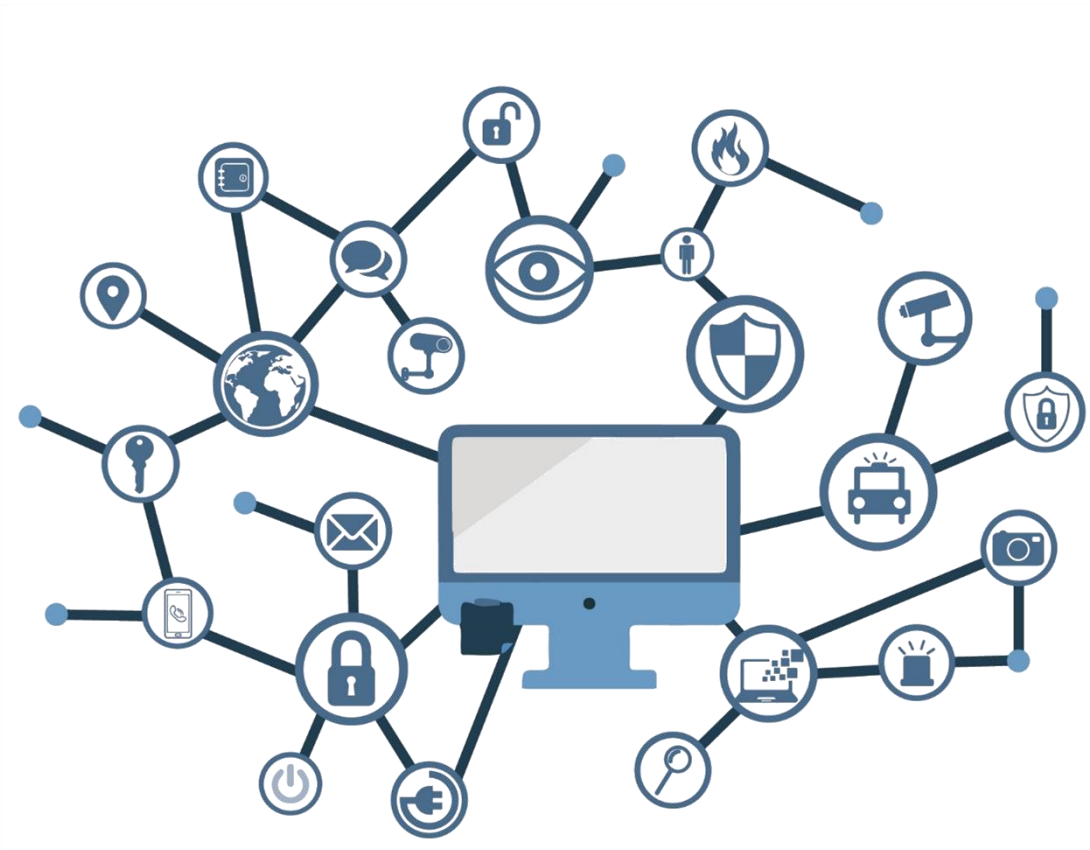
CCN
centro criptológico nacional

Guía de Seguridad de las TIC
CCN-STIC 652A

SEGURIDAD EN CONTENEDORES



Julio 2020

MINISTERIO DE DEFENSA

Edita:

2.5.4.13=Qualified Certificate: AAPP-SEP-M-SW-KPSC, ou=sello electrónico, serialNumber=S2800155J, o=CENTRO CRIPTOLOGICO NACIONAL, c=ES
2020.07.09 17:33:07 +02'00'

NIPO: 083-20-172-5

Fecha de Edición: julio de 2020
Palo Alto ha participado en la realización y modificación del presente documento.

PRÓLOGO

En un mundo cada vez más complejo y globalizado, en el que las tecnologías de la información y la comunicación (TIC) desempeñan un papel de suma importancia, hemos de ser conscientes de que la gestión adecuada de la ciberseguridad constituye un reto colectivo al que necesariamente hemos de enfrentar. Resulta necesario garantizar la protección de la capacidad económica, tecnológica y política de nuestro país, máxime cuando la proliferación de ataques dirigidos y el robo de información sensible representan una realidad incontestable.

Por ello, resulta imprescindible estar al día de las amenazas y vulnerabilidades asociadas al uso de las nuevas tecnologías. El conocimiento de los riesgos que se ciernen sobre el ciberespacio ha de servir para implementar con garantías las medidas, tanto procedimentales como técnicas y organizativas, que permitan un entorno seguro y confiable.

La Ley 11/2002, de 6 de mayo, reguladora del Centro Nacional de Inteligencia (CNI), encomienda al Centro Nacional de Inteligencia el ejercicio de las funciones relativas a la seguridad de las tecnologías de la información y de protección de la información clasificada, a la vez que confiere a su Secretario de Estado Director la responsabilidad de dirigir el Centro Criptológico Nacional (CCN).

Partiendo del conocimiento y la experiencia del CNI sobre amenazas y vulnerabilidades en materia de riesgos emergentes, el Centro realiza, a través del Centro Criptológico Nacional, regulado por el Real Decreto 421/2004, de 12 de marzo, diversas actividades directamente relacionadas con la seguridad de las TIC, orientadas a la formación de personal experto, al empleo de tecnologías de seguridad adecuadas y a la aplicación de políticas y procedimientos de seguridad.

Precisamente, esta serie de documentos CCN-STIC es un claro reflejo de la labor que este organismo lleva a cabo en materia de implementación de seguridad, permitiendo la aplicación de políticas y procedimientos, pues las guías han sido elaboradas con un claro objetivo: mejorar el grado de ciberseguridad de las organizaciones, conscientes de la importancia que tiene el establecimiento de un marco de referencia en esta materia que sirva de apoyo para que el personal de la Administración lleve a cabo la difícil tarea de proporcionar seguridad a los sistemas de las TIC bajo su responsabilidad.

Con esta serie de documentos, el Centro Criptológico Nacional, en cumplimiento de sus cometidos y de lo reflejado en el Real Decreto 3/2010 por el que se regula el Esquema Nacional en el ámbito de la Administración electrónica, contribuye a mejorar la ciberseguridad española y mantener las infraestructuras y los sistemas de información de todas las administraciones públicas con unos niveles óptimos de seguridad. Todo ello, con el fin de generar confianza y garantías en el uso de estas tecnologías, protegiendo la confidencialidad de los datos y garantizando su autenticidad, integridad y disponibilidad.

Julio de 2020

Paz Esteban López
Secretaria de Estado
Directora del Centro Criptológico Nacional

ÍNDICE

1. INTRODUCCIÓN

Las herramientas y metodologías de seguridad tradicionales no sirven para proteger las aplicaciones nativas en la nube, que ahora están vinculadas al trabajo de los desarrolladores y a entornos de varias nubes que ya no dependen de infraestructuras concretas. El cambio obedece a varios motivos:

- Los desarrolladores y los equipos de DevOps desempeñan un papel fundamental en el diseño y la implementación de aplicaciones nativas en la nube, y a menudo actúan al margen de los equipos de seguridad y tecnología tradicionales. En este contexto, la seguridad debe integrarse con una infraestructura y unas herramientas que van cambiando según lo decidan los desarrolladores.

- Las organizaciones tienen más opciones de computación que no dudan en aprovechar. Pueden optar, por ejemplo, por implementaciones de varias nubes o de nube híbrida donde los hosts (máquinas virtuales) coexisten con contenedores, Kubernetes®, contenedores como servicio (CaaS, por sus siglas en inglés) y funciones sin servidor (serverless)

- Los entornos nativos en la nube están sujetos a cambios constantes y de gran magnitud. Para los equipos de seguridad, la automatización es imprescindible para proteger los microservicios que utiliza una organización, que se multiplican y varían con el tiempo.

La plataforma de seguridad para las cargas nativas en nube deberá proteger cualquier recurso, sean hosts, máquinas virtuales, contenedores e implementaciones sin servidor de forma integral, en cualquier nube y a lo largo de todo el ciclo de desarrollo (no solo en la ejecución -run-, sino también en la elaboración -build- y en la distribución -ship-).

