Hatching Security:
Using LinuxKit as a Security Incubator

Tycho Andersen
tycho@docker.com

Riyaz Faizullabhoy
riyaz@docker.com

docker

LinuxKit

docker

Why LinuxKIT

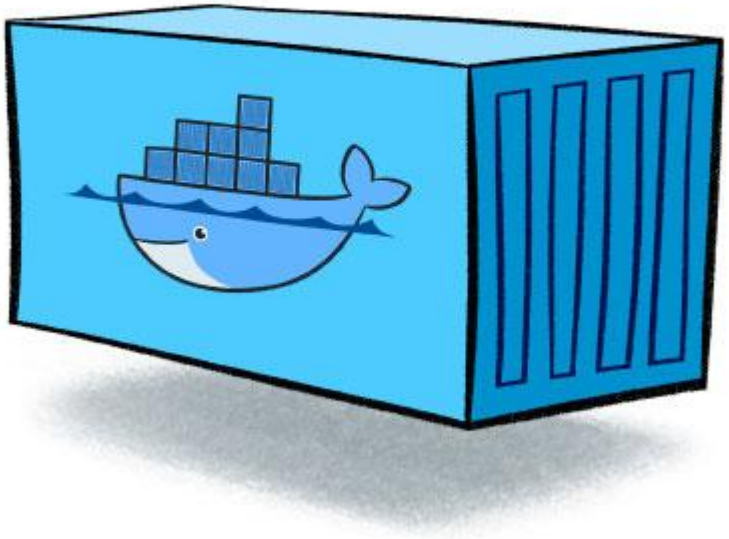
Why LinuxKIT

Google Cloud Platform

amazon web services

Why LinuxKIT

Why LinuxKIT



Why LinuxKIT

```
# echo "evil!" > /bin/sh
/bin/sh: can't create /bin/sh: Read-only file system
```

Why LinuxKIT

Why LinuxKit

Why LinuxKIT

What is LinuxKit

flexible

- batteries included, but **removable**
- built from **containers**
- **fast** to build
- **fast** to boot
- run Docker, Kubernetes or **whatever** you like

```yaml
kernel:
  image: linuxkit/kernel:4.9.47
  cmdline: "console=tty0 console=ttyS0 console=ttyAMA0"
init:
  - linuxkit/init:6d11a1f9d299d3425e78cce80dfba8b236d20412
  - linuxkit/runc:a1b564248a0d0b118c11e61db9f84ecf41dd2d2a
  - linuxkit/containerd:7d97282aebe36040fcdd3378a95562d440d34a9d
  - linuxkit/ca-certificates:e44b0a66df5a102c0e220f0066b0d904710dcb10
onboot:
  - name: sysctl
    image: linuxkit/sysctl:154913b72c6f1f33eb408609fca9963628e8c051
  - name: dhcpcd
    image: linuxkit/dhcpcd:f3f5413abb78fae9020e35bd4788fa93df4530b7
    command: ["/sbin/dhcpcd", "--nobackground", "-f", "/dhcpcd.conf", "-1"]
onshutdown:
  - name: shutdown
    image: busybox:latest
    command: ["/bin/echo", "so long and thanks for all the fish"]
services:
  - name: getty
    image: linuxkit/getty:797cb79e0a229fcd16ebf44a0da74bcec03968ec
```

What is LinuxKit

immutable

- **read-only** rootfs
- **immutable** in production
- designed to be managed by **external tooling**

What is LinuxKit

secure

- **minimal** set of packages and daemons
- **opinionated** defaults
- **cryptographically sign** all dependencies
- work with broader Linux Security **community**

6 Months of LinuxKit

2017-07-05: LinuxKit Security SIG Meeting

LINUXKIT SIG Recap:

Type Safe System Daemons

- What if all system daemons were rewritten in type-safe languages?
  - examples of DNS / HTTPS in https://github.com/linuxkit/linuxkit/tree/master/projects/miragesdk

LINUXKIT SIG Recap:

