Protección de aplicaciones web en DevOps – Parte 1

Angie Daniela Ruiz Alfonso

Angi Paola Jiménez Pira

Juan Sebastián Díaz Salamanca

Escuela Colombiana de Ingeniería Julio Garavito

Seguridad y privacidad de TI

Diego Alexander López Correa

2020

Tabla de contenido

Resumen

El nuevo enfoque de colaboración que brinda DevOps permite a los equipos trabajar de forma más cercana, aportando mayor agilidad al negocio y notables incrementos en la productividad, brindándoles empoderamiento para prevenir, descubrir y solucionar problemas de seguridad. Este cambio cultural permite a las empresas acelerar el ciclo de vida de desarrollo sus aplicaciones web sin dejar de lado la seguridad desde su comienzo, estableciendo así prácticas mucho más sólidas.

Dentro de DevOps, existe una filosofía relacionada al paradigma de seguridad, llamada DevSecOps, basada en la integración de la seguridad en los procesos DevOps y en una cultura bajo el concepto de Seguridad cómo Código. La cual tiene dos objetivos principales, que son la velocidad de entrega en prácticas ágiles, y las pruebas de seguridad entre iteraciones sin frenar los ciclos de entrega. Esta pretende reemplazar métodos tradicionales de seguridad, con comunicación y responsabilidad compartida durante todo el proceso de entrega del código.

Introducción

DevOps es un conjunto de prácticas, herramientas y filosofías culturales que unen el desarrollo de software (Dev) y las operaciones de TI (Ops) y aumentan la capacidad de una organización para ofrecer aplicaciones en ciclos de desarrollo más cortos y ajustados a los objetivos empresariales.

La implementación de DevOps en el mercado ha crecido de manera exponencial, según un estudio de CA Technologies el 88% de 1425 ejecutivos declararon que han adoptado DevOps o planean hacerlo en los próximos 5 años y según los informes de estado de DevOps presentados en los últimos años por Puppet, las organizaciones que han adoptado DevOps han obtenido mejor rendimiento que las que no, sin embargo, los aspectos de seguridad en DevOps siguen siendo una preocupación a la hora de desarrollar aplicaciones. En este informe se explicará el papel que juegan los DevOps en la seguridad de aplicaciones, cuál es la interacción deben tener los desarrolladores con los expertos en seguridad a la hora de desarrollar este tipo de productos, así mismo, se definirá una serie

de estrategias que estos equipos deben implementar con el fin de obtener una aplicación que cumpla con ciertos estándares de seguridad, de modo que se garantice un producto funcional y seguro a los clientes que lo requieren.

DevOps y su impacto negativo en la seguridad de aplicaciones web

El desarrollo de software ágil es algo que hemos tenido presente por casi dos décadas, desde que se publicó el manifiesto original, a partir de esto los equipos de desarrollo de software se han esforzado por obtener productos que respondan a las necesidades de los clientes en el menor tiempo posible, para lograr esto se implementan DevOps, una metodología que involucra diferentes equipos que colaboran entre sí para el desarrollo de un software de manera rápida y eficiente. Sin embargo, para los equipos de seguridad de TI, el auge de DevOps también ha provocado problemas en la gestión de la seguridad y el riesgo, avanzar demasiado rápido sin un proceso de seguridad adecuado deja a las organizaciones expuestas a aplicaciones con funcionalidades deficientes, fallos en el software o de seguridad que son factores clave que impactan fuertemente en la calidad del producto.

El crecimiento exponencial de la economía inducido por la cantidad de aplicaciones web desarrolladas en los últimos años han provocado una expansión acelerada de DevOps en el mercado, lo que ha ejercido una gran presión sobre los equipos de desarrollo para alcanzar la completitud del producto en tiempos más cortos, en este ejercicio se sacrifica seguridad por velocidad pues el tiempo estándar de desarrollo de aplicaciones que antes era meses puede reducirse a semanas, poniendo en riesgo la aplicación desde el momento cero de su lanzamiento y suponiendo que el funcionamiento actual del producto se mantendrá a futuro sin tener en cuenta el constante surgimiento de nuevas amenazas.

