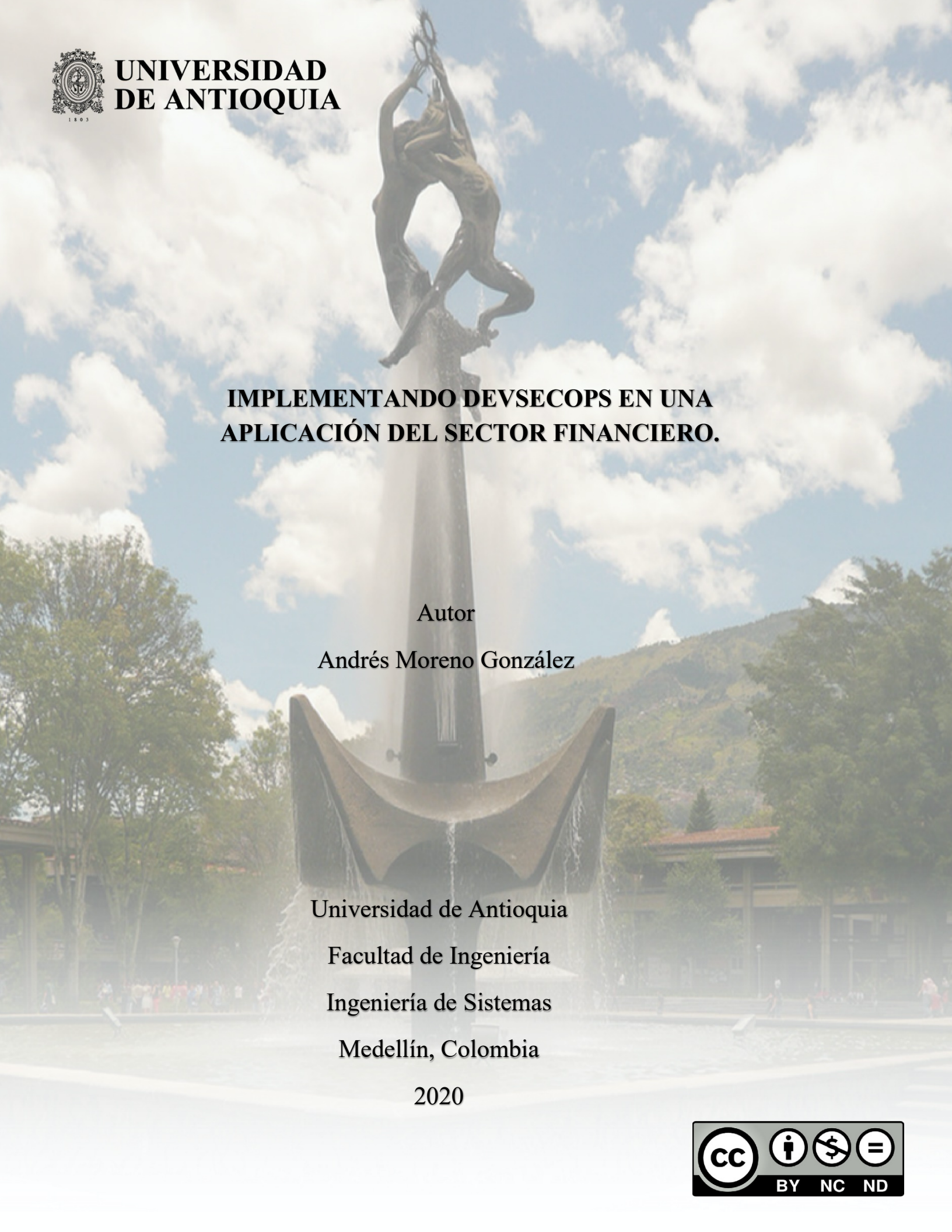


UNIVERSIDAD DE ANTIOQUIA

IMPLEMENTANDO DEVSECOPS EN UNA APLICACIÓN DEL SECTOR FINANCIERO.

Autor

Andrés Moreno González

Universidad de Antioquia

Facultad de Ingeniería

Ingeniería de Sistemas

Medellín, Colombia

2020

Implementando DevSecOps en una aplicación del sector financiero

Andrés Moreno González

Informe de práctica presentado como requisito parcial para optar al título de:
Ingeniero de Sistemas

Asesores:
Carlos Mauricio Duque Restrepo, Ingeniero de Sistemas
David Vásquez Rivera, Ingeniero de Sistemas

Universidad de Antioquia
Facultad de ingeniería
Ingeniería de sistemas
Medellín, Colombia
2020

IMPLEMENTANDO DEVSECOPS EN UNA APLICACIÓN DEL SECTOR FINANCIERO.

