UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

El empleo de las buenas prácticas de las alarmas, notificaciones y administraciones de seguridad en el desarrollo de software

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

El mundo del desarrollo de software es un entorno dinámico donde el enfoque en la implementación de tecnologías ágiles y robustas debe ir acompañado de una sólida infraestructura de seguridad. Es crucial no solo centrarse en la funcionalidad y eficiencia del software, sino también en garantizar su integridad y protección ante posibles amenazas cibernéticas. En este contexto, la correcta aplicación de protocolos de seguridad, como auditorías, alarmas, notificaciones, y otras medidas de adminstración de seguridad, desempeñan un papel fundamental en la administración de la seguridad del software.

Este estudio se enfoca en ampliar el entendimiento de los protocolos y técnicas que se utilizan para asegurar un desarrollo de software robusto y protegido. Se hará especial hincapié en la implementación eficaz de alarmas y notificaciones, la realización exhaustiva de auditorías periódicas, y la adecuada administración de la seguridad en todas las etapas del ciclo de vida del software. Estas prácticas son clave para garantizar que el software desarrollado cumpla con los estándares de seguridad requeridos y brinde una experiencia segura para los usuarios finales.

El objetivo final es promover la conciencia y la adopción de prácticas de seguridad efectivas en el desarrollo de software, contribuyendo a la construcción de un entorno digital más protegido y confiable para todos los usuarios

Desarrollo

Todo proceso de seguridad en cualquier trabajo tanto relacionado con la informática o no tiene la necesidad de cumplir con ciertos protocolos o medidas de seguridad para evitar robos de información y vulnerabilidad.Una de las principales medidas para aplicar en estos sistemas son las auditorías, pues son realizadas por una entidad externa que aseguran que se esté cumpliendo lo establecido.

La auditoría es un proceso que busca mejorar la gestión de una organización, por lo tanto la necesidad de estas es necesaria en práticamente cualquier institcuión, obviamente también necesitan de su aplicación las empresas que se dedican al desarrollo de software para cumplir diversos parámetros de seguridad.

Para la realización de todo proceso auditorio es necesario definir los pasos que se deben seguir:

1. **Planificación**:

En este paso se definen que análisis se va a llevar a cabo, pues,generalmente las auditorías son en áreas y parámetros específicos.En el caso de las auditorías en el desarrollo de software ser destinadas al equipo de levantadores de requisitos , a los desarrolladores, especialistas en ciberseguridad.

2. **Recopilicación de información y procedimientos para auditar**

En cambio acá es donde los auditores se encargaría de realizar una o varias listas de chequeos para comprobar las condiciones de la institución en el área que será auditada.

Las preguntas que se pueden realizar en una lista de chequeo pueden variar sobre todo en el estado del software, si está en planeación, fue

iniciado,entregado, las preguntas varían según el estado de ejecución del proyecto.

Ejemplo de una lista de chequeo en un software que todavía no se ha empezado a desarrollar pudiera ser

- ¿Se han definido claramente los requisitos de seguridad para el software?
- ¿Se ha realizado un análisis de riesgos de seguridad para identificar posibles amenazas y vulnerabilidades?
- ¿Se han establecido políticas de control de acceso y se han implementado mecanismos de autenticación fuertes?

En el caso que ya el proyecto esté en fase de desarrollo las preguntas pudieran ser las mismas,con la única variación de comprobar si fueron realizadas,ejemplo:

- ¿Se establecieron políticas de control de acceso y mecanismos de autenticación fuertes?
- ¿Se utilizó cifrado de datos sólido tanto en reposo como en tránsito?
- ¿Se implementaron medidas de seguridad para proteger contra ataques de inyección, como SQL Injection?

3. Ejecución de la Auditoría

En el paso 3, como dice su nombre es llevar a cabo lo que se planificó sobre todo teniendo en cuenta lo señalado en la realización de las listas de chequeos.

4.Informe de la Auditoría

Es el momento de cierre, cuando ya se llevó a cabo la auditoría siendo necesaria compartir esta información con la empresa auditada para conjunta o individualmente tomen las medidas requeridas para corregir errores.

