UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

Tareas necesarias para Producir software bien protegido (PW),

Seminario de Seguridad en el Desarrollo de Software

Grupo:2304

Integrantes:

Erick Carlos Miralles Sosa:erickcms@estudiantes.uci.cu
Marcos Daniel Artiles Delgado:marcosdad@estudiantes.uci.cu
Sabrina D'Lory Ramos Barreto:sabrinadlrb@estudiantes.uci.cu
Keylan Valdés García:keylanvg@estudiantes.uci.cu
Sheila Hernández Falcón:sheilahf@estudiantes.uci.cu

Introducción

Cualquier ejecución de software requiere la realización de un conjunto de tareas con el fin de garantizar seguridad,es de acá donde surgen estas estipuladas por el NIST para lograr Producir software bien protegido (PW).Dentro de él se encuentran tareas fundamentales.

Fundamentalmente a este grupo de tareas pertenecen la utilización de formas de modelado de riesgos, el seguimiento y mantenimiento de requisitos,incorporar soporte para el uso de funciones y servicios de seguridad ,revisar el diseño del software para confirmar y hacer cumplir q con todos los requisitos de seguridad ,otro punto sería la adquisición y mantenimiento de componentes de software bien seguros y seguir con las prácticas de encriptación necesarias.

Desarrollo

El uso modelado de riesgos ,de amenazas y de ataques obviamente permite conocer las debilidades a la que se enfrenta la insitución en cierto desarrollo de software.En este labor, si existe una correcta conformación de equipos de ciberseguridad, es donde entran en función los equipos Red Team pues son los encargados de penetrar las defensas e informar al Blue Team sobre las vulnerabilidades,todo esto con el fin de eliminarlas,permitendo a la institución tomar medidas de reforzamiento de la ciberseguridad antes que sean vulnerados por verdaderos “ciberdelicuentes”.

Lo mencionado anterirormente conlleva a la creación de informes donde se identifican y caracterizan los distintos tipos de datos con los que se interactúa a la hora de desarrollar el software,también se llevan a cabo evaluaciones para la protección de datos confidenciales, protección de la identificación, la autenticación y el control de acceso, incluida la gestión de credenciales. y generalmente también de por sí los Red Team proporcionan los informes necesarios a los Blue Team y estos a su vez realizan informes a la empresa y llevan a la prática la solución de vulnerabilidades. Aunque si tienen el presupuesto necesario se recomiendan a las institcuiones la contratación de un tercer team, el Purple Team, quien es el que garantiza la comunicación entre los otros dos equipos, pues estos muchas veces por celos o por desconocimiento del lenguaje técnico del otro no siempre llegan a un intercambio positivo.

También es necesario la capacitación del personal, pues se pueden tomar las medidas necesarias por parte de los equipos de ciberseguridad que si un trabajador no protege los bienes informáticos puede conllevar a desastrosas pérdidas para la empresa, por tanto esta debe tomar medidas y acciones como la contratación de expertos que den charlas y modos de actuar sobre el correcto comportamiento de los empleados

.

Llevar las medidas anteriores a la práctica es una tarea fundamental por lo que alguno de estas medidas pudieran ser:

1. **Creación de un Marco de Clasificación de Datos**: Se debe diseñar un marco de clasificación de datos sólido y fácil de entender, que determine los niveles de clasificación y los controles de seguridad asociados.Donde cada empleado solo pueda acceder a la información y permisos que le correspondan .
2. **Implementación de la Clasificación de Datos**: Una vez desarrollado el marco, se debe implementar en la tecnología elegida, como Microsoft 365, utilizando etiquetas de sensibilidad para proteger los datos mediante encriptación y marcado de contenido.
3. **Automatización y Capacitación**: Para manejar grandes volúmenes de datos, se recomienda la automatización del proceso de clasificación de datos. Incluye capacitar a los usuarios clasificando correctamente los datos en su trabajo diario.
4. **Gestión de Riesgos y Cumplimiento**: La clasificación de datos debe integrarse con las políticas de prevención de pérdida de datos (DLP) y otras aplicaciones de cumplimiento para reducir la superficie de ataque hacia los datos sensibles.
5. **Evaluación y Mejora Continua**: Es importante establecer un flujo de trabajo continuo para clasificar datos nuevos o actualizados.

