**Conocimientos y experiencia**

**SDLC & Desarrollo de Seguridad**

1. De acuerdo a tu experiencia profesional enumera y explica las etapas del SDLC donde incorporaste controles de seguridad.

En mi experiencia, mi enfoque fue integrar la seguridad en cada fase del ciclo de vida de desarrollo de software para minimizar riesgos desde el principio.

* **Fase de Requisitos**: Desde el inicio, trabajamos en el modelado de amenazas con los equipos de producto y desarrollo para poder identificar los posibles puntos de ataque y definir los requisitos de seguridad.
* **Fase de Diseño**: En esta etapa, realicé revisiones de la arquitectura y el diseño para asegurar que se siguieran los principios de seguridad, como el de mínimo privilegio y defensa en profundidad. Un caso práctico fue la revisión de la arquitectura de microservicios de la aplicación en donde se resolvieron temas de encriptación de la comunicación para los tokens de sesión que se generan, ya que antes no eran tan complicados de descifrar..
* **Fase de Desarrollo**: Promovimos las prácticas de codificación segura a través de sesiones de capacitación para los desarrolladores, para esto me apoyé tanto de fuentes como OWASP top 10 y de nuestro partner Palo Alto para incluir esto en nuestro proyecto de App Security..
* **Fase de Pruebas**: Aquí es donde implementamos las pruebas SAST y SCA en nuestro pipeline CI/CD, como explicaré más adelante. También realizamos pruebas de penetración manuales en entornos de pre-producción. Todo esto se realizó con Prisma Cloud.
* **Fase de Despliegue y Mantenimiento**: Monitoreamos constantemente los logs de seguridad y utilizamos herramientas para la gestión de vulnerabilidades en producción, de tal manera que teníamos visibilidad por parte del dev como para el equipo de seguridad, en donde se integró Jira para poder gestionar los findings en los pipelines y su solución respectiva.

1. ¿En qué parte del pipeline CI/CD aplicaste las pruebas SAST, DAST, SCA?

Bueno, para las pruebas de SAST se implementaron en un stage dedicado a la seguridad dentro de nuestro pipeline de GitLab CI/CD. Usé herramientas como checkov y twiscli para escanear el código fuente y las configuraciones de iac. En este caso checkov me ayudó a validar las configuraciones de IaC, mientras que twisclil para analizar el código en busca de secretos expuestos y licencias.

Para el análisis de composición de software, también usé twiscli que me permitió identificar vulnerabilidades en las librerías de terceros que utilizábamos en nuestros proyectos. Además se realizaba un escaneo pasivo de todos los repos, con prisma para encontrar cualquier finding con respecto a sbom o secrets.

Las pruebas DAST se realizaban en un job posterior, una vez que la aplicación ya estaba desplegada en un entorno de pruebas, para esto integramos el prisma code analysis que nos ayudó a validar las vulnerabilidades y analizar secretos, licencias, y configuraciones de IaC, asímismo configuramos thresholds específicos para la criticidad de las vulnerabilidades y las reglas de cumplimiento para asegurarnos de que solo el código seguro avanzara en el pipeline.

1. Describe cómo implementaste o implementarías un modelo de “shift-left security” en una organización que apenas comienza su madurez en DevSecOps.

En mi experiencia, implementar el "shift-left" no se trata de complicar el trabajo de los desarrolladores, sino de hacerlo más fácil y seguro desde el principio, primero me enfocaría en la gente. Daría capacitaciones prácticas sobre el OWASP Top 10 para que los equipos entiendan por qué la seguridad es importante. Luego, integraría herramientas de análisis estático directamente en el IDE para que puedan corregir vulnerabilidades mientras escriben el código, sin tener que esperar, posterior a esto plantearía un proyecto de automatización del análisis del ci/cd tanto con herramientas open source como pagadas, de forma que con esto se logre analizar todo lo que se deployee y se configure en modo alert hasta comprobar cuál es la postura de seguridad de aplicaciones de la empresa y establecer los umbrales respectivos iniciales de bloqueo y de análisis, garantizando que con el tiempo se reduzcan las vulnerabilidades presentes en el código que ya se deployeo y se corrijan todos los nuevos desarrollos y al final con las métricas realizaría un análisis para ir mejorando los parámetros de configuración de los bloqueos para evitar falsos positivos e incrementar el control de seguridad en el CI/CD.

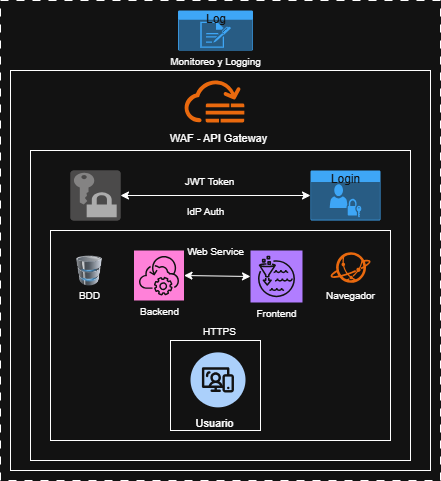
1. Explica cómo has trabajado el análisis de vulnerabilidades comunes (OWASP Top 10) en un flujo CI/CD (procesos automáticos, tareas manuales)?