5.Seguimiento

Una auditoría no es realmente efectiva si no se trabaja por la constancia de lograr que los parámetros defectuosos detectadas hayan sido corregidos

Recolección de trazas

Con la realización de los pasos mencionados anteriormente se detectan trazas, las cuales son registros detallados de las operaciones realizadas por los usuarios, los sistemas o los procesos de software. Los cuales pueden incluir acciones realizadas, cambios en la configuración, accesos a datos sensibles, errores del sistema, entre otros eventos relevantes para la seguridad informática. La recolección de trazas permite a los auditores tener una visión completa de lo que sucede en el sistema, facilitando la detección de actividades anómalas o maliciosas.

Este proceso es fundamental para identificar vulnerabilidades, prevenir ataques y asegurar la integridad y confidencialidad de la información.Estas trazas se pueden obtener con las ya mencionadas listas de chequeos, pero sobre todo para esto se emplea el sistema de gestion de logs(SIEM),los logs son archivos que registran eventos específicos dentro de un sistema.

Para llevar a la práctica un sistema de gestión de logs primeramente tendríamos que seleccionar un programa de gestión de logs ejemplo pudiera ser ELK Stack, la cual es es una combinación de tres herramientas de código abierto: Elasticsearch, Logstash y Kibana, que trabajan juntas para proporcionar una solución completa de análisis y monitoreo de registros.(Javier Lasheras,mayo 2023). Siendo necesario seguir los siguientes pasos:

1.Instalar estas 3 herramientas siguiendo las instrucciones de instalación proporcionadas en la documentación oficial de Elastic.

2.Crear archivo de configuración para Logstash que especifique cómo recopilar los logs de la fuente, el siguiente ejemplo recopila logs de un servidor web que registra accesos al puerto 80:

```
input {
    file {
        path => "/var/log/nginx/access.log"
        start_position => "beginning"
    }
}
output {
    elasticsearch {
        hosts => ["localhost:9200"]
        index => "nginx-access-%{+YYYY.MM.dd}"
    }
}
```

3.Ejecutamos la siguiente instrucción en la línea de comandos: bin/logstash -f logstash.conf, lo que nos permite la ejecución de Logstash

4. Posteriormente abrimos un navegador y abrimos la dirección http://localhost:5601 y comprobamos la existencia de la máquina remota de Kibana , posteriormente entraremosa la interfaz de Kibana pudiendo interactuar con los datos.

5. En Kibana, se podrá ir a la pestaña "Management", seleccionado “Index Patterns y luego en Create index pattern, introducimos un nombre de ínidice

6. Visualizar los Datos de Log: Después iremos a la pestaña "Discover" para explorar y visualizar los datos de logs indexados por Logstash.

7. Aquí se podrá realizar búsquedas, filtrar datos, crear visualizaciones y paneles personalizados para analizar la información de los logs de forma interactiva.
8.Con todo lo anterior correctamente configurado ya tenemos listo a ELK Stack.

Para llevar a cabo una buena auditoría es necesario distinguir y diferenciar conceptos básicos como lo son trazas en la auditoría funcional y trazas en la auditoría de seguridad.

Trazas en la auditoría funcional

Las trazas para la auditoría funcional se centran en verificar que el software cumple con los requisitos y especificaciones definidos para su funcionamiento. Esto incluye la validación de procesos como la autorización, autenticación, manejo de sesiones, y validación de datos de entrada, entre otros. Estas trazas buscan asegurar que el comportamiento del software en su ejecución es el esperado y definido dentro de sus especificaciones, sin alteraciones que puedan afectar su funcionalidad.

Trazas en la auditoría de seguridad

Las trazas para la auditoría de seguridad se enfocan en identificar y mitigar vulnerabilidades que puedan comprometer la seguridad del software y de los datos que maneja. Esto incluye aspectos como el manejo de excepciones, auditoría y trazas (logs), servicios web, y criptografía. La auditoría de seguridad evalúa si el software protege adecuadamente contra modificaciones no autorizadas y si cumple con las políticas de seguridad establecidas, prestando especial atención a las recomendaciones y vulnerabilidades detectadas por organizaciones como OWASP y NIST.

