Containers can actually improve your security story(!)

Maya Kaczorowski, Google Cloud
June 12 2019
Maya Kaczorowski
Security PM, Google Cloud
@MayaKaczorowski
Objection:

“My security team is opposed to containers and Kubernetes”
Security people like to complain about containers and Kubernetes

- What’s a koober net ease
- I can’t use my IDS, firewall, ...
- Containers don’t contain
- I am stuck with it, help me
“70 percent of change programs fail to achieve their goals, largely due to employee resistance and lack of management support.”

Changing change management, McKinsey & Co.
Agenda

1. How container security is different
2. Traditional software supply chain and patch management
3. Ideal software supply chain and best practices in image maintenance, patching, and validation
How container security is different
... container security isn’t that different from any other security
Threats seen in the wild

2018

- **February**
  - Tesla
    - Unsecured Kubernetes dashboard with cloud account credentials
    - **Not exploited**
    - Used to mine cryptocurrency

2018

- **May**
  - Shopify
    - Researcher could access and replay kubelet credentials
    - **Not exploited**

2018

- **June**
  - Weight Watchers
    - Unsecured Kubernetes dashboard with sensitive data, including credentials
    - **Not exploited**

2019

- **April**
  - Docker Hub
    - Database with 190k+ Docker Hub accounts exposed
    - Unknown impact
  - Docker Hub
    - Public images with embedded cryptocurrency mining malware
    - **Used to mine cryptocurrency**

---

Google Cloud
Container security threats & risks

**INFRASTRUCTURE SECURITY**
- Privilege escalation
- Credential compromise
- Kubernetes API compromise
- Over-privileged users

**SOFTWARE SUPPLY CHAIN**
- Unpatched vulnerability
- Supply chain vulnerability
- Zero day exploit on common library

**RUNTIME SECURITY**
- DDoS
- Node compromise and exploit
- Container escape
- Flood event pipeline
How can I use Kubernetes security features to protect my identities, secrets, and network?

How can I use native GCP functionality, like IAM, audit logging, and networking?
SOFTWARE SUPPLY CHAIN

Is my container image secure to build and deploy?

- How can I make sure my container images are vulnerability-free?
- How can I make sure the images I built aren’t modified before they are deployed?
RUNTIME SECURITY

Is my container secure to run?

- How can I identify a container acting maliciously in production?
- How can I take action to protect and isolate my workload?
- How can I securely scale my containers deployment?
How is securing a container different?

**Surface of Attack**
- Minimalist host OS and limits the surface of an attack.

**Resource Isolation**
- Host resources are separated using namespaces and cgroups.

**Permissions**
- Access controls are for app privileges and shared resources.

**Lifecycle**
- Containers have a shorter, better defined lifecycle.
Traditional software supply chain and patch management
Traditional software supply chain
Traditional patch management

01
Get patch

From the distributor, some random mailing list, a vendor. Not always sent to the security team.

02
Take down server n=1 and apply patch

Test the patch in prod! Take some unimportant workload down to make sure nothing goes too bad.

03
Repeat for n servers, where n is unknown

It worked! Now do it again, for everything you think is affected. Miss a bunch of it.
Problems with traditional patch management

- Spreadsheet-driven management
- Down time
- 0days are scary
- Unclear what’s running in your infrastructure / what’s running where / if you even need a patch
Ideal software supply chain
Containers are meant to be short-lived, frequently redeployed, immutable, and help you ‘shift left’
DevSecOps?!?
Running containers allows you to adopt a fundamentally different security model

Containers give you a software supply chain

Containers let you patch continuously, automatically

Containers mean you can actually tell if you’re affected by a new vulnerability
Containers give you a software supply chain
What's different about supply chains with containers

VM based

Debug
Patch
Update
Restart

Monolithic application

Production environment

Manual adjustment

Hard
What's different about supply chains with containers

VM based
- Hard
- Debug
- Patch
- Update
- Restart

Monolithic application

Production environment

Manual adjustment

Container based
- Easy
- Build & deploy
- Build
- Test
- Scan
- Analysis
- QA
- Re-build & re-deploy

Pod
- Microservice

Production environment
Containers let you enforce a software supply chain.
Containers let you patch continuously, automatically
Constantly patch your registry... and roll out as normal

01 Patch the image in your registry
Figure out what’s affected, and apply the patch everywhere you need it.

02 Test, validate, and roll out
Roll out the patch like you would any other infrastructure change, going incrementally.

03 Load balance traffic over
When testing is successful, move traffic over to the new, patched workload, with no downtime.
Containers enable passive patching
not just uptime, but up-to-time
Vulnerability mitigation strategies

**Update packages**
apt-get update & upgrade gets you pretty far. Do this daily.

**Remove packages**
Do you really need $6.022 \times 10^{23}$ debian packages installed on your production image?

**Smaller distro**
In many cases, you can get away with a smaller distro like Alpine or Debian Slim.
Moving to a smaller base

Vanilla → Patched → Minimal → Distroless
Containers mean you can actually **tell if you’re affected** by a new vulnerability.
Check your registry and compare to what you deployed
Figure out what’s in production

Find all the containers in prod
kubectl get pods resolve everything to a digest

Find out what is in those containers
Package manifests, application dependencies

Find out what vulnz are in those packages
Cross reference BOM with CVE databases
Instead, container security should be:

- Centralize and lock down release pipeline
- Build images from trusted sources
- Streamline image scanning and analysis
- Deploy only trusted images
- Monitor continuously
You have a container registry
  > Scan for vulnerabilities

Start here

You have a mandated base image
  > Make it minimal

You have a centralized CI/CD pipeline
  > Enforce what’s deployed
Running containers allows you to adopt a fundamentally different security model:

- Containers give you a **software supply chain**
- Containers let you **patch continuously**, automatically
- Containers mean you can actually **tell if you’re affected** by a new vulnerability
Learn more

Blog post: goo.gl/Ew6hYa

cloud.google.com/containers/security
That’s a wrap.

Learn more:
cloud.google.com/containers/security