2. RECOMENDACIONES GENERALES

Existen distintos esfuerzos por parte de organizaciones y analistas orientados a la estandarización y creación de modelos de referencia para la seguridad en contenedores. En el año 2017, el NIST (National Institute of Standards and Technology) realizó una publicación especial (SP 800-190) orientada a la securización de containers y a los elementos de dicho ecosistema.

A lo largo de dicha guía se identifican los riesgos de seguridad asociados al uso de containers, y también se proponen contramedidas para reducir la exposición y proteger las aplicaciones contenerizadas. El NIST identifica una necesidad clara de abordar la seguridad en containers, sustentada en dos premisas:

- Existen soluciones de seguridad open-source puntuales y dispersas.
- Las soluciones de seguridad tradicionales no alcanzan a cubrir el entorno único de containers.

La publicación especial NIST SP 800-190 es especialmente relevante para empresas públicas y agencias gubernamentales, siendo de obligado cumplimiento en países como USA. Muchas empresas privadas también siguen sus recomendaciones en un intento de mejorar su postura de seguridad y reducir la exposición de los entornos cloud-nativos.

Encontramos retos de seguridad únicos, asociados al uso de containers en el ámbito de la empresa pública. La anatomía de un container favorece tremendamente la automatización; Docker por ejemplo identifica cómo se empaqueta, cómo se almacenan y despliegan aplicaciones. Debido a que los containers encapsulan todas sus dependencias, se pueden mover muy fácilmente desde un entorno de desarrollo, a test y de ahí a producción. En esta línea, empresas como Google lanzan 2 billones de containers por semana en su infraestructura. Esto plantea retos de escalabilidad, y las agencias y gobiernos deben pensar en qué controles pueden establecer para securizar despliegues masivos. La aproximación tradicional de crear y mantener controles o políticas de seguridad estáticas para cada entorno, deja de ser válida.

El cambio se ha convertido en una constante. Las organizaciones han cambiado sus ciclos de desarrollo y han pasado de una release de software al trimestre, a varias releases por semana. Con cada nueva versión las aplicaciones pueden variar, aunque ligeramente, y por tanto es necesario aplicar controles de seguridad continuos y automatizados.

Otro de los retos que se plantean es la necesidad de pensar y adoptar la seguridad por parte de equipos que tradicionalmente no la han considerado una prioridad. La seguridad debe comenzar desde el desarrollo de software y, por tanto, cuando las agencias y empresas públicas definan su modelo DevSecOps deben pensar en inyectar controles de seguridad no intrusivos en cada una de las etapas de los pipelines. Como recomendación general, estos controles deberán permitir alertar, dando visibilidad de los riesgos de seguridad a los equipos involucrados, pero también deberán permitir bloquear un despliegue si no se cumplen las políticas de seguridad fijadas para un entorno concreto.

Intentando resumir los principales requisitos que una solución de seguridad de containers debería ofrecer encontramos:

Modelo de Prestación	Se recomienda que la solución pueda ser ofrecida en modelo autogestionado, pero también como una suscripción en formato SaaS para que las empresas y agencias públicas

	puedan elegir aquel modelo que mejor esté alineado con sus necesidades.
Arquitectura	Basada en el despliegue de agentes, y consola centralizada. Se recomienda que las organizaciones busquen la mayor simplicidad y eviten despliegues basados en agentes por cada imagen (alterar imágenes) o que inyecten binarios en containers en ejecución, al igual que aquellas arquitecturas basadas en side-cars. Se recomienda por tanto que el agente se despliegue a nivel de host (o nodo en Kubernetes) y se pueda desplegar como DeamonSet o similar en entornos de Kubernetes. La solución elegida deberá ofrecer una variedad de agentes dependiendo el tipo de carga de trabajo a securizar (máquinas virtuales, containers, containers como servicio, funciones como servicio, etc.)
Soporte Multi Cloud	La solución elegida deberá proporcionar soporte para despliegues de containers on-premise pero también en entornos de nube pública, soportando despliegues de containers sobre IaaS en los principales proveedores (AWS, Azure, Google, Alibaba, Oracle, IBM, etc.). Adicionalmente, la solución deberá soportar despliegues de containers en servicios gestionados de containers o Kubernetes como Openshift, GKE, AKS, EKS, ECS, Fargate, Anthos, etc.
Soporte de Sistemas Operativos	La solución elegida deberá soportar los principales sistemas operativos, como CentOS, Debian, EulerOS, RedHat Enterprise Linux, Ubuntu, Windows Server, VMWare, Amazon Linux, GCOOS, etc.
Orquestadores	La solución elegida deberá tener un amplio soporte de orquestadores, como Kubernetes, DC/OS, Docker Swarm, Openshift, PCF PAS.
Container Runtimes	Se deben soportar los principales runtimes como Docker, CRI-O, cri-containerd.
Protección de las capas de imágenes	La solución deberá ofrecer una protección de imágenes en cada una de sus capas, identificando vulnerabilidades y fallos de cumplimiento. Se deberán soportar imágenes basadas en Alpine, Amazon Linux, Amazon Linux 2, BusyBox, CentOS, Debian, Red Hat Enterprise Linux, SUSE, Ubuntu, Windows Server

Serverless	La solución debe ser extensible hacia protección de funciones serverless, soportando runtimes como Node.js, Python, C#, Java, etc.
Casos de Uso / Ámbitos de Acción	La solución deberá incluir funcionalidades alrededor de: Gestión de vulnerabilidades a lo largo de todo el ciclo de vida Gestión de compliance a lo largo de todo el ciclo de vida Seguridad en runtime Firewall de red nativo en nube con capacidades de microsegmentación. Firewall de aplicación (WAF) nativo en nube Integración con herramientas de CI/CD Integración con registros de imágenes, públicos y privados Control de Acceso
Automatización de Infraestructura	La solución deberá soportar chequeos de templates de IaC tales como Terraform, CloudFormation templates, Kubernetes yaml, etc.

3. ARQUITECTURA

La solución de protección ha de proteger todos sus activos de la nube independiente del lugar donde estos estén ejecutándose, nube pública, nube privada, híbrida o entornos aislados,independientemente del objetivo a proteger sean contenedores, funciones serverless, hosts físicos o virtualizados o cualquier combinación de todos ellos. Además ha de contar con Threat Prevention y AI para poder proteger estos entornos y realizar un análisis del comportamiento de los mismos.

La arquitectura de la solución ha de contar con dos partes diferenciadas, un plano de gestión y una Agente. En el plano de gestión es donde estará la consola de la herramienta, desde esta consola se podrá monitorizar y controlar el entorno de contenedores y donde se configuraran las reglas de protección de los propios contenedores. Esta misma, deberá de ofrecer la posibilidad de poder ser desplegada tanto en un entorno SaaS de nube pública como en un entorno On-Premise. El Agente, será la parte operativa de la solución y deberá tener la capacidad de integrarse con los principales entornos de Contenedores, On premise o Cloud, funciones serverless y Host físicos y Virtuales

La solución deberá de ser Agentless, es decir no tendrá la necesidad de desplegar ningún tipo de agente en los hosts, contenedores o aplicaciones. Además no podrá instalarse como módulo en el kernel de los contenedores, para evitar tocar el propio kernel y crear dependencias dentro de él. Se deberá de desplegar de la forma más agnóstica posible dentro del entorno, como un contenedor más dentro del grupo de

contenedores o como un componente más, dentro de la función serverless u un servicio dentro de un Host.

Tras desplegarse la solución esta deberá ser inmediatamente funcional y pudiendo proteger el entorno donde esté directamente. Además, deberá descubrir todos los contenedores en el entorno donde esta se despliegue, están protegidos por la herramienta o no y si no lo estuvieran se podrán añadir de manera muy sencilla.

3.1 CONSOLA

Será la consola de gestión de la plataforma, es decir la GUI de la herramienta, que permitirá crear reglas de seguridad, configurar y controlar el despliegue de la herramienta y donde se podrá monitorizar desde una perspectiva de seguridad todo el entorno de Contenedores. Esta consola además proveerá una API, para poder dar acceso a los clientes que quieran controlar la herramienta desde sus propias consolas e integraciones con productos de terceros.

