Ingeniería de la Ciberseguridad
Curso 2023/2024

Caso práctico 1:

Nivel de madurez de un equipo de desarrollo.

Análisis y Gestión del Riesgo

Realizado por:

Gabriel Izquierdo González

Mario Ruano Diaz

Juan Antonio Suárez Suárez

ÍNDICE

Entregable 1: Listado de 15 prácticas o controles, y justificación de su elección

Hemos optado por las prácticas/controles específicos (organizados en dimensiones) para el desarrollo de software en ChaseMyCash, tomando como punto de referencia el modelo OWASP DSOMM y el estándar OWASP ASVS. A continuación, detallamos dichas prácticas/controles seleccionados.

Codificación y despliegue

- **Compilación (Build)**: Es crucial contar con procesos definidos, optimizados y documentados para la compilación del código. Este control está directamente vinculado con otros controles seleccionados, siendo fundamental para el desarrollo seguro.
- **Despliegue (Deployment):** Garantizar un despliegue eficiente es esencial para permitir a los desarrolladores probar la aplicación de diversas maneras, ahorrando tiempo y evitando posibles problemas de seguridad.
- **Backups regulares:** La responsabilidad del equipo de desarrollo incluye implementar y automatizar la realización de copias de seguridad periódicas. Esto implica identificar los datos y sistemas críticos que requieren respaldo, configurar herramientas automáticas de respaldo, definir políticas de retención de datos y realizar pruebas regulares de restauración para asegurar la disponibilidad y la integridad de los datos en caso de pérdida o fallo del sistema.
- **Gestión de versiones y parches:** Facilita la rápida detección y solución de vulnerabilidades del software, garantizando la seguridad continua de la aplicación mediante actualizaciones regulares de sus elementos y bibliotecas. Asimismo, la administración de versiones mantiene el código central ordenado y estructurado, promoviendo la colaboración entre los miembros del equipo de desarrollo.

Cultura y organización

- **Formación y orientación**: La formación en DevSecOps es necesaria para los empleados, especialmente aquellos involucrados en el desarrollo, garantizando un desarrollo seguro y la prevención de errores en etapas tempranas.
- **Seguridad en las comunicaciones**: Las comunicaciones cliente-servidor deben estar cifradas para proteger los datos del cliente contra amenazas externas.

Implementación

- **Verificación de código malicioso**: Implementar medidas de detección y gestión de código malicioso es crucial al interactuar con la aplicación por I/O o red.
- **Validación/sanitización de input**: Analizar correctamente las entradas de datos previene ataques de inyección.
- **Control de privilegios, acceso y autenticación:** La gestión de privilegios, acceso y autenticación se refiere a las prácticas y herramientas que aseguran que únicamente las personas autorizadas puedan entrar a los recursos y datos confidenciales en un ambiente de desarrollo. Esto involucra establecer políticas de seguridad y configurar roles y permisos basados en el principio de "menor privilegio". Esto implica el uso de herramientas automatizadas para realizar pruebas de seguridad y verificar la autenticación y el acceso antes de poder hacer cualquier tipo de implementación dentro del código.
- **Desarrollo y control de fuentes:** Contribuye a fortalecer la seguridad de las aplicaciones web al detectar y corregir vulnerabilidades desde el inicio del proceso de desarrollo, además de simplificar la administración segura de todo el ciclo de vida del desarrollo de software

Recopilación de información

- **Logs**: se ha optado por este enfoque pues los registros predeterminados se almacenan únicamente de forma estándar y es esencial configurarlos para supervisar exhaustivamente las actividades y eventos en las aplicaciones y sistemas.
- **Monitoreo**: los registros predeterminados no se emplean de forma explícita para realizar monitoreo alguno, por lo tanto, es crucial utilizarlos para establecer conexiones entre los incidentes y generar alertas de manera efectiva.

Test y verificación

- **Análisis estático de código**: Realizar análisis estático de código para detectar funciones problemáticas antes de que se conviertan en deudas técnicas.
- **Análisis dinámico de código**: Detectar vulnerabilidades tempranas mediante técnicas de análisis dinámico de código.
- **Consolidación:** Es fundamental para administrar la seguridad de un sistema de manera efectiva, ya que ofrece una visión completa de los riesgos y permite que los responsables tomen decisiones informadas sobre las acciones correctivas a aplicar.

