Máster Universitario en Investigación en Ingeniería de Software y Sistemas Informáticos

Analizador automático de vulnerabilidades en formularios con subida de ficheros

Trabajo Fin de Máster
Itinerario de Ingeniería del Software. Código de Asignatura: 31105151.

AUTOR: DAVID GARCÍA GONZÁLEZ
DIRECTOR: JOSÉ ANTONIO CERRADA SOMOLINOS

Curso Académico 2017-2018
Convocatoria Septiembre

Máster Universitario en Investigación en Ingeniería de Software y Sistemas Informáticos

ITINERARIO: Ingeniería de Software.

CÓDIGO DE ASIGNATURA: 31105151.

TÍTULO DEL TRABAJO: Analizador automático de vulnerabilidades en formularios con subida de ficheros

TIPO DE TRABAJO: Tipo A, proyecto específico propuesto por un profesor

AUTOR: DAVID GARCÍA GONZÁLEZ

DIRECTOR: JOSÉ ANTONIO CERRADA SOMOLINOS

HOJA DE CALIFICACIONES

DECLARACIÓN JURADA DE AUTORÍA DEL TRABAJO CIENTÍFICO, PARA LA DEFENSA DEL TRABAJO FIN DE MASTER

Fecha: 7/6/2018.

Quién suscribe:

Autor(a): **DAVID GARCÍA GONZÁLEZ**
D.N.I/N.I.E./Pasaporte.: **76948685-T**

Hace constar que es el autor(a) del trabajo:

Analizador automático de vulnerabilidades en formularios con subida de ficheros

En tal sentido, manifiesto la originalidad de la conceptualización del trabajo, interpretación de datos y la elaboración de las conclusiones, dejando establecido que aquellos aportes intelectuales de otros autores, se han referenciado debidamente en el texto de dicho trabajo.

DECLARACIÓN:

- ✓ Garantizo que el trabajo que remito es un documento original y no ha sido publicado, total ni parcialmente por otros autores, en soporte papel ni en formato digital.
- ✓ Certifico que he contribuido directamente al contenido intelectual de este manuscrito, a la génesis y análisis de sus datos, por lo cual estoy en condiciones de hacerme públicamente responsable de él.
- ✓ No he incurrido en fraude científico, plagio o vicios de autoría; en caso contrario, aceptaré las medidas disciplinarias sancionadoras que correspondan.

Fdo.

Autorización

Autorizo/amos a la Universidad Nacional de Educación a Distancia a difundir y utilizar, con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firma del/los Autor/es

Resumen

Los ataques a los sistemas informáticos están creciendo exponencialmente en estos tiempos. Los atacantes se aprovechan de las vulnerabilidades en estos sistemas para poder llevar a cabo su objetivo, que pasa desde el robo de información (ciberespionaje), la denegación del servicio e incluso la modificación o destrucción de la información.

Un punto de entrada a los sistemas de los que se aprovechan los hackers son los formularios web, de hecho, la mayoría del TOP de vulnerabilidades registradas por organizaciones como OWASP o CWE están relacionadas con formularios web: SQL Injection, XSS, etc.

El riesgo que corre una organización a publicar un formulario web que contenga vulnerabilidades es altísimo, como comentábamos antes, pasa desde el espionaje hasta la destrucción de la información de nuestros sistemas. Pero si además estos formularios permiten adjuntar ficheros entonces el riesgo que asumimos se multiplica. Estos ficheros entran directamente a nuestra organización, y sin tener un control minucioso sobre ellos, cualquier hacker nos podrá introducir en nuestra organización cualquier tipo de malware.

En este máster desarrollaremos una herramienta para analizar las vulnerabilidades relacionadas con la subida de ficheros de los formularios web para un marco de desarrollo completo. Con esta herramienta pretendemos minimizar el riesgo que asumimos al exponer un formulario con subida de ficheros en internet.

Palabras clave: vulnerabilidades, formularios web, ataques informáticos, hackers, espionaje.

Tabla de contenido

Tabla de contenido **7**

Tabla de ilustraciones **10**

1 Introducción **12**

2 Objetivos **15**

3 Estado del arte en la materia **16**

3.1 Vulnerabilidades relacionadas con subida de ficheros en formularios web **16**

3.1.1 CWE-434: Unrestricted file Upload with Dangerous type 16

3.1.2 CWE-400: Uncontrolled Resource Consumption ('Resource Exhaustion') 17

3.1.3 CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') 18

3.2 Soluciones existentes **18**

4 Descripción de la propuesta original realizada **22**

4.1 Desarrollo de una librería de validación de ficheros **22**

4.1.1 Validación del tipo de fichero 22

4.1.2 Validación del nombre del fichero 24

4.1.3 Validación de la longitud del nombre del fichero 25

4.1.4 Limitación del tamaño del fichero 25

4.1.5 Limitar la subida de ficheros a usuarios autenticados y autorizados 25

4.1.6 Detección de código malicioso incrustado en ficheros PDF 26

4.2 Desarrollo de una herramienta de validación de código fuente **27**

4.2.1 Desarrollo del plugin para SpotBugs 28

4.2.2 Integración de la herramienta en un entorno de desarrollo. 29

4.2.3 Integración de la herramienta en una plataforma de evaluación de calidad de código fuente. 33

4.3 Desarrollo de una herramienta de auditoría **35**

4.4 Ventajas e inconvenientes de la solución propuesta frente a otras soluciones existentes **36**

5 Experimentación y contrastación de la propuesta **38**

5.1 Experimentación de vulnerabilidades sobre el formulario no seguro **42**

5.1.1 CWE-434: Unrestricted file Upload with Dangerous type 42

5.1.2 CWE-400: Uncontrolled Resource Consumption ('Resource Exhaustion') 50

5.1.3 CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') 53

5.2 Experimentación de vulnerabilidades sobre el formulario seguro **57**

5.2.1 CWE-434: Unrestricted file Upload with Dangerous type 59

5.2.2 CWE-400: Uncontrolled Resource Consumption ('Resource Exhaustion') 64

5.2.3 CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') ... 66
5.3 Experimentación de la herramienta de auditoría ... 69
5.4 Experimentación de la herramienta de validación de código fuente ... 72
6 Conclusiones y trabajos futuros ... 76
7 Bibliografía ... 78
8 Anexos ... 80
8.1 Código fuente de la librería de validación ... 80
8.1.1 POM.XML ... 80
8.1.2 AuditoriaFicherosDao.java ... 82
8.1.3 AuditoriaFicherosDaoImpl ... 83
8.1.4 DocumentDetector.java ... 85
8.1.5 DocumentDetectorFactory.java ... 85
8.1.6 BasicDocumentDetectorImpl.java ... 86
8.1.7 PdfDocumentDetectorImpl.java ... 87
8.1.8 InsecureFileUploadedException.java ... 88
8.1.9 AuditoriaFicheros.java ... 88
8.1.10 AuditoriaFicherosService.java ... 92
8.1.11 AuditoriaFicherosServiceImpl.java ... 92
8.1.12 ErrorValidacion.java ... 93
8.1.13 TipoFichero.java ... 94
8.1.14 UploadedFileValidator.java ... 95
8.1.15 UploadedFileValidatorImpl.java ... 95
8.1.16 BusquedaAuditoriaFicherosVO.java ... 101
8.1.17 orm.xml ... 102
8.1.18 secureapi-persistence.xml ... 103
8.1.19 datasource-context.xml ... 103
8.1.20 service-context.xml ... 104
8.1.21 secureapi-context.xml ... 105
8.2 Código fuente del plugin de Spotbugs ... 105
8.2.1 pom.xml ... 106
8.2.2 InsecureFileUploadDetector.java ... 108
8.2.3 findbugs.xml ... 110
8.2.4 messages.xml ... 110
8.2.5 InsecureFileUploadDetectorTest.java ... 111
8.2.6 FileUploadFormBadCase.java ... 112
8.2.7 FileUploadFormGoodCase.java ... 113
8.3 Código fuente del prototipo con un formulario seguro y un formulario inseguro 115
8.3.1 pom.xml ... 115
8.3.2 AppUser.java ... 117
8.3.3 UserDao.java ... 118

8.3.4 UserDaoImpl.java 119
8.3.5 UserDetailsServiceImpl.java 119
8.3.6 BusquedaAuditoriaFicherosForm.java 120
8.3.7 FileUploadForm.java 122
8.3.8 LoginForm.java 123
8.3.9 SecureFileUploadForm.java 124
8.3.10 security-web-context.xml 127
8.3.11 web-context.xml 130
8.3.12 commons-logging.properties 131
8.3.13 log4j.properties 131
8.3.14 faces-config.xml 131
8.3.15 web.xml 133
8.3.16 busquedaAuditoriaFicheros.jsp 138
8.3.17 fileupload_showimg.jsp 142
8.3.18 fileupload.jsp 143
8.3.19 index.jsp 144
8.3.20 login.jsp 145
8.3.21 securefileupload.jsp 146

Tabla de ilustraciones

Ilustración 1 Evolución de los ciberataques gestionados por el CCN-CERT 12
Ilustración 2 Vulnerabilidades publicadas en el portal del CCN-CERT 13
Ilustración 3 WSO WebShell. Permite la navegación por el sistema de ficheros y ejecución de comandos desde una consola 17
Ilustración 4: Javadoc del método de validación de ficheros de la librería ESAPI 19
Ilustración 5: Logo de la aplicación SpotBugs 28
Ilustración 6: Instalación de SpotBugs en Eclipse 30
Ilustración 7: Configuración de nuestro plugin de SpotBugs en Eclipse 31
Ilustración 8: Opciones de menú de SpotBugs 32
Ilustración 9: Perspectiva de SpotBugs 33
Ilustración 10: Logo de la herramienta SonarQube 34
Ilustración 11: Visión general de SonarQube 34
Ilustración 12: Prototipo de la aplicación de consulta de trazas de auditoría 35
Ilustración 13: Pantalla principal del prototipo de pruebas 39
Ilustración 14: Acceso al formulario inseguro 40
Ilustración 15: Selección de una imagen de nuestro equipo 40
Ilustración 16: Resultado del envío del formulario con la imagen al servidor 41
Ilustración 17: Selección de un fichero con la extensión renombrada 42
Ilustración 18: Contenido del directorio donde se almacenan los ficheros en nuestro servidor 43
Ilustración 19: Resultado de envío de malware ejecutable con extensión renombrada 44
Ilustración 20: Configuración del proxy en OWASP ZAP 45
Ilustración 21: Configuración del navegador Mozilla Firefox para establecer el proxy a la aplicación OWASP ZAP 46
Ilustración 22: Selección del fichero malware.jpg, que en realidad es un .exe 47
Ilustración 23: Edición de la petición HTTP en OWASP ZAP 48
Ilustración 24: Renombrado del cuerpo de la petición HTTP en OWASP ZAP 49
Ilustración 25: Resultado del envío de un fichero ejecutable editando la petición HTTP con una herramienta proxy 50
Ilustración 26: Selección de un fichero de 102MB 51
Ilustración 27: Subida de un fichero de 100MB 52
Ilustración 28: Fichero de 100MB en el directorio del servidor 53
Ilustración 29: Selección de un fichero desde el formulario inseguro 54
Ilustración 30: Captura de la petición HTTP con una herramienta proxy 55
Ilustración 31: Modificación del parámetro filename de la petición para cambiar la ubicación 55
Ilustración 32: Envío de un fichero con una relativa en su nombre 56
Ilustración 33: Listado del directorio /tmp para comprobar que el fichero se ha escrito en un directorio donde no debería 57
Ilustración 34: Pantalla de autenticación para poder acceder al formulario seguro 58
Ilustración 35: Acceso al formulario seguro 59
Ilustración 36: Selección de un fichero con la extensión renombrada 60
Ilustración 37: Mensajes de error de validación al intentar adjuntar un fichero que no cumple las condiciones de validez 61

ILUSTRACIÓN 38: SELECCIÓN DEL FICHERO MALWARE.JPG, QUE EN REALIDAD ES UN .EXE 62
ILUSTRACIÓN 39: RENOMBRAMOS EL NOMBRE DEL FICHERO CON UN PROXY WEB 63
ILUSTRACIÓN 40: ERROR DE VALIDACIÓN DE UN FICHERO EJECUTABLE 64
ILUSTRACIÓN 41: SELECCIÓN DE UNA IMAGEN DE 17MB 65
ILUSTRACIÓN 42: MENSAJE DE ERROR TAMAÑO DE FICHERO EXCEDIDO 66
ILUSTRACIÓN 43: CAPTURA DE UNA PETICIÓN HTTP PARA EDITAR EL NOMBRE DEL FICHERO 67
ILUSTRACIÓN 44: RENOMBRADO DEL NOMBRE DEL FICHERO EN LA HERRAMIENTA PROXY 68
ILUSTRACIÓN 45: ERROR QUE INDICA QUE EL NOMBRE DEL FICHERO NO CUMPLE LAS CONDICIONES DE VALIDEZ 69
ILUSTRACIÓN 46: BÚSQUEDA DE LOS ÚLTIMOS DOCUMENTOS QUE NO HAN SIDO VALIDADOS SATISFACTORIAMENTE 70
ILUSTRACIÓN 47: TRAZA DE AUDITORÍA EN LA QUE SE PUEDE OBSERVAR UN FICHERO CUYA EXTENSIÓN NO SE CORRESPONDE CON EL TIPO MIME ESPERADO 71
ILUSTRACIÓN 48: TRAZAS DE AUDITORÍA DE FICHEROS SUBIDOS SIN NINGÚN ERROR DE VALIDACIÓN 71
ILUSTRACIÓN 49: OPCIÓN DE MENÚ PARA ANALIZAR NUESTRO PROYECTO EN BÚSQUEDA DE VULNERABILIDADES 73
ILUSTRACIÓN 50: ERRORES DE VALIDACIÓN EN NUESTRO PROYECTO PROTOTIPO 73
ILUSTRACIÓN 51: MARCA DE ERROR DE VALIDACIÓN EN UN FICHERO FUENTE DE NUESTRO PROYECTO 74
ILUSTRACIÓN 52: DESCRIPCIÓN DETALLADA DEL ERROR 74

1 Introducción

Según un informe del CCN-CERT (Centro Criptológico Nacional) de 2016 [1] , se espera un incremento del 40% en los ciberataques a la Administración y a empresas de interés estratégico. En el siguiente gráfico podemos observar cómo desde el año 2009 el número de ciberataques gestionados por este centro no ha parado de crecer. De hecho, del año 2014 al 2016 el número de ciberataques ha crecido un 40%.

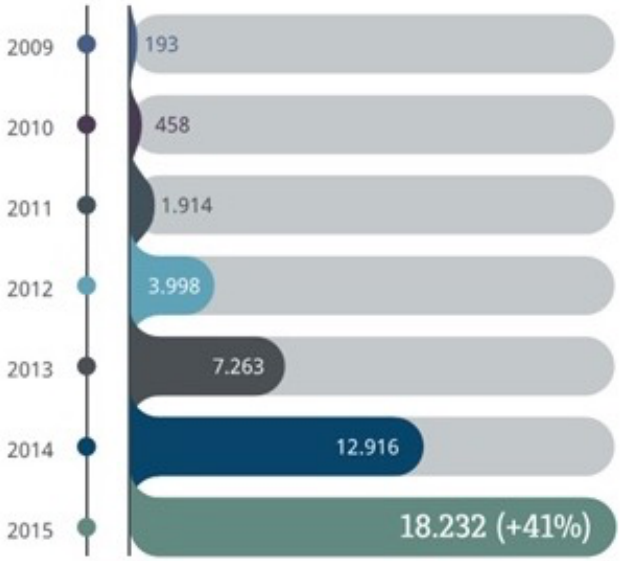

Ilustración 1 Evolución de los ciberataques gestionados por el CCN-CERT

Para que el ataque tenga éxito, los hackers necesitan aprovecharse de una vulnerabilidad en el sistema, por lo tanto, si minimizamos el número de vulnerabilidades presentes en nuestros sistemas, minimizamos también el riesgo de sufrir un ataque. Esta conclusión, aunque parece básica, según el mismo informe del CCN-CERT, las vulnerabilidades software han crecido en los últimos años.

[1] Informe CCN-CERT IA-09/16 de Ciberamenazas 2015/Tendencias 2016.
https://www.ccn-cert.cni.es/informes/informes-ccn-cert-publicos/1483-ccn-cert-ia-0916-ciberamenazas-2015-tendencias-2016-resumen-ejecutivo/file.html

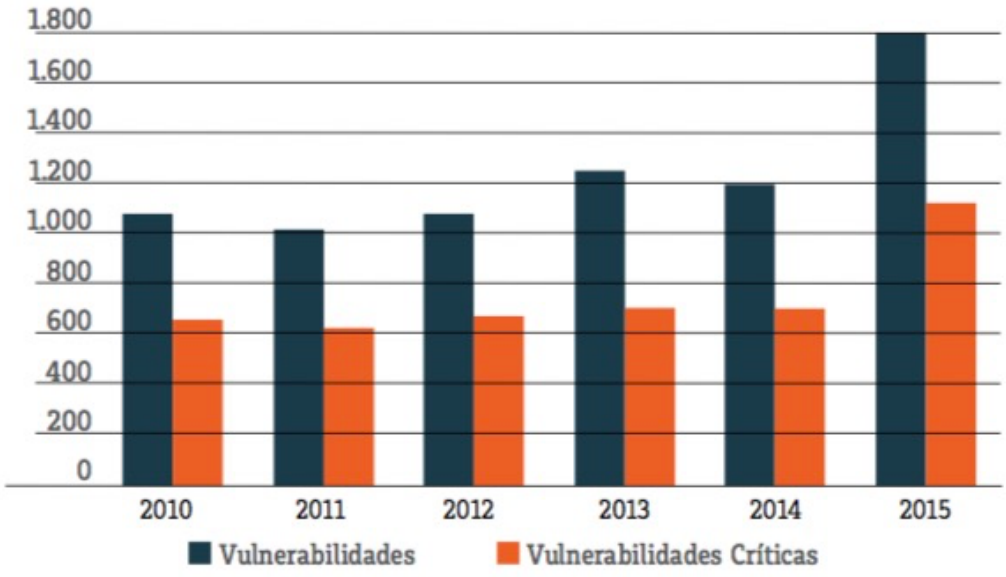

Ilustración 2 Vulnerabilidades publicadas en el portal del CCN-CERT

Un elemento software donde se suelen acumular las vulnerabilidades son los formularios web. Según organizaciones como OWASP o CWE, la mayoría de vulnerabilidades más graves están relacionadas con la entrada de información al sistema por medio de formularios web: Injection, Cross-Site Scripting (XSS), etc.

Si además, en el formulario permitimos a cualquier usuario subir un fichero a nuestros servidores entonces estamos introduciendo un riesgo mucho más elevado a toda nuestra organización ya que si este formulario tiene vulnerabilidades, un atacante podría comprometer nuestro sistema.

Para recalcar la seriedad del problema, es necesario destacar que en el momento de redactar esta memoria, tenemos 121 entradas en la base de datos de vulnerabilidades CVEDetails (Common Vulnerabilities and Exposures)[2] relacionadas con la vulnerabilidad "CWE-434: Unrestricted Upload of File with Dangerous Type"[3].

[2] Security Vulnerabilities Related To CWE-434.
https://www.cvedetails.com/vulnerability-list/cweid-434/vulnerabilities.html

[3] CWE-434: Unrestricted Upload of File with Dangerous Type.
https://cwe.mitre.org/data/definitions/434.html

Para poder ofrecer la funcionalidad de subir ficheros de manera segura, debemos de validar minuciosamente las entradas y desarrollar formularios seguros.

En los próximos apartados detallaremos las vulnerabilidades más conocidas relacionadas con subida de ficheros en formularios web y desarrollaremos una librería para JSF (Java Server Faces) que nos permita publicar formularios seguros.

2 Objetivos

Partiendo del concepto de seguridad perimetral, nuestro objetivo será añadir múltiples capas de defensa a nuestros sistemas para poder publicar formularios seguros y minimizar de esta forma las posibilidades de que un atacante pueda conseguir sus objetivos.

El primer paso será realizar un estudio de las vulnerabilidades y amenazas más comunes a las que nos enfrentamos relacionadas con la subida de ficheros en formularios web.

Una vez tengamos claras estas vulnerabilidades y las amenazas a las que estaríamos expuestos, desarrollaremos una herramienta para analizar y detectar automáticamente formularios que contengan algún tipo de vulnerabilidad relacionada con los ficheros.

El siguiente paso será encontrar una solución para eliminar estas vulnerabilidades, que en nuestro caso consistirá en el desarrollo de una librería que se encargue de realizar las validaciones necesarias para asegurar que el fichero que nos envían es válido.

Por último, necesitaremos analizar periódicamente el código de las aplicaciones web para detectar si se ha desarrollado un nuevo formulario sin haber aplicado las soluciones necesarias para eliminar las vulnerabilidades encontradas.

Para fijar el ámbito del proyecto, esta herramienta se centraría en analizar el código dentro de la arquitectura software de nuestra organización, que consiste en aplicaciones web J2EE desarrolladas con Java Server Faces. Por lo tanto, quedaría fuera del proyecto la validación del código para otros lenguajes de programación o marcos de desarrollo como por ejemplo Struts o Spring MVC.

3 Estado del arte en la materia

A continuación, vamos a analizar las vulnerabilidades más comunes relacionadas con la subida de ficheros en formularios web. Después vamos a estudiar las soluciones recomendadas para eliminarlas.

3.1 Vulnerabilidades relacionadas con subida de ficheros en formularios web

Según el CWE (Common Weakness Enumeration), en el momento de la redacción de esta memoria las vulnerabilidades relacionadas con la subida de ficheros en formularios web más comunes son las siguientes:

3.1.1 CWE-434: Unrestricted file Upload with Dangerous type

Un sistema con esta vulnerabilidad permitiría a un atacante la subida de un fichero de un tipo que no se correspondería con el tipo de fichero esperado por la aplicación.

Para explicar la gravedad de esta vulnerabilidad, vamos a mostrar una serie de posibles ataques:

Secuestro de la información

En un formulario en el que esperamos que se adjunte un documento PDF, el atacante consigue enviarnos un fichero ejecutable con malware. Este fichero con malware entraría en nuestra organización y cualquier persona lo podría ejecutar, comprometiendo de esta forma a toda nuestra organización.

Imaginemos, por ejemplo, que el fichero que nos envían es un ransomware[4]. **¡Podrían cifrarnos los equipos de toda nuestra organización!**

Ciberespionaje

Si lo que pretenden es el ciberespionaje, podrían enviarnos un software para poder acceder a nuestro entorno de manera remota (RAT: Remote Access Trojan).

Instalación de un WebShell en nuestro sistema

[4] Ransomware: Encripta todos los ficheros del ordenador y nos piden el pago de una cantidad para enviarnos la clave que desencripta los ficheros.

Si nuestra aplicación permite la subida de ficheros en un directorio que sea accesible desde el servidor web entonces podríamos ser víctimas de la instalación de un WebShell.

Un WebShell es una aplicación que nos instalan en el servidor web y permite en remoto ejecutar instrucciones en nuestro sistema, navegar por nuestro sistema ficheros, etc.

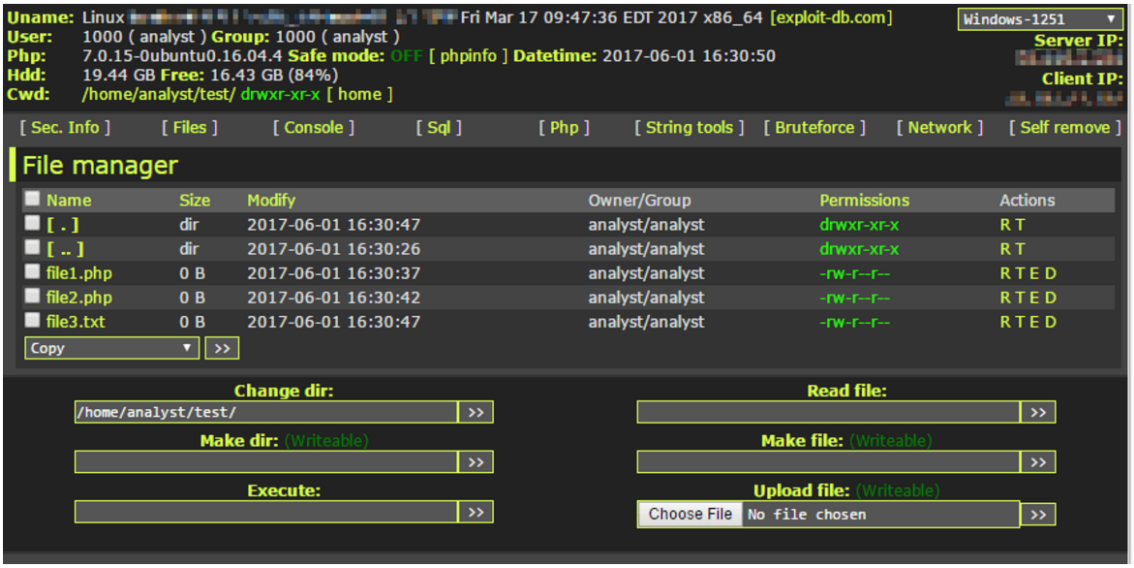

Ilustración 3 WSO WebShell. Permite la navegación por el sistema de ficheros y ejecución de comandos desde una consola

Este ataque también está relacionado con la vulnerabilidad "CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')" que describiremos más adelante.

3.1.2 CWE-400: Uncontrolled Resource Consumption ('Resource Exhaustion')

La aplicación no limita adecuadamente la cantidad de recursos que utiliza un usuario, lo que le permite consumir más recursos de los necesarios.

Si nuestra aplicación contiene un formulario con subida de ficheros y no verificamos el tamaño de los ficheros que recibimos, un atacante sería capaz de adjuntarnos ficheros de un tamaño enorme, si además tampoco limitamos el número de peticiones que éste nos puede solicitar, entonces podríamos sufrir un ataque de **Denegación de Servicio** (**DOS**). Permitiríamos saturar el espacio libre de disco de nuestros sistemas, lo que daría lugar a múltiples errores en el sistema operativo y dejando al sistema exhausto. Finalmente, la aplicación dejaría de funcionar.

3.1.3 CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

Si nuestra aplicación utiliza un parámetro de entrada para construir la ruta en la que se almacenará un fichero y no validamos correctamente el parámetro de entrada, entonces podríamos permitir a un atacante construir una ruta a un directorio no permitido.

Suele ser muy habitual almacenar el fichero en un directorio del sistema utilizando el mismo nombre del fichero recibido, por ejemplo, recibimos el fichero "documento1.pdf", y en nuestro sistema lo almacenamos en `"/tmp/ficheros/documento1.pdf".`

Pero imaginemos que nos adjuntan un fichero con el siguiente nombre:

`../../../etc/passwd`

Si no validamos correctamente el nombre del fichero, podrían llegar a **sobrescribir ficheros protegidos del sistema operativo**.

3.2 Soluciones existentes

Aunque parezca sorprendente, no hemos encontrado ninguna implementación en Java para la validación completa de formularios con ficheros.

El famoso proyecto para el desarrollo de librerías de seguridad OWASP ESAPI[5] solo contiene funcionalidades que realizan una validación muy básica de los ficheros: solo comprueba el nombre, el tamaño y la extensión.

assertValidFileUpload

```
public void assertValidFileUpload(java.lang.String context,
                                  java.lang.String directorypath,
                                  java.lang.String filename,
                                  java.io.File parent,
                                  byte[] content,
                                  int maxBytes,
                                  java.util.List<java.lang.String> allowedExtensions,
                                  boolean allowNull)
                           throws ValidationException,
                                  IntrusionException
```

Validates the filepath, filename, and content of a file. Invalid input will generate a descriptive ValidationException, and input that is clearly an attack will generate a descriptive IntrusionException.

Specified by:

assertValidFileUpload in interface Validator

Parameters:

context - A descriptive name of the parameter that you are validating (e.g., LoginPage_UsernameField). This value is used by any logging or error handling that is done with respect to the value passed in.
directorypath - The file path of the uploaded file.
filename - The filename of the uploaded file
content - A byte array containing the content of the uploaded file.
maxBytes - The max number of bytes allowed for a legal file upload.
allowNull - If allowNull is true then an input that is NULL or an empty string will be legal. If allowNull is false then NULL or an empty String will throw a ValidationException.

Throws:

ValidationException
IntrusionException

Ilustración 4: Javadoc del método de validación de ficheros de la librería ESAPI

El problema de esta librería es que valida el tipo del fichero a partir de la extensión y no realiza una validación real a partir del contenido del fichero.

Hemos encontrado un hilo interesante en el proyecto de OWASP en el que varios miembros del proyecto discutían si sería interesante ampliar la librería para añadir validación del contenido del fichero. La conclusión a la que habían llegado fue que sería

[5] ESPAI: Proyecto OWASP para implementar una API de Seguridad Empresarial. https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API/es

prácticamente imposible implementar una función que validara correctamente todos los tipos de ficheros. Varios de los integrantes del hilo votaban por no implementar dicha validación debido a la gran complejidad a la que se habría que enfrentar. Además, recomienda que cada empresa se implemente su versión de la librería acorde a sus necesidades.

He aquí dos entradas del hilo muy interesantes:

Hilo textual:

> *"My position for safe upload is that you go all the way and do it right or disable the feature. For esapi to give a partial solution in the ref impl is dangerous, in my highly opinionated opinion, cause of how vulnerable it is.*
>
> *..*
>
> *Again, just my opinion. File upload is brutal* ***- one of the difficult parts of web app sec****."*
>
> *Jim Manico jim.manico at owasp.org - Sat May 16 19:53:54 EDT 2009.*

Traducción:

> *"Mi postura para la subida segura es que hagamos toda la implementación y lo hagamos bien o deshabilitamos la funcionalidad. Es peligroso para esapi ofrecer una solución parcial, en mi juiciosa opinión, debido a lo vulnerable que es.*
>
> *..*
>
> *De Nuevo, solo mi opinión. La subida de ficheros es brutal – una de las partes más difíciles en seguridad de aplicaciones web."*
>
> *Jim Manico jim.manico at owasp.org - Sat May 16 19:53:54 EDT 2009.*

Otro mensaje del hilo interesante:

> *"****File content validation is an intractable problem*** *most of the time. Virii signatures are easily bypassable and most file formats can have dangerous content by design.*
>
> *This is not to mention that every file format is different and the majority of time malicious intentions can be expressed in many ways.*

> *My vote is not to attempt to "solve" this problem. Businesses may be able to implement a version of this API that validates batch records or something, but the spec should denote the limited scope."*
>
> *Arshan Dabirsiaghi arshan.dabirsiaghi at aspectsecurity.com Sat May 16 20:56:16 EDT 2009.*

Traducción:

> *"**La validación del contenido del fichero es un tema intratable** la mayor parte del tiempo. Las firmas de los virus son fáciles de saltar y la mayoría de formatos de fichero por diseño pueden tener contenido peligroso.*
>
> *Esto sin mencionar que cada formato de archivo es diferente y la mayoría de veces las intenciones maliciosas se pueden expresar de muchas maneras.*
>
> ***Mi voto es no intentar resolver este problema. Las empresas pueden implementar una versión de esta API** que valide los registros por lotes o algo así, pero la especificación debería de denotar un alcance limitado."*
>
> *Arshan Dabirsiaghi arshan.dabirsiaghi at aspectsecurity.com Sat May 16 20:56:16 EDT 2009.*

Puede leerse el hilo completo en la siguiente página web: https://lists.owasp.org/pipermail/owasp-esapi/2009-May/000527.html

A parte de esta librería, solo hemos encontrado algún proyecto personal en GitHub, pero en ninguno de ellos había una organización importante detrás como podría ser OWASP, CWE, Apache, etc.

4 Descripción de la propuesta original realizada

Nuestra estrategia para minimizar las vulnerabilidades relacionadas con recepción de ficheros se basará en 3 acciones:

1. Desarrollo de una librería Java para JSF que realice las validaciones necesarias en los ficheros recibidos. Cualquier fichero que no supere la validación no será aceptado y mostraremos un error al usuario.
2. Desarrollo de una herramienta que analice el código fuente de nuestras aplicaciones web y verifique automáticamente si tenemos algún formulario web que presente vulnerabilidades relacionadas con subida de ficheros.
3. Desarrollo de una herramienta de auditoría en la que se almacenará una traza por cada fichero recibido. Esta auditoría contendrá información útil en caso de que fallen todas las defensas y recibamos un fichero con malware. Si esto sucediera, esta información nos permitiría estudiar el ataque y mejorar nuestra herramienta. También serviría para investigar la autoría del ataque.

En los siguientes apartados comentaremos con más detalle cada punto.

4.1 Desarrollo de una librería de validación de ficheros

Desarrollaremos una única librería para todas las webs implementadas en JSF en nuestra organización. Esta librería se encargará de realizar una serie de validaciones en los ficheros para mitigar las vulnerabilidades relacionadas.

Las comprobaciones que realizará esta librería será las siguientes:

4.1.1 Validación del tipo de fichero

El tipo de fichero esperado por la aplicación se debe de corresponder realmente con el tipo de fichero recibido.

