Hatching Security:
Using LinuxKit as a Security Incubator

Tycho Andersen
tycho@docker.com

Riyaz Faizullabhoy
riyaz@docker.com
Why Linux Kit
Why LinuxKit

- Docker Logo
- Google Cloud Platform Logo
- Windows Logo
- Amazon Web Services Logo

Version 1.12.0-rc2-beta16 (build: 9493)
f615be9fb245904fbcf1aa0cad251d418c869428

Copyright © 2016 Docker Inc. All Rights Reserved.
Docker and the Docker logo are trademarks of Docker Inc.
registered in the U.S. and other countries.
Why LinuxKit
Why LinuxKit
# echo "evil!" > /bin/sh
/bin/sh: can't create /bin/sh: Read-only file system
Why LiNIX KIT
Why LinuxKit
What is **LINUX KIT**

**flexible**

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

• batteries included, but removable
• built from containers
• fast to build
• fast to boot
• run Docker, Kubernetes or whatever you like
• flexible
What is **LINUX KIT**

**immutable**

- **read-only** rootfs
- **immutable** in production
- designed to be managed by **external tooling**
What is **Linux Kit**

**Secure**

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

linuxKit Security SIG

Agenda

- Introductions & Administrivia
- Deep dive: Memorizer
- Projects: open floor for updates
- Next meeting: July 19th

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
SIG Recap:

LandLock LSM

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

WireGuard

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

- Separate parts of the kernel into more and less privileged partitions
- Maps to containers
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
Classic attack

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

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

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

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

0xffff8801214b9000 (kernel)

0x00007fbc334f000 (user)

0x1214b9000 (physical)
Updated attack

```c
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
On allocation

allocate 0x00007fbcd334f000
On map/unmap

CPU core

CPU core

CPU core

CPU core

TLB flush

map 0x00007fbc0d334f000
void flush_tlb_kernel_range(unsigned long start, unsigned long end) {
    ... 
    on_each_cpu(do_kernel_range_flush, &info, 1);
}
/*
 * 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

map 0x00007fbcd334f000

TLB flush
Xeon E5-2650, 24 cores/48 threads
Xeon E3-1240, 4 cores/8 threads

![Graph showing time (seconds) vs. number of cores with 3 lines: noxpfo, xpfo, and slowdown.](image-url)
Amlogic Coretex A53 4 cores (odroid-C2)
XPFO links

- v6 posting: https://lkml.org/lkml/2017/9/7/445
Future Interests

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

- make containers contain
THANK YOU