Como es lógico las trazas necesita de cierta seguridad pues su caída en manos escrupulosas pudiera provocar efectos desastrosos para la institcuión.Por esto para garantizar la seguridad y confidencialidad de las trazas de auditoría obtenidas , es fundamental almacenar esta información en un lugar seguro y restringido, donde solo los miembros autorizados y responsables de la gestión de seguridad puedan acceder a ella.Las diferentes formas para salvaguardar esta información pudiera ser:

- **Servidor de Seguridad o Centro de Operaciones de Seguridad (SOC):**Permite el almacenamiento de trazas de auditoría, garantizando medidas de seguridad adicionales y acceso restringido para proteger los datos sensibles de la empresa.
- **Los SIEM:** mencionados anteriormente no solo son herramientas diseñadas analizar trazas de auditoría para la detección de amenazas , si no también permiten almacenar las trazas donde no solo garantiza su seguridad, sino que también facilita el monitoreo y análisis de eventos de seguridad.
- **Almacenamiento en la Nube Segura:** Es fundamental elegir proveedores de confianza con medidas sólidas de protección de datos,que nos ofrezcan buena seguridad.
- **Acceso Basado en Roles y Políticas de Seguridad:**Implementar controles de acceso basados en roles y políticas de seguridad para garantizar que solo los usuarios autorizados puedan acceder a las trazas de auditoría,
- **Encriptación de Datos Sensibles:**Aplicar encriptación a las trazas de auditoría almacenadas para proteger su confidencialidad

Si llevamos a la práctica lo anteriormente planteado se podrá decir que habremos realizado habremos realizado una auditoría bastante aceptable, pero para volverla más profesional deberíamos definir una política eventos que obligatoriamente tenemos que analizarle trazas para que no se olvide realizársela a estos eventos, los cuales son prioritarios de análisis, algunos de estos pudiera ser:

1. Inicio de Sesión y Cierre de Sesión: Los registros de inicio y cierre de sesiones de usuarios, incluyendo detalles como la hora, la ubicación, la dirección IP, el usuario, y si el acceso fue exitoso o fallido.

2. Cambios de Permisos y Accesos: Trazas que registren cualquier cambio en los permisos de usuario, ajustes de privilegios, concesión de accesos a recursos críticos, y actualizaciones en los roles de usuario.

3. Acceso a Datos Sensibles o Críticos: Registro de accesos a datos sensibles, modificaciones en registros importantes, transferencias de información crítica, y cualquier acción relacionada con activos de alto valor.

4. Errores y Excepciones: Registros de errores críticos, excepciones no controladas, fallos en la aplicación, y cualquier evento anómalo que pueda indicar posibles problemas de seguridad o vulnerabilidades.

5. Actividades de Auditoría y Monitoreo: Registro de acciones de auditores, análisis de seguridad, informes de cumplimiento normativo, monitoreo de eventos de seguridad, y cualquier otra actividad relacionada con la auditoría y el control.

6. Intentos de Acceso no Autorizados: Eventos que indican intentos de acceso no autorizados, intentos de intrusión, denegaciones de servicio, actividades sospechosas,.

La obligatoriedad de recogida de trazas a estos eventos es esencial para establecer una sólida vigilancia de la seguridad en el desarrollo de software, pero la recolección de trazas podría provocar problemas de espacio de disco, la no retención de registros críticos y la ineficiencia en registros de actividad, para evitar estos problema surge la necesidad de la aplicación de una política llamada rotación de trazas.

Rotación de trazas

Existen diferentes tipos de rotación de trazas , algunas de estas son:

1. **Basado en el Tiempo:** Eliminación de trazas después de un período específico de tiempo, como semanas, meses o años, o incluso solo alguna horas, dependiendo de los requerimientos de retención de datos ,las políticas de la empresa y la necesidad de confidencialidad de la información.

2. **Basado en el Tamaño:** División y eliminación de trazas una vez que alcanzan un tamaño máximo predefinido, lo que ayuda a controlar el crecimiento de los archivos de registro y a optimizar el espacio de almacenamiento.