Informe práctica empresarial

Autor:
Andrés Moreno González

Asesores:
Carlos Mauricio Duque
David Vasquez Rivera

Universidad de Antioquia

Facultad de Ingeniería

Departamento de sistemas

Medellín, Colombia

Septiembre 2020

Índice

Índice de figuras

Índice de tablas

1. Resumen

La forma en la que desarrollamos software se encuentra en constante evolución, pasamos de pasos bien definidos y ejecutados secuencialmente, a un desarrollo que debe ser rápido y con altos estándares de calidad, buscando satisfacer a un mercado que se transforma y cambia constantemente hasta el día de hoy. Sin embargo, algunas veces la seguridad no se considera como un factor primordial, en consecuencia las compañias sufren pérdidas invaluables. Por esto, actualmente en la industria, vemos el nacimiento de nuevas metodologías como DevSecOps. Este marco de trabajo pretende que todas las personas involucradas en el ciclo de desarrollo de software tomen conciencia de la importancia de la seguridad, de lo que puede hacer esta para evitar quedar expuestos ante posibles ataques. El objetivo de este trabajo es incluir herramientas que permitan detectar vulnerabilidades en etapas tempranas del desarrollo, a través de la implementación de escaneos de tipo SAST, DAST y de detección de vulnerabilidades de Docker en los Pipelines de integración continua, usados actualmente en el proyecto.

Palabras clave: DevSecOps, DevOps, Desarrollo Ágil, SAST, DAST, Contendores.

2. Introducción

Desarrollar software es complejo, tanto que en 1994 el grupo Standish publicó el reporte del caos [9] donde mostraba cifras alarmantes: tan sólo el 16 % de los proyectos de software eran exitosos, el 53 % llegaban a ser operacionales pero no cumplían con la planeación inicial, por último, el 31 % eran cancelados. A través de los años podemos ver (Tabla 1 [9] [16]) que a pesar de que las cifras de proyectos fallidos han decrecido, sigue siendo difícil cumplir con el tiempo y el presupuesto estimados.

Año	Exitoso(%)	No cumple con planeación(%)	Fallido(%)
1994	16	53	31
1996	27	33	40
1998	26	46	28
2000	28	49	23
2004	29	53	18
2006	35	46	19
2009	32	44	24
2011	29	49	22
2013	31	50	19
2015	29	52	19

Tabla 1: Cifras del reporte del caos desde 1994 hasta 2015.

La forma en la cual se desarrollaba software en los años 90 no era efectiva, con frecuencia se concentraba en llenar interminables páginas de documentación y planeación, más que en entregar productos de valor para las personas. Métodos como Waterfall y RUP(Rational Unified Process) no permitían a los equipos reaccionar rápidamente al cambio, y hacían que cumplir con las restricciones de tiempo y presupuesto, fuera difícil.

Esta es la razón por la cual en el año 2001 un grupo de 17 expertos de la industria se reúne para trazar unos nuevos valores y principios, estos permitirían a los equipos de desarrollo trabajar de manera eficiente y ser capaces de responder al cambio [4] . Es así como nació el manifiesto ágil, el cual guía los principios usados en la industria para llevar a cabo la mayor parte de los proyectos de software hasta el día de hoy [14].

Los principios que define el manifiesto ágil son los siguientes[4] :

1. Individuos e interacciones sobre procesos y herramientas.
2. Software funcionando sobre documentación extensiva.
3. Colaboración con el cliente sobre negociación contractual.
4. Respuesta ante el cambio sobre seguir un plan.

A pesar de que el desarrollo ágil quiere llevar software a producción de manera rápida, en la práctica es difícil debido a que los equipos de operaciones

(Ops), quienes coordinan el despliegue y atención de incidentes en producción, normalmente no están alineados con el equipo de desarrollo (Dev) [14]. Como consecuencia de esto, nació el término DevOps, este se refiere a un conjunto de valores, prácticas y herramientas que permiten la colaboración entre los equipos de operaciones y desarrollo [20].

A medida que la cultura DevOps se ha vuelto más popular, muchas empresas han empezado a adoptar prácticas asociadas a esta, pero la seguridad no ha sido una prioridad. Según Gartner menos del 20 % de los arquitectos de seguridad han sido involucrados en las iniciativas de DevOps de sus empresas [23]. Con el fin de integrar seguridad en las propuestas de DevOps surge DevSecOps, promoviendo la colaboración entre desarrolladores y el equipo de operaciones, e incluyendo también expertos en seguridad.

Uno de los clientes de PSL (Productora de Software S. A.) incluye en su proceso de desarrollo prácticas de la cultura DevOps, pero no abarca verificaciones estrictas de seguridad. Este cliente pertenece al sector financiero, debido a los acuerdos de confidencialidad firmados no es posible revelar el nombre de esta empresa, por lo que ahora en adelante nos referiremos a ella como Financial Company. Al advertir la carencia de verificaciones de seguridad se propone este proyecto, con el fin de detectar de manera temprana vulnerabilidades de seguridad en el software que entrega a sus clientes.