Entregable 2: Modelo de madurez con las tablas explicativas para los 15 controles

Para cada uno de los controles seleccionados en la sección previa, se han definido los niveles de madurez de acuerdo con la Tabla 1. El nivel 1 representa un entendimiento muy básico, prácticamente nulo, de las prácticas de seguridad, el nivel 2 implica la aplicación de prácticas de seguridad básicas, el nivel 3 conlleva la implementación de prácticas de seguridad de alto nivel, y, por último, el nivel 4 implica la ejecución de prácticas avanzadas de seguridad a gran escala.

#	Control	Nivel 1	Nivel 2	Nivel 3	Nivel 4
1	**Compilación (Build)**	No hay un proceso definido que detalle cómo se lleva a cabo la compilación de la aplicación.	Hay un procedimiento documentado que describe cómo se compila y ejecuta la aplicación.	Al concluir la compilación, se proporciona una lista detallada del inventario de software (SBOM, por sus siglas en inglés). Esta lista enumera todas las versiones de cada dependencia de la aplicación para identificar posibles vulnerabilidades que puedan afectarla.	El código generado por la aplicación se firma digitalmente para prevenir su manipulación y la ejecución de código no autorizado, como el malware.
2	**Despliegue (Deployment)**	No hay un proceso definido que detalle cómo se lleva a cabo el despliegue de la aplicación.	Hay un procedimiento documentado que describe cómo se implementa la aplicación.	Se utilizan imágenes de repositorios de confianza.	Se integra un sistema de gestión de variables de entorno seguras. Además, se realiza una verificación automatizada para garantizar que la aplicación se haya desplegado correctamente.
3	**Backups**	No hay políticas ni procesos formalizados para la gestión de respaldos	Se implementan respaldos de manera ocasional, pero aún carecen de una estructura formal.	Se establecen políticas y procesos formales para la gestión de respaldos	Se monitorean y evalúan continuamente los procesos de respaldo para mejorar la eficiencia y la seguridad.

4	**Gestión de versiones y parches**	Ausencia de procesos definidos. Se aplican actualizaciones de forma reactiva ante incidentes críticos, sin una estrategia planificada.	Implementación de políticas básicas. Se planifican actualizaciones aunque no de manera sistemática, reconociendo la importancia de mantener el software actualizado.	Establecimiento de procesos formales y herramientas de gestión. Se llevan a cabo evaluaciones periódicas de vulnerabilidades para identificar y mitigar riesgos.	Integración completa en la estrategia de seguridad. Uso de herramientas avanzadas y automatización para una gestión eficiente y continua de versiones y parches.
5	**Formación y Orientación**	La empresa no ofrece capacitación en ciberseguridad a sus empleados o proporciona una formación muy básica en seguridad para prevenir fallos en el software, como una inyección SQL que podría ser explotada.	Cada equipo cuenta con un responsable de seguridad, se proporciona formación regular en seguridad para todos los empleados, así como formación periódica para los responsables de seguridad. Además, se reconoce y premia la buena comunicación y transparencia en la empresa.	Se llevan a cabo talleres de seguridad, como "build it, break it, fix it", con el fin de mejorar el aprendizaje de componentes seguros para su posterior desarrollo. También se realizan revisiones periódicas de seguridad del código fuente (SCA), en las cuales participan desarrolladores y administradores de sistemas.	La seguridad se integra en todos los equipos, lo que significa que todos los miembros del equipo del proyecto son responsables de la seguridad. Además, se proporciona formación regular en seguridad para el personal externo que participe en el desarrollo de software.
6	**Seguridad en las comunicaciones**	Las comunicaciones cliente-servidor no están cifradas y viajan en texto plano.	Todas las comunicaciones cliente-servidor están cifradas y se emplea una pasarela de pago segura desde la aplicación.	Las comunicaciones en la red interna de la organización están cifradas mediante WPA2 y se asegura de que el WPS esté desactivado. Además, los certificados digitales TLS se actualizan y generan adecuadamente.	Se lleva a cabo una auditoría anual de las comunicaciones internas y externas (cliente-servidor) de la empresa para verificar su seguridad. Existe una política clara sobre qué se puede y qué no compartir a través de la red.