La consola deberá poder desplegarse como modelo SaaS, es decir en una nube privada provista por el proveedor de la solución o en el propio entorno privado del cliente, es decir en la propia infraestructura del cliente

La consola de la solución deberá tener la mismas capacidades y usar el mismo software indistintamente del modo de cómo esté desplegada, SaaS o en la infraestructura del cliente, dando de este modo la capacidad de elegir al cliente libremente el modo de despliegue sin perder ninguna funcionalidad

El acceso a la consola deberá ser de modo seguro, siendo usado el protocolo seguro HTTPS, cifrando las comunicacio nes entre el administrador y la consola evitando que puedan ser monitorizadas por terceros. También se podrá facilitar el acceso a la consola mediante el protocolo HTTP, aunque este no se recomienda su uso, debido a la falta de cifrado.

3.2 AGENTE

El Agente, será la pieza de software que ejecutará las reglas que se han configurado desde la consola de gestión, monitorizará el tráfico y el software desplegado en nuestro entorno Cloud. Se instalará en los hosts o clusters de nuestra entorno y podrá desplegarse en los distintos lugares de nuestra Cloud

El Agente podrá desplegarse en los siguientes entornos adaptados a cada uno de ellos y se podrán gestionar desde la misma consola aplicando reglas comunes y particulares para cada entorno.

- **Contenedores:** Se desplegará como un contenedor en cada Host donde se estén ejecutando contenedores, que se quieran proteger. Podrá desplegarse de dos maneras:
    - **Manualmente:** es decir, cada vez que se añada un nuevo host de contenedores, se deberá de desplegar manualmente este Agente dentro del nuevo host.
    - **Automáticamente:** es decir, podrá formar parte de un sistema orquestador de contenedores, como Kubernetes, OpenSwift o Docker Swarm y cada vez que se añada un nuevo Host automáticamente desplegará el Agente. Esta automatización también se aplicará en el caso que el escalado del entorno sea sacando host del cluster.
- **Host:** Cuando se quiera desplegar en un Host o en una Máquina Virtual este Agente se desplegará como un servicio dentro del Sistema Operativo. Se podra desplegar en dos variantes:
    - Systemd Service, en Hosts donde se esté usando un Sistema Operativo basado en Linux
    - Windows Service, para Hosts donde se esté ejecutando un Sistema Operativo de Microsoft
- **CaaS:** Se desplegará un contenedor dentro de la infraestructura del proveedor de CaaS y se encargará de securizar los contenedores que se encuentren dentro de ese servicio
- **Serverless:** Se incluirá como parte del código dentro de la función programada.

Por defecto entre el plano de control y el Agente se establecerá una conexión TCP cifrada y segura utilizando TLS, garantizando de esta manera que no se pueda interceptar o manipular dicha sesión.

4. SOPORTE DE PLATAFORMAS

La solución deberá de soportar la protección en distintos entornos, como puede ser Host físicos y/o virtuales, contenedores y funciones serverless, ofreciendo de esta manera una protección global a todos los entornos de nube pública y privada que se quiera proteger. A continuación se detallaran todos los entornos donde se podrá desplegar la solución.

4.1 HOST FISICOS

La solución deberá soportar el despliegue en Host donde no se estén ejecutando contenedores, de esta manera se protegerá a todos los hosts que existan en el entorno, independientemente de su propósito. Se ejecutará como un servicio dentro del Host, siendo necesaria su compatibilidad tanto con Linux como con Windows.

Será necesario que soporte la protección de los siguientes Sistemas Operativos:

- **Amazon Linux 2**
- **CentOS 7, CentOS 8**
- **Debian 9, Debian 10**
- **EulerOS V2.0SP3, V2.0SP5**
- **Red Hat Enterprise Linux 7, Red Hat Enterprise Linux 8**
- **Ubuntu Server 18.04 LTS, 16.04 LTS**
- **Windows Server 2016, Windows Server 2019**

4.2 CONTENEDORES

La solución deberá soportar el despliegue en los Host donde se estén ejecutando los contenedores, deberá de proteger tanto el host como los contenedores que se estén ejecutando dentro del mismo. La solución en ningún caso se instalara nada dentro del contenedor, evitando así la necesidad de tener que crear de nuevo el contenedor con software nuevo dentro del mismo. El plano de control de la solución se desplegará como un contenedor dentro del Host, sin modificar nada internamente, ni del Host ni de los contenedores.

La solución se podrá desplegar tanto en modo stand alone, es decir, despliegue manual del contenedor agente, pero a su vez se podrá integrar dentro de los orquestadores como DaemonSet o global services de DockerSwarm

4.2.1 KUBERNETES

La solución deberá poder desplegarse e implementarse en cualquier implementación de Kubernetes, tanto sea una implementación On Premise o en cualquier servicio de Nube que ofrezca Kubernetes como servicio. A continuación se muestran los servicios:

- **Amazon Elastic Container Service for Kubernetes (Amazon EKS)**
- **Azure Container Service with Kubernetes**
- **Azure Kubernetes Service (AKS)**
- **DC/OS Kubernetes**
- **Google Kubernetes Engine (GKE)**
- **IBM Kubernetes Service (IKS)**
- **Alibaba Cloud Container Service for Kubernetes**

Se desplegará un único contenedor de Consola como “Deployment“, lo que garantiza que siempre se esté ejecutando dentro del entorno Openshift. Por otro lado el contenedor de Agente, deberá ser desplegado como “DaemonSet”, para garantizar

que este contenedor, se ejecutará dentro de todos los nodos del cluster de Openshift que se deba de proteger.

4.2.2 AMAZON ECS

La solución podrá desplegarse de igual manera en el entorno de Amazon ECS, donde se podrán desplegar tanto el Consola como el Agente dentro de un Cluster de Amazon ECS y proteger los contenedores que se ejecutarán dentro de ese Cluster, sin necesidad de instalar ningún agente o software dentro de la imagen de los contenedores del entorno.

Se desplegará un único contenedor de Consola en un nodo del cluster y se desplegará un contenedor del Agente en cada nodo del cluster, para proteger a los contenedores que existan dentro de ese nodo.

4.2.3 OPENSHIFT

La solución se podrá desplegar en un entorno de OpenShift, donde se podrán desplegar tanto el Consola como el Agente dentro de un Cluster de OpenShift y proteger los contenedores que se ejecutarán dentro de ese Cluster, sin necesidad de instalar ningún agente o software dentro de la imagen de los contenedores del entorno. Deberán poder ser desplegados en cualquier entorno OpenShift 3.9 o superior.

Se desplegará un único contenedor de Consola como “ReplicationController”, lo que garantiza que siempre se esté ejecutando dentro del entorno Openshift.Por otro lado el contenedor de Agente, deberá ser desplegado como “DaemonSet”, para garantizar que este contenedor, se ejecutará dentro de todos los nodos del cluster de Openshift que se deba de proteger.

4.2.4 DC/OS O MESSOS

La solución podrá desplegarse en un entorno DC/OS (Marathon/Messos), dado que este tipo de entornos, corresponden a un Sistema Operativo distribuido, el despliegue de la solución será mediante un único contenedor de la Consola en un nodo público esclavo que pueda ser accesible desde el exterior del cluster y de un contenedor del Agente en cada nodo slave privado del cluster que se quiera proteger.

4.2.5 DOCKER SWARM

La solución se podrá desplegar en un entorno de DockerSwarm, donde se podrán desplegar tanto el Consola como el Agente dentro de un Cluster de DockerSwarm y proteger los contenedores que se ejecutarán dentro de ese Cluster, sin necesidad de instalar ningún agente o software dentro de la imagen de los

contenedores del entorno. La solución deberá integrarse en los propios servicios nativos de DockerSwarm.

Se desplegará un único contenedor de Consola como un "Service", lo que garantiza que siempre se esté ejecutando dentro del entorno DockerS warm.Por otro lado el contenedor de Agente, deberá ser desplegado como "Global Service", para garantizar que este contenedor, se ejecutará dentro de todos los nodos del cluster de DockerSwarm que se deba de proteger.

4.2.6 VMWARE TANZU ENTERPRISE PKS

La solución podrá desplegarse de igual manera en el entorno de VMware Tanzu Enterprise PKS, dado que esta solución es un entorno Kubernetes ejecutándose en la nube de VMware.