A nivel de infraestructura la aplicación cuenta con restricciones de acceso a las máquinas donde se ejecuta, y prácticas de seguridad de los recursos usados en la plataforma de la nube sobre la cual corre, además, Financial Company tiene una persona delegada a mantener la confidencialidad de los datos y a permitir o negar el acceso a estos.

A pesar de tener estas garantías durante el desarrollo del software, no se tienen instrumentos que permitan identificar vulnerabilidades en los contenedores usados o en el código que se lleva a producción. Es por esto que se propone la integración de herramientas(SAST o DAST) que contribuyan a detectar de manera temprana posibles fallos de seguridad.

3. Estado del arte

A pesar de que en la industria la practica de DevSecOps ha ido ganando popularidad y terreno en los ultimos años, en la academia es un tema que no ha sido tan explorado. Es posible encontrar algunas implementaciones (similares a la propuesta de esta práctica empresarial) como [17], [25], [5], [21] , una guía sobre como implementar DevSecOps publicada por la Universidad Carnegie Mellon [22] y un estudio que compara los tiempos de ejecución entre Pipelines, que usan herramientas para la detección de vulnerabilidades, y Pipelines que no las usan [3].

En el caso de las implementaciones logramos observar que todas incorporan tanto herramientas de SAST como de DAST, y en aquellas que emplean contenedores se hace uso de herramientas para el escaneo de estos. En el caso de [17] y [5] encontramos dos propuestas muy similares pero con la diferencia de que

[17] se centra en el desarrollo de un SDLC seguro, mientras que [5] se interesa en automatizar controles de seguridad en ciertas fases del SDLC. Es decir, el enfoque presentado por [17] es integral, mientras que [5] es parcial. Otra diferencia entre ambas propuestas son las herramientas seleccionadas para realizar los análisis necesarios (Tabla 2) y el uso de un estándar para ejecutar la propuesta en el caso de [17] el seleccionado es IEE 830 - 1998.

Tipo de herramienta	[5]	[17]
SAST	Jenkins Warnings Next Generation	SonarQube
DAST	OWASP ZAP	OWASP ZAP
Docker Scanner	Anchore Engine	ClairOS
Dependency Scanner	N/A	OWASP Dependency Checker
Sistema de Integración	Jenkins	Jenkins

Tabla 2: Herramientas usadas por [17] y [5].

La información que provee [25] sobre su implementación de DevSecOps es mucho más reducida y se limita a mencionar las herramientas usadas, sin aportar detalles de la implementación de las mismas. En este caso, tenemos que como herramienta DAST utiliza OWASP ZAP y como herramientas SAST utiliza Checkstyle, PMD, y Find Bugs, la ejecución de ambos análisis es realizado en paralelo mediante un Pipeline de Jenkins.

En el caso de [21] se presentan una serie de buenas practicas, para ayudar a implementar seguridad en los desarrollos de robótica. Mostrando primero los conceptos relacionados al DevOps y como se han aplicado en la robótica, y después, incluyendo en estos conceptos las prácticas de seguridad con el fin de obtener un producto más seguro. El trabajo presentado por [21] puede ser resumido con la Figura 1.

La guía presentada por la universidad de Carnegie Mellon [22], sirve para implementar DevSecOps en entornos altamente regulados como lo son los departamentos de defensa o otras instituciones gubernamentales. Provee información acerca de lo que es DevSecOps, sus principios y sus beneficios. Describe las actividades necesarias para realizar su implementación, es decir, la adopción de la cultura DevSecOps, el despliegue de tecnologías y como se deben adaptar los procesos existentes. Termina siendo una buena referencia para la consulta de conceptos y principios, no soóo para los ambientes altamente regulados sino para cualquier organización o persona interesada en estas prácticas.

Por último [3] trata de responder la pregunta: ¿Cómo se afecta el tiempo de ejecución en un Pipeline de CI/CD al incrementar el número de pruebas de seguridad?. Se hace uso de un Pipeline implementado en Azure y se agregan herramientas para análisis de dependencias(SAST y DAST). La aplicación sobre la cual se realizaron las pruebas fue OWASP Juice Shop, esta fue diseñada para ser vulnerable con el fin de ser utilizada en entrenamientos de seguridad. Al

