anchore

WHITEPAPER

Using Anchore to Help Achieve NIST SP 800-190 Compliance

Anchore and NIST SP 800-190

Introduction

The National Institute of Standards and Technology Special Publication 800-190 was designed to explain the security concerns associated with container technologies. In addition, recommendations are made throughout the publication for addressing the outlined concerns.

Below details how Anchore can help with execution and mapping of NIST SP 800-190 Sections 4.1 - 4.5.

The below mappings are only specific to Anchore and similar technologies. Capabilities such as runtime and host security are not discussed, although they are part of the NIST SP 800-190 publication.

4.1 Image Countermeasures

4.1.1 Image Vulnerabilities

Organizations should employ the appropriate set of tooling and security checks to uncover any vulnerabilities at both the operating system and application layer.

Anchore will scan images and create a manifest of packages. From this manifest, there is the ability to run checks for image vulnerabilities. In addition,the ability to periodically check if new vulnerabilities have been published that directly impact a package contained within a relevant image manifest.

Anchore has the ability to be integrated with common CI tools (Jenkins), or in an adhoc manner from a command line. From these integrations, policy checks can be enforced to potentially fail builds.

Anchore is very much Docker v2 registry agnostic and can be integrated with an on-premise image registry, public or private Dockerhub, AWS, Azure, Artifactory, etc..

Anchore provides image checks and vulnerability info for the following base layers:

- Alpine
- Ubuntu
- CentOS
- Debian
- Red Hat Enterprise Linux

Anchore provides image checks and vulnerability info for the following third party packages contained within images:

- NodeJS
- Ruby
- Python
- Java

4.1.2 Image Configuration Defects

Policies can be created that enforce Dockerfile and image best practices. As an example, Anchore allows the ability to look for a base image to be in existence via a regex check. These regular expressions can be used to enforce policies specific to image layers, files, etc.

Anchore provides a mechanism for stopping images built from Dockerfiles that have exposed port 22. As outlined in the NIST publication, SSH and other remote administration tools that provide remote access should never be enabled within containers. In this case, an Anchore policy check can be set up to block this image if it exposes port 22.

A common security best practice is reduce attack surface areas as much as possible. In the case of Docker images, having a set list of base layers from minimalistic technologies (Alpine Linux) that are allowed use, aligns with the above best practice. An Anchore policy can be configured to check for specific base image layers via a `FROM` instruction existing in a Dockerfile.

Other use cases for Anchore policy checks:

- Dockerfiles must have `USER` instruction.
- Dockerfiles must have `HEALTHCHECK` instruction.

4.1.4 Embedded clear text secrets

Clear text secrets should never be present inside of images. They should be stored outside of the image and provided dynamically at runtime as needed.

Anchore provides mechanisms to search for secrets that may be contained within an image via a regex check (filename or content). The common ones available by default are:

- `AWS_ACCESS_KEY`
- `AWS_SECRET_KEY`
- `PRIV_KEY`
- `DOCKER_AUTH`
- `API_KEY`

4.1.5 Use of untrusted images

As outlined in the publication, organization should maintain a set of trusted images and registries. By only allowing trusted images to be used, the risk of untrusted, vulnerable images is mitigated. There are several tools that can be used to help with achieving this:

- Anchore
- Notary
- Jenkins
- Docker Registry

More details around how to setup a secure container based CI/CD Pipeline here: https://anchore.com/blog/container-security-cicd-pipeline-open-source/

4.2 Registry Countermeasures

4.2.2 Stale images in the registry

There are two ways to mitigate the risk of stale images being used:

- Organizations can prune registries that contain vulnerable images
- Best practices should be collectively agreed upon to only allow access to images that are fresh.

Within Anchore, a policy check can be set up to look for a tag of an image that has not been deemed stale. As an example, if a development team builds a `my-app:latest` image inheriting from `my-base-layer:latest` within their trusted registry that they have built, an Anchore Dockerfile check can for `my-base:latest` to be present. Along with the appropriate amount of automation and checks, the risk of using stale and older images is greatly decreased.

4.3 Orchestrator Countermeasures

This section can be achieved with Active Directory and Network tooling. No use for Anchore here.

4.4 Container Countermeasures

4.4.1 Vulnerabilities within the runtime software

While this can only partly handled by Anchore itself, it is important for organizations to shift security check as far left as possible in the development lifecycle in order to catch Common Vulnerabilities and Exposures (CVEs) prior to a container being deployed. Anchore provides policy gates and checks for 3rd party packages (NPM, GEM, JAVA, PY), in order to provide insight into any potential vulnerabilities. Setting up these checks is a good place to start.

4.4.3 Insecure container runtime configurations

Again, this is not a section that is completely handled by Anchore, as the focus itself is on runtime configurations. However, by aligning the allowable Anchore policies with CIS Docker benchmarks, this section can be considered partially compliant.

Within the CIS Docker Benchmark document the following are achievable with Anchore:

- Create a `USER` for the container.
- Use of trusted base images for containers.
- Do not install unnecessary packages in the container.
- Scan and rebuild the images to include security patches. Anchore can scan images frequently to check for vulnerabilities and trigger alerting if found. With the appropriate response automation, when vulnerabilities are found, they should be quickly patched, the images rebuilt, and new containers instantiated.
- Enable content trust. Only sign images with Notary and push them to trusted registry after they have passed Anchore policy checks.
- Add `HEALTHCHECK` instruction to the container image.
- Do not use update instruction alone in the Dockerfile.
- Use `COPY` instead of `ADD` in Dockerfile.
- Do not store secrets in Dockerfiles.
- Install verified packages only.

Some of these are redundant and will be accomplished as part of the Image Countermeasures section.

4.4.5 Rogue containers

Anchore used in conjunction with CI tooling, image signing, and the appropriate development, testing, and production pipelines/environments will help to mitigate the risk of rogue containers being used. As an example, if only specific images can be pushed to the appropriate registries via a Jenkins user, the chance of rogue containers being instantiated in a production environment by an unauthorized user is greatly reduced. Anchore policy checks as part of this pipeline can be set up to enforce baseline requirements for vulnerabilities and compliance.

4.5 Host Countermeasures

This section can be achieved with Network Tools, Active Directory, and human input/controls. No use for Anchore here.

Conclusion

The NIST SP 800-190 details the best ways to counter the security risks that are presented with container technology usage. Organizations should focus implementing the relevant sections of this document with the correct automation and tooling in order to fulfill their compliance requirements.

anchore