1. **PROCESO:** Gestión de Tecnologías de la Información

2. **SUB PROCESO:** NA.

3. **OBJETIVO:** Establecer lineamientos para el desarrollo de software seguro con el fin implementar en la entidad aplicaciones seguras.

4. **CONTENIDO:**

4.1 Principios de desarrollo seguro

- ✓ "Partir siempre de un modelo de permisos mínimos, es mejor ir escalando privilegios por demanda de acuerdo a los perfiles establecidos en las etapas de diseño.
- ✓ Si se utiliza un lenguaje que no sea compilado, asegurarse de limpiar el código que se pone en producción, para que no contenga rutinas de pruebas, comentarios o cualquier tipo de mecanismo que pueda dar lugar a un acceso indebido.
- ✓ Nunca confiar en los datos que ingresan a la aplicación, todo debe ser verificado para garantizar que lo que está ingresando a los sistemas es lo esperado y además evitar inyecciones de código.
- ✓ Hacer un seguimiento de las tecnologías utilizadas para el desarrollo. Estas van evolucionando y cualquier mejora que se haga puede dejar obsoleta o inseguras versiones anteriores.
- ✓ Todos los accesos que se hagan a los sistemas deben ser validados.
- ✓ Para intercambiar información sensible utilizar protocolos para cifrar las comunicaciones, y en el caso de almacenamiento la información confidencial debería estar cifrada utilizando algoritmos fuertes y claves robustas.
- ✓ Cualquier funcionalidad, campo, botón o menú nuevo debe agregarse de acuerdo a los requerimientos de diseño. De esta forma se evita tener porciones de código que resultan siendo innecesarias.
- ✓ La información almacenada en dispositivos móviles debería ser la mínima, y más si se trata de contraseñas o datos de sesión. Este tipo de dispositivos son los más propensos a ser que se pierdan y por lo tanto su información puede ser expuestas más fácilmente.
- ✓ Cualquier cambio que se haga debería quedar documentado, esto facilitará modificaciones futuras.
- ✓ Poner más cuidado en los puntos más vulnerables, no hay que olvidar que el nivel máximo de seguridad viene dado por el punto más débil."[1]

4.2 Consideraciones para el desarrollo seguro

- ✓ Manejo de Entradas**;** Nunca confíe en las entradas (Buenas prácticas en el desarrollo de software)

  - ➢ Una de las medidas más importante de defensa que los desarrolladores pueden tomar es validar las entradas que recibe su software. Esto no es sólo responsabilidad de los desarrollos, también comprende la validación de metodologías y procedimientos de desarrollo.
  - ➢ Revisiones de entradas no seguras. Las entradas son la mayor causa de algunas de las vulnerabilidades más peligrosas (incluyendo desbordamiento de Buffer, inyección SQL, entre otras).
  - ➢ Si una aplicación consta de más de un proceso, valide las entradas para cada proceso incluso si la entrada es proporcionada por otra parte de la aplicación. Valide la entrada incluso si esta es

[1] https://www.welivesecurity.com/la-es/2014/02/28/10-principios-basicos-para-desarrollo-seguro/

entregada sobre una conexión segura, si llega de una fuente confiable o si está protegida por permisos estrictos de archivo.

- Revisar las variables de entrada (incluyendo variables de entorno, valores de registro, servicios de red y nombres de ruta), teniendo en cuenta que pueden ser vulnerables a ataques y alterar su contenido.

✓ **Que se debe validar.** Es importante tener en cuenta los sigguientes aspectos para llevar a cbabo una validación del desarrollo:

- Datos Incompletos o No validos: No repare los datos de entrada que fallen las validaciones de entrada, sólo rechácelos.
- Longitud de las entradas: Siempre realice la revisión contra un mínimo y un máximo de longitud esperada.
- Comprobación del límite de entradas numéricas: Siempre verifique las entradas numéricas contra valores máximos y mínimos. Sin una comprobación de límite para entradas numéricas, los atacantes pueden crear un desbordamiento de enteros.
- Filtrado de Meta-caracteres: Verifique que no existan caracteres especiales como por ejemplo una comilla simple (‘) que es un carácter en solicitudes SQL o periodos dobles (slash o backslash) ya que pueden ser utilizados para acceder a rutas de sistemas de archivos.

✓ **Control de Mensajes de Error**

- No revele información sensible en las respuestas de error, incluyendo detalles del sistema, identificadores de sesión o información de la cuenta.
- Utilice errores controlados que no muestran la depuración o el seguimiento de la pila de la información.
- Implemente mensajes de errores genéricos y utilice páginas de error personalizadas.
- La aplicación debe manejar errores controlados de aplicación y no debe revelar configuración del servidor.
- La lógica de errores identificados, implica denegar el acceso por defecto.