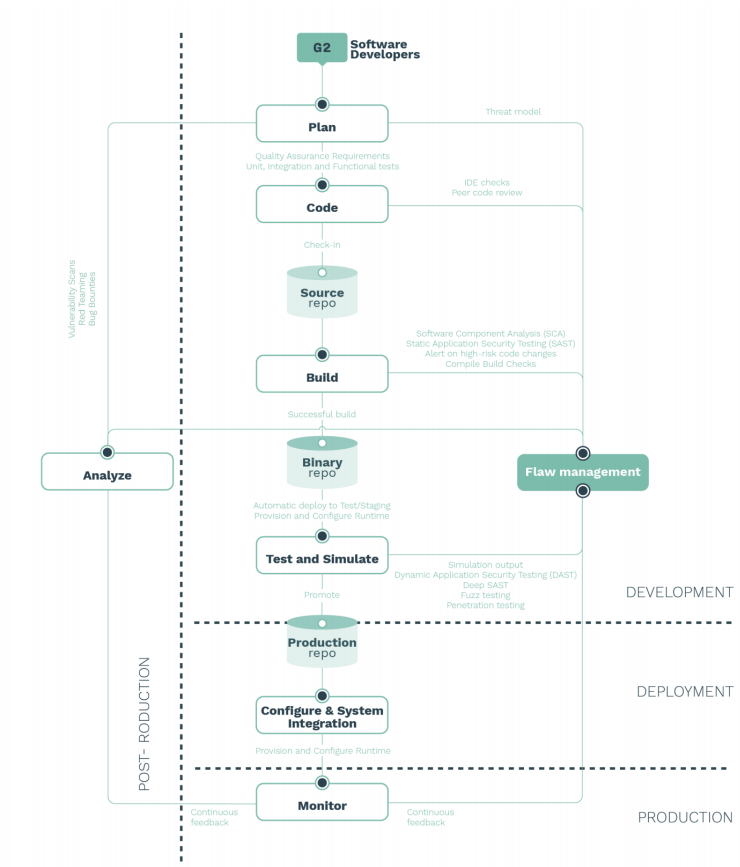


Figura 1: Diagrama de flujo sobre como aplicar DevSecOps en robótica. [21]

final, se concluye que las herramientas no cambian de manera significativa el tiempo de ejecución agregando solamente hasta 1/3 del tiempo original.

4. Objetivos

4.1. Objetivo general

Plantear e implementar una estrategia que garantice prácticas de seguridad teniendo como base el Application Security Verification Standard (ASVS), a través del uso de herramientas de análisis de código en los pipeline de integración continua.

4.2. Objetivos específicos

- Buscar y seleccionar herramientas que permitan el análisis de código y contenedores en los pipeline de integración continua.
- Establecer una estrategia de desarrollo, que provea una línea base de politicas y estandares de seguridad, teniendo en cuenta el ASVS y las posibles vulnerabilidades que las herramientas de SAST o DAST seleccionadas sean capaces de detectar.
- Configurar y desplegar las herramientas seleccionadas.

5. Marco teórico

Con el fin de entender un poco más los terminos usados en este informe se presenta esta sección.

5.1. Ciclo de vida de desarrollo de software (SDLC)

Antes de llegar a las manos de un usuario final el software pasa por una serie de pasos y atraviesa un proceso largo y complejo. La forma más común de hacer esto es atraves del Ciclo de vida de desarrollo de software el cual es un marco que define diferentes tareas que se deben realizar en cada uno de los pasos [19]. Estos pasos son: definición de requerimientos, diseño de software, fase de implementación o codificación, fase de verificación o pruebas, y finalmente el proceso de despliegue del producto. Desde una perspectiva tradicional la ejecución secuencial de estos pasos lleva a el desarrollo basado en la metodología de Cascada (Waterfall), esta metodología no tiene la capacidad de adaptarse a cambios o iterar de nuevo sobre un fase anterior. [13]

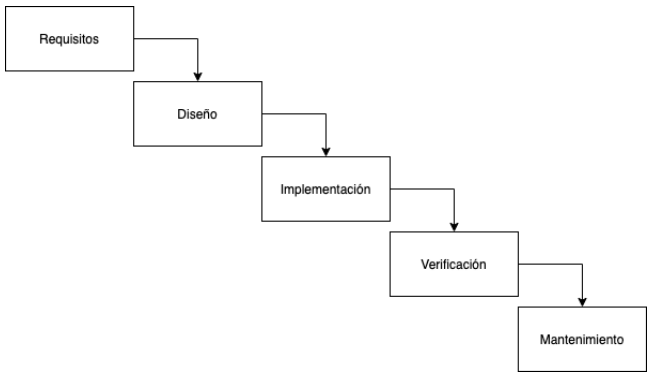


Figura 2: Metodología en cascada.
[13]

Si bien el SDLC puede convertirse en la metodología Cascada para desarrollar software cuando sus pasos son ejecutados de forma secuencial, la verdad es que al ser un marco de trabajo puede adaptarse a diferentes necesidades y desde diferentes puntos de vista. Algunas metodologías derivadas del SDLC son por ejemplo la ya menciona metodología en cascada, la metodología en espiral, el Modelo-V, el prototipado rápido y el método incremental [2].