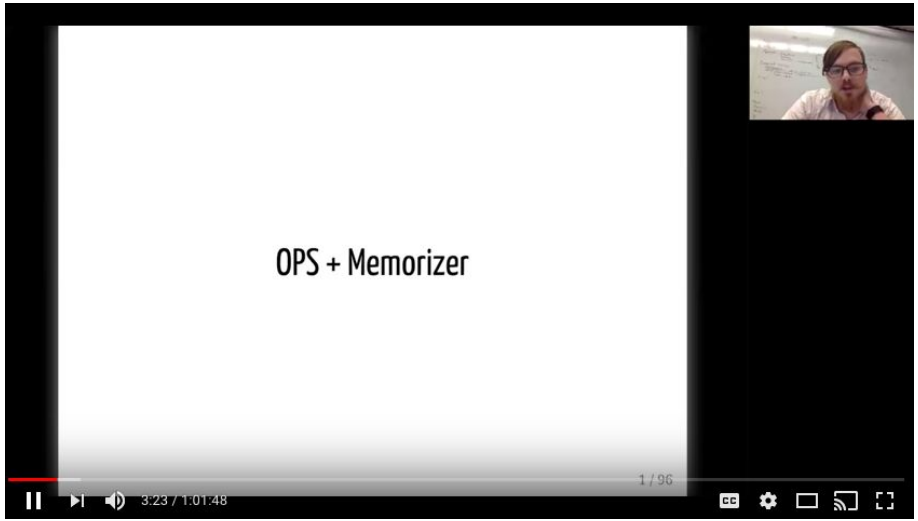
LandLock LSM

- Robust, configurable LSM rules
- Powered by eBPF
- Exciting for container landscape

LINUXKIT SIG Recap:

Memorizer

- Dynamic kernel tracing tool
  - makes use of KASAN
  - examples: https://github.com/linuxkit/linuxkit/tree/master/projects/memorizer
- Goal: produce useful output for LSMs and other higher level policy decisions



LinuxKit SIG Recap:

WireGuard

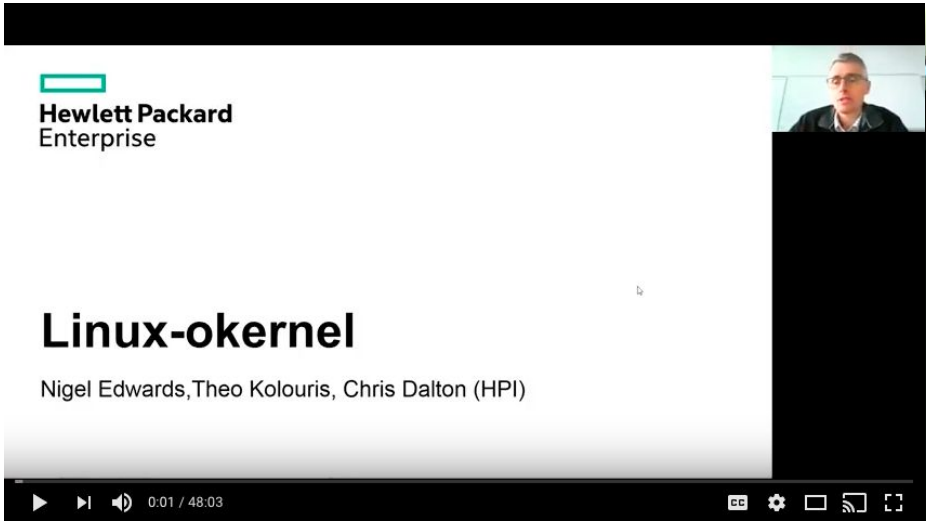
- Modern VPN implementing The Noise Protocol
  - only a few thousand lines of code!
- Now included in LinuxKit userspace and kernels

LINUXKIT SIG Recap:

HPE okernel

- Separate parts of the kernel into more and less privileged partitions

- Maps to containers
  - Examples: https://github.com/linuxkit/linuxkit/tree/master/projects/okernel



eXclusive Page Frame Ownership (XPFO)