Cuando varios equipos colaboran en el desarrollo de un mismo software la seguridad de convierte en un problema de todos, si estos equipos están desalineados las consecuencias pueden incluir código inseguro, vulnerabilidades, configuraciones incorrectas, contraseñas codificadas de forma

no segura y debilidad de seguridad de las aplicaciones que causan disfunción operativa o son objetivos fáciles para los atacantes, es por eso que el enfoque estándar de seguridad no funciona por la separación entre los desarrolladores y los que se encargan de seguridad, estos siempre deben trabajar de la mano, de esta manera se puede garantizar que los cambios implementados por cada integrante del proceso sean sometidos tanto a las pruebas habituales de la aplicación como pruebas dirigidas a seguridad; si bien hay una serie de herramientas creadas por proveedores de seguridad que respaldan las profesionales, los desarrolladores necesitan un conjunto diferente de herramientas cuando se trata de DevOps y seguridad en sus aplicaciones ya que este factor debe mantenerse en el tiempo, además, en la era de DevOps y DevSecOps la seguridad tiene que encontrar la manera no solo de agregar defensa, sino de proporcionar un nuevo valor al producto. Teniendo en cuenta esto se han definido algunas prácticas recomendables a la hora de desarrollar software con DevOps.

DevOps y seguridad en aplicaciones: Mejores prácticas

Cooperación e integración

Durante el desarrollo con DevOps deben incorporarse códigos éticos, métodos y protocolos de seguridad que deben ser cumplidos por todos los involucrados en la creación del producto de manera que se garantice la integridad de la aplicación y sean coherentes con la empresa cliente.

Control de credenciales

Se debe centralizar el acceso mediante un ambiente de cooperación y transparencia entre el equipo de seguridad y desarrolladores para determinar alcances y privilegios de los diferentes usuarios de la aplicación.

Automatización

La automatización de procesos dada en las empresas es una oportunidad para modernizar y robustecer la seguridad de la organización, un momento para identificar y estandarizar los procesos en función de responder automáticamente a las amenazas, para lograr satisfacer la demanda de retos de seguridad que ofrece DevOps en el desarrollo de aplicaciones web debemos tener en cuenta el factor de gestión de riesgos para aplicaciones.

Gestión de riesgo para aplicaciones desarrolladas con DevOps

La revolución de las DevOps ofrece la oportunidad de modernizar y robustecer seguridad en aplicaciones web, ya que el ambiente de desarrollo y producción son creados en el mismo momento y bajo las mismas reglas y estándares de seguridad.

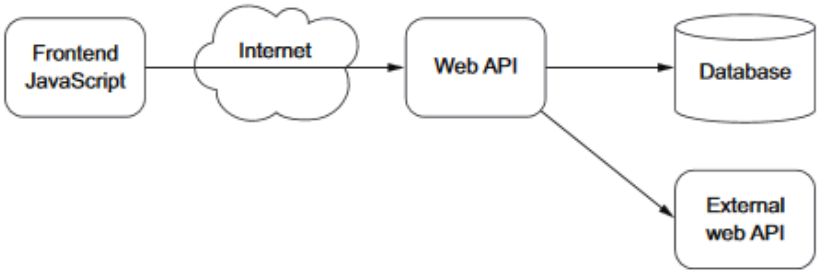
Para robustecer la seguridad es importante llevar a cabo una gestión de riegos donde es vital tener una visibilidad integral de la aplicación durante todo el tiempo de desarrollo, es decir, no separar los factores asociados a seguridad de otros aspectos relacionados con la aplicación, además de esto, se deben estudiar activos de información, sistemas, procesos y personal involucrado en el desarrollo.

