Trabajo Fin de Máster
Ingeniería de Telecomunicación

Seguridad en la integración continua de la metodología ágil y la filosofía DevOps

Autor: Eleazar Rubio Sorrentino

Tutor: Juan Manuel Vozmediano Torres

Dep. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Trabajo Fin de Máster

Ingeniería de Telecomunicación

Seguridad en la integración continua de la metodología ágil y la filosofía DevOps

Autor:

Eleazar Rubio Sorrentino

Tutor:

Juan Manuel Vozmediano Torres

Profesor Titular

Dep. de Ingeniería Telemática

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2017

Trabajo Fin de Máster: Seguridad en la integración continua de la metodología ágil y la filosofía DevOps

Autor: Eleazar Rubio Sorrentino
Tutor: Juan Manuel Vozmediano Torres

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Apartado de agradecimientos.

Eleazar Rubio Sorrentino

Sevilla, 2017

Resumen

En los últimos tiempos de las empresas dedicadas al desarrollo de software como servicio (SaaS), los conceptos de metodología ágil y la filosofía Development and Operations (DevOps) están cobrando, cada vez más, un papel fundamental para el desarrollo de las mismas[10]. Según un estudio de alcance mundial realizado recientemente por CA Technologies, más del 75 por ciento de las organizaciones españolas coinciden en que las metodologías ágiles y DevOps son cruciales para el éxito de la transformación digital[16].

Este nuevo modo de entender el mundo del desarrollo de software (SW) posee una serie de elementos comunes, cada uno de ellos implementado con herramientas cada vez más conocidas y populares para las empresas que lo llevan a la práctica:

- Plataformas de desarrollo colaborativo y control de versiones de SW (por ejemplo GitHub), donde se almacena el código desarrollado por las mismas.

- Diferentes entornos o infraestructuras de trabajo para los desarrolladores, que van a permitir un desarrollo y despliegue continuo para las mejoras del producto: entornos de desarrollo, entornos de seguro de calidad o Quality Assurance (QA), preproducción, producción o entorno final, etc.

- Software de integración continua, en inglés Continuous Integration (CI), con las que automatizar los trabajos de despliegue de software.

- Plataformas basadas en el despliegue de contenedores (generalmente Docker) que encapsulan los distintos elementos que componen el producto final y optimizan los recursos utilizados en las máquinas que los contienen, en su mayoría subcontratadas a terceras compañías (Amazon Web Services, Microsoft Azure, etc.).

En el presenta Trabajo Fin de Máster (TFM) se pretende aportar e introducir al proceso comentado (y haciendo uso de las herramientas que este provee) una serie de análisis de seguridad que, ejecutados periódicamente, provean a la compañía de informes con los que poder identificar los siguientes problemas de seguridad durante el desarrollo de sus aplicaciones, siempre en continua integración con la línea de trabajo:

1. Si la aplicación creada tiene Vulnerabilidades (Common Vulnerabilities and Exposures (CVE)) en las librerías de dependencias de código utilizadas.

2. Si la imagen que se va a emplear para desplegar el contenedor de dicho SW contiene vulnerabilidades conocidas al nivel de Sistema Operativo (SO).

TODO - Cambiar enlaces por referencias bibliográficas.

Abstract

Lately inside Software as a Service (SaaS) companies, the agile methodology and DevOps philosophy concepts are taking a fundamental role for the development of them[10]. According to a recent global survey by CA Technologies, more than 75 percent of Spanish organizations agree that DevOps and agile methodologies are crucial to the success of digital transformation[16].

This new way of understanding the world of SW development has a various common elements, each of them implemented with well known and popular tools for the companies that put it into practice:

- Collaborative development platforms and SW version control systems (for example GitHub), where the code developed is keeped.
- Different environments and infrastructures for developers, which will allow to continuous development and deployment for product enhancements: development environments, Quality Assurance QA environments, pre-production, production or final environment, etc.
- Continuous integration CI software, with which to automate the software deployment.
- Platforms based on container deployment (usually Docker) that encapsulate elements that make up the final product and optimize the resources used in the machines that contain them, mostly subcontracted to third parties (Amazon Web Services, Microsoft Azure, etc.).

In the present master's thesis is intended to contribute and introduce to the process discussed (making use of the tools it provides), security analyzes that periodically executed provide the company with reports with which Identify the following security issues along of the process of development applications, always in continuous integration with the company pipeline:

1. If the application created has Vulnerabilities (CVE) in the code dependencies used.
2. If the image used to deploy the container of the mentioned SW contains known vulnerabilities at the Operative System (OS) level.

