Docker, Docker, Give Me The News, I Got A Bad Case Of Securing You

David Mortman
Chief Security Architect, Dell Software
david.mortman@software.dell.com
@mortman

Introduction

What's the big deal about Docker/AppC anyways? We've had containers for 20 years. What new things are they bringing to the table aka how are they different from LXC and traditional containers? What these new container formats, which leverage existing technologies introduce is that they ease not just the ability to run applications in isolation but more importantly they vastly ease the build and ship parts of the equation as well. They take the ease of use of application deployment that Chef/Puppet etc to the next level while simultaneously reducing complexity at the same time. The value of this shift cannot be overstated.

This value is added by wrapping the container in a metadata layer (actually multiple layers but that's another story) that describes the configuration of the container and its contents. As a result of this wrapping, are no longer just about security but are now also to all intents and purposes another packaging format with all of the benefits and downsides that come with said systems. Where this gets particularly exciting however is that this isn't limited to an individual executable being deployed but can be entire application stacks. This goes well beyond a traditional package stating what its dependencies are and enabled one stop shipping of an entire application in one fell swoop. This creates benefits for everyone from dev to qa to production regardless of the underlying development frameworks being used. It also addressed many of the problems traditionally faced by organizations trying to leverage multi-cloud or multi-cloud-account

application architectures.

Of course, it's not all rainbows and unicorns, using tools like containers does introduce their own sets of security considerations, none of which are insurmountable, but nonetheless need taking into consideration.

So what are the general issues that containers introduce? Unsurprisingly, these are mostly they are a lot of the same issues that virtualization and cloud introduced over the last decade or so but with some twists and variations. But at a high level it's just not that different. So that's actually some pretty good news. Security people, sometimes myself included looooove to bash on new tech. Well I got some news for y'all. Containers are here to stay. So it's time to get on the bus or get run over by it.

So there have been a handful of really fun posts from people showing how you can trivially get root on a box if you have root level access in a container and sometimes even if you don't.

With one main exception (which I'll get to later), these now require that you already have root on the host OS already. Over the past year the Docker team has done a tremendous job making security fixes and also changing to much more sensible default configuration modes.

Escapes Are Trivial!

Well They Were!

Namespaces (except user namespaces (yet!)
cgroups
dedicated network stacks
manifest signing (and it's getting better!)

What Do I Need To Do?

That being said, there's a bunch of stuff to keep in mind when deploying Docker that you'll want to do to further harden your systems. (h/t to Docker CIS Benchmark)

Locking down docker:

Restrict network traffic between containers

turn on auditd for docker for files and network

    Then monitor/audit those logs

only use ssl/tls enabled registries (default)

don't enable docker to listen on network port but if you must enable tls auth

lock down all config files to root.root and perms of 644 or tighter

lock down all certs/keys to root.root and perms of 400

run containers as non-root users

only use trusted images

    More on that later

minimize package installs

1 app/process per container

Restrict Linux Kernel Capabilities within containers

For example, capabilities such as below are usually not needed for container process:

NET_ADMIN

SYS_ADMIN

SYS_MODULE

Don't use privileged containers

Do not mount sensitive host system directories on containers

Eg /etc /dev /proc

Don't ssh into containers use nsenter

Don't use priviliged ports if at all possible

Set reasonable maximums for memory usage

Set reasonable cpu priority

Set reasonable ulimits

Mount containers root partition at read-only

Restrict inbound traffic to specific interfaces

Limit automated container restarts to a small number

Don't share hosts namespaces or devices to containers

backups (duh!)

get logs elsewhere and centralize

minimal number of images

minimal containers per host

Use trusted containers

Supply chains ← 30% of images have vulns??!!

patch your containers

don’t use chef/puppet etc

attribution issues

Further enhancing your security

use apparmor and/or selinux

use secomp (limits syscall and syscall arugments on a case by case basis)

Docker Bench Security

SecComp -- limits syscall and syscall arugments on a case by case basis

LXD

Apcera

Coming improvements

V2 registry/Notary/TUF

- Notary has a concept of freshness
- 4 keys
  - root role: like a CA root
  - targets (signs the content, aka sign tag to hash mapping (can self verify against the registry v2)),
  - timestamps (freshness),
  - snapshots (allows you to fix versions of dependencies)
- survivable to key compromise

User namespaces (in runc in 1.8) -- e.g. map root to non-root, only has root privs in containers i.e.

What's Left?

Areas that need work still

Kernel's keyring isn't namespaced (SELinux helps here)

Managing secrets

- Vault (hashicorp)
- Keywhiz (from square)

API needs Authn/Authz

Making this all much much easier

- Ease of use of secomp
- Ease of use of selinux/apparmor

Logging

Orchestration

Resources:

https://d3oypxn00j2a10.cloudfront.net/assets/img/Docker%20Security/WP_Intro_to_container_security_03.20.2015.pdf

https://docs.docker.com/articles/security/

https://github.com/docker/docker-bench-security

https://benchmarks.cisecurity.org/downloads/show-single/index.cfm?file=docker16.100

http://container-solutions.com/docker-security-cheat-sheet/

https://github.com/GDSSecurity/Docker-Secure-Deployment-Guidelines

http://www.ubuntu.com/cloud/tools/lxd

https://www.apcera.com

Wrap Up

Q&A

Docker, Docker, Give Me The News, I Got A Bad Case Of Securing You

David Mortman
Chief Security Architect, Dell Software
david.mortman@software.dell.com
@mortman