Las siguientes validaciones se suelen realizar de manera común para intentar verificar que un documento es del tipo esperado, pero como demostraremos a continuación estas soluciones no son del todo seguras, ya que un atacante tiene mecanismos para saltarse estas validaciones:

4.1.1.1 Solución vulnerable: Comprobación del tipo de fichero por la extensión

Es muy común actualmente para verificar que recibimos un documento de un tipo concreto comprobar solo la extensión de su nombre.

Pero no basta solo con analizar la extensión del fichero. Alguien podría renombrar un fichero ejecutable (".exe" o ".sh") a una extensión típica de una imagen, por ejemplo ".jpg" para burlar la validación.

4.1.1.2 Solución vulnerable: Utilizar el atributo "accept" del formulario HTML

La etiqueta `<form>` de HTML permite especificar el tipo de fichero que acepta el servidor con el atributo "`accept`".

Por ejemplo, el siguiente formulario solo aceptaría ficheros de tipo imagen.

```
<form action="/action_page">
      <input type="file" name="pic" accept="image/*">
      <input type="submit">
</form>
```

El problema con esta solución es que, aunque el navegador no permitiría seleccionar un fichero que no sea una imagen, el atacante podría utilizar una herramienta tipo proxy para interceptar y modificar la petición HTTP y cambiar el nombre del fichero antes de enviarla al servidor.

4.1.1.3 Solución válida: Comprobar el tipo de fichero analizando su contenido

La solución más segura actualmente consiste en determinar el tipo de fichero analizando su contenido. Nuestra herramienta utilizará la librería Apache Tika[6] para detectar y extraer metadatos de los documentos que nos indiquen el tipo de contenido real.

4.1.2 Validación del nombre del fichero

Todos los caracteres de escape y de control deben de ser eliminados del nombre del fichero y de su extensión.

Debemos evitar recibir nombre de fichero como por ejemplo: "../etc/passwd" y que puedan escribir el fichero fuera del directorio habilitado para ello.

4.1.2.1 Solución vulnerable: Comprobar el nombre del fichero con una lista negra

Algunas aplicaciones utilizan solo una lista negra para validar el nombre del fichero. El problema es que las listas negras tienen una serie de inconvenientes que permiten a un usuario saltarse la validación:

1. La lista podría ser muy extensa y difícil de mantener, por lo tanto, puede que falten algunos valores que no deberían de ser permitidos.
2. A veces es posible saltarse la validación cambiando letras de mayúsculas a minúsculas.
3. Es posible saltarse la validación si utilizamos caracteres de control como por ejemplo el carácter nulo (0x00) al final del nombre prohibido.

4.1.2.2 Solución válida: Comprobar el nombre del fichero con una lista blanca

En nuestro caso utilizaremos una expresión regular, y solo aceptaremos caracteres alfanuméricos y un único punto "." para separar el nombre de su extensión.

Nuestra expresión regular sería algo así:

---

[6] Apache Tika: Librería de Apache para detectar y extraer texto y metadatos de miles de tipos de ficheros. http://tika.apache.org/index.html

```
[a-zA-Z0-9]{1,200}\.[a-zA-Z0-9]{1,10}
```

4.1.3 Validación de la longitud del nombre del fichero

No podemos permitir introducir nombres de fichero de cualquier longitud. Por ejemplo, para una partición NTFS, la longitud máxima del nombre de un fichero (incluyendo su extensión) son 255 caracteres. Si nos adjuntasen ficheros con nombres de más de 255 caracteres daría lugar a errores inesperados que un atacante podría aprovechar.

4.1.3.1 Solución válida: Comprobar la longitud del nombre del fichero

La solución es sencilla, si el nombre del fichero tiene una longitud mayor de 255 caracteres se devolverá un error.

4.1.4 Limitación del tamaño del fichero

Debemos de limitar también el tamaño máximo de fichero. Si aceptamos ficheros enormes podríamos sufrir ataques de denegación de servicio.

La solución es sencilla, basta con configurar un tamaño máximo y validar que el tamaño del fichero no supere dicho tamaño. En caso de que lo supere se devolverá un error.

4.1.5 Limitar la subida de ficheros a usuarios autenticados y autorizados

En la medida de lo posible debemos de limitar la funcionalidad de subida de ficheros a usuarios que estén autenticados y autorizados a ello.

4.1.5.1 Solución válida: Proteger el formulario en cuestión con autenticación y autorización

Debemos de evitar que sea posible acceder al formulario a un usuario sin haber sido autenticado y autorizado satisfactoriamente.

A veces no es posible, ya que podríamos querer formularios accesibles a usuarios sin solicitarles un registro previo, pero al menos debemos de minimizar estos tipos de formularios.

La librería de validación podría verificar que le pasamos el nombre de usuario autenticado que ha accedido al formulario.

Un dato importante a almacenar en las trazas de auditoría sería el nombre de usuario, de esta manera, en caso de tener que investigar un futuro ataque, dispondríamos de más información para intentar averiguar la autoría del ataque.

4.1.6 Detección de código malicioso incrustado en ficheros PDF

El formato PDF[7] tiene la característica de que puede contener ficheros incrustados y código JavaScript dentro del PDF. Esta característica le da mucha funcionalidad a este formato, pero tiene como gran inconveniente que un fichero en PDF puede contener código malicioso incrustado en su interior.

Debemos de añadir una validación a nuestra librería para que detecte si un PDF tiene ficheros incrustados o contiene ejecución de código JavaScript. En este caso, debería de mostrar un error y no permitir subir el fichero a nuestro servidor.

Una mejora que se ha estudiado ha sido intentar analizar los ficheros y el código JavaScript que pudiera contener el PDF, ya que no todos los ficheros PDF incluirán código malicioso. El problema con el que nos hemos encontrado es que sería necesario implementar prácticamente un antivirus para poder diferenciar correctamente si un PDF

[7] PDF: Portable Document Format. Formato de fichero desarrollado por Adobe para intercambio de documentos multiplataforma. Desde el año 2008 es un formato abierto. Para más información consultar la web de Adobe: https://acrobat.adobe.com/es/es/acrobat/about-adobe-pdf.html

contiene código malicioso o no. Debido a la gran complejidad hemos descartado implementar esta solución.

4.2 Desarrollo de una herramienta de validación de código fuente

El segundo paso en nuestra estrategia para mitigar las vulnerabilidades relacionadas con subida de ficheros consiste en una herramienta que analice nuestro código fuente y nos avise si ha encontrado un formulario vulnerable.

La idea de esta herramienta es que pueda funcionar de dos maneras: integrada en un entorno de desarrollo e integrada en una plataforma de evaluación de calidad del código fuente. Más adelante explicaremos en más detalle cada opción.

Para desarrollar el analizador de código estático hemos optado por implementar un nuevo plugin para SpotBugs[8]. SpotBugs es un programa que utiliza un analizador de código estático para encontrar errores en el código. Este programa de base ya implementa múltiples análisis para encontrar más de 400 errores comunes en Java. Pero además se puede extender para poder añadir nuestras propias reglas. Este programa se integra perfectamente con Eclipse[9], y con herramientas de compilación, empaquetado como ANT [10] y Maven[11].

[8] SpotBugs: Programa que utiliza un analizador de código estático en busca de errores en Java. https://spotbugs.github.io/

[9] Eclipse IDE: Entorno de desarrollo muy común para el lenguaje Java, aunque también tiene soporte para más lenguajes. https://www.eclipse.org/ide/

[10] Apache ANT: Herramienta utiliza para construir aplicaciones Java. https://ant.apache.org/

[11] Apache Maven: Herramienta para gestionar la construcción de aplicaciones Java. https://maven.apache.org/

Ilustración 5: Logo de la aplicación SpotBugs

SpotBugs tiene licencia GNU LGPL (Licencia Pública General Reducida de GNU), por lo que nos permite utilizar la herramienta en programas libres y programas privados. Por lo tanto, no tenemos ningún problema en cuanto a la licencia de este programa.

4.2.1 Desarrollo del plugin para SpotBugs

El funcionamiento de SpotBugs se basa en analizar el bytecode de clases Java en busca de errores. Se suele desarrollar una clase por cada error que queramos detectar, y esta clase analizará las clases Java compiladas, sus métodos y sus atributos, y en el caso de que encontremos algún patrón podemos reportar el error.

El desarrollo del plugin de SpotBugs consiste básicamente en desarrollar una librería Java siguiendo varios pasos:

1. Implementar nuestro detector del error. Para ello debemos de escribir una clase Java que implemente una determina interfaz (`edu.umd.cs.findbugs.Detector`) de SpotBugs.

```
public class InsecureFileUploadDetector implements Detector {

   @Override
   public void visitClassContext(ClassContext classContext) {
   …
   }

   @Override
   public void report() {
   …
   }

}
```

2. Escribir un caso de prueba con JUnit para asegurarnos que nuestro plugin encuentra el error. Debemos de escribir también casos de prueba para asegurarnos que no nos encuentra falsos positivos.
3. Escribir varios ficheros de configuración en XML que describen nuestro detector de errores.

Una vez tengamos la librería Java con el analizador ya podemos integrarla en nuestro entorno de desarrollo y en nuestra plataforma de evaluación de calidad de código fuente.

4.2.2 Integración de la herramienta en un entorno de desarrollo.

Es interesante que se encuentre la vulnerabilidad lo antes posible, por lo tanto, sería ideal que la herramienta avise al desarrollador mientras está programando si el código que acaba de escribir presenta alguna vulnerabilidad de este tipo. De esta manera el desarrollador solucionaría la vulnerabilidad inmediatamente. En nuestro caso haremos la integración en el entorno de desarrollo Eclipse.

El primer paso será instalar SpotBugs desde Eclipse Marketplace:

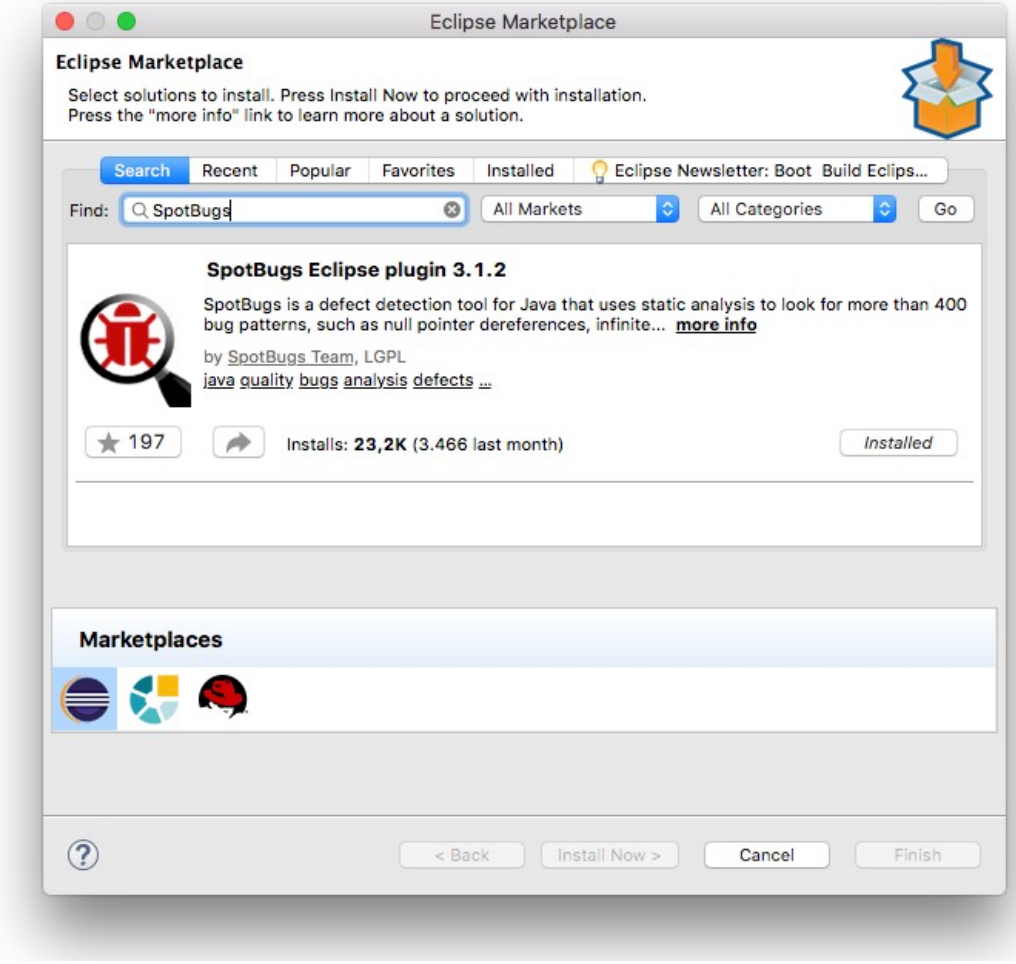

Ilustración 6: Instalación de SpotBugs en Eclipse

Una vez lo tengamos instalado, debemos de configurar SpotBugs para añadir el plugin que implementa nuestra librería:

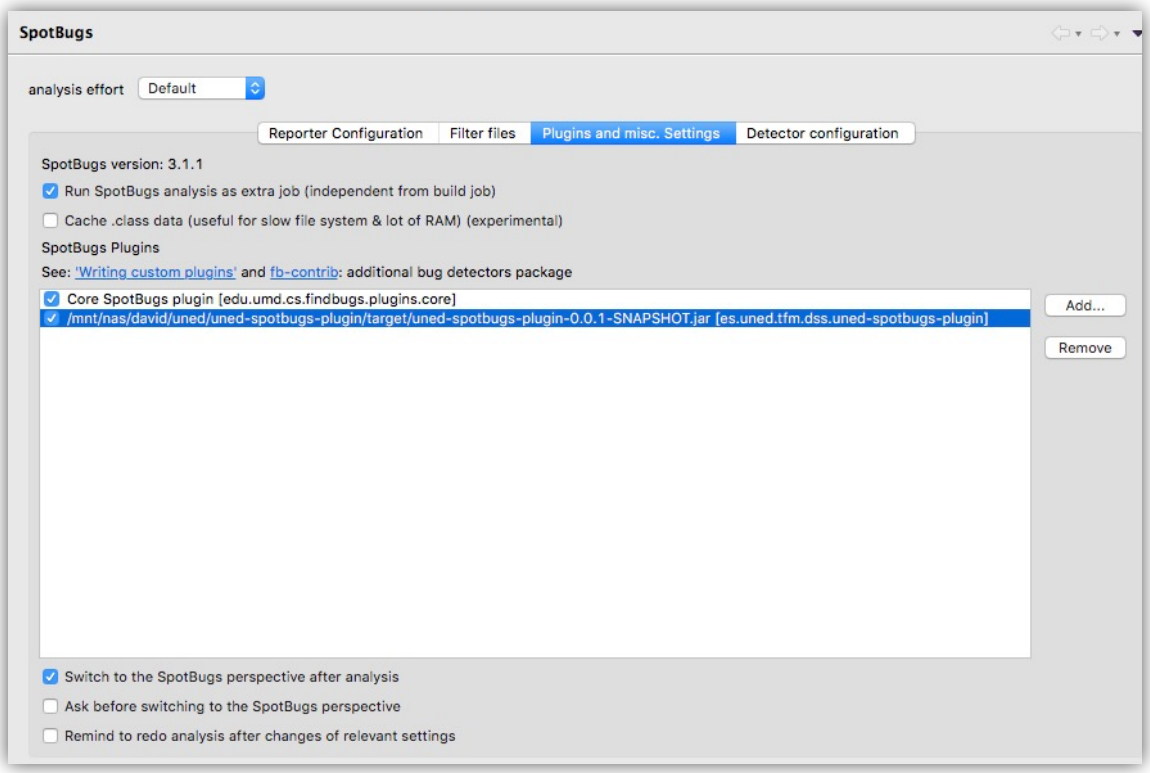

Ilustración 7: Configuración de nuestro plugin de SpotBugs en Eclipse

Una vez tengamos instalado correctamente ya tendremos la opción disponible para buscar errores en nuestro código.

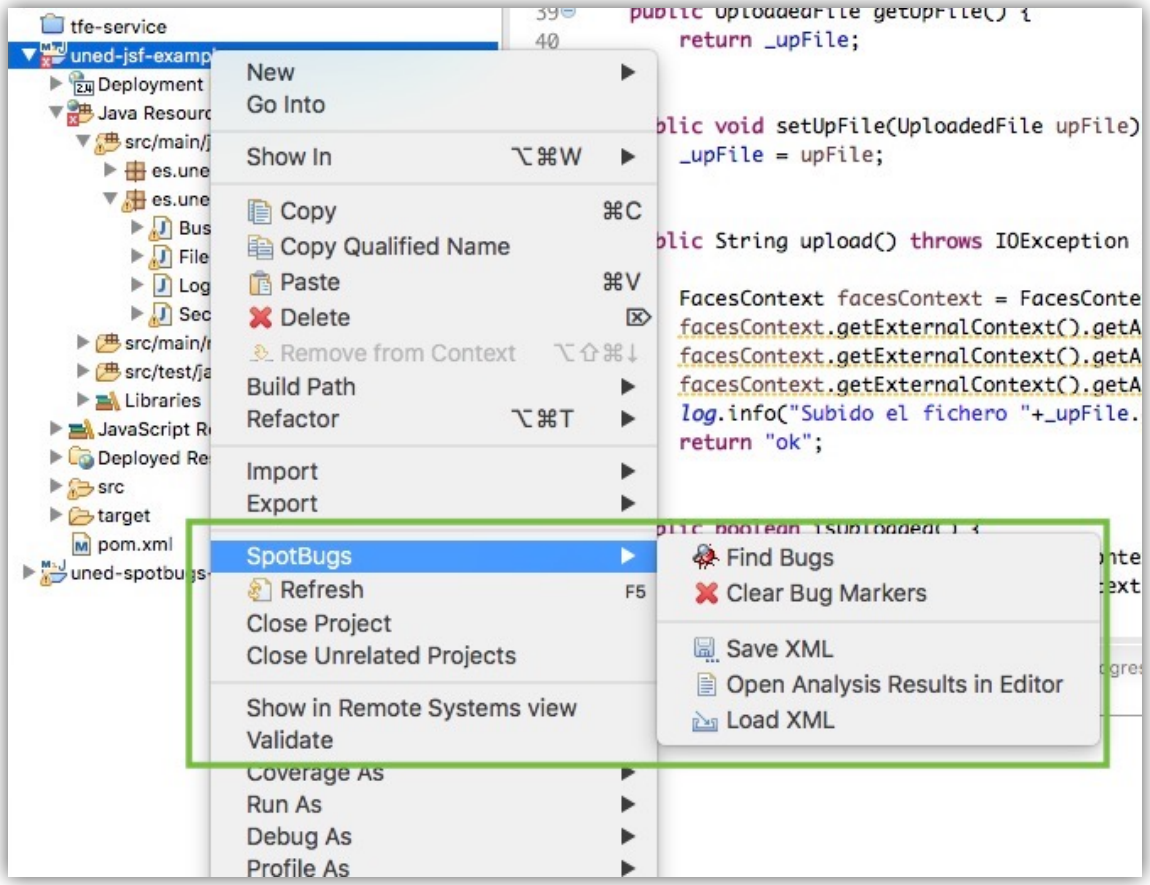

Ilustración 8: Opciones de menú de SpotBugs

Si lanzamos el proceso de búsqueda de errores, nos abrirá una nueva perspectiva en Eclipse en la que nos indicará el listado de errores encontrados en la parte izquierda.

En la parte central nos abrirá el editor de código y nos marcará la línea en la que se ha encontrado el problema.

En la parte derecha se mostrará una pequeña descripción del error y consejos para subsanarlo.

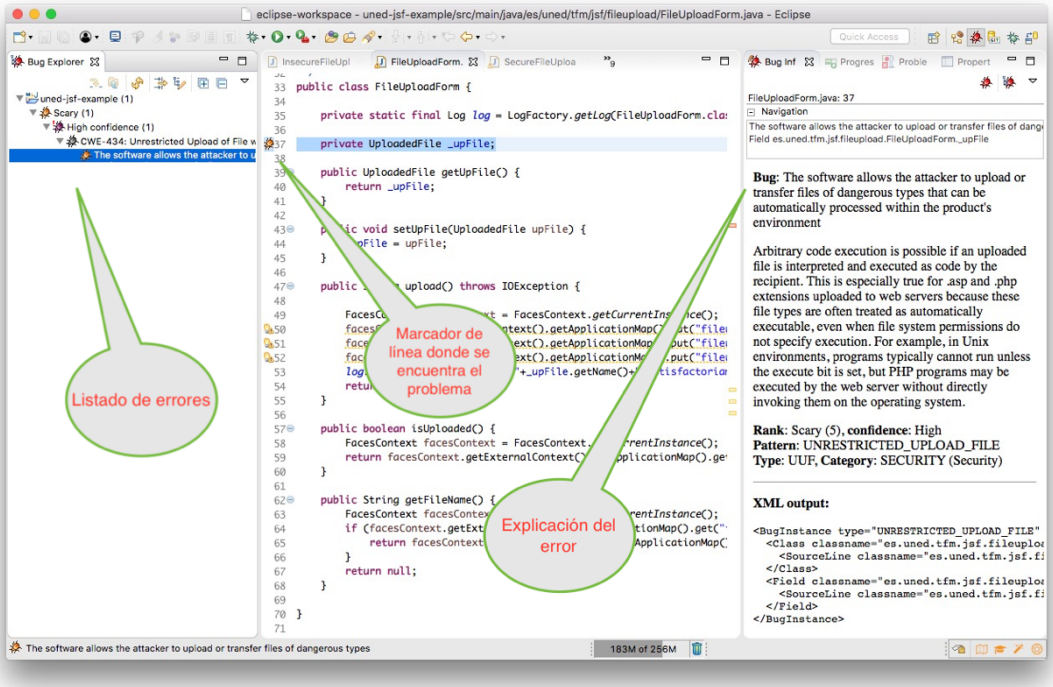

Ilustración 9: Perspectiva de SpotBugs

Con esta herramienta el programador es capaz de detectar posibles errores en el mismo momento en el que está escribiendo el código.

4.2.3 Integración de la herramienta en una plataforma de evaluación de calidad de código fuente.

Además de avisar al programador mientras está escribiendo el código, también necesitamos tener la posibilidad de analizar todo el código fuente de nuestro repositorio para poder encontrar vulnerabilidades en todas las aplicaciones.

Estas herramientas nos dan una métrica de calidad del código y nos muestran la evolución en el tiempo de esta métrica. De esta forma podemos conocer si estamos mitigando las vulnerabilidades o si por el contrario siguen creciendo a lo largo del tiempo.

Actualmente una herramienta muy utilizada para este fin es SonarQube (https://www.sonarqube.org/).

Ilustración 10: Logo de la herramienta SonarQube

SonarQube viene de base con más de 440 reglas de validación para el lenguaje Java, pero además existen plugins para poder incorporar tus propias reglas. De esta manera, añadiremos nuestra librería como un plugin más para que pueda detectar también las reglas que implementemos en nuestra librería.

Ilustración 11: Visión general de SonarQube

4.3 Desarrollo de una herramienta de auditoría

La librería de validación de ficheros se encargaría también de almacenar trazas de auditoría por cada fichero analizado. Para poder consultar estas trazas de manera sencilla vamos a desarrollar una aplicación web que nos permita consultar estas trazas.

La apariencia de esta aplicación web sería algo así:

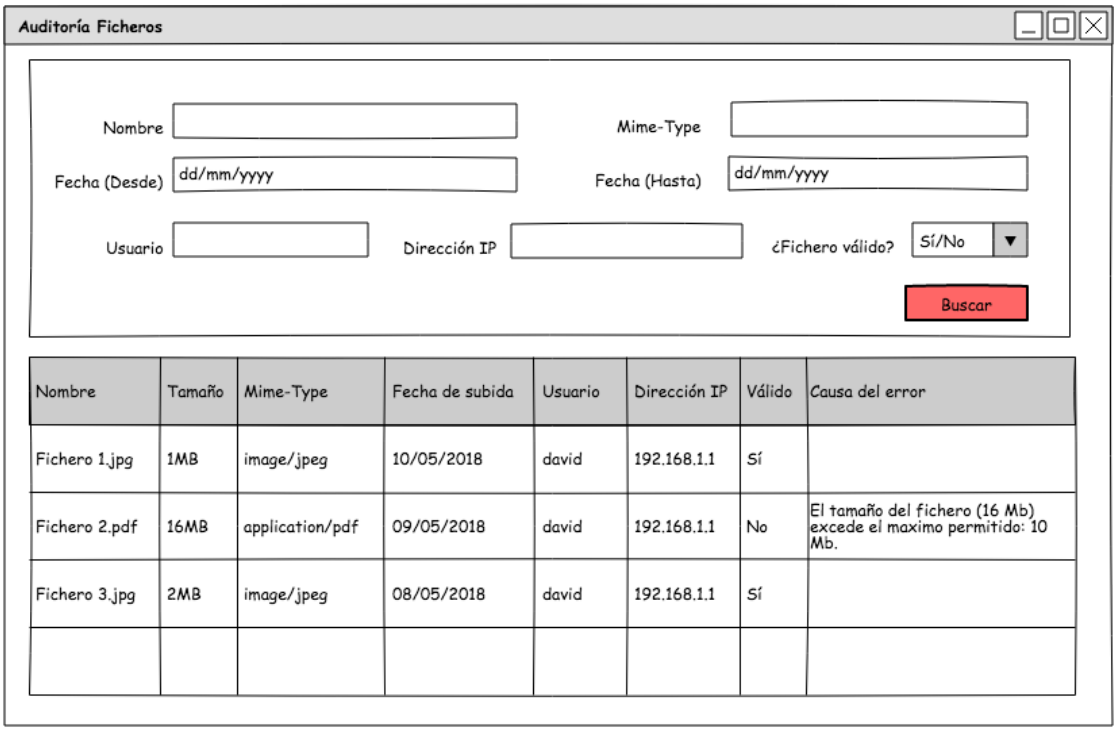

Ilustración 12: Prototipo de la aplicación de consulta de trazas de auditoría

Además, tendríamos que incluir autenticación y autorización para que solo el personal autorizado pueda consultar las trazas.

Con esta herramienta conseguimos varios objetivos:

- Información útil para averiguar la autoría de un posible ataque
- Información para mejorar continuamente la librería de validación. En el supuesto caso de que nuestra librería no hubiera detectado el error de validación podríamos analizar el fichero para mejorar las validaciones.
- Información para generar estadísticas, por ejemplo:
    - ¿cuántos ataques hemos sufrido en un año?
    - ¿De qué zonas del mundo son las IPs desde la que nos hacen los ataques?
    - ¿Cuánto tamaño ocuparían los ficheros recibidos?
    - ¿Qué días recibimos más ficheros?
    - ¿Qué días recibimos más ataques?

4.4 Ventajas e inconvenientes de la solución propuesta frente a otras soluciones existentes

La solución completa que proponemos es totalmente novedosa. No hemos encontrado ninguna solución que contenga las tres herramientas sobre las que se basa esta solución (librería de validación, herramienta de búsqueda de vulnerabilidades y herramienta de auditoría). El único punto en el que podríamos haber optado por utilizar una solución existente hubiera sido la librería de validación, ya que podríamos haber optado por utilizar la librería de validación de ficheros del proyecto OWASP ESAPI. Para toma una decisión hemos estudiado las siguientes ventajas e inconvenientes de cada opción:

Ventajas:

1. **Flexibilidad**: Desarrollamos una librería adaptada a nuestros requisitos. Validamos lo que en nuestra organización vemos interesante validar.
2. **Orientada a Java Server Faces**: Nuestras aplicaciones web están todas desarrolladas con la tecnología JSF, por lo que sería más fácil para los desarrolladores incorporar la validación si ésta ya está pensada para utilizar de manera sencilla en las páginas de JSF.
3. **Más completa**: La librería de OWASP ESAPI a día de hoy no valida el tipo de fichero a partir del contenido del mismo, solo se basa en la extensión del nombre del fichero. A nuestra organización les parece insuficiente esta validación basada en la extensión.

4. **Auditoría**: Además de realizar la validación, esta librería almacenará una traza de auditoría por cada fichero analizado.

Inconvenientes:

1. **Tiempo y esfuerzo de desarrollo**: Como es lógico, tenemos que emplear tiempo y esfuerzo en desarrollar y probar esta nueva librería. Si hubiéramos optado por utilizar la librería de OWASP ESAPI solo tendríamos que incorporarla a nuestros proyectos.
2. **Pérdida de nueva funcionalidad en caso de evolución**: A día de hoy el proyecto está bastante parado, pero podría suceder que en los próximos meses la organización de un impulso al proyecto y amplíe las validaciones del proyecto. En este caso perderíamos esta funcionalidad y tendríamos que implementarlo nosotros en nuestra librería.

A modo de resumen, podemos ver las ventajas y desventajas en la siguiente tabla:

Tabla 1: Ventajas y desventajas en el desarrollo de una librería de validación

ESTRATEGIA	VENTAJAS	DESVENTAJAS
Desarrollo de una nueva librería	1. Flexibilidad 2. Orientada a Java Server Faces 3. Más completa 4. Auditoría	1. Tiempo y esfuerzo de desarrollo 2. Pérdida de nueva funcionalidad en caso de evolución
Utilizar la librería de OWASP ESAPI	1. No es necesario desarrollo extra 2. Nos aprovecharíamos de (posibles) futuras mejoras.	1. No realiza suficientes validaciones según nuestro criterio 2. No almacena trazas de auditoría

Creemos que el desarrollo de una nueva librería nos ofrece más ventajas que desventajas, por lo tanto, se desarrollará una nueva librería.

5 Experimentación y contrastación de la propuesta

Hemos desarrollado un piloto para probar nuestra solución antes de aplicarla en el desarrollo de nuestros proyectos finales. El piloto consiste en una aplicación web desarrollada con JSF que contiene:

- Un formulario no seguro, desarrollado de la misma manera que el resto de formularios de nuestras aplicaciones web. En este formulario probaremos las vulnerabilidades descritas anteriormente.
- Un formulario seguro, partimos del formulario inseguro, pero le aplicamos las validaciones que hemos desarrollado. Sobre este formulario probaremos que ya no existen las vulnerabilidades que presentaba el formulario anterior.
- Un enlace a la herramienta de consulta de trazas de auditoría. Con esta herramienta podremos ver las trazas que se almacenan al hacer las pruebas sobre el formulario seguro.

La pantalla principal de nuestro prototipo tiene la siguiente apariencia:

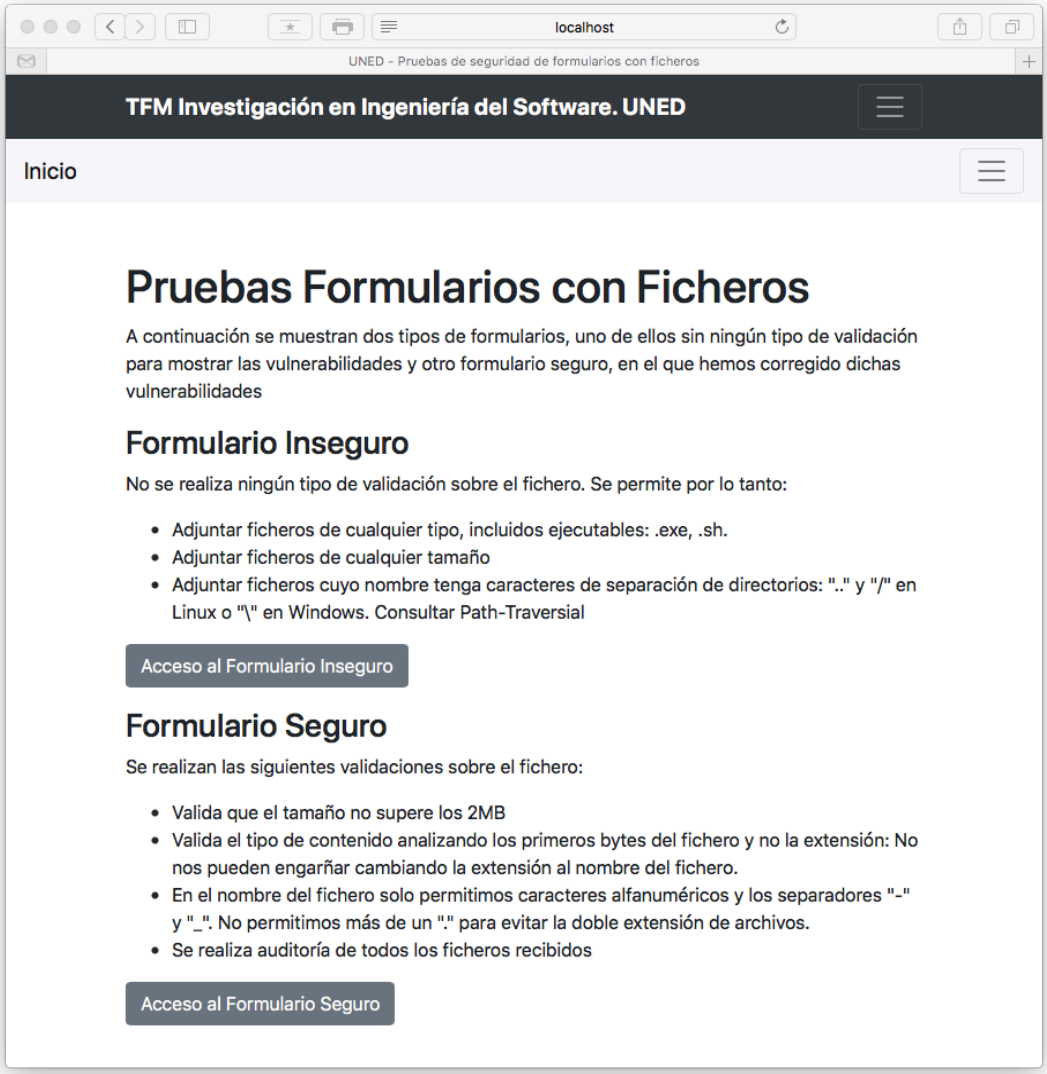

Ilustración 13: Pantalla principal del prototipo de pruebas

La funcionalidad de nuestro prototipo consiste en formulario que nos solicita un fichero de tipo imagen, una vez seleccionemos el fichero, se guardará en un directorio del servidor y se mostrará la imagen en la pantalla. También se nos muestra un enlace para descargarla.

