12 técnicas para la construcción de aplicaciones seguras con Java

Investigador principal

Diego León García Pineda

Cooinvestigadores

Juan Diego Calderón

Ricardo Montoya Gallego

Mateo Gil López

Diego Ramírez

Santiago Gil

Daniel Soto

Introducción

Hoy en día se ha visto que es necesario adoptar buenas prácticas de programación ya que los índices de violaciones para extraer información están en aumento debido al desconocimiento o posibles errores que se cometen al desarrollar aplicaciones informáticas; la falta de información y conocimientos relacionados al tema de seguridad informática, hacen que ésta sea altamente vulnerable. La importancia de acoger prácticas relacionadas a mitigar el riesgo de fuga de información, se debe a las altas amenazas de pérdida de datos, dinero y otra información vital para el funcionamiento de cualquier negocio o empresa donde sea necesarios implementar soluciones de software.

Actualmente, el mundo está siendo influenciado por la tecnología a tal punto que las empresas son cada vez más dependientes de un sistema de software, por tanto, los equipos de desarrollo deben garantizar que éste sea seguro con el fin de proteger el activo más importante denominado, información. Un banco, una página de compras por internet o hasta un sistema de supermercado; es un blanco para personas con malas intenciones, es por ello que en el trascurso del texto se abordarán varias temáticas para prevenir y mitigar el riesgo de ataques, además se darán a conocer diferentes prácticas que se deben tener en cuenta a la hora de desarrollar una aplicación.

Justificación

Construir aplicaciones seguras ya no es un plus sino una obligación que se debe abordar y afrontar por parte de cualquier equipo de desarrollo en un proyecto; la información es el activo más importante de una empresa y muchos atacantes desean acceder a ella por diversos métodos, por lo tanto, se han diseñado algunas técnicas para prevenir y/o mitigar posibles ataques en los cuales cualquier proyecto de software se vea involucrado. Además de ello, se mostrarán algunas buenas prácticas de programación para el fácil mantenimiento del mismo, tales como: métodos de ofuscación, cifrado de código, entre otras.

Contenido

Técnica 1

Revisión Top 10 OWASP 2017

El objetivo del proyecto OWASP es crear conciencia acerca de la seguridad de las aplicaciones mediante la identificación de algunos de los riesgos más críticos que se encuentran en las organizaciones. A continuación, se listarán las vulnerabilidades más comunes:

A1 – Injection

Las fallas de inyección bajo SQL, XXE o LDAP hacen que un atacante acceda a un sistema logrando romper la seguridad del mismo sin la debida autorización, esto como resultado puede generar fuga de información.

A2 – Broken Authentication and Session Management

Las técnicas de autenticación y autorización para el control de acceso de una aplicación, a menudo se suelen implementar de forma incorrecta, permitiendo a los atacantes romper fácilmente la seguridad por medio de claves, tokens, xploids y poder acceder al sistema de forma temporal o permanente.

A3 – Cross-Site Scripting (XSS)

Permite a los atacantes ejecutar scripts en el navegador de la víctima con el fin de secuestrar sesiones de usuario, desfigurar sitios web o redirigir al usuario a sitios maliciosos.

A4 – Broken Access Control

Las restricciones que se configuran en el sistema en cuanto a los contenidos y su nivel de confidencialidad a menudo se implementan mal, los atacantes se aprovechan de esto para acceder a información sensible, logrando hacer cambios no deseados en la aplicación como modificar el acceso a la misma.

A5 – Security Misconfiguration

Los valores predeterminados con los que vienen herramientas, frameworks, motores de bases de datos, no son seguros ya que estos vienen de fábrica.

A6 – Sensitive Data Exposure

En aplicaciones web y API, es frecuente ver que no se protegen datos sensibles como historias clínicas, número de cuentas bancarias, tarjetas de crédito, etc, lo cual puede hacer que un atacante los robe y cometer diferentes delitos informáticos.

