**22 - Sesión**

El servidor hace uso de las sesiones de acuerdo a la especificación "[State Management Mechanism](http://tools.ietf.org/html/rfc6265)” del protocolo HTTP el cual almacena la información de la sesión en formato de cookies cifradas (ver [Plug.Session.COOKIE](http://hexdocs.pm/plug/)) las cuales tienen cada una un periodo máximo de vida de 10 minutos en estado inactivo. Se entiende por estado inactivo de la sesión como el lapso de tiempo en el que fue realizada la ultima petición al servidor y la hora actual del mismo.

Debido al nivel de seguridad y de rastreo de actividad relacionado con el trato de la información ya sea de consulta y/o proecsamiento, todas las sesiones de la aplicación corresponden a usuarios no anónimos con lo cual se asegura que que cada sesión activa en la aplicación corresponde a una entidad válida.

**Configuración de inactividad:**

Rails:

YourApp::Application.config.session\_store :cookie\_store,

:key => ‘psrdd\_session',

:expire\_after => 10.minutes

Elixir:

ok = cowboy\_session\_config:set([

{cookie\_name, <<“psrdd\_session">>},

{expire, 6000}

])

**Diagrama de Inicio de Sesión exitoso:**

**Expiración de sesión por inactividad.**

En cada petición, el servidor realiza una validación de sesión en la que se verifica que ésta se encuentra activa. Si el resultado de tal validación es correcta el servidor de aplicaciones permite el flujo normal al recurso solicitado garantizando así la funcionalidad de la aplicación, en caso contrario el flujo de la petición es redirigido al formulario de autenticación para que se pueda crear nuevamente la sesión si es que ésta no existe o ya se encuentra caducada.

**Diagrama de Petición con sesión no válida:**

**23 - Línea base de seguridad**

**Implementación de autenticación de los usuarios (internos o clientes)**:

Conforme a lo especificado en el punto *28 - Consulta de información*en la autenticación se involucra el uso de la FIEL del usuario.

**Implementación de mecanismo de no repudio de transacciones**:

Relacionado al punto anterior (Autenticación de usuarios) se garantiza que Diverza, no almacena la llave privada de la FIEL de ningún cliente siendo este el factor que garantiza que la entidad que interactua con *la aplicación* es el propietario de la información a la cual se le otorga el acceso.

**Protección contra inyección de código**:

Como se detalla en el punto *26 - Validación de Seguridad del Aplicativo* se garantiza la seguridad de inyección de código mediante mecanismos de validación y escape de la información.

**Inicio de sesión seguro**:

Se garantiza la seguridad en el acceso a la aplicación al proteger los datos enviados/recibidos la aplicación mediante el uso de un canal cifrado basado en HTTP/TLS como se menciona en el punto *26 - Validación de Seguridad del Aplicativo (A8: Insufficient Transport Layer Protection)*

**Validación de datos de entrada / salida para evitar errores en el procesamiento de la información**:

Para garantizar la prevención de errores en el procesamiento de la información se ha unificado la codificación de los datos que la aplicación procesa. La codificación es UTF-8

Los valores de entrada son validados sintáctica y semánticamente de acuerdo al tipo de dato que mejor lo represente, al igual que los datos de salida los valores contenidos son evaluados para que los caracteres especiales considerados de riesgo sean escapados para evitar la inyección de código ajeno a la aplicación. Existe el término de “Dato Requerido” para referirse a aquellos datos que mas allá de ser informativos, son utilizados por la aplicación y no pueden contener valores nulos o vacios.

**Manejo de errores**:

Existen dos formatos de errores:

Servicios: Se refiere a un formato JSON generado como respuesta de una petición errónea mediante un servicio REST.

Web: Se refiere a un mensaje en la sección de notificaciones que pertenece a la estructura del sitio web de la aplicación.

En ambos casos se muestra un código de error de uso interno con el cual clasifica el tipo de error ocurrido seguido de un mensaje con la descripción del fallo en el cual se utiliza un lenguaje poco técnico para facilitar la interpretación del fallo presentado.

