Cloud Native Security Map

Slack Channel: **#sig-security-whitepaper-map**

Participants:

- Brandon Lum
- Emily Fox
- Chase Pettet
- Diego Comas
- Ash Narkar
- Vinay Venkataraghavan
- John Li
- Alok Raj

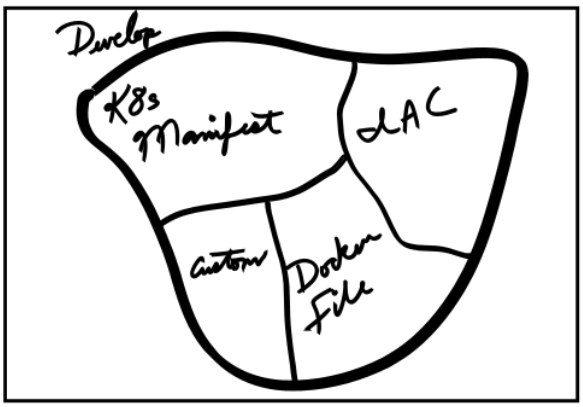
Goals and Non-goals of Cloud Native Security Map (“Landscape” v2)

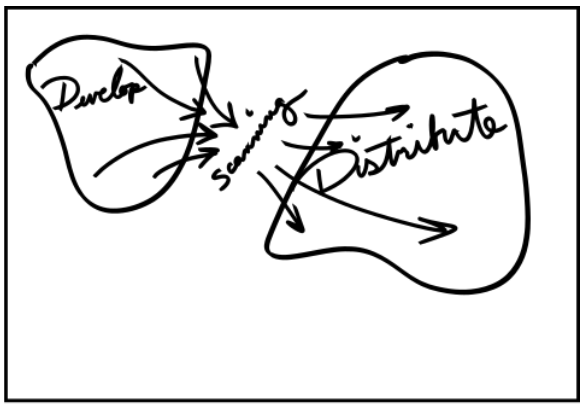
- Provide a mapping of CNCF and open source projects to areas of CN Security whitepaper
- Provide a practical viewpoint and information on topics in the CN Security whitepaper
- Identify gaps in CN Security in the ecosystem and make recommendations to TOC
- Help educate practitioners of what technologies can be used in practice and how they tie into each other
- Provide practical tips or examples for how to use tools within this category, or why they are important (I.e. example breaches, etc.)
- Provide a reference for frameworks to utilize when developing CN Security solutions and architectures.

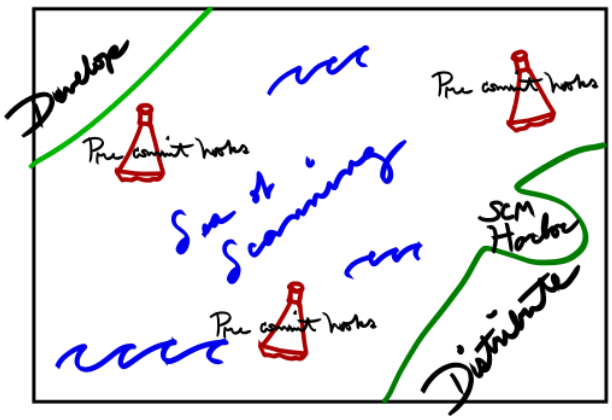
Non-goals:

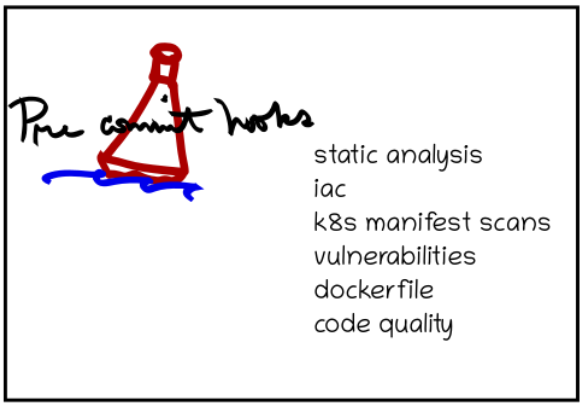
- Not an implementation guide on how to implement CN Security
    - Demonstrative, not procedural
- Not a checklist of what to do
- Not one technology focused (i.e. not taking 1 reference architecture and developing the landscape around it).

