**Caso de Estudio – Estrategia y Cultura de Seguridad**

Con respecto a este caso planteado la situación es media crítica pero manejable. El problema principal es que no tienen una base sólida de seguridad, lo que ha provocado incidentes, sobre todo en la autenticación que es la puerta de entrada a todo.

* El plan de acción que propongo no es solo técnico, sino que busca un cambio cultural. La estrategia es un enfoque de "shift-left" para que la seguridad sea responsabilidad de todos, no solo de un equipo. Nos vamos a enfocar en ir de lo más urgente a lo más estratégico, empezando por arreglar la autenticación y, de a poco, construyendo un programa de AppSec robusto. Esto lo vamos a lograr en un plazo de 3 a 6 meses.

### **Quick wins:**

En este contexto vamos a relacionar las quick wins con las acciones de alto impacto y baja complejidad que se pueden implementar en los primeros meses para generar confianza y sentar las bases de un programa de seguridad más chévere.

**Técnicos:**

* Como la prioridad es remediar las brechas que se tiene en autenticación, se debe empezar por ahí entonces empezaría con multifactor en la autenticación así mismo, para esto lo haría con servicio Oauth seguro y arquitectura de seguridad en autenticación, es decir sanitización de entradas, que todos los campos de login sean seguros y que no se tengan mensajes genéricos al momento de poner credenciales incorrectas.
* Se debe integrar una herramienta para escanear el historial de Git y los archivos en busca de credenciales expuestas de tal forma que vamos minimizando la superficie de ataque en la auth.
* Otro paso opcional pero que también me ha ayudado a darle prioridad a la remediación aplicado a este contexto, ya que los problemas los tenemos en la autenticación haría inteligencia con hudson rocks o alguna otra herramienta similar, para poder que tantos datos esten expuestos con respecto a usuarios y contraseñas, de esta forma dimensiono y actúo de forma gradual o inmediata.

**Cultura:**

* Dentro del flujo de gestión de vulnerabilidades incluir un canal de comunicación y un sistema de incidencias con alguna herramienta tipo jira o slack para tener la trazabilidad correcta y de paso ir haciendo los playbook necesarios para cada vulnerabilidad.

### **Métricas:**

Para medir la mejora, se debe ir más allá de los reportes. Estas son las métricas que yo monitorearía en el plazo de 3-6 meses

* Medir el porcentaje de código nuevo que está siendo escaneado por SAST y el porcentaje de dependencias cubiertas por SCA. La meta es alcanzar el 100%.
* Se mide la cantidad de vulnerabilidades críticas o de alta severidad que se introducen con cada nuevo pull request. El objetivo es que este número sea lo más cercano a cero posible.

### **Plan de acción: Cronograma de 3-6 meses**

| **Mes** | **Actividad** | **Objetivo** |
| --- | --- | --- |
| **Mes 1-2** | **Fase 1: Remediación en curso** |  |
|  | 1. Arreglar la Autenticación | Implementar medidas de seguridad urgentes para proteger las cuentas de usuario en la API y la app móvil. |
|  | 2. Seguridad CI/CD con respecto a secrets | Detectar y eliminar credenciales expuestas en el código del repositorio para evitar accesos no autorizados. |
|  | 3. Análisis de dependencias | Identificar vulnerabilidades en librerías de terceros que puedan ser explotadas. |
|  | 4. Capacitación básica | Concientizar al equipo de desarrollo sobre los riesgos de seguridad y la importancia de las buenas prácticas. |
| **Mes 3-4** | **Fase 2: Profundización e integración** |  |
|  | 1. Modelado de Amenazas | Identificar y documentar los posibles vectores de ataque en la API y la app móvil. |
|  | 2. Pruebas DAST en Pre-producción | Integrar pruebas de seguridad dinámicas para encontrar vulnerabilidades en la aplicación en ejecución. |
|  | 3. Creación de un "Security Champion" | Capacitar a un desarrollador de cada equipo para que lidere y promueva las prácticas de seguridad. |
|  | 4. Desarrollo de Estándares de Codificación Segura | Crear un documento con buenas prácticas de codificación para la API y la app. |
| **Mes 5-6** | **Fase 3: Modelo de madurez de la postura** |  |
|  | 1. Pentesting | Realizar un test de penetración integral para validar todas las medidas de seguridad implementadas. |
|  | 2. Programa de gestión de vulnerabilidades | Establecer un flujo claro para la remediación de vulnerabilidades, con SLAs definidos. |
|  | 3. Definir métricas de seguridad | Medir el progreso del programa de seguridad y reportarlo a la dirección. |
|  | 4. Política de AppSec formal | Documentar una política oficial de seguridad de aplicaciones que guíe el desarrollo futuro. |

El cronograma presentado es una guía clara de los pasos a seguir. Ahora bien, respecto a la columna de "Responsables", la dejé abierta porque eso ya depende del personal que la empresa tenga y de quiénes sean los más aptos para cada tarea. La idea es que la compañía asigne a las personas indicadas para que cada actividad avance.

### **Capacitación:**

La capacitación para los desarrolladores debe ser interactiva y gamificada para que sea efectiva. La meta es que en lugar de ser una charla teórica de dos horas que se olvida, la seguridad se convierta en una habilidad que practican y perfeccionan de forma continua.

Mi estrategia se basa en usar plataformas que ya tienen un enfoque de "aprender haciendo". En lugar de crear un entorno desde cero, aprovecho estas herramientas que ya están probadas:

Fase 1 - En lugar de una presentación de dos horas, haríamos sesiones más cortas y concisas, de 30-45 minutos. En estas, presentaríamos los conceptos básicos del OWASP Top 10 y nos enfocaríamos en las vulnerabilidades más relevantes. La idea es que la teoría sea el mapa antes de empezar la aventura.

Fase 2 - Inmediatamente después de la teoría, pasaríamos a la acción. Usaríamos plataformas como TryHackMe para que los desarrolladores pongan en práctica los conceptos de la presentación en un ambiente guiado. Luego, para el desarrollo de habilidades más avanzadas, usaríamos Hack The Box con retos más complejos y gamificados.

Fase 3 - Para evaluar lo aprendido, organizaría un "Capture The Flag" interno. Sería un reto práctico donde los equipos tienen que encontrar y corregir vulnerabilidades en una aplicación de prueba lo que nos da un indicador real de lo que aprendieron. Además, se mantendría un ciclo de refuerzo continuo. Si el pipeline detecta una vulnerabilidad, se le asignaría un módulo de capacitación específico al desarrollador responsable para que refuerce ese conocimiento, asegurando que los errores no se repitan.

Fase 4 - También se podría realizar un reto final donde los equipos tengan que auditar una aplicación de prueba en busca de vulnerabilidades y proponer un plan de remediación. Los desarrolladores que se destaquen en este reto y en los de Hack The Box serían identificados como los futuros "Security Champions", recibiendo mentoría para que puedan liderar las iniciativas de seguridad en sus equipos.