Pudiendo además deestas medida y acciones garantizar el Seguimiento y mantenimiento de los requisitos de seguridad, riesgos y decisiones de diseño del software siempre y cuando se apliquen de manera periódica.

Incorporar soporte para el uso de funciones y servicios de seguridad estandarizados en el desarrollo de software es una práctica recomendada que aporta beneficios significativos en términos de eficiencia, seguridad y mantenimiento del software. Esto implica integrar el software con sistemas existentes de gestión de registros, gestión de identidades, control de acceso y gestión de vulnerabilidades en lugar de crear implementaciones

propietarias de funciones y servicios de seguridad. Algunas maneras de hacerlo serían:

1. Integración con Sistemas de Gestión de Registros: Utilizar servicios de registro estandarizados como ELK (Elasticsearch, Logstash, Kibana) o Splunk para recopilar, almacenar y analizar los registros de la aplicación. Esto facilita la detección temprana de anomalías y ataques, .
2. Gestión de Identidades: Implementar soluciones de gestión de identidades como OAuth 2.0 o OpenID Connect para manejar la autenticación y autorización de usuarios.
3. Control de Acceso: Integrar con sistemas de control de acceso como LDAP o Active Directory para gestionar el acceso a los recursos de la aplicación. Esto asegura que solo los usuarios autorizados tengan acceso a ciertas funcionalidades, reduciendo el riesgo de acceso no autorizado .
4. Gestión de Vulnerabilidades: Utilizar herramientas de gestión de vulnerabilidades como Nessus o Qualys para escanear el software en busca de vulnerabilidades conocidas y aplicar parches de seguridad de manera oportuna. Esto ayuda a mantener la seguridad del software a lo largo del tiempo .
5. Integración Continua de Seguridad: Asegurar que la seguridad se integre en todas las etapas del ciclo de vida del desarrollo de software (SDLC), no solo como una fase de prueba final. Esto implica aplicar prácticas de seguridad desde el principio, como el desarrollo seguro , revisiones de seguridad y pruebas de penetración .

Otra tarea fundamental es el cumplimiento del diseño del software lo vea el profesional más capacitado o menos, pero es necesario mirarlo y cumplir con las inidicaciones necesarios las cuales se describen y se logran

1. **Definición de Requisitos de Seguridad**: Antes de la revisión, es crucial identificar y definir claramente los requisitos de seguridad necesarios para el software. Esto incluye la identificación de amenazas,

riesgos y factores de cumplimiento, recomendándose el uso de un sistema de Gestión del Ciclo de Vida del Desarrollo de Aplicaciones (ADLM), pues permite una información mucho más fluida

2. **Capacitación y Concienciación**: Aunque parezca repetitivo es uno de los aspectos que aparecen como necesario en todas las etapas y tareas del desarrollo seguro de software pero es fundamental pues ayuda a asegurar que todos los miembros del equipo comprendan sus responsabilidades y estén equipados para aplicar prácticas seguras,para llevarse a la práctica pues a través de reuniones y cursos.
3. **Uso de un Marco de Desarrollo de Software Seguro**: Alinear las prácticas con un marco de desarrollo de software seguro bien establecido, como el Marco de Desarrollo de Software Seguro (SSDF) de NIST, puede proporcionar estructura y consistencia a los esfuerzos del equipo

La adquisición y el mantenimiento de componentes de software bien seguros, como bibliotecas, módulos, middleware y marcos de terceros son de gran valuarte para las organizaciones logrando con ellos:

- **Reducción de Costos**: La integración de la seguridad en el ciclo de vida del desarrollo de software (SDLC) puede ayudar a reducir los costos asociados con la corrección de problemas de seguridad y evitar tener que comprar otros o reparar estos bienes logrando ahorro significativo.
- **Mejora de la Calidad del Software**: El enfoque en el desarrollo de software seguro y confiable puede resultar en un software de mejor calidad que cumple con las necesidades de los usuarios finales y ayuda a las organizaciones a alcanzar sus objetivos comerciales.
- **Mayor Seguridad**: Integrar la seguridad en el proceso de desarrollo de software ayuda a reducir el riesgo de brechas de seguridad y ciberataques, protegiendo la reputación de una organización y minimizando el impacto de los incidentes de seguridad.

