
	BPECU-SEG-032 STD CODIFICACIÓN SEGURA	BPECU-SEG-032
---	---------------------------------------	---------------

 BANCO PICHINCHA <small>En confianza.</small>	Política relacionada:	Código: BPECU-SEG-032
	Torre Tecnológica: Seguridad	Versión: 2.1
	Aprobador del Estándar: SEGURIDAD – Diego Vásquez	Fecha de creación: 23/08/2021
	Custodio: Diseño & Innovación	Fecha de liberación: 26/09/2024

ESTÁNDAR PARA CODIFICACIÓN SEGURA DE APLICACIONES

La información descrita en el presente documento es de uso reservado y exclusivo del BANCO PICHINCHA C.A. Está prohibida su reproducción sin previa autorización o su utilización en otros fines distintos para el cual fue entregada.

Clasificación del documento: INTERNO

Página 1 de 20

CONTROL DE CAMBIOS

VERSIÓN	DETALLE	ELABORADO POR	REVISADO POR	FECHA DE APROBACIÓN VERSIÓN
1.0	Creación del Estándar	Aarón Echeverría [Diseño & Innovación] Jorge Sánchez [Diseño & Innovación] Christopher Ortiz [Diseño & Innovación] Steven Coronado [Diseño & Innovación]	Abraham Rodríguez [IT Security Expert BP] Fredy Morantes [IT Security Expert BP]	02/06/2022
1.1	Modificación del Estándar 6.1 Subsección 2 se detalla que se tratan de servidores de BP 6.1 Subsección 8 se cambia JWT por Access Tokens Se añade 6.1 Subsección 9 Tiempo de expiración Token JWT 6.2 Subsección 3 se añade versión TLS 6.3 subsección 11 se cambia dom por Document Object Model 6.7 Subsección 2 Se especifica que el captcha se valida en el lado servidor 6.7 Subsección 4: Se especifica que la validación de las contraseñas se realiza en el front y en el back 6.8 Subsección 7: Se especifica que se lo realiza a nivel infraestructura 6.10 Subsección 5: Refraseo de mensajes de error	Jorge Sánchez [Diseño & Innovación]	Omar Romero [IT Security Expert BP]	07/07/2023

La información descrita en el presente documento es de uso reservado y exclusivo del BANCO PICHINCHA C.A. Está prohibida su reproducción sin previa autorización o su utilización en otros fines distintos para el cual fue entregada.

	<p>6.11 se añade la nota Aplica para el perfil de bdd o usuario con el rol específico</p> <p>Se agregan nuevos términos en Glosario</p>			
2.0	Reestructuración	<p>Jorge Sánchez [Ciberseguridad TCS]</p> <p>Sebastian Robles [Arquitecto de Software]</p>	<p>Marcelo Burgos [Arquitecto Ciberseguridad Sr]</p> <p>Nelson Pillajo [Experto Seguridad Aplicativa]</p> <p>Bernardo Medina [Experto Procesos]</p> <p>Jorge Villafuerte [Líder Arquitectura de Software]</p>	22/05/2024
2.1	<p>Se actualiza el Anexo: Estándar de Seguridad de Datos</p> <p>Se eliminan los Anexos: Resolución SB-2021- 2263 Norma de control Superintendencia de Bancos</p> <p>Instructivo de Certificación de Seguridad en Aplicaciones</p>	<p>Jorge Sánchez [Ciberseguridad TCS]</p>	<p>Nelson Pillajo [Experto Seguridad Aplicativa]</p>	26/09/2024

LISTA DE DISTRIBUCIÓN

EMPRESA	ÁREA	RESPONSABLE
BP	Arquitectura de Ciberseguridad	Diego Vásconez
BP	Seguridad Aplicativa	Robert Chávez / Nelson Pillajo
BP	Arquitectura de Desarrollo	Jorge Villafuerte
TCS	Ciberseguridad	Héctor Vásquez
TCS	Seguridad Aplicativa	Fernando Pozo

La información descrita en el presente documento es de uso reservado y exclusivo del BANCO PICHINCHA C.A. Está prohibida su reproducción sin previa autorización o su utilización en otros fines distintos para el cual fue entregada.