A7 – Insufficient Attack Protection

La mayoría de las aplicaciones y API carecen de la capacidad básica para detectar, prevenir y responder a ataques tanto manuales como automáticos. La protección de ataque va más allá de la

validación de entrada básica e involucra detección automática, registro, respuesta e incluso bloqueo de intentos de explotación.

A8 – Cross-Site Request Forgery (CSRF)

Un ataque CSRF fuerza al navegador de una víctima conectada a enviar una solicitud HTTP falsificada, que incluye la cookie de sesión de la víctima y cualquier otra información de autenticación incluida automáticamente, a una aplicación web vulnerable. Tal ataque permite al atacante obligar al navegador de la víctima a generar solicitudes que la aplicación vulnerable cree que son solicitudes legítimas de la víctima.

A9 – Using Components with Known Vulnerabilities

Si se “explota” (falla) un componente vulnerable, dicho ataque puede facilitar la pérdida grave de datos o la toma del servidor. Las aplicaciones y las API que usan componentes con vulnerabilidades conocidas pueden socavar las defensas de las aplicaciones y permitir varios ataques e impactos

A10 – Underprotected APIs

Las aplicaciones modernas a menudo implican aplicaciones de cliente enriquecidas y API, como JavaScript en el navegador y aplicaciones móviles, que se conectan a una API de algún tipo (SOAP / XML, REST / JSON, RPC, GWT, etc.). Estas API a menudo no están protegidas y contienen numerosas vulnerabilidades.

Técnica 2

Detectar y evitar vulnerabilidades

A continuación, se listará una serie de recomendaciones para el manejo de la información en la organización en diferentes situaciones.

- **Correo electrónico:** Es necesario proteger la información involucrada en comercio electrónico, por eso es importante protegerla en redes públicas de cualquier actividad fraudulenta.
- **Transacciones en línea:** La información que se encuentra en las transacciones en línea es susceptible a enrutamiento incompleto, alteración de mensajes no autorizados, duplicación, reutilización de transacciones.
- **Información disponible al público:** la integridad de la información en un sistema público, debe ser protegida para evitar un uso incorrecto o fraudulento de la misma
- **Autenticación de usuarios para conexiones externas:** Los métodos de autenticación deben ser usados apropiadamente para controlar el acceso de los diferentes usuarios al sistema.
- **Restricción de acceso a la información:** se debe definir una política de control de acceso a las diferentes funcionalidades del sistema, por medio de roles y perfiles.
- **Validación de datos de entrada:** se deben validar los datos de entrada con el fin que se garantice que sean correctos y apropiados.
- **Validación de datos de salida:** se deben validar las salidas para comprobar que la información sea consistente y se exponga sólo lo relevante
- **Fugas de información:** se debe evitar publicar información sensible
- **Administración de contraseñas:** se debe generar una política con el fin de definir las claves de acceso al sistema, se considera usar contraseñas fuertes.
- **Protección de datos y privacidad de información personal:** Se debe tener en cuenta el marco legal vigente, reglamentos y cláusulas del contrato.
- **Controles de las redes:** se debe de hacer monitoreo constante a las redes para protegerlas de amenazas, incluyendo la información del tráfico de las mismas, también se debe identificar los servicios propios o de terceros.
- **Controles de acceso al código fuente de los programas:** El acceso al código fuente de los programas, debe ser restringido.
- **Control de vulnerabilidades técnicas:** Es conveniente realizar revisiones periódicas en los sistemas de información con el fin de identificar las vulnerabilidades y tomar acciones pertinentes.

Técnica 3

Pruebas unitarias

¿Qué es una prueba?

Una prueba en desarrollo de software es una manera de detectar que un componente o fragmento de código funciona correctamente.