También acá es necesario realizar análisis sobre cada uno de estos componentos de software de terceros ejemplo de los cuales pueden ser

Revisión y Evaluación de Componentes de Software de Terceros

· I**dentificación de Componentes de Terceros**: Utilice herramientas de gestión de dependencias para identificar todos los componentes de terceros utilizados en su proyecto. Esto puede incluir bibliotecas, frameworks y APIs.

· **Evaluación de Componentes**: Realice una evaluación de seguridad de cada componente de terceros utilizando herramientas de análisis de código estático y dinámico, así como bases de datos de vulnerabilidades conocidas.

· **Revisión Periódica**: Establezca un proceso de revisión periódica para los componentes de software de terceros, especialmente si se prevé que se utilicen de manera diferente en el futuro.

· **Actualizaciones y Parches**: Asegúrese de aplicar actualizaciones y parches de seguridad a los componentes de terceros de manera oportuna para mitigar las vulnerabilidades conocidas.

Todo lo anterior mencionado varía según las necesidad y los riegos pues existen varios tipos, el riesgo operacional,estratégico,reputacional, de seguridad de la información,de la reputación y de la transición.

Configuraciones Seguras para Componentes de Software

- **Definición de Configuraciones Seguras**: Determine las configuraciones seguras para los componentes de software de terceros y documente estas configuraciones.

· **Implementación de Configuraciones como Código**: Utilice herramientas de gestión de configuración como Ansible, Chef o Puppet para automatizar la implementación de estas configuraciones seguras.

Información de Procedencia para Componentes de Software

- **Generación de SBOM**: Utilice herramientas de gestión de inventario de software (SBOM) para generar una lista detallada de todos los componentes de software utilizados, incluyendo su procedencia y versión.Pudiendo ser estas Oddo Inventory,Alegra,Zoho Inventory
- **Análisis de Composición de Fuente y Binario**: Realice análisis de composición de fuente y de software binario para identificar dependencias y evaluar el riesgo de seguridad de cada componente.

Repositorio de Software para Componentes de Código Abierto Autorizados

1. **Establecimiento de Repositorio:** Cree un repositorio privado de software para alojar componentes de código abierto autorizados. Utilice herramientas como GitHub Enterprise o Bitbucket para gestionar el acceso y la revisión de código.
2. **Revisión y Examen:** Realice revisiones de seguridad y examenes de código en los componentes de código abierto antes de incorporarlos al repositorio.
3. **Documentación de Procedimientos:** Documente los procedimientos para la revisión, examen y aprobación de componentes de código abierto para asegurar la consistencia y la seguridad.

Procesos para Actualizar Componentes de Software

1. **Monitoreo de Actualizaciones**: Establezca un proceso de monitoreo para las actualizaciones de los componentes de software, utilizando herramientas de gestión de dependencias y feeds de vulnerabilidades.Empleando herramientas como Paessler PRTG para el monitoreo de infraestructuras de servidores físicos y virtuales,Site24*7,Zabbix.
2. **Pruebas de Actualización**: Realice pruebas de seguridad y funcionalidad en las nuevas versiones de los componentes antes de su implementación en el entorno de producción.

Como paso final es necesario cumplir estrictamente tanto con las medidas anterirormente mencionadas como las otras que estimula el NIST tanto en el área de Producir Software Bien Seguro (PW) como en las demás pues para lograr producir un software con esas características es necesario primero la existencia de una organización sólida, la protección de este seguro para posteriormente se pueda producir y posteriormente para poder responder a vulnerabilidades y lograr mejora constantemente.

