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

Fase 1 – Meses 1 y 2

En los dos primeros meses, el objetivo debe centrarse en eliminar riesgos evidentes que puedan ser explotados con un nivel de esfuerzo bajo o intermedio, especialmente aquellos vinculados a la autenticación, control de acceso y exposición innecesaria de información. La fintech ha demostrado tener fallas en estos puntos y, sin visibilidad y respuesta inmediata, es altamente probable que se repitan los incidentes.

Antes de implementar procesos o políticas, se debe establecer una línea base técnica sobre el estado real del producto, tanto en comportamiento en producción como en su arquitectura de desarrollo. No se trata solo de revisar código o configuraciones, sino de validar de forma práctica cómo responde el sistema bajo condiciones adversas. En este sentido, se requiere una evaluación ofensiva dirigida:

- Analizar el mecanismo actual de autenticación desde la app móvil hacia la API REST.
- Verificar si los tokens tienen vencimiento, si hay control de revocación, si se implementa correctamente PKCE (en caso de OAuth2), y si el backend realiza validaciones firmes en lugar de confiar en el cliente.
- Revisar el modelo de autorización: asegurar que los controles de acceso no sean exclusivamente del lado del cliente y que no existan fallos de privilegios horizontales o verticales.

Este análisis debe ser realizado desde una lógica ofensiva, no solo desde una auditoría de cumplimiento. El objetivo no es detectar todos los bugs, sino aquellos vectores que pueden permitir comprometer cuentas, exfiltrar datos, o escalar privilegios.

Simultáneamente, se deben desplegar herramientas de análisis automatizado sobre el código y sus dependencias. La incorporación de análisis estático (SAST) permitirá identificar patrones de riesgo comunes que no siempre son evidentes, como validaciones ausentes, manejo inseguro de errores, o exposición de datos en logs. Por su parte, el análisis de componentes (SCA) identificará versiones vulnerables de bibliotecas utilizadas y establecerá un SBOM inicial. Este inventario será esencial en fases posteriores, ya que muchas vulnerabilidades no provienen del código propio sino de la cadena de suministro.

Una vez cubierto el análisis técnico, es necesario abordar de inmediato la gestión de secretos. En la mayoría de los productos en crecimiento, es común encontrar claves de acceso, tokens de servicios y credenciales embebidas directamente en el código o gestionadas sin control en pipelines de CI/CD. Esta práctica representa uno de los vectores de inicialización de ataque más frecuentes. Se debe identificar y eliminar cualquier secreto estático, e implementar un sistema de gestión centralizado con control de acceso granular y rotación periódica.

Por último, todos los hallazgos deben consolidarse y documentarse bajo un modelo de trazabilidad clara: descripción técnica, nivel de riesgo (basado en impacto real y probabilidad), recomendación técnica de remediación, responsable asignado y plazo de corrección. No es viable que seguridad se limite a emitir reportes: el ciclo de seguimiento debe establecerse desde esta fase.

La expectativa al cierre de este primer bloque no es que la organización tenga una postura madura, sino que haya eliminado los puntos de falla básicos que comprometen la operación. A partir de ahí, será posible diseñar controles sostenibles e integrarlos al flujo de trabajo sin fricción innecesaria.

Fase 2 – Meses 3 y 4

Con las vulnerabilidades críticas contenidas y un diagnóstico claro del entorno, el siguiente paso consiste en comenzar a integrar controles de seguridad directamente en el flujo de desarrollo, sin interrumpir la velocidad de entrega ni generar fricción innecesaria con los equipos de ingeniería. El foco aquí debe estar en la automatización de verificaciones, la formalización de criterios técnicos mínimos y la adopción de prácticas sostenibles que permitan elevar el estándar de calidad de manera transversal.

Incorporar seguridad en el pipeline de CI/CD es una prioridad ineludible. En este punto ya no se trata solo de detectar fallas, sino de establecer filtros efectivos que impidan que código con riesgo llegue a entornos productivos. Para ello, se deben incorporar escaneos estáticos (SAST) en las etapas de integración, empleando herramientas como Semgrep, correctamente ajustadas al contexto técnico del producto. No basta con reglas genéricas; es necesario trabajar reglas personalizadas que reflejen las decisiones arquitectónicas y patrones comunes del stack de desarrollo actual.

La seguridad en dependencias también debe integrarse como control automático. Un SCA como Snyk o Trivy debe correr con cada build, identificar vulnerabilidades conocidas (por CVE), verificar licencias, y sobre todo, generar un SBOM por versión compilada. Este material debe ser almacenado como artefacto, no solo como reporte, y asociado a cada release. En el contexto fintech, donde la trazabilidad es crítica ante cualquier auditoría o incidente, el SBOM deja de ser un extra y se convierte en parte esencial del ciclo de vida del software.

Tan importante como detectar problemas es evitar que pasen inadvertidos. Por ello, los resultados de SAST y SCA deben tener incidencia directa en el flujo de aprobación: si una vulnerabilidad crítica nueva se detecta, el pipeline debe fallar. Si hay una vulnerabilidad que persiste por motivos técnicos o de negocio, debe estar registrada formalmente como deuda técnica y controlada por criterios explícitos, no por omisión.