Adicionalmente a la notificación al usuario de la aplicación, los errores son escritos en el Log como evidencia del fallo en el que ademas se registra la fecha y hora del servidor.

Aunque los errores generalmente son acuse de un mal funcionamiento y requieren de registro, existen errores de los cuales no es trascendente dejar evidencia, por ejemplo, si un usuario desea modificar su contraseña pueden presentarse estos dos casos en los que se abordan dos tipos de errores:

* *En el formulario, la nueva contraseña y la verificación de contraseña no coinciden*: Se genera un error y se notifica al usuario, como se realiza mediante la aplicación web el mensaje se muestra en la área de notificaciones pero no es necesario dejar evidencia de que el usuario ha tecleado mal su nueva contraseña.
* *Los datos introducidos por el usuario son validos pero la aplicación no puede conectarse con la Base de datos*: Al ocurrir el error se notifica al usuario acerca del fallo y adicionalmente, se registra en el Log de la aplicación ya que afecta a la operación.

**Endurecer el sistema. (Hardening)**:

* **Acceso a la Aplicación:** No es posible accesar a la aplicación si no es mediante una sesión activa como resultado del proceso de autenticación. Con esto se asegura que la funcionalidad de la aplicación no se expone a entidades anónimas que pretendan consultar o alterar la información expuesta por *la aplicación*.
* **Transferencia Segura de Datos**: Los datos que son enviados/recibidos están protegidos mediante un mecanismo de cifrado sobre el protocolo de transporte de datos de tal forma que, aunque los datos (paquetes) que viajan sobre Internet pueden ser interceptados, no podrán ser interpretados como datos claros.
* **Datos Almacenados Cifrados**: La información almacenada en la Base de Datos es previamente cifrada para prevenir la privacidad y el robo de información. (Complementar con el mecanismo de cifrado, ¿Se realiza el  cifrado de la información con la misma llave o una distinta por cliente?)

**25 - Gestión de la integridad de datos**

**Integridad**

Se utiliza un mecanismo de persistencia transaccional para garantizar la integridad de la información.

**Monitoreo**

Para monitorear el estado de las transacciones se utiliza un servicio externo que se encarga de ejecutar flujos de procesos que contemplan resguardo de información en DB, de esta forma se asegura que el mecanismo transaccional funciona correctamente.

**Validación**

Seguido de la ejecución del flujo del monitoreo, se verifica que la información escrita en DB tenga consistencia e integridad, es decir, que los datos enviados en la ejecución del flujo sean los mismos escritos en DB y que no no exista perdida de información. Para verificar que se registren los eventos de cambio de información, se consume un servicio que modifica los datos previamente guardados lo cual debe agregar un registro de bitácora el cual se valida para tener la seguridad de que el mecanismo de escritura de eventos está activo.

**26 - Validación de seguridad del aplicativo**

**Implementar controles de validación de seguridad de la aplicación.**

**A1: SQL Injection**:

Es garantizada mediante el uso de la biblioteca Ecto la cual funciona como intermediario entre la aplicación y la Base de Datos. El uso de la biblioteca garantiza la validación y procesamiento de las sentencias SQL que se envían al servidor de Base de Datos mediante el uso de un DSL (Domain Specific language por sus siglas en inglés) el cual evalua los valores variables para escapara los caracteres utilizados en el ataque de SQL Injection.