Storyboard Examples









Storyboard 2: Simple Map + Navigation Storyboard:
https://docs.google.com/presentation/d/1IzfmyJ18sF-hauE371xhHxXGoSkShDVs1DKf4i-63xc/edit

Meeting Notes and Discussions

2. Naming of SIG-Security “Landscape”

Make suggestions and bring ideas to community

Suggestions

- e.g. 1
- Cloud Native Security Panorama [Fox]
- Cloud Native Security Topography [Fox]
    - ~~Cloud Native Security Topology [?Chase]~~
- Security Lifecycle [Chase] [+1 Vinay]
- Cloud Native Integrated Security Pipeline [Chase]
- Cloud Native Security Start Left Maturity [Chase]
- Cloud Native Security Trail Map[Chase]
- Captain Planet’s Cloud Native Security Hero’s [Chase]
- The Cloud Native Security Travel Map [Ash] [+½ add Security]
- Cloud Native Security Travel Guide (based on the idea below) [Brandon]
- Cloud Native Security Compass (based on the map idea below) [Brandon]
- Guided Tour of Cloud Native Security [Vinay]
- Cloud Native Security Map [Fox]

3. Organization of SIG-Security Landscape

- Content Example from initial landscape development: https://docs.google.com/document/d/1b-sAh1DHaSjg48ww3cGISdP_uasF9t_bQEuFnWbQgXQ/edit
- Propose several ideas on what to do, individuals/groups sign up to propose ideas at next meeting
- Proposals:
    - Example A. Overview of white paper, click into different categories for details and projects. For each header/sub-header, provide relevant links to other aspects that people may wish to pursue. I.e. for signing of artifacts, relevant, would be the various aspects of signing and also linking to runtime, because there is verification enforcement. This way users can explore cloud native security in a more holistic way.
    - Example : Topographical map (similar to CNCF landscape trail with more directions, similar to game of pods (https://kodekloud.com/p/game-of-pods-game)
    - Example B.
    - If we go with the *Travel Map* theme we can highlight different phases of the Cloud Native lifecycle as the points on the map. For example, if we look at the cloud native

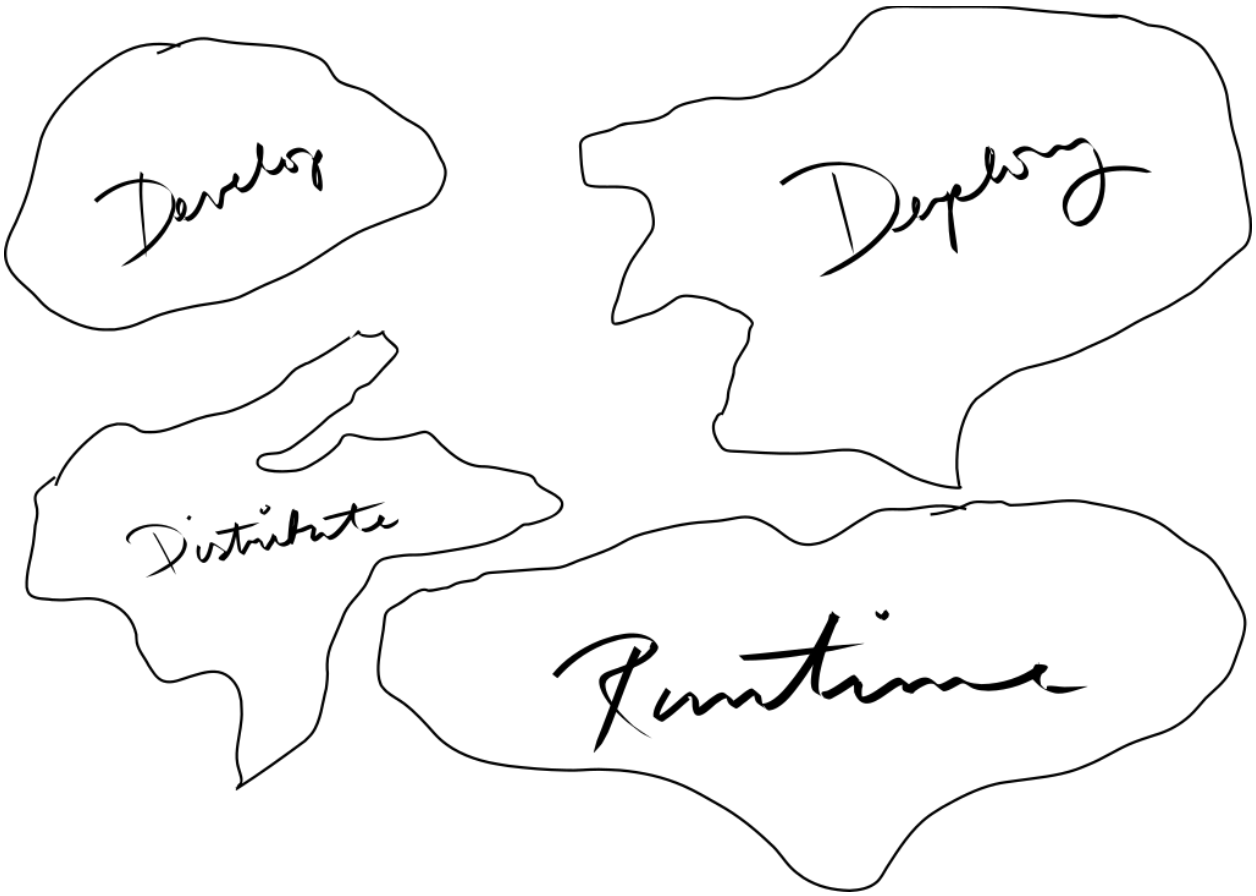
whitepaper, then “Develop”, “Distribute”, “Deploy” and “Runtime” would be the continents and within each phase we could delve deeper with details. The “journey” from one continent to another, say from the “Develop” to “Distribute” phase would be represented as a flight path. So users can start from a particular phase (*continent*) and then follow the flight path to the logical next phase. Users would initially explore different aspects (countries/cities) of a particular phase (continent) and upon exploration would then be guided to the next phase.

