Objective

The main objective of this POC was to find the suitable tool for docker image and container scanning. The findings were based on the criteria that the tools must be open source and should be efficient enough to detect  bugs and vulnerabilities in docker images and containers that are stored and run inside our infrastructures . The overall finding result, the one that is generated from successful scanning of images and containers, would help in minimizing threats and risk in our products .

Methodology

To perform the possible POC , we need to first select a tool which would suffice our needs. For this , a number of approaches were initiated for finding the tools . With through search we landed on a number of tools but among them we found “anchor” really promising.

Anchore is an open source tool for deep image inspection and vulnerability scanning . Anchore Engine is fully-featured and flexible, and can work within a wide variety of environments and development pipelines. The open source package provides most of the features and they have enterprise version as well . The major solutions provided by anchor are :

1. CI/CD Security
2. Registry Auditing
3. Kubernetes Services
4. Container Compliance

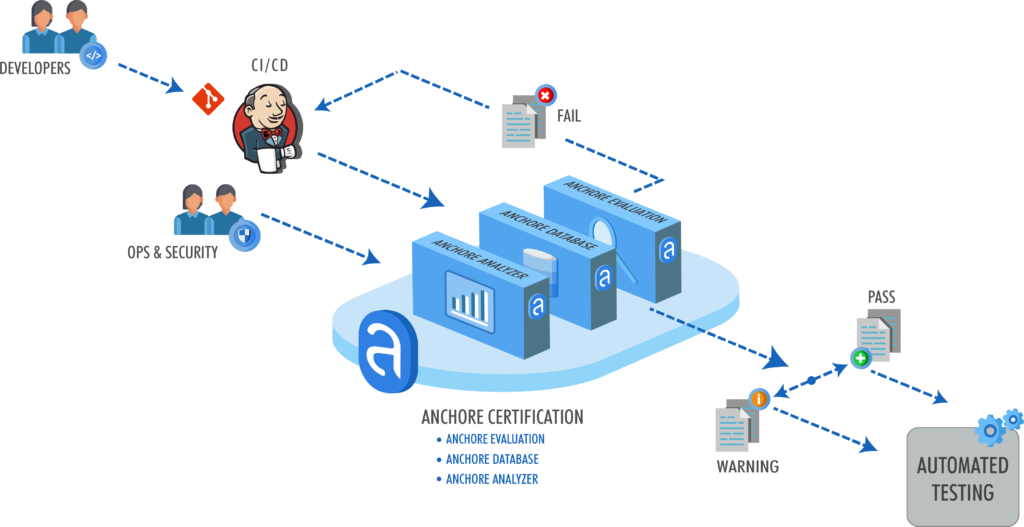
Features of Anchor Engine

IMAGE ANALYSIS

Perform deep inspection of container images, cataloging all operating system packages, files and software artifacts such as Ruby GEMs, JARs, and Node modules.

CI/CD INTEGRATION

Integrate Anchore Engine into CI/CD pipelines to ensure that builds are only successful when images meet custom security and compliance requirements. The below picture depicts the ci/cd integration process for Anchore .



CONTINUOUS MONITORING

Policies are continuously evaluated to catch issues created when images are updated, CVEs are added or removed, or new best practices are established.

Installation, Running Scanner and Report Generation

Install with Docker Compose

Requirements

The following instructions assume you are using a system running Docker v1.12 or higher, and a version of Docker Compose that supports at least v2 of the docker-compose configuration format.

* A stand-alone installation will requires at least 4GB of RAM, and enough disk space available to support the largest container images you intend to analyze (we recommend 3x largest container image size). For small images/testing (basic Linux distro images, database images, etc), between 5GB and 10GB of disk space should be sufficient.

Step 1: Setup installation location

Create a directory in which to store your configuration files.

mkdir ~/aevolume

cd ~/aevolume

Step 2: Copy configuration files

Download the latest Anchore Engine container image, which contains the necessary docker-compose.yaml and configuration files that the deployment requires.