✓ **Seguridad a nivel de aplicación, Sistema operativo y base de datos**

Las acciones e impactos que se presentan en el siguiente cuadro son una guía clave de los lineamientos para el desarrollo de software seguro.

SEGURIDAD A NIVEL DE APLICACIÓN	
ACCIONES	**IMPACTO**
SQL INJECTION O COMMAND INJECTION	Es la habilidad de modificar la sentencia SQL del aplicativo para ejecutar código SQL arbitrario sobre el motor de base de datos. Se presenta cuando no existe una validación de entradas de usuario. Podría afectar la confidencialidad e integridad de los datos almacenados en una base de datos que una aplicación web tenga acceso.
CROSS SITE SCRIPTING XSS	Es la habilidad de ejecutar código script en el servidor de forma arbitraria. Ocurre cuando no se validan entradas en el aplicativo. Se usa para robar sesiones web, cookies, robar archivos y hasta producir phishing. Podría afectar la confidencialidad de los usuarios

	a la aplicación WEB, así como la imagen de la entidad.
BUFFER OVERFLOWS O BUFFER OVERRUNS	Es el ataque más común sobre las aplicaciones. Se presenta cuando no se valida correctamente el uso de memoria: se excede el tamaño de un arreglo o se excede el valor de una variable.
USO DE SESIONES	podría comprometer la confidencialidad e integridad al permitir que pueda ser posible modificar la actividad de los usuarios en la aplicación WEB
USO DE COOKIES	podría comprometer la confidencialidad e integridad al permitir que pueda ser posible modificar la actividad de los usuarios en la aplicación WEB
REVISIÓN DE CÓDIGO FUENTE POR ELEMENTOS MALICIOSOS	El uso de código proveniente de internet para las aplicaciones web de la entidad, podrían tener contenido malicioso.
PREVENCIÓN DE INYECCIÓN DE ARCHIVOS MALICIOSOS	Se podría comprometer la confidencialidad, integridad o disponibilidad de la aplicación WEB
ESTABLECER CADUCIDAD DE APLICACIONES WEB NO UTILIZADAS	Al desconocer que las páginas web se tienen almacenadas en el Servidor Web, podría permitir que pagines antiguas expongan con problemas con la integridad, disponibilidad o confidencialidad.
REVISIÓN POR MENSAJES DE ERROR	Algunos mensajes de error permiten conocer la versión del servidor web, así como posibles falencias que podrían poner en riesgo la seguridad de la aplicación
REVISAR PÁGINAS QUE CONTENGAN CAJAS DE TEXTO PARA EVALUAR SU VULNERABILIDAD A HERRAMIENTAS AUTOMATIZADAS(SPAMBOTS)	El permitir que herramientas de ejecución automatizada sobre cajas de texto, podría afectar el funcionamiento de la aplicación, de tal forma que los datos reales que un usuario podría ingresar son sustituidos por datos falsos o erróneos.
LINEAMIENTOS DE CONTRASEÑAS EN CUENTAS DE USUARIO EN BASES DE DATOS	El permitir contraseñas sencillas o por defecto comprometer la confidencialidad, integridad o disponibilidad
REVISIÓN POR CONFIGURACIÓN POR DEFECTO	La configuración por defecto en plataforma, podría permitir que cualquier usuario, pudiera tener acceso no autorizado al servidor y con ello afectar la confidencialidad, integridad o disponibilidad de las aplicaciones web que mantiene el mismo.
REVISIÓN POR ARCHIVOS POTENCIALMENTE SENSIBLES (ZIP, RAR, CONFIG, CFG, ETC.)	El dejar archivos zip, rar, etc. Podría permitir a un atacante obtener toda la información con respecto al funcionamiento de la aplicación WEB
USO DE TLS	El no usar TLS en servidores Web, expone a que los datos que fluyen entre el servidor y el cliente, puedan ser capturados en la red.
REVISIÓN DE DIRECTORIOS WEB QUE SE VISUALIZAN COMO DIRECTORIOS	Al poder ver los directorios web es posible conocer el contenido del sitio WEB
ESTABLECER UNA ADECUADO CONTROL DE AUTORIZACIÓN	El no tener un adecuado modelo de autorización sobre los directorios de un servidor Linux, puede exponer la información sensible de las diferentes aplicaciones a usuarios no autorizados.
REVISAR MECANISMO DE NO REPUDIACIÓN	La ejecución de scripts, aplicaciones sobre bases de datos en los servidores Linux, y estos no dejen evidencia de datos como Fecha, Hora, Usuario, Actividad que realizo, sobre la base de datos,

