Seguridad

Respuesta a inquietudes y sugerencias sobre la seguridad en SemanticWebBuilder

Sergio Martínez

Agosto 2014

Seguridad

Respuesta a inquietudes y sugerencias sobre la seguridad en SemanticWebBuilder

Vulnerabilidades

Protección contra SQL Injection

La inyección (SQL, xPath, LDAP, Scripts) consiste en permitir que un atacante pueda alterar las sentencias de código que serán enviadas a diferentes procesadores (parsers), esto ocurre por que en la programación se integre información que viene desde una fuente insegura (Internet) con el código local. Para evitarlo se debe restringir el uso de parsers, usar parsers parametrizados ó usar tablas de correspondencia entre los valores externos y lo que se agrega a las sentencias de código que se envían a los parsers.

SemanticWebBuilcer (SWB) en sus recursos básicos y sistema de administración transforma todos los valores recibidos a triples y sus 6 queries están parametrizados, evitando así inyecciones a la base de datos en uso; el otro punto donde se interactúa con un parser es cuando se activa el repositorio externo LDAP en donde ocurre un bind, en este caso el bind es parametrizado.

Protección contra Cross Site Scripting (XSS)

El Cross Site Scripting es uno de los problemas de seguridad más difíciles de resolver pues consiste en que un atacante pueda introducir datos que al momento de procesarse por un navegador provoquen el cambio de contexto de operación del navegador, ejecutando código colocado por el atacante. Para evitar este tipo de ataque se debe evitar colocar en html generado información que de origen provenga de una fuente insegura sin haberle dado un tratamiento. En SWB en el modelado identificamos que propiedades pueden guardar código (que viene desde la administración de la plataforma) y en las demás se codifica hacia html evitando así que al momento de renderearlo por el navegador esos datos se puedan interpretar como código. Periódicamente se realizan revisiones al código para ubicar y corregir los otros puntos en donde se pudiera mostrar datos que provienen de Internet.

Protección contra Cross Site Request Forgery (CSRF)

El Cross Site Request Fergery consiste en que un atacante coloca una "bomba" en algún portal o la hace llegar por correo electrónico de tal manera que la víctima pueda dar click en la "bomba", esta bomba hará que el navegador o el cliente de correo habra una petición al servicio que se pretende atacar y que si la víctima se encuentra autenticada en el sistema, el navegador no podrá diferenciar si fue una acción legitima de la víctima o es producto del ataque.

SWB crea todos sus elementos de manera desactivada y los borrados se envían a la papelera de reciclaje; por lo anterior, en el caso de que un atacante obtuviera toda la información necesaria para realizar un cambio en los elementos en la administración, debería lograr que la víctima caiga en CSRFs serializados para poder realizar un daño significativo.

Manejo de protocolo HTTPS

El HTTPS es manejado por el Servidor de aplicaciones, WebServer o balanceador según se tengan configurados los dispositivos en un portal, SWB opera adecuadamente en HTTP y en HTTPS

Encripción de información sensible en capa de persistencia (podría ser de una o dos vías dependiendo del caso de uso)

Para SWB el único dato sensible que se utilizan son las contraseñas de usuarios, estas se persisten digeridas con SHA-512.

En la biblioteca SWBUtils.CryptoWrapper existen wrappers listos para realizar digestiones, cifrados AES128, generación de llaves RSA e intercambio seguro de llaves mediante Diffie-Hellman Key-Agreement.

Cada instancia de SWB contiene un par propio de llaves con el que se puede interactuar para obtener información de monitoreo de la instancia.

Registro de Usuario

SWB cuenta con un recurso para realizar el registro de usuarios, éste es básico y genérico, por lo que se recomienda realizar un recurso mediante el cuál se realice el registro siguiendo las limitaciones y regulaciones que apliquen.

Autenticación

El archivo de propiedades security.properties define las opciones de seguridad que se pueden usar en SWB como son:

- **password/minlength**: define el tamaño mínimo para aceptar una contraseña como válida
- **password/differFromLogin**: define si SWB deberá asegurarse de que la contraseña sea distinta al login para aceptarla como válida.
- **password/complexity**: define el nivel de complejidad que se usará para aceptar una contraseña como válida, cuentea con tres niveles none - el usuario puede usar la combinación que prefiera, simple - el usuario deberá incluir letras y números en su contraseña y complex - el usuario deberá incluir letras, números y caracteres especiales en la definición de la contraseña.
- **password/forceChangeOnFirstLogon**: fuerza al usuario a cambiar su contraseña la primera vez que se firma en el sistema, esto sirve para los casos en que los usuarios son creados por un tercero.
- **password/expiresInDays**: fuerza al usuario a cambiar su contraseña una vez que han transcurrido el número de días indicado por este valor.
- **account/inactiveInDays**: desactiva una cuenta que ha estado inactiva por la cantidad de días indicados por este valor; para volver a activarla se requiere que un administrador la active desde la administración de SWB
- **account/sendMailOnLogon**: enviará un correo electrónico al usuario avisando cada que este se firme en el sistema.
- **session/restrictToSingleIP**: evita que existan dos autenticaciones simultáneas al mismo usuario con diferentes IPs.
- **session/restrictMultipleLogon**: evita que existan dos o más sesiones simultáneas con el mismo usuario.
- **login/encryptData**: utiliza un sistema de cifrado de llave pública para el envío de las contraseñas en la pantalla de autenticación de SWB

Cualquier otra regla se deberá implementar en el registro y en el módulo de autenticación que se usen.

CAPTCHA

Existe un CAPTCHA para los formularios semánticos generados de manera automático por la herramienta, y el API de generación de imágenes está disponible para los recursos que se desarrollen en la herramienta.

Pruebas de Seguridad

Para las pruebas de seguridad y en el espíritu de mejorar la herramienta, nos gustaría poder colaborar en las mismas y sugerimos que previo a la prueba se haya realizado lo siguiente:

- Negación de Servicios por consumo de recursos: Configurar adecuadamente los parámetros de memoria del JVM, caches de recursos y el sistema de alertas (Modo ataque)
- Cleartext submission of Password: Configurar login/encryptData en security.properties
- User agent-dependent response: Revisar las reglas que se hayan aplicado al sitio
- Password field submitted using GET method: Este dependerá de como se implemente el registro de usuarios y el cambio/recuperación de contraseña, los recursos actuales no hacen diferencia en el método por el cual se suministran los datos
- SSL cookie without secure flag set: las cookies que SWB coloca son sin el secure flag y/o HttpOnly activado
- Session token in URL: La sesión se coloca como parámetros para los applets en la administración
- File upload functionality: Existen varios mecanismos usados principalmente en la administración
- Database connection string disclosed: Que la conexión se configure vía JNDI, o se haya reemplazado la contraseña en el DB.properties por su versión cifrada
- Frameable response (potential Clickjacking): No existe ninguna protección contra este tipo de ataques