Securing the Container Pipeline

Cem Gürkök

Lead InfoSec Engineer

salesforce

dockercon 16

Securing the Container Pipeline

Cem Gürkök
Lead InfoSec Engineer
Salesforce
@CGurkok

Agenda

- Threats
- Container pipelines and integrity
- Monitoring containers, hosts, apps, networks
- Digital Forensics
- Vulnerability Management
- Hardening
- Demo

Threats

Container Threats & Challenges

Run-time

- Container exploit and resource exposure (App)
- Breaking out of container
- Cross-container attacks
- Resource overuse (DoS)

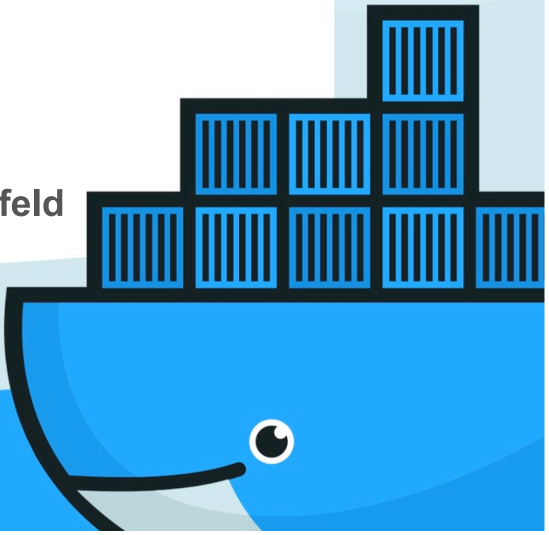
At-rest or transport

- Tampering of images
- Unpatched OS or applications

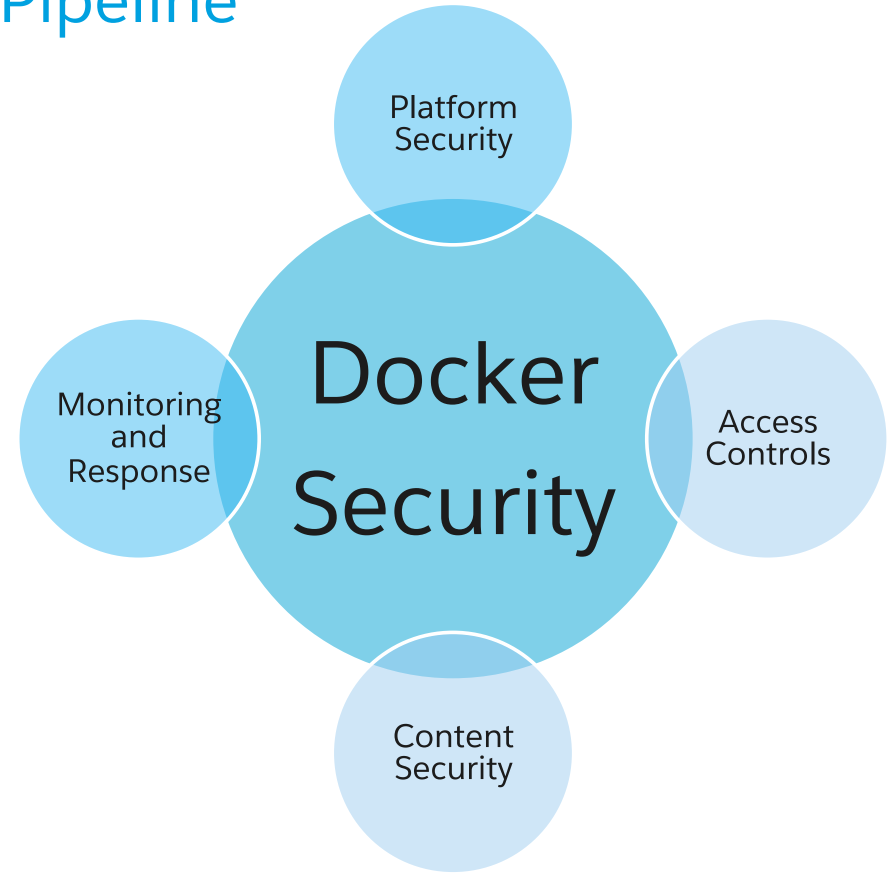
Mitigations

“As we know, there are **known knowns**; there are things we know we know. We also know there are **known unknowns**; that is to say we know there are some things we do not know. But there are also **unknown unknowns**—the ones we don’t know we don’t know.”