A continuación, se listan las características de una prueba unitaria

- Una prueba unitaria consiste en evaluar solamente pequeñas cantidades de código, en resumen, solamente se prueba el código del requerimiento específico.
- Se aíslan de otro código y de otros desarrolladores: Solo se prueba exclusivamente el código relacionado con el requerimiento y no interfiere con el trabajo hecho por otros desarrolladores.
- Solamente se prueban los endpoints públicos: Esto principalmente porque los disparadores de los métodos privados son métodos públicos por lo tanto se abarca el código de los métodos privados dentro de las pruebas.
- Los resultados son automatizados: Cuando se ejecutan las pruebas se pueden hacer de forma individual o de forma grupal.
- Estas pruebas las hace el motor de prueba y los resultados de los mismos deben de ser precisos con respecto a cada prueba unitaria desarrollada
- Repetible y predecible: No importa el orden y las veces que se repita la prueba, el resultado siempre debe de ser el mismo.
- Son rápidos de desarrollar: Contrariamente a lo que piensan los desarrolladores, acerca del alto tiempo que se invierte en una prueba normal, las pruebas unitarias generalmente deben de ser simples y rápidas de desarrollar.
- Difícilmente una prueba unitaria deba de tomar más de cinco minutos en su desarrollo.

Beneficios de las pruebas unitarias

- Lo que se busca con las pruebas unitarias es mejorar la calidad ya que se pueden reducir los tiempos de depuración y corrección de incidencias
- Se pueden realizar cambios cuando el código necesite ser factorizado o mejorado
- Permiten probar o depurar un módulo sin necesidad de disponer del sistema completo
- Permiten reducir los problemas y tiempos dedicados a la integración
- Permite hacer pruebas de regresión de manera automática
- Sirven para detectar problemas antes de enviar la aplicación a producción

Herramientas:

JUnit : es un framework de código abierto desarrollado especialmente para crear, ejecutar y hacer reportes de estado de conjuntos de Prueba Unitaria automatizadas hechos en lenguaje Java.

JUnit es uno de los frameworks más populares en Java para realizar pruebas unitarias y llevar un desarrollo utilizando la práctica de Test Driven Development.

Ejemplo de prueba con JUNIT

En el ejemplo a continuación, se pueden ver diferentes Test a un método De la clase EmpleadoBR cuyo método se llama, calculaSalarioBruto. En la Prueba Unitaria se puede observar que siempre se deben comparar el resultado real y el resultado esperado.

```
public class EmpleadoBRTest {
  @Test
  public void testCalculaSalarioBruto1() {
    float resultadoReal = EmpleadoBR.calculaSalarioBruto(
                          TipoEmpleado.vendedor, 2000.0f, 8.0f);
    float resultadoEsperado = 1360.0f;
    assertEquals(resultadoEsperado, resultadoReal, 0.01);
  }

  @Test
  public void testCalculaSalarioBruto2() {
    float resultadoReal = EmpleadoBR.calculaSalarioBruto(
                          TipoEmpleado.vendedor, 1500.0f, 3.0f);
    float resultadoEsperado = 1260.0f;
    assertEquals(resultadoEsperado, resultadoReal, 0.01);
  }

  @Test
  public void testCalculaSalarioNeto1() {
    float resultadoReal = EmpleadoBR.calculaSalarioNeto(2000.0f);
    float resultadoEsperado = 1640.0f;
    assertEquals(resultadoEsperado, resultadoReal, 0.01);
  }

  @Test
  public void testCalculaSalarioNeto2() {
    float resultadoReal = EmpleadoBR.calculaSalarioNeto(1500.0f);
    float resultadoEsperado = 1230.0f;
    assertEquals(resultadoEsperado, resultadoReal, 0.01);
  }
}
```

Técnica 4

Manejo de cadenas de caracteres

Al construir una aplicación segura, no sólo basta con que sea difícil de hurtar la información que maneja, también se debe tener en cuenta qué tan eficiente es al procesar la información, es por ello que cuando se vayan a concatenar grandes cantidades de datos, a continuación se darán unas recomendaciones.

Uso de las clases String, StringBuilder y StringBuffer.

Todas estas Clases se usan para cadenas de caracteres que pueden cambiar,

En general se usan String, StringBuilder o StringBuffer según lo siguiente:

- Usaremos String si la cadena de caracteres no va a cambiar.
- Usaremos StringBuilder si la cadena de caracteres puede cambiar y solamente tenemos un hilo de ejecución.
- Usaremos StringBuffer si la cadena de caracteres puede cambiar y tenemos varios hilos de ejecución.