7	**Verificación de código malicioso**	No se realizan búsquedas de código malicioso.	Se realiza una verificación periódica del código fuente completo, así como de las bibliotecas de terceros utilizadas.	Se formaliza el proceso y se desarrolla un programa de formación y procedimientos para los analistas de código.	El procedimiento está estandarizado y se integra en el desarrollo de la aplicación. Los desarrolladores más centrados en la seguridad revisan el código antes de que pase a la rama de preproducción. Se proporciona una lista de verificación para asegurar que esté libre de ataques basados en tiempo, llamadas a servidores o destinos no autorizados, así como para identificar posibles puertas traseras, rootkits, etc.
8	**Validación y saneamiento de entrada**	No se lleva a cabo la evaluación de las entradas de datos.	El equipo de desarrollo posee conocimientos sobre los posibles ataques que pueden ocurrir, así como métodos para prevenirlos.	Se están estandarizando ciertas pruebas para las entradas de datos. El equipo está capacitado en desarrollo seguro para prevenir ataques de inyección. Antes de pasar a producción, se ha realizado un análisis exhaustivo de todas las entradas de datos.	El equipo de desarrollo está plenamente consciente de los posibles ataques de inyección que podrían ocurrir. Antes de pasar a la rama de preproducción, todas las entradas son sometidas a un riguroso análisis mediante una serie de pruebas diseñadas para demostrar que la aplicación está protegida contra dichos ataques.

9	**Control de privilegios, acceso y autenticación**	En este nivel, las políticas de acceso son débiles y se asignan privilegios de forma indiscriminada. La autenticación es básica y el monitoreo de actividades es limitado, lo que deja a la organización vulnerable a amenazas de seguridad.	Se implementan políticas básicas de seguridad y se mejora la autenticación con medidas como contraseñas seguras y autenticación de dos factores. Aunque aún hay áreas de mejora, se reconoce la importancia de proteger los sistemas contra accesos no autorizados.	En este nivel, se establecen procesos sólidos y se aplican políticas claras de gestión de accesos. Se implementa la autenticación multifactorial y se realiza un monitoreo continuo para detectar y responder a actividades sospechosas, mejorando la seguridad de la organización.	Se integran tecnologías avanzadas y automatización para una gestión más eficiente de privilegios y accesos. Se utiliza análisis avanzado para una respuesta proactiva ante amenazas, promoviendo una cultura de seguridad cibernética en toda la organización y garantizando un entorno seguro y protegido.
10	**Desarrollo y control de código fuente**	No hay prácticas ni herramientas formalizadas para el control de versiones ni la gestión de código fuente.	Se utilizan prácticas básicas de control de versiones, pero aún no hay un enfoque formal.Implementación de un sistema de control de versiones básico.	Se establecen políticas y procesos formales para el control de versiones y gestión de código fuente. Documentación de políticas claras para el control de versiones y gestión de código fuente. Implementación de un flujo de trabajo definido para la gestión de cambios.	Se monitorean y evalúan continuamente los procesos de control de versiones para mejorar la eficiencia y la seguridad. Implementación de métricas para evaluar la efectividad del control de versiones. Revisión y auditoría regular de los cambios en el código fuente.

11	**Gestión Logs**	Se envían todos los registros generados por el sistema, la web, la infraestructura, etc.	Los registros se almacenan teniendo en cuenta el Reglamento General de Protección de Datos (RGPD) al momento de la supresión de los datos de información personal identificable (PII).	La aplicación genera y envía registros de actividad (logs) por sí misma.	Los registros de la propia aplicación se correlacionan con el monitoreo para poder detectar y lanzar alertas de manera efectiva.
12	**Monitoreo**	Se recopilan métricas tanto de las aplicaciones como del sistema para identificar incidentes y cuellos de botella.	Se establecen umbrales para las métricas, y se generan alertas cuando estos umbrales son sobrepasados. Además, las métricas recopiladas se visualizan en tiempo real.	Se recopilan métricas avanzadas relacionadas con la disponibilidad y estabilidad, y se desactivan aquellas métricas que no se utilizan para evitar el consumo innecesario de recursos. Además, se agrupan las métricas para acelerar el análisis.	Se recopilan métricas de control y defensa, y se combinan con pruebas para ayudar a identificar errores de programación. Además, se dispone de un panel de control relacionado para visualizar los incidentes.
13	**Análisis estático de código**	No se llevan a cabo pruebas estáticas en el código de la aplicación.	Se revisa que las dependencias utilizadas en la aplicación no contienen vulnerabilidades conocidas.	Se implementan herramientas de análisis de código estático para detectar funciones vulnerables y/o obsoletas.	Antes de agregar nuevo código a producción, se emite un informe de vulnerabilidades al programador. Además, se habilita la eliminación de código que no se utiliza.