3. Rotación por Ciclos: Mantenimiento de múltiples archivos de trazas rotativos de modo que los registros más antiguos se reemplacen o eliminen automáticamente al completar un ciclo definido (semana, mes, año, etc.).

4. **Conservación de Registros Clave:I**dentificación y conservación de registros críticos o de alta importancia durante períodos más largos, garantizando la disponibilidad de registros esenciales para auditorías y análisis forenses.

Estas técnicas de rotación permiten diversas ventajas:

1. Gestión de Espacio en Disco: Evita consumir espacio de almacenamiento y afectar el rendimiento del sistema.

2. Seguridad y Confidencialidad: Protege la información sensible contenida en las trazas mediante la eliminación segura de registros antiguos y la implementación de medidas de cifrado o anonimización.

3. Facilitar la Búsqueda y Análisis: Facilita la búsqueda, el análisis forense y la investigación de incidentes en caso de ser necesario.

Alarmas y notificaciones

De la vida prácticas sabemos que las alarmas son los dispositivos que suenan cuando llaman a nuestra puerto o son “mensaje de comunicación urgente” que nos mandan , mientras que de las notificaciones de las redes sociales sabemos que son los que nos avisa cuando cualquier evento que nos incumbe ha sucedido.Pues también están presentes en el desarrollo de software y prácticamente con la misma funcionalidad.

En el desarrollo de software, las alarmas desempeñan un papel crucial en la supervisión, detección temprana de problemas y notificación de eventos importantes. Su uso puede contribuir a mejorar la fiabilidad, disponibilidad y seguridad de las aplicaciones. Algunas formas en las que se utilizan las alarmas en el desarrollo de software incluyen:

1. Monitoreo del Desempeño: Las alarmas pueden configurarse para alertar sobre la degradación del rendimiento de una aplicación, como picos de carga en servidores, tiempos de respuesta lentos o consumo excesivo de recursos.

2. Detección de Errores: Las alarmas pueden activarse ante la aparición de errores críticos en la aplicación, como fallas en la autenticación, problemas de conectividad, excepciones no controladas, etc.

3. Seguridad y Vulnerabilidades: Las alarmas se utilizan para notificar sobre posibles brechas de seguridad, intentos de intrusión, accesos no autorizados o comportamientos anómalos que podrían indicar un ataque.

4. Disponibilidad del Sistema: Las alarmas pueden alertar sobre la caída de servicios, fallos en componentes clave o interrupciones en la disponibilidad del sistema para garantizar una pronta acción y resolución de problemas.

Para la aplicación de estas pueden realizarse diferentes técnicas:

1. **Integración con Herramientas de Monitoreo**: Se configuran alarmas en herramientas de monitoreo como Prometheus, Grafana, Nagios, entre otras, para supervisar métricas clave y generar alertas en tiempo real.

Para este paso , por ejemplo en Nagios para usar las alarmas el primer paso sería la instalación del propio servicio Nagios guiándonos por su documentación oficial, luego podríamos definir un servicio específico, por ejemplo, un servicio de ping a un host remoto editando el archivo configuración en esta dirección: ./user/local/nagios/etc/objects/localhost.cfg. Simplemente configuración la alarma que queramos hacer, cuando hagamos click mandemos un mensaje alarma, etcétera y después definiríamos el servicio y los comandos:

```
define service{
    use                     generic-service
    host_name               example-host
    service_description     Ping
    check_command           check_ping!100.0,20%!500.0,60%
}
```

Ejemplo de Definición de Comando para Notificaciones

```
define command{
    command_name    notify-by-email
    command_line    /usr/bin/printf "%b" "***** Nagios *****\\n\\nNotification Type: $NOTIFICATIONTYPE$\\nHost: $HOSTNAME$\\nState: $HOSTSTATE$\\nAddress: $HOSTADDRESS$\\nInfo: $SERVICEOUTPUT$\\n" | /usr/bin/mail -s "** $NOTIFICATIONTYPE$ alert - $HOSTNAME$ is $HOSTSTATE$ **" $CONTACTEMAIL$
}
```

Y como paso Final reiniciamos el Servicio de Nagios

2. Notificación a Equipos Responsables: Las alarmas se envían a equipos de desarrollo, operaciones o seguridad para abordar y solucionar rápidamente problemas identificados en el funcionamiento del software.