ÍNDICE

1. OBJETIVO	6
2. ALCANCE	6
3. REFERENCIAS.....	6
4. RESPONSABLES - PARTICIPANTES.....	6
5. ESTÁNDAR.....	7
5.1. Control de Accesos.....	7
5.1.1 Referencia.....	7
5.1.2 Práctica.....	7
5.2. Falla Criptográfica	9
5.2.1 Referencia.....	9
5.2.2 Práctica.....	9
5.3. Inyección	10
5.3.1 Referencia Normativa	10
5.3.2 Práctica.....	11
5.4. Mala Configuración de Seguridad	12
5.4.1 Referencia Normativa	12
5.4.2 Práctica.....	12
5.5. Componentes Vulnerables y Deprecados	13
5.5.1 Referencia Normativa	13
5.5.2 Práctica.....	13
5.6. Fallas de Autenticación e Identificación	14
5.6.1 Referencia Normativa	14
5.6.2 Práctica.....	14

5.7. Fallas de Integridad en Software y Datos	15
5.7.1 Referencia Normativa	15
5.7.2 Práctica	15
5.8. Insuficiente Monitoreo o Registro de Logs	16
5.8.1 Referencia Normativa	16
5.8.2 Práctica	16
5.9. Server Side Request Forgery	17
5.9.1 Referencia Normativa	17
5.9.2 Práctica	17
6. MEDICIÓN	18
7. ANEXOS.....	18
8. GLOSARIO Y DEFINICIONES.....	19



1. OBJETIVO

Desarrollar aplicaciones y/o componentes tecnológicos que cumplan las buenas prácticas de Desarrollo Seguro alineadas a los requerimientos de Banco Pichincha.

2. ALCANCE

Este documento norma las actividades de desarrollo seguro que deben aplicarse en desarrollos de código que formen parte de iniciativas y/o proyectos nuevos y/o existentes de Banco Pichincha. El mismo es de cumplimiento obligatorio para todos los equipos de banco, proveedores o cualquier tercero involucrado en la construcción de software.

3. REFERENCIAS

- Common Weakness Enumeration (CWE Top 25)
- NIST SP 800-218
- NIST 1800 – 3B
- NIST SP 800 – 57
- NIST 800 – 95
- NIST 800 – 53
- NIST 800 – 123
- NIST 800 - 63B
- OWASP Cheatsheets series
- OWASP Top 10 2021
- Secure Software Development Framework (SSDF)
- Safecode
- The Software Alliance (BSA)
- MITRE ATT&CK
- OWASP Top Ten Proactive Controls 2018

4. RESPONSABLES - PARTICIPANTES

- Ciberseguridad – TCS
- Mantenimiento – TCS
- Certificación de Seguridad – TCS
- Arquitectura de Desarrollo – Banco Pichincha
- Arquitectura de Ciberseguridad – Banco Pichincha
- Seguridad Aplicativa – Banco Pichincha

5. ESTÁNDAR

5.1. Control de Accesos

5.1.1 Referencia

ITEM	OWASP Cheat Sheet	OWASP TOP 10 2021	CWE	NIST 1800-3B	Safe Code
5.1.2.1	Autorización	A01	284	5.4	NA
5.1.2.2	Autorización	A01	NA	NA	NA
5.1.2.3	Autorización	A01	285	5.4	NA
5.1.2.4	Autorización	A01	22	NA	NA
5.1.2.5	Autorización	A01	285	5.4	NA
5.1.2.6	Autorización	A01	284	5.3.1	NA
5.1.2.7	Autorización	A01	284	5.3.1	NA
5.1.2.8	Autorización	A01	284	5.3.1	NA
5.1.2.9	Autorización	A01	284	5.3.1	NA
5.1.2.10	Autorización	A01	22, 284, 285	5.4	NA

5.1.2 Práctica

Los fallos en el control de accesos suelen provocar divulgación de información, modificación no autorizada o destrucción de los datos o la ejecución de actividades no alineadas a las funciones del personal. Este control está orientado a microservicios que están expuestos a internet y son consumidos desde un cliente sin una API, en el caso de servicios que solo son accesibles mediante una API (Azure API Management, Apigee, AWS API Gateway, etc) estos controles deben ser verificados a través de la API y no en el servicio directamente.

5.1.2.1 Los controles de acceso deben ser implementados en el lado del servidor de la aplicación o en la API, donde el atacante no puede modificar la comprobación del control de acceso. Esto se debe realizar mediante una capa de autorización y control de accesos a los diferentes endpoints del servicio que permita el acceso a los recursos si la autorización y control de acceso es válida, en caso contrario deben rechazarse las peticiones y accesos no autorizados (En el caso de que un servicio reciba un acceso no autorizado debe devolver un estado "401 Unauthorized y/o un mensaje de error" y finalizar inmediatamente la conexión con el cliente solicitante)

5.1.2.2 Implementar mecanismos de control de acceso únicos y reutilizarlos en toda la aplicación, incluyendo la correcta configuración del uso de Cross Origin Resource Sharing (CORS) lo cual evita el acceso a la API desde orígenes no autorizados/no fiables. Por defecto solo se debe permitir peticiones de *.pichincha.com en la configuración de CORS.