- Introduced in “Rethinking Kernel Isolation” by Kemerlis, Polychronakis, and Keromytis
- Protects against ret2dir attacks
- 29 files changed, 1013 insertions(+), 57 deletions(-)
- Implementation supports x86 and arm64

docker

Classic attack

```
struct file_operations {
   int (*flush) (...)
};
```

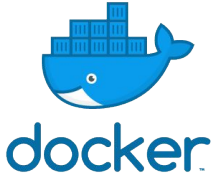
```
// kernel text
int do_flush(...)
{
    ...
}
```

```
// userspace memory
int bad_flush(...)
{
    commit_creds(prepare_kernel_cred(0));
}
```

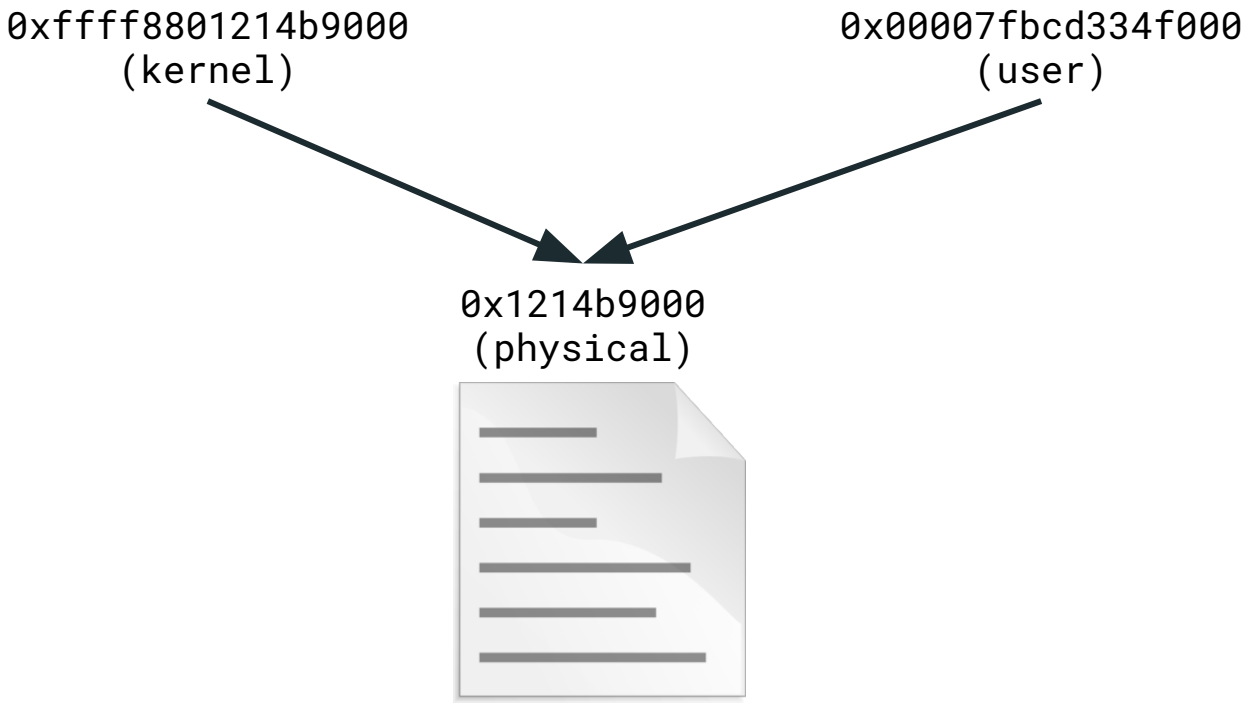
docker

Classic attack

- PaX UDEREF
- SMEP+SMAP on x86
- PXN on ARM



Basic idea


0xffff8801214b9000
(kernel)
0x00007fbcd334f000
(user)
0x1214b9000
(physical)