Asegurar el código en un entorno de desarrollo DevOps también requiere administrar la cadena de suministro de software y verificar la seguridad de los componentes y frameworks comunes, adoptar un framework para automatizar las pruebas, usar herramientas rápidas de análisis estático y escanear automáticamente en busca de vulnerabilidades sería lo ideal. Si bien las empresas pueden adoptar servicios de pago, muchas de las ofertas de código abierto pueden dar a los desarrolladores un buen comienzo verificando bibliotecas de código abierto y los componentes comunes en busca de vulnerabilidades, centrarse en el componente del programa de código abierto es importante porque constituye la mayor parte de código en cualquier aplicación, dice Sonatype's Weeks. *"Cuando miras los componentes de código abierto, forman entre el 80% y el 90% de la huella de la aplicación en la actualidad, y el resto es código personalizado"*, dice. "*Por lo tanto, cuando observa los componentes de código abierto, son la parte más importante de la aplicación"*.

Herramientas de código abierto como Metasploit Framework, OpenVAS y Arachni permiten automatizar las pruebas de ataque. Metasploit Framework puede intentar la explotación específica de problemas, mientras que OpenVAS facilita el escaneo de una red en busca de vulnerabilidades. Finalmente, Arachni es un escáner de seguridad de aplicaciones web.

Securing and Testing Web Apps

Los servicios web modernos se componen de varias capas que interactúan entre sí mediante *Hypertext Transfer Protoco*l (HTTP) a través de la red, a continuación, se muestra un diseño a alto nivel que representa la forma de interacción entre diferentes capas de backend y frontend, que junto a la base de datos conforman un servicio web:



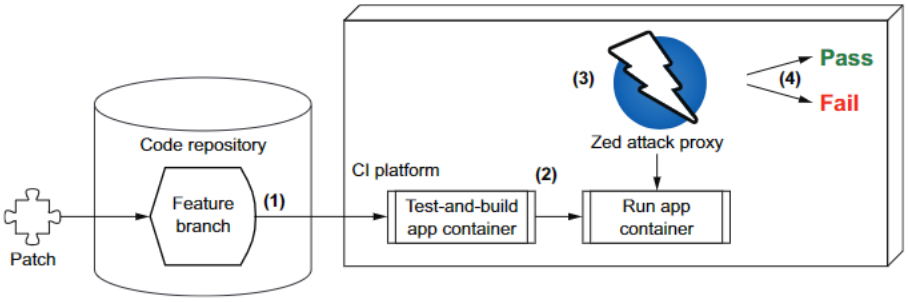
Se puede observar que la capa de front-end consume unos servicios ofrecidos el back-end, que a su vez interactúa con la base de datos y posiblemente otras API web para el cumplimiento efectivo de sus objetivos.

La simplicidad de una página no implica que tenga más o menos vulnerabilidades, es importante saber identificar las vulnerabilidades de una página web sea de forma manual o utilizando una herramienta que realice un escaneo automático, hay que tener presente que detectar todas las vulnerabilidades de forma manual puede ser una tarea tediosa, los equipos de seguridad por lo general realizan un escaneo de vulnerabilidades manual, cuando se realiza una auditoría de una aplicación, de forma semanal o mensual, pero este enfoque hace que sea lento este proceso y como los análisis solo se ejecutan periódicamente, es posible que los servicios vulnerables se

implementen en producción durante un tiempo antes de que se detecten dichos problemas, este flujo de trabajo puede mejorar utilizando métodos DevOps, donde en lugar de ejecutar escaneos definidos con cierta periodicidad, se ejecuten escaneos cada vez que el código es verificado en una rama de funciones del repositorio, acercando así las pruebas de seguridad a las pruebas unitarias y de integración, con el fin de que los desarrolladores detecten problemas de seguridad mientras el código está en proceso y no cuando se ejecuta en producción.

El escaneo en busca de vulnerabilidades web puede llevar horas y no es adecuado para un flujo de trabajo en el que los desarrolladores necesitan iteraciones rápidas sobre los cambios en el código. Es necesario un escaneo rápido, conocido como un análisis de línea de base, ya que se centra en controles elementales en lugar de una evaluación de vulnerabilidad exhaustiva, donde al final el resultado del escaneo determina si se aprueba o rechaza el cambio.

Para ver los posibles problemas en una página, sea una página simple o no, es recomendable detectar si la página está protegida contra secuencias de comandos entre sitios, falsificación de solicitudes entre sitios, secuestro de clics y fugas de datos.