5.2. Desarrollo Ágil

El desarrollo Ágil de software es un grupo de metodologías basadas en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan atraves de la colaboración entre equipos autogestionados y funcionales [2]. El desarrollo Ágil tiene sus inicios en el manifiesto Ágil, el cual es una colección de practicas publicadas por expertos de la industria los cuales no estaban conformes con la forma en la cual se desarrollaba software en los años 90 e inicios de los 2000. Los pequeños incrementos y la forma Ágil de trabajar permite a los clientes involucrarse más en el proceso de desarrollo. Por medio de estos incrementos, el cliente es capaz de entender a donde va el proyecto, haciendo más facil realizar temprano modificaciones y permitiendo ajustarse a un mercado cambiante [13].

Los cuatro valores del desarrollo ágil a fueron mencionados en la introducción de este reporte, estos valores inspiraron los siguientes 12 principios.

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.

2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.

3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.

4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.

5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.

6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.

7. El software funcionando es la medida principal de progreso.

8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.

9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.

10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.

11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.

12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

5.3. DevOps

El término DevOps tiene sus orígenes en 2008, cuando P. Debois presentó, en la conferencia Ágil en Toronto, la necesidad de resolver los conflictos que se producían entre el equipo de desarrollo y el equipo de operaciones cuando debían colaborar para alcanzar tiempos de respuesta rápidos a las necesidades de los clientes [6]. Desde ese momento el término ha tomado diferentes interpretaciones y su definición exacta no es clara[18][8]. A pesar de esto tomaremos la definición propuesta por [18] ya que toma como base una revisión bibliográfica y aplica los conceptos que se encuentran son más recurrentes. "DevOps es una metodología de desarrollo que apunta a cerrar la brecha entre el equipo de desarrollo y el equipo de operaciones, haciendo énfasis en la comunicación y la colaboración, la integración continua, el aseguramiento de la calidad y la entrega de valor con despliegues automatizados utilizando un conjunto de prácticas de desarrollo".

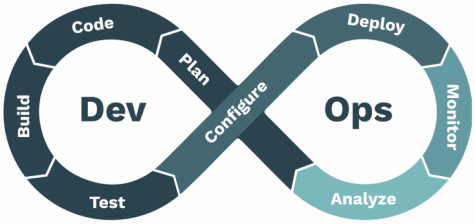


Figura 3: Ciclo de desarrollo DevOps.
[21]

Dependiendo de la fuente escogida como la fundación o guía para las practicas de DevOps , la metodología se caracteriza por sus 4 o 5 principios: Los CALM o CALMS. El acrónimo significa colaboración, automatización, compartir(sharing), y medición, con un principio adicional apoyo(lean) el cual se encuentra en algunas definiciones [13].

5.4. DevSecOps

Actualmente DevOps es una práctica bastante adoptada a pesar de esto la forma en la cual se integra la seguridad es una pregunta sin una respuesta clara[1]. Es por esto que nace DevSecOps es una iniciativa que apunta a adaptar las prácticas de seguridad para incluirlas en el proceso de DevOps, una definición más precisa la provee Gartner: "DevSecOps es la integración de la seguridad en el desarrollo DevOps de la forma más fluida y transparente posible. Idealmente, esto se hace sin reducir la agilidad o velocidad de los desarrolladores o sin requerir que los desarrolladores cambien sus herramientas en el ambiente de desarrollo"[12] .

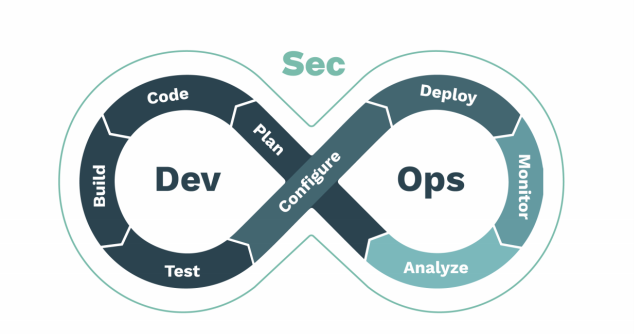


Figura 4: Ciclo de desarrollo DevSecOps.
[21]

Con la adición de Sec a DevOps, se realiza un enfasis en la integración de la seguridad. Justo como en DevOps, donde las separaciones entre los desarrolladores y operaciones se borran, DevSecOps apunta a borrar la separación entre estos equipos y los expertos en seguridad [13].

5.5. Integración continua

La integración continua es una práctica de desarrollo de software donde los miembros de un equipo integran su trabajo con frecuencia, generalmente cada persona lo hace al menos una vez al día, lo que lleva a múltiples integraciones por día. Cada integración se verifica mediante una compilación automatizada (incluida la prueba) para detectar errores de integración lo más rápido posible. Muchos equipos encuentran que este enfoque lleva a una reducción considerable de problemas de integración [10].

5.6. Contenedor

Un contenedor es una unidad estándar de software que empaqueta código y todas sus dependencias, de modo que la aplicación se ejecute de manera rápida y confiable de un ambiente de computación a otro [7].