Por lo tanto se seguirá la misma manera de despliegue que en Kubernetes, desplegándose un único contenedor de Consola como "Deployment", lo que garantiza que siempre se esté ejecutando dentro del entorno Openshift.Por otro lado el contenedor de Agente, deberá ser desplegado como "DaemonSet", para garantizar que este contendor, se ejecutará dentro de todos los nodos del cluster de Openshift que se deba de proteger, sin necesidad de instalar ningún agente o software dentro de la imagen de los contenedores del entorno.

4.2.7 WINDOWS

Además de host basados en Linux, la solución deberá soportar containers ejecutados sobre un host Windows Server. Asimismo, tendrá que ser compatible independientemente del Sistema Operativo que esté ejecutando el contenedor, es decir deberá soportar que ejecute tanto contenedores Windows y Linux. El agente nunca alterará las imágenes incluyendo binarios dentro de los contenedores o añadirá ficheros al Sistema Operativo.

4.3 FUNCIONES SERVERLESS

La solución deberá soportar la integración con funciones Serverless, deberá proteger las funciones en el tiempo de ejecución de las mismas y además monitorear que el funcionamiento de la función es el mismo para el que fue diseñado. La solución ser capaz de controlar y securizar las siguientes acciones dentro de la función:

- Actividad de los procesos.
- Conexiones de red
- Las modificaciones del Sistema de Ficheros

La solución deberá de soportar los siguientes tipos de función que se ejecuten en el entorno Serverless:

- C# (.NET Core 2.1)
- Java 8
- Node.js 10.x
- Python 2.7, 3.6, and 3.7

5. ÁMBITOS DE ACCIÓN

A continuación se detallan los principales casos de uso y funcionalidades descritas en el capítulo de recomendaciones generales.

5.1 SEGURIDAD EN HOST

El sistema operativo del host que aloja el entorno de contenedores es quizá uno de los aspectos más importantes y a veces más descuidados a la hora de hablar de seguridad en estos entornos.

Cualquier compromiso a nivel de host puede dar acceso al entorno completo de aplicaciones y por ello, securizar un entorno de contenedores, también requiere de securizar los hosts donde se alojan, comprobando regularmente y aplicando actualizaciones a todos los componentes software instalados en el sistema operativo anfitrión.

A este respecto, se necesita una línea de defensa específica que escanee continuamente paquetes vulnerables en el host, que podrían suponer puntos de entrada para un atacante.

Se habrán de auditar todos los intentos de autenticación al host, monitorizar todas las anomalías presentes, y registrar cualquier escalado para realizar tareas privilegiadas en el sistema.

Y se habrá de contar con una herramienta por línea de comandos que también permita lanzar estos escaneos bajo demanda.

5.2 GESTIÓN DE VULNERABILIDADES

Será necesario identificar y prevenir vulnerabilidades a lo largo de todo el ciclo de vida de la aplicación de forma integrada en pipelines CI/CD, lo que ayudará a los desarrolladores a encontrar y solucionar vulnerabilidades antes de que las imágenes lleguen al registro.

Habrá de contar con un plugin para Jenkins, entre otros, que establezca unas guías para los equipos de seguridad y desarrollo, de tal forma que el primero pueda crear políticas y definir estándares de cumplimiento que habrán de cumplir las imágenes en la fase de generación.

Por ejemplo, definir una política para que en caso de encontrar cualquier vulnerabilidad crítica en una imagen, bloquearla en fase de elaboración.
Además, será necesario contar con una herramienta por línea de comandos que permita escanear imágenes buscando vulnerabilidades o aspectos de cumplimiento, que además integrará estas capacidades de escaneo en el pipeline CI/CD o en cualquier otra herramienta propia.

Otro aspecto importante a tener en cuenta es el seguimiento en el cumplimiento. Después de desarrollar una imagen sin vulnerabilidades, esta se puede poner en un registro para su distribución, pero lo que hoy es una imagen sin vulnerabilidades mañana puede no ser así. Por eso es imprescindible la monitorización continua de las vulnerabilidades presentes en las imágenes en los registros, así como de los contenedores en ejecución, para lo que se dispondrá de un feed de inteligencia actualizado varias veces al día con nuevas amenazas (CVEs, malware, vulnerabilidades de día cero, etc), que serán evaluadas continuamente contra todas las imágenes y contenedores del entorno, generando informes a partir de los cuales podrán derivarse acciones sobre la inteligencia presente en los mismos.

Además, es necesario disponer de algún mecanismo que permita supervisar en el contexto del entorno de contenedores, para identificar los mayores riesgos y priorizarlos para remediarlos, con el siguiente detalle:

- Por cuantos CVEs estamos impactados a lo largo del tiempo, de qué severidad (baja, media, alta, crítica) para cada tipo (imagen, host, función).

- Cuántas imágenes, hosts o funciones están impactadas por vulnerabilidades de cualquier severidad a lo largo del tiempo, para por ejemplo poder actuar sobre los afectados por al menos una vulnerabilidad crítica, según la política definida.

- Una lista clasificada de las vulnerabilidades más críticas en el entorno separadas por host, imagen y función, que permitirá priorizar los esfuerzos en la remediación, ordenada por una puntuación de riesgo basada en las severidad del CVE y de otros factores que permiten priorizar la mitigación como:
    - ¿Hay un fix disponible?
    - ¿La vulnerabilidad puede ser explotada para ejecutar código arbitrario?
    - ¿El componente es vulnerable a ataques DoS?
    - ¿La vulnerabilidad se ha reportado recientemente?
    - ¿Existe algún exploit para dicha vulnerabilidad?
    - ¿La vulnerabilidad es fácilmente explotable?
    - ¿La vulnerabilidad es explotable remotamente a través de la red?
    - ¿La vulnerabilidad afecta a un contenedor expuesto a Internet?
    - ¿La vulnerabilidad afecta a un contenedor con puertos abiertos de

entrada?
    - ¿La vulnerabilidad afecta a un contenedor corriendo con privilegios elevados?
    - ¿La vulnerabilidad afecta a un contenedor corriendo sin ningún perfil de seguridad?
    - ¿La vulnerabilidad afecta a un contenedor corriendo con el flag *privileged*?
    - ¿La vulnerabilidad afecta a un componente actualmente en uso?

- Arboles de riesgo que, para cada CVE impactado, incluso pudiéndolo introducir manualmente en un cambio de búsqueda, muestre las imágenes, namespaces, contenedores y hosts afectados por esa vulnerabilidad en un árbol, con el conocimiento disponible de qué vulnerabilidades impactan a qué paquetes, qué paquetes están en qué imágenes, qué contenedores se derivan de esas imágenes, qué contenedores corren en qué namespaces, y que hosts corren qué contenedores, lo que además permite tener una visión global de la exposición a una vulnerabilidad a lo largo de todos los componentes del entorno.

Además, se podrán crear políticas que especifiquen cómo tratar las vulnerabilidades, con políticas de acción alert o block para según qué severidad de las mismas y a lo largo del entorno de una forma segmentada (por contenedores, imágenes, hosts, etiquetas, namespaces), condicionada o no a que el vendor ya haya sacado un fix, opcionalmente con excepción de una lista de CVEs o etiquetas para los que se puede sobreescribir la acción de forma indefinida o hasta una fecha de expiración configurada, y pudiendo establecer opcionalmente un período de gracia desde el momento de la publicación de la vulnerabilidad para disponer de un tiempo para corregirla sin impactar al servicio.

Ejemplo: Bloquear el despliegue en producción de contenedores con vulnerabilidades de severidad crítica, sólo si el vendor ha sacado algún fix y solo a partir del N-ésimo día desde la publicación de la vulnerabilidad.

Por último, será necesario poder descomponer cada imagen en capas, para destacar aquellos componentes con vulnerabilidades, que permitirá evaluar cómo ha sido introducida cada vulnerabilidad en la imágen como punto de partida para su remediación.

5.3 ESCANEOS

En fase inicial, se realizará un escaneo inicial de todas las imágenes en el host, y después de forma periódica cada 24h al menos, cuando se creen nuevas imágenes o se modifiquen las existentes, o bien bajo demanda cuando se desee.

