Política de desarrollo seguro

ITAPP

Gestión de seguridad - POPBL
MONDRAGON UNIBERTSITATEA | MACC

Hoja de control

Organismo	**Equipo de desarrollo del producto ITAPP**		
Proyecto	ITAPP		
Entregable	Política de desarrollo seguro		
Autor	Haritz Saiz, Xabier Gandiaga, Álvaro Huarte, Xabier Etxezarreta, Onintza Ugarte y Ander Bolumburu		
Versión/Edición	0001	**Fecha Versión**	30/05/20
Aprobado por		**Fecha Aprobación**	08/06/20
		Nº Total de Páginas	9

REGISTRO DE CAMBIOS

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
0100	Versión inicial	Ander Bolumburu	25/05/20

CONTROL DE DISTRIBUCIÓN

Nombre y Apellidos
Ander Bolumburu Casado

Contenido

1. Objetivo

El objetivo de este control es garantizar la seguridad de la información en los entornos de diseño e implementación dentro del ciclo de vida de desarrollo de los sistemas de información.

Las medidas consideradas en este control son los siguientes:

- Controlar estrictamente los entornos de desarrollo, test, staging y producción.
- Los directivos responsables de los sistemas de aplicaciones deberían ser también responsables de la seguridad del proyecto o del entorno de producción. Ellos deberían garantizar que todas las propuestas de cambio en los sistemas son revisadas para verificar que no comprometen la seguridad del sistema.
    - Referirse a Requisitos de seguridad del software - Arquitectura
- Se debe contemplar la seguridad al ciclo de desarrollo del software en todas sus fases, desde el inicio hasta el final.
    - Referirse a Requisitos de seguridad del software – Principios de desarrollo seguro
- Se debe tratar el desarrollo e implementación como un proceso de cambio integrando las mejoras de seguridad en las actividades de gestión de cambios (documentación y formación para usuarios y desarrolladores)

Una falta de control en estos aspectos permite la materialización de potenciales amenazas:

- Compromiso de la información: Intercepción, espionaje en remoto, recuperación de medios reciclados, divulgación, datos de fuentes no fiables, manipulación de software, detección de la localización
- Fallos técnicos: Mal funcionamiento, saturación del sistema de información, mal funcionamiento del software, exposición de la mantenibilidad del sistema de información
- Acciones no autorizadas: Uso no autorizado de equipos, copia fraudulenta de software, corrupción de datos, comportamientos no autorizados, procesamiento ilegal de datos
- Compromiso de las funciones: Error en el uso, abuso de privilegios, suplantación de identidad, denegación de acciones.

2. Alcance

La presente Política de Desarrollo Seguro se dicta en cumplimiento de las disposiciones legales vigentes, con el objeto de gestionar adecuadamente la seguridad de la información y los sistemas informáticos de ITAAP.

Esta Política debe ser conocida y aplicada por todos los miembros del equipo. En caso de que los miembros del equipo de desarrollo no tengan la capacidad de evitar, encontrar y reparar vulnerabilidades, se les formará con el fin de que sean capaces.

El hecho de ser capaces consiste en tener las habilidades necesarias para desarrollar una funcionalidad nueva de la aplicación de forma segura y a su vez su correspondiente conjunto de validación que verifique el correcto funcionamiento del desarrollo. Todo esto tiene cabida en el pipeline el cual es el encargado de automatizar el proceso de validación de código y etiquetarlo en función del resultado.

A su vez, para desarrollar software de forma segura, se deben conocer las posibles amenazas que conciernen al mundo del desarrollo de hoy en día, se les formará con la finalidad de que sean capaces de contemplar la ciberseguridad en todos los campos del desarrollo.

3. Definiciones y roles

Mediante este apartado se procederá a definir tanto aspectos necesarios para la comprensión del control como los diferentes roles y responsables identificados:

3.1 Definiciones

- **Desarrollo Seguro**: Requisito para generar un servicio, arquitectura, software y sistema seguro, desde la perspectiva del resguardo de la información.
- **Vulnerabilidades:** Se refiere a alguna condición de debilidad o fragilidad que se encuentra presente en el activo identificado. Usualmente se traduce en una debilidad o ausencia de control, que posibilita la ocurrencia de un incidente y que pueden afectar a uno o más activos de información.
- **Equipo de Desarrollo:** Corresponde a los funcionarios institucionales o personal externo que forman parte del grupo a cargo de analizar y programar las funcionalidades de proyectos de desarrollo de software y sistemas.

3.2 Roles

- **Desarrolladores:** Cumplen cabalmente con las disposiciones y requerimientos establecidos en la presente política. Cada miembro del equipo deberá velar por la correcta implementación de las normas de desarrollo seguro de software, así como del cumplimiento por parte de su equipo de trabajo.
- **Jefe de desarrollo**: Encargado de verificar que el desarrollo marcha correctamente tal y como se define en las políticas.

4. Modo de operación

En este apartado se dará a conocer el proceso a seguir en el desarrollo del software.

Este se trata de un control en el que se establecen reglas para que la seguridad de la información sea tenida en cuenta en todo el proceso de desarrollo del software y en todo el ciclo de vida de este.