14	**Análisis dinámico de código**	No se lleva a cabo ningún tipo de análisis dinámico.	En el proceso de compilación de CI/CD, se incorpora un paso en el que se realiza un análisis de la aplicación utilizando herramientas automatizadas que informan sobre vulnerabilidades.	Todas las vulnerabilidades reportadas por el escáner se centralizan en una aplicación, lo que permite visualizar las diferencias entre las vulnerabilidades introducidas en cada iteración del código.	Se incluyen diversas herramientas DAST con el objetivo de aumentar las posibilidades de detección de vulnerabilidades. Además, se realiza un análisis de cobertura del código para identificar las partes que aún no están cubiertas por pruebas.
15	**Consolidación**	No se ha implementado el control de seguridad en absoluto.	Se identifican y gestionan los falsos positivos de manera rudimentaria, y se da prioridad a la resolución de defectos críticos que podrían comprometer la seguridad del sistema.	Se proporciona una forma básica de visualizar los defectos detectados, lo que facilita la comprensión de su naturaleza y gravedad.Se implementa una visualización básica de los defectos identificados durante el proceso de desarrollo.	Se recopilan y analizan datos sobre la gestión de parches y las respuestas a problemas de seguridad, lo que permite mejorar la eficacia de los procesos de gestión de vulnerabilidades y la capacidad de respuesta ante incidentes.

Tabla 1. Controles y niveles de madurez asociados.

Entregable 3: Punto de partida y nivel de madurez objetivo para cada control (junto con su justificación)

#	Control	Punto de partida	Nivel deseado
1	**Compilación**	Nivel 1. Falta documentación del proceso de compilación.	Nivel 3: Implementar un proceso documentado en 6 meses, incluyendo la creación de una lista SBOM para garantizar la seguridad de las dependencias *Justificación:* La documentación detallada y la gestión de dependencias seguras son esenciales para la seguridad y confiabilidad del código. La SBOM reduce riesgos de seguridad.
2	**Despliegue**	Nivel 1. No hay un proceso detallado de despliegue.	Nivel 3. Establecer un proceso para gestionar dependencias y actualizarlas regularmente en 6 meses. *Justificación:* Un despliegue eficiente y documentado mejora la seguridad y la confiabilidad. El uso de imágenes de repositorios confiables reduce riesgos..
3	**Backups**	Nivel 1: No hay políticas ni procesos formalizados para la gestión de respaldos. *Justificación:* En este punto, las políticas y procesos de respaldo son limitados o inexistentes. El punto de partida implica la concienciación sobre la importancia de los respaldos y la implementación de respaldos ad hoc. Se comienza a considerar la retención y recuperación de respaldos.	Nivel 4: Se monitorean y evalúan continuamente los procesos de respaldo para mejorar la eficiencia y la seguridad. *Justificación:* El objetivo es establecer políticas y procesos formales para la realización, retención y recuperación de respaldos. Esto incluye la documentación de políticas claras, la implementación de un plan de respaldos regular y la concienciación y formación sobre la importancia de la seguridad en la gestión de respaldos. Esto sienta las bases para una gestión de respaldos más robusta y eficiente.
4	**Gestión de versiones y parches**	Nivel 1: Ausencia de procesos definidos. Se aplican actualizaciones de forma reactiva ante	Nivel 4: Implementar en 6 meses la integración completa de seguridad como herramientas avanzadas y automatización para la gestión de parches y versiones.

		incidentes críticos, sin una estrategia planificada.	*Justificación: establecer políticas y procesos formales para la gestión de versiones y la aplicación de parches son esenciales para mantener un entorno actualizado y seguro.*
5	**Formación y orientación**	Nivel 1. Falta formación en seguridad.	Nivel 4. Implementar formación en 6 meses con pequeños cursos formativos y talleres. *Justificación:* La formación continua es clave. Implementar talleres y cursos en seguridad mejora la conciencia y la calidad del código.
6	**Seguridad en las comunicaciones**	Nivel 1. Comunicaciones cifradas, pero se requiere mejora.	Nivel 4.Se busca llevar a cabo auditoria anual de las comunicaciones internas y externas en 6 meses. *Justificación: :* El objetivo es establecer políticas y procesos formales para garantizar la seguridad en las comunicaciones. Esto incluye la implementación de protocolos de seguridad, como TLS para cifrar las comunicaciones. Se deben documentar las políticas y procedimientos para el manejo seguro de datos en tránsito, y se debe concienciar al personal sobre las amenazas y mejores prácticas.
7	**Verificación de código malicioso**	Nivel 1. No hay búsquedas de código malicioso	Nivel 3. Realizar formaciones intensivas en 6 meses con el fin de formar a los analistas de código. *Justificación: La capacitación en la detección de malware mejora la seguridad del código y el conocimiento del equipo.*
8	**Validación/sanitización de input**	Nivel 1. Falta evaluación de entradas de autenticación.	Nivel 4. Mejorar conexiones internas en 6 meses con el fin de que todas las entradas sean sometidas a un riguroso análisis mediante una serie de pruebas. *Justificación:* Reforzar conexiones y evaluar la red interna contribuye a una comunicación más segura. *Justificación: Concienciar al equipo sobre posibles ataques y la prevención de los mismos*