1. **Preparar la Organización (PO)**: Asegurarse de que las personas, procesos y tecnología de la organización estén preparados para realizar el desarrollo de software seguro. Esto implica la formación y capacitación de los desarrolladores en las prácticas de codificación segura, así como la implementación de políticas y procedimientos que apoyen el desarrollo seguro desde el inicio [1].
2. **Proteger el Software (PS)**: Implementar medidas para proteger todos los componentes del software contra la manipulación y el acceso no autorizado. Esto incluye el uso de sistemas de gestión de acceso, el cifrado de datos en reposo y en tránsito, y la implementación de firewalls y sistemas de detección de intrusiones [1].
3. **Producir Software Bien Seguro (PW)**: Asegurarse de que el software producido tenga vulnerabilidades de seguridad mínimas en sus versiones. Esto se logra mediante la adopción de prácticas de codificación segura, como la validación de entrada, el manejo seguro de errores, la inyección de dependencias seguras y la minimización del código. Además, se deben realizar revisiones de seguridad de código y pruebas de penetración para identificar y corregir vulnerabilidades [1]

También son necesario llevar a la práctica estos enfoques , como pudiera ser:

- **Validar todas las entradas y validar y codificar correctamente todas las salidas**

Asegurarse de que todas las entradas al sistema sean validadas contra una lista de permisos aceptables y rechazar cualquier entrada que no cumpla con las especificaciones. Además, codificar correctamente todas las salidas para evitar vulnerabilidades como Cross-Site Scripting (XSS),empleando herramientas como **OWASP Input Validation Cheat Sheet**: para guiar el proceso de validación de entradas [3].**OWASP Output Encoding Cheat Sheet**: Para asegurar la codificación correcta de las salidas [3].**Herramientas de Pruebas de Seguridad Automatizadas**: Para realizar pruebas de fuzzing y análisis estático y dinámico en busca de vulnerabilidades de seguridad

- **Evitar el uso de funciones y llamadas no seguras**

Utilizar funciones y APIs seguras proporcionadas por el lenguaje de programación o el framework en uso. Evitar el uso de funciones que presenten riesgos de seguridad conocidos, pudiéndose usar guías proporcionadas por estándares como OWASP-

Proporcionar capacidades de registro y rastreo

· Implementar capacidades de registro y rastreo para monitorear el comportamiento del sistema, identificar anomalías y facilitar la investigación de incidentes de seguridad.Esto se puede lograr **Sistemas de Gestión de Logs**: Como ELK (Elasticsearch, Logstash, Kibana) o Splunk para recopilar, almacenar y analizar logs.

Conclusiones

Con el estudio, comprensión y análisis de la información proporcionada por el NIST en la descripción en las Tareas de Desarrollo Seguro se arribaron a las siguientes conclusiones:

- El SSDF(Marco de Desarrollo de Software Seguro) ofrece un marco estructurado para alinear y priorizar las actividades de desarrollo de software seguro con los requisitos de negocio, tolerancias de riesgo y recursos de una organización
- Las prácticas del SSDF están diseñadas para ser medibles y orientadas a resultados,o sea teniendo como objetivo que cada institución se compare en estas prácticas para saber que “tan bien” están y cuanto le faltan.
- Al seleccionar y aplicar las prácticas del SSDF, es importante considerar factores como el costo, la factibilidad y la aplicabilidad,pues si la organización no presenta los recursos adecuados no se pueden implementar todos los consejos.
- Cada grupo necesita de la correcta gestión e implementación del otro para que en ellos existe una correcta seguridad, pues van de la mano

Referencias Bibliográficas

1. "Software Engineering for Secure Systems: Industrial and Research Perspectives" de Marco Vieira y William A. Enck.

2. "Secure Coding in C and C++" de Robert C. Seacord.

3. "Building Secure Software: How to Avoid Security Problems the Right Way" de John Viega y Gary McGraw.

4. "Computer Security: Art and Science" de Matt Bishop.

5. "Software Security: Building Security In" de Gary McGraw.

6. "Secure Software Development: A Risk Management Approach" de Jason Grembi.

7. "The CERT Guide to Secure Coding" de Robert C. Seacord.

8. "Secure Software Development: A Practitioner's Guide" de Mark G. Graff y Kenneth R. van Wyk.

9. "Software Security: Building Security In" de Fredrick P. Brooks Jr.

10. "Secure Coding in C and C++" de David LeBlanc y Michael Howard.

11. "Software Security: Theories and Systems" de Shengbing Wang y Josep Domingo-Ferrer.

12. "Secure Coding in C and C++" de Dean F. Jerde.

13. "Secure Software Development: Building Security In" de Cemil Demirarslan.

14. "Software Security: Theories and Applications" de Shiuhpyng Winston Shieh.

15. "Secure Coding in C and C++" de Lidong Zhang y Christophe Morin