Veamos un ejemplo:

Accedemos al formulario:

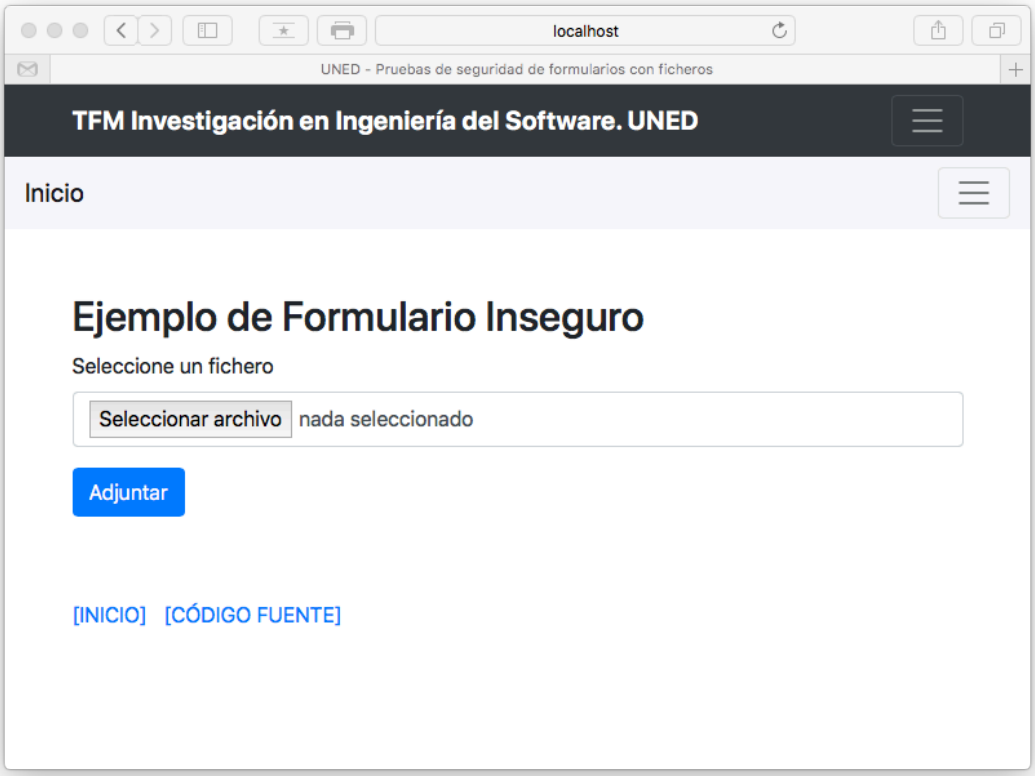

Ilustración 14: Acceso al formulario inseguro

Seleccionamos una imagen:

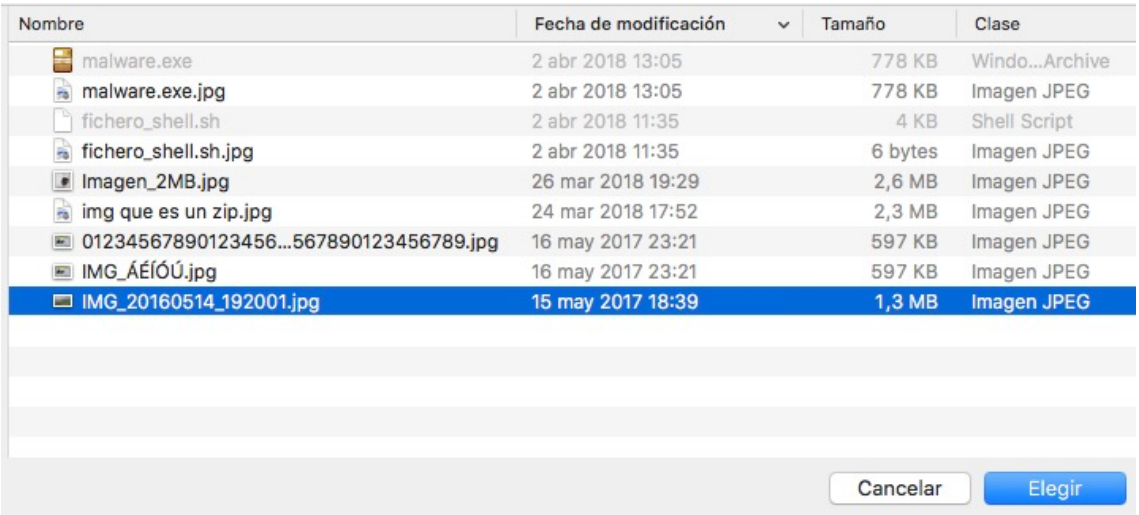

Ilustración 15: Selección de una imagen de nuestro equipo

Cuando enviamos el formulario la imagen se sube al servidor y se muestra la imagen por pantalla. También nos muestra un enlace para descargar el fichero.

Ilustración 16: Resultado del envío del formulario con la imagen al servidor

Este sería el funcionamiento normal y deseado de la aplicación, pero como veremos en los siguientes apartados se podría aprovechar este formulario para efectuar ataques a nuestra organización.

5.1 Experimentación de vulnerabilidades sobre el formulario no seguro

Vamos a probar cada una de las vulnerabilidades descritas anteriormente para comprobar si explotando cada una de ellas conseguiríamos realmente efectuar un ataque.

5.1.1 CWE-434: Unrestricted file Upload with Dangerous type

Un sistema con esta vulnerabilidad permitiría a un atacante la subida de un fichero de un tipo que no se correspondería con el tipo de fichero esperado por la aplicación.

5.1.1.1 Prueba 1: Enviamos un fichero ejecutable enmascarando su extensión

Nuestro formulario inseguro acepta ficheros de tipo .jpg, un tipo muy común para las imágenes. ¿Qué pasaría si renombramos un ejecutable con cualquier tipo de malware?

Renombramos:

malware.exe -> malware.exe.jpg

Accedemos al formulario inseguro y seleccionamos el fichero “malware.exe.jpg”:

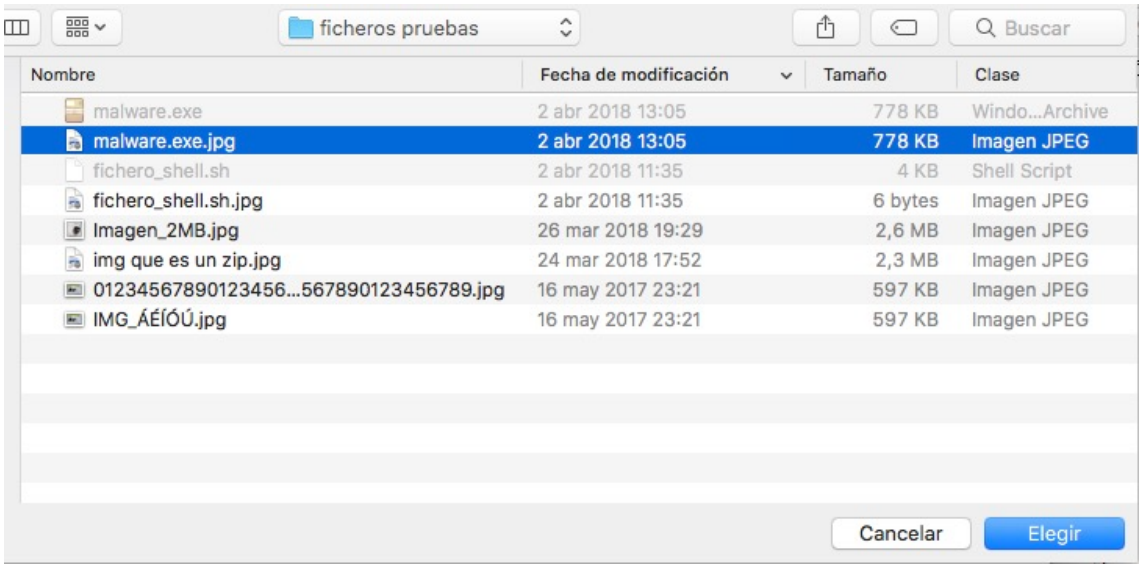

Ilustración 17: Selección de un fichero con la extensión renombrada

El resultado ha sido que nos ha permitido subir nuestro ejecutable con malware al servidor. Ahora ese fichero está dentro de nuestra organización y cualquiera lo podría ejecutar.

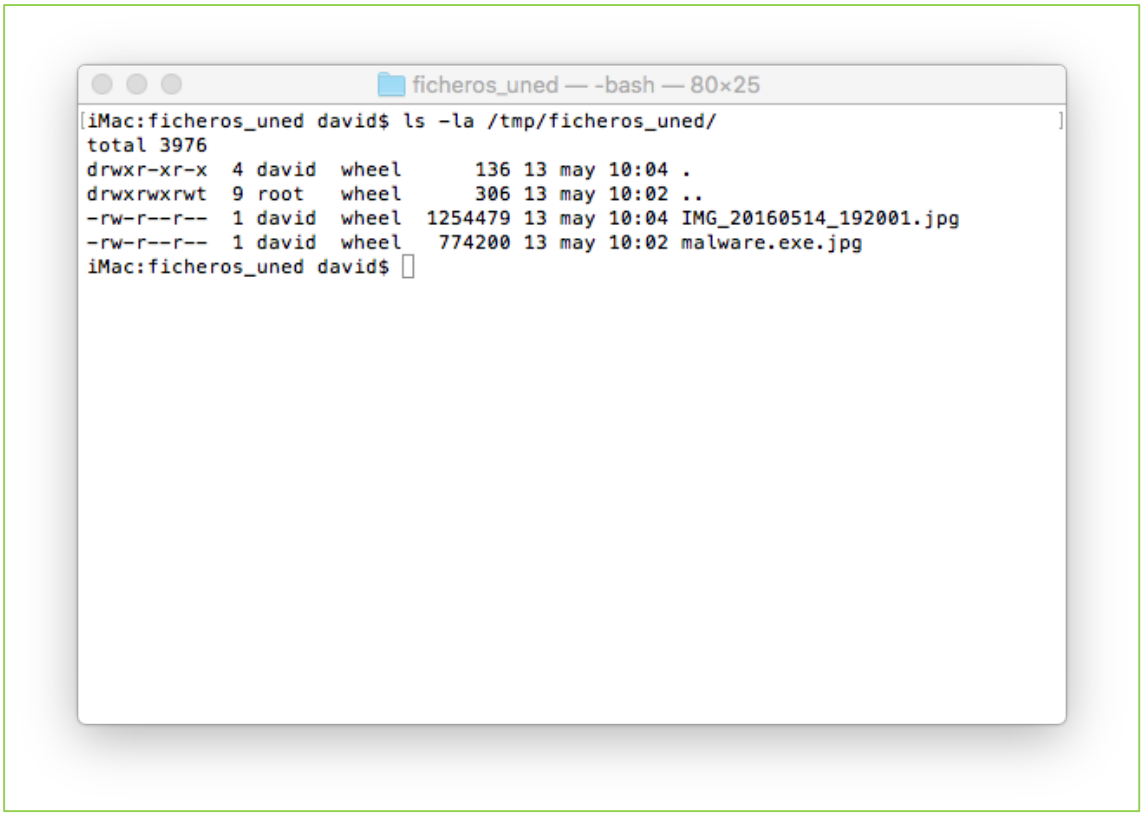

Ilustración 18: Contenido del directorio donde se almacenan los ficheros en nuestro servidor

En nuestro prototipo los ficheros se almacenan en un directorio temporal y no los necesitamos abrir, pero en nuestra organización estos ficheros se muestran en las aplicaciones de gestión y el personal que debe de tramitar esta información sí que los necesitan abrir para consultar su contenido. Aquí sí que tenemos un riesgo real de que alguien abra alguno de estos ficheros.

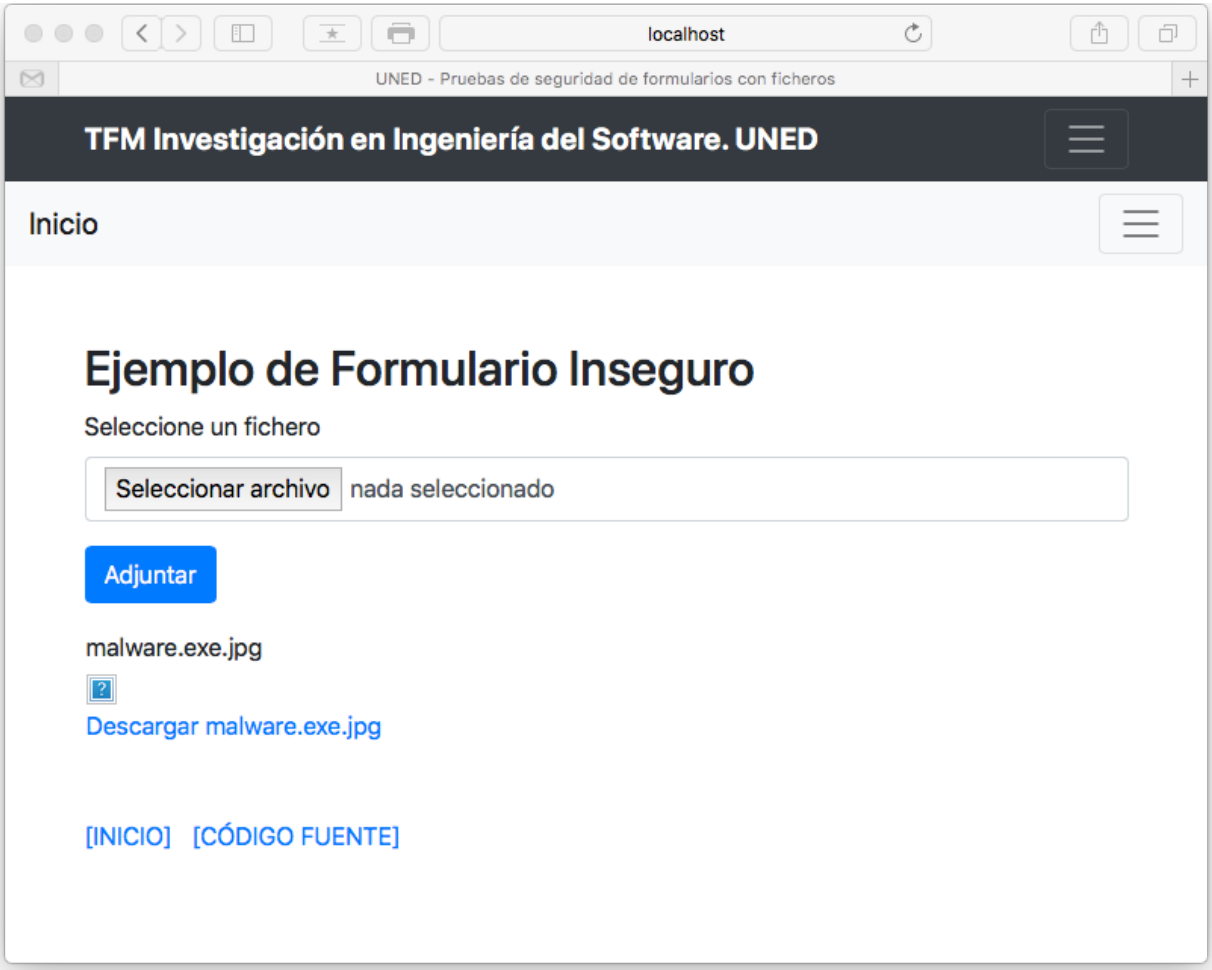

Ilustración 19: Resultado de envío de malware ejecutable con extensión renombrada

En nuestro prototipo, lógicamente la imagen no se ve porque en realidad no era ninguna imagen, pero si pulsamos en el enlace "Descargar malware.exe.jpg" sí nos descargaría el malware a nuestro equipo.

5.1.1.2 Prueba 2: Enviamos un fichero ejecutable cambiando el nombre del fichero con una herramienta Proxy web

Para realizar esta prueba vamos a renombrar nuestro malware.exe a malware.jpg para engañar al navegador y que nos permita seleccionar el fichero, pensando que es en realidad una imagen.
Después utilizaremos la herramienta Proxy OWASP ZAP para poder editar la petición HTTP antes de enviarla al servidor y cambiar el nombre del fichero a su extensión original, de malware.jpg a malware.exe.

Configuramos en la aplicación OWASP ZAP un puerto en el que escuchará para interceptar las peticiones. En nuestro caso configuramos que intercepte las peticiones que vayan al puerto 8082 en nuestro equipo local (localhost):

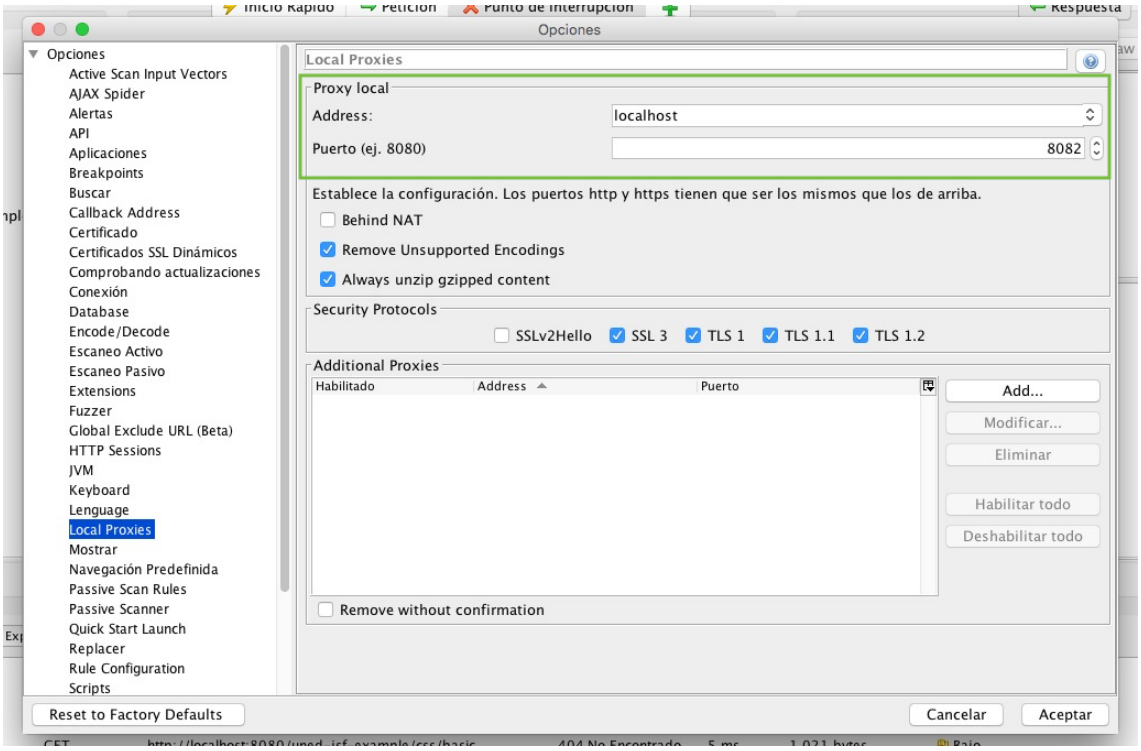

Ilustración 20: Configuración del proxy en OWASP ZAP

También tenemos que configurar nuestro navegador para indicarle que las peticiones las realice al proxy de OWASP ZAP que acabamos de configurar (localhost en el puerto 8082):

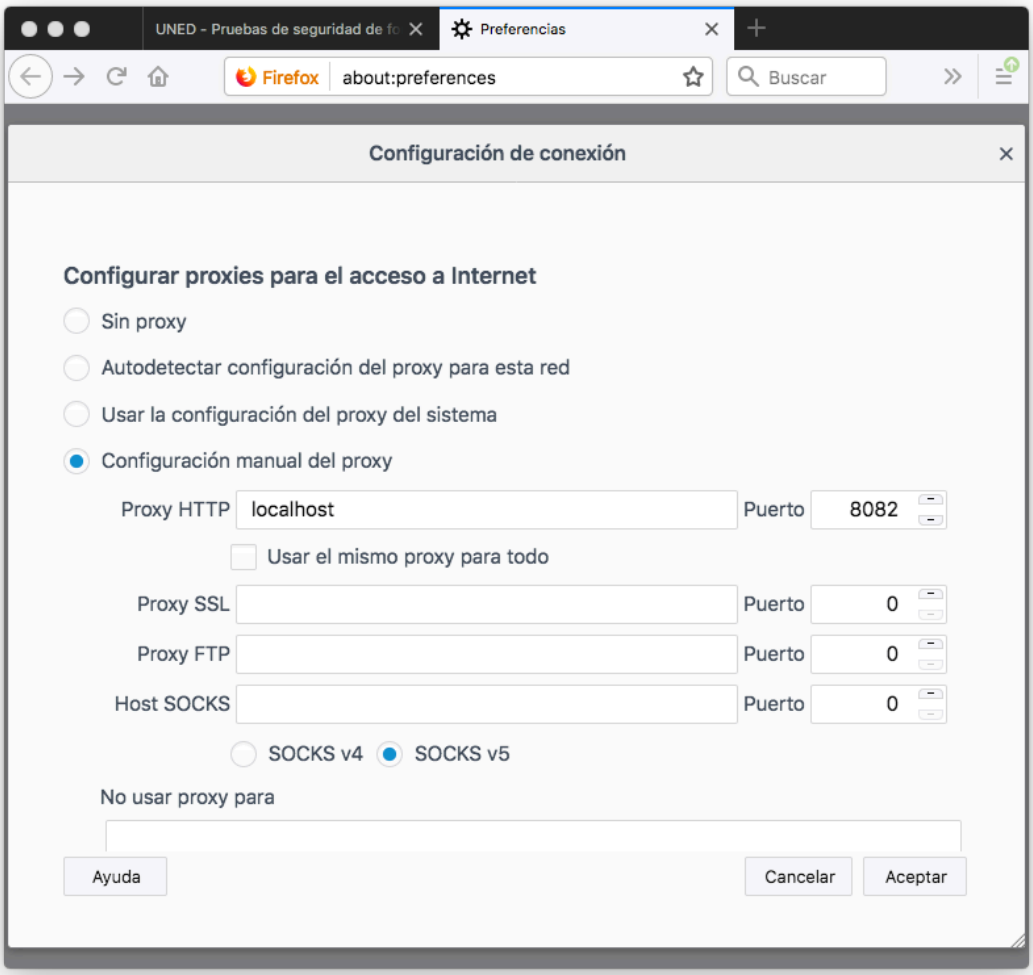

Ilustración 21: Configuración del navegador Mozilla Firefox para establecer el proxy a la aplicación OWASP ZAP

Una vez tengamos el proxy configurado y escuchando y el navegador configurado para que envíe las peticiones a dicho proxy ya podemos comenzar la prueba.

Accedemos al formulario y seleccionamos el fichero malware.jpg:

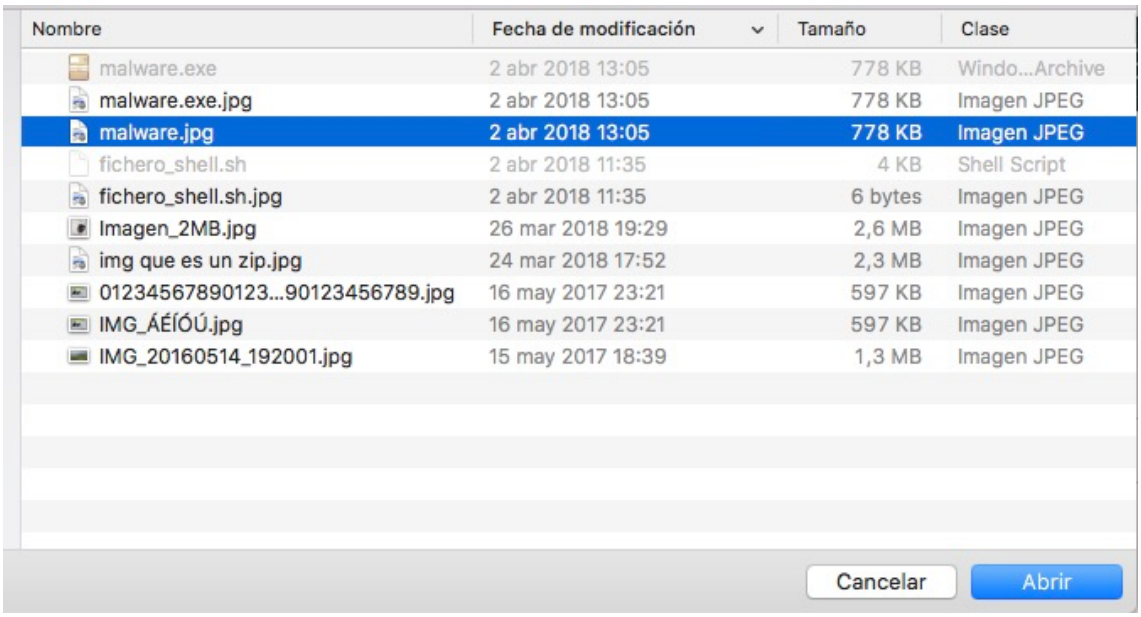

Ilustración 22: Selección del fichero malware.jpg, que en realidad es un .exe

Enviamos el formulario, ahora la aplicación OWASP ZAP intercepta la petición HTTP y nos muestra tanto la cabecera como el cuerpo en un área de texto para poder editarla antes de reenviarla al servidor.

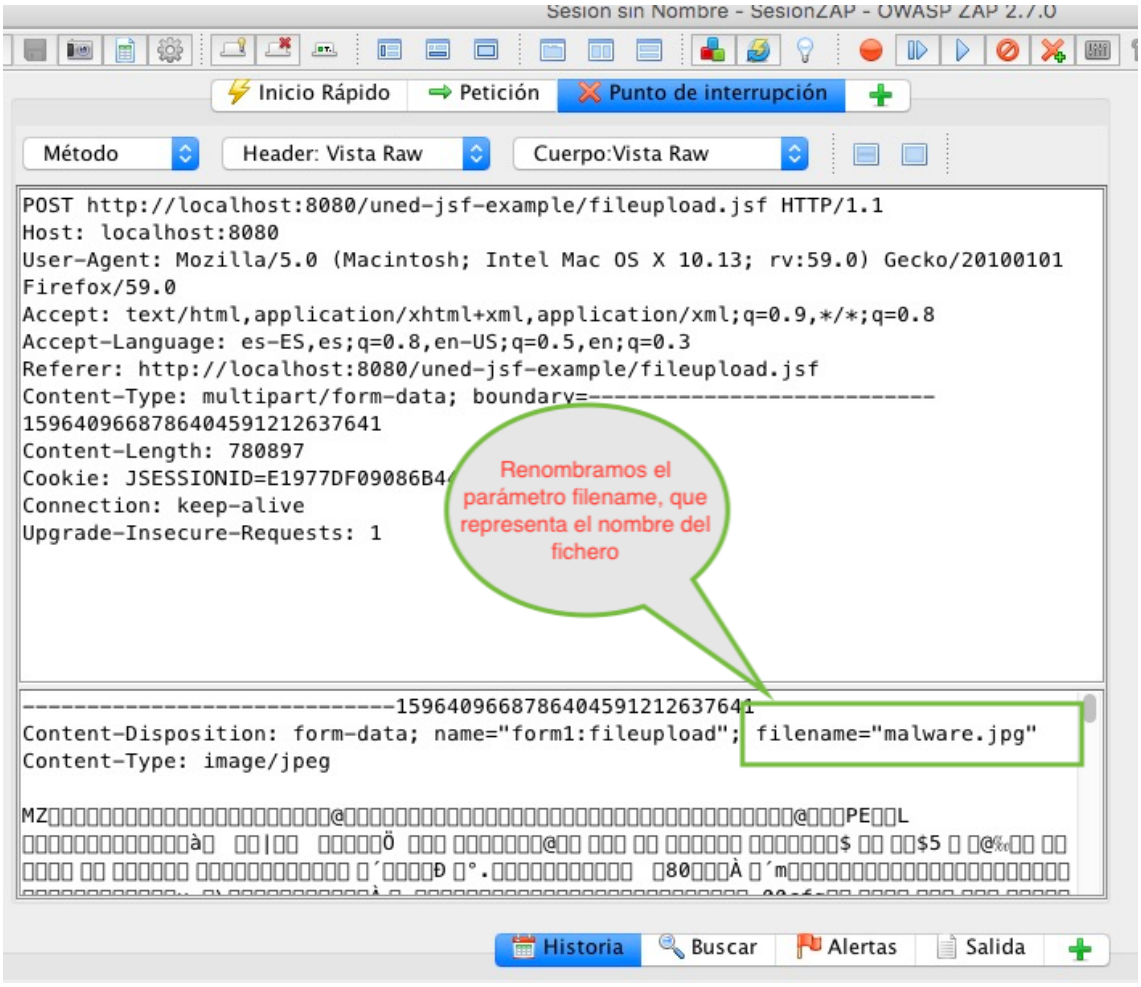

Ilustración 23: Edición de la petición HTTP en OWASP ZAP

Renombramos el nombre del fichero a su extensión original: “malware.exe”:

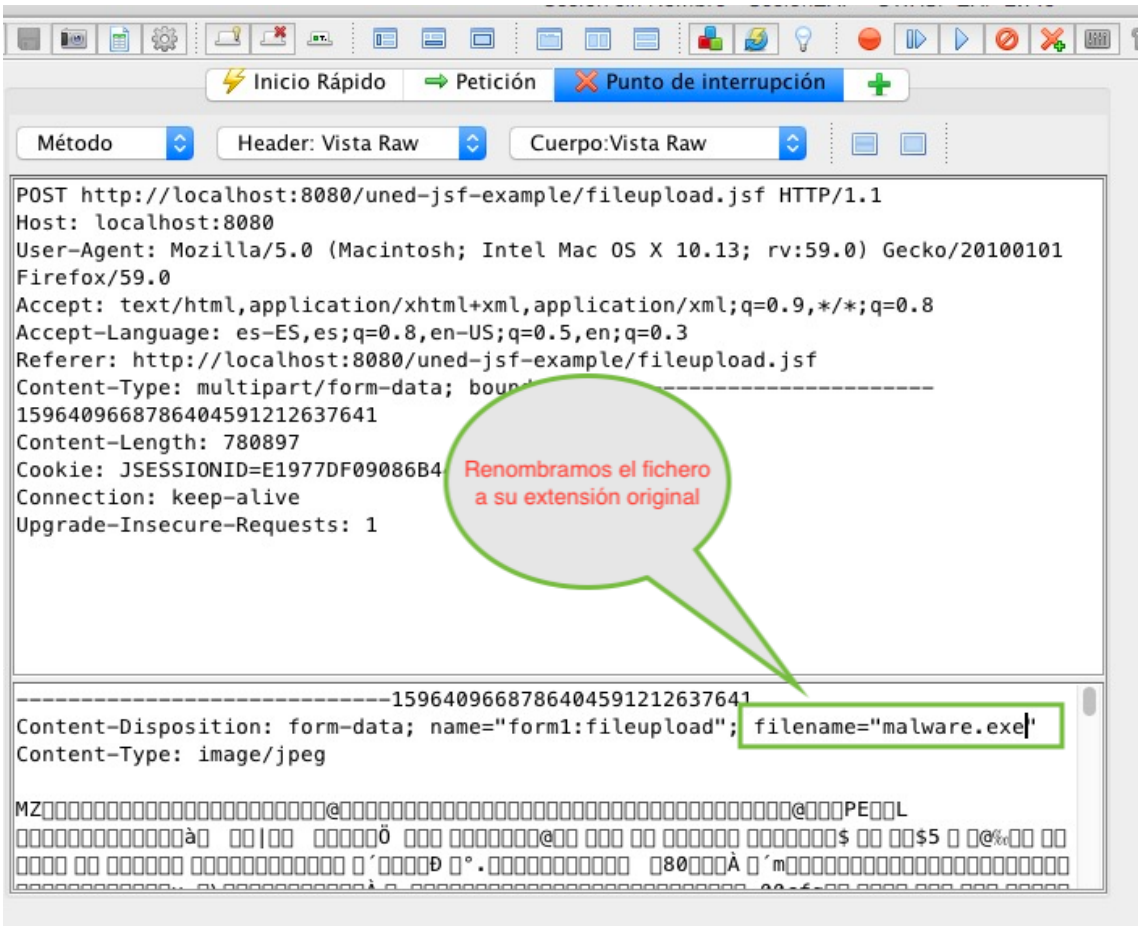

Ilustración 24: Renombrado del cuerpo de la petición HTTP en OWASP ZAP

Una vez editada la respuesta a nuestro antojo reenviamos la petición.