En el caso de que sea necesario permitir el consumo desde otros orígenes mediante CORS se debe revisar con Ciberseguridad BP para determinar su factibilidad

5.1.2.3 Los permisos del sistema deben respetar la propiedad de la información evitando que cualquier usuario pueda crear, leer, modificar o eliminar información que no le pertenece respetando el principio de menor privilegio. Esto se debe realizar mediante la generación de consultas en las capas de repositorios y modelos, que solo permitan a los usuarios autorizados crear, leer, modificar o eliminar información mediante el identificador único del usuario verificado y autorizado a través del token de acceso.

5.1.2.4 Para proteger la información confidencial del servidor web y/o aplicaciones, se deben implementar medidas que incluyen: desactivar la función de "listado de directorios", eliminar archivos con metadatos sensibles y evitar almacenar archivos de copia de seguridad en las raíces web. Además, se debe evitar exponer archivos que no son estrictamente necesarios para el funcionamiento de la aplicación en el servidor web y/o aplicaciones.

5.1.2.5 Se debe limitar el procesamiento de peticiones de los servicios y/o APIs para minimizar el daño de las herramientas de ataque automatizadas, esto se debe realizar mediante configuración de herramientas que se ejecuten antes del procesamiento del servicio y permitan filtrar las peticiones tales como: WAFs y Rate Limits. (Se debe aplicar la validación de Captcha en caso de que el servicio esté expuesto a internet sin autenticación)

5.1.2.6 Se debe implementar medidas que permitan el control de privilegios del lado del servidor, en caso de que los tokens de acceso contengan el rol del usuario estos deben ser verificados y validados antes de procesar la petición. Esto con el propósito de evitar la elevación de privilegios que permita a un atacante actuar como usuario sin haber iniciado sesión o actuar como administrador cuando se ha iniciado sesión como usuario estándar.

5.1.2.7 Se debe evitar la manipulación de metadatos de los tokens de acceso, cookie o campo oculto que puedan ser manipulados para elevar privilegios o abusar de la invalidación del token. Para esto, se deben implementar medidas del lado del servidor que permitan verificar la firma de los tokens de acceso y en caso de que no puedan ser verificados se debe rechazar la solicitud.

5.1.2.8 Se debe invalidar los identificadores de sesiones y/o access tokens en el servidor después del cierre de sesión. Adicionalmente, los identificadores de sesiones y/o access tokens deben ser eliminados del cliente o navegador. En caso de contar con autenticación mediante OAuth2 se debe hacer uso del endpoint revoke para invalidar el access token.

5.1.2.9 Los identificadores de sesión y/o tokens de acceso deben tener un tiempo máximo de vida (TTL) de 10 mins y no deben poseer datos sensibles (Ver sección 7. glosario y definiciones) en el payload del token de acceso para que la oportunidad de acceso de un atacante se vea reducida. En caso de requerir un tiempo de vida (TTL) mayor, se debe verificar con Ciberseguridad su factibilidad.

5.1.2.10 Evitar el acceso a páginas no permitidas y forzadas por parte del usuario mediante la obligación de autenticación para evitar el acceso a recursos no autorizados. En el caso de aplicaciones no monolitas (separadas backend del frontend) se debe contar con un proceso de redireccionamiento hacia el login o mostrar un mensaje de usuario no autenticado desde el frontend.

5.2. Falla Criptográfica

5.2.1 Referencia

ITEM	OWASP CheatSheet	OWASP TOP 10 2021	CWE	NIST SP 800-57	Safe Code
5.2.2.1	Protección de la privacidad del usuario	A02	312	NA	NA
5.2.2.2	Almacenamiento o criptográfico	A02	310, 311	5.2	NA
5.2.2.3	Almacenamiento o criptográfico	A02	310, 311	5.2	NA
5.2.2.4	Seguridad estricta de transporte HTTP	A02	319	Appendix A	NA
5.2.2.5	Seguridad estricta de transporte HTTP	A02	312	NA	NA
5.2.2.6	Almacenamiento o criptográfico	A02	310	NA	NA

5.2.2 Práctica

Se deben proteger los datos sensibles o personales en tránsito y reposo mediante encriptación y/o protocolos seguros, para evitar que los atacantes puedan robar o modificar datos para llevar a cabo fraudes con tarjetas de crédito, robos de identidad u otros delitos. La metodología de clasificación de los datos está a cargo de la Gerencia del Gobierno de Datos (Anexo 8) y se identifican qué datos son sensibles según las leyes de privacidad, los requisitos reglamentarios o las necesidades del negocio.