También se escanearán los hosts (Linux, Windows) e imágenes de máquinas virtuales (Linux AMIs) en búsqueda de vulnerabilidades que afecten a cualquier paquete (de sistema operativo o distribución, binarios JAR, gems de Ruby, node.js, python, mysgl, nuget, o propios) instalado o a cualquier aplicación específica.

Se escanearán en búsqueda de CVEs publicados, vulnerabilidades por malas configuraciones, malware, vulnerabilidades de día cero, temas relacionados con cumplimiento normativo y secretos.

Además, será posible escanear imágenes en cualquier registro público o privado tales como Docker Hub, ECR, ACR, Docker Registry v2, Docker Trusted Registry, Amazon EC2 Container Registry (ECR), Azure Container Registry (ACR), Google Container Registry, Alibaba Container Registry, Harbor, IBM Cloud Container Registry, JFrog Artifactory Docker Registry, Openshift Docker Registry, Sonatype Nexus, CoreOS Quay, y se podrá usar webhooks para lanzar un escaneo cuando se añadan imágenes nuevas al registro o se modifiquen las existentes.

5.4 CUMPLIMIENTO NORMATIVO

Una de las funciones necesarias también es la de monitorizar y asegurar el cumplimiento normativo y/o de buenas prácticas, nuevamente, a lo largo de todo el ciclo de vida de una aplicación, y de forma automatizada, tanto en hosts, imágenes, contenedores, funciones, clusters y clouds.

Se dispondrán ya de cientos de chequeos con las buenas prácticas, para habilitar aquellas que estén alineadas con la política de seguridad de la organización.

Estos chequeos han de estar basados en los estándares (GDPR, PCI, NIST SP 800-190, HIPAA) de la industria, tales como varios CIS (Center for Internet Security) Benchmarks (Docker, Kubernetes, Distribution Independent Linux, AWS Foundations), y otros propios de la solución, así como definir otros personalizados con scripts (bash, powershell) u otros más avanzados con XCCDF.

Las reglas de cumplimiento se aplican de igual forma que las de vulnerabilidades, y son aplicadas en tiempo de creación de los componentes, previniendo la creación de un contenedor si viola la política.

Ejemplos de chequeos en imágenes o contenedores son: imágenes con malware, uso de contenedores privilegiados, imagen creada con usuario root, (secretos) claves privadas almacenadas en la imagen o información sensible en variables de entorno, imagen no trusted, imagen no actualizada, etc.

Ejemplos de chequeos en hosts: (Para Windows) Firmas de antivirus actualizadas,

Windows Defender habilitado, Windows Defender ejecutándose, Windows Update habilitado, Windows Update configurado para instalar automáticamente nuevas actualizaciones, (Para Linux) iptables instalado, SNMP Server no habilitado, Samba no habilitado, etc.

Además, se podrán declarar los registros/repositorios/imágenes de confianza permitidos en el entorno, registrando, alertando y bloqueando opcionalmente cualquier ejecución en la que no se confíe.

Esto permitiría el control en la reutilización de software, a partir de imágenes en registros públicos (como Docker Hub) que podría ir en contra de la política de seguridad de una organización que desearía tener controlados los proveedores autorizados de software y puntos de distribución del mismo, para evitar posibles sabotajes. Como ejemplo, posibilitará limitar el entorno de producción a imágenes aprobadas en el registro privado corporativo de confianza, bloqueando cualquier imagen procedente de cualquier repositorio público.

Será necesario un cuadro de mandos único a nivel de cumplimiento histórico para todos los componentes (imágenes, contenedores, hosts) del entorno, junto con una lista de chequeos de cumplimiento fallidos en base a las políticas definidas.

Igualmente, para dinamizar el descubrimiento de todas las cargas nativas en nube, será necesaria una función que permita monitorizar las diferentes nubes para detectar la presencia de cualquier servicio clandestino que no esté debidamente securizado.

Al menos habrá de cubrir AWS, Azure y GCP, incluidas sus plataformas de Kubernetes.

5.5 PROCESOS DE INTEGRACIÓN/IMPLEMENTACIÓN CONTINUAS (CI/CD)

Se habrá de integrar la seguridad dentro de los flujos de trabajo de integración continua, para encontrar problemas y solucionarlos antes de pasar a producción.

Será necesario contar con un plugin de Jenkins que permita incorporar el escaneo de vulnerabilidades y chequeos de cumplimiento en el pipeline de integración continua.

Se permitirán o bloquearán desarrollos dependiendo del tipo de problemas encontrados y las políticas definidas.

Incorporando el escaneo a la fase de desarrollo en el flujo de trabajo, permitirá evidenciar de forma temprana los problemas de seguridad detectados y los

desarrolladores recibirán información de lo que necesita ser corregido para pasar las políticas establecidas.

También será necesario disponer de una utilidad por línea de comandos para invocar los escaneos desde otras herramientas o flujos de trabajo.

5.6 DEFENSA EN TIEMPO DE EJECUCIÓN

Es una de las partes más complicadas de securizar, dado que las herramientas de seguridad tradicionales no están diseñadas para monitorizar la ejecución de los contenedores, ya que no pueden ver su interior y mucho menos establecer una línea base de cómo debería comportarse.

Por tanto, habrán de protegerse los entornos de ejecución gracias al aprendizaje automático, que creará modelos en tiempo de ejecución para todas las versiones de cualquier aplicación, de manera que se aplique el criterio del mínimo privilegio y se establezcan los comportamientos autorizados.

Este modelado por comportamiento será granular (contenedor, imágen, host, etiqueta, namespace) y en diferentes dimensiones:

- Sistema de ficheros: Protección frente a actividad sospechosa a nivel de sistema de ficheros como cambios en cualquier fichero fuera de las carpetas aprendidas en el modelo, cambios a binarios o certificados, cambios en ficheros de configuración de la cuenta administrativa SSH, presencia de malware.
- Conexiones de red: Conectividad IP (lista blanca y lista negra de puertos en escucha, ips y puertos en conexiones salientes), detección de escaneo de puertos, de raw sockets, DNS (lista de dominios aprendidos, lista de dominios malos/botnets conocidos actualizados a través de un feed incluido en la solución, listas propias de dominios buenos y malos).
- Procesos: Lista de procesos aprobados que pueden ser ejecutados en el contenedor.
- Host: Procesos aprobados que pueden ser ejecutados junto con los ficheros con los que interactúan.

Una vez activado el modelo, los sensores monitorizarán en estas 4 dimensiones los diferentes contenedores en ejecución, y las políticas determinarán como se tratarán las violaciones: alertar, prevenir la acción (el proceso anómalo o procesos específicos, la integridad del sistema de ficheros, las peticiones DNS anómalas), o parar la ejecución del contenedor.

Se podrán definir listas blancas o listas negras para complementar los modelos aprendidos que no hayan capturado completamente la totalidad de actividades legítimas en un contenedor.

Se incluirán capacidades de detección de malware en los contenedores gracias a disponer de firmas de malware actualizado por el feed de la solución.

Igualmente se podrá modelar la actividad del host en sí mismo, en base a los servicios permitidos junto con los procesos que puede correr y los ficheros a los que tiene acceso, alertando o previniendo cualquier actividad que suponga una violación, incluyendo la presencia de malware, conexiones a sitios maliciosos, resoluciones DNS de dominios maliciosos, auditando conexiones entrantes o salientes, y detectando lecturas o escrituras en ficheros críticos como certificados, secretos, ficheros de configuración, cambios en binarios o en atributos como permisos, propietario, marcas de tiempo o enlaces.

De forma adicional, se podrán utilizar expresiones personalizadas para detectar comportamientos en ejecución que podrán ser utilizadas en reglas también personalizadas, en base a atributos de procesos invocados, escrituras en sistema de ficheros, y conexiones salientes.

Para una mejor y más efectiva respuesta ante incidentes, será necesario contar con un panel visual que relacione diferentes violaciones para asociarlos dentro del mismo ataque o incidente de seguridad, ofreciendo la relación de evidencias que provocaron el incidente, informes de escaneo de vulnerabilidades para los elementos (imagen, contenedor, y host) implicados, conexiones entrantes/salientes al/del contenedor implicadas en el incidente e información forense acerca de lo sucedido antes, durante y después de la brecha de seguridad.