Como podemos observar el resultado ha sido que hemos podido subir el fichero "malware.exe" a nuestro servidor sin ningún problema:

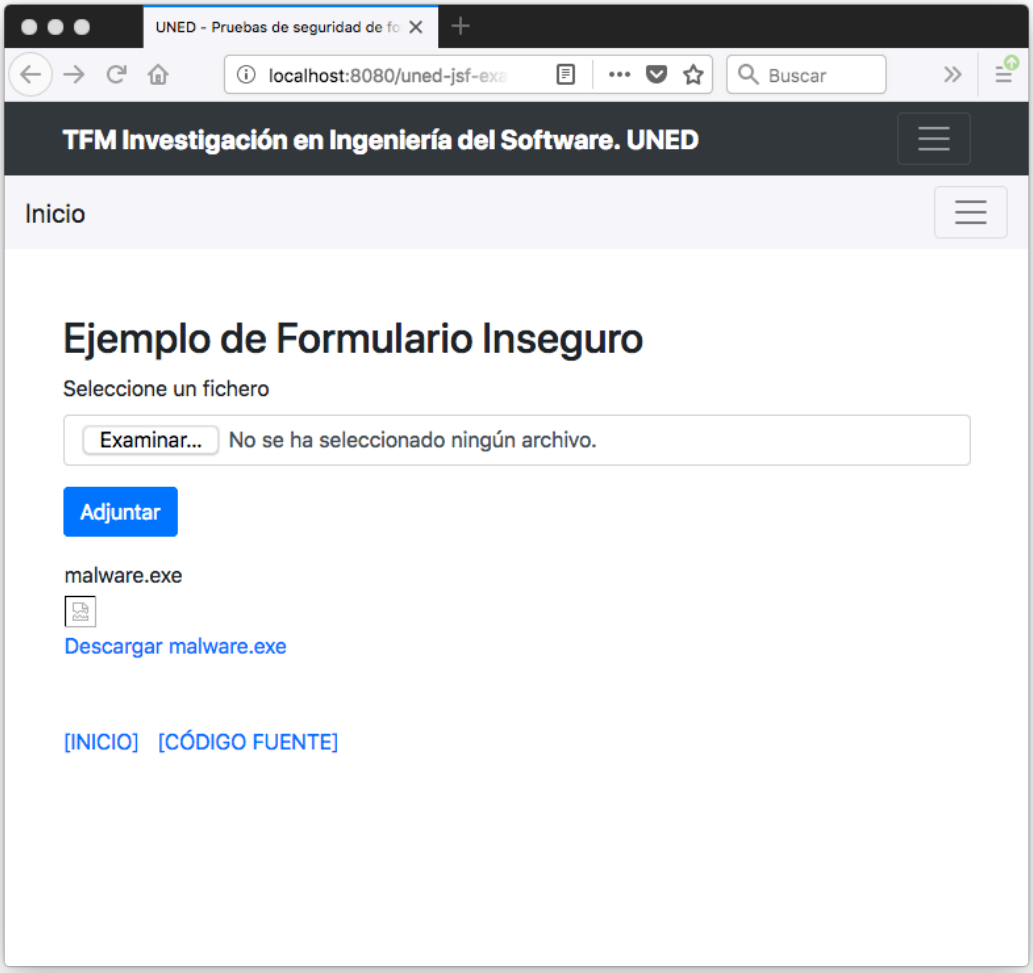

Ilustración 25: Resultado del envío de un fichero ejecutable editando la petición HTTP con una herramienta proxy

En este momento tenemos el fichero malware.exe en nuestra organización, cualquier persona podría ejecutarlo, pudiendo ser los resultados desastrosos.

5.1.2 CWE-400: Uncontrolled Resource Consumption ('Resource Exhaustion')

La aplicación no limita adecuadamente la cantidad de recursos que utiliza un usuario, lo que le permite consumir más recursos de los necesarios.

5.1.2.1 Prueba 1: Subida de un fichero de gran tamaño

Para hacer la prueba vamos a intentar subir un fichero de más de 100MB. Además, el hecho de que la aplicación no valide correctamente el tipo de contenido no es necesario enviar realmente una imagen, cualquier fichero que tenga una extensión .jpg nos servirá para realizar el ataque.

Accedemos al formulario, seleccionamos el fichero de 100 MB y enviamos el formulario:

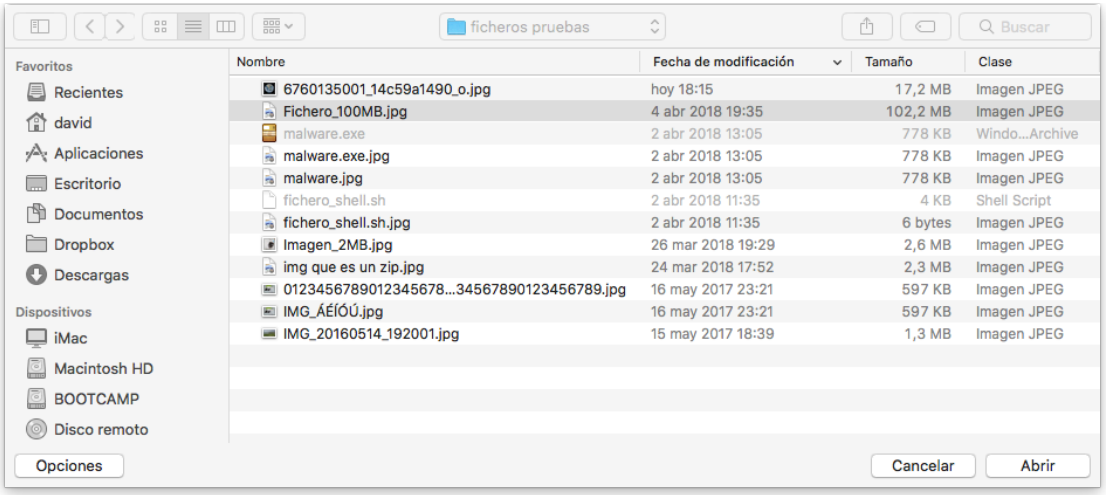

Ilustración 26: Selección de un fichero de 102MB

El resultado ha sido que hemos conseguido enviar al servidor un fichero de más de 100MB.

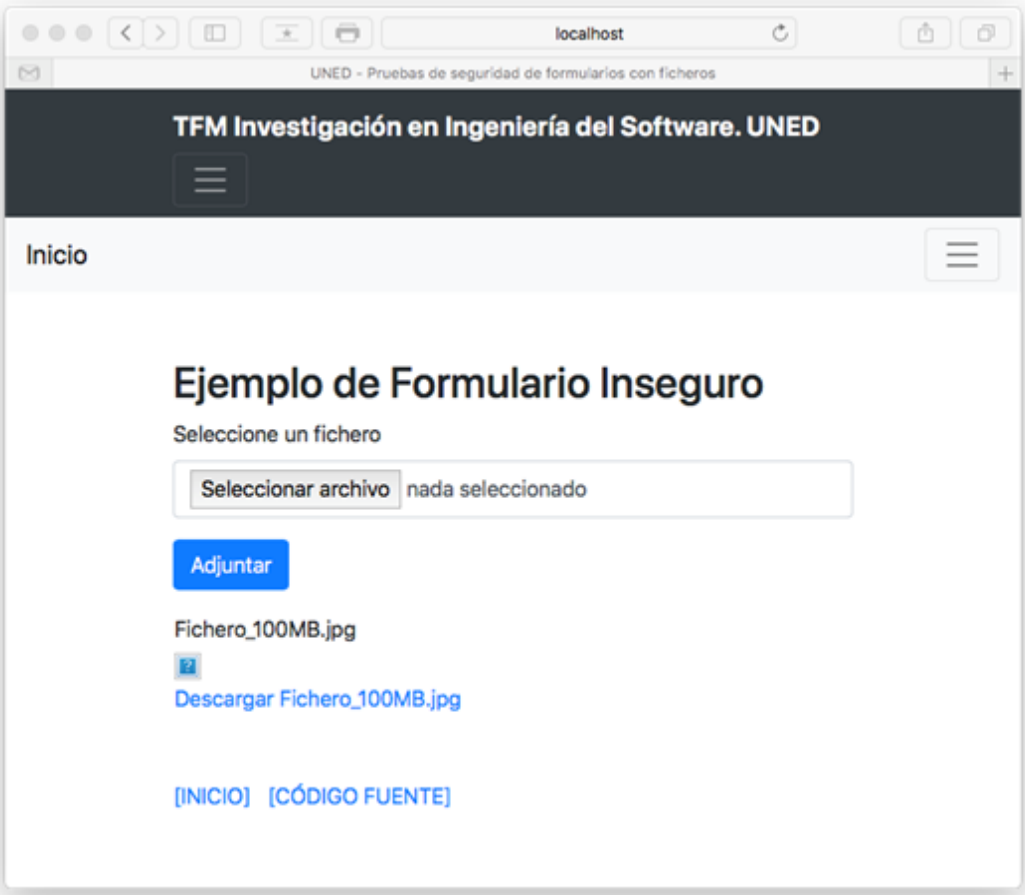

Ilustración 27: Subida de un fichero de 100MB

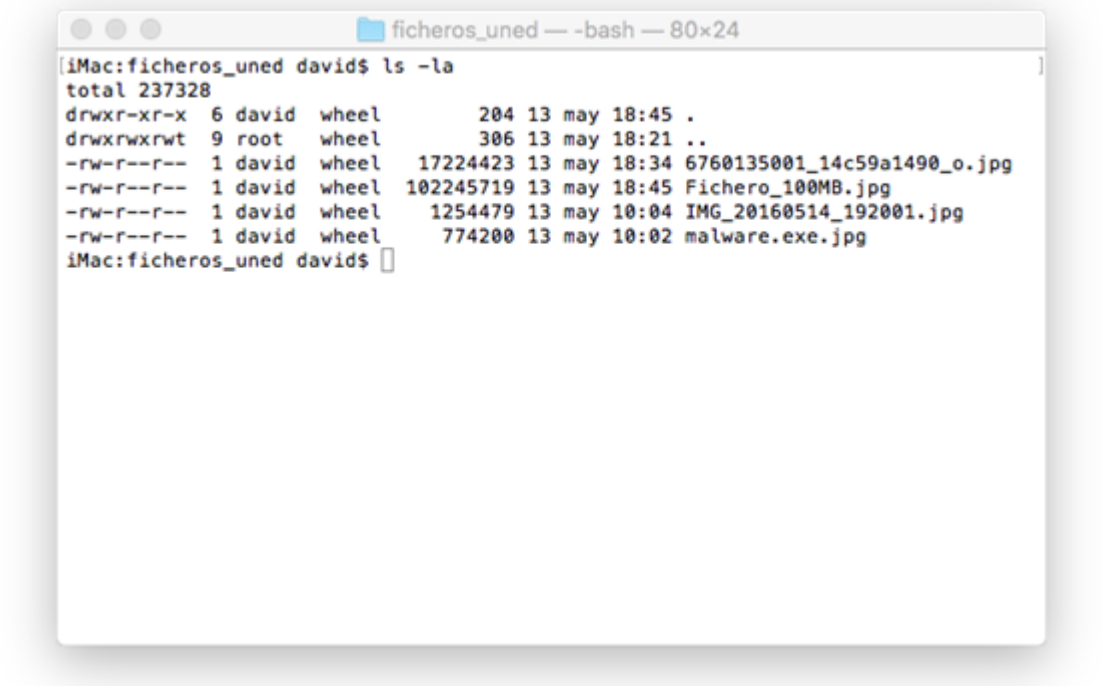

Ilustración 28: Fichero de 100MB en el directorio del servidor

Si repetimos la misma petición HTTP miles de veces con una herramienta para tal fin podremos dejar sin espacio el disco del servidor. Si esto ocurriera la aplicación dejará de funcionar correctamente, por lo tanto, el atacante ha conseguido su objetivo.

5.1.3 CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

Si nuestra aplicación utiliza un parámetro de entrada para construir la ruta en la que se almacenará un fichero y no validamos correctamente el parámetro de entrada, entonces podríamos permitir a un atacante construir una ruta a un directorio no permitido.

5.1.3.1 Prueba 1: Modificamos el nombre del fichero con una herramienta Proxy web

El fin de la prueba será intentar escribir en el servidor en un directorio diferente al que se ha configurado para almacenar los ficheros que se suben al servidor. Si logramos este ataque podríamos llegar a sobrescribir cualquier fichero del servidor.

Para llevar a cabo este ataque necesitaremos de nuevo una herramienta que nos permita capturar la petición HTTP y editarla antes de enviarla definitivamente al servidor.

Abrimos de nuevo la herramienta OWASP ZAP y configuramos el proxy para que escuche en el puerto 8082, que será al que enviaremos las peticiones desde nuestro navegador.

Accedemos al formulario inseguro y seleccionamos un fichero cualquiera.

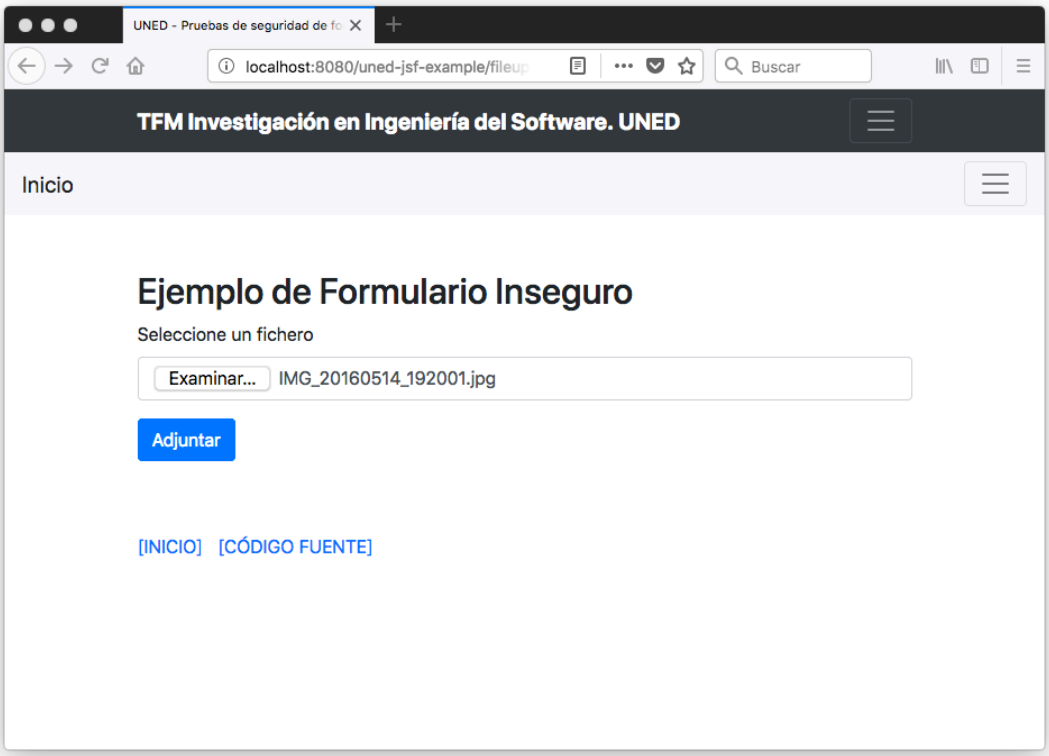

Ilustración 29: Selección de un fichero desde el formulario inseguro

Enviamos el formulario y nos salta la pantalla de OWASP ZAP con la petición capturada:

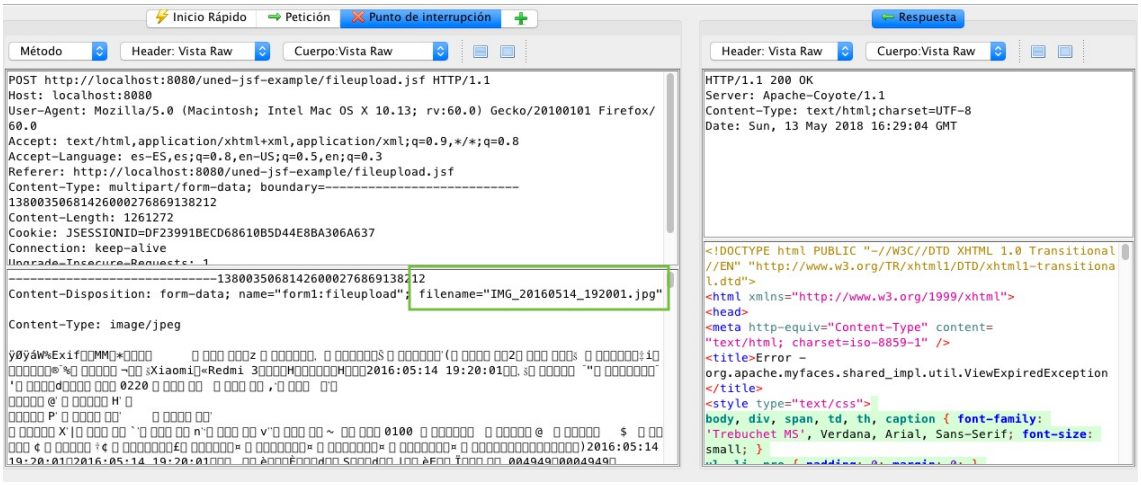

Ilustración 30: Captura de la petición HTTP con una herramienta proxy

Modificamos el parámetro “filename” y añadimos una ruta relativa:

“../../tmp/imagen_uned.jpg”

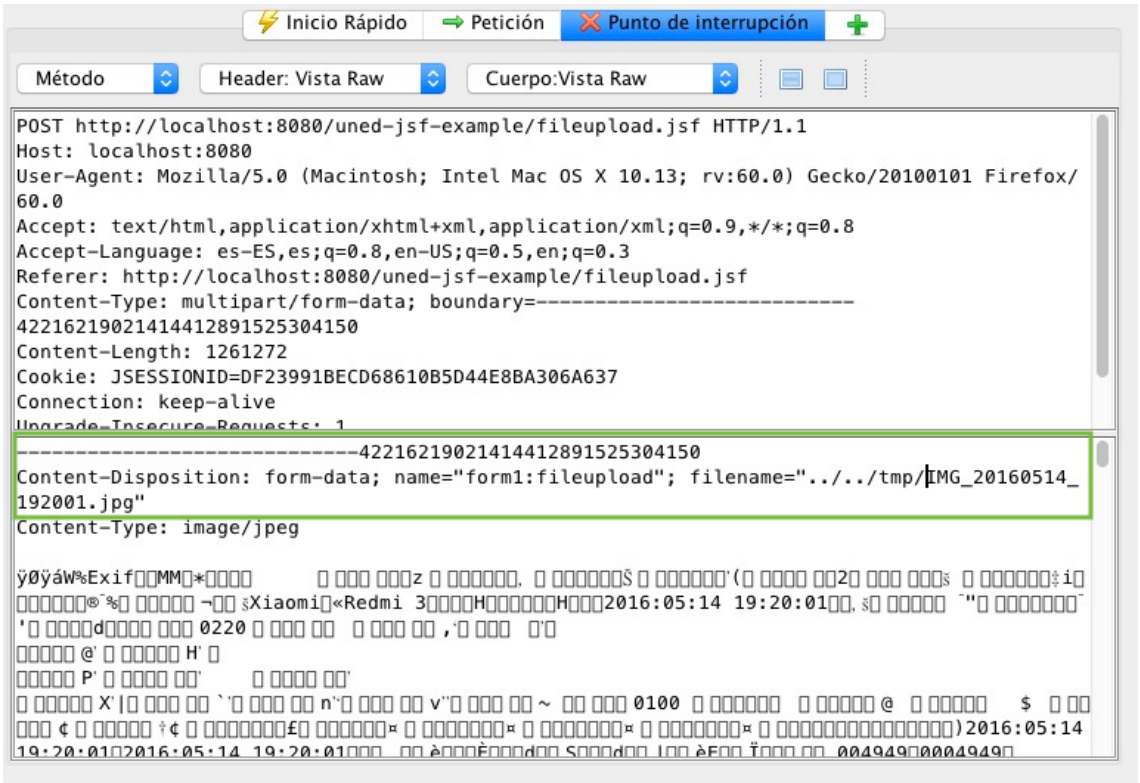

Ilustración 31: Modificación del parámetro filename de la petición para cambiar la ubicación

Reenviamos la petición y comprobamos que la imagen se ha subido correctamente:

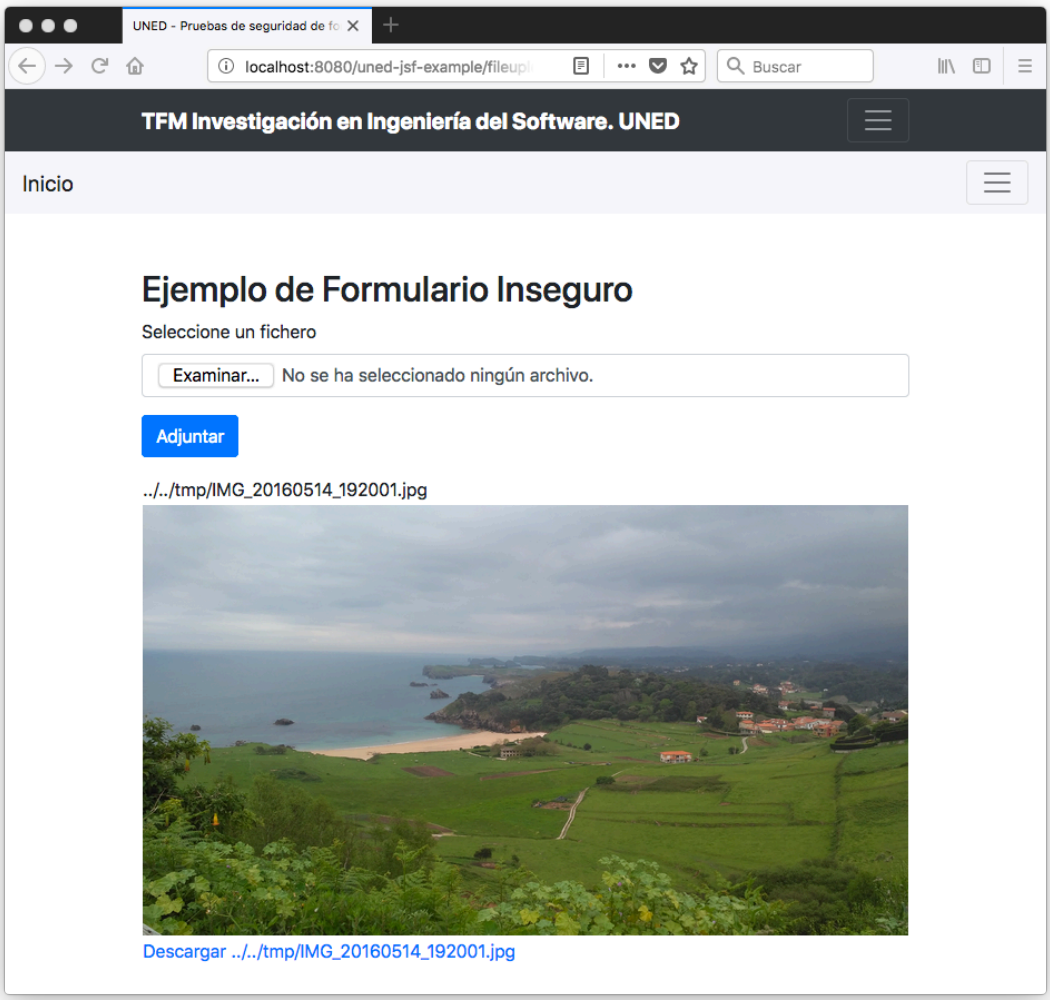

Ilustración 32: Envío de un fichero con una relativa en su nombre

Si vamos al servidor y comprobamos si la imagen está en el directorio especificado "/tmp" vemos que sí se ha escrito:

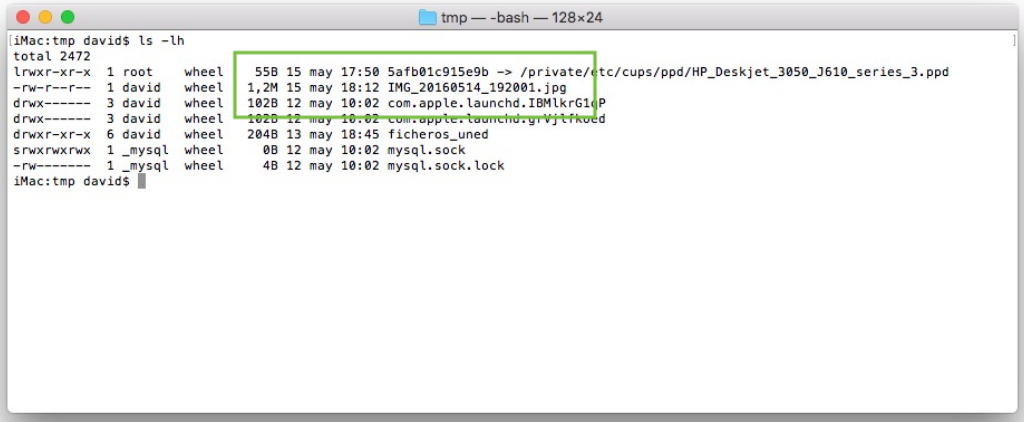

Ilustración 33: Listado del directorio /tmp para comprobar que el fichero se ha escrito en un directorio donde no debería

El fichero se ha escrito en el directorio "/tmp" en lugar de "/tmp/ficheros_uned", que era el directorio que la aplicación tiene configurado para dejar los ficheros.

Aprovechando esta vulnerabilidad un atacante podría llegar a copiar cualquier fichero en cualquier directorio del servidor, incluso podría llegar a sobrescribir los propios ficheros de la aplicación, por ejemplo, una aplicación Java podría sobrescribir cualquier fichero. jsp o .class, ¡**modificando el funcionamiento de la misma**!

5.2 Experimentación de vulnerabilidades sobre el formulario seguro

Una vez verificado que podemos aprovechar todas estas vulnerabilidades que presenta el formulario no seguro para efectuar un ataque a nuestra organización, vamos a repetir las mismas pruebas sobre el formulario seguro para contrastar si hemos conseguido eliminar dichas vulnerabilidades.

El formulario seguro tiene exactamente la misma apariencia que el formulario no seguro, pero hemos añadido un requisito la autenticación y autorización según las recomendaciones del apartado 4.1.5.1 Solución válida: Proteger el formulario en

cuestión con autenticación y autorización. Así que intentamos acceder al formulario seguro y si no estamos autenticados nos mostrará la pantalla de autenticación:

Ilustración 34: Pantalla de autenticación para poder acceder al formulario seguro

Introducimos nuestras credenciales y si son válidas accedemos al formulario:

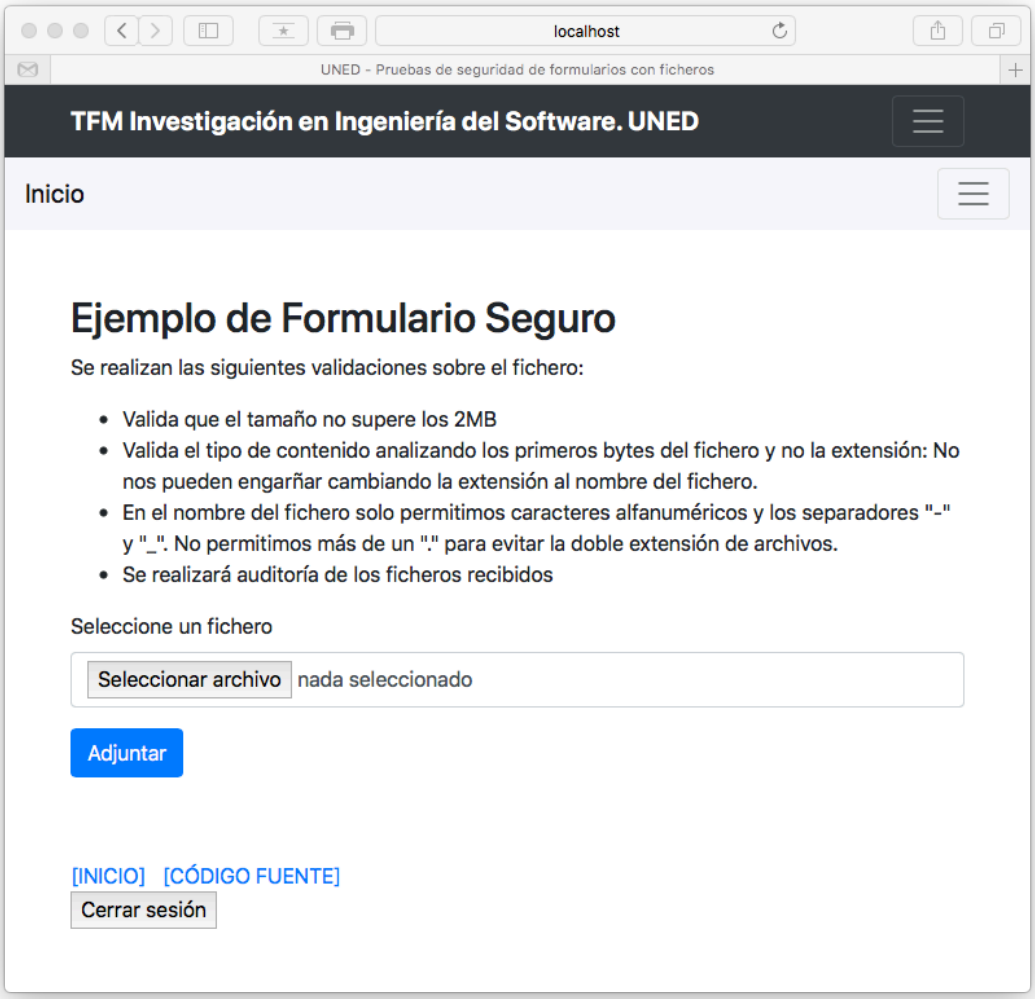

Ilustración 35: Acceso al formulario seguro

Una vez hemos accedido vamos a comenzar con las pruebas:

5.2.1 CWE-434: Unrestricted file Upload with Dangerous type

Un sistema con esta vulnerabilidad permitiría a un atacante la subida de un fichero de un tipo que no se correspondería con el tipo de fichero esperado por la aplicación.

5.2.1.1 Prueba 1: Enviamos un fichero ejecutable enmascarando su extensión

Nuestro formulario inseguro acepta ficheros de tipo .jpg, un tipo muy común para las imágenes. ¿Qué pasaría si renombramos un ejecutable con cualquier tipo de malware?

