

DOCKER

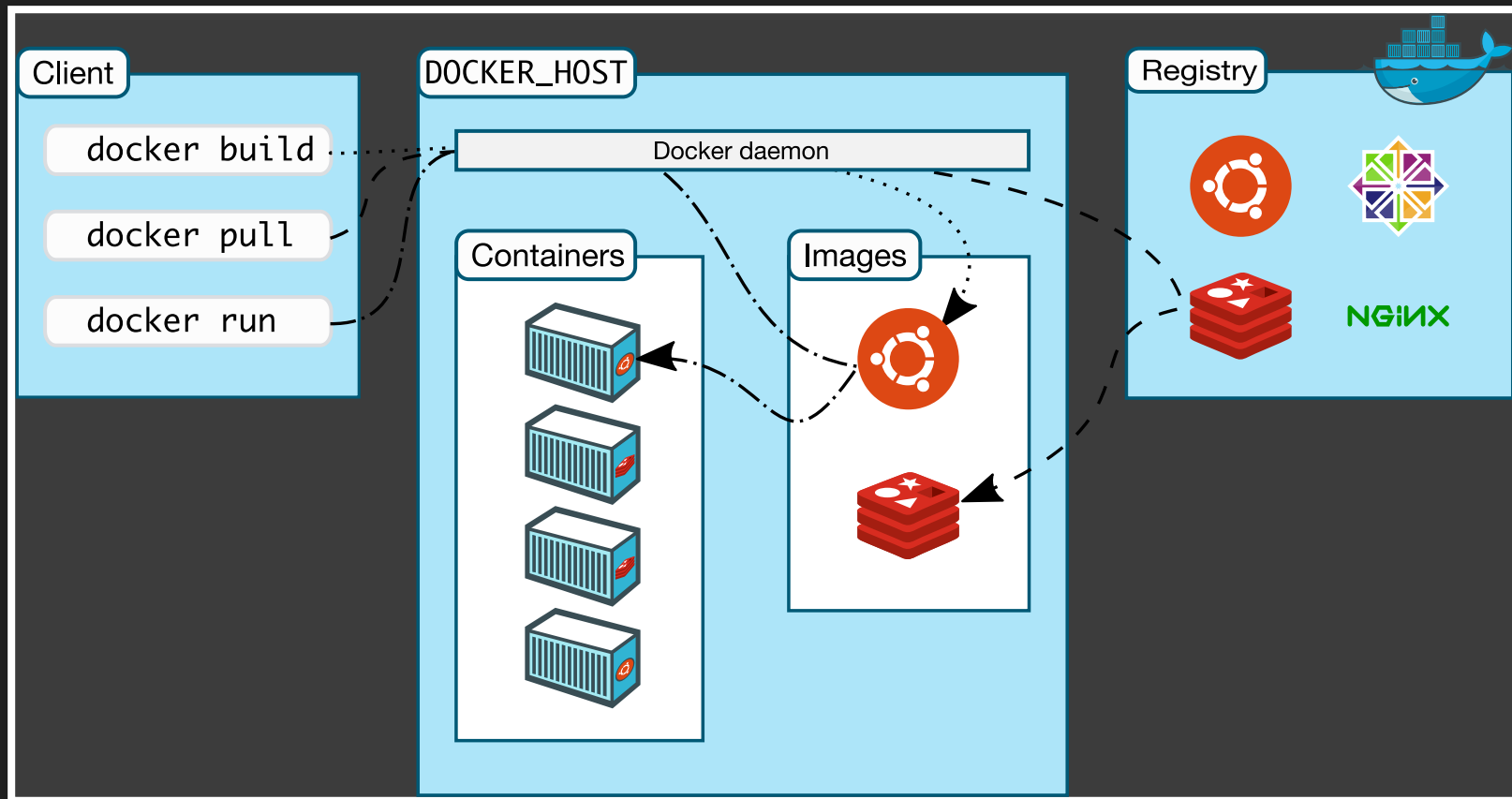
HUH?