			fortalecerá la seguridad del código.
9	**Control de privilegios, acceso y autenticación**	Nivel 2. El equipo cuenta con licencia Teams de Github que les proporciona herramientas avanzadas de control de acceso, autenticación y autorización además del registro "Github Audit Log"	Nivel 4. Integrar de manera mejorada las herramientas proporcionadas por Github en 6 meses y emplear el uso de técnicas de Inteligencia Artificial para analizar grandes cantidades de datos y patrones de comportamiento sospechosos. *Justificación:* El equipo de desarrollo debería integrar aún más estas herramientas. Esto garantiza que la gestión de identidades y accesos sea una parte integral de todo el ciclo de vida del desarrollo *Justificación:* Utilizar técnicas de inteligencia artificial y aprendizaje automático. El equipo de desarrollo podría considerar la implementación de estas técnicas para analizar y monitorear el entorno de desarrollo en tiempo real para detectar cualquier actividad sospechosa.
10	**Desarrollo y control de código fuente**	Nivel 2. Varias ramas en funcionamiento.	Nivel 3. Implementar fases superiores para alcanzar este nivel. *Justificación*: Con ramas de producción y prueba en funcionamiento, la implementación de fases superiores es alcanzable y fortalecerá la madurez del control.
11	**Gestión de Logs**	Nivel 2. Varias ramas en funcionamiento. Los registros se almacenan teniendo en cuenta el Reglamento General de Protección de Datos (RGPD)	Nivel 3. Implementar fases superiores para alcanzar que la aplicación genere y envíe registros de logs por si misma. *Justificación:* la gestión efectiva de logs es esencial para la seguridad, la conformidad normativa, la eficiencia operativa y la capacidad de respuesta rápida a incidentes

12	**Monitoreo**	Nivel 1. Recopilación de métricas, pero sin umbrales ni alertas.	Nivel 4. Establecer umbrales y alertas en 6 meses con el fin de recopilar métricas de control y defensa para identificar errores de programación. *Justificación:* Establecer umbrales y alertas mejora la capacidad de respuesta y facilita la toma de decisiones en tiempo real.
13	**Análisis estático de código**	Nivel 1: Solo test unitarios y de integración automáticos.	Nivel 3: Realizar análisis estático en 6 meses con el fin de tener herramientas análisis de código para detectar funciones vulnerables u obsoletas. *Justificación:* El análisis estático previene problemas graves y mejora la calidad del código.
14	**Análisis dinámico de código**	Nivel 1. No se realizan análisis dinámicos de código.	Nivel 3: Incluir análisis dinámico en 6 meses con el fin de poder reportar todas las vulnerabilidades encontradas en cada iteración del código. *Justificación: El* análisis dinámico detecta vulnerabilidades en tiempo de ejecución, mejorando la seguridad de la aplicación.
15	**Consolidación**	Nivel 1. No se ha implementado el control de seguridad en absoluto.	Nivel 3: En 6 meses se puede conseguir implementar una visualización básica de los defectos identificados durante el proceso de desarrollo. *Justificación:* Gestionar eficazmente la seguridad y el desarrollo de software, sino que también es una práctica clave para promover la eficiencia operativa, la mejora continua y la toma de decisiones fundamentada en datos sólidos

Tabla 2. Niveles de madurez actuales vs. niveles de madurez deseados.

Además, hemos condensado la información de la Tabla 2 en los diagramas de tela de araña que se muestran a continuación, para que ChaseMyCash pueda tener una representación gráfica de la situación actual de su aplicación y los objetivos a alcanzar después de los primeros 6 meses.