

# Containers, Docker, and Security: State of the Union

# Who am I?

- Jérôme Petazzoni ([@jpetazzo](#))
- French software engineer living in California
- Joined Docker (dotCloud) more than 4 years ago  
(I was at Docker *before it was cool!*)
- I built and scaled the dotCloud PaaS  
(millions of containers, no *known* security issues)
- I learned a few things about running containers  
(in production)

# Outline

- Yesterday
- Today
- Tomorrow

# Yesterday

# Containers and Security yesterday

Last year, at LinuxCon...

- "Is it safe to run applications in containers?"

# Containers and Security yesterday

Last year, at LinuxCon...

- "Is it safe to run applications in containers?"

*really meant*

"Can one container break out, and into another?"

# Containers and Security yesterday

Last year, at LinuxCon...

- "Is it safe to run applications in containers?"

*really meant*

"Can one container break out, and into another?"

- Main concern: isolation

**What was the answer?**



# What was the answer?

Docker has changed its security status to

**It's complicated**



# What was the answer?

- "It's complicated"
- Long list of recommendations
  - some were easy  
(and automatically enforced by Docker)
  - some were not obvious  
(and had to be enabled manually)
  - some were hard to deploy  
(or required missing kernel features)
- [Video](#) and [Slides](#)

# How is this different today?

- People still ask about container isolation
- Much more frequently, they ask about *image security* and provenance
- They want to know:
  - if `docker pull debian` is what it claims to be
  - if `jpetazzo/dind` has vulnerabilities
  - if a given image has been vetted by their sec team

# Why has it changed?

- Who cares about container isolation?
  - hosting providers (more density = more \$\$\$)
  - PAAS (for rapid deployment; on-demand activation)

→ early adopters

- Who doesn't care about container isolation?
  - people who use VMs only because autoscaling
  - people who would put multiple components per machine anyway

→ second wave of users

# Today

# Docker and Containers Security Today

- Improving what we had yesterday  
(fine-grained permissions, immutable containers)
- Addressing new challenges  
(provenance, content verification, notary)
- Defense in depth  
(containers + VM)
- The infosec mindset  
(better upgrades, security benchmarks, policies)

# Finer-grained permissions

- Per-container ulimit
- Capability reduction  
--cap-drop / --cap-add  
e.g.: --cap-add net\_admin
- Device access restrictions  
--device (better than --privileged!)
- Improved handling of LSM  
(SELinux, AppArmor)

# Smaller attack surface

- Hardware management done on the host  
(no kernel, drivers, device handling... in containers)
- Package management is optional  
(once a container is built, it doesn't need to be changed)
- Minimal distros can be used  
(e.g. buildroot, Alpine Linux...)
- Less software = less risk



# Immutable containers

- `docker run --read-only`  
(makes it impossible to entrench in a container)
- Helps with vulnerability detection  
(audit can be performed on offline images)
- Even without `--read-only` flag:
  - copy-on-write prevents changes from being permanent
  - break a container when hacking it → it gets recycled
  - `docker diff` allows easy audit of changes

# Image provenance

- How can I trust `docker pull debian`?
  - I must trust upstream  
(i.e. Debian and whoever maintains the image)
  - I must trust Docker Inc.  
(operator of the Hub)
  - I must trust the transport  
(between the Hub and my Docker host)

# Image provenance

- How can I trust `docker pull debian`?
  - I must trust upstream  
(i.e. Debian and whoever maintains the image)
  - I must trust Docker Inc.  
(operator of the Hub)
  - I must trust the transport  
(between the Hub and my Docker host)
- That's a lot of trust

# "I don't want to trust anybody!"

- If you don't trust upstream, you have to ...
  - stop using `apt-get` and `yum` with public repos
  - rebuild everything from source
  - verify source integrity  
(full audit + review all changes)
- If you don't trust Docker Inc., you probably should ...
  - audit the whole Docker Engine code
  - audit every single patch that goes into Docker  
(if you can do that ... we're looking for reviewers)

# Security reminder

It's OK to be paranoid, but beware of:

- Bumps in the carpet  
(moving a problem rather than solving it)
- Usability  
(if security makes it hard/impossible to work,  
people *will* work around it!)
- Tinfoil hats

# Can we trust the transport?

- Registry v1 protocol had serious issues:
  - arbitrary layer IDs
  - no integrity check  
(other than TLS transport integrity)
- Registry v2 protocol has:
  - content-based layer IDs
  - signed image manifests

*Is that enough?*

# Notary: a better trust framework

# What are we trying to address?

- Distributed content should be signed
- Stealing a key should be hard
- Stealing a key shouldn't have dire consequences
- Replay attacks should be hard  
(=can't serve you yesterday's vulnerable version)
- Should use known models and research
- Existing distribution infrastructure should be used  
(=HTTP, HTTPS, FTP... are good)
- Trusting Docker Hub should not be mandatory



# Notary

- Based on TUF (The Update Framework)
- Sign content with offline keys
- Trust thresholds (require K out of N keys)  
  
(Stealing a key reduces signing requirements,  
but doesn't break the whole model)
- Guarantee freshness
- Distribute signed content on (potentially insecure) servers  
(leverage existing (insecure) transport and mirrors)
- Enabled in Docker 1.8 by setting DOCKER\_CONTENT\_TRUST

# Defense in depth

*So, VM or containers?*

# Defense in depth

*So, VM or containers?*

*VM **and** containers!*

# Defense in depth

*So, VM or containers?*

*VM **and** containers!*

- Reduce number of VMs  
(when security perimeter allows it)
- Colocated containers are safer than colocated processes
- Malicious code has to escape both layers
- Docker provides an extra layer of isolation
- Applications are *safer* with containers than without

# The infosec mindset

- Better upgrades
- Accurate, actionable security benchmarks
- Clear, sensible security policies

# Better upgrades

- Dockerfile = easy, fast, reliable builds and rebuilds
- "But now I have 1000s of container images to upgrade!"
- Yes, but that's way better than the 100s of server images that you had before
- The *organizational risk* is lower (because if something goes wrong, you have reliable rollbacks)

# Security benchmarks

- CIS (Center of Internet Security) Docker Benchmark
- Docker Bench (<https://dockerbench.com>) :  
automated assessment tool to check compliance

# Policies

- Docker Inc. (the company) and the Docker Project (open source) have clear security guidelines
- Mandatory code reviews (see [CONTRIBUTING.md](#)) to ensure quality of code base
- Quarterly security audits and pen tests of our infrastructure
- We support responsible disclosure



# Tomorrow

# Container security in the future

*Personal predictions - not Docker Inc.'s roadmap!*

# Container security in the future

*Personal predictions - not Docker Inc.'s roadmap!*

- Offline image audit
- Hardening of immutable containers (noexec, nosuid)
- Better GRSEC, PAX, LSM integration
- User namespaces (eventually!)
- Better default seccomp profiles

# Last words

David Mortman at DEFCON this year:

“ *A year ago, [Docker and security] was pretty horrible, six months ago it wasn't so bad, and now it's pretty usable.* ”

# Resources

- [Docker security page](#)
- [Docker security presentation at DockerCon 2015 SF](#)
- [Docker Security CheatSheet](#)
- [Notary on GitHub](#)
- [Docker Bench for Security](#)

Thanks!  
Questions?

@jpetazzo  
@docker