Entre los tipos de incidentes detectados habrán de estar:

- Puertas traseras para acceso administrativo
- Puertas traseras para acceso ssh
- DoS por fuerza bruta
- Presencia de criptomineros
- Exfiltración de información
- Secuestro de procesos
- Ataque a kubernetes
- Movimiento lateral
- Escaneo de puertos como técnica de reconocimiento
- Servicios tratando de utilizar más privilegios de los que tiene

5.7 FIREWALLS PARA ENTORNOS NATIVOS EN NUBE

Se deberán disponer de diferentes funcionalidades firewall que por un lado permitan proteger los microservicios frente a ataques y anomalías y por otro permitan única y exclusivamente las comunicaciones deseadas entre los diferentes microservicios.

Deberán aprender la topología de las aplicaciones ofreciendo una microsegmentación adaptada para todos los microservicios, protegiendo a los contenedores frente a amenazas.

5.7.1 FIREWALL DE APLICACIÓN WEB

Se habrá de disponer un firewall de aplicación web para securizar aplicaciones web tanto "containerizadas" como no "containerizadas", con independencia de la nube, orquestador, nodo o direccionamiento en el que corra, y proporcionará al menos las siguientes capacidades para proteger a las aplicaciones web:

- Inyección SQL
- Cross site scripting
- Herramientas de ataque (crawlers, pentesting, etc)
- Peticiones mal formadas
- Cross-site request forgery
- Clickjacking
- Shellsock
- Patrones en cabeceras HTTP
- Habilitar o no subidas de ficheros y de qué tipos
- Eliminar mensajes de error para evitar obtención de inteligencia:
  - Protección frente a ataques de fuerza bruta
  - Seguimiento de códigos de error en las respuestas
  - Eliminar cabeceras reveladoras de la huella digital de la aplicación
  - Directory Traversal
  - Detección de fuga de información
- Lista blanca y lista negra de ips en el acceso a la aplicación
- Puertos HTTP y HTTPS en los que escucha la aplicación
- Lista de URLs permitidas en la aplicación

5.7.2 FIREWALL DE NIVEL 4 CON MODELADO AUTOMÁTICO

Y además, también se habrá de contar con un firewall de nivel 4, que utilizará machine learning para modelar el tráfico entre contenedores creando reglas que permitirán distinguir el tráfico bueno del malo sin intervención manual y actuando

como un firewall este-oeste entre contenedores para entre otras cosas evitar movimientos laterales en caso de comprometer el perímetro más externo.

Este firewall habrá de tener las siguientes capacidades:

- Permitir segmentación a nivel de contenedor para limitar movimientos laterales y posibles exfiltraciones de datos.
- Aplicar, monitorizar y visualizar el impacto de la política de microsegmentación.
- Mostrar gráficamente en un panel como se comunican los contenedores, pudiendo bloquear al instante e interactivamente cualquier comunicación no deseada, desde el mismo panel.
- Aprendizaje automático de las comunicaciones entre microservicios, pudiendo aplicar directamente la política aprendida y bloqueando el resto de comunicaciones no permitidas.
- Posibilidad de refinar manualmente la política aprendida.
- Todo lo anterior mencionado para contenedores, también aplicable a hosts.

5.8 CONTROL DE ACCESO

Será necesario establecer y monitorizar, de forma centralizada, las medidas de control de acceso para las aplicaciones nativas en nube, con el objetivo de ofrecer el acceso necesario y menos privilegiado posible a los diferentes equipos (devops, devsecops, etc) para desempeñar sus funciones y poder monitorizar y auditar cualquier actividad en la infraestructura.

Se dispondrá de un control de acceso basado en roles que permita establecer políticas para quién puede acceder a que recursos de qué entorno y con qué comandos.
Las reglas permitirán ser aplicadas de forma granular a nivel de host, imágen, contenedor, y etiquetas, con expresiones basadas en patrones.

Habrán de estar disponibles al menos los siguientes roles:

- Administrador: Acceso completo para administradores de seguridad.
- Operador: Para equipos de operaciones de seguridad.
- Auditor: Solo lectura para auditores y equipos de cumplimiento.
- DevSecOps: Visualizar pero no cambiar políticas ni configuraciones.
- Gestor de vulnerabilidades: DevOps encargados también de vulnerabilidades y cumplimiento.
- DevOps: Monitorizar escaneos y herramientas que permitan determinar porque el pipeline se ha parado.

- Gestor de despliegues: Personal de DevOps encargado de los despliegues de los componentes de la solución de seguridad.
- Usuarios: Para interactuar con los contenedores de forma remota vía comandos.
- Integración contínua: Para interactuar con los plugins que incluyen escaneos de vulnerabilidades y cumplimiento dentro del pipeline.

Además se podrán aplicar políticas de control de acceso para restringir quién puede hacer push o pull en los registros privados, integrado con los servicios de directorio corporativos (directorio activo, ldap, etc) o proveedores de identidades a través de SAML.

6. RECOMENDACIONES EN LA OPERACIÓN

Parte del éxito de una solución de seguridad para contenedores, además de la tecnología elegida, depende también de cómo se operacionalice en cada entorno.
A continuación, ofreceremos unos consejos y recomendaciones de referencia para poner en operación la solución, a través de un viaje en el tiempo.

6.1 CRONOGRAMA

Un cronograma maestro que mostraría una vista a ojo de pájaro de las fases necesarias para operacionalizar la solución, dispondría los siguientes pasos:

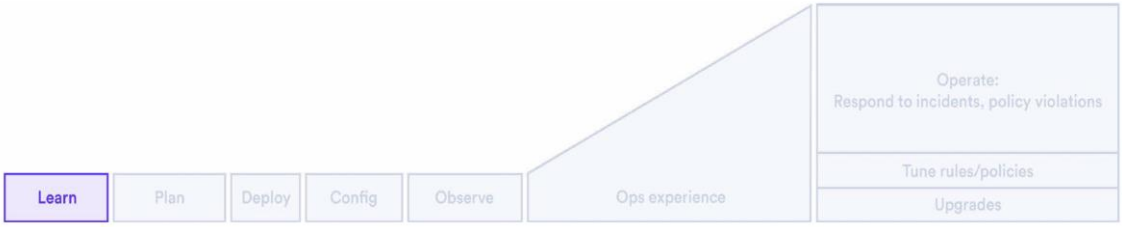
Que de forma general se podrían describir como:

- Aprendizaje sobre los conceptos de la solución y cómo funcionan.
- Mapear la solución al entorno existente, eligiendo el modo de despliegue idóneo y personalizarlo a las necesidades concretas, enfocándolo también desde la automatización, la alta disponibilidad y la recuperación ante desastres.
- Securizar el entorno, configurando la solución y aplicando sucesivamente protección frente a vulnerabilidades, cumplimiento, defensa en ejecución y firewalls, por este orden.

- Mantener y operacionalizar la solución, respondiendo a incidentes y alertas, añadiendo y modificando las políticas según sea necesario por presencia de nuevas aplicaciones o actualización de las existentes y actualizando la solución conforme salgan nuevas versiones.

Que detallamos a continuación:

6.1.1 APRENDIZAJE



Descargar el software y familiarizarse con los conceptos de la solución.

6.1.2 PLANIFICACIÓN

Planear cómo se realizará el despliegue de la solución, estudiando los métodos de instalación disponibles que mejor se ajusten a cada organización y entorno que dependerán de:

- Cuantas consolas desplegar. ¿Una por entorno?¿Una para producción y otra para el resto de entornos? ¿Una para todos los entornos?
- Donde se desplegará la consola (SaaS u on-premise)
- Qué relación existe entre los equipos de personas y los entornos
- Qué tipo de cargas se securizarán (hosts, contenedores, funciones, etc), si son de tipo IaaS o PaaS, etc
- Qué topologías de red existen. Por ejemplo, en entornos aislados será necesario instalar una consola específica como plano de gestión para el plano de datos de ese entorno
- Como se hará el despliegue (UI, API, línea de comandos, etc)

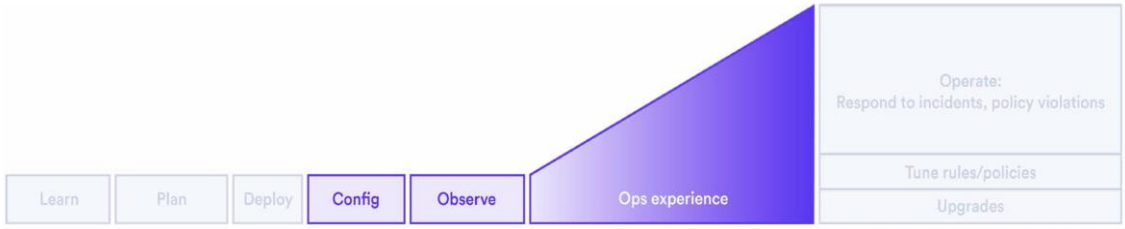
6.1.3 DESPLIEGUE

Realizar el despliegue de la solución acorde a lo decidido en la fase anterior.