5.2.2.1 Evitar almacenar datos sensibles que no son necesarios para el modelo de negocio descartando lo antes posible esta información. En caso de que se requiera almacenar los datos sensibles, se debe hacer uso de tokenización (PCI DSS), enmascaramiento y/o encriptación.

5.2.2.2 Para mitigar la fuga de información sensible se debe aplicar cifrado de datos en reposo. Para ello se debe usar el Cifrado de Datos Transparente (TDE) el cual ofrece encriptación en tiempo real haciendo uso de algoritmos fuertes y estables en el servidor de base de datos.

5.2.2.3 No se permite el uso de algoritmos, protocolos criptográficos, esquemas de padding y/o funciones criptográficas deprecadas o débiles, mediante la verificación de

que estos son actuales y criptográficamente fuertes tales como AES - GSM 256, RSA 2048, SHA-256, SHA-384, Optimal Asymmetric Encryption Padding (OAEP) o Probabilistic Signature Scheme (PSS). En caso del tránsito de datos sensibles se debe hacer uso del Esquema de Encriptación de Datos en Tránsito de Banco Pichincha (Anexo 2).

5.2.2.4 Se debe evitar que los datos en tránsito viajen en texto plano por lo que deben ser cifrados con protocolos seguros como TLS V1.3 con cifrado de secreto perfecto (PFS), priorización de cifrado por el servidor, y parámetros seguros. Se debe hacer uso de directivas como HTTP Strict Transport Security (HSTS), Secure File Transfer Protocol (SFPT). (Nota: Bajo previa justificación, en caso de no poder usar TLS versión 1.3 se debe revisar con Ciberseguridad la factibilidad de usar mínimo versión 1.2)

5.2.2.5 Desactivar el almacenamiento en caché en la respuesta de la API que contenga datos sensibles mitigando posibles filtraciones de datos. (Nota: Esto no hace referencia a caché de browser, caché de base de datos o caché de procesamiento de las peticiones)

5.2.2.6 En el caso de requerir utilizar valores aleatorios se debe utilizar Universal Unique Identifier V4 para añadir una aleatoriedad criptográfica

5.3. Inyección

5.3.1 Referencia Normativa

ITEM	OWASP CheatSheet	OWASP TOP 10 2021	CWE	NIST 800 - 95	Safe Code
5.3.2.1	Prevención de inyección	A03	77	A.6	NA
5.3.2.2	Prevención de inyección	A03	75, 77, 917	A.6	NA
5.3.2.3	Prevención de inyección	A03	77	A.6	NA
5.3.2.4	Prevención de inyección	A03	77	A.6	NA
5.3.2.5	Prevención de inyección	A03	83	A.6	NA
5.3.2.6	Prevención de Cross Site Scripting	A03	79, 80	A.6.3	NA
5.3.2.7	Prevención de Cross Site Scripting	A03	79	A.6.3	NA
5.3.2.8	Prevención de Cross Site Scripting	A03	79	A.6.3	NA
5.3.2.9	Prevención de inyección	A03	98	A.6.3	NA



5.3.2 Práctica

Los tipos de inyección se generan al enviar datos no fiables a un intérprete como comando o consulta. (Esto aplica para todos los desarrollos front, back, base de datos y sus diferentes entornos dentro de Banco Pichincha)

5.3.2.1 Se debe validar los datos suministrados por el usuario, estos deben ser filtrados o saneados de lado del servidor y/o servicio, esto se debe realizar mediante filtrado y/o validación de caracteres para el tipo de dato según el modelo de negocio. (Por ejemplo, para un tipo de dato de cédula solo debe permitirse números)

5.3.2.2 Se debe verificar y transformar el formato de codificación a UTF8 antes de aplicar un proceso de filtrado y validación, adicionalmente no permitir la utilización de etiquetas HTML y/o saltos de línea (en el caso que no se requiera text areas) que puedan ser utilizados por un atacante.

5.3.2.3 Se debe utilizar consultas parametrizadas (prepared statement) y utilizar parámetros para pasar los valores a las consultas y/o llamadas para evitar que un atacante pueda inyectar código malicioso en servicios y/o microservicios.

5.3.2.4 Se debe utilizar Object-Relational Mapping (ORMs) actualizados y seguros que contengan una API de código parametrizada que nos permita crear prepared statements de manera flexible y confiable.

5.3.2.5 Se debe limitar el parámetro de tamaño de página (page size) en los endpoints donde se hace uso de paginación y adicionalmente hacer uso del comando LIMIT en las consultas a la base de datos. Tomar en consideración que este control debe ser implementado acorde a la funcionalidad del servicio.