Website attacks and content security

Cross-site scripting and Content Security Policy

El ataque de scripting entre sitios, comúnmente conocido como XSS, es la vulnerabilidad web más prevalente en ese momento (haciéndose cada vez más famoso al ser el problema de seguridad más reportado en los sitios web modernos). Estos ataques son causados por la inyección de código fraudulento en un sitio web, este código se ejecuta desde el navegador, es decir, desde la capa de front-end que al tener comunicación con el back-end y a su vez con la base de datos puede obtener información sensible que se encuentre alojada allí o puede perjudicar la integridad de los datos ejecutando sentencias que los altere de forma indebida, es por esto, que suele ser el primero modo de operación de los atacantes, pero XSS es solo un problema en las páginas HTML donde se pueden abusar de los scripts y estilos para ejecutar código malicioso, y es posible en una API web donde se pueden usar para devolver HTML o datos de alimentación en otras páginas HTML.

El ataque conocido como DOM XSS, modifica el Modelo de objetos de documento (DOM) del navegador, se logra inyectando código en una de las cadenas de consulta de parámetros, la recomendación general para las aplicaciones web es que validen la entrada del usuario (revisando cada campo recibido y comparándolos con la expresión regular que se espera) y escapar de todos los datos devueltos a los usuarios antes de mostrarlos en la página(la mayoría de los lenguajes tienen bibliotecas para escapar del contenido, las cuales el navegador no interpretará la cadena de escape como HTML válido y, por lo tanto, no dará lugar a la ejecución del código). En las páginas web que admiten entradas de formatos complejos como SVG o XML se debe integrar Content Security Policy (CSP), este mecanismo habilita un canal mediante el cual una aplicación web puede decirle a los navegadores web lo que debe y no debe ejecutarse al representar el sitio web, se puede usar CSP para bloquear ataques XSS declarando una política que prohíbe la ejecución de scripts en línea y establece que solo se debe confiar en el contenido que proviene del mismo origen. La declaración del CSP se realiza a través de un encabezado HTTP devuelto por la aplicación con cada respuesta HTTP, se puede enviar el encabezado del CSP desde cualquier componente de la infraestructura que se encuentre en la ruta de la aplicación web o directamente desde el servidor web, aunque devolver encabezados de seguridad desde el servidor web es una buena manera de garantizar que los encabezados estén siempre configurados, se recomienda administrar CSP directamente en el código de la aplicación para facilitar la implementación y las pruebas para los

desarrolladores, además, si bien es cierto que CSP proporciona una forma de definir qué es y qué no es una interacción aceptable, lo cual es excelente para la seguridad, también requiere cierto esfuerzo, esfuerzo que se logra simplificar si los desarrolladores administran el mecanismo directamente en el código en lugar de que el equipo de seguridad lo configure en el componente front-end. Una manera de probar que CSP funciona correctamente es abrir la consola del navegador e intentar correr un comando o línea de código, si está bien configurado, debería lanzar un error, pues desde allí no se puede ejecutar nada.

Cross-site request forgery

Un componente importante o clave de las páginas web, es el hecho de que un sitio puede enlazar recursos que están alojados en otro sitio web. Es un buen modelo de colaboración mutua que funciona de una manera en la cual ambos sitios se benefician de información, pero nunca intentan cambiar el contenido de la otra. Este tipo de comportamiento es útil pero no protege en contra de abusos. Un ataque Cross-site request forgery (CSRF) toma ventaja de esta vulnerabilidad y abusa de los enlaces entre sitios para forzar un usuario a realizar acciones las cuales no se supone que debe hacer.

Un caso es que, un usuario puede entrar a una página maliciosa sin tener conocimiento de que ésta tiene algún tipo de script o código peligroso, y lo que hace sin saberlo es ejecutar código dirigido a otra página, por ejemplo, por medio de una petición GET y manda una petición maliciosa a la otra dirección web en donde ésta última la identifica cómo legítima y la procesa. Se pueden dar casos de eliminar datos o información importante con el objetivo de hacer un daño en específico.