# docker pull docker.io/anchore/anchore-engine:latest

Next, copy the included docker-compose.yaml to the ~/aevolume/ directory.

# docker create --name ae docker.io/anchore/anchore-engine:latest

# docker cp ae:/docker-compose.yaml ~/aevolume/docker-compose.yaml

# docker rm ae

Once these steps are complete, your ~/aevolume/ workspace should now look like this:

# ls ~/aevolume

docker-compose.yaml

Step 3: Download and run the containers

Download the containers listed in the docker-compose.yaml, and run the entire setup using the docker-compose CLI. NOTE: by default, all services (including a bundled DB instance) will be transient, and data will be lost if you shut down/restart

# cd ~/aevolume

# docker-compose pull

# docker-compose up -d

Step 4: Verify service availability

After a few moments (depending on system speed), your Anchore Engine services should be up and running, ready to use. You can verify the containers are running with docker-compose:

# cd ~/aevolume

# docker-compose ps

                Name                               Command State   Ports

-------------------------------------------------------------------------------------------------------

aevolume\_anchore-db\_1                   docker-entrypoint.sh postgres Up 5432/tcp

aevolume\_engine-analyzer\_1              /docker-entrypoint.sh anch ... Up 8228/tcp

aevolume\_engine-api\_1                   /docker-entrypoint.sh anch ... Up 0.0.0.0:8228->8228/tcp

aevolume\_engine-catalog\_1               /docker-entrypoint.sh anch ... Up 8228/tcp

aevolume\_engine-policy-engine\_1         /docker-entrypoint.sh anch ... Up 8228/tcp

aevolume\_engine-simpleq\_1               /docker-entrypoint.sh anch ... Up 8228/tcp

You can run a command to get the status of the Anchore Engine services:

# cd ~/aevolume

# docker-compose exec engine-api anchore-cli system status

Service policy\_engine (anchore-quickstart, http://engine-policy-engine:8228): up

Service simplequeue (anchore-quickstart, http://engine-simpleq:8228): up

Service catalog (anchore-quickstart, http://engine-catalog:8228): up

Service analyzer (anchore-quickstart, http://engine-analyzer:8228): up

Service apiext (anchore-quickstart, http://engine-api:8228): up

Engine DB Version: 0.0.10

Engine Code Version: 0.4.0

**Note:** The first time you run Anchore Engine, it will take some time (10+ minutes, depending on network speed) for the vulnerability data to get synced into the engine. For the best experience, wait until the core vulnerability data feeds have completed before proceeding. You can check the status of your feed sync using the CLI:

# cd ~/aevolume

# docker-compose exec engine-api anchore-cli system feeds list

Feed                   Group LastSync                           RecordCount

vulnerabilities        alpine:3.3 2018-06-27T17:13:53.509309Z        457

vulnerabilities        alpine:3.4 2018-06-27T17:13:59.103245Z        594

vulnerabilities        alpine:3.5 2018-06-27T17:14:05.000942Z        649

vulnerabilities        alpine:3.6 2018-06-27T17:14:10.606606Z        632

vulnerabilities        alpine:3.7 2018-06-27T17:14:17.673851Z        767

vulnerabilities        centos:5 2018-06-27T17:14:46.616051Z        1270

vulnerabilities        centos:6 2018-06-27T17:15:18.600668Z        1266

vulnerabilities        centos:7 2018-06-27T17:15:41.468527Z        657

vulnerabilities        debian:10 2018-06-27T17:18:16.960078Z        17494

vulnerabilities        debian:7 2018-06-27T17:21:20.058941Z        20455

vulnerabilities        debian:8 None                               0

vulnerabilities        debian:9 None                               0

vulnerabilities        debian:unstable None                               0

vulnerabilities        ol:5 None                               0

vulnerabilities        ol:6 None                               0

vulnerabilities        ol:7 None                               0

vulnerabilities        ubuntu:12.04 None                               0

vulnerabilities        ubuntu:12.10 None                               0

vulnerabilities        ubuntu:13.04 None                               0

vulnerabilities        ubuntu:14.04 None                               0

vulnerabilities        ubuntu:14.10 None                               0

vulnerabilities        ubuntu:15.04 None                               0

vulnerabilities        ubuntu:15.10 None                               0

vulnerabilities        ubuntu:16.04 None                               0

vulnerabilities        ubuntu:16.10 None                               0

vulnerabilities        ubuntu:17.04 None                               0

vulnerabilities        ubuntu:17.10 None                               0

vulnerabilities        ubuntu:18.04 None                               0

As soon as you see RecordCount values > 0 for all vulnerability groups, the system is fully populated and ready to present vulnerability results. Note that feed syncs are incremental, so the next time you start up Anchore Engine it will be ready immediately. The CLI tool includes a useful utility that will block until the feeds have completed a successful sync:

# docker-compose exec engine-api anchore-cli system wait

Starting checks to wait for anchore-engine to be available timeout=-1.0 interval=5.0

API availability: Checking anchore-engine URL (http://localhost:8228)...

API availability: Success.

Service availability: Checking for service set (catalog,apiext,policy\_engine,simplequeue,analyzer)...

Service availability: Success.

Feed sync: Checking sync completion for feed set (vulnerabilities)...

Feed sync: Checking sync completion for feed set (vulnerabilities)...

...

...

Feed sync: Success.

Step 5: Begin using Anchore

Start using the anchore-engine service to analyze images - a short example follows which demonstrates a basic workflow of adding a container image for analysis, waiting for the analysis to complete, then running content reports, vulnerability scans and policy evaluations against the analyzed image.

# docker-compose exec engine-api anchore-cli image add docker.io/library/debian:7

...

...

# docker-compose exec engine-api anchore-cli image wait docker.io/library/debian:7

Status: analyzing

Waiting 5.0 seconds for next retry.

Status: analyzing

Waiting 5.0 seconds for next retry.

...

...

# docker-compose exec engine-api anchore-cli image content docker.io/library/debian:7 os

Package                       Version License

apt                           0.9.7.9+deb7u7 GPLv2+

base-files                    7.1wheezy11 Unknown

debconf                       1.5.49 BSD-2-clause

...

...

# docker-compose exec engine-api anchore-cli image vuln docker.io/library/debian:7 all

Vulnerability ID        Package                 Severity Fix Vulnerability URL

CVE-2005-2541           tar-1.26+dfsg-0.1+deb7u1                 Negligible None https://security-tracker.debian.org/tracker/CVE-2005-2541

CVE-2007-5686           login-1:4.1.5.1-1+deb7u1                 Negligible None https://security-tracker.debian.org/tracker/CVE-2007-5686

CVE-2007-5686           passwd-1:4.1.5.1-1+deb7u1                Negligible None https://security-tracker.debian.org/tracker/CVE-2007-5686

CVE-2007-6755           libssl1.0.0-1.0.1t-1+deb7u4              Negligible None https://security-tracker.debian.org/tracker/CVE-2007-6755

...

...

...

# docker-compose exec engine-api anchore-cli evaluate check docker.io/library/debian:7

Image Digest: sha256:92d507d81bd3b0459b121215f6f9d8249bb154c8b65e041942745dcc6309a7b5

Full Tag: docker.io/library/debian:7

Status: pass

Last Eval: 2018-11-06T22:51:47Z

Policy ID: 2c53a13c-1765-11e8-82ef-23527761d060

Results

The results obtained from docker image and container scanner were very informative . The results thus delivered were very insightful and detailed indicating the current vulnerability and bugs information based on standard criteria .

Next step

 Next step would be to integrate the tool into our CI/CD pipeline and scan thoroughly in artifacts ( images and containers ) in our infrastructure .