docker

CONTAINERS

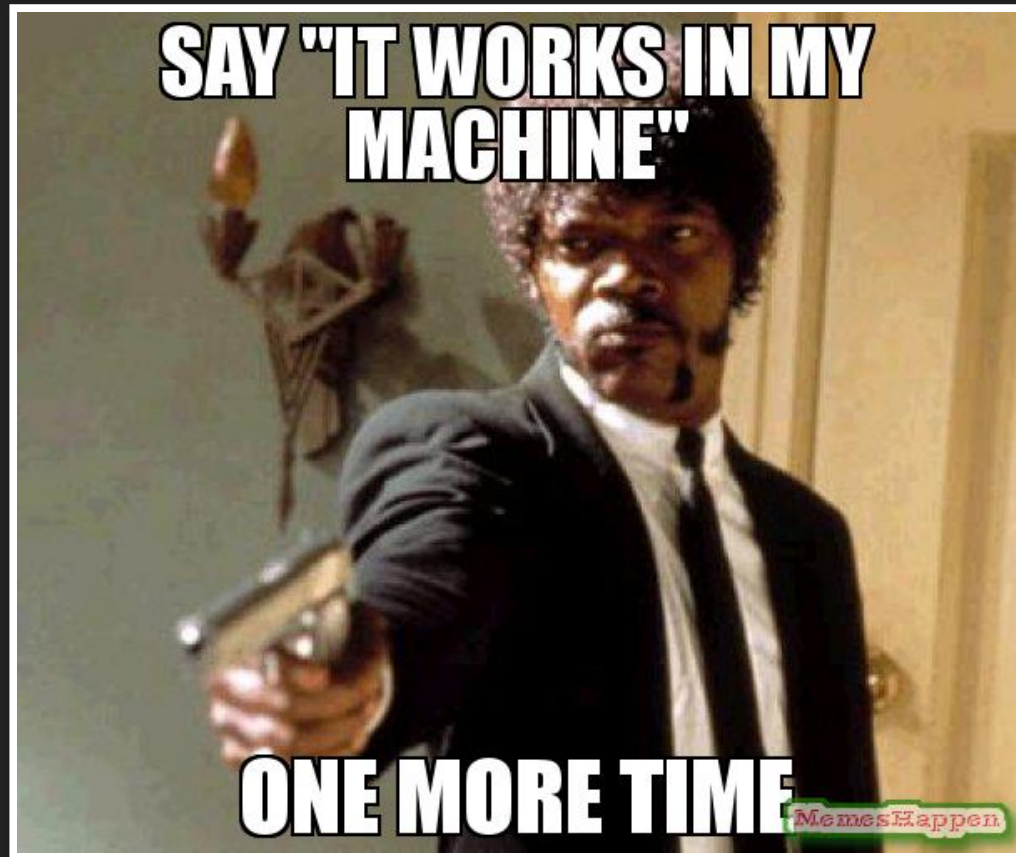




KERNEL PRIMITIVES


- namespaces - control what a process can see
- cgroups - control what a process can do
- LSM - AppArmor, SELinux, Capabilities, Seccomp

IT WORKS



DOCKER HUB





Docker Store is the new place to discover public Docker content. [Check it out →](#)



[Explore](#) [Help](#) [Sign up](#) [Sign in](#)

Repositories (2933)

All

	uhopper/hadoop public automated build	69 STARS	500K+ PULLS	➤ DETAILS
	sequenceiq/hadoop-docker public automated build	543 STARS	500K+ PULLS	➤ DETAILS
	bde2020/hadoop-namenode public automated build	8 STARS	1M+ PULLS	➤ DETAILS
	anchorfree/hadoop-slave	0	1M+	➤


A man with a beard, wearing a white dress shirt and a striped tie, is seated at a desk in an office. He has a wide-eyed, open-mouthed expression of surprise or shock, looking towards the left. His hands are clasped together on the desk. In the foreground, the back of another person's head and shoulders are visible, suggesting a conversation. The background is a blurred office environment with a computer monitor and some papers on the desk.

CONTAINERS FOR ALL!

- "put your apps in a container!" they said
- "devs can manage them!" they said
- "super easy!" they said




CONTAINER BREAKOUTS





**Twistlock**

SECURITY ALERTS

Get Twistlock

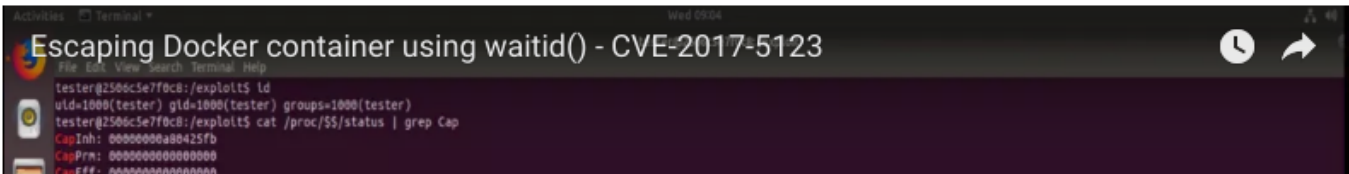
Escaping Docker container using waitid() – CVE-2017-5123

 Dec 27, 2017 | by [Daniel Shapira](#)

share:    

This post describes how I exploited the `waitid()` vulnerability in order to modify the [Linux capabilities](#) of a Docker container to gain elevated privileges, and ultimately escape the container jail. If you want to see how Twistlock would stop this vulnerability in its tracks, check out my [follow up blog](#).

But before we dive in, since an image is worth a thousand words, here is my exploit in action. It modifies the containerized process capabilities structure in memory, resulting in a gain of `CAP_SYS_ADMIN` and `CAP_NET_ADMIN` capabilities. This results in the ability to enable promiscuous mode on `eth0` (docker bridge for the container):



```
Escaping Docker container using waitid() - CVE-2017-5123
tester@2506c5e7f0c8:/exploit$ ld
uid=1000(tester) gid=1000(tester) groups=1000(tester)
tester@2506c5e7f0c8:/exploit$ cat /proc/$$/status | grep Cap
CapInh: 00000000a00425fb
CapPrm: 0000000000000000
CapEff: 0000000000000000
```

FINAL THOUGHTS

- Increased productivity at the cost of complexity and security?
- Is DockerHub a convenient malware distribution channel?
- Thousands of small containers talking to each other - more attack surface?
- Patch hell - what versions of software is running in these things? Old vulns = exploitation opportunities