Es posible protegerse de un ataque CSRF usando un token de rastreo cuando la página principal de la web se construye y luego el navegador lo devuelve cuando se envía una solicitud maliciosa, cómo la de eliminación. De esta manera, el sitio malicioso no tiene forma de acceder a los datos que el navegador y la página víctima intercambian, así que no puede forzar al navegador a ejecutar la sentencia o código fraudulento. Esto provoca que, si un token no está presente en la petición,

entonces se rechaza. El token puede estar escondido en el cuerpo HTML de la página a la cual se debe proteger como un campo oculto y cada vez que un usuario entre, lo genera de manera aleatoria por medio de un HMAC (hash-based message authentication code). De ahí en adelante, si las peticiones de este usuario no contienen este HMAC, se marcará la petición como inválida y no se ejecutará.

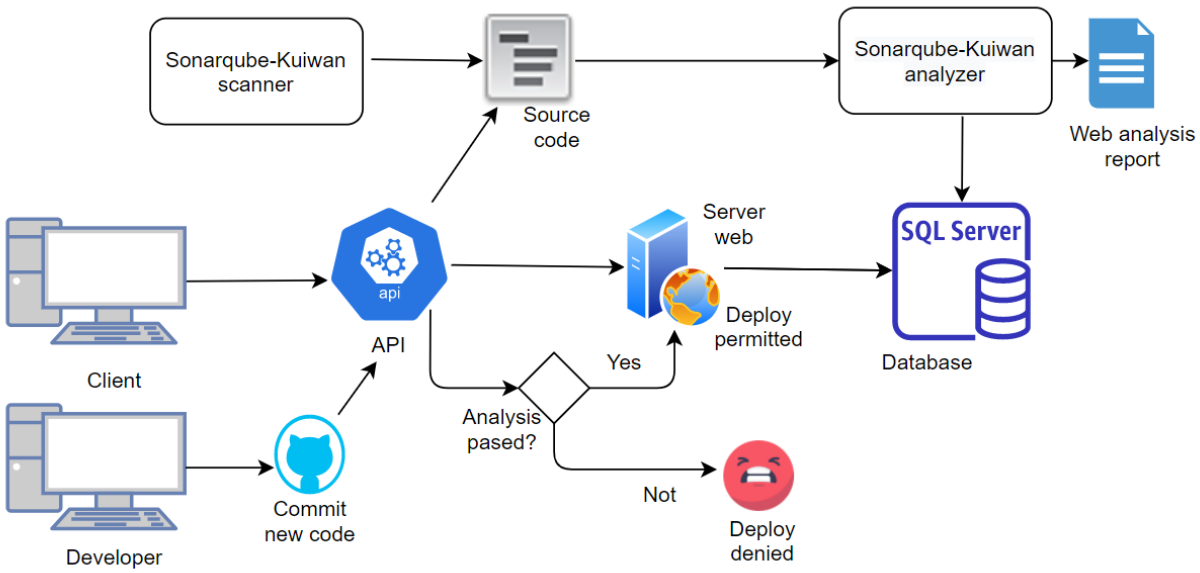
Clickjacking and IFrames protection

La técnica del IFrame es compatible con los navegadores web y puede habilitar un vector de ataque peligroso llamado clickjacking, porque antes los sitios a menudo incrustaban contenido entre sí mediante marcos en línea y la etiqueta HTML <iframe>. Este secuestro de clics es una técnica que permite que un sitio fraudulento engañe a un usuario para que haga clic en un enlace invisible que apunta a un sitio diferente, pero desde el punto de vista del navegador, este clic fraudulento es legítimo lo cual es peligroso para el usuario engañado. Las protecciones contra ello no están habilitadas de forma predeterminada y los desarrolladores deben agregarlas de forma manual.

El enfoque moderno para protegerse contra el secuestro de clics es usar CSP para establecer una política para child-src "self", lo que indica que el sitio solo debe estar enmarcado por una página que comparte el mismo origen y no otro. Otro método para protegerse contra el secuestro de clics es configurar el encabezado HTTP X-FRAME-OPTIONS, devolviendo este encabezado con un valor de SAMEORIGIN, logrando el mismo efecto que usar la directiva CSP, debido a que no todos los navegadores son compatibles con la directiva child-srcd de CSP. Lo más recomendable será entonces tener un CSP configurado en la página de inicio del invocador y agregar el encabezado X-FRAME-OPTIONS también.