A continuación, se harán dos pruebas concatenando una frase 5000 veces, en una máquina con sistema operativo Windows 10, con 4 Gigabytes de memoria RAM y procesador Core i5. Para esta prueba se usó el IDE netbeans.

Ejercicio:

Concatenar la frase “Universidad católica” 5000 veces, usando String y StringBuilder.

Prueba 1:

Concatenación usando clase String

```
public static void main(String[] args) {
   long startTime = System.currentTimeMillis();
    String strPrueba = "Universidad Católica";
    String tmpString = "";

    int numIteraciones = 5000;
    for (int i = 0; i < numIteraciones; i++) {
       tmpString = tmpString + strPrueba;

    }

    System.out.println(tmpString);
    long endTime = System.currentTimeMillis() - startTime;
    System.out.println(endTime);
}
```

Tiempo de la prueba, 1313 ms

```
run:
Universidad CatólicaUniversidad CatólicaUniversidad CatólicaUniversidad CatólicaUniversidad CatólicaUniversidad CatólicaUniversidad CatólicaUniversidad (
1313
BUILD SUCCESSFUL (total time: 1 second)
```

Concatenación usando clase StringBuilder

```java
public static void main(String[] args) {
    long startTime = System.currentTimeMillis();
     String strPrueba = "Universidad Católica";
     String tmpString = "";
     StringBuilder strBuilder = new StringBuilder();
     int numIteraciones = 5000;
     for (int i = 0; i < numIteraciones; i++) {

        strBuilder.append(strPrueba);
     }

     System.out.println(strBuilder.toString());
     long endTime = System.currentTimeMillis() - startTime;
     System.out.println(endTime);
}
```

Tiempo de la prueba, 147 ms

```
Universidad CatólicaUniversidad CatólicaUniversidad CatólicaUniversidad CatólicaUniversidad CatólicaUniversidad Cató
147
BUILD SUCCESSFUL (total time: 0 seconds)
```

Conclusión:

Stringbuilder es mucho más eficiente a la hora de realizar una concatenación de datos.

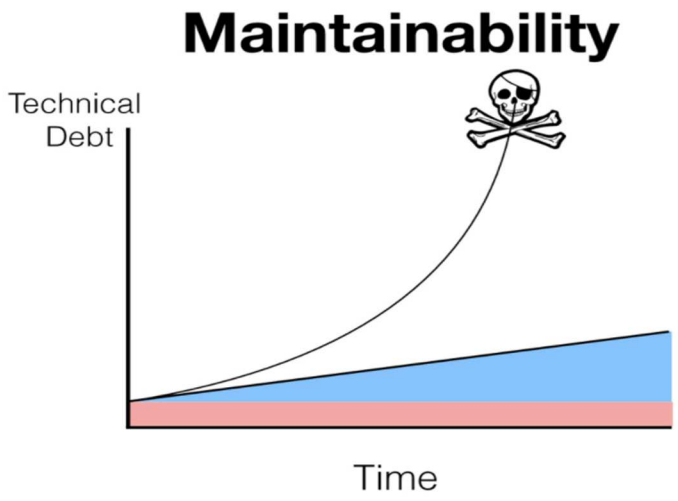
Técnica 5

Análisis estático de código

La calidad del código es imprescindible a la hora de realizar mantenimientos en la aplicación y se pueden realizar de dos formas:

- **Automático:** El análisis realizado por un programa de ordenador, o análisis automático, reduce la complejidad que supone detectar problemas en la base del código ya que los busca utilizando a unas reglas que tiene predefinidas.
- **Manual:** El análisis realizado por una persona, o análisis manual, se centra en apartados propios de nuestra aplicación en concreto como, por ejemplo, determinar si las librerías que está utilizando nuestro programa se están utilizando debidamente o si la arquitectura de nuestro software es la correcta**.**

En el análisis estático de código es muy común hablar de **deuda técnica,** la cual consiste “en el coste a pagar por usar malas prácticas de desarrollo”, esto se traduce en aumento del tiempo para corregir un error.