Renombramos:

```
malware.exe -> malware.exe.jpg
```

Accedemos al formulario inseguro y seleccionamos el fichero “malware.exe.jpg”:

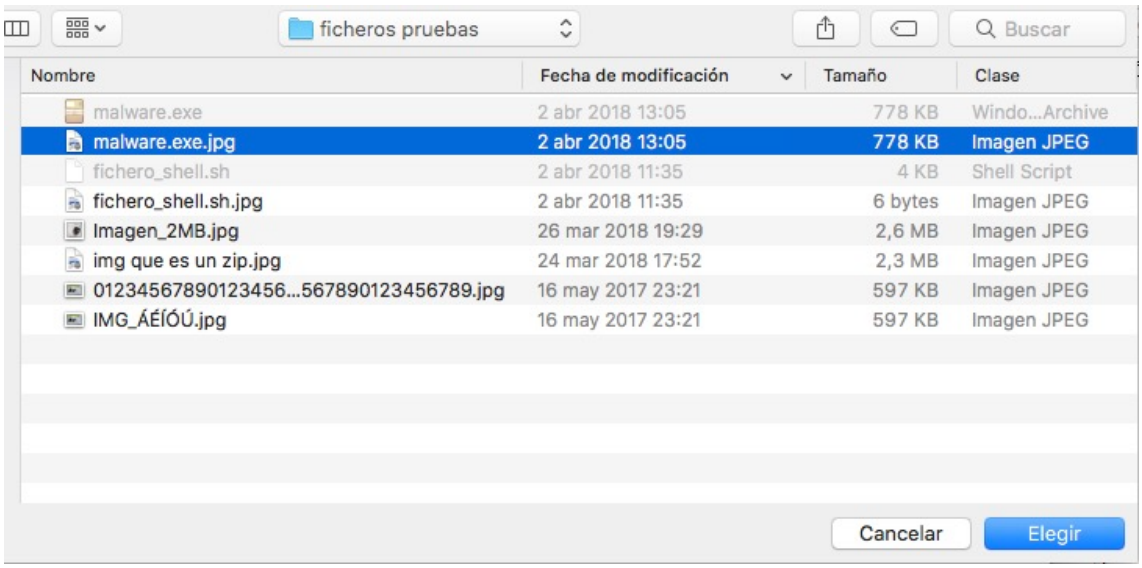

Ilustración 36: Selección de un fichero con la extensión renombrada

Si intentamos adjuntar el fichero “malware.exe.jpg” el resultado es el siguiente:

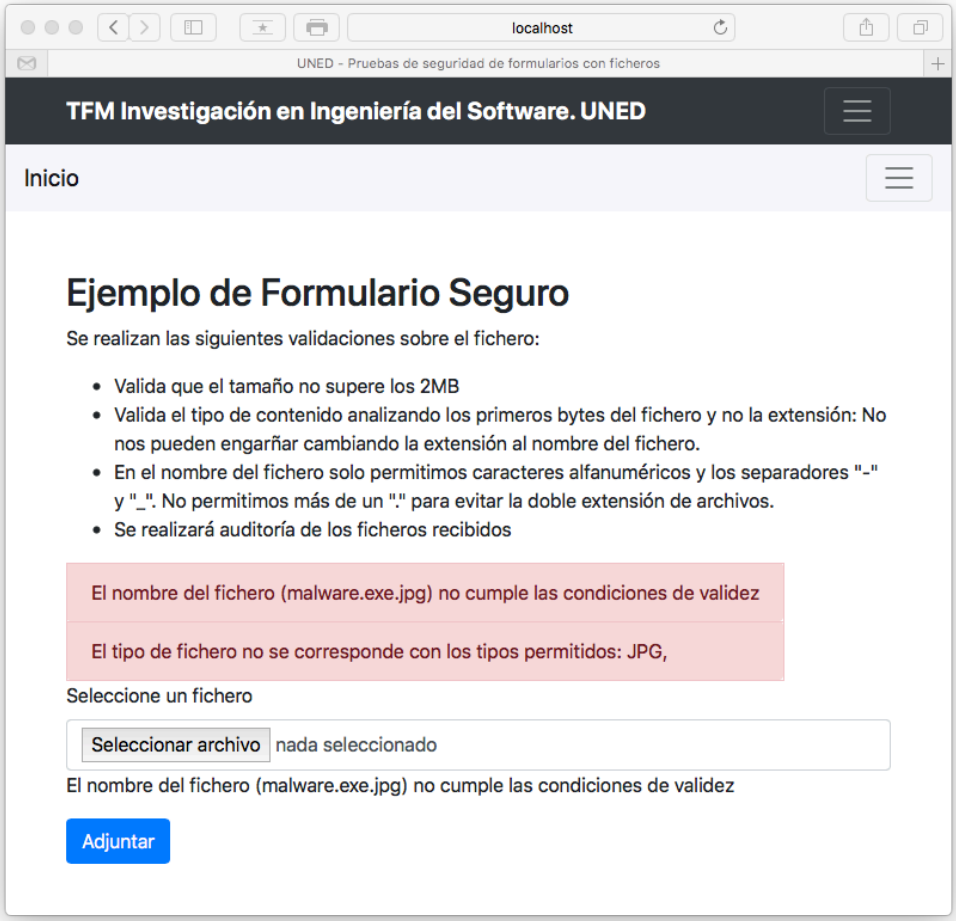

Ilustración 37: Mensajes de error de validación al intentar adjuntar un fichero que no cumple las condiciones de validez

El resultado es el esperado, la aplicación nos muestra un mensaje de error y no nos permite subir el fichero a nuestro servidor.

5.2.1.2 Prueba 2: Enviamos un fichero ejecutable cambiando el nombre del fichero con una herramienta Proxy web

Para realizar esta prueba vamos a renombrar nuestro malware.exe a malware.jpg para engañar al navegador y que nos permita seleccionar el fichero, pensando que es en realidad una imagen.

Después utilizaremos la herramienta Proxy OWASP ZAP para poder editar la petición HTTP antes de enviarla al servidor y cambiar el nombre del fichero a su extensión original, de malware.jpg a malware.exe.
Accedemos al formulario y seleccionamos el fichero malware.jpg:

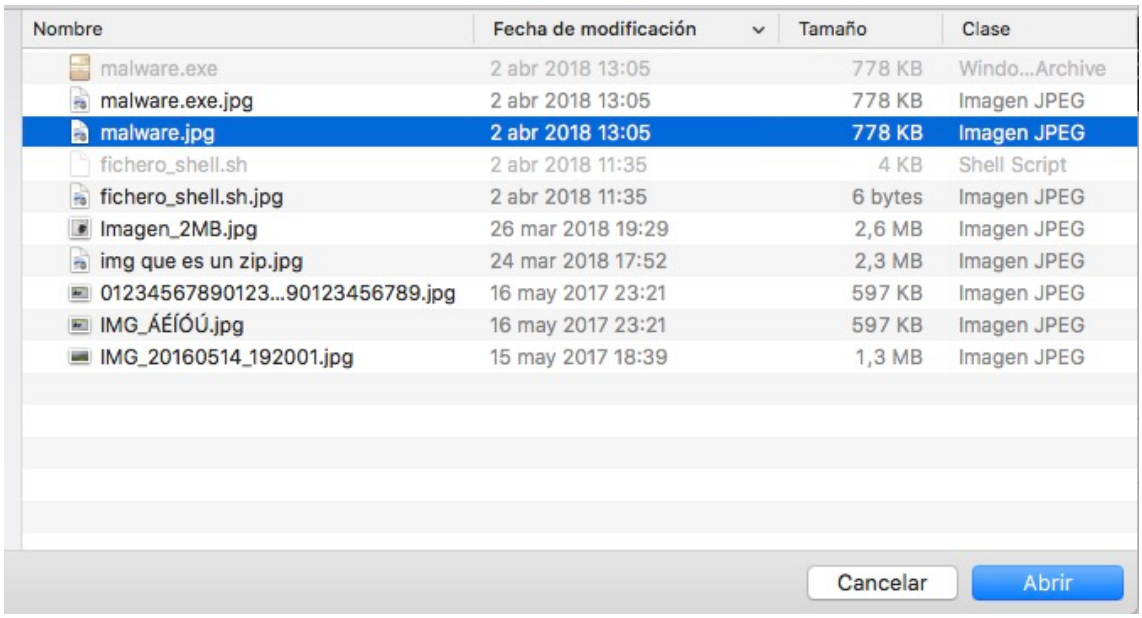

Ilustración 38: Selección del fichero malware.jpg, que en realidad es un .exe

Enviamos el formulario, ahora la aplicación OWASP ZAP intercepta la petición HTTP y nos muestra tanto la cabecera como el cuerpo en un área de texto para poder editarla antes de reenviarla al servidor.

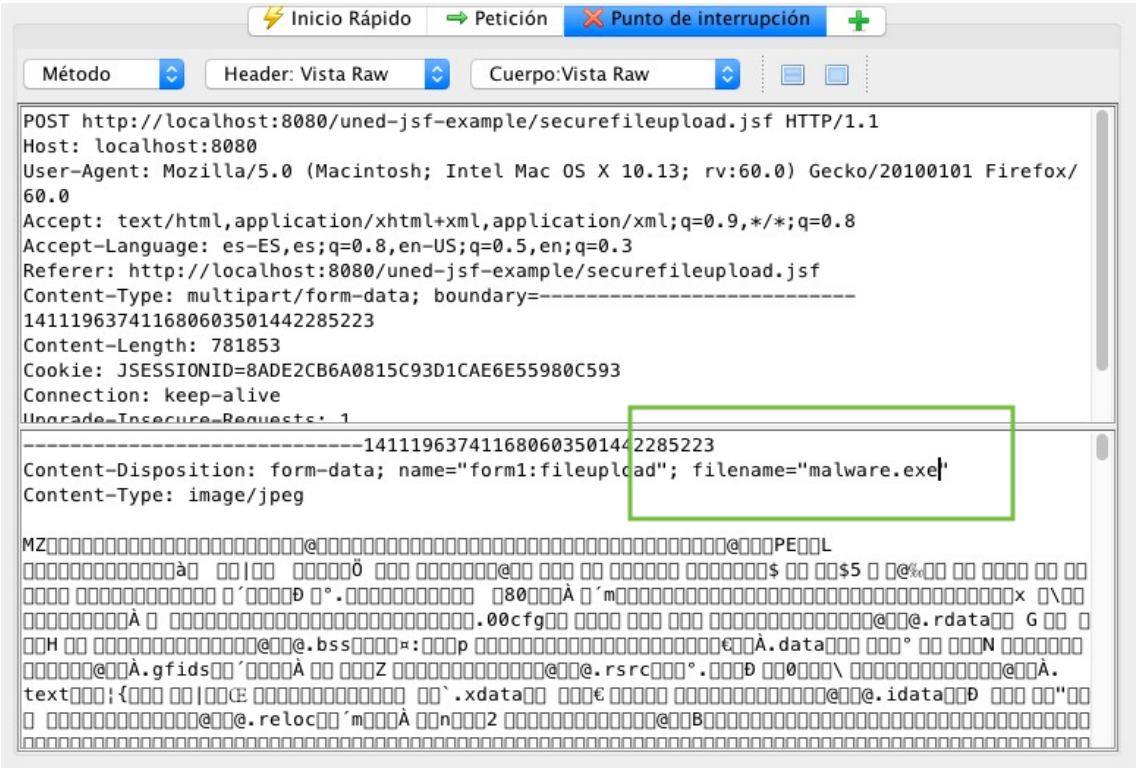

Ilustración 39: Renombramos el nombre del fichero con un proxy web

Reenviamos la petición HTTP una vez modificada y el resultado es el esperado, nos muestra un error de validación y no nos permite adjuntar el fichero con malware:

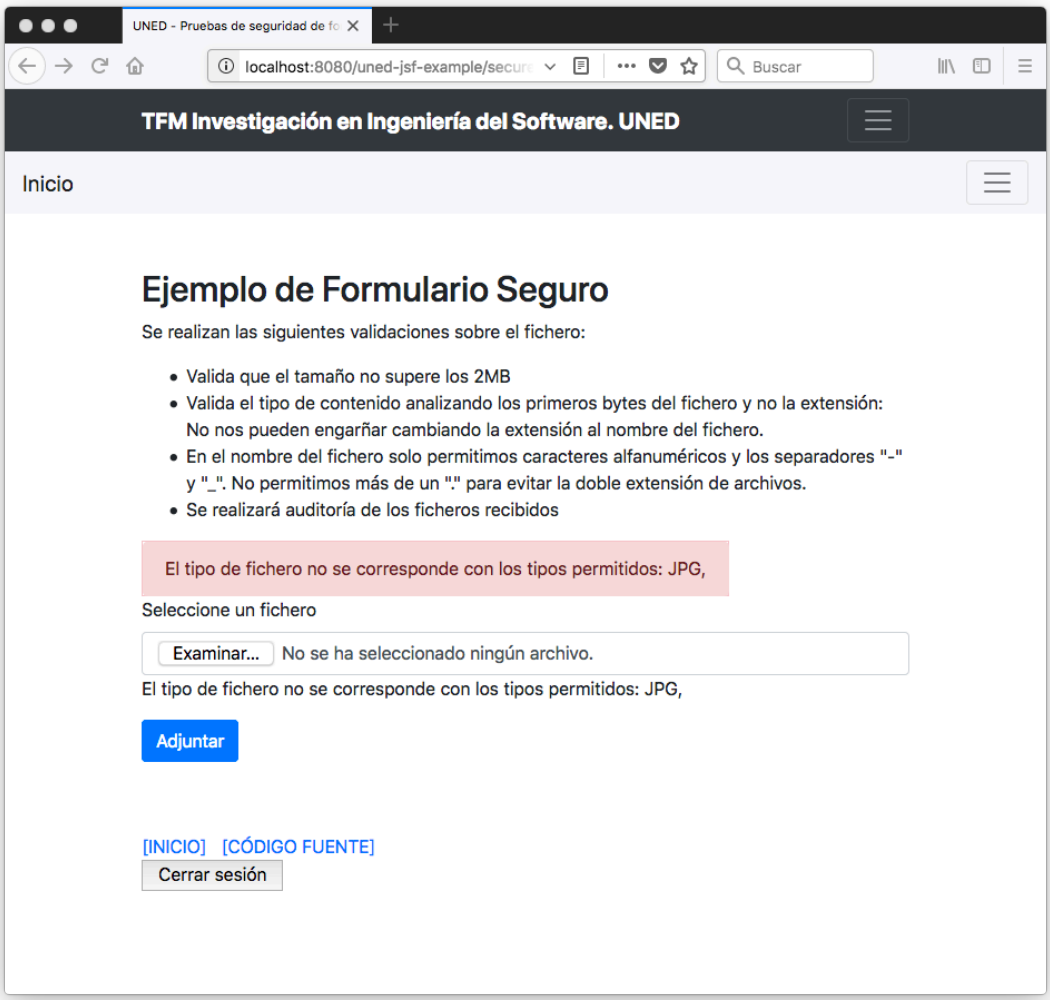

Ilustración 40: Error de validación de un fichero ejecutable

5.2.2 CWE-400: Uncontrolled Resource Consumption ('Resource Exhaustion')

La aplicación no limita adecuadamente la cantidad de recursos que utiliza un usuario, lo que le permite consumir más recursos de los necesarios.

5.2.2.1 Prueba 1: Subida de un fichero de gran tamaño

En este caso para realizar la prueba sí necesitamos que el fichero sea realmente una imagen, ya que nos validará también que el tipo de contenido se corresponda con la

extensión. Debemos de intentar subir una imagen de gran tamaño para comprobar si realmente nos valida el tamaño de los ficheros.

Nuestro formulario seguro está configurado para que acepte ficheros de 10Mb como máximo. Si intentamos subir un fichero de más de 10Mb la aplicación nos deberá de mostrar un mensaje de error.

Accedemos al formulario y seleccionamos una imagen de 17Mb que tenemos en nuestro equipo:

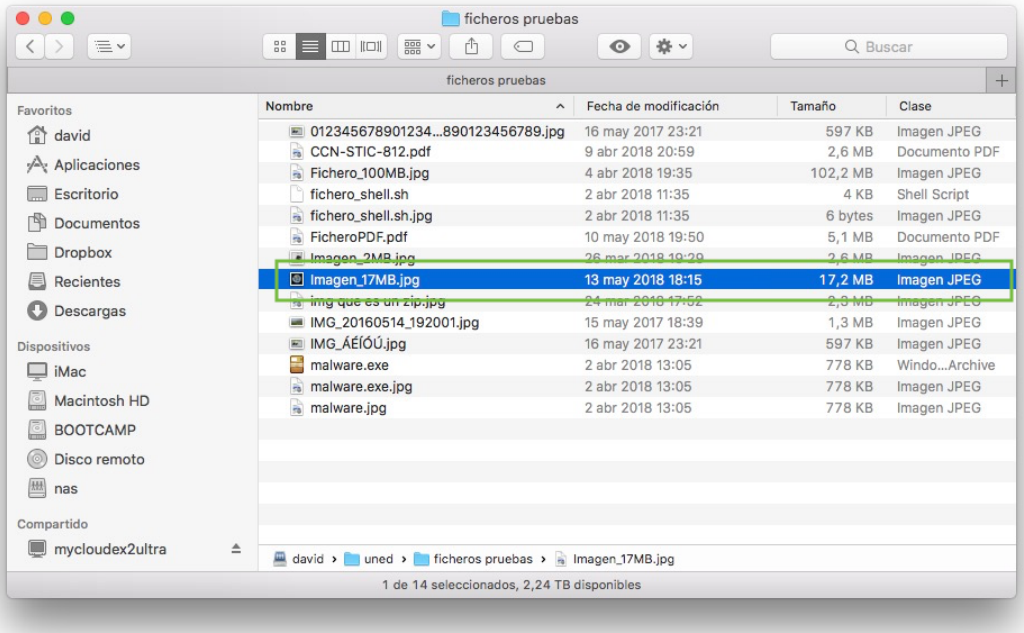

Ilustración 41: Selección de una imagen de 17Mb

Enviamos el formulario y la aplicación nos devuelve un mensaje de error:

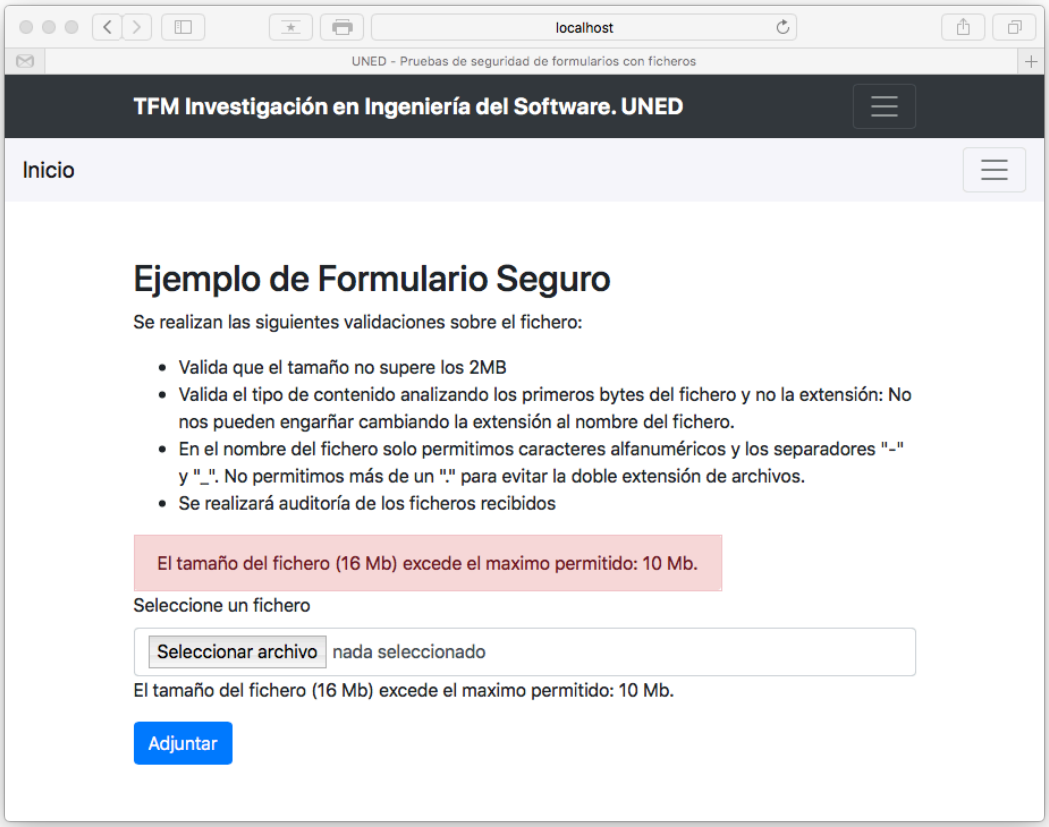

Ilustración 42: Mensaje de error tamaño de fichero excedido

Con esta validación de tamaño y la obligatoriedad de autenticarnos para acceder al formulario reducimos los riesgos de sufrir un ataque de Denegación de Servicio (DOS).

5.2.3 CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

Si nuestra aplicación utiliza un parámetro de entrada para construir la ruta en la que se almacenará un fichero y no validamos correctamente el parámetro de entrada, entonces podríamos permitir a un atacante construir una ruta a un directorio no permitido.

5.2.3.1 Prueba 1: Modificamos el nombre del fichero con una herramienta Proxy web

El fin de la prueba será intentar escribir en el servidor en un directorio diferente al que se ha configurado para almacenar los ficheros que se suben al servidor. Si logramos este ataque podríamos llegar a sobrescribir cualquier fichero del servidor.

Para llevar a cabo este ataque necesitaremos de nuevo una herramienta que nos permita capturar la petición HTTP y editarla antes de enviarla definitivamente al servidor.

Abrimos de nuevo la herramienta OWASP ZAP y lo configuramos en modo escucha para capturar las peticiones HTTP.

Accedemos al formulario seguro y seleccionamos un fichero cualquiera. Enviamos el formulario y capturamos la petición con la herramienta proxy.

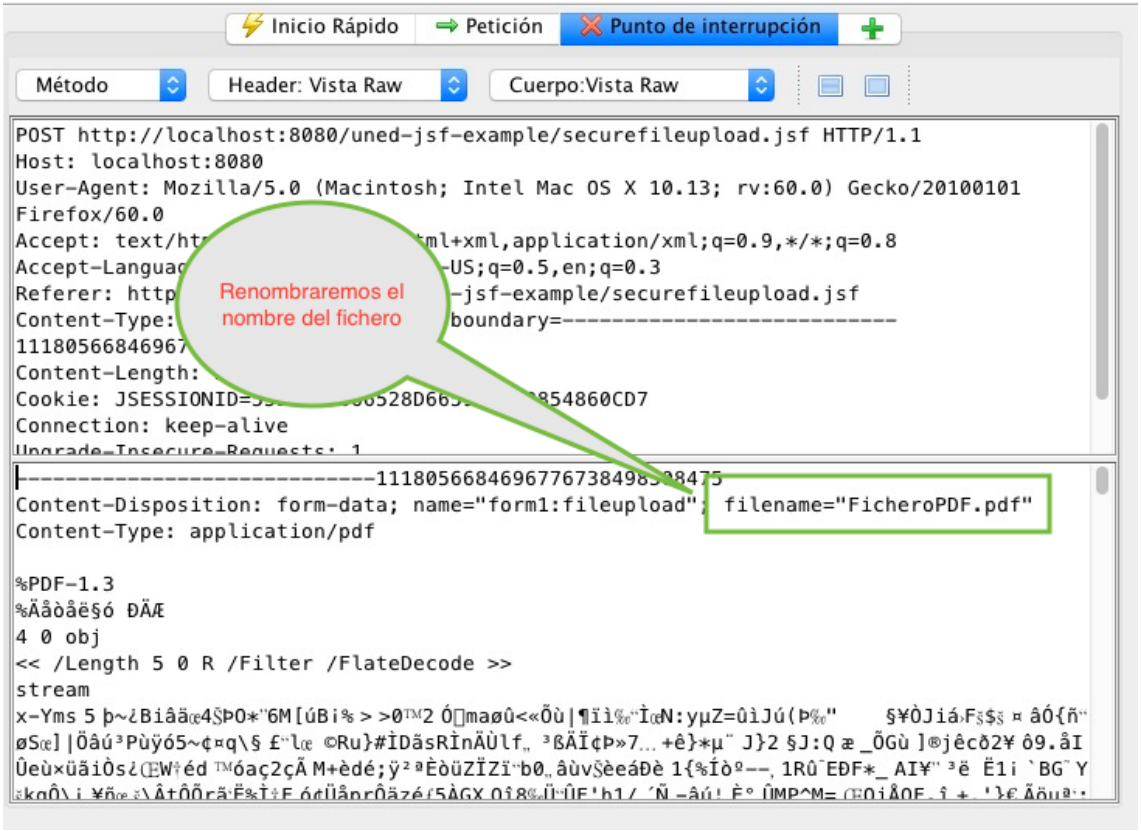

Ilustración 43: Captura de una petición HTTP para editar el nombre del fichero

Modificamos el parámetro “filename” y le establecemos una ruta relativa:

“../../tmp/imagen_uned.jpg”

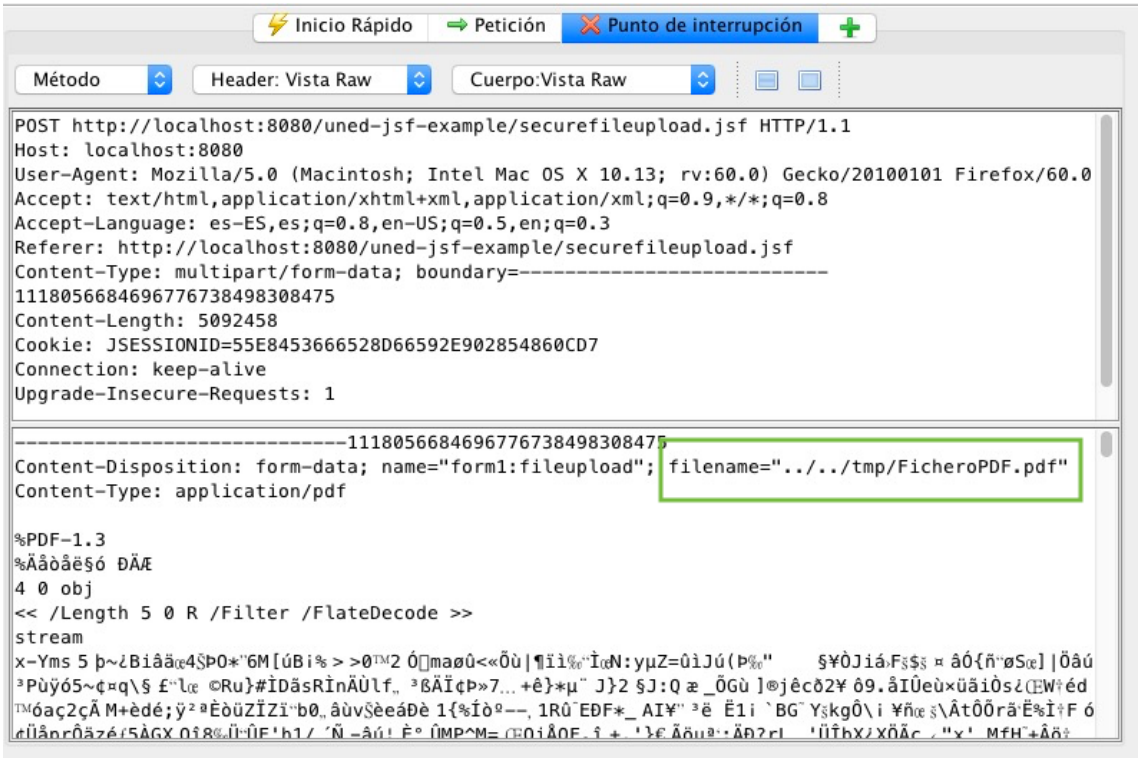

Ilustración 44: Renombrado del nombre del fichero en la herramienta proxy

Reenviamos la petición y la respuesta del servidor es la esperada. Nuestra aplicación no acepta nombres de ficheros con caracteres problemáticos como “..” o “/”.

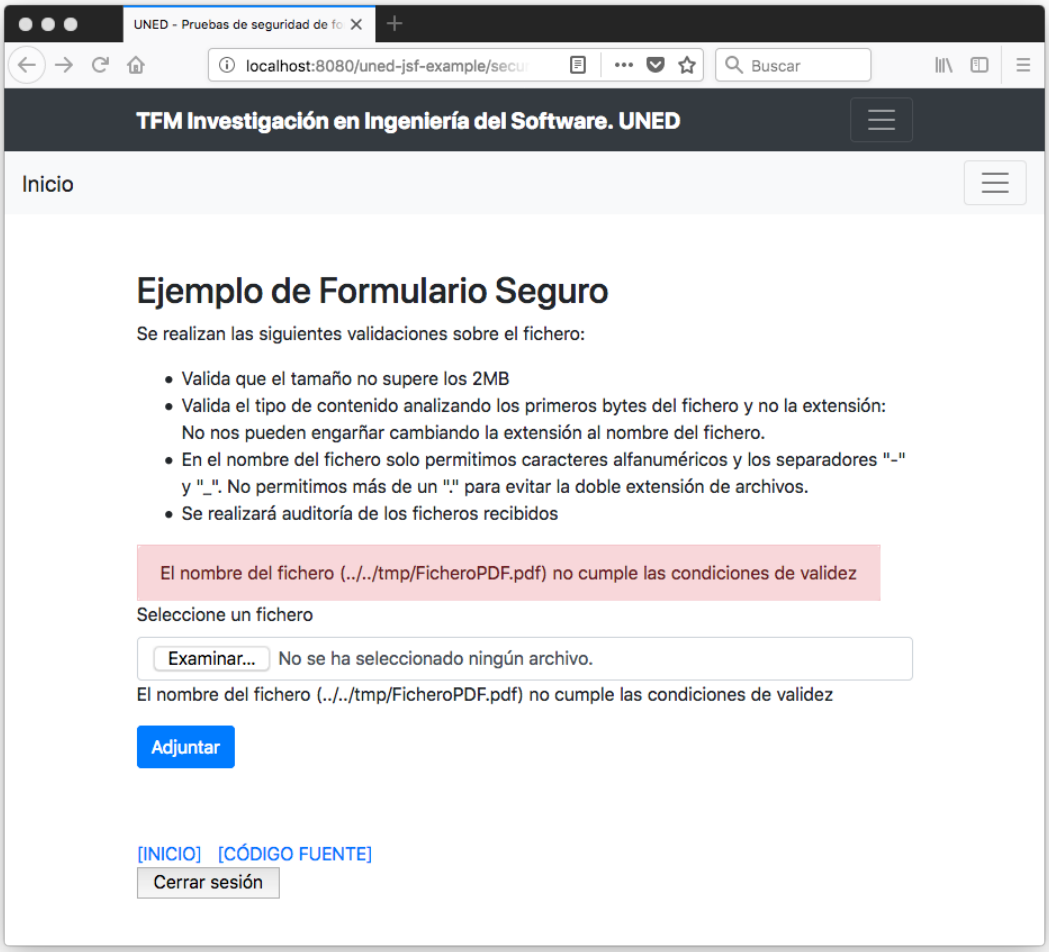

Ilustración 45: Error que indica que el nombre del fichero no cumple las condiciones de validez

Resuelta esta vulnerabilidad, será mucho más difícil que un atacante sobrescriba ficheros sensibles de nuestro servidor o nuestra aplicación.

5.3 Experimentación de la herramienta de auditoría

Tras la realización de las pruebas accedemos a la herramienta de auditoría desde nuestra aplicación prototipo para comprobar si realmente ha almacenado las trazas de auditoría para todos los ficheros, tanto los ficheros validados satisfactoriamente como los ficheros rechazados por no cumplir con las reglas de validación.

Desde el prototipo tenemos un enlace que nos lleva a la herramienta de auditoría. Nos autenticamos, ya que es información sensible y no debería ser visible por cualquier persona y accedemos al buscador.

Vamos a realizar una búsqueda de los últimos ficheros enviados y que no han sido validados satisfactoriamente para comprobar que se han registrado los errores de las pruebas realizadas en los apartados anteriores:

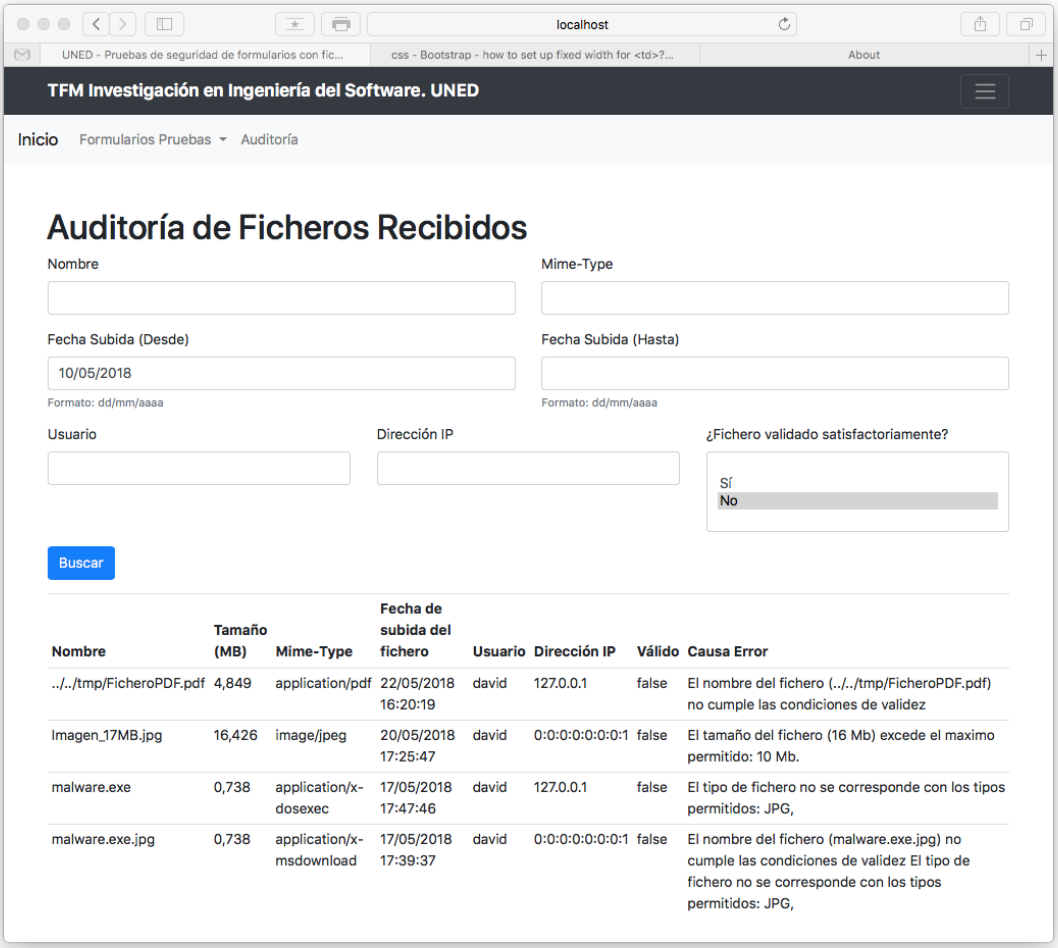

Ilustración 46: Búsqueda de los últimos documentos que no han sido validados satisfactoriamente

La herramienta nos da información muy útil, como la dirección IP desde la que se originó la petición HTTP, la fecha y la hora del intento y una descripción de la causa del error.

Fijémonos en la traza de la siguiente imagen, aunque el fichero tiene extensión ".jpg", la aplicación ha detectado correctamente su tipo de contenido y nos indica que es una aplicación (tipo MIME "application/xmsdownload").

