SECURING CONTAINERS, ONE PATCH AT A TIME

@crosbymichael

Code Monkey

Docker

agenda

- what are cves?
- are cves good?
- history of security && containers
- CVE-2016-9962
- Q&A

whois

- Docker maintainer
- libcontainer author
- OCI runtime spec and runc maintainer
- containerd author/maintainer

man cve

Common Vulnerabilities and Exposures

Common database of vulnerabilities

Naming scheme:
CVE-1999-0067

https://cve.mitre.org/

docker17 con

cve == bad?

Are CVEs Bad?

docker17 con

cve == good

No!

cve == good

CVEs help make software secure

Its part of the security process

docker17 con

cve process

1. Receive report
2. Verify findings
3. Fix vuln
4. Issue a new CVE
5. Send fix to downstreams (distros, clouds, select groups of users)
6. After patch approval, 1 week embargo
7. Public release

docker17 con

process

Security is a process, not a sprint

before runc

/proc/sys/kernel/hotplug

hotplug

Two weeks before Docker 0.7 w/ libcontainer

@ibuildthecloud

Used hotplug to run script as root on the host\

/me sad :(

guess root

```
/var/lib/docker/containers
```

parse id

```
# cat /proc/1/cgroup
11:cpuset:/docker/4fa7f0f0eba4bb475242f3f4f7014370f2a8ba
84657fca29c164f9f77ef9b507

/var/lib/docker/containers/4fa7f0f0eba4bb475242f3f4f7014
370f2a8ba84657fca29c164f9f77ef9b507
```

write script to container /

```
/hack.sh
```

send to hotplug

```
echo
/var/lib/docker/containers/4fa7f0f0eba4bb475242f3f4f7014
370f2a8ba84657fca29c164f9f77ef9b507/hack.sh >
/proc/sys/kernel/hotplug
```

security #1

libcontainer had a focus on security since day 1

CVE-2016-9962

using ptrace to access privileged process fds

preface

There is always two processes:

- Parent (in the host)
- Child (in the container)

docker17 con

container creation (init)

1. Create Namespaces
2. Setup Root Filesystem
3. Apply LSMs
4. Drop capabilities
5. Change user/groups
6. Sync with parent over FIFO
7. Exec()

docker17 con

container creation (exec)

1. Join Namespaces
2. Apply LSMs
3. Drop capabilities
4. Change user/groups
5. Exec()

LXC CVE-2016-8649

a process that is joining a container could be ptrace-d by a process already inside the container.

passing fd to host's /proc to setup LSM labels

reported by @cyphar on OCI maintainer list

docker17 con

runc CVE process

1. Is runc affected?
2. What is the severity?
3. Proposed fixes.
4. Can anything else use this attack vector?

docker17 con

Is runc affected?

runc passes the state directory fd to the init and exec processes

required by `runc create/run` but not `runc exec`

docker17 con

container creation (init)

1. Create Namespaces
2. Setup Root Filesystem
3. Apply LSMs
4. Drop capabilities
5. Change user/groups
6. *Sync with parent over FIFO*
7. Exec()

docker17 con

Is runc affected?

DEMO!

severity

1. Does CAP_SYS_PTRACE block this?
2. Does apparmor or selinux protect?
3. Was this fixed in a newer kernel?

docker17 con

severity

1. Does CAP_SYS_PTRACE block this?
    a. nope
2. Does apparmor or selinux protect?
    a. apparmor kinda / selinux kinda
3. Was this fixed in a newer kernel?
    a. meh

docker17 con

proposed fixes

Remove the fd from exec, ez pz

But...

Can anything else use this attack vector?

docker17 con

the tonis factor

Give @tonistiigi 1 vuln, he will give you 3 more back.

docker17 con

the tonis factor

Additional way to exploit the same fd vuln

One extra vuln with /proc/{pid}/exe

/proc/{pid}/exe

Replace binary file by writing to it

Super symlink that can travel across space and time

~ echo hi >> /proc/{some pid}/exe

~ echo ‘#!/bin/sh;evil stuff’

docker17 con

Why is this happening?

We drop CAP_SYS_PTRACE so why is this happening?

Docker drops CAP_SYS_PTRACE by default

can we use dumpable?

Set the state of the "dumpable" flag, which determines whether core dumps are produced for the calling process upon delivery of a signal whose default behavior is to produce a core dump.

docker17 con

rules for ptrace

rules for ptrace

no man pages...no
google...no

kernel source:

https://github.com/torvalds/linux/blob/master/kernel/ptrace.c#L265

docker17 con

rules for ptrace

You can ptrace if:

You are the parent of the process && process is dumpable

You have CAP_SYS_PTRACE

docker17 con

dumpable

What if we changed the dumpable settings for the init processes?

Dumpable is reset on execve, perfect!

Side-effects? You cannot strace runc.

Dumpable Fix

```
/* make the process non-dumpable */
if (prctl(PR_SET_DUMPABLE, 0, 0, 0, 0) != 0) {
    bail("failed to set process as non-dumpable");
}
```

Did it work?

Stopped walking the fd

Stopped overwriting the /proc/{pid}/exe

But...

Did it work? Kinda...

Before dumpable fix: Reproduced 100/100

After dumpable fix: Reproduced ~3/100

docker17 con

Where can it race?

The fd can only be accessed when process is dumpable.

Cannot be accessed after it is closed.

Dumpable is reset on execve( ).

docker17 con

Hard Questions

the fds have O_CLOEXEC set

docker17 con

Hard Questions

So...

When are O_CLOEXEC fds actually closed?

docker17 con

exec syscall implementation

```
void setup_new_exec(struct linux_binprm * bprm)
{
    if (uid_eq(current_euid(), current_uid()) &&
    gid_eq(current_egid(), current_gid()))
        set_dumpable(current->mm, SUID_DUMP_USER);
    else
        set_dumpable(current->mm, suid_dumpable);
...
    do_close_on_exec(current->files);
```

execve flow

1. Load binary
2. Set /proc/{pid}/exe link to new binary
3. Change dumpable setting
4. Close all O_CLOEXEC fds
5. Run new code

https://github.com/torvalds/linux/blob/v4.9/fs/exec.c#L1290-L1318

docker17 con

runc Patch

Close fds before calling execve in the container's init.

https://github.com/opencontainers/runc/commit/50a19c6ff828c58e5dab13830bd3dacde268afe5

docker17 con

Linux Kernel Patch

@cyphar's patch to linux changing the order of O_CLOEXEC and resetting dumpable

https://github.com/torvalds/linux/commit/613cc2b6f272c1a8ad33aefa21cad77af23139f7

Thank You!

@crosbymichael

docker17 con