5.3.2.6 Se debe hacer uso de APIs de código en el front end que sean seguras contra XSS y evitar el uso de APIs de código inseguras como innerHTML, outerHTML y/o setAttribute que permitan la modificación del Document Object Model (DOM) en el navegador sin previa validación y filtrado.

5.3.2.7 Se debe evitar una malformación de sintaxis de formato de respuesta de un servicio y/o microservicio, por ejemplo, en el caso de un servicio donde su respuesta es en formato XML se debe realizar un escapado (Escape) de los caracteres especiales y convertirlos a entidades XML, en el caso de un formato de respuesta JSON los caracteres especiales deben ser escapados en base al estándar ECMA (Ejemplo: \u0027, \)

5.3.2.8 Se deben implementar las cabeceras de políticas de seguridad de contenidos (CSP) como control de mitigación de defensa en profundidad contra el XSS. (En el caso de que sea necesario consumir desde otros orígenes diferentes a *.pichincha.com mediante CSP se debe revisar con Ciberseguridad BP para determinar su factibilidad)

5.3.2.9 Validar la carga de archivos mediante la verificación del contenido y el formato del archivo. Se debe mantener controles del tamaño máximo permitido para la subida de archivos a lo estrictamente necesario por el modelo de negocio. Evitar el almacenamiento permanente de archivos en el mismo servidor en donde se encuentra alojada la aplicación y/o el servicio mediante el uso de servidores de almacenamiento externos en donde no exista el permiso de execute del file system.

5.4. Mala Configuración de Seguridad

5.4.1 Referencia Normativa

ITEM	OWASP CheatSheet	OWASP TOP 10 2021	CWE	NIST 800 - 95	Safe Code
5.4.2.1	Content Security Policy	A05	829	NA	NA
5.4.2.2	HTTP Security Response Headers	A05	611	3.5.2	NA
5.4.2.3	HTTP Security Response Headers	A05	829, 942	NA	NA
5.4.2.4	Error Handling	A05	755	NA	NA
5.4.2.5	Abuse Case	A05	276	NA	NA

5.4.2 Práctica

La mala configuración de seguridad suele ser el resultado de configuraciones por defecto, configuraciones de seguridad deficientes en nube, cabeceras HTTP mal configuradas, y mensajes de error verbales que contienen información sensible.

5.4.2.1 Hacer uso del header Content Security Police (CSP) con los orígenes mínimos para el correcto funcionamiento de la aplicación. Se recomienda agregar la cabecera de CSP y evitar contener valores por defecto como "*" que puedan permitir el consumo de muchos orígenes de manera no controlada.

5.4.2.2 En caso de aplicaciones legacy en las cuales no pueda utilizarse Content Security Policy se debe utilizar la cabecera X-Frame-Options DENY, SAMEORIGIN, ALLOW-FROM URI para prevenir al sitio ser ubicado desde fuera del dominio y así mitigar ataques tipo Clickjacking.

5.4.2.3 Hacer uso del header Cross Origin Resource Sharing (CORS) con los orígenes mínimos para el correcto funcionamiento de la aplicación. Se recomienda agregar las cabeceras de CORS y evitar contener valores por defecto como "*" que puedan permitir el consumo de muchos orígenes de manera no controlada.

5.4.2.4 Controlar los errores producidos por la aplicación en tiempo de ejecución, de manera que los errores de sistema no sean devueltos al usuario, sino que estos sean mensajes de error genéricos, para contrarrestar una divulgación de información de la plataforma y/o infraestructura.

5.4.2.5 Evitar la salida de aplicaciones a producción con configuraciones por defecto, en donde estén habilitados usuarios o formularios de pruebas, páginas o servicios no utilizados, mismos que se convierten en vectores de ataque utilizados por el atacante.

5.5. Componentes Vulnerables y Deprecados

5.5.1 Referencia Normativa

ITEM	OWASP CheatSheet	OWASP TOP 10 2021	CWE	NIST 800 - 123	Safe Code
5.5.2.1	Parcheado virtual	A06	937, 1035, 1104	4.2.1	NA
5.5.2.2	Parcheado virtual	A06	937, 1035, 1104	4.1	NA
5.5.2.3	Parcheado virtual	A06	937, 1035, 1104	5.1	NA
5.5.2.4	Parcheado virtual	A06	937, 1035, 1104	5.1	NA
5.5.2.5	Parcheado virtual	A06	16937, 1035, 1104	5.1	NA

5.5.2 Práctica

Los componentes tecnológicos o librerías son parte fundamental del desarrollo del software, por lo cual, se debe hacer un análisis de vulnerabilidades de manera adecuada para mitigar riesgos que pueden ser explotados por un atacante.

