Lessons learned securing microservices and containers

Emil Man
Product Security Leader
Honeywell Safety and Productivity Systems
- Containers share host kernel
- Have own bins/libs for application
- Run in namespace on Linux kernel
- Run only what is necessary for their application
- Should run only processes directly necessary for their application execution
- Containers should be stateless
- "Cattle not Pets"
Holistic Security and securing the ecosystem of software development
The components or code you are re-using from others
The code repository or laptop

The Build server
The deployment server

</>

The test infrastructure

Container Repositories

All major targets for attackers

Discretionary Access control based on projects

Least Privilege Access

Role Based Access controls
• Sniff for sensitive data
• Extract data from storage
• Dead Container = LOST Forensics
- Rate Limiting
- Authentication
- Load Balancing
- API Gateway and Web Application Firewalls
- Data Input and Output validation through said API
Service Mesh

- Service mesh is an array of proxies alongside containers
- Each proxy provides a gateway of interactions between containers
- Encrypts traffic between them
- Can verify machine identity to ensure it is talking to a “trusted” container.
- Central controller orchestrates connections.
- Control pane knows each interaction
A wide variety of products that have varying business needs

Maturity takes time

Individual function needs such as Devops, Security, Marketing, Finance – etc – all need to be satisfied

Starting too big, with something far too complex

Adding microservice containers around monolith software
Kubernetes Culture Change

Justin Warren  Contributor  Enterprise & Cloud

Justin is Chief Analyst at Pund-It, consultants and advisors.

Kubernetes is a hot topic these days, as is service mesh (as I wrote about recently) and basically all things cloud native.

But while container-based software deployment is de rigueur among those at the aggressive leading edge of software development, the actual infrastructure management remains—for the most part—something they’d prefer not to deal with personally.

Kubernetes management remains a complex and challenging system to configure and run, though great strides have been made since the platform was first unveiled to the wider world in 2014.

The most common solution is to use Kubernetes that’s maintained by someone else. This makes quite a bit of sense, and follows the same logic as utility services generally: why run your own power station when you can rent a fraction of its capacity as a fungible resource on demand? Leave the complexities of maintaining transmission lines and generators and the grid generally to specialists and concentrate on using the electricity generated in whatever way works best for you.
Vaulting

- No passwords stored in plaintext or in source code inside containers
- Regenerated or revoked if compromised
- Can be easily rotated if needed
• Identity and access management around everything
• Integrated IAM at all steps of our software supply chain
• East to transition people in and out of projects
Gartner Security Controls in Containers

Risk-based controls

Often provided outside of container security products

Basic controls

Foundational controls

L7 network segmentation

Behavior-based controls

Vulnerability management

DAP, FCAP and WAF

API gateways

Authorization between services, MSA resilience

Software policies, SCA, image signing

Secrets management

Hardening, CIS benchmarks, K8s benchmarks, SELinux
Data is the “new Oil” of the 21st century

- How are you securing my data
- Persistence of data outside of the container
- Storage that has proper controls
- Multi-tenancy through customer-controlled encryption keys
- Encryption and Personally Identifiable Information identified from the beginning
- Regulatory compliance & data privacy
Security aware people embedded in engineering teams
FIN