Users can start from any phase that best suits their needs and the map would act as a recommendation as to what can be further explored.

The different CNCF projects would be represented as countries within the continent and we could show how you go from one to another. For example, you would likely do authentication followed by authorization. So now you have a path from SPIFFE/SPIRE to OPA.

We can also convey the maturity of a project with different sized dots on the map, to give users an idea of adoption etc. So typically the graduated projects would be larger dots compared to the sandbox ones. (Ash)

- Really nice thinking, but I’m unsure of the dot sizes as a proxy for maturity. Maybe colors? Idk. I worry about larger and smaller dots being difficult to parse in this context for the semantics. (Chase)



Additional notes:

Levels of implementation/Security - guide on level 1 do ABC, level 2 do extra XYZ

4. Content of Security Landscape

- Develop and split up content creation in the future
- Smallest set to showcase flow
- Topic overview + related projects

Develop

- ...
- K8s Manifest Scanning [Ash]

    - Organizations are increasingly adopting the “Shift Security Left” philosophy which involves working with development and compliance teams to detect security issues during early development phases thereby reducing security risks towards the end of the delivery pipeline.
      Kubernetes manifests allows users to define the resources (eg. Deployments, Service) for their application/service. These resources are then created by sending a request to the Kubernetes API server. Kubernetes provides a mechanism called Admission Control that can be leveraged to control the desired-state of the cluster. It could be beneficial if we could enforce custom security policies much earlier in the development pipeline, integrate these checks with existing CI frameworks and report any issues even before deploying these resources to the cluster. Hence scanning K8s manifests as part of the development pipeline will not only help uncover issues earlier but can also potentially reduce the application’s attack surface. Some of the security policies that could be enforced are:
        - Prohibit container images that use the “latest” tag
        - Prohibit container images from specified registries
        - Prevent all containers from running as “root”
        - Ensure all containers specify cpu and memory limits
        - Require all ingresses to have TLS configured
        - Prevent services from creating external load balancers

      Related projects:
        - Conftest / OPA
- Code Review
- Dockerfile Scanning [Diego]
    - Identifying weaknesses as early as possible is a great benefit to integrate security early and avoid tedious and expensive changes once the problem becomes a vulnerability in production.
      Putting aside the application code and dependencies, in Docker containers the earliest point is the source of creation - the Dockerfile.
      There are tools available to identify problems in the content of a Dockerfile, starting from essential things like avoiding to set the user as root.