— **Donald Rumsfeld**

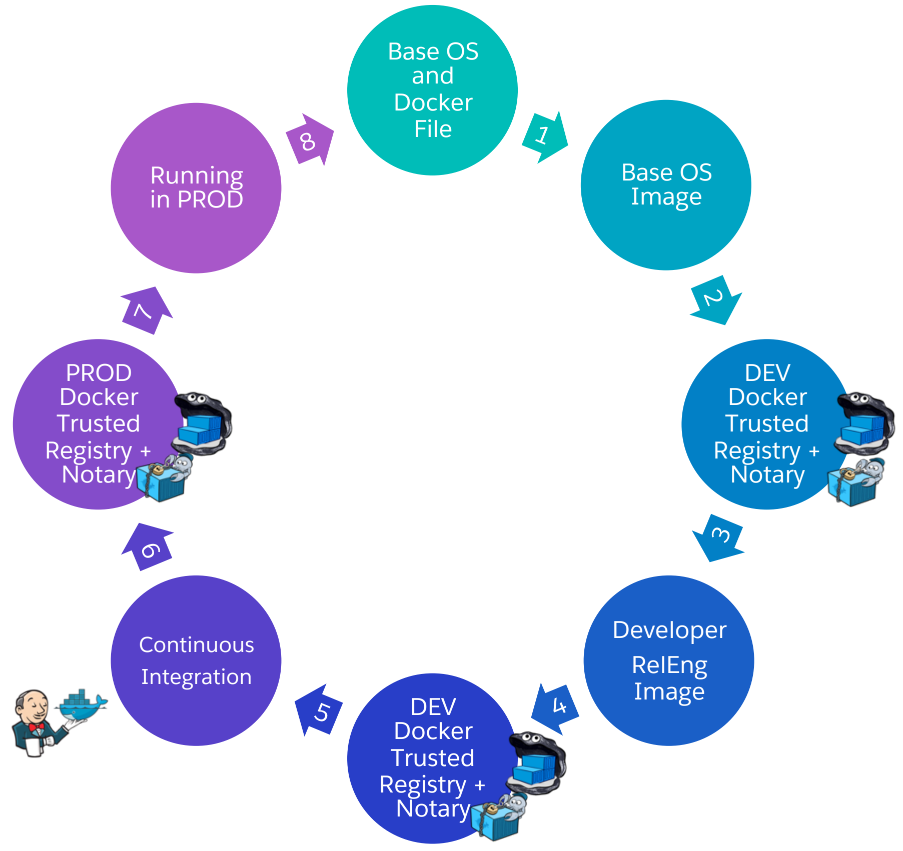


Securing the Pipeline



The Pipeline

Container Pipeline & Security



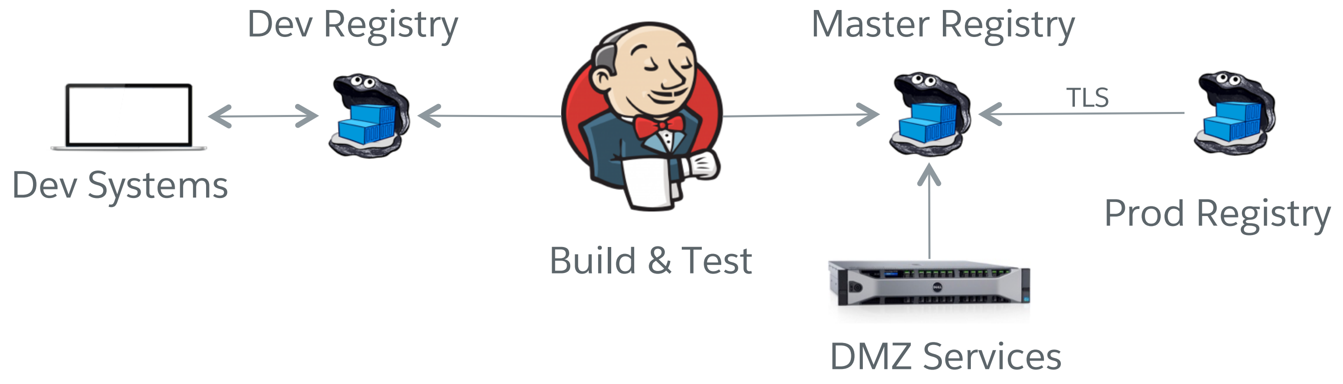
Base OS and Docker File
1
Base OS Image
2
DEV Docker Trusted Registry + Notary
3
Developer RelEng Image
4
DEV Docker Trusted Registry + Notary
5
Continuous Integration
6
PROD Docker Trusted Registry + Notary
7
Running in PROD
8

Monitoring in all steps.

1. Security Review and Hardening
2. Signing, Authentication, Image Vulnerability Scans
3. Authentication, Verification
4. Signing, Authentication, Image Vulnerability Scans
5. Authentication
6. Authentication, Verification
7. Authentication, Verification, Vulnerability Scans
8. Incident Response, Digital Forensics, Patching

Access Control: Authentication

- LDAP over SSL for Docker image transactions:
  - Users (Devs, RelEng)
  - Service accounts
- Mutual TLS Authentication for registry replication



Container Integrity

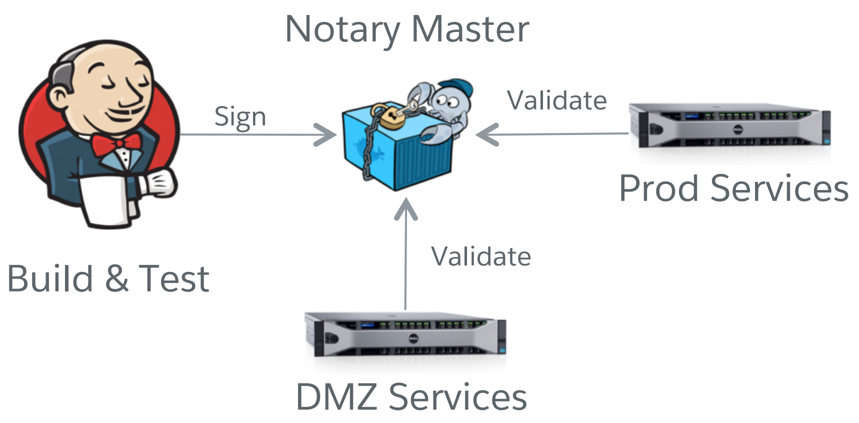
Docker Trusted Registry (DTR)