5.7. Pipeline de despliegue

Un Pipeline de despliegue es una implementación automatizada de los procesos de construcción,despliegue, pruebas y salida a producción de una aplicación [15].

5.8. Source Code Analysis Tools (SAST)

En español comúnmente referido como herramientas de análisis de código estático, son herramientas diseñadas para analizar el código fuente o versiones compiladas del código para ayudar a encontrar vulnerabilidades de seguridad [24].

5.9. Dynamic Application Security Testing(DAST)

En español comúnmente referido como herramientas de análisis dinámico de software, son herramientas diseñadas para detectar vulnerabilidades de seguridad en aplicaciones corriendo [11].

6. Metodología

El desarrollo de la propuesta se realizará en 3 fases:

- **Búsqueda y selección de herramientas:** Durante esta fase se definirá, junto con el asesor interno, los criterios de selección de las herramientas de modo que se ajuste de la mejor manera posible a las necesidades actuales de Financial Company. Además, se realizará una búsqueda de los diferentes productos en el mercado y por último, se efectuará una calificación para escoger la herramienta que será seleccionada para integrar el ciclo de desarrollo.
- **Propuesta de integración en ciclo de desarrollo actual:** Durante esta fase se definirá en qué punto del ciclo de desarrollo se incluirá la herramienta, así como las políticas que deberán seguir los desarrolladores para garantizar un mínimo de seguridad.
- **Configuración y despliegue:** Durante esta etapa final, se realizarán las configuraciones necesarias de acuerdo con la propuesta definida en el paso anterior, desplegando la herramienta para su integración en el ciclo de desarrollo del equipo.

7. Resultados y análisis

Como resultado de estas prácticas empresariales se logró incluir una serie de herramientas que realizan análisis de seguridad en diferentes etapas del desarrollo, no sólo esto también se logró llevar a los desarrolladores talleres para que

tuvieran presentes las amenazas y las posibles formas de defensa a la hora de desarrollar código, adicionalmente, se realizó un análisis del estado actual de la aplicación tomando como base el estandar de seguridad ASVS(Application Security Verification Standard).

Las herramientas incluidas fueron escogidas teniendo en cuenta criterios como: el tipo de licencia que tenían, las vulnerabilidades que estas podían detectar, su popularidad (Medida en estrellas de Github), la cantidad de colaboradores (Medido en contribuidores en Github), si se tenía algún bloqueo técnico, y finalmente, si estas presentaban algún tipo de facilidad para ser incluidas en el Pipeline existente. Al final, las herramientas seleccionadas fueron: OWASP ZAP para realizar escaneo de tipo DAST, Clair para realizar el escaneo de las imagenes de docker y Sonarqube junto con PMD para realizar escaneo SAST.

El proyecto cuenta con una cantidad considerable de Pipelines de integración continua, esto debido a que la aplicación principal presenta una arquitectura basada en microservicios, por lo que es necesario tener pipelines para cada uno, así como para la aplicación que estos componen.

Dentro del repositorio de un microservicio se encontraran típicamente un Jenkinsfile, un Dockerfile y en caso de estar disponible un sonar-project.properties. El Jenkinsfile define el Pipeline de integración continua del repositorio, el Dockerfile describe la imagen del repositorio(adicionalmente se tiene un script sh para lanzar el microservicio), y el sonar-project.properties contiene la configuración para poder escanear el código mediante SonarQube. Para un microservicio el Jenkinsfile describe un Pipeline similar al siguiente:

Figura 5: Pipeline de un microservicio.

Al implementar la propuesta el Pipeline deberia ser similar al mostrado en la Figura 6.

Figura 6: Pipeline de un microservicio con análisis de contenedores y SAST.

En las etapas de Analysis e Image Scan se utilizan las herramientas PMD y Clair respectivamente. A continuación se muestran los resultados al ejecutar cada uno de los respectivos análisis en las Figuras 7 y 8.

En el caso de los resultados de los contenedores se encontró que para algunos microservicios se utilizaba como base una imagen hecha a la medida, por lo que no se detectaba ninguna vulnerabilidad. Caso contrario para las imágenes utilizadas en el servicio de elasticsearch, estos son los resultados que se muestran en la Figura 8.

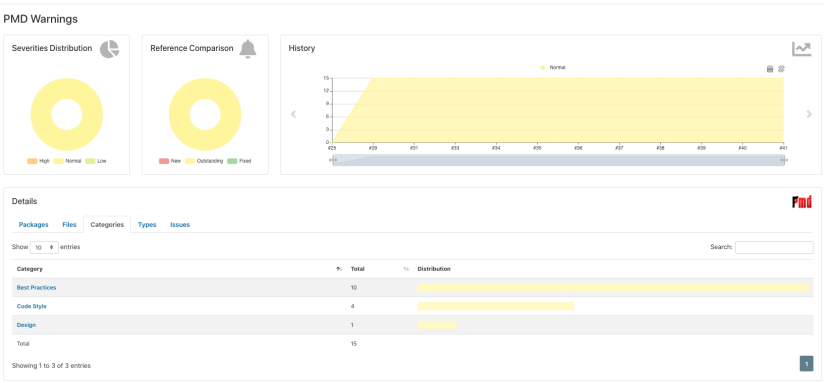


Figura 7: Resultados de SAST.