Nombre	Tamaño (MB)	Mime-Type	Fecha de subida del fichero	Usuario	Dirección IP	Válido	Causa Error
../../tmp/FicheroPDF.pdf	4,849	application/pdf	22/05/2018 16:20:19	david	127.0.0.1	false	El nombre del fichero (../../tmp/FicheroPDF.pdf) no cumple las condiciones de validez
Imagen_17MB.jpg	16,426	image/jpeg	20/05/2018 17:25:47	david	0:0:0:0:0:0:0:1	false	El tamaño del fichero (16 Mb) excede el maximo permitido: 10 Mb.
malware.exe	0,738	application/x-dosexec	17/05/2018 17:47:46	david	127.0.0.1	false	El tipo de fichero no se corresponde con los tipos permitidos: JPG,
malware.exe.jpg	0,738	application/x-msdownload	17/05/2018 17:39:37	david	0:0:0:0:0:0:0:1	false	El nombre del fichero (malware.exe.jpg) no cumple las condiciones de validez El tipo de fichero no se corresponde con los tipos permitidos: JPG,

Ilustración 47: Traza de auditoría en la que se puede observar un fichero cuya extensión no se corresponde con el tipo mime esperado

Vamos a realizar ahora otra búsqueda indicando que nos muestre solo los ficheros que se han subido correctamente sin ningún tipo de error:

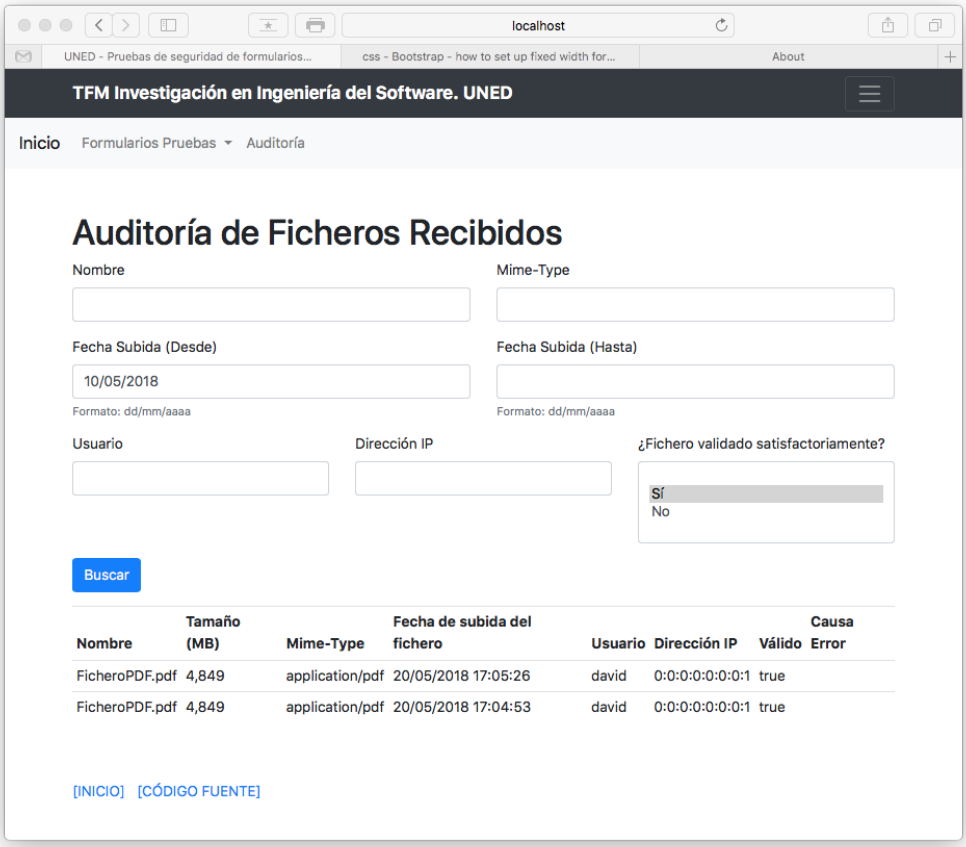

Ilustración 48: Trazas de auditoría de ficheros subidos sin ningún error de validación

Con esta herramienta hemos conseguido los objetivos fijados en la propuesta original, que eran los siguientes:

- Información útil para averiguar la autoría de un posible ataque
- Información para mejorar continuamente la librería de validación. En el supuesto caso de que nuestra librería no hubiera detectado el error de validación podríamos analizar el fichero para mejorar las validaciones.
- Información para generar estadísticas, por ejemplo:
    - ¿cuántos ataques hemos sufrido en un año?
    - ¿De qué zonas del mundo son las IPs desde la que nos hacen los ataques?
    - Etc.

5.4 Experimentación de la herramienta de validación de código fuente

Vamos a probar que nuestra herramienta de validación de código nos muestra un error en el punto donde tengamos un formulario web en JSF donde no estemos aplicando las validaciones implementadas anteriormente.

También debemos de probar que no nos da falsos positivos, es decir, no nos debe de marcar ningún error en nuestro formulario web seguro, en el que tenemos aplicadas las validaciones.

Accedemos a la herramienta Eclipse, y en el proyecto que queramos analizar pulsamos la opción “SpotBugs >> FindBugs”.

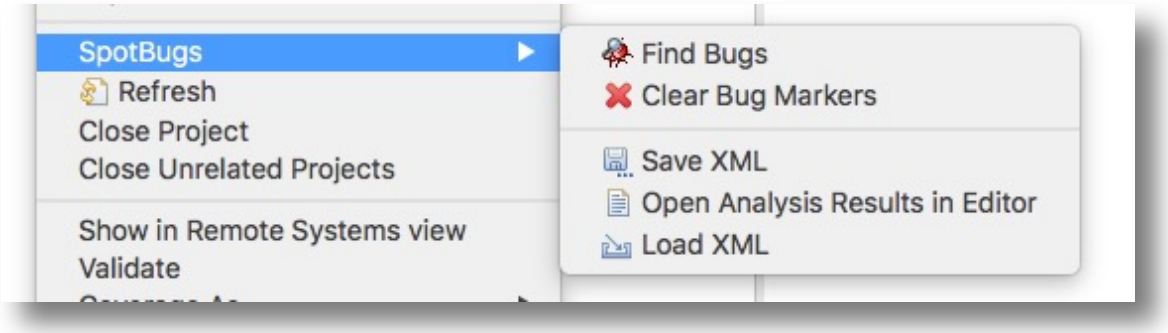

Ilustración 49: Opción de menú para analizar nuestro proyecto en búsqueda de vulnerabilidades

Si analizáramos nuestro prototipo debería de marcarnos solo un error, y este error estaría en la clase que implementa el formulario no seguro.

Abrimos la perspectiva de SpotBugs y comprobamos que solo hay un resultado, del tipo “CWE-434: Unrestricted Upload of File with Dangerous Type”, que es el error que detecta nuestra herramienta:

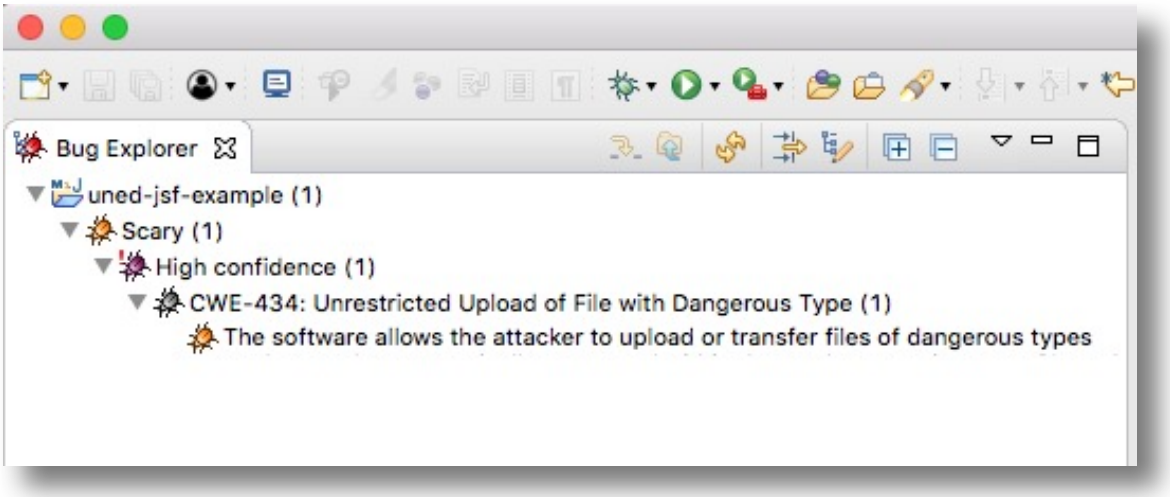

Ilustración 50: Errores de validación en nuestro proyecto prototipo

Si pulsamos sobre el error nos abre la clase en la que se encuentra el error y además nos marca la línea de código donde existe el problema:

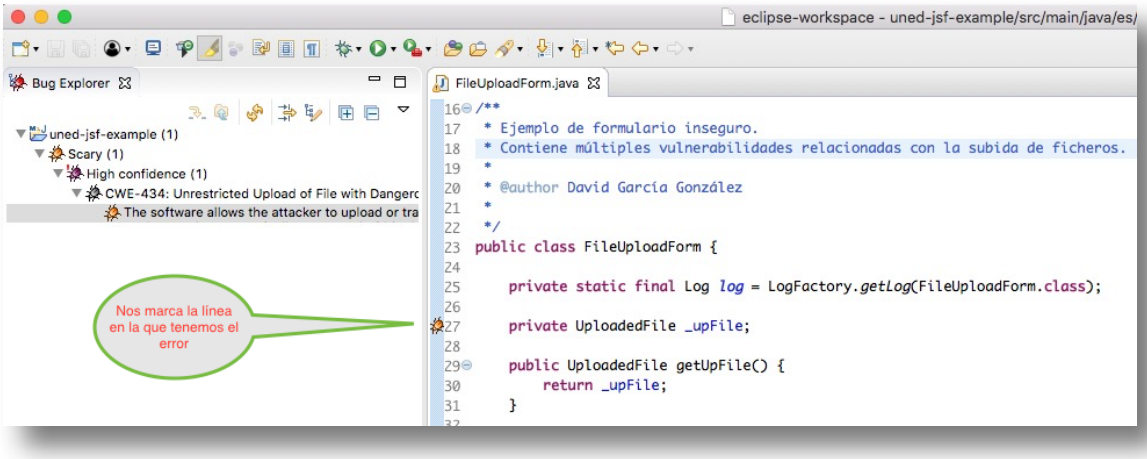

Ilustración 51: Marca de error de validación en un fichero fuente de nuestro proyecto

En la misma pantalla, se dan instrucciones al programador del error:

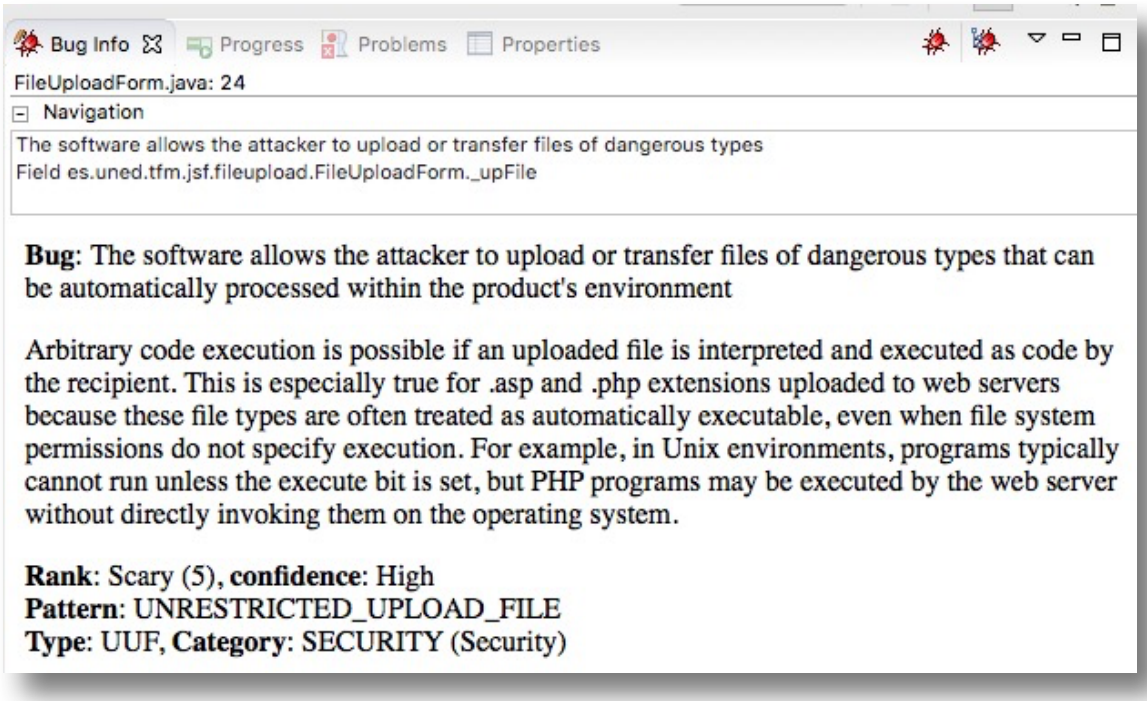

Ilustración 52: Descripción detallada del error

Con esta información el programador puede detectar estas vulnerabilidades antes de que el código se ponga en un sistema en producción.
Además, es importante resaltar que no ha reportado ninguna incidencia en la clase que implementa el formulario seguro. Esto es importante porque si lo hubiera hecho, estaríamos ante un falso positivo. Si nuestra herramienta reportara muchos falsos positivos perdería su eficacia, ya que los programadores no confiarían en su correcto funcionamiento y dejarían de utilizarla.

6 Conclusiones y trabajos futuros

Una primera conclusión que podemos sacar tras este estudio sería la sensación de una falta de preocupación por esta vulnerabilidad. Hemos encontrado muy poca información que ofreciera soluciones para mitigar estos problemas, tan solo páginas especializadas con OWASP o CWE y algún blog de algún programador que prácticamente repiten las mismas recomendaciones que describen en OWASP y CWE.

Tampoco hemos encontrado librerías Java que incluyan una validación completa y exhaustiva de ficheros. La famosa librería ESAPI de OWASP incluye un método con una validación muy superficial e incompleta para utilizar en un sistema en producción.

Una explicación a la falta de sensibilidad podría ser la falta de conocimiento de los riesgos que asumimos al no tener estas vulnerabilidades controladas y minimizadas.

Otra explicación sería que asumimos este riesgo para ganar en usabilidad. Si aplicamos estas validaciones podría resultar frustrante para los usuarios que la aplicación le empiece a dar errores del tipo “el nombre del fichero no es correcto”, “el tamaño del archivo es demasiado grande”, “el formato del fichero no es válido”, etc.…

Precisamente en este campo es donde podríamos realizar mejoras y trabajos futuros. Deberíamos de mejorar la herramienta para que muestre los mínimos errores de validación al usuario, pero siempre manteniendo la misma exigencia en seguridad.

Otra mejora en la que podría haber interés sería incluir estas validaciones en los propios frameworks de desarrollo, especificaciones y herramientas. Sería seguir el famoso concepto de **seguridad por defecto**. Si por ejemplo JSF realizara las validaciones sobre los ficheros sin configurar nada, la mayoría de aplicaciones serían mucho más seguras. Debería de permitir también relajar estas restricciones y adaptarlas a las necesidades de cada proyecto, pero en este caso ya sería el desarrollador el que tendría que definir sus necesidades, por lo tanto, ya no sería una cuestión de falta de conocimiento o de simple olvido en la planificación.

Y por último, la principal faceta en la que se debería de trabajar en el futuro es en la concienciación de los programadores y responsables de proyecto TIC de la importancia de la seguridad en las aplicaciones web. Como hemos demostrado en este documento, es tan sencillo como utilizar una herramienta proxy web para introducir malware en una organización o modificar el comportamiento de cualquier aplicación. Si hiciéramos una

encuesta, la mayoría de personas involucradas en un proyecto web no sabría que están expuestos a sufrir un ataque tan grave solo por contener un formulario con ficheros en su aplicación web.

7 Bibliografía

Apache Software Foundation. (2018). *Apache Tika - a content analysis toolkit*. Recuperado el 2018, de Apache Tika: http://tika.apache.org/index.html

Carnegie Mellon University, Software Engineering Institute. (noviembre de 2017). *IDS56-J. Prevent arbitrary file upload*. Recuperado el 2018, de SEI External Wiki: https://wiki.sei.cmu.edu/confluence/display/java/IDS56-J.+Prevent+arbitrary+file+upload

CCN-CERT Centro Criptológico Nacional. (Abril de 2016). *Informe CCN-CERT IA-09/16 de Ciberamenazas 2015/Tendencias 2016 Resumen Ejecutivo.* Recuperado el Mayo de 2018, de CCN-CERT: https://www.ccn-cert.cni.es/informes/informes-ccn-cert-publicos/1483-ccn-cert-ia-0916-ciberamenazas-2015-tendencias-2016-resumen-ejecutivo/file.html

CCN-CERT Centro Criptológico Nacional. (Mayo de 2018). *Informe CCN-CERT IA-09/18 de Ciberamenazas y Tendencias 2018. Resumen Ejecutivo 2018.* Recuperado el 2018, de CCN-CERT: https://www.ccn-cert.cni.es/informes/informes-ccn-cert-publicos/2856-ccn-cert-ia-09-18-ciberamenazas-y-tendencias-2018-resumen-ejecutivo-2018/file.html

CCN-CERT Centro Criptológico Nacional. (Mayo de 2018). *Informe CCN-CERT IA-09/18 de Ciberamenazas y Tendencias. Edición 2018.* Recuperado el Junio de 2018, de CCN-CERT: https://www.ccn-cert.cni.es/informes/informes-ccn-cert-publicos/2835-ccn-cert-ia-09-18-ciberamenzas-y-tendencias-edicion-2018-1/file.html

CWE. (marzo de 2018). *CWE-434: Unrestricted Upload of File with Dangerous Type*. Recuperado el 2018, de CWE Common Weakness Enumeration: http://cwe.mitre.org/data/definitions/434.html

Dalili, S. (2012). *File UploadersVulnerabilities File in the hole!* Recuperado el 2018, de Chair for network and data security: https://www.nds.rub.de/media/attachments/files/2012/11/File-in-the-hole.pdf

F. L., D. M., R. S., D. S., & D. S. (2014). *Java Coding Guidelines: 75 recommendations for realiable and secure programs.* Addison-Wesley.

J. L. (2009). *[OWASP-ESAPI] File Content Validation*. Recuperado el 2018, de https://lists.owasp.org/pipermail/owasp-esapi/2009-May/000527.html

M. M., & L. R. (2010). *Secure and resilient software development.* CRC Press Taylor & Francis Group.

MITRE Corporation. (Junio de 2018). *Security Vulnerabilities Related To CWE-434*. Recuperado el Junio de 2018, de CVE Details. The ultimate security vulnerability

datasource: https://www.cvedetails.com/vulnerability-list/cweid-434/vulnerabilities.html

OWASP. (2017). *Unrestricted File Upload*. Recuperado el 2018, de OWASP: https://www.owasp.org/index.php/Unrestricted_File_Upload

OWASP. (mayo de 2018). *Protect FileUpload Against Malicious File*. Recuperado el 2018, de OWASP: https://www.owasp.org/index.php/Protect_FileUpload_Against_Malicious_File

SANS Institute. (2010). *Getting Owned By Malicious PDF - Analysis.* Recuperado el 2018, de SANS. The most trusted source for information security training, certification, and research.: https://www.sans.org/reading-room/whitepapers/malicious/owned-malicious-pdf-analysis-33443

8 Anexos

8.1 Código fuente de la librería de validación

8.1.1 POM.XML

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>es.uned.tfm.dss</groupId>
  <artifactId>secureapi</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>secureapi</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <myfaces.version>1.1.10</myfaces.version>
    <tomahawk.version>1.1.14</tomahawk.version>
    <tika.version>1.17</tika.version>
    <spring.version>3.2.18.RELEASE</spring.version>
    <acegi-security.version>1.0.4</acegi-security.version>
    <hibernate.version>3.6.10.Final</hibernate.version>
    <itextpdf.version>5.5.13</itextpdf.version>
    <commons-logging.version>1.1</commons-logging.version>
    <log4j.version>1.2.17</log4j.version>
  </properties>

  <dependencies>

    <dependency>
      <groupId>org.apache.myfaces.tomahawk</groupId>
      <artifactId>tomahawk</artifactId>
      <version>${tomahawk.version}</version>
    </dependency>

    <dependency>
      <groupId>org.apache.myfaces.core</groupId>
      <artifactId>myfaces-api</artifactId>
      <version>${myfaces.version}</version>
    </dependency>

    <dependency>
      <groupId>org.apache.tika</groupId>
      <artifactId>tika-core</artifactId>
      <version>${tika.version}</version>
    </dependency>
```

```
    <dependency>
      <groupId>org.acegisecurity</groupId>
      <artifactId>acegi-security</artifactId>
      <version>${acegi-security.version}</version>
    </dependency>

    <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-entitymanager</artifactId>
      <version>${hibernate.version}</version>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-jdbc</artifactId>
      <version>${spring.version}</version>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>${spring.version}</version>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-orm</artifactId>
      <version>${spring.version}</version>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-web</artifactId>
      <version>${spring.version}</version>
    </dependency>

    <dependency>
      <groupId>com.itextpdf</groupId>
      <artifactId>itextpdf</artifactId>
      <version>${itextpdf.version}</version>
    </dependency>

    <dependency>
      <groupId>commons-logging</groupId>
      <artifactId>commons-logging</artifactId>
      <version>${commons-logging.version}</version>
    </dependency>

    <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>${log4j.version}</version>
    </dependency>

    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
```

```
      <version>5.1.6</version>
      <scope>provided</scope>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-test</artifactId>
      <version>${spring.version}</version>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>1.7</source>
          <target>1.7</target>
          <encoding>UTF-8</encoding>
        </configuration>
      </plugin>
    </plugins>
  </build>

</project>
```

8.1.2 AuditoriaFicherosDao.java

```
package es.uned.tfm.dss.secureapi.dao;

import java.util.List;

import es.uned.tfm.dss.secureapi.model.AuditoriaFicheros;
import es.uned.tfm.dss.secureapi.vo.BusquedaAuditoriaFicherosVO;

/**
 * Interfaz que define las operaciones relacionadas con el almacenamiento en
base de datos de las trazas de auditoría
 *
 * @author David García González
 *
 */
public interface AuditoriaFicherosDao {

	/**
	* Almacena una traza de auditoría en base de datos
	*
	* @param auditoriaFicheros
	* @return La traza almacenada
	*/
	public AuditoriaFicheros save(AuditoriaFicheros auditoriaFicheros);

	/**
	* Realiza una búsqueda de trazas de auditoría en base de datos
	*
	* @param busquedaAuditoriaFicherosVO Los parámetros de búsqueda
	* @return Listado con las trazas de auditoría que se ajustan a los
parámetros de búsqueda
	*/
	public List<AuditoriaFicheros> find(BusquedaAuditoriaFicherosVO
busquedaAuditoriaFicherosVO);

}
```

8.1.3 AuditoriaFicherosDaoImpl

```
package es.uned.tfm.dss.secureapi.dao.impl;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;

import es.uned.tfm.dss.secureapi.dao.AuditoriaFicherosDao;
import es.uned.tfm.dss.secureapi.model.AuditoriaFicheros;
import es.uned.tfm.dss.secureapi.vo.BusquedaAuditoriaFicherosVO;

/**
 * Implementación por defecto de las operaciones relacionadas con el
```

```
 * almacenamiento en base de datos de las trazas de auditoría
 *
 * @author David García González
 *
 */
public class AuditoriaFicherosDaoImpl implements AuditoriaFicherosDao {

	@PersistenceContext
	private EntityManager entityManager;

	/**
	* @{inheritDoc}
	*/
	public AuditoriaFicheros save(AuditoriaFicheros auditoriaFicheros) {

		this.entityManager.persist(auditoriaFicheros);
		this.entityManager.flush();

		return auditoriaFicheros;

	}

	/**
	* @{inheritDoc}
	*/
	@Override
	public List<AuditoriaFicheros> find(BusquedaAuditoriaFicherosVO
busquedaAuditoriaFicherosVO) {

		final Query query =
entityManager.createNamedQuery("busquedaAuditoriaFicheros");
		query.setParameter("nombre",
getLikeExpression(busquedaAuditoriaFicherosVO.getNombre()));
		query.setParameter("fechaSubidaDesde",
busquedaAuditoriaFicherosVO.getFechaSubidaDesde());
		query.setParameter("fechaSubidaHasta",
busquedaAuditoriaFicherosVO.getFechaSubidaHasta());
		query.setParameter("ipSubida",
getLikeExpression(busquedaAuditoriaFicherosVO.getIpSubida()));
		query.setParameter("usuarioSubida",
getLikeExpression(busquedaAuditoriaFicherosVO.getUsuarioSubida()));
		query.setParameter("mimeType",
getLikeExpression(busquedaAuditoriaFicherosVO.getMimeType()));
		query.setParameter("valido",
busquedaAuditoriaFicherosVO.getValido());

		@SuppressWarnings("unchecked")
		List<AuditoriaFicheros> results = query.getResultList();
		return results;
	}

	private String getLikeExpression(String valor) {
		if (valor != null) {
			return "%" + valor + "%";
		} else {
			return valor;
		}
```

```
	}

}
```

8.1.4 DocumentDetector.java

```
package es.uned.tfm.dss.secureapi.detector;

import org.apache.myfaces.custom.fileupload.UploadedFile;

/**
 * Interfaz que define las operaciones que deberan de implementar los
 * validadores de cada tipo de documento
 *
 * @author David García González
 *
 */
public interface DocumentDetector {

	/**
	* Valida si el documento es seguro
	*
	* @param file
	*            El fichero a validar
	* @return true: Si es seguro false: Si no es seguro
	*/
	boolean isSafe(UploadedFile file);

}
```

8.1.5 DocumentDetectorFactory.java

```
package es.uned.tfm.dss.secureapi.detector;

import es.uned.tfm.dss.secureapi.detector.impl.BasicDocumentDetectorImpl;
import es.uned.tfm.dss.secureapi.detector.impl.PdfDocumentDetectorImpl;

/**
 * Factoría que construye el validador de documentos necesario para cada tipo
de
 * fichero
 *
 * @author David García González
 *
 */
public class DocumentDetectorFactory {

	/**
	* Construye un validador de documentos a partir del mime-type del
fichero que
```

```
	 * queramos validar.
	 *
	 * @param mimeType
	 *            El tipo mime del fichero a validar
	 *
	 * @return El detector de documentos correspondiente para el tipo mime
	 */
	public static DocumentDetector getDocumentDetector(String mimeType) {
		if ("application/pdf".equalsIgnoreCase(mimeType)) {
			return new PdfDocumentDetectorImpl();
		}

		return new BasicDocumentDetectorImpl();

	}

}
```

8.1.6 BasicDocumentDetectorImpl.java

```
package es.uned.tfm.dss.secureapi.detector.impl;

import org.apache.myfaces.custom.fileupload.UploadedFile;

import es.uned.tfm.dss.secureapi.detector.DocumentDetector;

/**
 * Implementación básica de un validador de documentos. Devuelve que es válido
 * si el fichero no es vacío.
 *
 * @author David García González
 *
 */
public class BasicDocumentDetectorImpl implements DocumentDetector {

  /**
   * {@inheritDoc}
   */
  @Override
  public boolean isSafe(UploadedFile file) {
    if ((file != null) && file.getSize() > 0) {
      return true;
    } else {
      return false;
    }
  }

}
```

8.1.7 PdfDocumentDetectorImpl.java

```
package es.uned.tfm.dss.secureapi.detector.impl;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.myfaces.custom.fileupload.UploadedFile;

import com.itextpdf.text.pdf.PdfArray;
import com.itextpdf.text.pdf.PdfDictionary;
import com.itextpdf.text.pdf.PdfName;
import com.itextpdf.text.pdf.PdfReader;

import es.uned.tfm.dss.secureapi.detector.DocumentDetector;

/**
 * Implementación del validador de documentos específica para ficheros PDF.
 *
 * Devuelve que es válido si el PDF no contiene código Javascript inscrustado.
 *
 * @author David García González
 *
 */
public class PdfDocumentDetectorImpl implements DocumentDetector {

  /** LOGGER */
  private static final Log LOG =
LogFactory.getLog(PdfDocumentDetectorImpl.class);

  /**
   * {@inheritDoc}
   */
  @Override
  public boolean isSafe(UploadedFile file) {
    boolean safeState = false;
    try {
      if ((file != null) && file.getSize() > 0) {
        // Load stream in PDF parser
        // If the stream is not a PDF then exception will be throwed
        // here and safe state will be set to FALSE
        PdfReader reader = new PdfReader(file.getBytes());
        // Check 1:
        // Detect if the document contains any JavaScript code
        String jsCode = reader.getJavaScript();
        if (jsCode == null) {
          // OK no JS code then when pass to check 2:
          // Detect if the document has any embedded files
          PdfDictionary root = reader.getCatalog();
          PdfDictionary names = root.getAsDict(PdfName.NAMES);
          PdfArray namesArray = null;
          if (names != null) {
            PdfDictionary embeddedFiles =
names.getAsDict(PdfName.EMBEDDEDFILES);
            namesArray = embeddedFiles.getAsArray(PdfName.NAMES);
          }
          // Get safe state from number of embedded files
```

```
          safeState = ((namesArray == null) || namesArray.isEmpty());
        }
      }
    } catch (Exception e) {
      safeState = false;
      LOG.warn("Error mientras se analiza el PDF", e);
    }
    return safeState;
  }

}
```

8.1.8 InsecureFileUploadedException.java

```
package es.uned.tfm.dss.secureapi.exceptions;
```

```
/**
 * Excepcion que se produce cuando un fichero no valida las restricciones de
 * seguridad
 *
 * @author David García González
 *
 */
public class InsecureFileUploadedException extends Exception {

  private static final long serialVersionUID = 4433981127811822784L;

  private String code;

  public InsecureFileUploadedException(String code, String message) {
    super(message);
    this.code = code;
  }

  public String getCode() {
    return code;
  }

}
```

8.1.9 AuditoriaFicheros.java

```
package es.uned.tfm.dss.secureapi.model;

import java.text.NumberFormat;
import java.util.Date;
import java.util.Locale;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
```

```java
import javax.persistence.Id;

/**
 * Entidad JPA que representa una traza de auditoría.
 *
 * @author David García González
 *
 */
@Entity(name = "uned_auditoria_ficheros")
public class AuditoriaFicheros {

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;

  private String nombre;

  private Long tamanio;

  private String mimeType;

  private Date fechaSubida;

  private String usuarioSubida;

  private String ipSubida;

  private String hashFichero;

  private String algoritmoHash;

  private Boolean valido;

  private String causaError;

  public String getNombre() {
    return nombre;
  }

  public void setNombre(String nombre) {
    this.nombre = nombre;
  }

  public Long getTamanio() {
    return tamanio;
  }

  public void setTamanio(Long tamanio) {
    this.tamanio = tamanio;
  }

  public String getTamanioInKB() {
    if (tamanio != null) {
      float kb = tamanio / 1024f;
      NumberFormat nf = NumberFormat.getInstance(new Locale("es", "ES"));

      return nf.format(kb);
```

```
    }
    return null;
  }

  public String getTamanioInMB() {
    if (tamanio != null) {
      float mb = tamanio / (1024f * 1024f);
      NumberFormat nf = NumberFormat.getInstance(new Locale("es", "ES"));

      return nf.format(mb);
    }
    return null;
  }

  public String getMimeType() {
    return mimeType;
  }

  public void setMimeType(String mimeType) {
    this.mimeType = mimeType;
  }

  public Date getFechaSubida() {
    return fechaSubida;
  }

  public void setFechaSubida(Date fechaSubida) {
    this.fechaSubida = fechaSubida;
  }

  public String getUsuarioSubida() {
    return usuarioSubida;
  }

  public void setUsuarioSubida(String usuarioSubida) {
    this.usuarioSubida = usuarioSubida;
  }

  public String getIpSubida() {
    return ipSubida;
  }

  public void setIpSubida(String ipSubida) {
    this.ipSubida = ipSubida;
  }

  public String getHashFichero() {
    return hashFichero;
  }

  public void setHashFichero(String hashFichero) {
    this.hashFichero = hashFichero;
  }

  public String getAlgoritmoHash() {
    return algoritmoHash;
  }
```

```
  public void setAlgoritmoHash(String algoritmoHash) {
    this.algoritmoHash = algoritmoHash;
  }

  public Long getId() {
    return id;
  }

  public Boolean getValido() {
    return valido;
  }

  public void setValido(Boolean valido) {
    this.valido = valido;
  }

  public String getCausaError() {
    return causaError;
  }

  public void setCausaError(String causaError) {
    this.causaError = causaError;
  }