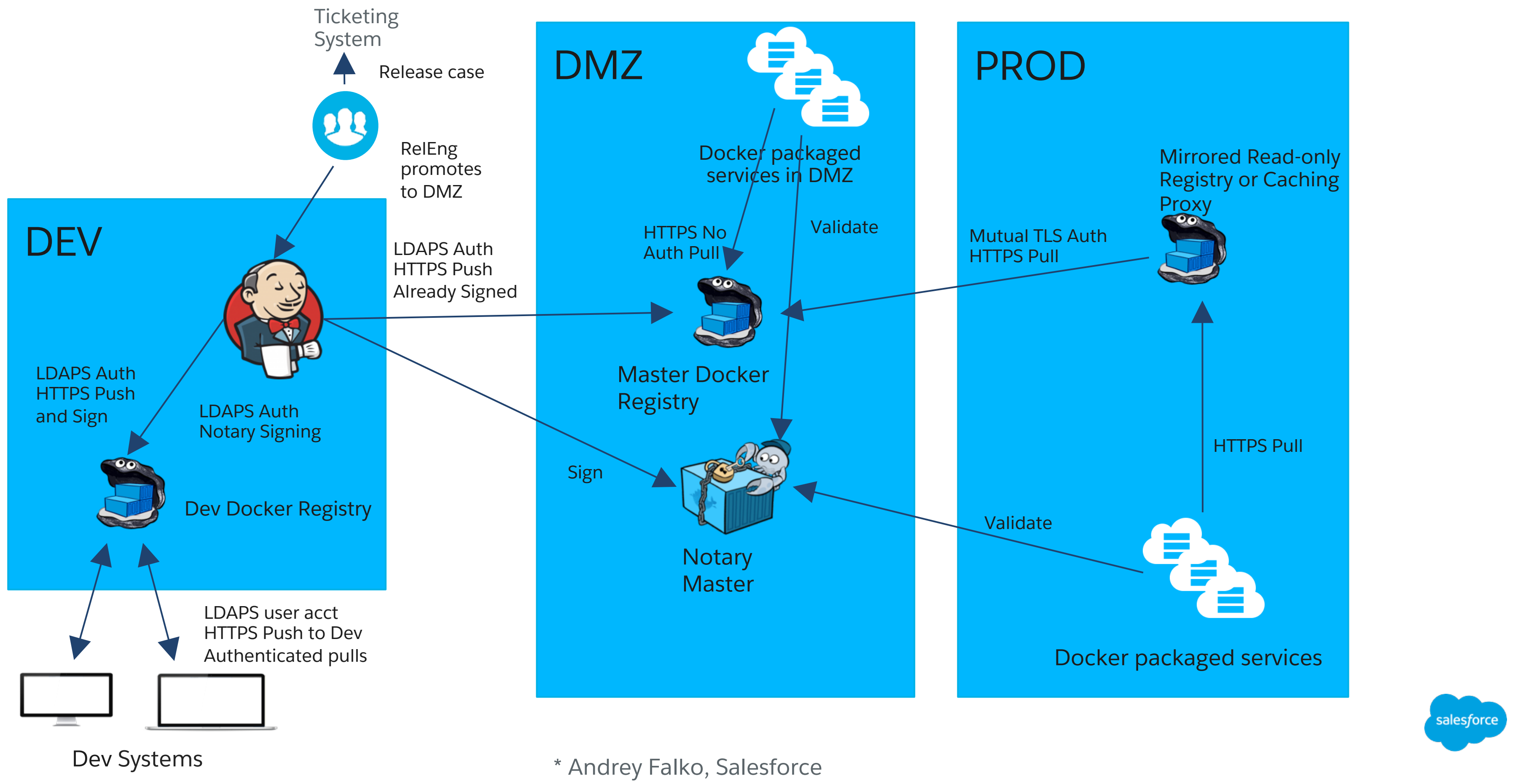
- On-premise
- Authenticated transactions with LDAPS authentication
- DEV and PROD user and image separation
- Users will not be able to disable signing validation
- Validation will be transparent to the users

Container Integrity

Docker Notary

- Enable Docker Content Trust on consumers
- Can enable signing checks on every managed host
- Signature verification transparent to users




Ticketing System
Release case
RelEng promotes to DMZ
DEV
LDAPS Auth HTTPS Push Already Signed
LDAPS Auth HTTPS Push and Sign
LDAPS Auth Notary Signing
Dev Docker Registry
LDAPS user acct HTTPS Push to Dev Authenticated pulls
Dev Systems
DMZ
Docker packaged services in DMZ
HTTPS No Auth Pull
Validate
Master Docker Registry
Sign
Notary Master
PROD
Mirrored Read-only Registry or Caching Proxy
Mutual TLS Auth HTTPS Pull
HTTPS Pull
Validate
Docker packaged services
salesforce

* Andrey Falko, Salesforce

Hardening

Hardening: Host

- Frequent patching
- Install only needed components and libraries (i.e. no gcc or bash)
- Grsecurity/PaX for the kernel
- File system integrity monitoring
- Leverage Linux isolation capabilities!!

Hardening: Container

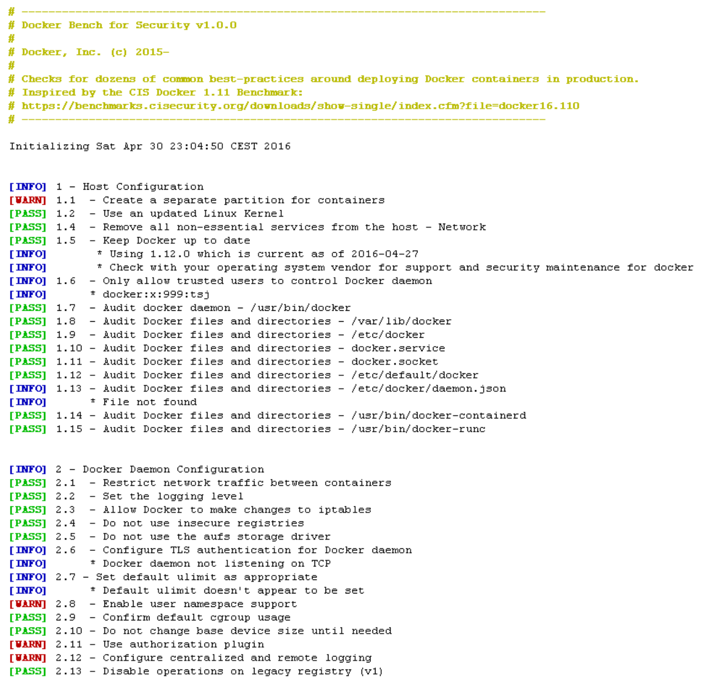
- Base image and app with latest updates/patches
- Leverage User namespaces (run as low priv user on host)
- Install only needed components and libraries (i.e. no gcc or ssh)