Índice

Introducción

You can ask 10 people for a definition of DevOps and likely get 10 different answers.

Dustin Whittle, 2014
Developer Advocate at Uber Developer Platform

En este primer apartado de la memoria se pretende realizar con el lector un recorrido a través del contexto que ha motivado el desarrollo del presente TFM para el Máster en Seguridad de la Información y las Comunicaciones de la Universidad de Sevilla, con el fin de aclarar el objetivo perseguido en su realización. Como conclusión al mismo, se definirá la estructura seguida durante la redacción, introduciendo cada uno de los apartados que se encontrarán a continuación.

Contexto y motivación

El año 2017, para empresas englobadas en todo tipo de sectores, está siendo el año de la transformación digital. Estos procesos de transformación exigen distintas formas de trabajo, más ágiles y colaborativas, con las que poder aplicar nuevas tecnologías que permitan conseguir los objetivos del negocio, en entornos que afrontan grandes desafíos culturales, organizativos y operativos e incluso pueden llegar a tener que lidiar con sistemas tecnológicos antiguos y casi obsoletos[2].

Es en este contexto donde el concepto DevOps empieza a sonar con más fuerza: el contexto de las metodologías ágiles.

De esta forma, DevOps es un concepto de trabajo, basada en el desarrollo de código, que usa nuevas herramientas y prácticas para reducir la tradicional distancia entre técnicos de programación y de sistemas, respondiendo a la necesidad experimentada por el sector tecnológico de dar una respuesta más rápida a la implementación y operación de aplicaciones. Este nuevo enfoque de colaboración que es DevOps permite a los equipos trabajar de forma más cercana, aportando mayor agilidad al negocio y notables incrementos de productividad.

Desde las pruebas de concepto hasta el lanzamiento, pasando por el *testing* y los entornos de prueba, todos los pasos involucrados requieren de la máxima agilidad posible (Figura 1.1), y eso pasa por integrar los procesos y los equipos de programación con los de sistemas[3].

Por otro lado, el concepto de contenedor de aplicación (aislamiento de espacio de nombres y gobernanza de recursos) a pesar de no ser un concepto novedoso, está cobrando cada vez más y más relevancia en el

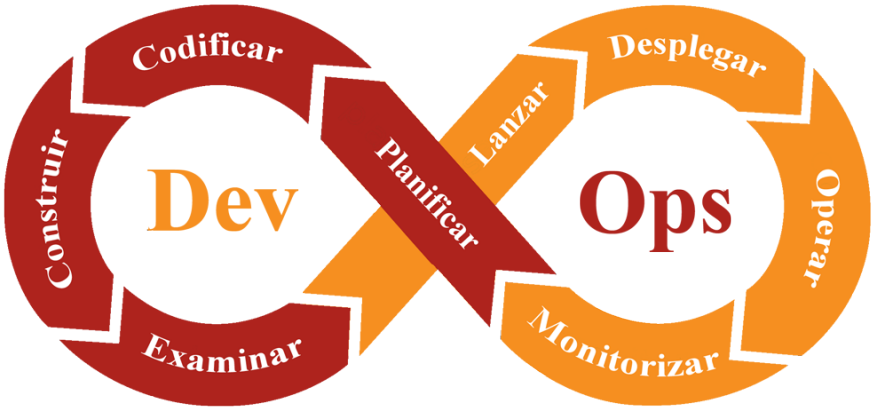

Figura 1.1 Introducción al proceso DevOps.

panorama de la empresa actual, de la mano de las continuas mejoras que experimentan las tecnologías que lo implementan, simplificando la administración y transformando la forma en que se desarrolla, distribuye y ejecuta el software, en forma de microservicio[1], además de proveer la habilidad de encapsular todo el entorno utilizado con el objetivo de ser desplegado en los sistemas de producción de la empresa, manteniendo las mismas características, aumentando la escalabilidad y disminuyendo notablemente los costes asociados a infraestructuras.

Los contenedores juegan un papel clave en un entorno DevOps porque soportan las implementaciones de la pila de desarrollo y operaciones completa y van en camino de formar parte de la definición básica de lo que se conocerá como DevOps en unos pocos años[6].