      Here are some other checks that will help improve the security of the container created using the Dockerfile:
        - Do not run as root or a sudoer user
        - Do not use latest tag for base images and pin to specific versions
        - Do not include ADD to fetch external resources
        - Do not include curl bashing
        - Do not store secrets

      You can create checks in your CI pipeline to scan these potential weaknesses and you can even enable a git-hook with something like Open Policy Agent Conftest to block those issues very early.

Related projects:

- Conftest : https://github.com/open-policy-agent/conftest/
- Hadolint : https://github.com/hadolint/hadolint

- ...
- Pre-commit hooks [Vinay]
  - Security Scans of Infrastructure as Code, Kubernetes manifests, software packages for vulnerabilities

    DevOps processes need to ensure that security is injected early and into existing tools and processes. One of the most fundamental processes executed by DevOps teams is that of making a Pull Request to check in new or updated software or infrastructure artifacts. Consequently, it is now possible and a best practice to ensure that all these artifacts (software libraries and packages, IaC and K8's manifests) are scanned when the Pull Request is made. By executing security scans at this phase, DevOps teams have full and early visibility into the presence of vulnerabilities in software libraries prior to merging the code with the main line branch. Additionally, the scanning of IaC and K8's manifests identifies security and compliance violations at this stage. Both types of failures are detected and the Pull Request is rejected due to these violations, which need to be addressed by the DevOps teams.

    Technologies: Anchore
  - …..
- Static Analysis tools[Diego]
  - Static analysis should be leveraged to identify vulnerabilities and weaknesses in software code, application manifests and IaC. This can be tested during the development stage of the SDLC. There are multiple benefits of doing static analysis testing like reducing the cost to fix a vulnerability as the weaknesses are found earlier in the SDLC.

    There are many commercial and open source tools that can be used for static analysis and these are a few examples :

    OSS tool for programming language :

    - **Python** : bandit
    - **Javascript** : eslint
    - **Go**: gosec
    - **NodeJS**: nodejs-scan
    - **PHP**: phpcs-security-audit
    - **.NET** : security-code-scan

    OSS tool for IaC:

        - **Terraform** : checkov, terrascan, tfsec, regula, terraform-compliance, conftest

    OSS for application manifests:
        - **Kubernetes** : Kubesec , Kubeaudit

- ...
- ...
- ...

Distribute

- ...
- ...
- **Image Scanning [Vinay]**
    - Scanning of images for vulnerabilities forms a key component of the Shift Left paradigm that is being adopted by cloud native DevOps teams. Scanning container images allows both Vulnerability Management teams as well as DevOps teams detailed visibility into the vulnerabilities and malware that may be present as part of their application code. These container images can be scanned either at the image build time or as they are resident in container registries such as Docker Hub.
    - Additionally, scanning container images provides visibility into vulnerabilities present in different layers that comprise the container image. The data obtained from scanning container images can be used by both DevOps and Security teams. The DevOps teams use the data to make the necessary fixes and updates to packages as necessary. The security teams on the other hand use the data to have full visibility into the vulnerability posture, build a risk model, which are then used to create image governance policies to prevent the deployment of risky or insecure images into the runtime environment.
    - Technologies: Clair, Harbor
- ...
- Artifacts & Images -Brandon
    - Registry Staging

      In order for artifacts to be usable and deployed, they need to be made available in a repository, these are usually done via registries. Registries store both the content of images as well as its metadata. It is important to configure the registries so that artifacts have high availability.

      Technologies: Docker Distribution, Quay

- Signing, Trust, and Integrity

  To preserve the integrity of artifacts, the contents and its metadata should be signed. The information to be signed includes the content of the image to prevent tampering, as well as signing of metadata to provide provenance of the artifact

  Technologies: Docker Content Trust, Red Hat Simple Signing
  Adjacent Technologies/Projects: skopeo, buildah, crio, docker distribution

Runtime Environment

- Compute
  - Orchestration [maybe emily…] container orchestration capabilities enable organization to deploy applications securely at the velocity their teams need.
    - Security Checks
      - Scan application manifests
      - https://www.cisecurity.org/benchmark/Kubernetes/
      - ~~kubescan~~
    - Threat Models & Matrices
      - Microsoft threat matrix based on MITRE ATT&CK framework for Kubernetes,
      - The Kubernetes Security Audit Working Group Security Threat Model
    - Attack Trees
      - The CNCF Financial User Group Kubernetes Attack Trees
    - Policies
      - Kyverno
    - Tech
      - K8s
        - Kubernetes is a robust container orchestration platform that provides teams with multiple areas of configuration. It is both portable and extensible, allow organizations to customize their deployments
      - Crossplane
        - Crossplane is a Kubernetes add-on that enables you to provision and manage infrastructure, services, and applications from kubectl.
      - Volcano

