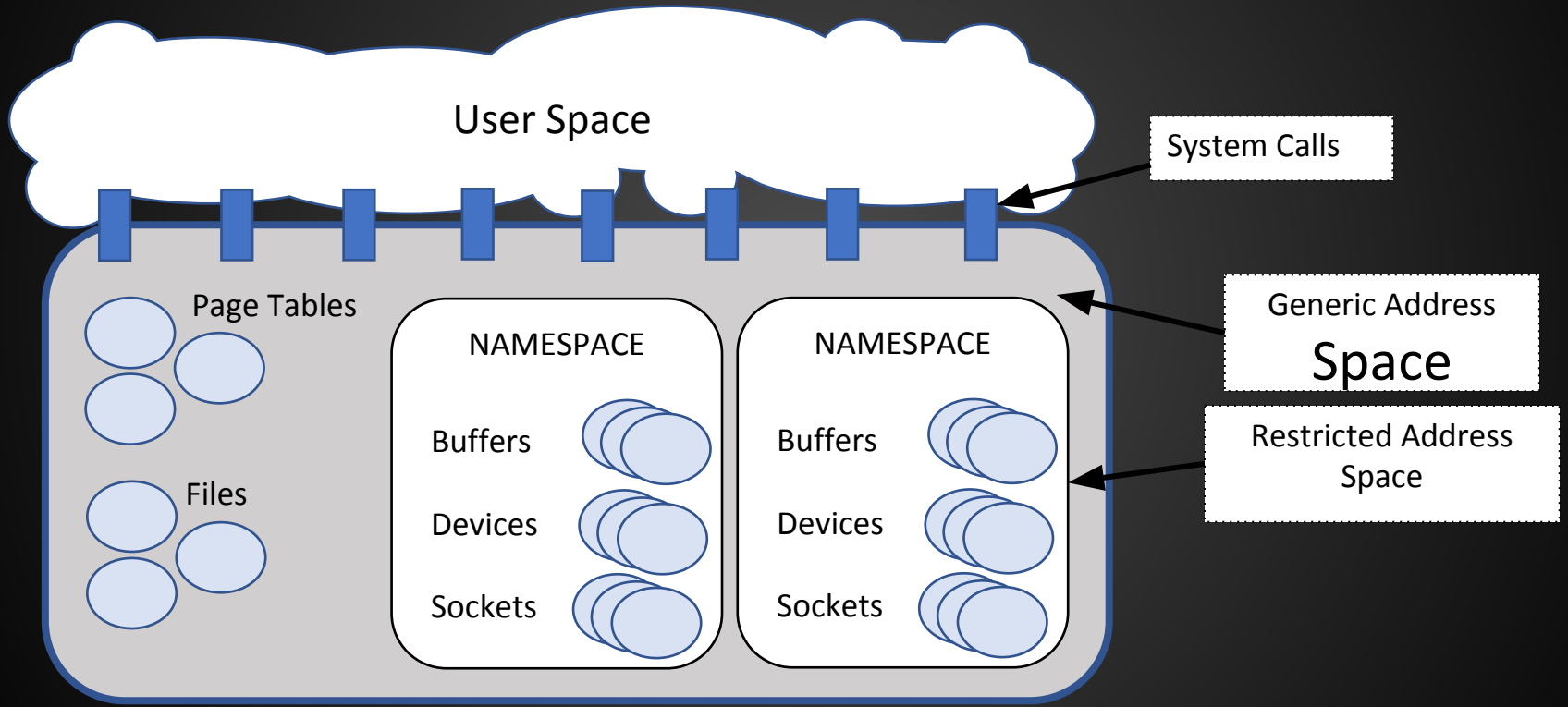
SCI page tables



SCI flow


system call
switch address space
access unmapped code
page fault
switch address space
is access safe?
Yes
map the page
No
kill process

Netns isolation overview



Netns isolation - page tables