	nulifica la posibilidad de establecer la responsabilidad de acciones que van en detrimento de los datos almacenados en los mismos.
SEGURIDAD A NIVEL DE SISTEMA OPERATIVO	
MÓDULO MODEVASIVE PARA APACHE	Usado para contener ataques Distribuidos Denegación de Servicios (DDoS) sobre el servidor web.
MÓDULO MODSECURITY PARA APACHE	Usado como firewall de aplicaciones web, ayuda a ataques de inyección de código, intrusiones no autorizadas sobre el aplicativo y sistema operativo. Permite filtrado de peticiones, técnicas evasivas, filtrado HTTPS, logs de auditoría entre otras funcionalidades.
SISTEMA OPERATIVO ACTUALIZADO	Si el sistema operativo usado es Oracle Linux debe ser actualizado a la última versión estable de la línea. Con parches de seguridad y vinculado al sistema de ULN que provee Oracle para actualizaciones.
FIREWALL DE SISTEMA OPERATIVO	Se cuenta con un firewall de sistema operativo que permite entradas a puertos no habilitados de la máquina, con el fin de adquirir información y acceso no autorizado a la misma.
INSTALACIÓN DE MALDET	MalDet (Malware Detection), es un aplicativo que se instala sobre el sistema operativo y se encarga de validar la presencia de software malicioso para luego eliminarlos o ponerlos como cuarentena para ser analizados por el administrador posteriormente.
SEGURIDAD A NIVEL DE BASE DE DATOS	
PROPIETARIO DE OBJETOS. ECENSO	✓ No deben existir conexiones directas a este usuario. ✓ La contraseña no se entrega. ✓ Cambios estructurales formalizando un control de cambio en producción ✓ Cambio de contraseña obligatoria cada 3 meses. ✓ Se bloquea el usuario después de 3 intentos fallidos de conexión y dura un día bloqueado el usuario. ✓ Auditoría de bd prendida sobe “CREATE SESSION” Auditoría de table, procedure, view.
USUARIO DE CONEXIÓN DE LA APLICACIÓN	✓ Se le otorgan privilegios CRUD sobre todas las tablas y objetos de CNP_WEB_INSCRICPCION. ✓ La contraseña nunca caduca. ✓ Tiempo de inactividad: 5 minutos. ✓ Se bloquea al usuario después de 10 intentos fallidos de conexión y dura un día bloqueado el usuario. ✓ Responsables contraseña: Administradores Capa media (Aplicación PHP).
USUARIO DE CONEXIÓN DE APLICACIÓN DE MONITOREO DE ECENSO	✓ Se otorgan privilegios solo de SELECT sobre tablas de CNP_WEB_INSCRIPCION. ✓ La contraseña nunca caduca. ✓ Tiempo de inactividad: 5 minutos. ✓ Se bloquea al usuario después de 10 intentos fallidos de conexión y dura un día bloqueado el usuario. Responsables contraseña: Administradores Aplicación Monitoreo
	✓ Privilegios de CRUD sobre CNP_WEB_INSCRIPCION Cambio de contraseña obligatoria cada 3 meses Se bloquea el usuario después de 3 intentos fallidos de conexión

USUARIO PARA INGENIEROS QUE BRINDAN SOPORTE AL APLICATIVO	y dura un día bloqueado el usuario. ✓ Auditoría de INSERT, UPDATE, DELETE, sobre las tablas: CNP_ADMIN_CONTROL, CNP_ADMIN_USUARIOS, CNP_ADMIN_USUARIOS_ADM ✓ Auditoría de sesión.

Tabla 1. Lineamientos para desarrollo seguro
Fuente: Adaptado de OWASP Top 10-2017

- ✓ Los ambientes de desarrollo, pruebas y producción se deben encontrar claramente definida, independiente y controlados. Se sugiere la no existencia de ambiente local del desarrollador (Maquinas de usuario). Para los ambientes de prueba y desarrollo, no deben contener información real vigente o copiada de los sistemas de producción.

4.3 Otros Lineamientos a Tener en Cuenta

Estos lineamientos se aplican a todos los funcionarios y contratistas, que intervienen en los procesos de planeación, liderazgo y desarrollo de proyectos de desarrollo de software.

- ✓ El acceso al código fuente, debe ser restringido exclusivamente a los Desarrolladores.
- ✓ Se recomienda incorporar buenas prácticas de desarrollo seguro, como por ejemplo el estándar OWASP (*Open Web Application Security Project)* dentro de los requisitos de seguridad.
- ✓ Exigir pruebas técnicas de vulnerabilidad, para autorizar su salida a producción.
- ✓ Los desarrolladores revisarán y determinarán la acción a seguir para el tratamiento de las vulnerabilidades, para evitar que tengan brechas de seguridad.
- ✓ Se debe hacer implementación de controles para el manejo de versiones del código fuente en ambientes de producción, calidad y desarrollo.