Hardening: Container

- Avoid using Docker with the --privileged flag
- Use --read-only when running containers (immutability)
- Avoid providing access to the docker user and group
- Limit and/or separate host and kernel device access

Hardening: Docker Bench for Security

- Docker Bench for Security to the rescue!
  - https://github.com/docker/docker-bench-security
- Checks based on best practices for hosts and containers



```
# ------------------------------------------------------------------------------
# Docker Bench for Security v1.0.0
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker 1.11 Benchmark:
# https://benchmarks.cisecurity.org/downloads/show-single/index.cfm?file=docker16.110
# ------------------------------------------------------------------------------

Initializing Sat Apr 30 23:04:50 CEST 2016


[INFO] 1 - Host Configuration
[WARN] 1.1  - Create a separate partition for containers
[PASS] 1.2  - Use an updated Linux Kernel
[PASS] 1.4  - Remove all non-essential services from the host - Network
[PASS] 1.5  - Keep Docker up to date
[INFO]       * Using 1.12.0 which is current as of 2016-04-27
[INFO]       * Check with your operating system vendor for support and security maintenance for docker
[INFO] 1.6  - Only allow trusted users to control Docker daemon
[INFO]      * docker:x:999:tsj
[PASS] 1.7  - Audit docker daemon - /usr/bin/docker
[PASS] 1.8  - Audit Docker files and directories - /var/lib/docker
[PASS] 1.9  - Audit Docker files and directories - /etc/docker
[PASS] 1.10 - Audit Docker files and directories - docker.service
[PASS] 1.11 - Audit Docker files and directories - docker.socket
[PASS] 1.12 - Audit Docker files and directories - /etc/default/docker
[INFO] 1.13 - Audit Docker files and directories - /etc/docker/daemon.json
[INFO]      * File not found
[PASS] 1.14 - Audit Docker files and directories - /usr/bin/docker-containerd
[PASS] 1.15 - Audit Docker files and directories - /usr/bin/docker-runc


[INFO] 2 - Docker Daemon Configuration
[PASS] 2.1  - Restrict network traffic between containers
[PASS] 2.2  - Set the logging level
[PASS] 2.3  - Allow Docker to make changes to iptables
[PASS] 2.4  - Do not use insecure registries
[PASS] 2.5  - Do not use the aufs storage driver
[INFO] 2.6  - Configure TLS authentication for Docker daemon
[INFO]      * Docker daemon not listening on TCP
[INFO] 2.7 - Set default ulimit as appropriate
[INFO]      * Default ulimit doesn't appear to be set
[WARN] 2.8  - Enable user namespace support
[PASS] 2.9  - Confirm default cgroup usage
[PASS] 2.10 - Do not change base device size until needed
[WARN] 2.11 - Use authorization plugin
[WARN] 2.12 - Configure centralized and remote logging
[PASS] 2.13 - Disable operations on legacy registry (v1)
```

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

Hardening: Vulnerability Management

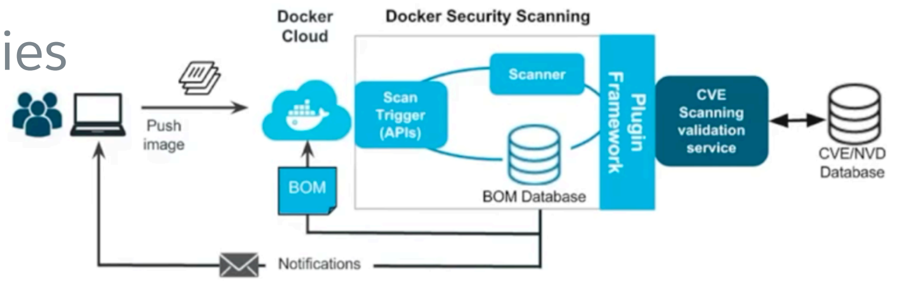
Image Scans with tools, such as Docker Security Scanning:

- Operating System
- Application source code and libraries

Network Scans with traditional vuln scanners:

- Discovery
- Exposed services

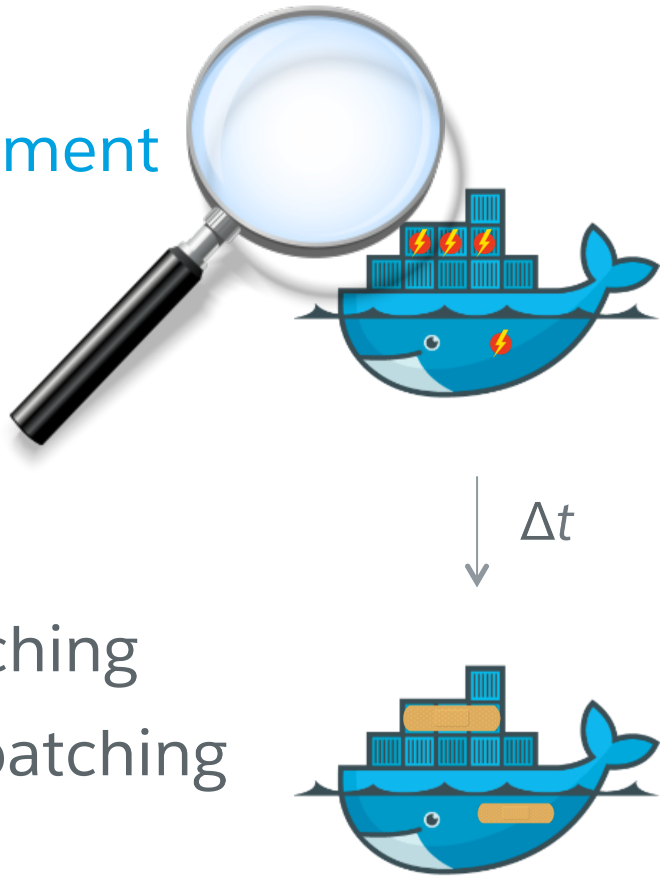
Auto and Manual source code audits



* “Securing the Software Supply Chain with Docker, ” May 2016, Nathan McCauley

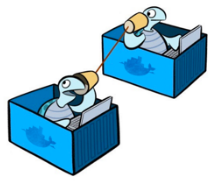
Hardening: Vulnerability Management

- Scanning
  - Docker Images
  - Applications
- Remediation
- Prioritization and SLAs for Patching
- Relaunching containers after patching