Además, la metodología DevOps representa una gran promesa a la hora de asegurar el desarrollo del software, ya que las organizaciones pueden potencialmente encontrar y remediar las vulnerabilidades con mayor frecuencia y al principio del ciclo de vida de la aplicación, ahorrando costes y tiempo. Conforme al informe de *"Seguridad de Aplicaciones y DevOps"* de octubre de 2016 promovido por Hewlett Packard Enterprise[7], que incluye tanto respuestas cualitativas como cuantitativas de profesionales de operaciones informáticas, líderes de seguridad y desarrolladores, se concluye que el 99 % de los encuestados confirma que la adopción de la cultura DevOps aporta la oportunidad de mejorar la seguridad de las aplicaciones. Sin embargo, solo el 20 % realizan análisis de seguridad de aplicaciones durante el desarrollo y el 17 % no utilizan ninguna tecnología que proteja sus aplicaciones, destacando una desconexión significativa entre la percepción y la realidad de la seguridad DevOps.

Es en el contexto planteado donde surge la idea del presente TFM: aportar mecanismos a la metodología DevOps que permitan analizar la seguridad de las aplicaciones desarrolladas y el contenedor que las albergará dentro de la infraestructura de la empresa, sin interferir de manera destructiva con el propio proceso de desarrollo y depliegue de la aplicación.

Objetivo

El objetivo del presente Trabajo Fin de Máster (TFM) es desarrollar un entorno, basado en contenedores, que pueda ser incluido en el proceso de desarrollo e integración continua de la empresa y con el que poder realizar tareas periódicas programadas para analizar estáticamente las posibles vulnerabilidades (CVETODO - ¿alguna más?) contenidas en las dependencias de aplicaciones desarrolladas mediante los

[1] Aproximación para el desarrollo software que consiste en construir una aplicación como un conjunto de pequeños servicios, los cuales se ejecutan en su propio proceso y se comunican con mecanismos ligeros (normalmente una API de recursos HTTP)

lenguajes de programación Ruby y NodeJS, además de analizar a nivel del SO vulnerabilidades presentes en las imágenes que van a constituir el contenedor que dará soporte a dichas aplicaciones.

Como medio para alcanzar el objetivo planteado se va a hacer uso de una serie de aplicaciones y herramientas, entres las que cabe destacar(TODO - no me gusta esta última frase), que serán desarrolladas en el apartado 3:

- GitHub[9]: Plataforma de desarrollo colaborativo y control de versiones de SW donde almacenar, entre otras, el código desarrollado y que será analizadp.
- Jenkins[11]: Software de integración continua (CI) con el que automatizar los trabajos periódicos de análisis estático a realizar.
- Docker[8]: Proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, con el que desplegar los distintos elementos requeridos.

Estructura de la memoria

Para facilitar la lectura de la memoria actual, se cree conveniente presentar un resumen de cómo se estructuran los diferentes apartados que contiene.

En el apartado actual, Introducción, se presenta el TFM que se va a realizar, aclarando el objetivo perseguido, el contexto en que surge y los aspectos que motivaron su realización.

El apartado 2, Descripción de la técnica, se pretende dar a conocer, de manera objetiva, las características de la realidad representada, con rasgos tales como elementos que la componen, funcionalidad, utilidad, etc. En concreto, el apartado comienza describiendo el

El apartado 3, Entorno de trabajo, está dedicado a conocer las herramientas utilizadas para poder llevar a cabo el desarrollo y la implementación

El apartado 4, titulado Desarrollo de la aplicación, es el apartado principal de la memoria. En él, se detalla el proceso a seguir durante el desarrollo e implementación de la ”””””””””””””””””””””””” presentada en el proyecto, comenzando por los Primeros pasos a seguir, hasta llegar a presentar el Resultado final obtenido.

Por último, el apartado 5, está dedicado a las Conclusiones y evaluaciones surgidas del proyecto, los objetivos alcanzados y las líneas futuras de trabajo surgidas durante la realización de éste.

”””””””””””””””””””””” PRESCINDIBLE ””””””””””””””””””””

Además, se advierte al lector que la mayor parte de herramientas y documentación utilizada para la realización del PFC se encuentran redactadas en inglés, incluyendo términos que en el ámbito del desarrollo de aplicaciones no poseen traducción literal al español. Es por esto, que en la redacción de la memoria actual no se utiliza traducción, ni letras cursivas, ni comillas para referir a dichos términos.

”””

TODO - Dejar claro que este proyecto no busca innovar en ninguno de sus apartado, el objetivo perseguido será recopilar herramientas y poder disponer de ellas y ejecutarlas en un entorno ”autodeplegable”

Descripción de la Técnica

A good DevOps organization will free up developers to focus on doing what they do best: write software.

Rob Steward, 2015
Global Vicepresident at Verint-Systems.