  @Override
  public String toString() {
    StringBuilder builder = new StringBuilder();
    builder.append("AuditoriaFicheros [");
    if (id != null)
      builder.append("id=").append(id).append(", ");
    if (nombre != null)
      builder.append("nombre=").append(nombre).append(", ");
    if (tamanio != null)
      builder.append("tamanio=").append(tamanio).append(", ");
    if (mimeType != null)
      builder.append("mimeType=").append(mimeType).append(", ");
    if (fechaSubida != null)
      builder.append("fechaSubida=").append(fechaSubida).append(", ");
    if (usuarioSubida != null)
      builder.append("usuarioSubida=").append(usuarioSubida).append(", ");
    if (ipSubida != null)
      builder.append("ipSubida=").append(ipSubida).append(", ");
    if (hashFichero != null)
      builder.append("hashFichero=").append(hashFichero).append(", ");
    if (algoritmoHash != null)
      builder.append("algoritmoHash=").append(algoritmoHash).append(", ");
    if (valido != null)
      builder.append("valido=").append(valido).append(", ");
    if (causaError != null)
      builder.append("causaError=").append(causaError);
    builder.append("]");
    return builder.toString();
  }

}
```

8.1.10 AuditoriaFicherosService.java

```
package es.uned.tfm.dss.secureapi.service;

import java.util.List;

import es.uned.tfm.dss.secureapi.model.AuditoriaFicheros;
import es.uned.tfm.dss.secureapi.vo.BusquedaAuditoriaFicherosVO;

/**
 * Interfaz que define las operaciones del servicio de el almacenamiento de
las
 * trazas de auditoría
 *
 * @author David García González
 *
 */
public interface AuditoriaFicherosService {

  /**
   * Almacena una traza de auditoría
   *
   * @param auditoriaFicheros
   * @return La traza almacenada
   */
  public AuditoriaFicheros save(AuditoriaFicheros auditoriaFicheros);

  /**
   * Realiza una búsqueda de trazas de auditoría
   *
   * @param busquedaAuditoriaFicherosVO
   *          Los parámetros de búsqueda
   * @return Listado con las trazas de auditoría que se ajustan a los
parámetros
   *         de búsqueda
   */
  public List<AuditoriaFicheros> find(BusquedaAuditoriaFicherosVO
busquedaAuditoriaFicherosVO);

}
```

8.1.11 AuditoriaFicherosServiceImpl.java

```
package es.uned.tfm.dss.secureapi.service.impl;

import java.util.List;

import org.springframework.transaction.annotation.Transactional;
```

```
import es.uned.tfm.dss.secureapi.dao.AuditoriaFicherosDao;
import es.uned.tfm.dss.secureapi.model.AuditoriaFicheros;
import es.uned.tfm.dss.secureapi.service.AuditoriaFicherosService;
import es.uned.tfm.dss.secureapi.vo.BusquedaAuditoriaFicherosVO;

/**
 * Implementación por defecto de las operaciones relacionadas con el
 * almacenamiento de las trazas de auditoría
 *
 * @author David García González
 *
 */
public class AuditoriaFicherosServiceImpl implements AuditoriaFicherosService
{

  // DAO que se encargará del almacenamiento en BBDD de las trazas de
auditoría
  AuditoriaFicherosDao auditoriaFicherosDao;

  /**
   * {@inheritDoc}
   */
  @Transactional
  public AuditoriaFicheros save(AuditoriaFicheros auditoriaFicheros) {
    return auditoriaFicherosDao.save(auditoriaFicheros);
  }

  /**
   * {@inheritDoc}
   */
  @Transactional
  public List<AuditoriaFicheros> find(BusquedaAuditoriaFicherosVO
busquedaAuditoriaFicherosVO) {
    return auditoriaFicherosDao.find(busquedaAuditoriaFicherosVO);
  }

  public AuditoriaFicherosDao getAuditoriaFicherosDao() {
    return auditoriaFicherosDao;
  }

  public void setAuditoriaFicherosDao(AuditoriaFicherosDao
auditoriaFicherosDao) {
    this.auditoriaFicherosDao = auditoriaFicherosDao;
  }

}
```

8.1.12 ErrorValidacion.java

```
package es.uned.tfm.dss.secureapi.validators.jsf;

/**
 * Clase que representa un error de validación
```

```
 *
 * @author David García González
 *
 */
public class ErrorValidacion {

  // Código del error
  private String codigoError;

  // Descripción del error
  private String mensaje;

  public ErrorValidacion(String codigoError, String mensaje) {
    this.codigoError = codigoError;
    this.mensaje = mensaje;
  }

  public String getCodigoError() {
    return codigoError;
  }

  public void setCodigoError(String codigoError) {
    this.codigoError = codigoError;
  }

  public String getMensaje() {
    return mensaje;
  }

  public void setMensaje(String mensaje) {
    this.mensaje = mensaje;
  }

}
```

8.1.13 TipoFichero.java

```
package es.uned.tfm.dss.secureapi.validators.jsf;
```

```
/**
 * Enumerado con los tipos de fichero que permitira validar la aplicación
 *
 * @author David García González
 *
 */
public enum TipoFichero {
  PDF("PDF", "application/pdf"),
  JPG("JPG", "image/jpeg");

  private final String extension;
  private final String mimeType;

  TipoFichero(String extension, String mimeType) {
```

```
    this.extension = extension;
    this.mimeType = mimeType;
  }

  public String getExtension() {
    return extension;
  }

  public String getMimeType() {
    return mimeType;
  }

}
```

8.1.14 UploadedFileValidator.java

```
package es.uned.tfm.dss.secureapi.validators.jsf;

import java.util.Set;

import org.apache.myfaces.custom.fileupload.UploadedFile;

import es.uned.tfm.dss.secureapi.exceptions.InsecureFileUploadedException;

/**
 * Interfaz que define las operaciones que realizarán los validadores de
 * ficheros
 *
 * @author David García González
 *
 */
public interface UploadedFileValidator {

  public Set<ErrorValidacion> validarFichero(UploadedFile file, TipoFichero
tiposValidos[], long maxSize)
      throws InsecureFileUploadedException;

}
```

8.1.15 UploadedFileValidatorImpl.java

```
package es.uned.tfm.dss.secureapi.validators.jsf.impl;

import java.io.IOException;
import java.io.InputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.text.Normalizer;
import java.text.Normalizer.Form;
import java.util.Date;
import java.util.HashSet;
```

```
import java.util.Set;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import javax.faces.context.FacesContext;
import javax.servlet.http.HttpServletRequest;

import org.acegisecurity.context.SecurityContextHolder;
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.lang.StringUtils;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.myfaces.custom.fileupload.UploadedFile;
import org.apache.tika.exception.TikaException;
import org.apache.tika.metadata.Metadata;
import org.apache.tika.parser.AutoDetectParser;
import org.apache.tika.parser.ParseContext;
import org.apache.tika.parser.Parser;
import org.apache.tika.sax.BodyContentHandler;
import org.xml.sax.ContentHandler;
import org.xml.sax.SAXException;

import es.uned.tfm.dss.secureapi.detector.DocumentDetector;
import es.uned.tfm.dss.secureapi.detector.DocumentDetectorFactory;
import es.uned.tfm.dss.secureapi.exceptions.InsecureFileUploadedException;
import es.uned.tfm.dss.secureapi.model.AuditoriaFicheros;
import es.uned.tfm.dss.secureapi.service.AuditoriaFicherosService;
import es.uned.tfm.dss.secureapi.validators.jsf.ErrorValidacion;
import es.uned.tfm.dss.secureapi.validators.jsf.TipoFichero;
import es.uned.tfm.dss.secureapi.validators.jsf.UploadedFileValidator;

/**
 *
 * Clase que realizará estas validaciones:
 *
 *
 * [X] Tener una configuración de extensiones aceptadas para cada aplicación y
 * una general por si acaso no se hubiera definido para dicha aplicación.
 * Consultar documentación myfaces Tomawawk:
 * https://myfaces.apache.org/tomahawk-
project/tomahawk12/tagdoc/t_inputFileUpload.html
 *
 * [X] Validar la extensión por el nombre del fichero y utilizando también la
 * librería apache Tika.
 *
 * [X] Configurar una expresión regular para nombres de ficheros válidos.
 *
 * [X] Limitar la longitud del fichero
 *
 * [X] Almacenar el resultado de la validación
 *
 * https://software-security.sans.org/blog/2009/12/28/8-basic-rules-to-
implement-secure-file-uploads
 * -
 * https://wiki.sei.cmu.edu/confluence/display/java/IDS56-
J.+Prevent+arbitrary+file+upload
 * - http://tika.apache.org/index.html
```

```
 *
 * @author David García González
 *
 */
public class UploadedFileValidatorImpl implements UploadedFileValidator {

  private static final String SHA_256_ALGORITHM = "SHA-256";
  private static final int MAX_FILENAME_LENGTH = 200;
  private static final String VALID_FILENAME_REGEX = "[a-zA-Z0-9-
_áéíóúÁÉÍÓÚñÑ]{1,200}.[a-zA-Z0-9]{1,10}";

  private AuditoriaFicherosService auditoriaFicherosService;

  private Log log = LogFactory.getLog(UploadedFileValidatorImpl.class);

  /**
   * Realiza la validación de un fichero y devuelve un conjunto con los
errores
   * encontrados
   */
  public Set<ErrorValidacion> validarFichero(UploadedFile file, TipoFichero
tiposValidos[], long maxSize)
      throws InsecureFileUploadedException {

    String fileName = file.getName();
    fileName = getNormalizedName(fileName);

    log.info("Validando el fichero: " + fileName);

    Set<ErrorValidacion> errores = new HashSet<ErrorValidacion>();

    // 1.- Validamos el tamaño del fichero
    if (file.getSize() > maxSize) {
      ErrorValidacion error = new ErrorValidacion("MAX_FILE_SIZE", "El tamaño
del fichero ("
          + getSizeInMb(file.getSize()) + " Mb) excede el maximo permitido: "
+ getSizeInMb(maxSize) + " Mb.");
      errores.add(error);
    }

    // 2.- Validamos el tipo de fichero
    if (!validarMimeType(file, tiposValidos)) {
      ErrorValidacion error = new ErrorValidacion("WRONG_MIME_TIPE",
          "El tipo de fichero no se corresponde con los tipos permitidos: " +
getTiposValidosStr(tiposValidos));
      errores.add(error);

    }

    // 3.- Validar la longitud del nombre del fichero
    if (fileName.length() >= MAX_FILENAME_LENGTH) {
      ErrorValidacion error = new ErrorValidacion("WRONG_FILE_NAME",
          "El nombre del fichero excede los " + MAX_FILENAME_LENGTH + "
caracteres");
      errores.add(error);

    }
```

```
    // 4.- Validar el nombre del fichero
    if (!validarNombreFichero(fileName)) {
      ErrorValidacion error = new ErrorValidacion("WRONG_FILE_NAME",
          "El nombre del fichero (" + fileName + ") no cumple las condiciones
de validez");
      errores.add(error);
    }

    // 5.- Validacion de tipos concretos
    DocumentDetector documentDetector =
DocumentDetectorFactory.getDocumentDetector(this.getContentType(file));

    if (!documentDetector.isSafe(file)) {
      ErrorValidacion error = new ErrorValidacion("PDF_WITH_JAVASCRIPT", "El
fichero PDF (" + fileName
          + ") no cumple las condiciones de seguridad porque contiene código
Javascript incrustado");
      errores.add(error);
    }

    AuditoriaFicheros audit = buildAuditoria(file);

    if (errores.size() == 0) {
      log.info("EL fichero: " + fileName + " ha sido validado
satisfactoriamente");
      audit.setValido(true);
    } else {
      audit.setValido(false);
      StringBuffer mensajeError = new StringBuffer();
      for (ErrorValidacion errorValidacion : errores) {
        mensajeError.append(errorValidacion.getMensaje());
        mensajeError.append("\n");
      }
      audit.setCausaError(mensajeError.toString());
      log.info("El fichero " + fileName + " ha sido validado con los
siguientes errores: " + audit.getCausaError());
    }

    try {
      auditoriaFicherosService.save(audit);
    } catch (Exception e) {
      log.error("No se ha podido almacenar la auditoria: " + audit, e);
    }

    return errores;
  }

  private long getSizeInMb(long maxSize) {
    return maxSize / 1024 / 1024;
  }

  private String getNormalizedName(String fileName) {
    if (!Normalizer.isNormalized(fileName, Form.NFKC)) {
      fileName = Normalizer.normalize(fileName, Form.NFKC);
    }
    return fileName;
  }
```

```
  private AuditoriaFicheros buildAuditoria(UploadedFile file) {
    AuditoriaFicheros audit = new AuditoriaFicheros();

    audit.setFechaSubida(new Date());

    String hash = generateBase64Hash(file, SHA_256_ALGORITHM);
    audit.setHashFichero(hash);
    audit.setAlgoritmoHash(SHA_256_ALGORITHM);

    audit.setMimeType(this.getContentType(file));
    audit.setNombre(getNormalizedName(file.getName()));
    audit.setTamanio(file.getSize());

    audit.setIpSubida(getRemoteAddress());
    getUsuario(audit);

    return audit;
  }

  private void getUsuario(AuditoriaFicheros audit) {
    String user = null;
    if (SecurityContextHolder.getContext().getAuthentication() != null) {
      user = SecurityContextHolder.getContext().getAuthentication().getName();
      audit.setUsuarioSubida(user);
    }
  }

  private String getRemoteAddress() {
    String ipAddress = null;
    HttpServletRequest request = (HttpServletRequest)
FacesContext.getCurrentInstance().getExternalContext()
        .getRequest();
    if (request != null) {
      ipAddress = request.getHeader("X-FORWARDED-FOR");
      if (ipAddress == null) {
        ipAddress = request.getRemoteAddr();
      }
    }
    return ipAddress;
  }

  private String generateBase64Hash(UploadedFile file, String algorithm) {
    try {
      MessageDigest digest = MessageDigest.getInstance(algorithm);
      byte[] hash = digest.digest(file.getBytes());
      byte[] base64 = Base64.encodeBase64(hash);
      return (new String(base64));
    } catch (NoSuchAlgorithmException e) {
      log.error("Error al generar el HASH del fichero", e);
    } catch (IOException e) {
      log.error("Error al abrir el fichero", e);
    }
    return null;
  }

  private String getTiposValidosStr(TipoFichero[] tiposValidos) {
```

```
    return StringUtils.join(tiposValidos, ", ");
  }

  private boolean validarNombreFichero(String fileName) {

    Pattern pattern = Pattern.compile(VALID_FILENAME_REGEX);
    Matcher matcher = pattern.matcher(fileName);
    return matcher.matches();
  }

  private String getContentType(UploadedFile f) {
    try {
      InputStream is = f.getInputStream();

      ContentHandler contenthandler = new BodyContentHandler();
      Metadata metadata = new Metadata();
      metadata.set(Metadata.RESOURCE_NAME_KEY, f.getName());
      Parser parser = new AutoDetectParser();

      parser.parse(is, contenthandler, metadata, new ParseContext());
      return metadata.get(Metadata.CONTENT_TYPE);
    } catch (SAXException e) {
      log.error("Error al intentar detectar el tipo MIME del fichero", e);
    } catch (TikaException e) {
      log.error("Error al intentar detectar el tipo MIME del fichero", e);
    } catch (IOException e) {
      log.error("Error al intentar detectar el tipo MIME del fichero", e);
    }
    return null;

  }

  private boolean validarMimeType(UploadedFile f, TipoFichero tiposValidos[])
{

    String contentType = getContentType(f);
    for (TipoFichero tipoFichero : tiposValidos) {

      /*
       * Si coincide con algun tipo entonces el fichero es valido
       */
      if (contentType != null &&
contentType.equalsIgnoreCase(tipoFichero.getMimeType())) {
        return true;
      }
    }

    return false;
  }

  public AuditoriaFicherosService getAuditoriaFicherosService() {
    return auditoriaFicherosService;
  }

  public void setAuditoriaFicherosService(AuditoriaFicherosService
auditoriaFicherosService) {
```

```
    this.auditoriaFicherosService = auditoriaFicherosService;
  }

}
```

8.1.16 BusquedaAuditoriaFicherosVO.java

```
package es.uned.tfm.dss.secureapi.vo;

import java.util.Date;

/**
 * Clase Value Object con los atributos que especificarán un criterio de
 * búsqueda sobre las trazas de auditoría
 *
 * @author David García González
 *
 */
public class BusquedaAuditoriaFicherosVO {

  private String nombre;

  private String mimeType;

  private String usuarioSubida;

  private Date fechaSubidaDesde;

  private Date fechaSubidaHasta;

  private String ipSubida;

  private Boolean valido;

  public String getNombre() {
    return nombre;
  }

  public void setNombre(String nombre) {
    this.nombre = nombre;
  }

  public String getMimeType() {
    return mimeType;
  }

  public void setMimeType(String mimeType) {
    this.mimeType = mimeType;
  }

  public Date getFechaSubidaDesde() {
    return fechaSubidaDesde;
  }
```

```
  public void setFechaSubidaDesde(Date fechaSubidaDesde) {
    this.fechaSubidaDesde = fechaSubidaDesde;
  }

  public Date getFechaSubidaHasta() {
    return fechaSubidaHasta;
  }

  public void setFechaSubidaHasta(Date fechaSubidaHasta) {
    this.fechaSubidaHasta = fechaSubidaHasta;
  }

  public String getIpSubida() {
    return ipSubida;
  }

  public void setIpSubida(String ipSubida) {
    this.ipSubida = ipSubida;
  }

  public Boolean getValido() {
    return valido;
  }

  public void setValido(Boolean valido) {
    this.valido = valido;
  }

  public String getUsuarioSubida() {
    return usuarioSubida;
  }

  public void setUsuarioSubida(String usuarioSubida) {
    this.usuarioSubida = usuarioSubida;
  }

}
```

8.1.17 orm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<entity-mappings xmlns="http://java.sun.com/xml/ns/persistence/orm"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence/orm orm_2_0.xsd"
  version="2.0">

  <!-- Búsqueda de trazas de auditoria -->
  <named-query name="busquedaAuditoriaFicheros">
    <query>
      <![CDATA[
      SELECT auditoria FROM es.uned.tfm.dss.secureapi.model.AuditoriaFicheros
AS auditoria
```

```
        WHERE
        ((:nombre IS NULL) OR (auditoria.nombre LIKE :nombre))
        AND ((:fechaSubidaDesde IS NULL) OR (auditoria.fechaSubida >=
:fechaSubidaDesde))
        AND ((:fechaSubidaHasta IS NULL) OR (auditoria.fechaSubida <=
:fechaSubidaHasta))
        AND ((:ipSubida IS NULL) OR (auditoria.ipSubida LIKE :ipSubida))
        AND ((:usuarioSubida IS NULL) OR (auditoria.usuarioSubida LIKE
:usuarioSubida))
        AND ((:mimeType IS NULL) OR (auditoria.mimeType LIKE :mimeType))
        AND ((:valido IS NULL) OR (auditoria.valido = :valido))
        ORDER BY
        auditoria.fechaSubida DESC
        ]]>
    </query>
  </named-query>

</entity-mappings>
```

8.1.18 secureapi-persistence.xml

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence

http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
  version="2.0">
  <persistence-unit name="secureapi" transaction-type="RESOURCE_LOCAL">
    <mapping-file>META-INF/orm.xml</mapping-file>
  </persistence-unit>
</persistence>
```

8.1.19 datasource-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
  xmlns:tx="http://www.springframework.org/schema/tx"
  xsi:schemaLocation="
              http://www.springframework.org/schema/beans
              http://www.springframework.org/schema/beans/spring-beans.xsd
              http://www.springframework.org/schema/tx
              http://www.springframework.org/schema/tx/spring-tx.xsd
              http://www.springframework.org/schema/context
              http://www.springframework.org/schema/context/spring-
context.xsd">

  <bean id="secureapiDS"
class="org.springframework.jndi.JndiObjectFactoryBean"
    scope="singleton">
    <property name="jndiName" value="java:comp/env/jdbc/secureapiDS" />
```

```
    <property name="resourceRef" value="true" />
  </bean>

</beans>
```

8.1.20 service-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
  xmlns:tx="http://www.springframework.org/schema/tx"
xmlns:p="http://www.springframework.org/schema/p"
  xsi:schemaLocation="
              http://www.springframework.org/schema/beans
              http://www.springframework.org/schema/beans/spring-beans.xsd
              http://www.springframework.org/schema/tx
              http://www.springframework.org/schema/tx/spring-tx.xsd
              http://www.springframework.org/schema/context
              http://www.springframework.org/schema/context/spring-
context.xsd">

  <context:component-scan base-package="es.uned.tfm.dss.secureapi.*" />

  <!-- ************ JPA configuration ************ -->
  <tx:annotation-driven transaction-manager="transactionManager" />

  <bean id="transactionManager"
class="org.springframework.orm.jpa.JpaTransactionManager">
    <property name="entityManagerFactory" ref="entityManagerFactory" />
  </bean>

  <bean id="entityManagerFactory"
    class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean"
>
    <property name="persistenceXmlLocation"
      value="classpath:META-INF/secureapi-persistence.xml" />

    <property name="persistenceUnitName" value="secureapi" />

    <property name="dataSource" ref="secureapiDS" />
    <property name="packagesToScan" value="es.uned.tfm.dss.secureapi.*" />
    <property name="jpaVendorAdapter">
      <bean
        class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
        <property name="showSql" value="true" />
        <property name="databasePlatform"
value="org.hibernate.dialect.MySQLDialect" />
      </bean>
    </property>
  </bean>
```

```
  <bean id="auditoriaFicherosDao"
    class="es.uned.tfm.dss.secureapi.dao.impl.AuditoriaFicherosDaoImpl" />


  <bean id="auditoriaFicherosService"
    class="es.uned.tfm.dss.secureapi.service.impl.AuditoriaFicherosServiceImpl
">
    <property name="auditoriaFicherosDao" ref="auditoriaFicherosDao" />
  </bean>

  <bean id="uploadedFileValidator"
    class="es.uned.tfm.dss.secureapi.validators.jsf.impl.UploadedFileValidator
Impl">
    <property name="auditoriaFicherosService" ref="auditoriaFicherosService"
/>
  </bean>


</beans>
```

8.1.21 secureapi-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="
              http://www.springframework.org/schema/beans
              http://www.springframework.org/schema/beans/spring-beans.xsd
              http://www.springframework.org/schema/tx
              http://www.springframework.org/schema/tx/spring-tx.xsd
              http://www.springframework.org/schema/context
              http://www.springframework.org/schema/context/spring-
context.xsd">

       <import resource="secureapi/service/service-context.xml"/>
       <import resource="secureapi/service/datasource-context.xml"/>

</beans>
```

8.2 Código fuente del plugin de Spotbugs

8.2.1 pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>es.uned.tfm.dss</groupId>
  <artifactId>uned-spotbugs-plugin</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <properties>
    <maven.compiler.target>1.8</maven.compiler.target>
    <maven.compiler.source>1.8</maven.compiler.source>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <spotBugsVersion>3.1.0</spotBugsVersion>
  </properties>

  <dependencies>
    <dependency>
      <groupId>com.github.spotbugs</groupId>
      <artifactId>spotbugs</artifactId>
      <version>${spotBugsVersion}</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>com.github.spotbugs</groupId>
      <artifactId>test-harness</artifactId>
      <version>${spotBugsVersion}</version>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>es.uned.tfm.dss</groupId>
      <artifactId>secureapi</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>org.apache.myfaces.core</groupId>
      <artifactId>myfaces-api</artifactId>
      <version>1.1.10</version>
      <scope>provided</scope>
    </dependency>

    <dependency>
      <groupId>org.apache.myfaces.core</groupId>
      <artifactId>myfaces-impl</artifactId>
```

```
      <version>1.1.10</version>
      <scope>provided</scope>
    </dependency>

    <dependency>
      <groupId>org.apache.myfaces.tomahawk</groupId>
      <artifactId>tomahawk</artifactId>
      <version>1.1.14</version>
      <scope>provided</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.apache.tika/tika-core -->
    <dependency>
      <groupId>org.apache.tika</groupId>
      <artifactId>tika-core</artifactId>
      <version>1.17</version>
      <scope>provided</scope>
    </dependency>

  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>xml-maven-plugin</artifactId>
        <version>1.0.1</version>
        <executions>
          <execution>
            <id>validate-spotbugs-configuration</id>
            <goals>
              <goal>validate</goal>
            </goals>
            <configuration>
              <validationSets>
                <validationSet>
                  <dir>src/main/resources</dir>
                  <includes>
                    <include>findbugs.xml</include>
                  </includes>
                  <systemId>findbugsplugin.xsd</systemId>
                </validationSet>
                <validationSet>
                  <dir>src/main/resources</dir>
                  <includes>
                    <include>messages.xml</include>
                  </includes>
                  <systemId>messagecollection.xsd</systemId>
                </validationSet>
              </validationSets>
            </configuration>
          </execution>
        </executions>
        <dependencies>
          <dependency>
            <groupId>com.github.spotbugs</groupId>
            <artifactId>spotbugs</artifactId>
```

```
            <version>${spotBugsVersion}</version>
          </dependency>
        </dependencies>
      </plugin>
    </plugins>
  </build>
</project>
```

8.2.2 InsecureFileUploadDetector.java

```
package es.uned.tfm.dss.spotbugs.plugin;

import java.util.Iterator;

import org.apache.bcel.classfile.Field;
import org.apache.bcel.classfile.JavaClass;
import org.apache.bcel.classfile.Method;
import org.apache.bcel.generic.ConstantPoolGen;
import org.apache.bcel.generic.INVOKEINTERFACE;
import org.apache.bcel.generic.Instruction;

import edu.umd.cs.findbugs.BugInstance;
import edu.umd.cs.findbugs.BugReporter;
import edu.umd.cs.findbugs.Detector;
import edu.umd.cs.findbugs.Priorities;
import edu.umd.cs.findbugs.ba.CFG;
import edu.umd.cs.findbugs.ba.CFGBuilderException;
import edu.umd.cs.findbugs.ba.ClassContext;
import edu.umd.cs.findbugs.ba.Location;

/**
 * Detector que comprueba si existen formularios JSF que trabajen con ficheros
y
 * no se aplique validación a los mismos
 *
 * @author David García González
 *
 */
public class InsecureFileUploadDetector implements Detector {

  private static final String UPLOADED_FILE_TYPE =
"org.apache.myfaces.custom.fileupload.UploadedFile";
  private static final String UNRESTRICTED_UPLOAD_FILE_ERROR_TYPE =
"UNRESTRICTED_UPLOAD_FILE";
  private static final String VALIDAR_FICHERO_INTERFACE =
"es.uned.tfm.dss.secureapi.validators.jsf.UploadedFileValidator";
  private static final String VALIDAR_FICHERO_METHOD = "validarFichero";
  private final BugReporter bugReporter;

  public InsecureFileUploadDetector(BugReporter bugReporter) {
    this.bugReporter = bugReporter;
  }
```

```
  @Override
  public void visitClassContext(ClassContext classContext) {

    JavaClass javaClass = classContext.getJavaClass();

    Field[] fields = javaClass.getFields();

    for (Field field : fields) {
      if (UPLOADED_FILE_TYPE.equalsIgnoreCase(field.getType().toString())) {
        Method[] methodList = javaClass.getMethods();

        boolean validated = false;
        for (Method m : methodList) {
          try {
            if (analyzeMethod(m, classContext)) {
              validated = true;
            }
          } catch (CFGBuilderException e) {
          }
        }

        if (!validated) {
          BugInstance bug = new BugInstance(this,
UNRESTRICTED_UPLOAD_FILE_ERROR_TYPE, Priorities.HIGH_PRIORITY)
              .addClass(javaClass)
              .addField(javaClass.getClassName(), field.getName(),
field.getSignature(), field.getAccessFlags());
          bugReporter.reportBug(bug);
        }
      }
    }
  }

  private boolean analyzeMethod(Method m, ClassContext classContext) throws
CFGBuilderException {

    ConstantPoolGen cpg = classContext.getConstantPoolGen();
    CFG cfg = classContext.getCFG(m);

    for (Iterator<Location> i = cfg.locationIterator(); i.hasNext();) {
      Location loc = i.next();

      Instruction inst = loc.getHandle().getInstruction();

      if (inst instanceof INVOKEINTERFACE) {
        INVOKEINTERFACE invoke = (INVOKEINTERFACE) inst;

        if (VALIDAR_FICHERO_METHOD.equals(invoke.getMethodName(cpg))
            && VALIDAR_FICHERO_INTERFACE.equals(invoke.getClassName(cpg))) {
          return true;
        }
      }
    }

    return false;
  }
```

```
  @Override
  public void report() {
    // Nada

  }

}
```

8.2.3 findbugs.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<FindbugsPlugin xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="findbugsplugin.xsd"
pluginid="es.uned.tfm.dss.uned-spotbugs-plugin">

  <Detector class="es.uned.tfm.dss.spotbugs.plugin.InsecureFileUploadDetector"
    reports="UNRESTRICTED_UPLOAD_FILE" speed="fast" />

  <BugPattern abbrev="UUF" type="UNRESTRICTED_UPLOAD_FILE"
    category="SECURITY" />
</FindbugsPlugin>
```

8.2.4 messages.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<MessageCollection xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="messagecollection.xsd">

  <Plugin>
    <ShortDescription>UNED Seucirty Plugin</ShortDescription>
    <Details>Dectecta varias vulnerabilidades de seguridad relacionadas
      con formularios con ficheros</Details>
  </Plugin>

  <Detector
class="es.uned.tfm.dss.spotbugs.plugin.InsecureFileUploadDetector">
    <Details>
      CWE-434: Unrestricted Upload of File with Dangerous Type
    </Details>
  </Detector>

  <BugPattern type="UNRESTRICTED_UPLOAD_FILE">
    <ShortDescription>CWE-434: Unrestricted Upload of File with
      Dangerous Type</ShortDescription>
    <LongDescription>The software allows the attacker to upload or
      transfer files of dangerous types
      that can be automatically processed within the product's
environment</LongDescription>
    <Details>
<![CDATA[
```

```
<p>Arbitrary code execution is possible if an uploaded file is interpreted and
executed as code by the recipient. This is especially true for .asp and .php
extensions uploaded to web servers because these file types are often treated
as automatically executable, even when file system permissions do not specify
execution. For example, in Unix environments, programs typically cannot run
unless the execute bit is set, but PHP programs may be executed by the web
server without directly invoking them on the operating system.</p>
]]>
    </Details>
  </BugPattern>

  <BugCode abbrev="UUF">CWE-434: Unrestricted Upload of File with
    Dangerous Type</BugCode>

</MessageCollection>
```

8.2.5 InsecureFileUploadDetectorTest.java

```
package es.uned.tfm.dss.uned.spotbugs.plugin;

import static edu.umd.cs.findbugs.test.CountMatcher.containsExactly;
import static org.junit.Assert.assertThat;

import java.nio.file.Path;
import java.nio.file.Paths;

import org.junit.Rule;
import org.junit.Test;

import edu.umd.cs.findbugs.BugCollection;
import edu.umd.cs.findbugs.test.SpotBugsRule;
import edu.umd.cs.findbugs.test.matcher.BugInstanceMatcher;
import edu.umd.cs.findbugs.test.matcher.BugInstanceMatcherBuilder;

public class InsecureFileUploadDetectorTest {
  @Rule
  public SpotBugsRule spotbugs = new SpotBugsRule();

  @Test
  public void testGoodCase() {
    Path path = Paths.get("target/test-classes",
"testcode.insecurefileupload".replace('.', '/'),
        "FileUploadFormGoodCase.class");
    BugCollection bugCollection = spotbugs.performAnalysis(path);

    BugInstanceMatcher bugTypeMatcher = new
BugInstanceMatcherBuilder().bugType("UNRESTRICTED_UPLOAD_FILE").build();
    assertThat(bugCollection, containsExactly(0, bugTypeMatcher));
  }

  @Test
  public void testBadCase() {
```

```
    Path path = Paths.get("target/test-classes",
"testcode.insecurefileupload".replace('.', '/'),
        "FileUploadFormBadCase.class");
    BugCollection bugCollection = spotbugs.performAnalysis(path);

    BugInstanceMatcher bugTypeMatcher = new
BugInstanceMatcherBuilder().bugType("UNRESTRICTED_UPLOAD_FILE").build();
    assertThat(bugCollection, containsExactly(1, bugTypeMatcher));
  }
}
```

8.2.6 FileUploadFormBadCase.java

```
package testcode.insecurefileupload;

import java.io.IOException;

import javax.faces.context.FacesContext;

import org.apache.myfaces.custom.fileupload.UploadedFile;

/**
 * Clase para probar que el validador encuentra un error en este fichero
 *
 * @author David García González
 *
 */
public class FileUploadFormBadCase {
  private UploadedFile fichero;
  private String nombreFichero = "";

  public UploadedFile getUpFile() {
    return fichero;
  }

  public void setUpFile(UploadedFile upFile) {
    fichero = upFile;
  }

  public String getName() {
    return nombreFichero;
  }

  public void setName(String name) {
    nombreFichero = name;
  }

  @SuppressWarnings("unchecked")
  public String upload() throws IOException {
    FacesContext facesContext = FacesContext.getCurrentInstance();
    facesContext.getExternalContext().getApplicationMap().put("fileupload_byte
s", fichero.getBytes());
```

```
    facesContext.getExternalContext().getApplicationMap().put("fileupload_type
", fichero.getContentType());
    facesContext.getExternalContext().getApplicationMap().put("fileupload_name
", fichero.getName());
    return "ok";
  }

  public boolean isUploaded() {
    FacesContext facesContext = FacesContext.getCurrentInstance();
    return
facesContext.getExternalContext().getApplicationMap().get("fileupload_bytes")
!= null;
  }