6.1.4 OBSERVACIÓN

Una vez instalada la solución, observar durante el primer mes para familiarizarse con los datos reportados en base a las políticas por defecto y entender cómo funcionan.
Se puede empezar por visualizar los 10 top CVEs detectados en el entorno.

6.1.5 OPERACIONALIZAR



Hay 4 ámbitos principales a poner en funcionamiento y operacionalizar:

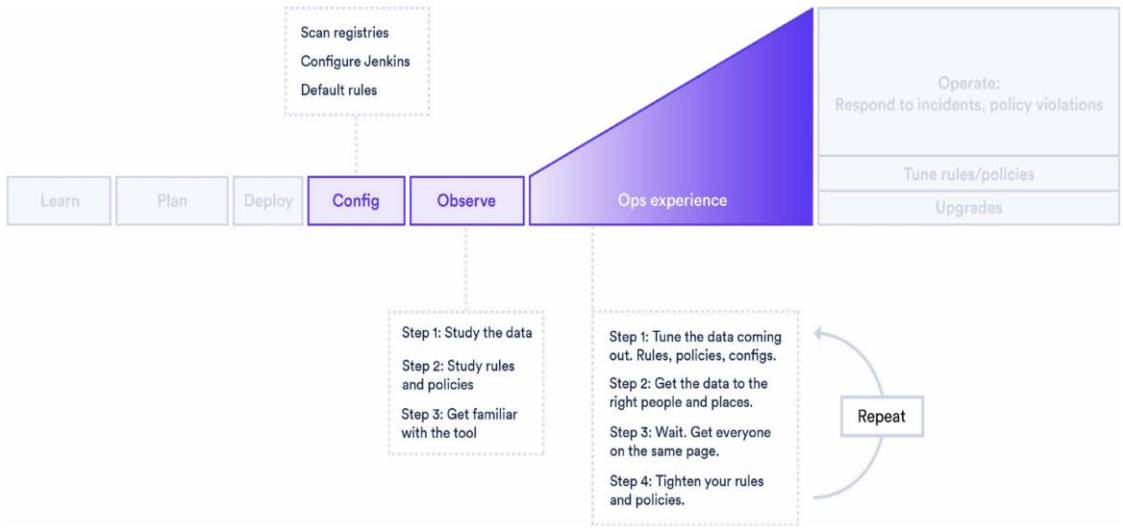
- Gestión de vulnerabilidades
- Cumplimiento
- Defensa en tiempo de ejecución
- Firewalls

Y cada uno de ellos se ha de operacionalizar en fases:

- **Fase 1**: Configurarlo, habilitar lo necesario.

- **Fase 2**: Observar la información resultante (informes, datos) y familiarizarse con la funcionalidad y sus capacidades.
- **Fase 3**: Aplicar la política paso a paso personalizándola progresivamente.

6.1.5.1 GESTIÓN DE VULNERABILIDADES



Tras la instalación de la solución, automáticamente escaneará vulnerabilidades en los hosts, imágenes y contenedores, y se analizará esta información resultante antes de configurar ninguna tarea de escaneo adicional.

A continuación, se configurará la solución para escanear los registros autorizados y se instalará y configurará el plugin para Jenkins o en caso de no utilizar Jenkins, la utilidad por línea de comandos que permita escanear las imágenes resultantes de la fase de Build.

La solución habrá de actuar como enlace entre los segmentos CI (Continuous Integration) y CD (Continous Deployment) del pipeline, pues las imágenes son generadas en CI, tras lo cual han de ser escaneadas automáticamente en búsqueda de vulnerabilidades y en función de los umbrales definidos en la política se permite su publicación o no en el registro de imágenes.

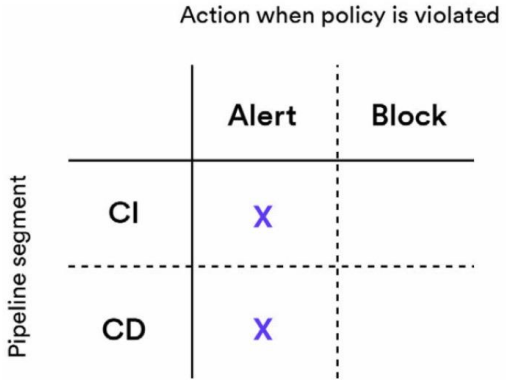
Por otra parte, en el segmento CD, el orquestador envía las imágenes desde el registro a producción para su ejecución en forma de contenedores, que nuevamente la solución habrá de escanear previamente a autorizar su ejecución, si está bajo los umbrales establecidos en la política en este punto, o se notificará y opcionalmente bloqueará en caso contrario.

Debe haber 3 puntos donde se pueda bloquear la ejecución de un contenedor:

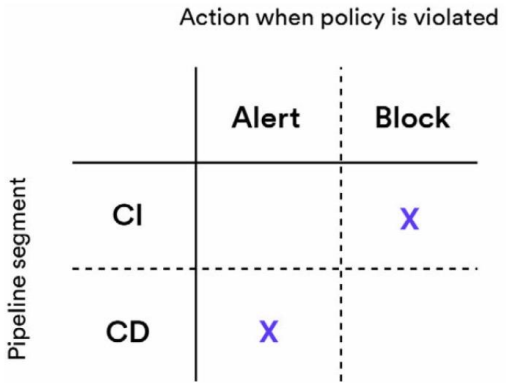
- En tiempo de generación de la imagen. Una vez se haya generado la imagen, habrá de ser escaneada para asegurar que cumple con las políticas de vulnerabilidades y cumplimiento establecidas.
- Antes de que el contenedor pase a ejecución: Antes de pasar a ejecución, para asegurar que cumple con las políticas de vulnerabilidades y cumplimiento.
- Cuando el contenedor está ejecutándose: Si se detecta cualquier anomalía que haga que el comportamiento se desvíe de su actividad base conocida, la funcionalidad de defensa en tiempo real puede parar la ejecución del contenedor.

¿Por qué es importante disponer de puntos de control en los segmentos CI y CD del pipeline? Porque una imagen sin vulnerabilidades en el momento de subirla al registro, pasado un tiempo puede pasar a tener vulnerabilidades críticas descubiertas con posterioridad.