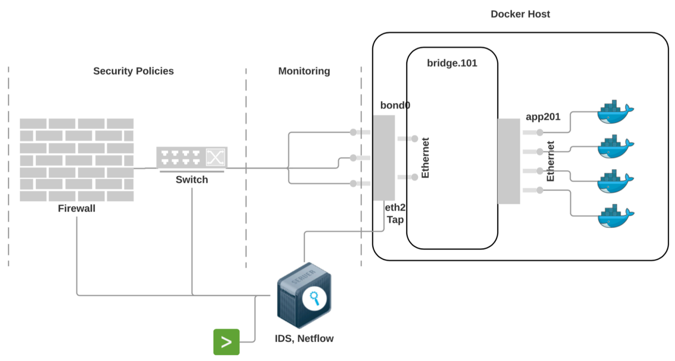


Monitoring

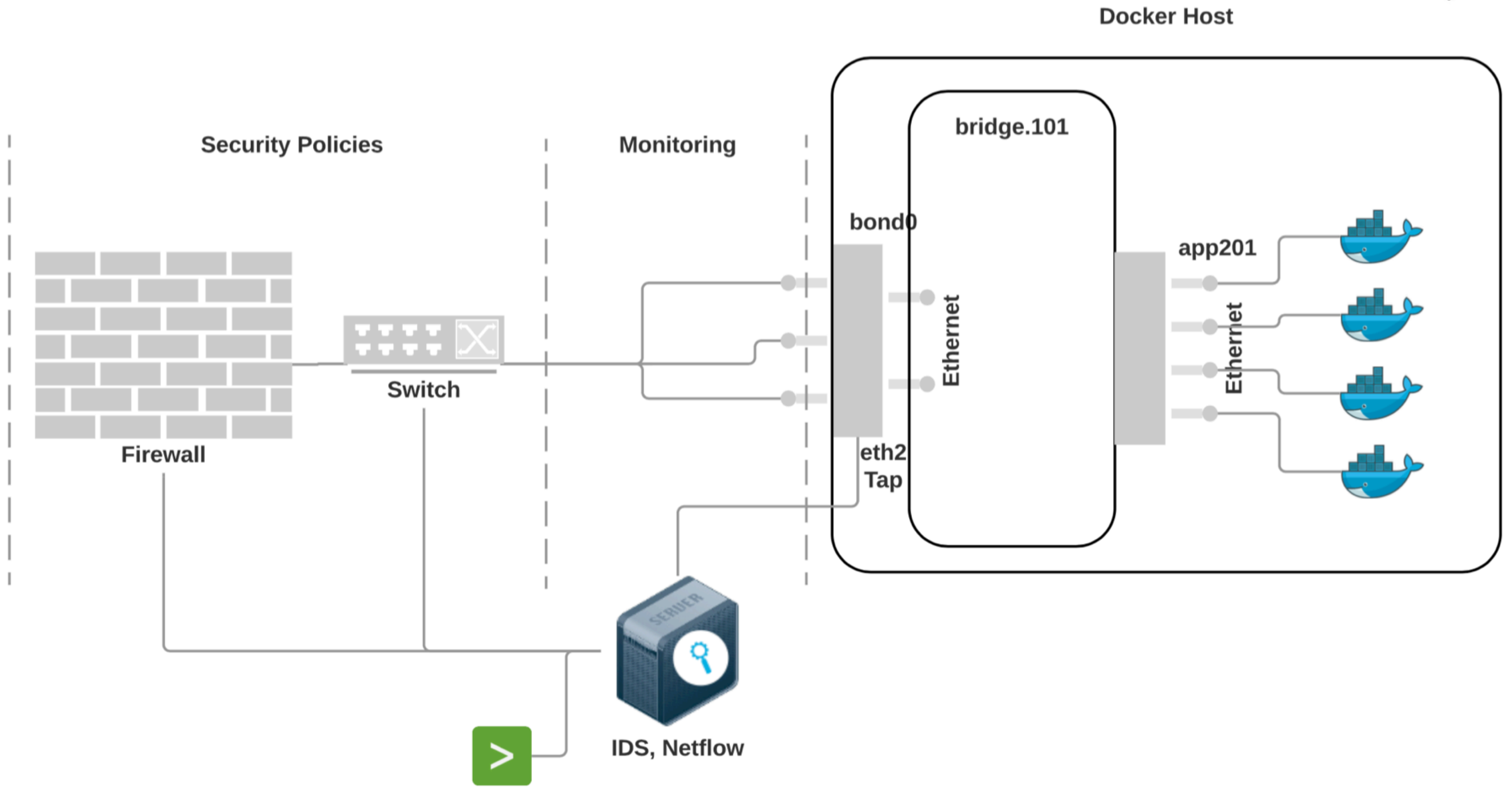
Network Infrastructure



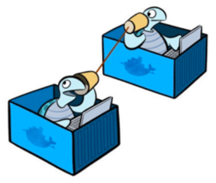
- Bridged networking on Host
- Containers assigned VNICs, IP addresses, and hostnames
- Containers isolated via VLANs (i.e. DB, Web App)
- Tap interface for monitoring
- Security Policies per VLANs and Zones



Network Infrastructure

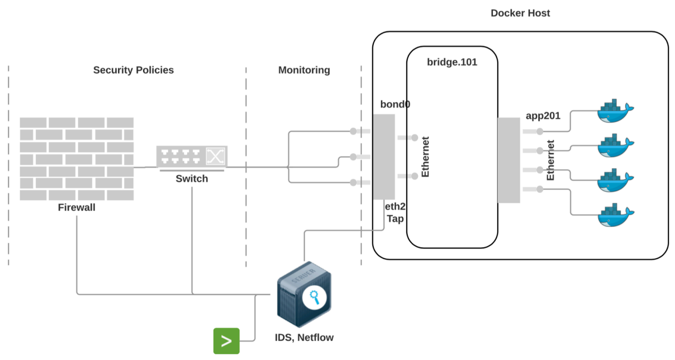


Monitoring: Network



Network traffic captured for:

- Inter-container communications
- Host communications
- Resource communications (i.e. DB, Public Internet)



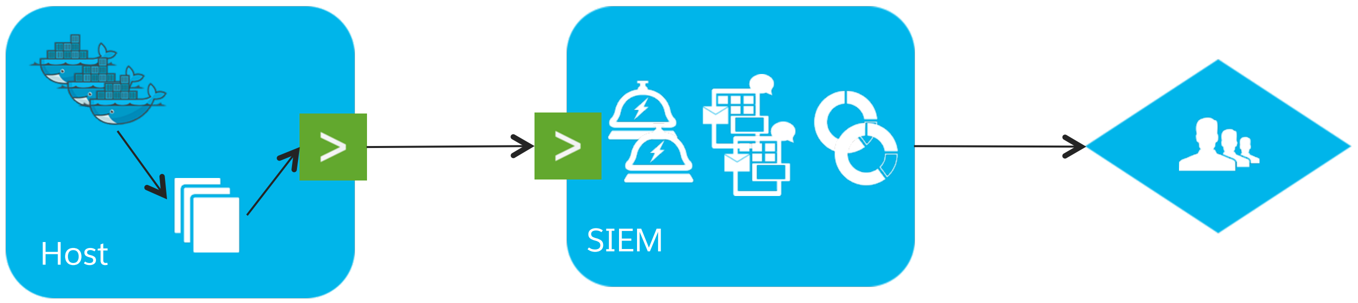
Network traffic sent to:

- IDS (Intrusion Detection System)
- Netflow generator
- Output sent to SIEM for analysis

Monitoring: Hosts

Logs:

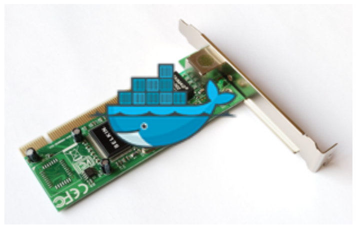
- All host logs are saved
- SIEM agents consume and forward the logs from hosts
- Monitoring, Dashboarding, Alerting at SIEM



salesforce

Monitoring: Containers & Apps

- Logs are monitored similar to host
- OS + Application logs
- Network activity monitoring
- IP address assignments
  - Netflows
  - IDS (Intrusion Detection System)
  - Raw Network Traffic Capture



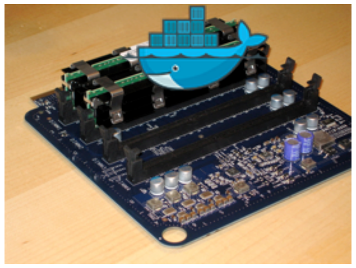
Monitoring: Host, Containers & Apps

Disk activity monitoring

- File system integrity
- Run time layer monitoring

Memory monitoring

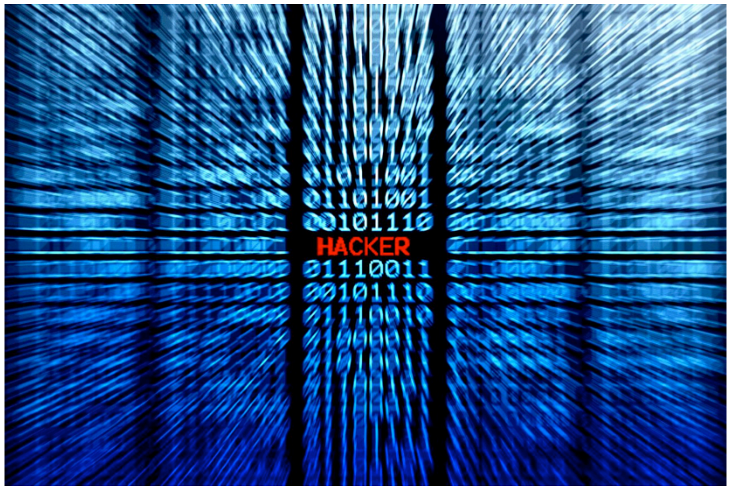
- Docker and container process activity
- Process integrity: Engine + Container