**A2: Cross-site Scripting**:

La información que se procesa en la aplicación se considera segura de este ataque debido a que los datos recolectados por cualquier mecanismo de entrada (a la aplicación) son validados semántica y sintácticamente para reducir los riesgos en su procesamiento. Adicionalmente a lo anteriormente mencionado y para el caso de recibir campos de texto libre en los cuales no pueda establecerse un formato de validación, se aplica santificación de texto en los cuales se escapan los caracteres que podrían servir para realizar la inyección de código. A continuación se muestra una lista de los caracteres que en los que se aplica escape al ser desplegados en cualquier navegador web:

|  |  |
| --- | --- |
| Caracter de riesgo | Valor de reemplazo |
| & | &amp; |
| < | &lt; |
| > | &gt; |
| " | &quot; |
| ' | &apos; ó &#x27; |
| / | &#x2F; |

**A3: Broken Authentication and Session Management**:

Toda funcionalidad de la aplicación que interactuar con datos sensibles está protegida por la política de sesiones (*ver 22 - Sesión*)

**A4: Insecure Direct Object Reference**:

La información solicitada solo podrá ser accesible si se evalúa que:

* *Se realiza mediante una sesión activa*: Se verifica que la petición se realice mediante una sesión válida y activa (*ver 22 - Sesión*).
* *El solicitante es el propietario de la información*: Se verifica que la información solicitada es propiedad de la entidad que representa la sesión con que se realiza la petición.

**A5: Cross-site Request Forgery**:

...

**A6: Security Misconfiguration**:

Las aplicaciones son publicadas mediante diferentes entornos de ejecución que pueden ser Development, Test y Production, este ultimo entorno es el mas estricto y es con el cual se utiliza para publicar las aplicaciones desarrolladas por Diverza hacia la red externa (Internet).

**A7: Failure to Restrict URL Access**:

Todos los recursos dinámicos están protegidos bajo validación de sesión. Para tener accesos a la funcionalidad de cada recurso se valida que se tenga una sesión valida, activa y que quien realice la consulta a tales recursos sea el propietario de la sesión, para tales efectos nos basamos en la especificación "[State Management Mechanism](http://tools.ietf.org/html/rfc6265)” del protocolo HTTP

**A8: Insufficient Transport Layer Protection**:

La comunicación utilizada para la transmisión de datos es mediante el uso de un canal seguro bajo el protocolo HTTPS de la especificación [HTTP/TLS](http://tools.ietf.org/html/rfc2818). Las llaves de cifrado utilizadas en la comunicación entre el cliente-servidor son vigentes y respaldadas por una CA (Certificate Authority).

El cifrado de datos en el transporte previene que la información pueda ser interpretada en caso de ser interceptada por software diseñado inspeccionar los paquetes entrantes/salientes de la red  (Sniffer)

* **Diagrama de Secuencia de la comunicación HTTPS:**

**A9: Unvalidated Redirects and Forwards**:

El procesamiento de la información y el trafico de esta dentro de la aplicación, está definido por un Flujo que no depende de datos variables provenientes del cliente, es decir, los flujos de navegación y direccionamiento están explícitos dentro de la aplicación por lo que no se consideran los redireccionamientos a recursos tomados como parámetros entre peticiones

**27 - Validación de datos del aplicativo**

**Implementar controles de validación de seguridad de la aplicación.**

**Uso de catálogos**:

Son almacenados en Tablas Catálogo aquellos datos que dada su naturaleza son estáticos o muy rara vez puede requerirse una modificación de los valores que representan. Este tipo de tablas de la Base de Datos no tienen mecanismo propio de actualización por la aplicación ya que la intención de dichas tablas es mas de lectura que de escritura, de esa forma se asegura también que ningún flujo rompa la integridad de los datos que dependen de tales catálogos.

**Validación de entradas**:

La validación de datos de entrada se realiza como acción preventiva de fallos de sistema ocasionados por datos erróneos como se detalla en *23 - Línea base de seguridad (Validación de datos de entrada / salida para evitar errores en el procesamiento de la información).*

**Codificación de salidas**:

Como se detalla en *26 - Validación de seguridad del aplicativo (A2: Cross-site Scripting)*, la aplicación asegura que no se ejecutará código ajeno a la aplicación mediante la explotación del ataque Cross Site Scripting para lo cual se escapan los caracteres que puedan representar un riesgo.