Para el análisis de vulnerabilidades me manejé con OWASP top 10 y con las de CVE, que en realidad van de la mano ya que unas se enfocan el proceso de desarrollo y las otras en vulnerabilidades específicas de paquetes o librerías desactualizadas o no confiables. para abordar estas vulnerabilidades usé el escaneo de prisma inicialmente en todos los repositorios para hacer una evaluación inicial de seguridad a nivel de sistema de control de versiones que en este caso era gitlab, de esta forma pude revisar que existían miles de vulnerabilidades en los repos, entonces se procedió a levantar reportes de todos los repos y archivos afectados corrigiendo los que eran prioritarios y asumiendo algunos que no tenían impacto significativo, posterior a esto ya abordamos el ci/cd en donde básicamente se creó un stage en el gitlab-ci.yml que lo que hacía es analizar cualquier vulnerabilidad detectada así como secrets o iac missconfigurations, esto de manera automatizada en los branches más criticos en este caso dev y stg.  
A nivel de procesos manuales dentro de este flujo se empezaron a hacer pentesting a los códigos, especialmente los que se realizaban en python, otro proceso manual era la asignación de los tickets a los devs revisando porque falló el pieline y si se debió a una vulnerabilidad muy critica o no para ir documentando y hardenizando las librerias que usan para poder tener un mejor control de la seguridad a ese nivel.

**Seguridad en Arquitectura**

1. Diseña una arquitectura segura de una aplicación web con autenticación federada, sesiones seguras y protección contra ataques comunes.

**Arquitectura segura webapp**

****

**Autenticación:**

Para diseñar esta arquitectura, me basé en un esquema básico de una webapp simple, en donde en lugar de manejar las contraseñas de los usuarios directamente en nuestra aplicación, utilizamos proveedor de identidad externo, lo que evita la administración de credenciales y le da la responsabilidad al idp que lo maneja de manera segura. El usuario es redirigido a este servicio para iniciar sesión y, al ser autenticado, el IdP le devuelve un token de acceso seguro.

**Transacciones:**

Todas las solicitudes pasan por un gateway, que actúa como nuestro punto de entrada único al backend. Este componente es crítico para la seguridad, ya que valida el token en cada solicitud, lo que nos permite verificar la identidad del usuario sin tener que manejar sus credenciales, por lo cual el gateway puede manejar funciones como el rate limiting para mitigar ataques de denegación de servicio.

**Control de Acceso:**

Una vez que la solicitud llega al backend, se aplican dos controles de seguridad clave. Primero, cada microservicio verifica los permisos del usuario para asegurarse de que tiene autorización para acceder a la información o realizar la acción solicitada.

**Protección contra ataques:**

Para protegernos de ataques en el perímetro, se pone un WAF para poder analizar y filtrar el tráfico entrante, bloqueando intentos de ataque conocidos antes de que lleguen a nuestra aplicación. De ahí para la gestión de las sesiones, utilizaría un enfoque seguro basado en tokens. En lugar de una sesión tradicional, se usa tokens de acceso de corta duración y tokens de refresco de vida más larga, lo que protege las sesiones contra secuestro.

#### **Monitoreo**

Para tener un buen monitoreo de nuestra app se plantea también un sistema de logging centralizado en todas las capas de la arquitectura, desde el gateway hasta la base de datos, lo que nos permite monitorear y responder a incidentes.

1. ¿Qué consideraciones de seguridad tomarías en una arquitectura serverless? Estándares y Regulaciones

Para mi en serverless debe haber la misma seguridad que en la infraestructura normal, aunque es algo que no se respeta actualmente por lo que piensan que el servidor ya está siendo manejado por nuestro proveedor, por eso tomaría las consideraciones completas que en este caso serían; primero el mínimo privilegio en donde cada función de estas le damos solo los permisos que necesita para hacer su trabajo y nada más.

Otra consideración y esta la digo como experiencia es nada de secretos en el código**.** Las claves, contraseñas y todo lo delicado lo guardamos aparte en un servicio seguro. A la función se lo entregamos justito en el momento que lo necesita.

Y la última pero no menos importante es que hay que validar la información que entra, desde que llega por la puerta principal que en este caso es el waf. Así evitamos cualquier ataque simple que se pueda colar en nuestra infra..

1. Explica cómo has utilizado el CWE en tu experiencia profesional para priorizar la corrección de vulnerabilidades.

Este es uno de mis punto fuertes ya que fue uno de mis primeros acercamientos a la gestión de vulnerabilidades, para mi es uno de los pasos más importantes en devsecops, yo lo usé para poder priorizar alrededor de 9k de vulnerabilidades en donde el primer paso obviamente es clasificarlas por criticidad, en este caso yo siempre me baso primero en el cvss y en el factor de explotación, después de esto procedo a ver el impacto que puede tener la vulnerabilidad si es explotada, es decir a que tipos de recursos está dirigida, para esto se genera un reporte con la lista de vuln en donde pueda filtrar por cvss, severidad, ambiente si es posible y por recurso y cve, de tal manera que puedo irlas filtrando y priorizando facilmente, posterior a esto se crean incidencias de forma automática o manual en donde se asigna los Sla y los responsables de dicha remediación, creo que es una forma muy sencilla de ir remediando poco a poco para no tener un mar de vulnerabilidades y agobiarse en no poder hacer nada que es algo que me pasaba a inicio pero para eso existe la priorización

1. Describe cómo has mantenido un SBOM actualizado en proyectos anteriores y por qué consideras que es importante basado en tu experiencia.

Siempre lo he realizado de manera automatizada, en este caso mi experiencia manteniendo actualizado el sbom ha sido con prisma cloud que tenía un apartado en donde se podía ver todo lo que se usaba en un repo especifico, desde versiones hasta dependencias, entonces usábamos esto para integrar el pipeline para que cada vez que se genera un build o se actualizan las dependencias, configuramos una herramienta para que escanee y genere un sbom nuevo automáticamente. Un caso particular que tuve fue que alguna vez cuando salió la vulnerabilidad de Log4j, pudimos saber en cuestión de minutos si alguna de nuestras aplicaciones estaba usando esa librería solo revisando el reporte sbom de prisma.