En conclusión, la deuda técnica afecta de forma considerable la mantenibilidad, por tanto, es importante usar prácticas que permitan reducirla al máximo.

¿cuál es el fin de los analizadores estáticos de código?

- Buscar partes del código que puedan reducir el rendimiento.
- Detectar malas prácticas que puedan provocar errores en el software.
- Descompilar el flujo de datos.
- Buscar líneas de código que puedan tener una excesiva complejidad.
- Detectar fallas en la programación que puedan suponer un problema en la seguridad.

Herramientas para lenguaje de programación Java para análisis estático de código

PMD: es una herramienta de calidad de código encargada de validar los estándares de construcción de un desarrollo. Es decir, chequea la sintaxis del código fuente que ha sido desarrollado, encontrando las ocurrencias de un determinado problema que haya sido previamente configurado para ser detectado.

SonarQube: es una plataforma para realizar análisis estático, de código abierto, en este se pueden obtener métricas con el fin de ayudar a mejorar la calidad del código de un programa.

Técnica 6

Ofuscación de código

Cuando se instalan aplicaciones autoejecutables (*.jar, *.apk), se les puede hacer ingeniería inversa a los archivos, con lo cual queda vulnerable el código fuente y por ende queda propensa a detectar diferentes vulnerabilidades o al hurto de propiedad intelectual.

Existen herramientas para realizar ingeniería inversa, a continuación, se listarán las más conocidas

- **Java Decompiler:** Es un programa que traduce archivos .class a código fuente java, se puede obtener de http://jd.benow.ca/

- **7Zip:** potente compresor y descompresor de archivos, se puede obtener de http://www.7-zip.org/

Al tener en cuenta que se le puede realizar ingeniería inversa a las aplicaciones, es necesario proteger la propiedad intelectual, por tal razón se pueden usar herramientas para ofuscar código.

¿Qué es ofuscar código?

Según el diccionario de la real academia, ofuscar significa ” oscurecer y hacer sombra” Se refiere a realizar cambios no destructivos, es decir, se hacen modificaciones en el código para que sea menos comprensible sin alterar el funcionamiento.

¿Qué es Proguard?

Es una herramienta ofuscadora de código la cual se puede usar en los archivos ejecutables, logrando así ocultar el código fuente de las aplicaciones, logrando así que este sea menos legible para personas que quieran realizar acciones poco honestas.

Técnica 7

Escaneo de metadatos

Cuando una aplicación se encuentra publicada en Internet, está expuesta a toda clase de ataques con el fin de desestabilizarla o poder extraer información de los usuarios de la misma, para ello se usa esta técnica para extraer información.

FOCA: es una herramienta para el escaneo de metadatos por medio de motores de búsqueda a como GOOGLE y Bing, a través de la URL del sitio.

¿Qué funciones tiene FOCA?

Usa los motores de búsqueda como GOOGLE o Bing para descubrir los archivos ofimáticos que tiene un dominio, los descarga masivamente, les extrae los metadatos, organiza los datos y nos muestra la siguiente información:

- Nombres de usuarios del sistema
- Rutas de archivos
- Versión del Software utilizado
- Correos electrónicos encontrados
- Fechas de Creación, Modificación e Impresión de los documentos.
- Sistema operativo desde donde crearon el documento
- Nombre de las impresoras utilizadas
- Permite descubrir subdominios y mapear la red de la organización
- Nombres e IP descubiertos en Metadatos
- Búsqueda de nombres de dominio en Web: Google o Bing [Interfaz web o API]
- Búsqueda de registros Well-Known en servidor DNS
- Búsqueda de nombres comunes en servidor DNS
- Búsqueda de IPs con resolución DNS

- Búsqueda de nombres de dominio con BingSearch ip
- Búsqueda de nombres con PTR Scanning del segmento de red con DNS interno
- Transferencia de Zonas
- Detección automática de DNS Cache
- Vista de Roles
- Filtro de criticidad en el log

¿Qué son los metadatos?