4.3.1 Consideraciones para el desarrollo de Aplicaciones Web Seguras

El siguiente resumen es una recopilación de los riesgos o fallas de seguridad más comunes que afectan las aplicaciones. Tomado de OWASP TOP 10.[2]

[2] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

A1:2017 Inyección	Las fallas de inyección, como SQL, NoSQL, OS o LDAP ocurren cuando se envían datos no confiables a un intérprete, como parte de un comando o consulta. Los datos dañinos del atacante pueden engañar al intérprete para que ejecute comandos involuntarios o acceda a los datos sin la debida autorización.
A2:2017 Pérdida de Autenticación	Las funciones de la aplicación relacionadas a autenticación y gestión de sesiones son implementadas incorrectamente, permitiendo a los atacantes comprometer usuarios y contraseñas, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios (temporal o permanentemente).
A3:201 Exposición de datos sensibles	Muchas aplicaciones web y APIs no protegen adecuadamente datos sensibles, tales como información financiera, de salud o Información Personalmente Identificable (PII). Los atacantes pueden robar o modificar estos datos protegidos inadecuadamente para llevar a cabo fraudes con tarjetas de crédito, robos de identidad u otros delitos. Los datos sensibles requieren métodos de protección adicionales, como el cifrado en almacenamiento y tránsito.
A4:2017 Entidades Externas XML (XXE)	Muchos procesadores XML antiguos o mal configurados evalúan referencias a entidades externas en documentos XML. Las entidades externas pueden utilizarse para revelar archivos internos mediante la URI o archivos internos en servidores no actualizados, escanear puertos de la LAN, ejecutar código de forma remota y realizar ataques de denegación de servicio (DoS).
A5:2017 Pérdida de Control de Acceso	Las restricciones sobre lo que los usuarios autenticados pueden hacer no se aplican correctamente. Los atacantes pueden explotar estos defectos para acceder, de forma no autorizada, a funcionalidades y/o datos, cuentas de otros usuarios, ver archivos sensibles, modificar datos, cambiar derechos de acceso y permisos, etc.
A6:2017 Configuración de Seguridad Incorrecta	La configuración de seguridad incorrecta es un problema muy común y se debe en parte a establecer la configuración de forma manual, *ad hoc* o por omisión *(*o directamente por la falta de configuración). Son ejemplos: *S3 buckets* abiertos, cabeceras HTTP mal configuradas, mensajes de error con contenido sensible, falta de parches y actualizaciones, *frameworks*, dependencias y componentes desactualizados, etc.
A7:2017 Secuencia de Comandos en Sitios Cruzados (XSS)	Los XSS ocurren cuando una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada; o actualiza una página web existente con datos suministrados por el usuario utilizando una API que ejecuta *JavaScript* en el navegador. Permiten ejecutar comandos en el navegador de la víctima y el atacante puede secuestrar una sesión, modificar (*defacement*) los sitios web, o redireccionar al usuario hacia un sitio malicioso.
A8:2017 Deserialización Insegura	Estos defectos ocurren cuando una aplicación recibe objetos serializados dañinos y estos objetos pueden ser manipulados o borrados por el atacante para realizar ataques de repetición, inyecciones o elevar sus privilegios de ejecución. En el peor de los casos, la deserialización insegura puede conducir a la ejecución remota de código en el servidor.
A9:2017 Componentes con vulnerabilidades conocidas	Los componentes como bibliotecas, *frameworks* y otros módulos se ejecutan con los mismos privilegios que la aplicación. Si se explota un componente vulnerable, el ataque puede provocar una pérdida de datos o tomar el control del servidor. Las aplicaciones y API que utilizan componentes con vulnerabilidades conocidas pueden debilitar las defensas de las aplicaciones y permitir diversos ataques e impactos.
A10:2017 Registro y Monitoreo Insuficientes	El registro y monitoreo insuficiente, junto a la falta de respuesta ante incidentes permiten a los atacantes mantener el ataque en el tiempo, pivotear a otros sistemas y manipular, extraer o destruir datos. Los estudios muestran que el tiempo de detección de una brecha de seguridad es mayor a 200 días, siendo típicamente detectado por terceros en lugar de por procesos internos

Imagen 1. Riegos en seguridad de aplicaciones
Fuente: OWASP Top 10-2017

RESUMEN DE CAMBIOS

Versión	Fecha	Numerales	Descripción de la modificación
1.0		Todos	Se crea el documento.

RESPONSABILIDAD Y AUTORIDAD		
Elaboró / Actualizó:	**Revisó:**	**Aprobó:**
Firma: Original Firmado	Firma: Original Firmado	Firma: Original Firmado
Nombre: Fernando Arturo Vargas Herrera	Nombre: Oscar Javier Suárez Ramos	Nombre: Oscar Javier Suárez Ramos
Cargo: Técnico Operativo	Cargo: Jefe Oficina de Tecnología	Cargo: Jefe Oficina de Tecnología
Dependencia: Oficina de Tecnología	Dependencia: Oficina de Tecnología	Dependencia: Oficina de Tecnología