3. Automatización de Respuestas: Las alarmas pueden desencadenar acciones automáticas, como reinicios de servicios, escalado automático de recursos o ejecución de scripts de recuperación.

4. Creación de Paneles de Control: Se construyen paneles de control con visualizaciones de alarmas y métricas para una supervisión centralizada y una respuesta ágil a eventos críticos.

También se podría decir que las alarmas serían un tipo de notificación ,la diferencia es que cualquier mensaje de información por el usuario sería una notificación, mientras que las alarmas son “mensajes de urgencia” pero para aclarar específicamente tendrían las siguientes diferencias.

1. Propósito:

- Alarmas: Se utilizan para señalar situaciones críticas, como emergencias, problemas de rendimiento, fallas de seguridad o eventos importantes que requieren una acción inmediata.
- Notificaciones: Suelen informar sobre eventos o actualizaciones importantes, como mensajes recibidos, actualizaciones de software, recordatorios, entre otros, pero no necesariamente indican situaciones críticas.

2. Naturaleza de la Información:

- Alarmas: Transmiten información vital y prioritaria que necesita una atención urgente para abordar problemas graves o situaciones de riesgo.
- Notificaciones: Proporcionan información útil o relevante que puede variar en importancia, pero que no necesariamente requiere una acción inmediata.

3. Grado de Urgencia:

- Alarmas: Tienen un carácter urgente y crítico, requiriendo una respuesta rápida para resolver situaciones problemáticas o mitigar riesgos.
- Notificaciones: Tienen un grado de urgencia menor y suelen ser informativas, sin necesidad de una acción inmediata.

4. Impacto en la Operatividad:

- Alarmas: Tienen un impacto directo en la operatividad y estabilidad del sistema, alertando sobre eventos que pueden afectar la disponibilidad o funcionamiento del software.
- Notificaciones: Aunque pueden proporcionar actualizaciones importantes, no tienen un impacto inmediato en la operatividad del sistema y suelen ser más informativas que críticas.

Ejemplos

1. Notificación push: La aplicación enviará notificaciones push a los dispositivos móviles de los usuarios, informándoles sobre eventos importantes, como una intrusión o una emergencia.

2. Llamada telefónica automatizada: La aplicación llamará automáticamente a los números de teléfono configurados para informar sobre una situación de alarma, proporcionando detalles y recomendaciones sobre cómo actuar.

3. Mensaje de voz: La aplicación enviará un mensaje de voz pregrabado a través de un sistema de telefonía automatizado, proporcionando información sobre la alarma y las acciones a tomar.

4. Integración con sistemas de megafonía: La aplicación se integrará con los sistemas de megafonía del edificio para emitir una alerta audible en toda la instalación, informando a todas las personas presentes sobre la situación.

5. Mensaje instantáneo: La aplicación enviará mensajes instantáneos a través de plataformas de mensajería como Slack o Microsoft Teams, notificando a los miembros del equipo de seguridad sobre la alarma y proporcionando detalles adicionales.

6. Integración con sistemas de videovigilancia: La aplicación se integrará con los sistemas de videovigilancia del edificio, mostrando automáticamente

las cámaras relevantes en tiempo real cuando se reciba una alarma, facilitando la evaluación de la situación.

7. Alertas por redes sociales: La aplicación enviará alertas y notificaciones a través de las redes sociales corporativas, informando a los empleados y al público en general sobre situaciones de emergencia y proporcionando instrucciones sobre cómo proceder.

También es de gran importancia que las notificaciones y alarmas relevantes de seguridad solo sean recibidas por el personal adecuado para esto se pueden tomar las siguientes medidas

1. Configuración de niveles de notificación: En el sistema de desarrollo de software, se establecerán diferentes niveles de notificación para los errores, como "información", "advertencia" y "error crítico". Solo los errores críticos serán notificados a las personas encargadas, mientras que los demás niveles se registrarán en un archivo de registro interno sin mostrar detalles técnicos al usuario.