Para comprender el desarrollo del trabajo aquí presentado, tal y como se ha llevado a cabo, se debe conocer la situación en que éste se desarrolla, la tecnología de la que se dispone y los elementos existentes y necesarios, de una manera objetiva.

Es por esto, que el apartado actual está orientado a conocer las características de la realidad representada y a introducir las bases tecnológicas del presente TFM, resaltando los conceptos más importantes.

Procesos de desarrollo en empresas TI

Pipelines y demás en empresas modernas... ¿Cómo se hacen las cosas? y comentar de qué forma no se va a interferir de manera destructiva en este proceso... y quizás esto último en otro apartado... ¿Incluyo aquí cómo puede ser ”más o menos” un día de trabajo DevOps?

(Software de) Integración Continua (Continuous Integration, CI)

OWASP

Breve introducción a lo que es, explicando que las herramientas utilizadas están recomendadas por ellos o, al menos, donde encontrar las recomendaciones de herramientas

CVE

Ruby y NodeJS

Breve presentación a la importancia de estos lenguajes de programación, comentando que lo que se ha hecho aquí es extensible a otros lenguajes, con otras herramientas similares.
Concepto dependencias de código.

Bundler is the de facto way of managing dependencies. It provides, among other things, a clear way of specifying required libraries and their versions, by keeping track of everything for you through Gemfile and (for applications) Gemfile.lock. Exactly the sort of information you'd need when checking for security vulnerabilities.

Análisis estático de dependencias de código

¿Qué es y cómo funciona?

Análisis estático de contenedores

¿Qué es y cómo funciona?

https://www.linuxadictos.com/docker-i-que-es-conociendo-la-ballena.html

http://www.javiergarzas.com/2015/07/que-es-docker-sencillo.html

https://www.redeszone.net/2016/02/24/docker-funciona-la-virtualizacion-contenedores/

La nube es cada vez más grande, más potente, cuenta con más usuarios que hacen uso de ella al mismo tiempo y, además, permite la ejecución de aplicaciones cada vez más potentes, por lo que, para garantizar el correcto funcionamiento de esta, tanto en el presente como en el futuro, es necesario utilizar una plataforma que optimice los recursos lo mejor posible y, al mismo tiempo, sea lo más escalable posible con el fin de poder ampliar sus características de forma sencilla cuando sea necesario.

La nube es sinónimo de virtualización. Ejecutar un sistema operativo virtual por cada instancia de una aplicación es un proceso muy pesado y poco optimizado, a la vez que lento. Por ello, la comunidad Linux ha trabajado en el concepto de contenedores, una nueva forma de optimizar recursos creando pequeños espacios virtuales de las aplicaciones necesarias cargando solo el núcleo de la aplicación y las dependencias, pero funcionando siempre sobre un único kernel, o sistema operativo. (Esquema de esta WEB)

Comunicaciones seguras (SSH e intercambio de Tokens)

¿Qué se ha hecho hasta ahora?

Entorno de trabajo

Technology is nothing. What's important is that you have a faith in people, that they're basically good and smart, and if you give them tools, they'll do wonderful things with them.

Steve Jobs, 1994
Businessman

Antes de comenzar el proceso de desarrollo de la aplicación es necesario preparar un entorno adecuado de trabajo, es decir, un conjunto de herramientas hardware y software que permitan llevar a cabo el proyecto con la mayor comodidad y precisión posible.

La correcta elección de un entorno de trabajo adecuado es fundamental a la hora de abordar cualquier tipo de proyecto, ya que el éxito o fracaso, o al menos la eficiencia del proceso de desarrollo del mismo, va a depender en gran medida de dicho entorno utilizado. El apartado actual presenta el entorno de trabajo utilizado para la realización de este TFM.

Git y GitHub

Los sistemas de control de versiones son programas cuyo principal objetivo es controlar los cambios producidos en el desarrollo de cualquier tipo de software, permitiendo conocer el estado actual de un proyecto, las personas que intervinieron en ellos, etc. Un buen control de versiones es tarea fundamental para la administración de un proyecto de desarrollo de software en general[1]. Git es uno de los sistemas de control de versiones más populares entre los desarrolladores, es gratuito, open source, rápido y eficiente, aunque gran parte su popularidad es debido a GitHub(Figura 3.7), un excelente servicio de alojamiento de repositorios de software que ofrece un amplio conjunto de características de gran utiilidad para el trabajo en equipo.