Cabe destacar que antes de comenzar a desarrollar nada se ha tenido en cuenta que el entorno en el que el software va a ser desarrollado es seguro. Para ello se ha procedido a la instalación de diferentes medidas de seguridad en cada uno de los equipos como pueden ser en control de acceso mediante credenciales/claves públicas.

Posteriormente, se ha contemplado la seguridad de la información detallando en un documento adjunto el proceso de diseño del software. Referirse al documento de Requisitos de Seguridad del Software. En él se han controlado, sanitizado y validado las diferentes entradas que pudiera tener el software con el fin de establecer las tecnologías para la confidencialidad e integridad de la información.

Además, en ese mismo documento se han definido unas pautas con el fin de obtener un diseño de código fuente seguro. Tras definir los requisitos de seguridad en la fase de diseño, es hora de conocer detalles sobre el almacenamiento y tratamiento del código fuente.

Resulta interesante resaltar que todo el código fuente de la aplicación debe ser alojado en GitLab con el fin de tenerlo centralizado y que cada uno de los aportes de código o commit puedan pasar por un proceso en el que se verifique (mediante un conjunto de validación) que el código tiene un mínimo de calidad y no rompe por completo la aplicación. Por lo que se podría definir cada aporte de código como un punto de verificación de la seguridad, ya que por cada aporte de código (commit) corresponde a un hito concreto.

La verificación consta tanto de tests del código como un análisis de código estático.

A la hora de gestionar las vulnerabilidades del software, la clave se encuentra en estos conjuntos de validación, ya que dependiendo de su precisión y variedad de situaciones que contemplen, mayor será porcentaje de intrusiones/vulnerabilidades que tendremos verificadas.

Además de este conjunto de validación se hará uso de una técnica conocida como Fuzzing. Esta consiste en probar con un gran abanico de tipos de inputs posibles para descubrir vulnerabilidades que no hayamos contemplado (por desconocimiento) a la hora de implementar el conjunto de validación.

Dentro del repositorio, la organización ha sido dividida en tres diferentes entornos (dev, pre-production y producción/máster) ya que esto facilita la organización del código fuente en base a su estrategia/política de despliegue además de la gestión de vulnerabilidades. Dicha política se basa en lo siguiente:

- **Entorno Dev:** Es la rama habitual de desarrollo, en el que equipo subirá sus aportes en el día a día. Cada uno de los aportes pasará por el conjunto de validación y en caso de que no lo cumpla se notificará por correo electrónico con el fin de que se le dé solución al problema lo antes posible.
    - Existe una rama dev específica para el modelo de aprendizaje profundo y otro para el código.
- **Entorno pre-production**: Una vez se tiene los suficientes aportes como para poder desplegar una nueva versión a un porcentaje concreto de los usuarios, se decide mover todo el código disponible a esta rama (mediante un merge request) y se despliega un pequeño número de equipos con esta versión de demostración con el fin de que los usuarios lo prueben y proporcione feedback al equipo de desarrollo.
- **Entorno producción:** Tras recibir el suficiente feedback (positivo o negativo) por parte de los usuarios, en caso de que sea positivo se decide desplegar esa versión, la cual ya se encuentra testeada frente a posibles vulnerabilidades, en el despliegue principal y poner a disposición de la totalidad de los clientes esa versión concreta.
- En caso de que el feedback sea negativo (alguna funcionalidad que no funcione, aunque no debería ya que sino no pasaría el conjunto de validación de la rama dev) se procede a retirar el despliegue de demo inmediatamente, solucionar los posibles problemas que pueda haber y volver a desplegarlo con el fin de obtener feedback una vez más.

A la hora de gestionar/controlar las diferentes versiones del software el mismo pipeline o proceso de validación es la encargada de etiquetar el código de la siguiente forma:

- En cuanto se hace un aporte de código al repositorio este se etiqueta con TEST ya que aún no sabemos si este código que se acaba de subir rompe por completo o no la aplicación.
    - Análisis de código estático y tests.
- Una vez verificado que ese aporte pasa el conjunto de validación, se etiqueta como STAGING para poder desplegar una demostración a parte de los usuarios. En caso de que no los pase simplemente el proceso acaba ahí y ese aporte no avanza de fase.
    - El conjunto de validación incluye tests y análisis de código estático.
    - El fuzzing se ejecuta en la versión de STAGING también.
- Tras verificar que el feedback de los usuarios es positivo se etiqueta ese aporte como PROD, que está listo para ser desplegado en el despliegue principal y termina el proceso.
- En el supuesto de que el feedback no sea el adecuado, el proceso acaba ahí, se elimina el despliegue que ofrece una demo a los usuarios y se procede a intentar solucionar lo que les descontenta.

Como Itaap hace uso de software de terceros (APIs de las redes sociales), resulta interesante indicar que este deberá ser tratado como si fuera un software propio, es decir, en cada aporte de código se verifica el funcionamiento de estas APIs en escenarios controlados.

En estos escenarios tenemos controlada tanto la entrada como la salida de estas APIs, en caso de que la salida no sea la esperada el aporte se marca como no funcional y se notifica al equipo de desarrollo con el fin de que den con la solución del problema.

Además, la información proporcionada por estas APIs es validada por el sistema de ingesta de datos. Esta validación consiste en verificar si los datos proporcionados corresponden a un formato concreto y si estos contienen los campos necesarios para el procesamiento del dato.