2. Mensajes genéricos para los usuarios: Cuando ocurre un error en la solución informática, en lugar de mostrar mensajes técnicos, se mostrarán mensajes genéricos al usuario, como "Se ha producido un error. Por favor, intente nuevamente más tarde" o "Lo sentimos, estamos experimentando dificultades técnicas". Estos mensajes no revelarán información técnica específica sobre el error.

3. Registro detallado de errores: Los errores serán registrados en un archivo de registro interno con información técnica detallada, como el tipo de error, la ubicación y el estado del sistema en el momento del error. Estos registros serán accesibles solo para las personas autorizadas, como los desarrolladores y administradores del sistema.

4. Notificaciones automáticas a las personas encargadas: Cuando se produzca un error crítico, el sistema enviará automáticamente una notificación a las personas encargadas, como los desarrolladores o el equipo de soporte técnico. Esta notificación contendrá información detallada sobre el error, permitiendo a los responsables tomar medidas inmediatas para solucionarlo.

5. **Acceso restringido a información técnica:** La información técnica sobre los errores, como los registros de errores y los detalles específicos del sistema, estará protegida y solo será accesible para las personas autorizadas. Se implementarán medidas de seguridad, como autenticación y cifrado, para garantizar que esta información no sea accesible para usuarios no autorizados.

Además de la seguridad que se debe emplear en la realización de la auditoría y en el control de las alarmas y notificaciones debemos tener una administración de seguridad independiente

Administración de la seguridad

En un desarrollo seguro de software, se puede emplear un Mecanismo de Administración de la Seguridad para gestionar usuarios, privilegios y análisis de trazas. Este mecanismo debe evitar el uso de herramientas de propósito general, como IDEs de desarrollo o herramientas de administración, a menos que sea estrictamente necesario y se justifique con previo aviso. Algunas medidas que se pueden toma serían:

1. Gestión de usuarios: Implementar un sistema de gestión de usuarios que permita crear, modificar y eliminar cuentas de usuario. Este sistema debe tener controles adecuados para verificar la identidad del usuario y garantizar que solo las personas autorizadas tengan acceso al sistema.

2. Gestión de privilegios: Establecer diferentes niveles de privilegios para los usuarios, asegurándose de que solo tengan acceso a las funciones y datos

necesarios para realizar sus tareas. Limitar los privilegios administrativos a un número reducido de personas autorizadas.

3. Análisis de trazas: Lo mencionado anteriormente, implementar un sistema de análisis de trazas que registre y monitoree las actividades realizadas por los usuarios en el sistema. Esto permitirá detectar comportamientos inusuales o sospechosos y tomar medidas preventivas.

4. Restricción del uso de herramientas generales: Evitar el uso de herramientas de propósito general, como IDEs de desarrollo o herramientas de administración, en entornos de producción. Estas herramientas pueden representar un riesgo de seguridad si se utilizan incorrectamente o si se accede a funciones no autorizadas.

5. **Justificación y aviso previo:** Si se utiliza una herramienta general en un entorno de producción, se debe justificar claramente el motivo y obtener la aprobación de los responsables de seguridad.Siendo necesario informar con antelación a todas las partes involucradas sobre el uso de la herramienta y las precauciones adicionales que se tomarán para garantizar la seguridad del sistema.

Además sería necesario para emplear un Mecanismo de Administración de la Seguridad seguir los siguientes pasos:

1. Identificar las características de la información a procesar: Determinar qué tipo de información se va a manejar, su nivel de confidencialidad, integridad y disponibilidad requerida. Esto ayudará a establecer los niveles de seguridad necesarios.

2. Establecer una administración centralizada: Implementar un sistema de administración centralizado que permita gestionar usuarios, privilegios y trazas de forma eficiente. Esto puede lograrse mediante el uso de un servidor web dedicado o utilizando un servidor existente con restricciones adicionales de seguridad.