5.5.2.1 Eliminar las dependencias no utilizadas al momento del despliegue a entornos productivos (Como por ejemplo: Herramienta de testing, dependencias de testing, de debugging, entre otras que no se necesitan para entornos productivos).

5.5.2.2 Evitar la utilización de versiones de librerías o componentes tecnológicos que contengan vulnerabilidades reportadas en repositorios públicos (CVE)

5.5.2.3 Para evitar la inclusión manual y/o copia de código de librerías o componentes tecnológicos. Se debe usar manejadores de paquetes (Maven, Gradle, NPM, Nuget, etc.) para reducir la posibilidad de incluir un componente modificado y/o malicioso.

5.5.2.4 Evitar la inclusión de librerías o componentes tecnológicos desde Repositorios o Artifacts que no forman parte de Banco Pichincha para reducir la posibilidad de incluir un componente modificado y/o malicioso.

5.5.2.5 No utilizar librerías o componentes que se encuentren deprecadas para evitar la explotación de vulnerabilidades reportadas en repositorios públicos (CVE). (En caso de requerir la utilización de librerías deprecadas se debe revisar con Ciberseguridad BP para determinar su factibilidad).

5.6. Fallas de Autenticación e Identificación

5.6.1 Referencia Normativa

ITEM	OWASP CheatSheet	OWASP TOP 10 2021	CWE	NIST 800 – 63B	Safe Code
5.6.2.1	Autenticación	A07	255	NA	NA
5.6.2.2	Autenticación	A07	1390	5.2.2	NA
5.6.2.3	Autenticación	A07	613	7.1.1	NA
5.6.2.4	Gestión de sesiones	A07	287, 613	7.1.2	NA
5.6.2.5	Gestión de sesiones	A07	200, 384	7.1	NA
5.6.2.6	Gestión de sesiones	A07	287, 613	10.1	NA

5.6.2 Práctica

Una incorrecta implementación de autenticación y gestión de sesiones podría permitir a los atacantes comprometer contraseñas, claves, así como explotar otros defectos de implementación para suplantar identidades.

5.6.2.1 No autocompletar las credenciales y ofuscar los campos de texto de tipo password de las aplicaciones. Esto se debe realizar mediante el uso de campos de texto de tipo password y el uso del atributo autocomplete, sólo se debe mostrar las credenciales mediante la autorización del usuario.

5.6.2.2 En el caso de contar con una funcionalidad que requiere del consumo de un API o servicio sin autenticación y que fue pensada en el modelo de negocio para ser manejado por usuarios finales, se debe implementar una validación de Captcha (Lado de servidor) que permita verificar que el usuario es legítimo mitigando ataques automatizados.

5.6.2.3 Hacer uso de estándares y protocolos de autenticación seguros y robustos como OAuth 2.0, SAML 2. En el caso de necesitar hacer uso de un protocolo distinto se debe validar con Ciberseguridad su factibilidad.

5.6.2.4 Los identificadores de sesión y/o access tokens deben almacenarse de forma segura a través del Session Storage del navegador, que permitan ser eliminados automáticamente al momento de cerrar el navegador sin un cierre de sesión.

5.6.2.5 Evitar que los identificadores de sesión viajen a través de URLs, pues son susceptibles de ser leídos por un atacante.

5.6.2.6 Implementar mecanismos de cierre de sesión que actúen al detectar 2 minutos de inactividad por parte de un usuario.

5.7. Fallas de Integridad en Software y Datos

5.7.1 Referencia Normativa

ITEM	OWASP CheatSheet	OWASP TOP 10 2021	CWE	NIST 800 - 123	Safe Code
5.7.2.1	Seguridad CI / CD	A08	345, 353	2.2	NA
5.7.2.2	Seguridad CI / CD	A08	345, 353, 494	NA	NA
5.7.2.3	Deserialización	A08	502	NA	NA
5.7.2.4	Abuse Case	A08	915	NA	NA

5.7.2 Práctica

Es importante la verificación de la integridad de los datos, para no permitir que un atacante pueda alterar las peticiones y modificar la correcta ejecución del servicio y/o microservicio creando la posibilidad de ataques de repetición, inyección y escalada de privilegios.

5.7.2.1 Evitar que los recursos externos (Archivos Javascript y CSS) utilizados en una aplicación hayan sido alterados mediante checksums de integridad, se debe usar el atributo integrity de tipo Sha-384 para evitar los ataques de inyección de código malicioso.