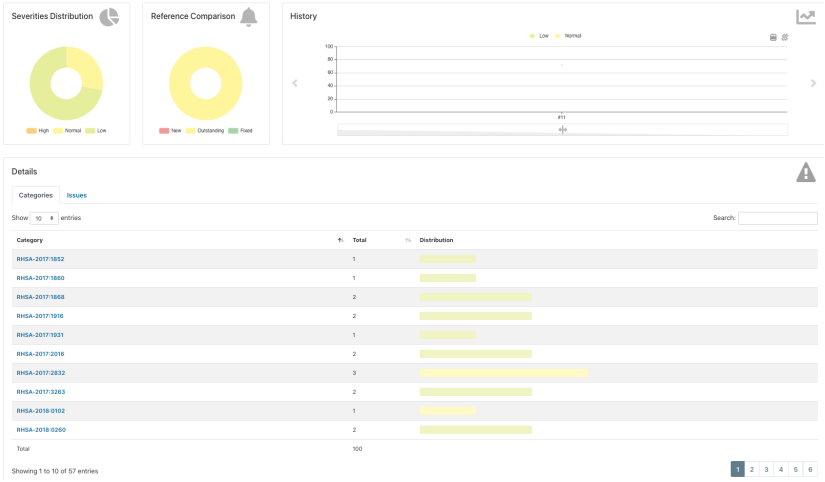
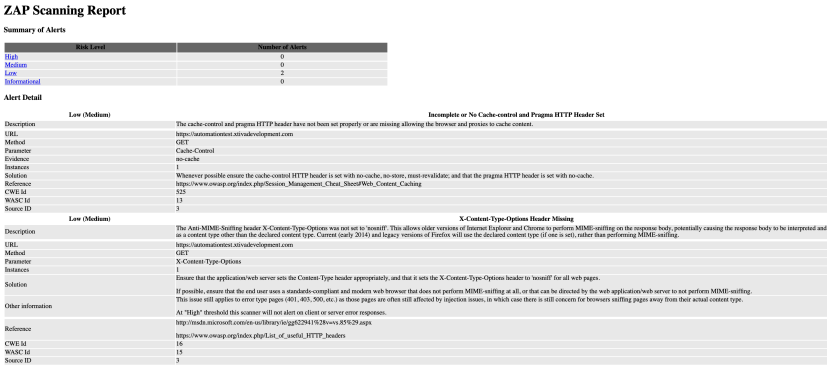


Figura 8: Resultados de análisis de contenedor de docker.

Los análisis de tipo DAST no se han logrado coordinar de manera adecuada para conseguir una ejecución completa, debido a que es más invasivo. Sólo se ha logrado un análisis de tipo araña (escaneo para descubrir automáticamente URLs en un sitio) Figura 9 .

Con la inclusión de estas herramientas en los Pipeline de integración continua de los microservicios se lograron detectar las siguiente cantidades de vulnerabilidades(ver Tabla 3).

Las herramientas son accionadas al realizar commits en el repositorio central, en el caso del análisis de docker y SAST son los commits realizados en cada microservicio y en el caso de DAST son los commits realizados en el repositorio que describe las versiones de la aplicación.



ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	0
Low	2
Informational	0

Alert Detail

Low (Medium)	Incomplete or No Cache-control and Pragma HTTP Header Set
Description	The cache-control and pragma HTTP header have not been set properly or are missing allowing the browser and proxies to cache content.
URL	https://automationtest.xtivadevelopment.com
Method	GET
Parameter	Cache-Control
Evidence	no-cache
Instances	1
Solution	Whenever possible ensure the cache-control HTTP header is set with no-cache, no-store, must-revalidate; and that the pragma HTTP header is set with no-cache.
Reference	https://www.owasp.org/index.php/Session_Management_Cheat_Sheet#Web_Content_Caching
CWE Id	525
WASC Id	13
Source ID	3

Low (Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	https://automationtest.xtivadevelopment.com
Method	GET
Parameter	X-Content-Type-Options
Instances	1
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Other information	This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php/List_of_useful_HTTP_headers
CWE Id	16
WASC Id	15
Source ID	3

Figura 9: Resultados de análisis de OWASP ZAP.

Tipo de herramienta	Vulnerabilidades(#)
SAST	45
DAST(Web scan mode)	2
Docker Scanner	125

Tabla 3: Cantidad de vulnerabilidades encontradas por tipo de herramienta.