Digital Forensics

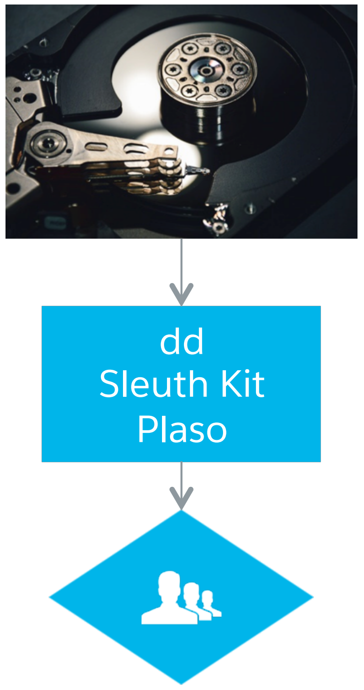
Digital Forensics

- Incident Response Plan/Policies
- Live/Post-mortem Memory Forensics
- Disk Forensics
- Network Monitoring/Forensics



Disk Forensics

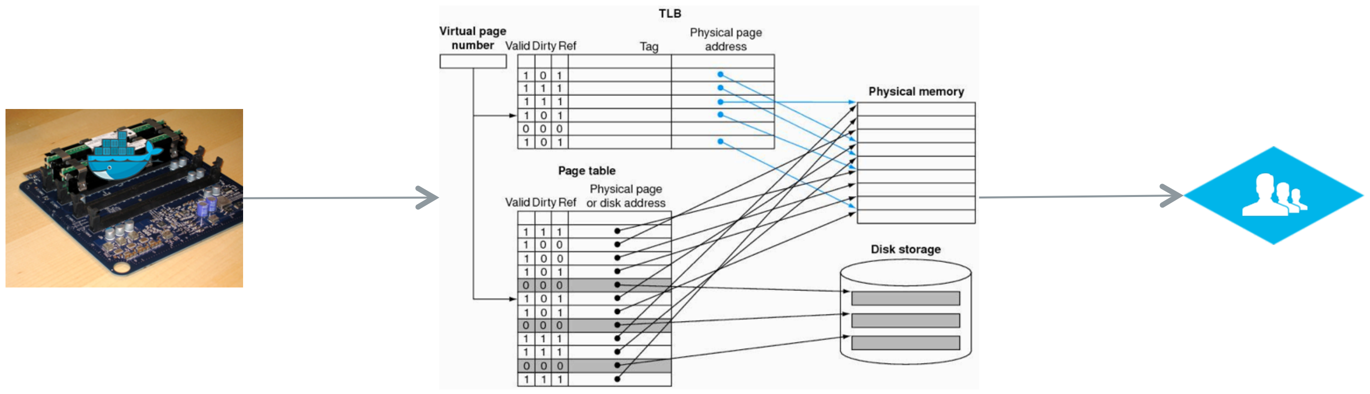
- Build supertimeline to have integrated view of events
- Data Sources:
  - Raw Disk Image
  - Log Files
  - Binaries
- Tools
  - The Sleuth Kit: File system analysis
  - Plaso: Build supertimeline
  - dd: Raw disk image



Memory Forensics

Why Memory Forensics?

- Nothing can hide in memory!
- Faster artifact discovery vs. disk forensics



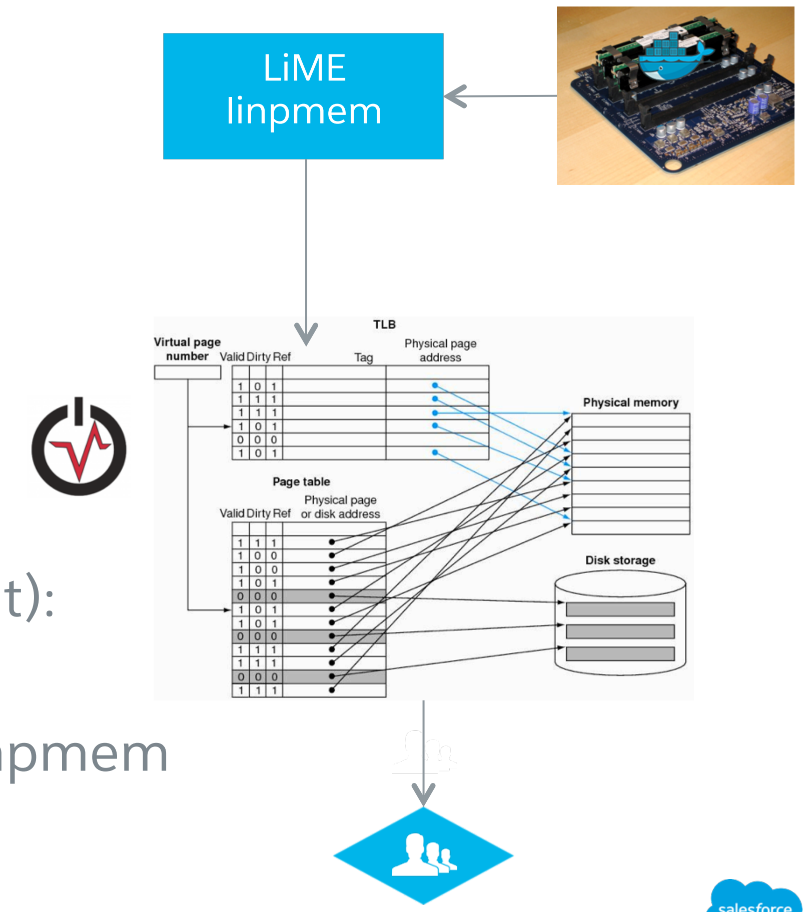
Memory Forensics

Analyze host memory

- Live /dev/*mem
- VM memory file
- Memory dump/sample

Tools:

- Analysis (most OS and sample format):
  - The Volatility Framework
- Memory sampling on Linux: LiME, linpmem



Memory Forensics: Process Hierarchy

```
.docker                 1045            1               0
.docker                 956             1               0
..docker-containe       1060            956             0
...docker-containe      8718            1060            0
...docker-containe      8713            1060            0
...docker-containe      8716            1060            0
...docker-containe      8711            1060            0
....mongod              8757            8711            999
....mongod              8723            8711            999
....mongod              8752            8711            999
....mongod              8760            8711            999
....mongod              8755            8711            999
....mongod              8763            8711            999
....mongod              8750            8711            999
....mongod              8758            8711            999
....mongod              8753            8711            999
....mongod              8761            8711            999
....mongod              8756            8711            999
....mongod              8751            8711            999
....mongod              8759            8711            999
....mongod              8754            8711            999
....mongod              8762            8711            999
....mongod              8749            8711            999
```

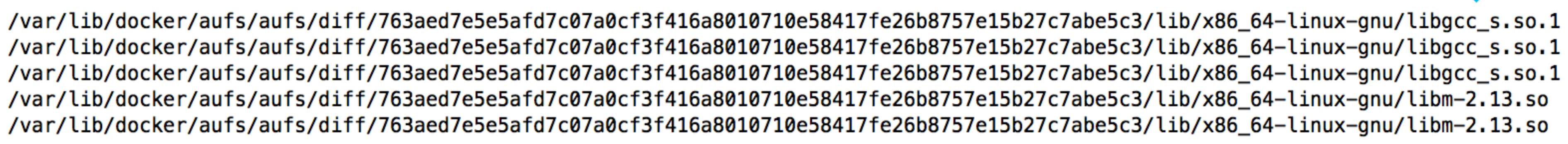
- pstree_hash [new]: View Docker processes in a tree view based on the PID hash table vs. linked list
- Use case: Detect rogue or injected child processes/containers

Memory Forensics: Temporary File Systems

```
1 -> /run
2 -> /sys/fs/cgroup
3 -> /sys/fs/cgroup
4 -> /proc/timer_stats/null
5 -> /sys/fs/cgroup
6 -> /dev
7 -> /sys/fs/cgroup
8 -> /proc/timer_stats/null
9 -> /dev/shm
10 -> /dev/shm
11 -> /run/user
12 -> /run/lock
13 -> /var/lib/docker/containers/06edc9011032ae51e2066a8fb82cc864ca4fe82f66827d0de5a060decf834359/shm
14 -> /run/shm
```

- tmpfs: lists and recovers tmpfs file systems from memory
- Use case: monitor file systems

Memory Forensics: Loaded Libraries



```
/var/lib/docker/aufs/aufs/diff/763aed7e5e5afd7c07a0cf3f416a8010710e58417fe26b8757e15b27c7abe5c3/lib/x86_64-linux-gnu/libgcc_s.so.1
/var/lib/docker/aufs/aufs/diff/763aed7e5e5afd7c07a0cf3f416a8010710e58417fe26b8757e15b27c7abe5c3/lib/x86_64-linux-gnu/libgcc_s.so.1
/var/lib/docker/aufs/aufs/diff/763aed7e5e5afd7c07a0cf3f416a8010710e58417fe26b8757e15b27c7abe5c3/lib/x86_64-linux-gnu/libgcc_s.so.1
/var/lib/docker/aufs/aufs/diff/763aed7e5e5afd7c07a0cf3f416a8010710e58417fe26b8757e15b27c7abe5c3/lib/x86_64-linux-gnu/libm-2.13.so
/var/lib/docker/aufs/aufs/diff/763aed7e5e5afd7c07a0cf3f416a8010710e58417fe26b8757e15b27c7abe5c3/lib/x86_64-linux-gnu/libm-2.13.so
```

- linux_proc_maps: shows process memory maps, their permissions and original file paths (executable and libraries)
- Use case: Detect Shared Library Injections

Memory Forensics: Process Integrity

```
Task               PID    Description                       Symbol Address
---------------- ------ --------------------------------- ------------------
- 0x0000000000400000  7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00   .ELF...........

?                   ^^

+ 0x0000000000400000  91 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00   .ELF...........

?                   ^^

docker                956 Change found at address:         0x0000000000400000
```

- process_compare [new]: Detect if user space binary has been tampered with in memory (in memory binary vs. on disk) [5]
- Works when binary symbols can’t be extracted

Summary

Platform Security	Content Security	Access Controls	Monitoring and Response
Isolation	Registry	LDAPS	IR Plan & Testing
Hardening	Notary	User Authentication	Vulnerability Management
Best Practices	Image/Code Signing	System Authentication	Network
Vulnerability Scans	Image/Code Scanning		Logs
			Forensics

thank y u

References

1. “CIS Docker 1.6 Benchmark,” Center for Internet Security
2. “Introduction to Container Security,” Docker.com
3. “Understanding and Hardening Linux Containers,” NCC Group
4. “The Volatility Framework,” https://github.com/volatilityfoundation/volatility
5. “Identifying the Unknown in User Space Memory,” Andrew White
6. “LiME,” https://github.com/504ensicsLabs/LiME
7. “linpmem,” http://www.rekall-forensic.com/docs/Tools/
8. “The Sleuth Kit,” http://www.sleuthkit.org/
9. “Plaso,” https://github.com/log2timeline/plaso