5.7.2.2 Se debe validar la vigencia y autoridad de los certificados LTS y HTTPS de los recursos externos (fuera de la red privada de la aplicación) consumidos en entornos productivos.

5.7.2.3 Toda deserialización que se realice del lado del servidor debe ser realizada sobre una clase que contenga sus atributos o miembros definidos (cédula tipo number, nombre tipo string, etc). En caso de que exista un error o una incompatibilidad de los objetos deserializados y los tipos de datos de las clases se debe generar un error y detener el procesamiento de la petición.

5.7.2.4 Verificar mediante el pull request (peer review) que no se haya insertado código malicioso o no esperado en el repositorio.

5.8. Insuficiente Monitoreo o Registro de Logs

5.8.1 Referencia Normativa

ITEM	OWASP CheatSheet	OWASP TOP 10 2021	CWE	NIST 800 - 123	Safe Code
5.8.2.1	Logging	A09	223, 778	6.1	NA
5.8.2.2	Logging	A09	223, 778	6.1	NA
5.8.2.3	Logging	A09	223, 778	6.1	NA
5.8.2.4	Logging	A09	223, 778	6.1	NA

5.8.2 Práctica

La gestión de logs es importante en el desarrollo de servicios y/o microservicios, ya que permite tener trazabilidad de las acciones que se realizan por los usuarios y/o clientes para detectar actividades sospechosas y prevenir posibles ataques.

5.8.2.1 Se debe hacer uso de logs transaccionales que permitan registrar todos los datos de entrada y de salida del lado del servidor.

5.8.2.2 Se debe hacer uso de logs de auditoría y registrar todos los eventos relacionados al modelo de negocio tanto en los casos de errores y/o casos de ejecución satisfactoria de una acción por parte de cliente.

5.8.2.3 Los registros de logs deben generarse en un formato legible alineado a las políticas de Banco Pichincha (Anexo 1), que puedan consumirse fácilmente y gestionarse por área responsable de la gestión de logs.

5.8.2.4 Para un correcto registro de logs, se deberá tomar como usuario que realiza la acción al usuario en sesión y no sacar al mismo de un parámetro, ya que este último es fácilmente manipulable por un atacante, pudiendo realizar acciones no autorizadas y evitando que queden registros de dichas acciones.

5.9. Server Side Request Forgery

5.9.1 Referencia Normativa

ITEM	OWASP CheatSheet	OWASP TOP 10 2021	CWE	NIST 800 – 63B	Safe Code
5.9.2.1	Prevención de la falsificación de peticiones del lado del servidor	A10	918	8.4	NA
5.9.2.2	Prevención de la falsificación de peticiones del lado del servidor	A10	918	8.4	NA
5.9.2.3	Prevención de la falsificación de peticiones del lado del servidor	A10	918	8.4	NA

5.9.2 Práctica

Los fallos SSRF se producen cuando una aplicación web obtiene un recurso remoto sin validar la URL proporcionada por el usuario.

5.9.2.1 No utilizar inputs de URLs de recursos externos para ser consumidos desde el lado del servidor. La lógica del consumo de recursos externos solo debe ser gestionada del lado del servidor. (No se deben usar URLs suministradas por el usuario para ser consumidas y peticionadas por el servicio y/o microservicio)

5.9.2.2 No se debe realizar redirecciones HTTP (Estatus 301 y 303) a servicios que no son parte de Banco Pichincha y estas redirecciones solo deben ser gestionadas por el servidor y no pueden ser suministradas por el cliente.

5.9.2.3 Se debe hacer uso de un token Cross Site Request Forgery (CSRF) de un solo uso para evitar que el servicio y/o microservicio sea consumido desde front ends no autorizados

6. MEDICIÓN

Implementador	Revisor	Tipo de Validación	Estrategia	Periodicidad
Chapter de Desarrollo	Seguridad Aplicativa	Manual	Medición Manual mediante pruebas de seguridad.	Mensual

Alcance:

Chapter de Desarrollo: Desarrollar nuevas iniciativas y/o proyectos en base al presente documento

Seguridad Aplicativa: En el Instructivo de Seguridad Aplicativa versión 1.2. Los criterios indicados en este documento serán de cumplimiento obligatorio dependiendo de la naturaleza y el enfoque de la historia de usuario, el encargado de determinar qué criterios deben ser aplicados / implementados en las historias de usuario es el Ingeniero de Seguridad Aplicativa asignados a cada una de las Tribus.