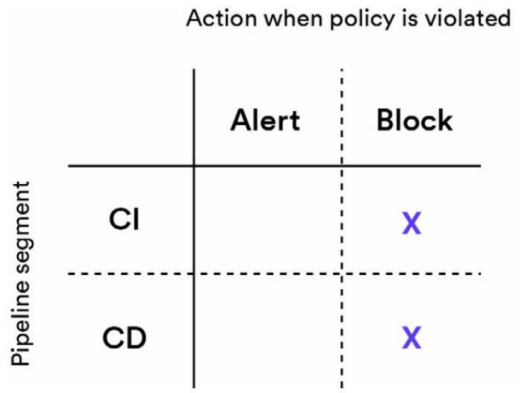
Se recomienda empezar con el mínimo impacto en el pipeline, simplemente notificando y observando la afectación, inicialmente sólo sobre los CVEs de severidad crítica:



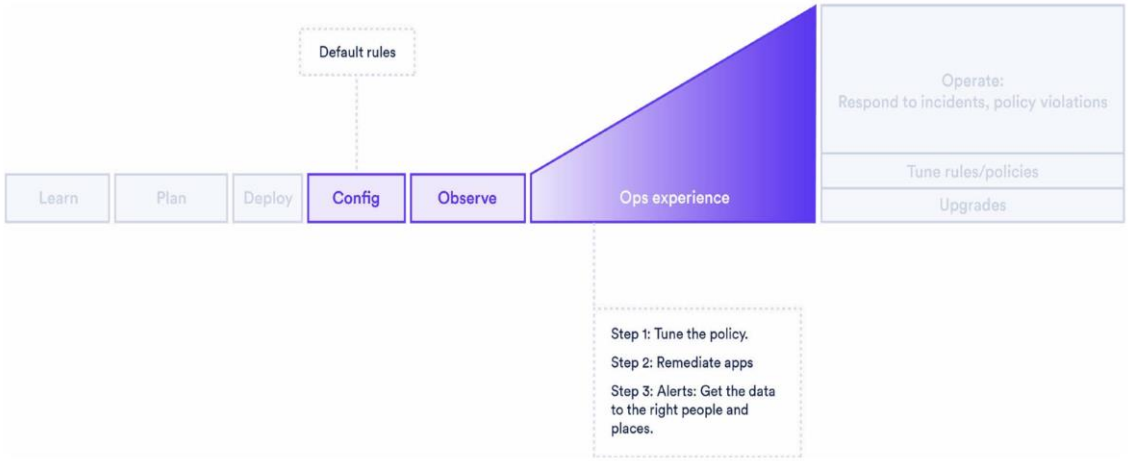
Posteriormente, se empezarán a establecer restricciones en CI bloqueando las imágenes que no cumplan con la política, en este punto solo para los CVEs de severidad crítica para los que haya fix, estableciendo un período de gracia para que los desarrolladores tengan tiempo para hacer las correcciones oportunas, antes de proceder a bloquearlo. Después se bajará el umbral de severidad, de crítica a alta, para considerar CVEs de severidad crítica y alta. En CD se seguirá manteniendo la notificación para cualquier violación:



Por último, habrá quien desee ir un paso más allá, activando el bloqueo también en CD para evitar que imágenes que no cumplan con la política establecida, se ejecuten en producción, para lo que se recomienda, igual que en CI, empezar con objetivos fácilmente alcanzables y progresivamente irlos incrementando a razón de confirmar el bajo índice de falsos positivos proporcionado por el feed de inteligencia de la solución, y activándose estos bloqueos solo cuando realmente hacen relación a problemas encontrados que necesitan ser resueltos:



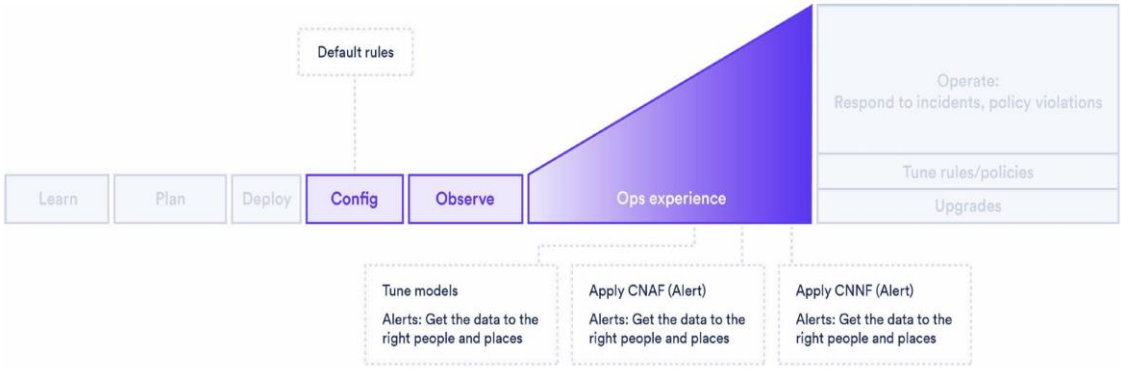
6.1.5.2 CUMPLIMIENTO



Se habrá de refinar la política por defecto de cumplimiento ofrecida por la solución, para adaptarla a cada entorno específico.

Por ejemplo, muchos componentes del orquestador requieren más privilegios que los necesarios para el resto de aplicaciones, debiendo ejecutarse como root para funcionar, por lo que se deberá exceptuar estos casos de la política de cumplimiento que exige que un contenedor no se ejecute como root.

6.1.5.3 DEFENSA EN TIEMPO DE EJECUCIÓN



Las funciones de defensa en tiempo de ejecución se basan en modelar de forma automática el funcionamiento de una imagen para que pueda ser securizada en tiempo de ejecución.

Y los modelos se componen de reglas lista blanca que reflejan que podrá hacer el contenedor, en base a lo que se ha aprendido como comportamiento base, que en el 90% de los casos será suficiente pero en el 10% restante el modelo no reflejará completamente el comportamiento del contenedor, habiendo de refinarlo para evitar falsos positivos que podrían hacernos pasar desapercibidos otras alertas relativas a actividad maliciosa o anómala.

Para ajustar el modelo, se recomienda actuar aplicación a aplicación, revisando las reglas aprendidas junto con el desarrollador, volviendo a forzar el aprendizaje o añadiendo manualmente las reglas. Esto debería ser necesario en un porcentaje pequeño de los modelos generados para todas las aplicaciones.

6.1.5.4 FIREWALLS

Para que la solución aprenda la topología de red de las aplicaciones y ofrecer una microsegmentación ajustada a todos los microservicios, son necesarios dos tipos de firewalls para proteger las aplicaciones a nivel de contenedor.

Un firewall de aplicación web, en caso de disponer de contenedores que gestionen peticiones web, que se configurará inicialmente para alertar y monitorizar

peticiones maliciosas, a partir de las cuales después se crearán reglas para bloquearlas de forma específica.

Un firewall de nivel 4 que modelará automáticamente el tráfico entre contenedores, y que inicialmente se configurará para notificar todo tráfico que no se ajuste al modelo, para después pasarlo a bloqueo una vez ajustado el mismo de forma manual.

6.1.6 MANTENIMIENTO Y OPERACIÓN

Una vez la solución está desplegada, configurada y refinada, a partir de aquí el trabajo debe consistir en:

- Responder a incidentes y otros eventos, con alta probabilidad de tratarse de un ataque, de alguna situación de no cumplimiento normativo, o de vulnerabilidades detectadas en el entorno.
- Refinar las políticas cuando se incorporan nuevas aplicaciones o se actualizan las existentes.
- Actualizar la solución conforme salen nuevas versiones de software.

6.2 ROLES

Como solución colaborativa, representantes de diferentes departamentos participarán en la planificación y operación de la misma, que, de forma alineada, el pipeline CI/CD será fluido y con controles de seguridad que controlarán el flujo de forma transparente.

Los participantes suelen ser:

- Seguridad: Con experiencia y autoridad en decisiones sobre la política de seguridad. Analizan e interpretan la información de auditoría resultante.
- DevOps: Conocen la infraestructura, la topología del entorno. Instalan la solución, despliegan los componentes de la misma y las integraciones con registros, pipelines, etc.

- Desarrollo: Conocen las aplicaciones que se ejecutan en el entorno, su composición, la relación entre componentes, etc. Corrigen los problemas de seguridad que paran el avance del pipeline CI/CD y ayudan a remediar los problemas de seguridad en ejecución. Ayudan a Seguridad a refinar los modelos que protegen las aplicaciones en ejecución.

Normalmente, la solución de seguridad será pilotada por uno o dos miembros del departamento de Seguridad y uno o dos miembros del departamento de *DevOps*.

Al equipo de *Desarrollo* se le otorgarán derechos solo lectura a informes sobre vulnerabilidades y herramientas para lanzar escaneos.

Tanto *Seguridad* como *Devops* comenzarán a trabajar con la solución desde el principio, entendiendo los conceptos y los flujos de trabajo, pero se recomienda no dejar de un lado a *Desarrollo* pues cuanto más quede insertada la solución dentro de sus flujos de trabajo, mejores resultados se obtendrán y menos tardarán en mitigar los problemas detectados de su competencia y menores serán los tiempos de interrupción.

Conviene planificar sesiones periódicas con Desarrollo para comentar expectativas, mostrarles cómo funciona la solución y cómo se integra en su flujo de trabajo.