3. Hospedar el mecanismo en un servidor separado: Si es posible, se recomienda alojar el Mecanismo de Administración de la Seguridad en un servidor web diferente al utilizado por los usuarios del sistema. Esto ayudará a reducir el riesgo de ataques externos y protegerá mejor los datos y la funcionalidad del mecanismo.

4. Aplicar restricciones de seguridad: Configurar el servidor web que hospeda el mecanismo para bloquear el acceso desde el exterior, utilizando firewalls, listas de control de acceso u otras medidas de seguridad. También se deben aplicar medidas de protección adicionales para asegurar el acceso interno, como autenticación multifactor o cifrado de datos.

5. Monitorear y auditar: Implementar un sistema de monitoreo y auditoría que registre las actividades realizadas en el Mecanismo de Administración de la Seguridad. Esto permitirá detectar intentos de acceso no autorizados o comportamientos sospechosos y tomar medidas correctivas.

6. Mantener actualizado el mecanismo: Realizar actualizaciones periódicas del software y aplicar parches de seguridad para proteger el Mecanismo de Administración de la Seguridad contra nuevas vulnerabilidades

Conclusiones

Después del estudio y análisis realizado sobre el desarrollo seguro de software con la aplicación de auditorías,alarmas,notificaciones y administración de seguridad podemos concluir:

- La aplicación de las auditorías son procesos complejos que llevan una especial organización,cuidado y colaboración entre los auditores y la insitución a auditar para lograr una auditoría completamente efectiva
- Las alarmas y notificaciones son claves para informar a los usuarios, pero a la misma vez hay que evitar que estos reciban información que no les corresponde.
- Es necesario aplicar una correcta adminstración de seguridad en todas las funcionalidades de la institución.
- Un desarrollo seguro de software, respaldado por auditorías y una administración efectiva de la seguridad, ayuda a proteger la reputación de la empresa y la confianza de los clientes

Bibliografía referenciada

- Javier Lasheras <<¿Qué es un stack de ELK y qué ventajas tiene implementarlo?>>,hiberus blog, 31-5-2023
- Jerry Oswald Quintanilla López <<Auditoria de desarrollo de Software>>,Prezi,octubre 2014.

Bibliografía consultada

- Ahmadi, M., & Kouhizadeh, M. (2017). Software security requirements engineering: A systematic review. Journal of Systems and Software, 124, 256-282.
- Anderson, R. (2008). Security engineering: A guide to building dependable distributed systems. John Wiley & Sons.
- Bishop, M. (2003). Computer security: Art and science. Addison-Wesley Professional.
- Dhillon, G. (2017). Principles of information systems security: Texts and cases. John Wiley & Sons.
- Gollmann, D. (2011). Computer security. John Wiley & Sons.
- Howard, M., & LeBlanc, D. (2003). Writing secure code. Microsoft Press.
- Landwehr, C., Bull, R., & McDermott, J. P. (1994). A taxonomy of computer program security flaws, with examples. DTIC Document.
- Lunt, T. F., & Hoyer, W. J. (1990). Issues in auditing computer-based systems. ACM Computing Surveys (CSUR), 22(3), 229-264.
- McGraw, G. (2006). Software security: Building security in. Addison-Wesley Professional.
- Pfleeger, C. P., & Pfleeger, S. L. (2014). Security in computing. Pearson Education.
- Rouse, M. (2015). Security information and event management (SIEM). TechTarget.
- Schneier, B. (2000). Secrets and lies: Digital security in a networked world. John Wiley & Sons.
- Shostack, A. (2014). Threat modeling: Designing for security. John Wiley & Sons.

- Sommer, P., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In Proceedings of the 2010 IEEE symposium on Security and Privacy (pp. 305-316).
- Stallings, W. (2017). Cryptography and network security: Principles and practice. Pearson Education.
- Gobierno de España <<Guía metodológica de auditoría funcional para Inspectores de Servicios de la AGE>>,diciembre de 2009
- Pérez Héctor <<La auditoría funcional basada en la medición de la gestión organizacional a través del Cuadro de Mando Integral>>8 de septiembre de 2022