7. ANEXOS

Anexo 1

<https://pichincha.atlassian.net/wiki/spaces/GDI/pages/2422702275/Nueva+Arquitectura+de+LOGs+Transaccionales+V1.0>

Anexo 2

<https://pichincha.atlassian.net/wiki/spaces/GDA/pages/2844852488/Especificaciones+tcnicas+del+esquema+de+encriptacion+HTTP>

Anexo 3

Política General de Seguridad de la Información.

<https://pichincha.sharepoint.com/:b:/r/sites/DOCUMENTACINSGSI/Documentos%20compartidos/POL%C3%8DTICAS/Poli%CC%81tica%20General%20de%20Seguridad%20de%20la%20Informacio%CC%81n%20y%20Ciberseguridad.pdf?csf=1&web=1&e=EU6uVA>

Anexo 4

Estándar de Seguridad de Datos

[https://pichincha.sharepoint.com/:f:/r/sites/ArquitecturaCiberseguridadBP-TCS/Documentos%20compartidos/Servicio%20Gesti%C3%B3n%20Estándares/Est%C3%A1ndares%20de%20seguridad%20\(publicados\)/DATOS?csf=1&web=1&e=9ml7Gf](https://pichincha.sharepoint.com/:f:/r/sites/ArquitecturaCiberseguridadBP-TCS/Documentos%20compartidos/Servicio%20Gesti%C3%B3n%20Estándares/Est%C3%A1ndares%20de%20seguridad%20(publicados)/DATOS?csf=1&web=1&e=9ml7Gf)



Anexo 5

Ley Orgánica de Protección de Datos Personales del Ecuador

<https://www.telecomunicaciones.gob.ec/wp-content/uploads/2021/06/Ley-Organica-de-Datos-Personales.pdf>

Anexo 6

Catálogo de Categorías de Datos Personales

https://pichincha.sharepoint.com/:x:/r/sites/GERENCIADECIBERSEGURIDAD2/DOCUMENTOS%20NORMATIVOS/EST%C3%81NDARES/LEY%20PROTECCION%20DATOS%20PERSONALES/Cat%C3%A1logo%20Categorias%20DP_reglas_anon.xlsx?d=wdcb2ad8f4bae4079b4a14e166fd58df6&csf=1&web=1&e=0AA0Ac

Anexo 7

Estándar de Usuarios y Claves

<https://pichincha.sharepoint.com/:b:/r/sites/GERENCIADECIBERSEGURIDAD2/DOCUMENTOS%20NORMATIVOS/EST%C3%81NDARES/USUARIOS%20Y%20CLAVES/ESTANDAR%20DE%20USUARIOS%20Y%20CLAVES%20V2.5.pdf?csf=1&web=1&e=V35XLt>

8. GLOSARIO Y DEFINICIONES

OWASP: Es un proyecto de código abierto dedicado a determinar y combatir las causas que hacen que el software sea inseguro. La Fundación OWASP es un organismo sin ánimo de lucro que apoya y gestiona los proyectos e infraestructura de OWASP.

CWE: Common Weakness Enumeration es un sistema de categorías para las debilidades y vulnerabilidades del software. Se sustenta en un proyecto comunitario con el objetivo de comprender las fallas en el software y crear herramientas automatizadas que se puedan utilizar para identificar, corregir y prevenir esas fallas.

SAST: (Static Application Security Testing) es una herramienta de pruebas de seguridad. Su caso de uso principal es informar de los problemas de seguridad y calidad en el código fuente estático.

DAST: (Dynamic Application Security Testing) es una prueba automatizada que evalúa la seguridad de una aplicación web en tiempo de ejecución.



XXS: Cross-site scripting (XSS por sus siglas en idioma inglés) es un tipo de vulnerabilidad informática o agujero de seguridad típico de las aplicaciones Web, que puede permitir a una tercera persona inyectar en páginas web visitadas por el usuario código.

XXE: La inyección de entidad externa XML (también conocida como XXE) es una vulnerabilidad de seguridad web que permite a un atacante interferir con el procesamiento de datos XML de una aplicación.

DOM: DOM significa Document Object Model, o lo que es lo mismo, la estructura del documento HTML.

Dato Personal: Dato que identifica o hace identificable a una persona natural, directa o indirectamente.

Datos personales crediticios: Datos que integran el comportamiento económico de personas naturales, para analizar su capacidad financiera.

Datos sensibles: Datos relativos a: etnia, identidad de género, identidad cultural, religión, ideología, filiación política, pasado judicial, condición migratoria, orientación sexual, salud, datos biométricos, datos genéticos y aquellos cuyo tratamiento indebido pueda dar origen a discriminación, atenten o puedan atentar contra los derechos y libertades fundamentales.