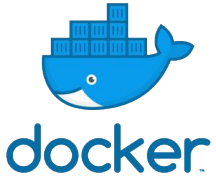
Updated attack

```
struct file_operations {
   int (*flush) (...)
};
```

```
// kernel text
int do_flush(...)
{
    ...
}
```

```
// userspace memory
// 0x00007fbcd334f000
int bad_flush(...)
{
    commit_creds(prepare_kernel_cred(0));
}
```

```
// userspace alias in kernel
// 0xffff8801214b9000
```

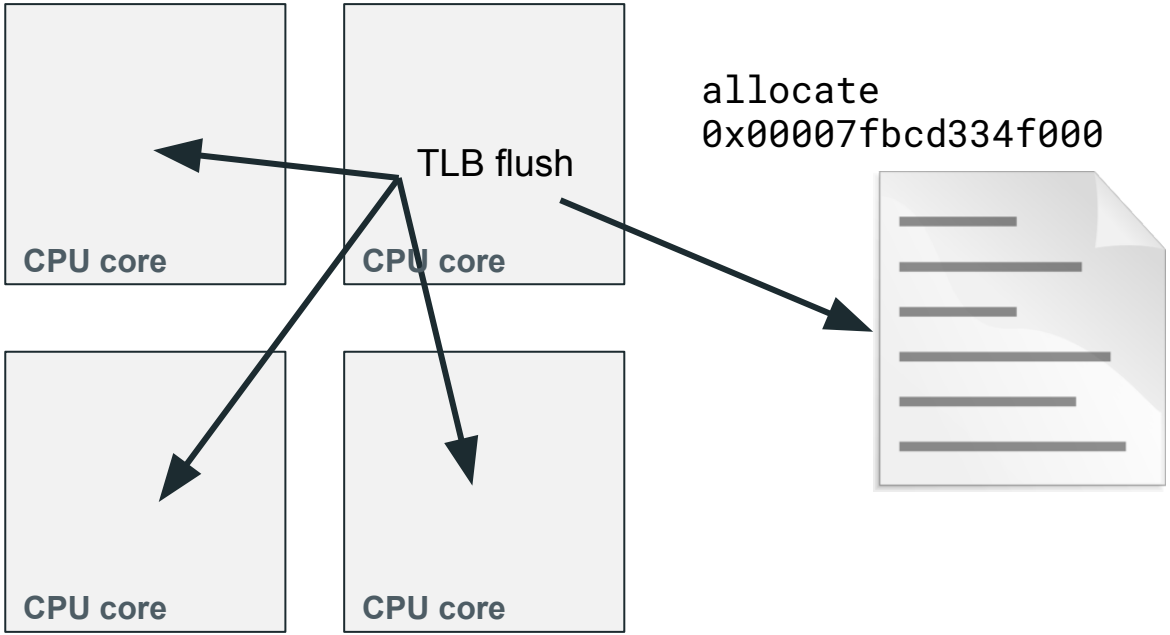


Enter XPFO!