        - Volcano enables organizations to run high-performance workloads on Kubernetes. It features powerful batch scheduling capability required by many classes of high-performance workloads, such as machine learning/deep learning, bioinformatics/genomics, and other big data applications
    - K3s
        - A production ready, easy to install, half the memory, lightweight kubernetes. All in a binary less than 100 MB.

- Containers
    - Runtime [Brandon]
    - ...
    - Image Trust & Content Protection -Brandon

      To ensure the confidentiality and integrity of the container images, verification of image signatures and image decryption is performed on the runtime. Container image signature verification ensures the integrity and provenance of the image, providing observability and assurance of images running on the cluster.

      Related Technologies/Projects: Docker Content Trust, Red Hat Simple Signing, Portieris

      Adjacent Technologies/Projects: skopeo, buildah, crio, docker distribution

      Container image decryption ensures that only authorized clusters with access to the decryption keys are allowed to use an image - and the image remains confidential otherwise.

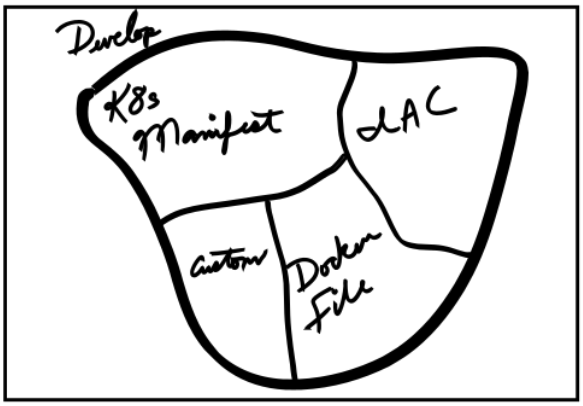
      Related Technologies/Projects: ocicrypt

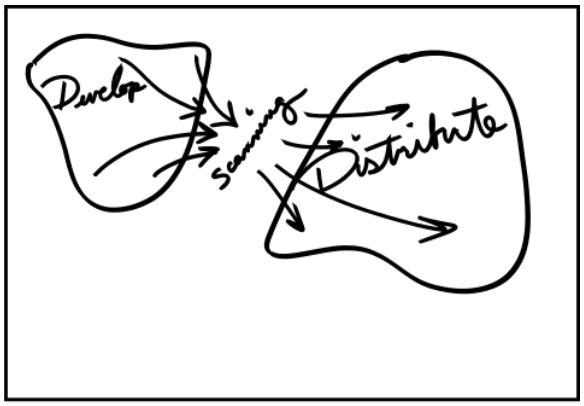
      Adjacent Technologies/Projects:crio, docker distribution, containerd

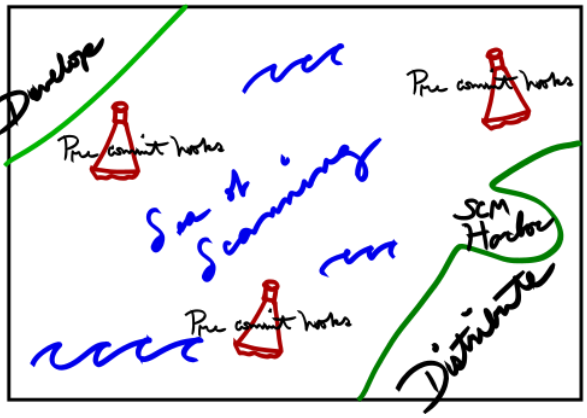
Links at top level Develop <> Distribute [Vinay]