A continuación se muestran algunas de las características que han llevado a GitHub a ser tan valorado entre los desarrolladores[13]:

- Permite versionar el código, es decir, guardar en determinado momento los cambios realizados sobre un archivo o conjunto de archivos con la oportunidad de tener acceso al historial de cambios al completo, bien para regresar a alguna de las versiones anteriores o bien para poder realizar comparaciones entre ellas.

- Gracias a la gran cantidad de repositorios de SW públicos que alberga, es posible leer, estudiar

Figura 3.1 Logotipo de GitHub.

y aprender de el código creado por miles de desarrolladores en el mundo, permitiendo incluso la oportunidad de adaptarlos a las necesidades propias de cada desarrollador, sin alterar el original y realizando una copia o fork[1] de este.

- Tras haber realizado un fork de un proyecto y haber realizado algunos ajuste, introducido alguna mejora o arreglado algún problema que este pudiera contener, es posible integrar los cambios realizados al proyecto original (previa supervisión de su propietario, administrador o alguno de sus colaboradores), por lo que un repositorio puede llegar a ser construido mediante la contribución una gran comunidad de desarrolladores.

- GitHub posee un sistema propio de notificaciones con el que poder estar informado de lo que ocurre en torno a un repositorio concreto, ya sea privado a la compañía o público a la comunidad.

- GitHub trae incorporado un visor de código, mediante el cual (y a través del navegador) es posible consultar el contenido de un archivo determinado, con la sintaxis correspondiente al lenguaje utilizado y sin necesidad de descargar una copia del mismo.

- Cada repositorio de SW albergado en GitHub cuenta con su propio seguimiento de incidencias, con un elaborado sistema de tickets, de manera tal que cualquier colaborador (o usuario en general) pueda reportar algún problema encontrado en la utilización el código o pueda simplemente sugerir nuevas características para que sean implementadas.

- Al ser una plataforma web es totalmente independiente al SO utilizado, siendo por otro lado Git compatible con los principales sistemas actuales: Linux, Windows, OSX.

- GitHub es gratuito e ilimitado para repositorios de proyectos públicos, sólo aquellos usuarios que deseen mantener proyectos en privado deberán pagar una cuota.

TODO - Una frase de cierre al apartado

Bundler-audit

Como ya fue comentado en el apartado 2 cada aplicación tiene sus dependencia, y estas a su vez pueden contener vulnerabilidades de seguridad. Encontrar las vulnerabilidades de seguridad que presenta una aplicación es una tarea necesaria y tediosa, que de ser obviada no impedirá que la aplicación generada siga siendo ejecutada como si todo estuviera funcionando en perfectas condiciones, pero que ocultará agujeros en la aplicación que podrán ser utilizados en diversa manera por algún usuario malintencionado.

Para cualquier aplicación escrita con Ruby y en ausencia de alguna herramienta automatizada de análisis de vulnerabilidades, el desarrollador del código deberá estar suscrito a cada lista de correo relacionada

[1] Copia exacta en crudo del repositorio original que podrá ser utilizada como un repositorio git cualquiera

con anuncios de seguridad de dependencias y realizar un seguimiento exclusivo de vulnerabilidades y actualizaciones de seguridad de cada una de las dependencias incluidas en la aplicación, para cada aplicación en la que participe[12].

Por el contrario, todo este proceso puede ser automatizado en Ruby gracias a bundler-audit[14]del grupo de colaboradores Rubysec, un verificador a nivel de parche para Bundler con las siguientes características:

- bundler-audit analiza las vulnerabilidades en las versiones de las gemas contenidas en el archivo de dependencias *Gemfile.lock* de la aplicación.
- Analiza las fuentes de dependencias que puedan ser inseguras (http://).
- Permite especificar avisos de seguridad que serán ignorados en el análisis, bien por ser un riesgo asumido por el desarrollador, una vulnerabilidad ya conocida y en la que se está actualmente trabajando o cualquier otro motivo.
- No requiere de conexión a internet para cada ejecución que se realice del análisis.
- Funcionando cruzando la información de dependencias recogidas del fichero *Gemfile.lock* con una lista de vulnerabilidades conocidas[15], basada en información pública existente en bases de datos como CVE.