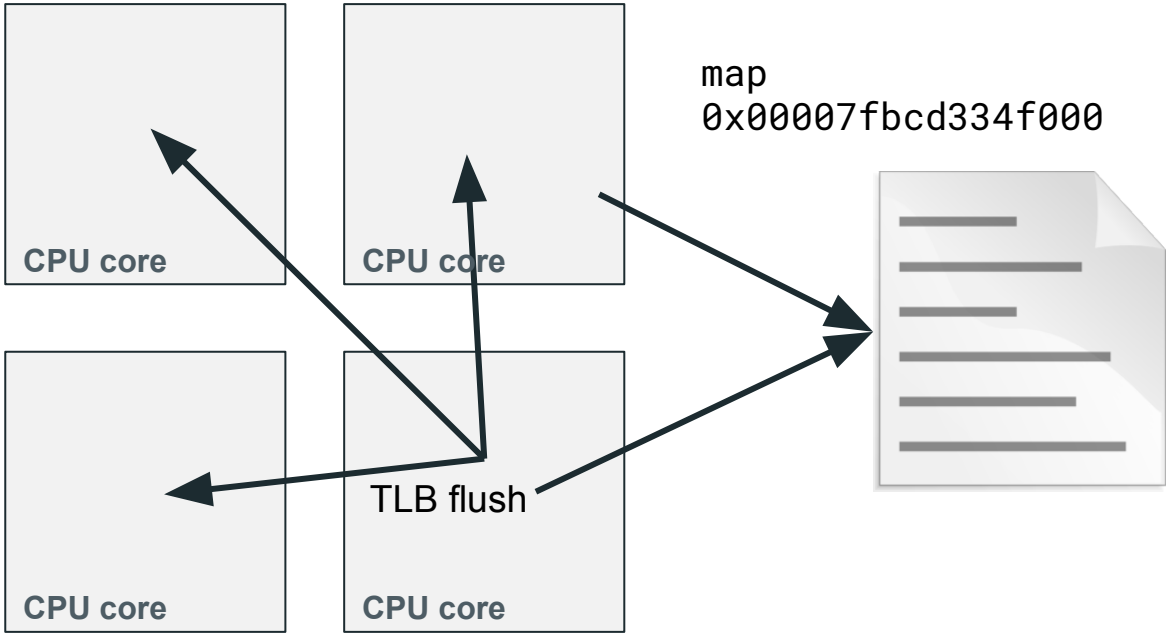
- Keep track of who owns page
- Map/unmap accordingly
- Flush TLB as necessary

docker

On allocation



On map/unmap


CPU core
CPU core
CPU core
CPU core
TLB flush
map
0x00007fbcd334f000

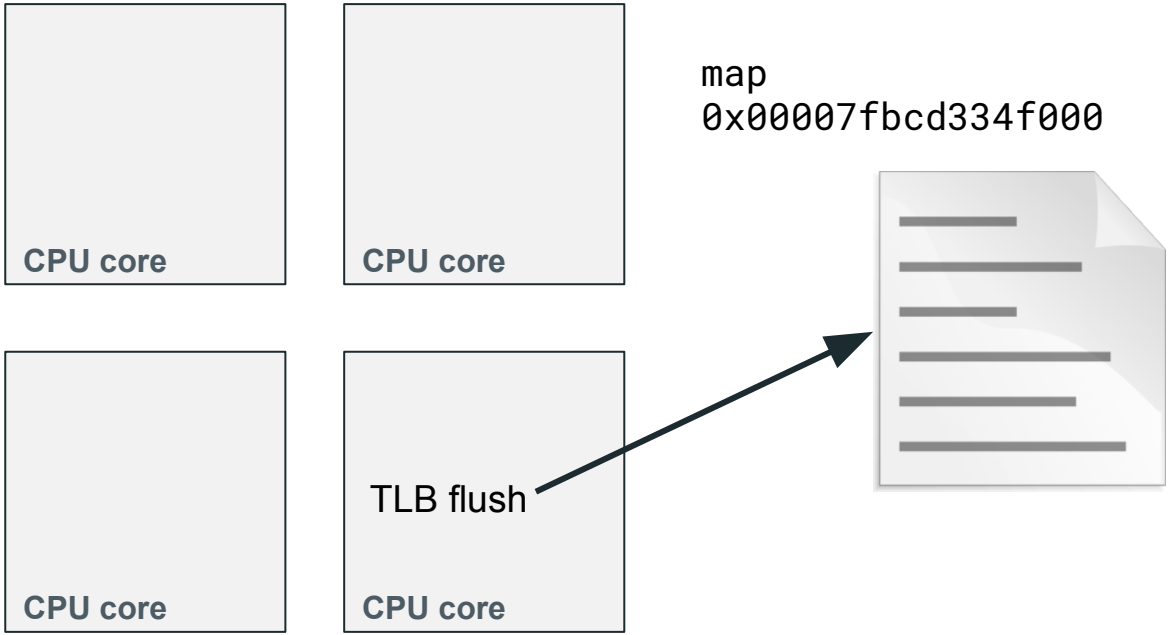
x86

```
void flush_tlb_kernel_range(unsigned long start, unsigned long end)
{
        ...
        on_each_cpu(do_kernel_range_flush, &info, 1);
}
```