Es información acerca de los datos y sirven para suministrar información de los datos producidos, es decir, información de contenido, condiciones, disponibilidad, entre otras.

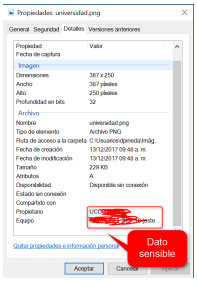
¿De dónde se puede extraer metadatos?

De archivos tipo pdf, jpg, png, etc

¿Por qué es importante ocultar o eliminar los metadatos?

Los metadatos pueden revelar tecnologías, correos electrónicos, fechas u otra información que puede ser útil para un atacante.

La imagen a continuación muestra los metadatos de un archivo png



Recomendaciones

Si se van a publicar imágenes, pdf u otro tipo de archivos en una aplicación o página web, es recomendable eliminar los metadatos más sensibles, para ello se usan las siguientes herramientas:

Doc Scrubber: Elimina metadatos de archivos Word, sitio oficial, http://www.brightfort.com/docscrubber.html

Rhdtool: Elimina metadatos de documentos Microsoft Office, sitio oficial, https://www.securitynull.net/tag/rhdtool/

MetaStripper: Elimina metadatos de archivos JPEG, sitio oficial http://www.photothumb.com/metastripper/

BeCyPDFMetaEdit: herramienta para la eliminación de metadatos de archivos pdf, sitio oficial http://www.becyhome.de/becypdfmetaedit/description_eng.htm

Técnica 8

Configuración del CHARSET

El juego de caracteres o CHARSET es para indicarle al navegador web el juego de caracteres que debe usar; es muy recomendable usarlo para las siguientes situaciones:

- Para evitar que el navegador web tenga que adivinar el conjunto de caracteres usados, se recomienda especificarlo directamente, en especial para archivos generados dinámicamente.
- ·Las consecuencias de no usar un CHARSET específico pueden resultar en vulnerabilidades XSS.
- Asegurar que la página se vea correctamente

Técnica 9

Evitar el almacenamiento en caché en los formularios HTML

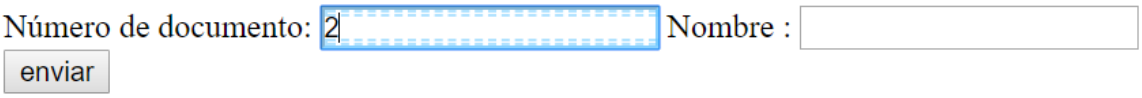
En algunos casos es necesario no tener en la caché los datos de un formulario html, debido a que se puede comprometer la identidad de las personas que han utilizado ese mismo formulario para almacenar información, en el ejemplo a continuación se verá con más detalle la situación

Como se puede apreciar, se listarán los datos que anteriormente han sido ingresados en el formulario, para evitar esto, se procede a realizar lo siguiente:

```
<html>

    <head>
        <!-- define juego de caracteres-->
        <meta charset="utf-8">
        <title>Mi página</title>

    </head>
    <body>
        <form>
          Número de documento: <input type="text" name="numero_documento" autocomplete="off">
          Nombre :  <input type="text" name="name2" >
                    <input type="submit" value="enviar">
</form>
    </body>

</html>
```

Se adiciona la propiedad autocomplete = off

Y como resultado ya no se almacenan en caché datos que pueden comprometer la identidad de una persona o información sensible



Técnica 10

Protección y filtrado en las entradas de datos

El ataque XSS o “Cross-Site Scripting” es un tipo de ataque que está orientado a las aplicaciones web y consiste en inyectar código malicioso en JavaScript.

Tipos de ataques XSS

Indirecto: Consiste en modificar el comportamiento de la página web por medio de la modificación variables que transportan valores de una página a otra, sin embargo no se modifica de forma permanente el código

Directo: Consiste en inyectar a través de los campos de un formulario código JavaScript y almacenarse esta información en la base de datos, se afecta el comportamiento del aplicativo web de forma permanente, un ejemplo concreto podría ser en el manejo de foros.

