Evaluación Técnica Parte #1 - Consultor Senior de Seguridad de Aplicaciones

Consultor:

Erazo Mendoza Jeremy Sebastián

Offensive Security and Exploit Development Engineer

Quito, Junio del 2025

Contenido

Define métricas clave para medir la mejora en seguridad (KPIs)

En un entorno como el de esta fintech —con baja madurez en políticas de desarrollo seguro, una API expuesta y antecedentes de incidentes por malas prácticas de autenticación— es fundamental establecer un conjunto de métricas que nos permitan medir, de forma técnica y objetiva, si realmente estamos mejorando la postura de seguridad y reduciendo la exposición al riesgo operativo.

Los siguientes KPIs no son solo métricas de cumplimiento; son indicadores diseñados para tener impacto real en el ciclo de desarrollo, en la toma de decisiones de ingeniería y en la sostenibilidad del modelo a largo plazo:

- **Tiempo medio de remediación (MTTR) para vulnerabilidades críticas o altas:** Este KPI nos dice si los equipos están resolviendo los problemas más urgentes en un plazo aceptable. En una organización que ya ha tenido incidentes por malas implementaciones, como en este caso, el objetivo no es solo detectar fallos, sino corregirlos antes de que lleguen a producción. Un MTTR elevado puede indicar que no hay priorización técnica real, o que los ciclos de release son demasiado lentos para reaccionar frente a riesgo activo.
- **Tasa de builds bloqueadas por findings de seguridad en el pipeline:** Medimos qué porcentaje de builds están siendo detenidas automáticamente por reglas de seguridad en herramientas como SAST o SCA. Esto es clave para saber si nuestros controles automáticos están funcionando como filtros reales, o si solo estamos reportando vulnerabilidades sin accionarlas. En etapas tempranas es normal que el número sea alto, pero si no desciende, es señal de resistencia o deuda técnica estructural.
- **Porcentaje de versiones desplegadas con SBOM completo y versionado:** El SBOM no es algo decorativo: si mañana aparece una CVE explotable en una librería como jsonwebtoken o express, necesitamos saber exactamente qué versión del backend o qué release móvil la incluye. Este KPI nos da trazabilidad real ante incidentes, algo indispensable en fintechs expuestas a auditorías y regulaciones.
- **Ratio de secretos gestionados en vault frente a secretos expuestos:** La gestión de secretos sigue siendo uno de los problemas más frecuentes que veo en startups que están creciendo rápido. Este KPI permite medir cuántas credenciales están siendo almacenadas de forma segura y centralizada (por ejemplo, en HashiCorp Vault o AWS Secrets Manager), y cuántas siguen apareciendo en repositorios, pipelines o entornos. Si no se controla esto, cualquier key mal protegida puede terminar siendo un vector de ataque lateral.
- **Adopción efectiva del checklist de seguridad en Pull Requests:** Más allá de las herramientas, la seguridad se construye en la revisión diaria. Este KPI nos permite ver si los equipos realmente están evaluando los puntos críticos durante los PRs: validaciones de entrada, autenticación, uso correcto de funciones criptográficas, etc. No basta con poner un checklist en GitHub; lo importante es que se aplique de forma consistente y que sirva como herramienta de refuerzo técnico.

- **Ratio de vulnerabilidades reabiertas:** Este indicador mide la calidad de los fixes. Muchas veces los desarrolladores cierran findings por presión de tiempo, pero sin resolver realmente la raíz del problema. Cuando eso pasa, el hallazgo vuelve a aparecer en el siguiente escaneo o en una nueva revisión manual. Un ratio alto de reabiertos muestra una señal clara: estamos corrigiendo mal o demasiado rápido.

Definición de principales Quick Wins técnicos y de cultura

Considerando el escenario actual de la fintech —una organización en crecimiento, con bajo nivel de madurez en desarrollo seguro, exposición a través de una app móvil conectada a una API REST, y antecedentes de incidentes por autenticación débil— se han identificado una serie de acciones tácticas inmediatas (*quick wins*) que pueden generar un impacto medible y sostenible en la postura de seguridad, tanto a nivel técnico como cultural.