x86

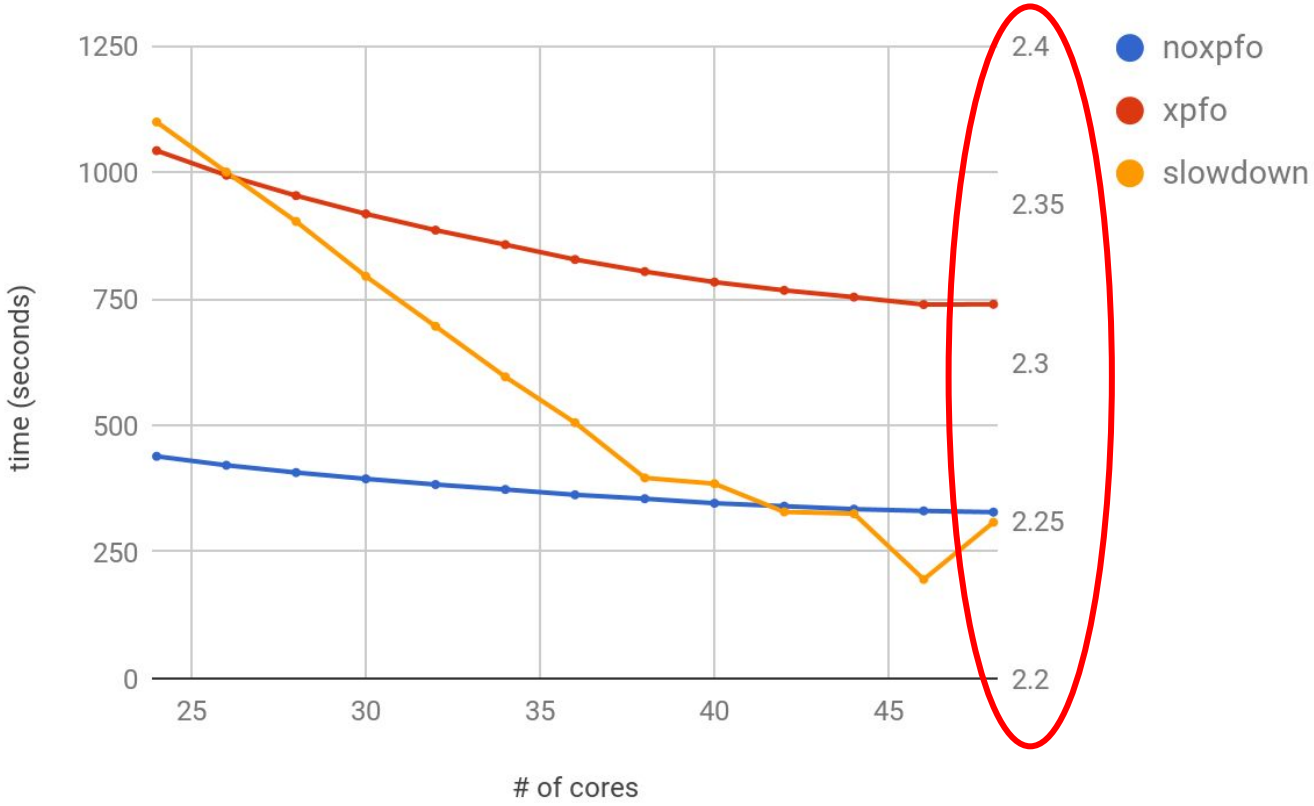
```
/*
 * Can deadlock when called with interrupts disabled. ...
 */
 WARN_ON_ONCE(cpu_online(this_cpu) && irqs_disabled()
              && !oops_in_progress);
```