El código 3.1 muestra un ejemplo del resultado obtenido a la salida del terminal de comandos al realizar un análisis estático de vulnerabilidades con bundler-audit al archivo *Gemfile.lock* de un proyecto:

Código 3.1 Ejemplo de uso de bundler-audit.

```
$ bundle audit
Name: actionpack
Version: 3.2.10
Advisory: OSVDB-91452
Criticality: Medium
URL: http://www.osvdb.org/show/osvdb/91452
Title: XSS vulnerability in sanitize_css in Action Pack
Solution: upgrade to ~> 2.3.18, ~> 3.1.12, >= 3.2.13

Name: actionpack
Version: 3.2.10
Advisory: OSVDB-89026
Criticality: High
URL: http://osvdb.org/show/osvdb/89026
Title: Ruby on Rails params_parser.rb Action Pack Type Casting Parameter
    Parsing Remote Code Execution
Solution: upgrade to ~> 2.3.15, ~> 3.0.19, ~> 3.1.10, >= 3.2.11

Name: activerecord
Version: 3.2.10
Advisory: OSVDB-90072
Criticality: Medium
URL: http://direct.osvdb.org/show/osvdb/90072
Title: Ruby on Rails Active Record attr_protected Method Bypass
Solution: upgrade to ~> 2.3.17, ~> 3.1.11, >= 3.2.12

Name: activerecord
Version: 3.2.10
Advisory: OSVDB-89025
```

```
Criticality: High
URL: http://osvdb.org/show/osvdb/89025
Title: Ruby on Rails Active Record JSON Parameter Parsing Query Bypass
Solution: upgrade to ~> 2.3.16, ~> 3.0.19, ~> 3.1.10, >= 3.2.11

Unpatched versions found!
```

Además, bundler-audit permite actualizar la base de datos ruby-advisory-db yanalizar el fichero *Gemfile.lock* desde el mismo comando, habilidad que resulta de gran utilidad para su ejecución en sistemas de Integración continua (IC), como muestra el código 3.2:

Código 3.2 Ejecutando bundler-audit tras la actualización de las vulnerabilidades conocidas.

```
$ bundle audit check --update
```

Por último, y como ya se ha mencionado con anterioridad en este apartado, es posible ignorar advertencias especificadas (código 3.3) por el usuario:

Código 3.3 Ignorar vulnerabilidades con bundler-audit.

```
$ bundle audit check --ignore OSVDB-108664
```

TODO - Cierre del apartado. de Rubysec para proyectos Ruby

Nsp

Nsp es la principal herramienta de interfaz de línea de comandos de Node Security Platform y es una de las herramientas más utilizadas para monitorizar la seguridad en aplicaciones node, permitiendo auditar el archivo *package.json* de dependencias de la misma.

Emplazamiento de Imagen

Figura 3.2 Escaneo de vulnerabilidades en las dependencias de Node JS.

Su utilización es simple, permitiendo incluso ignorar vulnerabilidades que puedan estar siendo tratadas en el momento del análisis y realizando consultas contra una base de datos de vulnerabilidades mantenida

por Node Security Platform, que contiene información recopilada de múltiples fuentes (CVE entre ellas). Basta con *"nsp check"* para analizar las vulnerabilidades de la aplicación, recibiendo un resultado como el mostrado en la Figura 3.2.

Docker

Figura 3.3 Logotipo de Docker.

Docker (Figura 3.3) es una herramienta de virtualización diseñada para aportar beneficios a desarrolladores, testers y administradores de sistemas, creando contenedores ligeros y portables para que las aplicaciones puedan ser ejecutadas en cualquier máquina, con sólo tener docker instalado e independientemente del SO de la propia máquina que lo contenga[5]. Esta plataforma de código abierto hace uso de las funciones de aislamiento de recursos que provee el kernel de Linux para dar lugar a contenedores independientes, dentro de los cuales se ejecutará una única aplicación con sus respectivas dependencias, funcionando siempre con este único kernel, en lugar de virtualizar uno por cada contenedor o máquina virtual, como muestra la Figura 3.4.

Figura 3.4 Pila de funcionamiento de docker.

Gracias al uso de docker dos desarrolladores no tendrán que, por ejemplo, preocuparse de la versión Java que cada uno tenga instalado en su propia máquina durante el desarrollo de la aplicación, ya que está podrá ser enviada de uno a otro en el interior de un contenedor, que funcionará de la misma manera en cualquiera de los dos entornos.

Además, virtualizar con docker ofrece una serie de ventajas respecto a hacerlo con máquinas virtuales convencionales[17]:

- **Portabilidad.** Los contenedores son portables, por lo que pueden ser trasladados fácilmente a cualquier otro equipo con Docker sin tener que volver a configurar nada.
- **Ligereza.** Al no virtualizar un sistema completo, sino solo lo necesario, el consumo de recursos es muy inferior.
- **Autosuficiencia.** Docker se encarga de todo de organizarlo todo, por lo que los contenedores tan solo deben tener lo necesario para que la aplicación funcione.