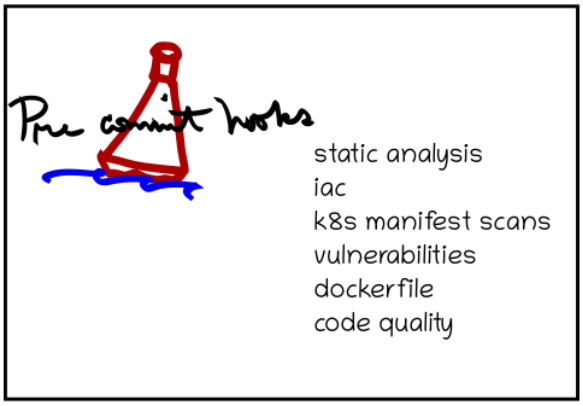
The Develop phase involved the security scanning of artifacts such as Infrastructure as Code and Kubernetes templates in a manner that is integrated into the IDE's and at the Pull Request phase. The next phase which is the Distribute phase is responsible for building the VM, Container or Serverless artifact called the image. The container images are built and pushed into a container registry. It is in this phase that the container images are scanned to obtain a

complete manifest of the vulnerabilities and malware that could be present in open source software packages, modules and libraries. Additionally, the IaC and Kubernetes manifests can be scanned at the distribute phase as well to detect security and compliance violations. Once all the security and compliance tests have been successfully executed does the next phase get triggered.









Links at 2nd level Image Scanning <> Artifacts & Images - Brandon

After images are scanned for vulnerabilities and compliance checks, they should then be prepared for distribution, all images, artifacts, along with their metadata has to be made available in registries. This brings us to preparing and securing the artifacts for distribution, ensuring the confidentiality, integrity and availability of the images and artifacts.

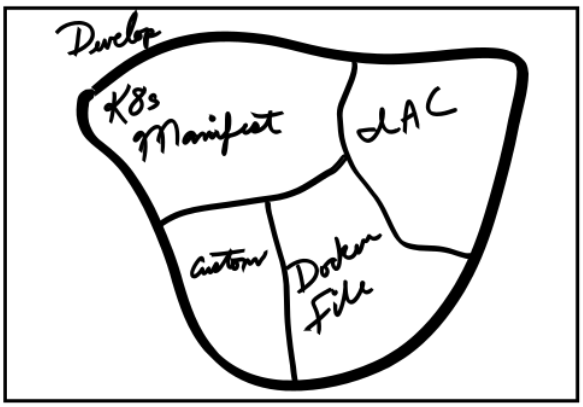
Links between Artifacts & Images and Container runtime on image trust and content protection - Brandon

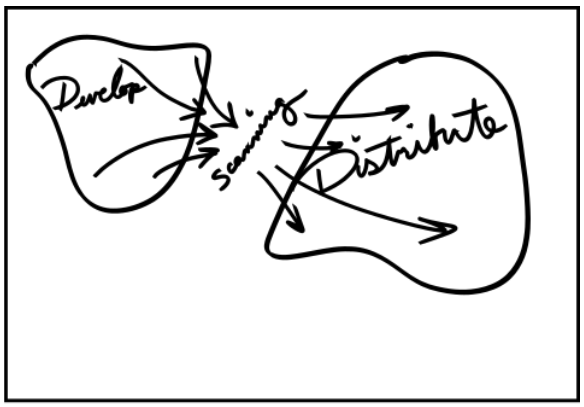
Protecting images and artifacts by means of signing and encryption provide image security, but just signing and encrypting them does not increase security if the consumers of these artifacts do not enforce the right controls. It is equally important to have their counterparts, verification and decryption.

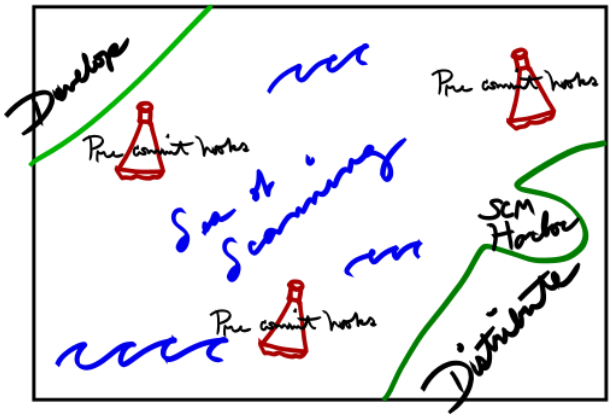
Links between Artifacts & Images and Key management - Brandon