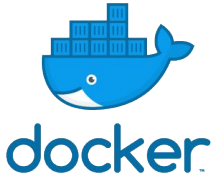
On map/unmap


CPU core
CPU core
CPU core
CPU core
TLB flush
map
0x00007fbcd334f000

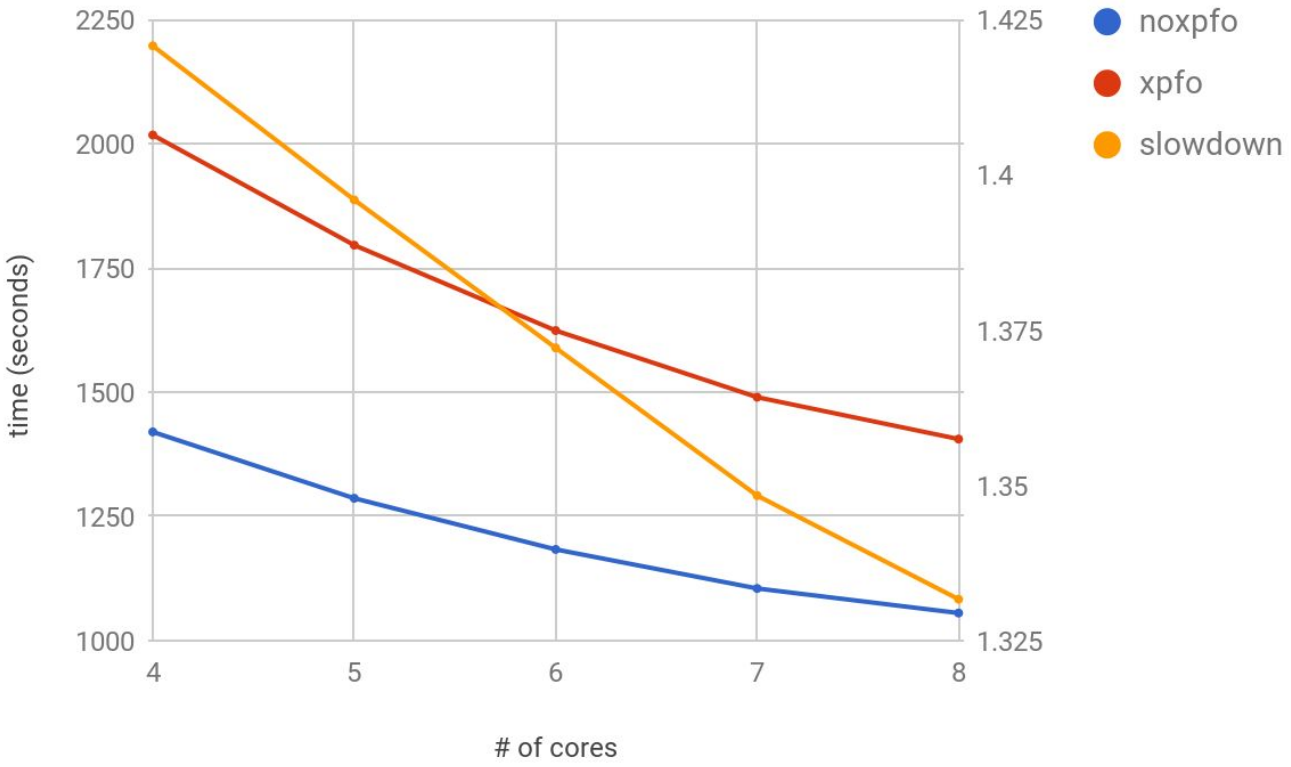
docker

Xeon E5-2650, 24 cores/48 threads

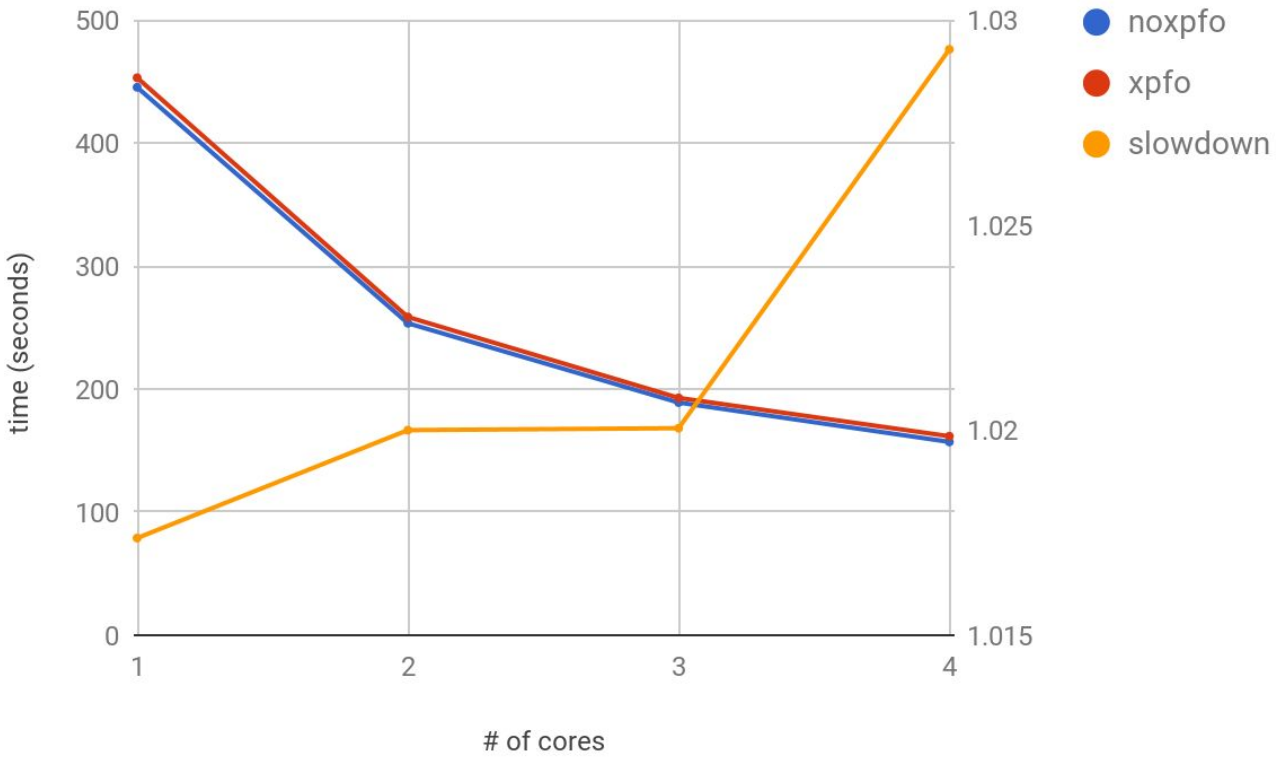


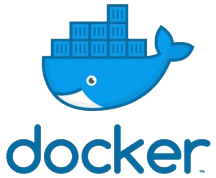


Xeon E3-1240, 4 cores/8 threads



Amlogic Coretex A53 4 cores (odroid-C2)





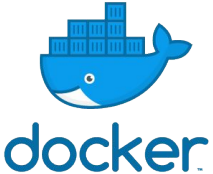
XPFO links

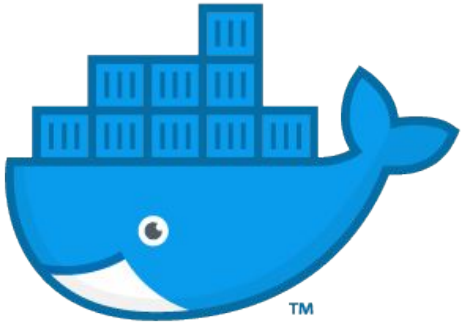
- Original paper: https://cs.brown.edu/~vpk/papers/ret2dir.sec14.pdf
- v6 posting: https://lkml.org/lkml/2017/9/7/445

docker

Future Interests

- collaborate on upstream Linux security:
  - cultivate a testbed and community in LinuxKit for trying new projects
- make containers *contain*





THANK YOU