Un sistema de contenedores Docker lo componen principalmente 5 elementos fundamentales:

- **Demonio:** proceso principal de la plataforma.
- **Cliente:** binario que constituye la interfaz al usuario y le permite interactuar con el Demonio.
- **Imagen:** Plantilla utilizada para crear el contenedor de la aplicación que se va a ejecutar en su interior.
- **Registros:** Directorios donde se almacenan las imágenes, tanto de acceso público como privado, similares a los de Linux, donde los usuarios publican sus propios contenedores de manera que aquellos otros usuarios que los necesiten puedan descargarlos directamente desde allí.
- **Contenedores:** Donde se almacena todo lo necesario (librerías, dependencias, binarios de la aplicación, etc.) para que la aplicación pueda ejecutarse de forma aislada.

Actualmente existen multitud de empresas que utilizan a docker como sistema de contenedores en sus centros de datos (Spotify, eBay, PayPal, etc.) y cuenta con el apoyo de grandes compañías de internet como Amazon o Google, lo que permite que Docker se encuentre en un proceso continuo de crecimiento y mejora.

Clair y Clairctl

Clair (Figura 3.5) es un proyecto de código libre desarrollado por CoreOS para el análisis estático de vulnerabilidades en contenedores de aplicaciones, que funciona de la siguiente manera[4]:

Figura 3.5 Clair.

1. A intervalos regulares, Clair realiza una ingesta de metadatos de vulnerabilidad desde un conjunto configurado de fuentes (entre ellos CVE) y los almacena en una base de datos PostgreSQL.
2. Un cliente configurado para utilizar la Interfaz de Programación de Aplicaciones (API) de Clair indexa y escanea sus imágenes de contenedores por capas,creando una lista de características presentes en la imagen y almacenándolas en la base de datos.

3. El cliente solicita a través de la API a la base de datos las vulnerabilidades conocidas para para una imagen en particular, correlacionando las vulnerabilidades y características de la imagen, de manera que esta no requiere un escaneo completo para cada petición.

4. En caso de que se produzcan actualizaciones para los metadatos de vulnerabilidad, se enviará una notificación que alertará a los sistemas de que se ha producido un cambio, siendo escaneada de nuevo la imagen la próxima vez que sea solicitado un informe.

Por otro lado, Clairctl es uno de los clientes más utilizados para alcanzar la API de Clair y se caracteriza por ser una herramienta ligera de interfaz de comando escrita con el lenguaje de programación desarrollado por Google, Go. Clairctl es capaz de hacer de puente entre registros[2] de imágenes tales como Docker Hub, Docker Registry o Quay.io (registro de imágenes de CoreOS). La Figura 3.6 muestra el esquema resultado de utilizar ambas herramientas para el análisis estático de contenedores de aplicaciones.

Emplazamiento de Imagen

Figura 3.6 Esquema de escaneo y detección de vulnerabilidades en imágenes Docker.

Slack

Jenkins

Jenkins (Figura 3.7) es una herramienta autónoma de código abierto que puede utilizarse para automatizar todo tipo de tareas, como la construcción, prueba y despliegue de software. Jenkins puede ser instalado a través de paquetes de sistemas nativos, Docker, o incluso puede ser ejecutado de manera independiente en cualquier máquina con el entorno de ejecución de Java instalado[11].

Jenkins posee, entre otras, las siguientes ventajas:

- **Continuous Integration and Continuous Delivery**: As an extensible automation server, Jenkins can be used as a simple CI server or turned into the continuous delivery hub for any project.

- **Easy installation**: Jenkins is a self-contained Java-based program, ready to run out-of-the-box, with packages for Windows, Mac OS X and other Unix-like operating systems.

[2] Un registro es un sistema de almacenamiento y distribución de imágenes de Docker, público o privado, que serán almacenadas en diferentes versiones, cada una con su correspondiente etiqueta.

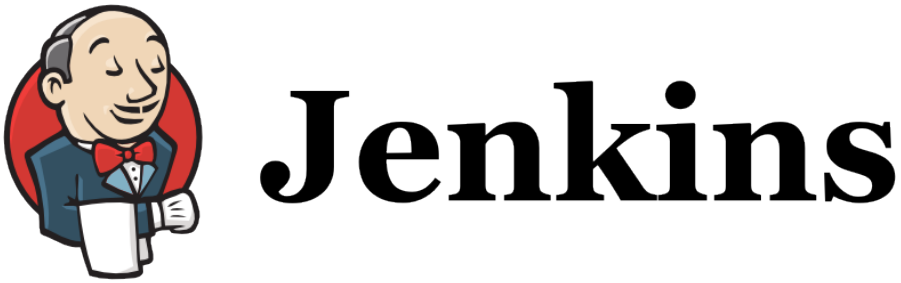

Figura 3.7 IC con Jenkins.