Quick Wins Técnicos

1. **Reforzar el flujo de autenticación sin rediseño completo del backend**
   Implementar OAuth 2.0 con PKCE para la app móvil, con validación robusta de los tokens JWT en la API y control explícito de sesión (fingerprint de dispositivo + ID de sesión único por login). Esta medida mitiga riesgos de *session hijacking*, *replay* o *token reuse*, sin necesidad de un rediseño arquitectónico.
2. **Integrar SCA con política progresiva de control sobre librerías vulnerables**
   Incorporar herramientas como Snyk o Dependency-Track al pipeline CI/CD, pero con una política de build permisivo al inicio: el build no se detiene, pero se genera una alerta y se registra el artefacto como "riesgoso". Esto permite visibilidad sin generar fricción en los equipos durante la transición.
3. **Clasificación lógica de endpoints para aplicar políticas diferenciadas por riesgo**
   Categorizar los endpoints de la API (ej. auth, transferencias, notificaciones, catálogos públicos) y aplicar políticas según el perfil de riesgo: mayor tiempo de expiración, lógica de auditoría, validación reforzada, control de tasa más estricto, etc. Esto reduce la exposición sin afectar todos los servicios por igual.
4. **Transición inmediata de secretos hacia gestión centralizada por identidad**
   Eliminar la gestión manual de secretos en pipelines y entornos de prueba, utilizando mecanismos de *workload identity* (como IAM Roles en AWS, o Google Workload Identity Federation). Esto elimina uno de los vectores más comunes de acceso lateral en CI/CD.
5. **Incorporación selectiva de reglas de SAST alineadas a riesgos del negocio**
   No se busca cobertura total inmediata, sino reglas personalizadas para casos relevantes: validaciones incorrectas de montos, exposición de errores por logs, bypass de lógica de negocio. Esto permite resultados de mayor calidad, con menor volumen de falsos positivos.

Quick Wins de Cultura Técnica

1. **Designación de “curadores de seguridad” por equipo técnico**
   En vez de los tradicionales “champions”, se asignan perfiles técnicos responsables de mantener buenas prácticas y apoyar la implementación de los controles de seguridad en su equipo. Esta figura no requiere jerarquía, pero sí criterio técnico. Ayuda a institucionalizar la seguridad dentro del equipo sin intervención externa constante.
2. **Generación de una memoria técnica de errores de seguridad en la organización**
   Formalizar un espacio (puede ser un canal interno, una wiki técnica o un repositorio) donde se documenten los errores de seguridad detectados, cómo se solucionaron y qué aprendizaje dejaron. Esta práctica evita repetir fallos comunes y promueve una cultura de transparencia técnica.
3. **Publicación de especificaciones mínimas de seguridad contextualizadas por stack**
   Cada stack tecnológico debe tener su propia hoja de referencia rápida con prácticas de seguridad mínimas: librerías aceptadas, funciones prohibidas, recomendaciones de configuración, patrones anti-patterns. Esto es mucho más efectivo que un estándar genérico, porque se vuelve operativo.
4. **Visualización explícita de deuda técnica de seguridad en el backlog**
   Cada hallazgo que no se corrige de forma inmediata debe pasar al backlog con visibilidad compartida. Esto obliga a los equipos a priorizar el riesgo desde una perspectiva técnica, y no a ocultarlo detrás de decisiones administrativas.
5. **Introducción de ejercicios prácticos internos estilo “tabletop attack”**
   Simulaciones de ataques sobre endpoints reales, ejecutadas por desarrolladores en tiempo controlado, permiten mejorar el entendimiento de vectores de ataque sin necesidad de contratar pentesters externos de forma continua. Este enfoque forma al equipo desde la práctica ofensiva.