Este ataque es útil para:

- Modificación del contenido
- Acceso a la información confidencial o restringida
- Robo de sesión
- El robo de identidad
- La alteración de la funcionalidad del aplicativo web
- Ataques de denegación de servicio.

Solución

Para evitar este tipo de situaciones es necesario proteger todas las entradas de datos y esto se hace a través de la librería **apache commons lang,** la cual consta de varios métodos que detectan sintaxis sospechosas de diferentes tipos de lenguajes de programación que se deseen inyectar en cadenas de caracteres; para ver más información visitar el sitio oficial https://commons.apache.org/proper/commons-lang/

Implementación

En el ejemplo a continuación se puede ver la implementación de la librería en un Servlet de Java, a continuación, se listarán los pasos.

1. Se debe tener en el proyecto la librería Apache commons lang.
2. Se debe importar la clase StringScapeUtils.
3. Usar el método de la clase del paso anterior que se necesite

En la imagen a continuación se pueden observar los diferentes métodos con los lenguajes de programación que es capaz de detectar la librería

A continuación, se mostrará la implementación de la librería

```java
@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * Default constructor.
     */
    public LoginServlet() {
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String password = request.getParameter("password");
        String passwordFiltrado = StringEscapeUtils.escapeJavaScript(password);
    }

}
```

Método para filtrar código JavaScript

Técnica 11

Criptografía

La palabra criptografía proviene del griego “criptos” que significa “oculto” y “grafe” de escritura que alude textualmente a la “escritura oculta”. La criptografía es la ciencia que resguarda documentos y datos que actúa a través del uso de las cifras o códigos para escribir algo secreto en documentos y datos que se aplica a la información que circulan en las redes locales o en internet.

En algunas ocasiones es pertinente usar la criptografía para proteger información sensible, sobretodo en aplicaciones que requieran acceso a una red de datos, esto con el fin de proteger la información de personas con intensiones maliciosas.

Tipos de cifrado:

- **Cifrado simétrico:** consiste en que dos o más usuario tienen una clave secreta única la cual sirve para cifrar o descifrar la información.
- **Cifrado asimétrico:** necesita de dos llaves una pública y otra privada para cifrar o descifrar la información.

Tipos de algoritmos de cifrado

En java existen números algoritmos para cifrar la información, entre éstos se tiene:

Algoritmos simétricos de cifrado proporcionadas para Java

- DES - KeyLength predeterminado de 56 bits (longitud)
- AES
- RC2, RC4 y RC5
- IDEA
- Triple DES - KeyLength por defecto 112 bits
- Blowfish - KeyLength defecto 56 bits
- PBE con MD5 y DES
- PBE con HmacSHA1 y DESede
- DES ede

Modos de cifrado

- BCE
- CBC
- CFB
- OFB
- PCBC

Algoritmos de cifrado asimétricos implementados para java

- RSA
- Diffie-Hellman - KeyLength defecto 1024 bits
- Hashing / Message Digest algoritmos implementados por SunJCE
- MD5 - por defecto 64 bytes de tamaño
- SHA1 - por defecto 64 bytes de tamaño

Algoritmos de cifrado más usados:

MD5 es un algoritmo muy usado para cifrar información, el link http://www.md5decrypt.org/, ofrece una forma para cifrar información, sin embargo algunos consideran que es un algoritmo vulnerable por lo que recomienda usar el SHA-1.

El algortimo tipo AES (Advanced Encryption Standard), es un algoritmo de clave simétrica y es mucho más seguro, es un algoritmo usado por el departamento de defensa de los estados unidos y consiste en un cifrado por bloques.