Complementando lo automatizado, se deben introducir prácticas mínimas pero formales en la revisión de código. Esto no implica procesos burocráticos, sino claridad en lo que se espera revisar: validación de entradas, control de acceso, manejo seguro de datos sensibles, uso correcto de funciones criptográficas, y tratamiento adecuado de errores. Lo ideal es que esto se documente como un checklist de seguridad integrado a los PRs, que sirva como guía viva y se mantenga alineado con los errores recurrentes detectados en el análisis inicial.

En paralelo, la organización necesita un conjunto base de políticas técnicas que marquen el mínimo aceptable para cualquier componente nuevo o modificado. Estas políticas deben enfocarse exclusivamente en áreas críticas: autenticación, protección de datos sensibles, gestión de sesiones, control de secretos y configuración de entornos. No se trata de diseñar un manual extenso e inmanejable, sino de redactar criterios concretos

que se puedan traducir en condiciones técnicas dentro del pipeline, o reglas aplicables por los equipos de desarrollo en su trabajo diario.

Una consideración importante en esta etapa es establecer ownership real sobre los componentes de seguridad. Cada módulo o microservicio debe tener responsables designados que puedan evaluar riesgos, tomar decisiones sobre excepciones y responder técnicamente ante findings no remediados. Esto permite establecer trazabilidad efectiva y una base sobre la cual se pueda construir una gobernanza más formal en fases posteriores.

Fase 3 – Meses 5 y 6

Con los controles mínimos ya integrados en el ciclo de desarrollo y la superficie de ataque primaria contenida, el foco en esta última etapa debe trasladarse hacia la sostenibilidad del modelo, la formalización de procesos y la maduración progresiva de la cultura técnica en torno a la seguridad. A diferencia de las fases anteriores, que estuvieron orientadas a contención y control, aquí la prioridad pasa a ser la **visibilidad, la repetibilidad y la adopción estructural** de prácticas seguras por parte del equipo.

Lo primero es establecer métricas claras que permitan monitorear la evolución de la postura de seguridad de forma objetiva. Estas métricas deben centrarse en indicadores técnicos reales, no en cumplimiento nominal. Algunos ejemplos clave que deben instrumentarse desde el entorno técnico:

- Porcentaje de builds bloqueadas por findings críticos en SAST/SCA.
- Tiempo medio de remediación de vulnerabilidades por criticidad (MTTR).
- Ratio de secretos gestionados en vaults frente a claves expuestas o compartidas informalmente.
- Porcentaje de código nuevo cubierto por reglas personalizadas de análisis estático.
- Cobertura de SBOM por versión de artefacto desplegado.

Estos indicadores no solo permiten mostrar avance, sino también identificar cuellos de botella reales en los procesos técnicos. Además, sirven como base para conversaciones informadas entre seguridad y desarrollo, desplazando la narrativa de “seguridad como freno” hacia “seguridad como acelerador del delivery confiable”.

En paralelo, debe implementarse un programa de formación continua en AppSec. A estas alturas, los errores que se siguen detectando ya no son atribuibles solo a desconocimiento, sino a la falta de asimilación de las prácticas esperadas. El programa no debe ser genérico ni voluntario. La formación debe diseñarse con base en los errores reales cometidos durante las fases anteriores. Si se detectaron patrones recurrentes en validaciones, manejo de tokens o exposición de logs, esos deben ser los módulos prioritarios de capacitación.

Una práctica efectiva en entornos de ingeniería es establecer un modelo de *AppSec Champions*, donde se identifique a uno o dos referentes técnicos por equipo de desarrollo que puedan actuar como punto de contacto directo con el equipo de seguridad, facilitar la adopción de controles, y elevar consultas sin burocracia. Este modelo reduce el desgaste

operativo del equipo de AppSec, pero también fortalece la apropiación técnica del modelo de seguridad desde dentro del equipo de ingeniería.

Por otro lado, debe iniciarse la definición de un marco mínimo de respuesta ante incidentes relacionados con la capa lógica de la aplicación. No se trata aún de construir un CSIRT formal, pero sí de establecer un procedimiento mínimo de notificación, validación técnica, asignación de responsables, y documentación de impacto ante cualquier evento relacionado con autenticación, datos sensibles o mal uso de la API. Incluso una simulación controlada de incidente (*tabletop exercise*) con los equipos involucrados permite validar si los flujos son realistas y si existen puntos de fallo que no están documentados.

Finalmente, es crítico iniciar una conversación estructurada con las áreas de producto y desarrollo sobre el ciclo de vida de vulnerabilidades no corregidas. A esta altura, existirán hallazgos que no han sido remediados, bien sea por dependencia técnica, complejidad o decisiones de negocio. Este es el momento para formalizar ese backlog como **deuda técnica de seguridad**, visibilizada, gestionada por riesgo y revisada periódicamente. La normalización silenciosa de vulnerabilidades en nombre de la agilidad es, en la práctica, una de las mayores causas de compromiso en fintechs con crecimiento acelerado.

El cierre de esta fase marca el punto donde la seguridad ya no depende exclusivamente de un equipo o una herramienta, sino que comienza a insertarse como parte natural del ciclo de vida del producto. Con indicadores, prácticas sostenibles, referentes internos y políticas mínimas aplicadas, la organización puede dar el siguiente paso hacia un modelo más robusto sin necesidad de frenar su ritmo de crecimiento.