**Validación y administración de contraseñas**:

**Administración de sesión**:

El uso de la sesión utiliza la especificación del protocolo HTTP "[State Management Mechanism](http://tools.ietf.org/html/rfc6265)” como se detalla en 22 - Sesión.

**Practicas de criptografía**:

Se utiliza el cifrado de información para cualquier dato sensible almacenado en Base de Datos utilizando el principio de Criptograma Asimétrica en el que se emplea el Certificado Publico como llave de cifrado y el Certificado Privado como llave para revertir tal proceso. Con el fin de garantizar la privacidad de los datos almacenados, es el propietario de la información la única entidad que posee la contraseña de la Certificado Privado.

**Administración de errores y bitácora**:

**Errores:**

Como se menciona en *23 - Línea base de seguridad (Manejo de errores)* los errores notificados pueden presentarse en dos diferentes formatos de acuerdo al método de comunicación con la aplicación.

* JSON: Para la comunicación Sistema-Sistema.
* Notificaciones: Para el cliente web de *la aplicación*.

**Bitácora:**

Los eventos a nivel de Sistema Operativo, Aplicación y Base de datos son almacenados en una Base de Datos cuya información está cifrada para evitar huecos en la seguridad y privacidad de los datos de los clientes. Los eventos provenientes de los tres niveles auditables son concentrados mediante el protocolo syslog.

Para garantizar el acceso a la información a la Autoridad Auditora (SAT) se realiza mediante una aplicación con acceso vía web la cual está protegida bajo el mismo mecanismo de seguridad de almacenamiento y transporte empleados por *la aplicación*.

**Protección de datos**:

Los datos almacenados en DB son protegidos mediante un mecanismo de cifrado reversible en el que se garantiza que únicamente el propietario de la información puede tener acceso a los datos en claro. Para garantizar lo anterior, se utiliza como llave de cifrado un certificado privado cuya contraseña coincide con la utilizada para el acceso de la aplicación lo cual garantiza la privacidad de los datos almacenados.

* **Creación de la llave de cifrado**:  Las llaves de encripción se generan el el momento en que se registra el usuario ya que incluso para guardar los datos del nuevo usuario se requieren estas llaves.
* **Almacenar/actualizar datos en DB:** La información que se almacenan es cifrada con la llave publica del usuario al que pertenece la sesión mediante la cual se solicitan nuevos registros o alteraciones de la información registrada en la Base de Datos  (Definir donde se almacena la llave privada)
* **Consulta de datos almacenados en DB**:  La consulta de la información en claro solo puede ser posible si reconoce la contraseña de la llave privada, es mediante este recurso con el cual se puede revertir el proceso de cifrado de los datos almacenados.

(Definir el caso cuando el usuario cambia su contraseña para que se actualice también la de la llave privada)

**Seguridad de la comunicación**:

Se garantiza la seguridad en el acceso a la aplicación al proteger los datos enviados/recibidos la aplicación mediante el uso de un canal cifrado basado en HTTP/TLS como se menciona en el punto *26 - Validación de Seguridad del Aplicativo (A8: Insufficient Transport Layer Protection)*

**Configuración del sistema**:

Version de las herramientas bajo el cual está construido

**Seguridad en la base de datos**:

**Administración de archivos**:

**Mejores prácticas de codificación**:

* Documentación: Las funciones, métodos, clases y/o módulos cuentan con documentación que describe su funcionalidad
* Se utiliza el principio de responsabilidad única sobre las funciones ó métodos que conforman la aplicación.
* Se evita cargar grandes volúmenes de información que puedan comprometer la estabilidad y rendimiento de la aplicación.
* El código fuente es administrado mediante un Sistema de Control de Versiones Distribuido en el cual se mantiene el historial de las versiones liberadas así como diferentes variaciones del código base en tiempo de desarrollo. Lo anterior con la finalidad de evitar la contención ocasionado por las modificaciones simultáneas a una misma base de código.
* Se realizan pruebas unitarias y de integración del código.