A continuación, se muestra cómo usarlo:

1. Se debe adjuntar al proyecto la librería **commons codec** de apache, sitio oficial https://commons.apache.org/proper/commons-codec/
2. Adjuntar código fuente java

```java
public class Cypher {
    private static final String KEY = "7977777777777777";

    public static String encrypt(String input){
        byte[] crypted = null;
        try{
            SecretKeySpec skey = new SecretKeySpec(KEY.getBytes(), "AES");
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, skey);
            crypted = cipher.doFinal(input.getBytes());
        }catch(Exception e){
            System.out.println(e.toString());
        }
        return new String(Base64.encodeBase64(crypted));
    }

    public static String decrypt(String input){
        byte[] output = null;
        try{
            SecretKeySpec skey = new SecretKeySpec(KEY.getBytes(), "AES");
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
            cipher.init(Cipher.DECRYPT_MODE, skey);
             output = cipher.doFinal(Base64.decodeBase64(input));
        }catch(Exception e){
            System.out.println(e.toString());
        }
        return new String(output);
    }
}
```

clave privada

método para cifrar

método para descifrar

3. Prueba del algoritmo y resultado


```java
public class Test {

    public static void main(String [] args){

        String palabra ="Semillero aplicaciones seguras";
        System.out.println("palabra sin cifrar: "+palabra);
        String palabraCifrada = Cypher.encrypt(palabra);
        System.out.println("palabra cifrada: "+palabraCifrada);
        String palabraDescifrada =  Cypher.decrypt(palabraCifrada);
        System.out.println("palabra descifrada: "+palabraDescifrada);

    }

}
```

Técnica 12:

Recomendaciones para la construcción de contraseñas seguras

Cada vez la generación de contraseñas se hace más compleja debido a que los sistemas de seguridad son más expuestos a la red y por ende los mecanismos de ataques informáticos son más sofisticados y frecuentes, por tal razón es pertinente seguir las siguientes recomendaciones

- Las contraseñas (tipo palabra) deben tener al menos 8 caracteres de longitud.
- Las contraseñas (tipo palabra) deben tener al menos 1 letra minúscula.
- Las contraseñas (tipo palabra) deben tener al menos 1 letra mayúscula.
- Las contraseñas (tipo palabra) deben tener al menos 1 dígito numérico.
- Las contraseñas (tipo palabra) deben tener al menos 1 carácter especial.

GLOSARIO

API: Application Programming Interface
APK: Android Application Package
DNS: Domain Name System
Endpoint: Interfaz donde los sistemas externos pueden enviar y recibir mensajes
GWT: Google Web Toolkit
IDE: Integrated Development Environment
IP: internet Protocol
JAR: Java archive
JSON: JavaScript Object Notation
LDAP: Lightweight Directory Access Protocol
OWASP: Open Web Application Security Project

REST: Representational State Transfer
RPC: Remote Procedure Call
SOAP: Simple Object Access Protoco
SQL: Structured Query Language
XML: Extensible Markup Language
XXE: XML Extenal Entity

Cibergrafía

https://www.owasp.org/index.php/Proyectos_OWASP

https://apiumhub.com/es/tech-blog-barcelona/beneficios-de-las-pruebas-unitarias/

https://anadreamy.wordpress.com/2012/05/30/apache-camel-que-es-un-endpoint/

https://si.ua.es/es/documentacion/c-sharp/documentos/pruebas/07pruebasunitarias.pdf

https://msdn.microsoft.com/es-es/communitydocs/alm/unit-test

http://www.jtech.ua.es/j2ee/publico/lja-2012-13/sesion04-apuntes.html

http://puntocomnoesunlenguaje.blogspot.com.co/2013/03/java-stringbuilder-stringbuffer.html

http://raulexposito.com/documentos/analisis-estatico-codigo/

http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/374

http://www.geoidep.gob.pe/metadatos/que-son-los-metadatos

https://losindestructibles.wordpress.com/2012/09/24/ataque-xss-cross-site-scripting/

http://www.dma.fi.upm.es/recursos/aplicaciones/matematica_discreta/web/aritmetica_modular/criptografia.html

https://www.redeszone.net/2010/11/04/criptografia-algoritmos-de-cifrado-de-clave-simetrica/

http://blog.capacityacademy.com/2013/08/16/seguridad-informatica-cifrado-simetrico-asimetrico-hashing/

https://www.owasp.org/index.php/Uso_de_las_Extensiones_Criptogr%C3%A1ficas_de_Java

http://revistas.uexternado.edu.co/index.php/propin/article/view/2476/3636