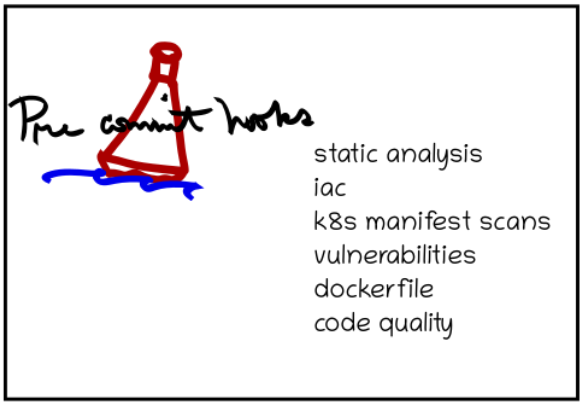
Protecting images and artifacts by means of signing and encryption involve cryptography, which is tied to key material. To ensure that controls are enforcing them, key management is as important to ensuring security of the artifacts. Mismanaged or unprotected keys lead to degredation of these security controls.

Storyboard Examples









Simple Map + Navigation

https://docs.google.com/presentation/d/1IzfmyJ18sF-hauE371xhHxXGoSkShDVs1DKf4i-63xc/edit

Content Templates (Example: Application Manifests)

Motivation (1-2 sentences)

Organizations are increasingly adopting the “Shift Security Left” philosophy which involves working with development and compliance teams to detect security issues during early development phases thereby reducing security risks towards the end of the delivery pipeline.

Threats and incidents (optional: if available)

Example catalog of incidents like the supply chain catalog

Description (3-5 sentences)

Application manifests allows users to define the resources (eg. Deployments, Service) for their application/service. These resources are then created by sending a request to the cluster orchestrator. It could be beneficial if we could enforce custom security policies much earlier in the development pipeline, integrate these checks with existing CI frameworks and report any issues even before deploying these resources to the cluster. Hence scanning application manifests as part of the development pipeline will not only help uncover issues earlier but can also potentially reduce the application’s attack surface.

Recommendations (Expect to be 1-5 bullets)

- Scan application manifests for insecure configurations
- Block insecure applications and provide remediation steps

Examples (instances of recommendations)

Example of applications manifests are Kubernetes YAML files, that are sent to the Kubernetes API server to deploy an application.

Some of the security policies that could be enforced on Kubernetes YAML files are:

- Prohibit container images that use the “latest” tag
- Prohibit container images from specified registries
- Prevent all containers from running as “root”
- Ensure all containers specify cpu and memory limits
- Require all ingresses to have TLS configured
- Prevent services from creating external load balancers

Projects (Just link to project)

- Conftest

- OPA

References (optional: related documents, blogs, etc.):

- NIST controls/docs
- Medium blogs

Template Links (Q: How deep layers? Just one level? More? Organization vs navigability)

Proposition: Links between top layers and one layer below, the rest are just island of things, and links between things that are relevant.

Links at top level Develop <> Distribute

The Develop phase involves the security scanning of artifacts such as Infrastructure as Code and Kubernetes templates in a manner that is integrated into the IDE's and at the Pull Request phase. The next phase which is the Distribute phase is responsible for building the VM, Container or Serverless artifact called the image. The container images are built and pushed into a container registry. It is in this phase that the container images are scanned to obtain a complete manifest of the vulnerabilities and malware that could be present in open source software packages, modules and libraries. Additionally, the IaC and Kubernetes manifests can be scanned at the distribute phase as well to detect security and compliance violations. Once all the security and compliance tests have been successfully executed does the next phase get triggered.

Links at 2nd level Image Scanning <> Artifacts & Images (Between continents)

After images are scanned for vulnerabilities and compliance checks, they should then be prepared for distribution, all images, artifacts, along with their metadata has to be made available in registries. This brings us to preparing and securing the artifacts for distribution, ensuring the confidentiality, integrity and availability of the images and artifacts.

Links between Artifacts & Images and Container runtime on image trust and content protection (across continents Develop <> Runtime)

Protecting images and artifacts by means of signing and encryption provide image security, but just signing and encrypting them does not increase security if the consumers of these artifacts do not enforce the right controls. It is equally important to have their counterparts, verification and decryption.

Links between Artifacts & Images and Key management (across continents Develop <> Runtime)

Protecting images and artifacts by means of signing and encryption involve cryptography, which is tied to key material. To ensure that controls are enforcing them, key management is as important to ensuring security of the artifacts. Mismanaged or unprotected keys lead to degradation of these security controls.