8. Conclusiones

En definitiva, DevSecOps es una gran herramienta para incluir la seguridad en todo el ciclo de desarrollo de productos de software, a pesar de ser una metodología muy nueva y que aún debe ser explorada se basa en principios más maduros como son Agile y DevOps, invitando no sólo a un equipo a tomar conciencia sino a toda una organización. Al implementar DevSecOps es importante hacer énfasis en los principios y valores más que en el uso de herramientas, tanto las personas involucradas en el desarrollo como las personas encargadas del producto deben entender los riesgos y vulnerabilidades a las que se ven expuestos sino se toman medidas para evitarlos. Los resultados de las herramientas de detección de vulnerabilidades deben ser tomados con cuidado, esto debido a que aún se pueden presentar casos de falsos positivos si las herramientas no se ajustan adecuadamente. Por esto mismo es importante definir niveles de tolerancia para que las ejecuciones de los Pipeline no se vean interrumpidos de manera innecesaria.

Referencias

[1] AMAA Ahmed. «DevSecOps: Enabling Security by Design in Rapid Software Development». Tesis de mtría. 2019.

[2] S. Balaji y M. Murugaiyan. «WATEERFALLVs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC». En: 2012.

[3] Jimmy Björnholm. *Performance of DevOps compared to DevSecOps: DevSecOps pipelines benchmarked!* 2020.

[4] Robert C. Martin. *Agile Software Development, Principles, Patterns, and Practices*. 1.ª ed. Pearson Education Limited, 2014, págs. 3-5.

[5] José Joaquin Caño Quintero. «DevSecOps: integración de herramientas SAST, DAST y de análisis de Dockers en un sistema de integración continua». Tesis de mtría.

[6] Jessica Diaz y col. «DevOps in practice: an exploratory case study». En: *Proceedings of the 19th International Conference on Agile Software Development: Companion*. 2018, págs. 1-3.

[7] Inc Docker. *What Is A Container?* 2020. URL: `https://www.docker.com/resources/what-container`.

[8] FMA Erich, Chintan Amrit y Maya Daneva. «A qualitative study of DevOps usage in practice». En: *Journal of Software: Evolution and Process* 29.6 (2017), e1885.

[9] J Laurenz Eveleens y Chris Verhoef. «The rise and fall of the chaos report figures». En: *IEEE software* 27.1 (2010), pág. 30.

[10] Martin Fowler. *Continuous Integration*. 2006. URL: `https://martinfowler.com/articles/continuousIntegration.html`.

[11] Inc Gartner. *Dynamic Application Security Testing*. 2020. URL: `https://www.gartner.com/en/information-technology/glossary/dynamic-application-security-testing-dast`.

[12] Inc Gartner. *Information technology - gartner glossary - devsecops*. 2020. URL: `https:%20//www.gartner.com/en/information-technology/glossary/devsecops`.

[13] Jonas Heilmann. *Application Security Review Criteria for DevSecOps Processes*. 2020.

[14] Aymeric Hemon y col. «From agile to DevOps: Smart skills and collaborations». En: *Information Systems Frontiers* 22.4 (2020), págs. 927-945.

[15] Jez Humble y David Farley. «Continuous Delivery: Reliable Software Releases through Build». En: *Test, and Deployment Automation. Addison-Wesley* (2010).

[16] InfoQ. 2020. URL: `https://www.infoq.com/articles/standish-chaos-2015/`.

[17] Sergio Iriz Ricote. «Seguridad en el ciclo de vida del desarrollo del software: DevSecOps». Tesis de mtría.

[18] Ramtin Jabbari y col. «What is DevOps? A systematic mapping study on definitions and practices». En: *Proceedings of the Scientific Workshop Proceedings of XP2016*. 2016, págs. 1-11.

[19] Mohd Ehmer Khan, SGM Shadab y Farmeena Khan. «Empirical Study of Software Development Life Cycle and its Various Models». En: ().

[20] Anna Koskinen. «DevSecOps: building security into the core of DevOps». En: (2019).

[21] Victor Mayoral-Vilches y col. «DevSecOps in Robotics». En: *arXiv preprint arXiv:2003.10402* (2020).

[22] Jose Morales y col. «Guide to Implementing DevSecOps for a System of Systems in Highly Regulated Environments». En: ().

[23] Håvard Myrbakken y Ricardo Colomo-Palacios. «DevSecOps: a multivocal literature review». En: *International Conference on Software Process Improvement and Capability Determination*. Springer. 2017, págs. 17-29.

[24] Owasp.org. *Source Code Analysis Tools*. 2020. URL: `https://owasp.org/www-community/Source_Code_Analysis_Tools`.

[25] BS Rahul, Prajwal Kharvi y MN Manu. «Implementation of DevSecOps using Open-Source tools». En: (2019).