- **Easy configuration**: Jenkins can be easily set up and configured via its web interface, which includes on-the-fly error checks and built-in help.
- **Plugins**: With hundreds of plugins in the Update Center, Jenkins integrates with practically every tool in the continuous integration and continuous delivery toolchain. If a plugin does not exist, you can code it and share with the community.
- **Extensible**: Jenkins can be extended via its plugin architecture, providing nearly infinite possibilities for what Jenkins can do.
- **Distributed**: Jenkins can easily distribute work across multiple machines, helping drive builds, tests and deployments across multiple platforms faster.
- It is an open source tool with great community support.
- It provides continuous integration pipeline support for establishing software development life cycle work flow for your application.
- It also provides support for scheduled builds & automation test execution.
- You can configure Jenkins to pull code from a version control server like GitHub, BitBucket etc. whenever a commit is made.
- It can execute bash scripts, shell scripts, ANT and Maven Targets.
- It can be used to Publish results and send email notifications.

Pipeline - A user-defined model of a continuous delivery pipeline, for more read the Pipeline chapter in this handbook.
Debo poner también la imagen de una pipeline de trabajo con verdes y rojos.

Desarrollo de la solución

Epígrafe.

AUTOR, AÑO

Apartado Desarrollo de la solución.

Conclusiones

Security is not a line in the sand. Protecting your business, customers, citizens' data, should be always your number one priority

Dr. Werner Vogels, 2017
CTO at Amazon.com

Apartado Conclusiones.

Índice de Figuras

Índice de Códigos

Bibliografía

[1] Israel Alcázar, *Introducción a Git y Github*, Junio 2014, Enlace.

[2] Elena Arrieta, *DevOps: la tecnología y el negocio deben hablar el mismo idioma*, Mayo 2017, Enlace.

[3] Claranet, *DevOps: qué es y cómo lo aplicamos*, Accedido: Agosto de 2017, Enlace.

[4] coreos/clair, Accedido: Septiembre de 2017, Enlace.

[5] Ana M. del Carmen García Oterino, *¿Qué es Docker? ¿Para qué se utiliza? Explicado de forma sencilla*, Julio 2015, Enlace.

[6] Alan R. Earls, *Construir un entorno DevOps con microservicios y contenedores*, Diciembre 2015, Enlace.

[7] Hewlett Packard Enterprise, *Application Security and DevOps*, Octubre 2016, Enlace.

[8] Docker Inc., Accedido: Agosto de 2017, Enlace.

[9] GitHub Inc., Accedido: Agosto de 2017, Enlace.

[10] Consultor IT, *Estudio CA Technologies sobre la importancia de la Agilidad y DevOps en el desarrollo de software [pdf de 20 pgs.]*, Enero 2017, Enlace.

[11] Jenkins, Accedido: Agosto de 2017, Enlace.

[12] Adam Prescott, *Automated vulnerability checking with bundler-audit and Travis*, Junio 2015, Enlace.

[13] Erlinis Quintana, *10 razones para usar Github*, Junio 2015, Enlace.

[14] rubysec/bundler audit, Accedido: Septiembre de 2017, Enlace.

[15] rubysec/ruby-advisory db, Accedido: Septiembre de 2017, Enlace.

[16] CA Technologies, *Accelerating Velocity and Customer Value with Agile and DevOps*, Enero 2017, Enlace.

[17] Rubén Velasco, *¿Qué es Docker? ¿Para qué se utiliza? Explicado de forma sencilla*, Febrero 2016, Enlace.

Glosario

CI	Continuous Integration III, V, VII, 3, 5
CVE	Common Vulnerabilities and Exposures III, V, 2, 9
DevOps	Development and Operations III, V, 1, 2
IC	Integración continua 10, 11, 17
OS	Operative System V
QA	Quality Assurance III, V
SaaS	software como servicio III, V
SO	Sistema Operativo III, 8
SW	software III, V, 3, 7, 8
TFM	Trabajo Fin de Máster III, 1–3, 5, 7
TI	Tecnología de la información VII, 5