Laboratorio

Arquitectura POC (Proof of Concept):



Detalle técnico de la POC:

Cuando un desarrollador realiza una actualización sobre código alojado en el repositorio de la API, esta actualización es permitida únicamente si pasa el análisis web del código estático, teniendo un reporte con un apetito mínimo de riesgo. En cambio, cuando un cliente consume la API, esta está disponible para su uso debido a que previamente ya se realizó el análisis web sobre el código estático que compone esta API, haciendo así posible su uso de forma segura. Realizar este análisis en la parte del desarrollo es de vital importancia debido a que se logra detectar la vulnerabilidad de manera rápida mitigando así el riesgo de ataque antes de desplegar el código que va a consumir la API, para que posteriormente esta pueda ser usada por los respectivos clientes.

Al implementar esta arquitectura en un ciclo de desarrollo de DevOps orientado a aplicaciones web contribuye a la implementación de una aplicación web más segura, sin que se ralentice el tiempo de su desarrollo, garantizando el cumplimiento de la metodología ágil (SCRUM) a la vez que se mantiene la integridad sobre el lanzamiento del producto final.

Conclusiones

Muchas de las herramientas utilizadas para brindar seguridad dentro del desarrollo web al estilo DevOps se originaron en un proyecto de código abierto, por lo que los desarrolladores tienen buenas opciones para ayudarse a probar y mejorar la seguridad de su código, pero estas no son suficientes y pueden llegar a requerir mucho tiempo para su correcta configuración, por ende, la gran importancia del cambio de mentalidad respecto a no sólo priorizar la seguridad en cada etapa, sino también considerarla como algo que no ralentiza el trabajo, ni la productividad, rompiendo así con esta estigmatización que se ha tenido a través del tiempo, logrando así cambiar hábitos, hacer la integración de la seguridad mucho más sencilla y aumentar la conciencia general sobre la seguridad.

Una fuerte cultura de DevOps es el soporte de una seguridad más fuerte. La cultura de compartir, en donde los equipos colaboran y usan herramientas para trabajar por metas comunes, donde cuentan con una autonomía, es una cultura en la cual la seguridad es verdaderamente una responsabilidad compartida, así se pueden identificar los problemas más rápido y enfrentarlos con mejores soluciones

Referencias

- Puppet, circleci, & slunpk. (2019). State of DevOps Report. https://media.webteam.puppet.com/uploads/2019/11/2019-state-of-devops-report-puppet-circleci-splunk_sml-1-1.pdf?_ga=2.29774134.881042528.1599761461-25174917.1599761461

- A. (2019, 25 febrero). Application Security in Devops. Kiuwan. https://www.kiuwan.com/blog/application-security-in-devops/

- G. (2020, 24 junio). *DevOps y Ciberseguridad: Nuevos retos y soluciones en seguridad de TI*. GB Advisors. https://www.gb-advisors.com/es/devops-y-ciberseguridad-nuevos-retos-y-soluciones-en-seguridad-de-ti/

- Lemos, R. (2019, 22 enero). *The best open-source DevOps security tools, and how to use them*. TechBeacon. https://techbeacon.com/app-dev-testing/best-open-source-devops-security-tools-how-use-them

- *¿Qué es la seguridad de DevOps? Desafíos de la seguridad de credenciales*. (2020, 17 julio). CyberArk. https://www.cyberark.com/es/what-is/devops-security/

- R. (2017, 3 diciembre). *¿ Que son los DevSecOps ?* CIBERSEGURIDAD. blog. https://ciberseguridad.blog/que-son-los-devsecops/

- *What is DevOps Security?* (2020). BeyondTrust. https://www.beyondtrust.com/resources/glossary/devops-security

- *DevOps: qué es y cómo lo aplicamos*. (2020, 21 julio). Claranet. https://www.claranet.es/devops-que-es-y-como-lo-aplicamos

- Libro: Securing DevOps Security in the Cloud. Autor: Julien Vehent. Editorial: MANNING Shelter Island.