}
```

8.2.7 FileUploadFormGoodCase.java

```
package testcode.insecurefileupload;

import java.io.IOException;

import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.component.UIInput;
import javax.faces.context.FacesContext;

import org.apache.myfaces.custom.fileupload.UploadedFile;

import es.uned.tfm.dss.secureapi.exceptions.InsecureFileUploadedException;
import es.uned.tfm.dss.secureapi.validators.jsf.TipoFichero;
import es.uned.tfm.dss.secureapi.validators.jsf.UploadedFileValidator;
import
es.uned.tfm.dss.secureapi.validators.jsf.impl.UploadedFileValidatorImpl;

/**
 * Clase para probar que el validador no encuentra ningún error en este
fichero
 *
 * @author David García González
 *
 */
public class FileUploadFormGoodCase {
  private UploadedFile fichero;
  private String nombreFichero = "";

  UploadedFileValidator uploadedFileValidator = new
UploadedFileValidatorImpl();

  public UploadedFile getUpFile() {
    return fichero;
  }
```

```
  public void setUpFile(UploadedFile upFile) {
    fichero = upFile;
  }

  public String getName() {
    return nombreFichero;
  }

  public void setName(String name) {
    nombreFichero = name;
  }

  @SuppressWarnings("unchecked")
  public String upload() throws IOException {

    FacesContext facesContext = FacesContext.getCurrentInstance();
    facesContext.getExternalContext().getApplicationMap().put("fileupload_byte
s", fichero.getBytes());
    facesContext.getExternalContext().getApplicationMap().put("fileupload_type
", fichero.getContentType());
    facesContext.getExternalContext().getApplicationMap().put("fileupload_name
", fichero.getName());
    return "ok";
  }

  public boolean isUploaded() {
    FacesContext facesContext = FacesContext.getCurrentInstance();
    return
facesContext.getExternalContext().getApplicationMap().get("fileupload_bytes")
!= null;
  }

  public void validateFile(FacesContext context, UIComponent toValidate,
Object value) {

    UploadedFile f = (UploadedFile) value;
    TipoFichero[] tipos = { TipoFichero.JPG };
    try {
      uploadedFileValidator.validarFichero(f, tipos, 1024);
    } catch (InsecureFileUploadedException e) {
      ((UIInput) toValidate).setValid(false);

      context.addMessage(toValidate.getClientId(context), new
FacesMessage(e.getMessage()));
    }

  }



}
```

8.3 Código fuente del prototipo con un formulario seguro y un formulario inseguro

8.3.1 pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>es.uned.tfm.dss</groupId>
  <artifactId>uned-jsf-example</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>jsf-example Maven Webapp</name>
  <url>http://maven.apache.org</url>

  <properties>
    <myfaces.version>1.1.10</myfaces.version>
    <tomahawk.version>1.1.14</tomahawk.version>
    <tika.version>1.17</tika.version>
    <acegi-security.version>1.0.4</acegi-security.version>
    <spring.version>3.2.18.RELEASE</spring.version>
  </properties>

  <dependencies>

    <dependency>
      <groupId>es.uned.tfm.dss</groupId>
      <artifactId>secureapi</artifactId>
      <version>0.0.1-SNAPSHOT</version>
    </dependency>

    <dependency>
      <groupId>org.apache.myfaces.tomahawk</groupId>
      <artifactId>tomahawk</artifactId>
      <version>${tomahawk.version}</version>
    </dependency>

    <dependency>
      <groupId>org.apache.myfaces.core</groupId>
      <artifactId>myfaces-api</artifactId>
      <version>${myfaces.version}</version>
    </dependency>
    <dependency>
      <groupId>org.apache.myfaces.core</groupId>
      <artifactId>myfaces-impl</artifactId>
      <version>${myfaces.version}</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-web -->
    <dependency>
```

```
      <groupId>org.springframework</groupId>
      <artifactId>spring-web</artifactId>
      <version>${spring.version}</version>
    </dependency>

    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>jstl</artifactId>
      <version>1.2</version>
    </dependency>
    <dependency>
      <groupId>taglibs</groupId>
      <artifactId>standard</artifactId>
      <version>1.1.2</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.apache.tika/tika-core -->
    <dependency>
      <groupId>org.apache.tika</groupId>
      <artifactId>tika-core</artifactId>
      <version>${tika.version}</version>
    </dependency>

    <dependency>
      <groupId>org.acegisecurity</groupId>
      <artifactId>acegi-security</artifactId>
      <version>${acegi-security.version}</version>
    </dependency>


    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>2.3</version>
      <scope>provided</scope>
    </dependency>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <finalName>uned-jsf-example</finalName>

    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>1.6</source>
          <target>1.6</target>
          <encoding>UTF-8</encoding>
        </configuration>
```

```
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>3.2.0</version>

      </plugin>
    </plugins>
  </build>

</project>
```

8.3.2 AppUser.java

```
package es.uned.tfm.jsf.authentication;

import java.io.Serializable;
import java.util.Set;

/**
 *
 * Clase que representa un usuario de la aplicación
 *
 * @author David García González
 *
 */
public class AppUser implements Serializable {

  private static final long serialVersionUID = 4153742545287876955L;

  private String firstName;

  private String lastName;

  private String login;

  private String password;

  private Set<String> roles;

  public AppUser(String firstName, String lastName, String login, String
password, Set<String> roles) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.login = login;
    this.password = password;
    this.roles = roles;
  }

  public String getFirstName() {
    return firstName;
  }
```

```
  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }

  public String getLastName() {
    return lastName;
  }

  public void setLastName(String lastName) {
    this.lastName = lastName;
  }

  public String getLogin() {
    return login;
  }

  public void setLogin(String login) {
    this.login = login;
  }

  public String getPassword() {
    return password;
  }

  public void setPassword(String password) {
    this.password = password;
  }

  public Set<String> getRoles() {
    return roles;
  }

  public void setRoles(Set<String> roles) {
    this.roles = roles;
  }

}
```

8.3.3 UserDao.java

```
package es.uned.tfm.jsf.authentication;

/**
 * Interfaz que define las operaciónes para la recuperación de usuarios de un
 * sistema
 *
 * @author David García González
 *
 */
public interface UserDao {

  /**
   * Devuelve un usuario a partir del nombre de usuario
```

```
    *
    * @param userName
    * @return El usuario, si existe en el sistema.
    */
   public AppUser findUser(String userName);

}
```

8.3.4 UserDaoImpl.java

```
package es.uned.tfm.jsf.authentication;

import java.util.HashSet;
import java.util.Set;

/**
 * Clase de ejemplo que simula la recuperación de usuarios de un sistema
 *
 * @author David García González
 *
 */
public class UserDaoImpl implements UserDao {

   /**
    * {@inheritDoc}
    */
   public AppUser findUser(String userName) {
     AppUser appUser = null;
     Set<String> roles = new HashSet<String>();
     if (userName.equals("admin")) {
       roles.add("ROLE_ALLACCESS");
       appUser = new AppUser("Administrador", "", "admin", "admin", roles);
     } else if (userName.equals("david")) {
       roles.add("ROLE_ALLACCESS");
       appUser = new AppUser("David", "García", "david", "david", roles);
     }
     return appUser;
   }

}
```

8.3.5 UserDetailsServiceImpl.java

```
package es.uned.tfm.jsf.authentication;

import org.acegisecurity.GrantedAuthority;
import org.acegisecurity.GrantedAuthorityImpl;
import org.acegisecurity.userdetails.UserDetails;
import org.acegisecurity.userdetails.UserDetailsService;
import org.acegisecurity.userdetails.UsernameNotFoundException;
import org.springframework.dao.DataAccessException;
```

```
/**
 * Implementación de la interfaz UserDetailsService que requiere el filtro de
 * seguridad Acegi para obtener los usuarios autorizados
 *
 * @author David García González
 *
 */
public class UserDetailsServiceImpl implements UserDetailsService {
  private UserDao userDao;

  public UserDetailsServiceImpl(UserDao userDao) {
    this.userDao = userDao;
  }

  /**
   * {@inheritDoc}
   */
  public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException, DataAccessException {
    AppUser user = userDao.findUser(username);
    if (user == null)
      throw new UsernameNotFoundException("No se ha encontrado el usuario: " +
username);
    else {
      return makeAcegiUser(user);
    }
  }

  private org.acegisecurity.userdetails.User makeAcegiUser(AppUser user) {
    return new org.acegisecurity.userdetails.User(user.getLogin(),
user.getPassword(), true, true, true, true,
        makeGrantedAuthorities(user));
  }

  private GrantedAuthority[] makeGrantedAuthorities(AppUser user) {
    GrantedAuthority[] result = new GrantedAuthority[user.getRoles().size()];
    int i = 0;
    for (String role : user.getRoles()) {
      result[i++] = new GrantedAuthorityImpl(role);
    }
    return result;
  }
}
```

8.3.6 BusquedaAuditoriaFicherosForm.java

```
package es.uned.tfm.jsf.fileupload;

import java.io.Serializable;
import java.util.List;

import es.uned.tfm.dss.secureapi.model.AuditoriaFicheros;
```

```
import es.uned.tfm.dss.secureapi.service.AuditoriaFicherosService;
import es.uned.tfm.dss.secureapi.vo.BusquedaAuditoriaFicherosVO;

/**
 * Formulario para realizar la búsqueda de trazas de auditoría
 *
 * @author David García González
 *
 */
public class BusquedaAuditoriaFicherosForm implements Serializable {

  private static final long serialVersionUID = 5251972962418461791L;

  AuditoriaFicherosService auditoriaFicherosService;

  private BusquedaAuditoriaFicherosVO busqueda = new
BusquedaAuditoriaFicherosVO();

  private List<AuditoriaFicheros> listaAuditoria;

  public String buscar() {

    listaAuditoria = auditoriaFicherosService.find(busqueda);

    return "ok";
  }

  public AuditoriaFicherosService getAuditoriaFicherosService() {
    return auditoriaFicherosService;
  }

  public void setAuditoriaFicherosService(AuditoriaFicherosService
auditoriaFicherosService) {
    this.auditoriaFicherosService = auditoriaFicherosService;
  }

  public BusquedaAuditoriaFicherosVO getBusqueda() {
    return busqueda;
  }

  public void setBusqueda(BusquedaAuditoriaFicherosVO busqueda) {
    this.busqueda = busqueda;
  }

  public List<AuditoriaFicheros> getListaAuditoria() {
    return listaAuditoria;
  }

  public void setListaAuditoria(List<AuditoriaFicheros> listaAuditoria) {
    this.listaAuditoria = listaAuditoria;
  }

}
```

8.3.7 FileUploadForm.java

```
package es.uned.tfm.jsf.fileupload;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import javax.faces.context.FacesContext;
import javax.servlet.http.HttpSession;

import org.apache.commons.io.IOUtils;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.myfaces.custom.fileupload.UploadedFile;

/**
 * Ejemplo de formulario inseguro. Contiene múltiples vulnerabilidades
 * relacionadas con la subida de ficheros.
 *
 * @author David García González
 *
 */
public class FileUploadForm {

  private static final Log log = LogFactory.getLog(FileUploadForm.class);

  private UploadedFile _upFile;

  public UploadedFile getUpFile() {
    return _upFile;
  }

  public void setUpFile(UploadedFile upFile) {
    _upFile = upFile;
  }

  /**
   * Realiza la subida del fichero al servidor. Almacena el fichero en un
   * directorio y lo guarda en sesión.
   *
   * @return
   * @throws IOException
   */
  public String upload() throws IOException {

    FacesContext facesContext = FacesContext.getCurrentInstance();
    HttpSession session = (HttpSession)
facesContext.getExternalContext().getSession(true);
    session.setAttribute("fileupload_bytes", _upFile.getBytes());
    session.setAttribute("fileupload_type", _upFile.getContentType());
    session.setAttribute("fileupload_name", _upFile.getName());

    File file = new File("/tmp/ficheros_uned/" + _upFile.getName());
```

```
    InputStream is = _upFile.getInputStream();

    OutputStream os = new FileOutputStream(file);

    IOUtils.copy(is, os);

    IOUtils.closeQuietly(is);
    IOUtils.closeQuietly(os);

    log.info("Subido el fichero " + _upFile.getName() + " satisfactoriamente
al directorio: " + file.getAbsolutePath());
    return "ok";
  }

  /**
   * Nos indica si se ha subido el fichero o no
   *
   * @return true: si se ha subido false: no se ha subido
   */
  public boolean isUploaded() {
    FacesContext facesContext = FacesContext.getCurrentInstance();
    HttpSession session = (HttpSession)
facesContext.getExternalContext().getSession(true);
    return (session.getAttribute("fileupload_bytes") != null);
  }

  /**
   * Devuelve el nombre del fichero que se ha subido. Si todavía no se ha
subido,
   * devuelve nulo.
   *
   * @return El nombre del fichero que se ha subido.
   */
  public String getFileName() {
    FacesContext facesContext = FacesContext.getCurrentInstance();
    HttpSession session = (HttpSession)
facesContext.getExternalContext().getSession(true);
    if (session.getAttribute("fileupload_name") != null) {
      return session.getAttribute("fileupload_name").toString();
    }
    return null;
  }

}
```

8.3.8 LoginForm.java

```
package es.uned.tfm.jsf.fileupload;

import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
```

```
import org.acegisecurity.ui.AbstractProcessingFilter;

/**
 * Formulario de autenticación para el formulario seguro
 *
 * @author David García González
 *
 */
public class LoginForm {

  public LoginForm() {
    Exception ex = (Exception)
FacesContext.getCurrentInstance().getExternalContext().getSessionMap()
        .get(AbstractProcessingFilter.ACEGI_SECURITY_LAST_EXCEPTION_KEY);

    if (ex != null)
      FacesContext.getCurrentInstance().addMessage(null,
          new FacesMessage(FacesMessage.SEVERITY_ERROR, ex.getMessage(),
ex.getMessage()));

  }

  private String userId;

  private String password;

  public String getUserId() {
    return userId;
  }

  public void setUserId(String userId) {
    this.userId = userId;
  }

  public String getPassword() {
    return password;
  }

  public void setPassword(String password) {
    this.password = password;
  }

}
```

8.3.9 SecureFileUploadForm.java

```
package es.uned.tfm.jsf.fileupload;

import java.io.File;
```

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Set;

import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.component.UIInput;
import javax.faces.context.FacesContext;
import javax.servlet.http.HttpSession;

import org.apache.commons.io.IOUtils;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.myfaces.custom.fileupload.UploadedFile;

import es.uned.tfm.dss.secureapi.exceptions.InsecureFileUploadedException;
import es.uned.tfm.dss.secureapi.validators.jsf.ErrorValidacion;
import es.uned.tfm.dss.secureapi.validators.jsf.TipoFichero;
import es.uned.tfm.dss.secureapi.validators.jsf.UploadedFileValidator;

/**
 * Ejemplo de formulario seguro. Se aplica una validación sobre el fichero
 *
 * @author David García González
 */
public class SecureFileUploadForm {

  private static final Log log =
LogFactory.getLog(SecureFileUploadForm.class);

  private static final int MAX_SIZE = 10 * 1024 * 1024; // 10MB
  private UploadedFile _upFile;
  private String _name = "";

  UploadedFileValidator uploadedFileValidator;

  public UploadedFile getUpFile() {
    return _upFile;
  }

  public void setUpFile(UploadedFile upFile) {
    _upFile = upFile;
  }

  public String getName() {
    return _name;
  }

  public void setName(String name) {
    _name = name;
  }

  /**
   * Realiza la subida del fichero al servidor. Almacena el fichero en un
   * directorio y lo guarda en sesión.
```

```
    *
    * @return
    * @throws IOException
    */
  public String upload() throws IOException {

    FacesContext facesContext = FacesContext.getCurrentInstance();
    HttpSession session = (HttpSession)
facesContext.getExternalContext().getSession(true);

    session.setAttribute("fileupload_bytes", _upFile.getBytes());
    session.setAttribute("fileupload_type", _upFile.getContentType());
    session.setAttribute("fileupload_name", _upFile.getName());

    File file = new File("/tmp/ficheros_uned_seguros/" + _upFile.getName());

    InputStream is = _upFile.getInputStream();

    OutputStream os = new FileOutputStream(file);

    IOUtils.copy(is, os);

    IOUtils.closeQuietly(is);
    IOUtils.closeQuietly(os);

    log.info("Subido el fichero " + _upFile.getName() + " satisfactoriamente
al directorio: " + file.getAbsolutePath());
    return "ok";
  }

  /**
   * Nos indica si se ha subido el fichero o no
   *
   * @return true: si se ha subido false: no se ha subido
   */
  public boolean isUploaded() {
    FacesContext facesContext = FacesContext.getCurrentInstance();
    HttpSession session = (HttpSession)
facesContext.getExternalContext().getSession(true);

    return (session.getAttribute("fileupload_bytes") != null);
  }

  /**
   * Devuelve el nombre del fichero que se ha subido. Si todavía no se ha
subido,
   * devuelve nulo.
   *
   * @return El nombre del fichero que se ha subido.
   */
  public String getFileName() {
    FacesContext facesContext = FacesContext.getCurrentInstance();
    HttpSession session = (HttpSession)
facesContext.getExternalContext().getSession(true);

    if (session.getAttribute("fileupload_name") != null) {
      return session.getAttribute("fileupload_name").toString();
```

```
    }
    return null;
  }

  /**
   * Validación del fichero
   *
   * @param context
   * @param toValidate
   * @param value
   */
  public void validateFile(FacesContext context, UIComponent toValidate,
Object value) {

    UploadedFile f = (UploadedFile) value;
    TipoFichero[] tipos = { TipoFichero.JPG, TipoFichero.PDF };
    try {
      Set<ErrorValidacion> errores = uploadedFileValidator.validarFichero(f,
tipos, MAX_SIZE);
      for (ErrorValidacion errorValidacion : errores) {
        ((UIInput) toValidate).setValid(false);
        context.addMessage(toValidate.getClientId(context), new
FacesMessage(errorValidacion.getMensaje()));
      }

    } catch (InsecureFileUploadedException e) {

      log.warn("Se ha intentado adjuntar un fichero no valido", e);
      ((UIInput) toValidate).setValid(false);

      context.addMessage(toValidate.getClientId(context), new
FacesMessage(e.getMessage()));
    }
  }

  public UploadedFileValidator getUploadedFileValidator() {
    return uploadedFileValidator;
  }

  public void setUploadedFileValidator(UploadedFileValidator
uploadedFileValidator) {
    this.uploadedFileValidator = uploadedFileValidator;
  }

}
```

8.3.10 security-web-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Configuración de la seguridad web con Acegi, incluye: - Autentificación
```

```
  por formulario - Acceso anónimo Los filtros web son especificados en el
bean
  filterChainProxy. -->

<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
  xmlns:tx="http://www.springframework.org/schema/tx"
xmlns:jee="http://www.springframework.org/schema/jee"
  xmlns:lang="http://www.springframework.org/schema/lang"
xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:util="http://www.springframework.org/schema/util"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
       http://www.springframework.org/schema/jee
       http://www.springframework.org/schema/jee/spring-jee-2.0.xsd
       http://www.springframework.org/schema/lang
       http://www.springframework.org/schema/lang/spring-lang-2.0.xsd
       http://www.springframework.org/schema/aop
       http://www.springframework.org/schema/aop/spring-aop-2.0.xsd
       http://www.springframework.org/schema/util
       http://www.springframework.org/schema/util/spring-util-2.0.xsd">

  <!-- Bean: filterChainProxy Usage: Requerido por la plataforma Description:
    Objeto que se encarga de encadenar los filtros de seguridad -->

  <bean id="filterChainProxy" class="org.acegisecurity.util.FilterChainProxy">
    <property name="filterInvocationDefinitionSource">
      <value>
        CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
        PATTERN_TYPE_APACHE_ANT
        /error/*=#NONE#
        /back*=#NONE#
        /**=httpSessionContextIntegrationFilter,logoutFilter,authenticationPro
cessingFilter,securityContextHolderAwareRequestFilter,anonymousProcessingFilte
r,exceptionTranslationFilter,filterInvocationInterceptor
      </value>
    </property>
  </bean>

  <bean id="httpSessionContextIntegrationFilter"
    class="org.acegisecurity.context.HttpSessionContextIntegrationFilter" />

  <bean id="securityContextHolderAwareRequestFilter"
    class="org.acegisecurity.wrapper.SecurityContextHolderAwareRequestFilter"
/>


  <bean id="authenticationProcessingFilter"
    class="org.acegisecurity.ui.webapp.AuthenticationProcessingFilter">
    <property name="filterProcessesUrl">
      <value>/j_acegi_security_check.jsp</value>
    </property>
    <property name="authenticationFailureUrl">
      <value>/login.jsf</value>
    </property>
    <property name="defaultTargetUrl">
```

```
      <value>/index.jsf</value>
    </property>
    <property name="authenticationManager">
      <ref bean="authenticationManager" />
    </property>
  </bean>

  <bean id="filterInvocationInterceptor"
    class="org.acegisecurity.intercept.web.FilterSecurityInterceptor">
    <property name="authenticationManager" ref="authenticationManager" />
    <property name="accessDecisionManager" ref="accessDecisionManager" />
    <property name="objectDefinitionSource">
      <value>
        CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
        PATTERN_TYPE_APACHE_ANT
        /busquedaauditoriaficheros.jsf=ROLE_ALLACCESS,ROLE_URLACCESS
        /securefileupload.jsf=ROLE_ALLACCESS,ROLE_URLACCESS
        /index.jsf=IS_AUTHENTICATED_ANONYMOUSLY
        /login.jsf=IS_AUTHENTICATED_ANONYMOUSLY
        /fileupload.jsf=IS_AUTHENTICATED_ANONYMOUSLY
        /**=IS_AUTHENTICATED_ANONYMOUSLY
      </value>
    </property>
  </bean>

  <bean id="accessDecisionManager"
class="org.acegisecurity.vote.AffirmativeBased">
    <property name="allowIfAllAbstainDecisions" value="false" />
    <property name="decisionVoters">
      <list>
        <bean class="org.acegisecurity.vote.RoleVoter" />
        <bean class="org.acegisecurity.vote.AuthenticatedVoter" />
      </list>
    </property>
  </bean>

  <bean id="authenticationManager"
class="org.acegisecurity.providers.ProviderManager">
    <property name="providers">
      <list>
        <ref local="daoAuthenticationProvider" />
      </list>
    </property>
  </bean>

  <bean id="daoAuthenticationProvider"
    class="org.acegisecurity.providers.dao.DaoAuthenticationProvider">
    <property name="userDetailsService" ref="userDetailsService" />
  </bean>

  <bean id="userDetailsService"
class="es.uned.tfm.jsf.authentication.UserDetailsServiceImpl">
    <constructor-arg ref="userRepository" />
  </bean>

  <bean id="userRepository"
class="es.uned.tfm.jsf.authentication.UserDaoImpl">
```

```
  </bean>

  <bean id="anonymousProcessingFilter"
    class="org.acegisecurity.providers.anonymous.AnonymousProcessingFilter">
    <property name="key" value="changeThis" />
    <property name="userAttribute" value="anonymousUser,ROLE_ANONYMOUS" />
  </bean>

  <bean id="exceptionTranslationFilter"
class="org.acegisecurity.ui.ExceptionTranslationFilter">
    <property name="authenticationEntryPoint">
      <bean
        class="org.acegisecurity.ui.webapp.AuthenticationProcessingFilterEntry
Point">
        <property name="loginFormUrl" value="/login.jsf" />
        <property name="forceHttps" value="false" />
      </bean>
    </property>
    <property name="accessDeniedHandler">
      <bean class="org.acegisecurity.ui.AccessDeniedHandlerImpl">
        <property name="errorPage" value="/accessDenied.jsf" />
      </bean>
    </property>
  </bean>

  <bean id="logoutFilter" class="org.acegisecurity.ui.logout.LogoutFilter">
    <constructor-arg value="/index.jsp" />
    <constructor-arg>
      <list>
        <bean class="org.acegisecurity.ui.logout.SecurityContextLogoutHandler"
/>
      </list>
    </constructor-arg>
    <property name="filterProcessesUrl">
      <value>/j_acegi_logout.jsp</value>
    </property>
  </bean>
</beans>
```

8.3.11 web-context.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
  xmlns:tx="http://www.springframework.org/schema/tx"
xmlns:jee="http://www.springframework.org/schema/jee"
  xmlns:lang="http://www.springframework.org/schema/lang"
xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:util="http://www.springframework.org/schema/util"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
       http://www.springframework.org/schema/jee
       http://www.springframework.org/schema/jee/spring-jee-2.0.xsd
```

```
       http://www.springframework.org/schema/lang
       http://www.springframework.org/schema/lang/spring-lang-2.0.xsd
       http://www.springframework.org/schema/aop
       http://www.springframework.org/schema/aop/spring-aop-2.0.xsd
       http://www.springframework.org/schema/util
       http://www.springframework.org/schema/util/spring-util-2.0.xsd">

  <import
    resource="classpath:/uned-jsf-example/security/security-web-context.xml"
/>

  <import resource="classpath*:/secureapi-context.xml" />

</beans>
```

8.3.12 commons-logging.properties

```
org.apache.commons.logging.Log=org.apache.commons.logging.impl.Log4JLogger
```

8.3.13 log4j.properties

```
#for debugging log4j itself
log4j.debug=false

#Logger-Priorities:
#DEBUG lowest, prints all messages
#INFO  prints all messages with FATAL, ERROR, WARN or INFO priority
#WARN  prints all messages with FATAL, ERROR or WARN priority
#ERROR prints all messages with FATAL or ERROR priority
#FATAL highest, prints only FATAL messages

# root logger
log4j.rootLogger=INFO, A1
log4j.appender.A1=org.apache.log4j.ConsoleAppender
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=%d [%t] %-5p %c - %m%n

# myfaces logger
log4j.logger.org.apache.myfaces=ERROR

# variable resolver logger
log4j.logger.org.apache.myfaces.el.VariableResolverImpl=DEBUG
log4j.logger.es.uned.tfm.dss.secureapi=DEBUG
```

8.3.14 faces-config.xml

```
<?xml version="1.0"?>
```

```
<!DOCTYPE faces-config PUBLIC
  "-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.1//EN"
  "http://java.sun.com/dtd/web-facesconfig_1_1.dtd">
<faces-config>

  <application>
    <variable-resolver>
      org.springframework.web.jsf.DelegatingVariableResolver
    </variable-resolver>

    <locale-config>
      <default-locale>es</default-locale>
      <supported-locale>en</supported-locale>
      <supported-locale>es</supported-locale>
    </locale-config>
  </application>

  <!-- Managed Beans -->

  <managed-bean>
    <managed-bean-name>fileUploadForm</managed-bean-name>
    <managed-bean-class>es.uned.tfm.jsf.fileupload.FileUploadForm</managed-
bean-class>
    <managed-bean-scope>request</managed-bean-scope>
  </managed-bean>

  <managed-bean>
    <managed-bean-name>secureFileUploadForm</managed-bean-name>
    <managed-bean-
class>es.uned.tfm.jsf.fileupload.SecureFileUploadForm</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>

    <!-- inyecciones -->
    <managed-property>
      <property-name>uploadedFileValidator</property-name>
      <value>#{uploadedFileValidator}</value>
    </managed-property>
  </managed-bean>

  <managed-bean>
    <managed-bean-name>loginBacking</managed-bean-name>
    <managed-bean-class>es.uned.tfm.jsf.fileupload.LoginForm</managed-bean-
class>
    <managed-bean-scope>request</managed-bean-scope>
  </managed-bean>

  <managed-bean>
    <managed-bean-name>busquedaAutidoriaFicherosForm</managed-bean-name>
    <managed-bean-
class>es.uned.tfm.jsf.fileupload.BusquedaAuditoriaFicherosForm</managed-bean-
class>
    <managed-bean-scope>session</managed-bean-scope>
    <managed-property>
      <property-name>auditoriaFicherosService</property-name>
      <value>#{auditoriaFicherosService}</value>
    </managed-property>
  </managed-bean>
```

```
  <navigation-rule>
    <from-view-id>*</from-view-id>
    <navigation-case>
      <from-outcome>go_fileupload</from-outcome>
      <to-view-id>/fileupload.jsp</to-view-id>
    </navigation-case>
    <navigation-case>
      <from-outcome>go_secure_fileupload</from-outcome>
      <to-view-id>/securefileupload.jsp</to-view-id>
    </navigation-case>
  </navigation-rule>

  <navigation-rule>
    <from-view-id>/login.jsp</from-view-id>
    <navigation-case>
      <from-outcome>login</from-outcome>
      <to-view-id>/j_acegi_security_check.jsp</to-view-id>
    </navigation-case>
  </navigation-rule>

  <navigation-rule>
    <from-view-id>*</from-view-id>
    <navigation-case>
      <from-outcome>logout</from-outcome>
      <to-view-id>/j_acegi_logout.jsp</to-view-id>
    </navigation-case>
  </navigation-rule>

  <navigation-rule>
    <from-view-id>*</from-view-id>
    <navigation-case>
      <from-outcome>busqueda</from-outcome>
      <to-view-id>/busquedaAuditoriaFicheros.jsp</to-view-id>
    </navigation-case>
  </navigation-rule>

</faces-config>
```

8.3.15 web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4">
  <description>uned-jsf-example</description>

  <!--Creación del contexto de JSF -->
       <context-param>
              <description>Spring config file locations</description>
              <param-name>contextConfigLocation</param-name>
```

```
            <param-value>
                  <!-- FRAMEWORK y SEGURIDAD -->
                  classpath:/uned-jsf-example/web-context.xml
            </param-value>
      </context-param>
  <context-param>
    <description>Comma separated list of URIs of (additional) faces config
files.
            (e.g. /WEB-INF/my-config.xml)
            See JSF 1.0 PRD2, 10.3.2
            Attention: You do not need to put /WEB-INF/faces-config.xml in
here.
    </description>
    <param-name>javax.faces.CONFIG_FILES</param-name>
    <param-value></param-value>
  </context-param>

  <context-param>
    <description>State saving method: "client" or "server" (= default)
            See JSF Specification 2.5.3</description>
    <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
    <param-value>client</param-value>
  </context-param>

  <context-param>
    <description>Only applicable if state saving method is "server" (=
default).
            Defines the amount (default = 20) of the latest views are stored
in session.</description>
    <param-name>org.apache.myfaces.NUMBER_OF_VIEWS_IN_SESSION</param-name>
    <param-value>20</param-value>
  </context-param>

  <context-param>
    <description>Only applicable if state saving method is "server" (=
default).
            If true (default) the state will be serialized to a byte stream
before it
            is written to the session.
            If false the state will not be serialized to a byte
stream.</description>
    <param-name>org.apache.myfaces.SERIALIZE_STATE_IN_SESSION</param-name>
    <param-value>true</param-value>
  </context-param>

  <context-param>
    <description>Only applicable if state saving method is "server" (=
default) and if
            org.apache.myfaces.SERIALIZE_STATE_IN_SESSION is true (= default)
            If true (default) the serialized state will be compressed before
it
            is written to the session. If false the state will not be
compressed.</description>
    <param-name>org.apache.myfaces.COMPRESS_STATE_IN_SESSION</param-name>
    <param-value>true</param-value>
  </context-param>
```

```
  <context-param>
    <description>This parameter tells MyFaces if javascript code should be
allowed in the
            rendered HTML output.
            If javascript is allowed, command_link anchors will have
javascript code
            that submits the corresponding form.
            If javascript is not allowed, the state saving info and nested
parameters
            will be added as url parameters.
            Default: "true"</description>
    <param-name>org.apache.myfaces.ALLOW_JAVASCRIPT</param-name>
    <param-value>true</param-value>
  </context-param>

  <context-param>
    <param-name>org.apache.myfaces.DETECT_JAVASCRIPT</param-name>
    <param-value>false</param-value>
  </context-param>
  <context-param>
    <description>If true, rendered HTML code will be formatted, so that it is
"human readable".
            i.e. additional line separators and whitespace will be written,
that do not
            influence the HTML code.
            Default: "true"</description>
    <param-name>org.apache.myfaces.PRETTY_HTML</param-name>
    <param-value>true</param-value>
  </context-param>
  <context-param>
    <description>If true, a javascript function will be rendered that is able
to restore the
            former vertical scroll on every request. Convenient feature if you
have pages
            with long lists and you do not want the browser page to always
jump to the top
            if you trigger a link or button action that stays on the same
page.
            Default: "false"</description>
    <param-name>org.apache.myfaces.AUTO_SCROLL</param-name>
    <param-value>true</param-value>
  </context-param>

    <context-param>
             <description>
                   Validate managed beans, navigation rules and ensure that
forms are not nested.
             </description>
             <param-name>org.apache.myfaces.VALIDATE</param-name>
             <param-value>false</param-value>
    </context-param>

  <context-param>
    <description>A class implementing the
                org.apache.myfaces.shared.renderkit.html.util.AddResource
                interface. It is responsible to
```

```
                    place scripts and css on the right position in your HTML
document.
            Default:
"org.apache.myfaces.shared.renderkit.html.util.DefaultAddResource"
            Follow the description on the MyFaces-Wiki-Performance page to
enable
            StreamingAddResource instead of DefaultAddResource if you want to
            gain performance.
    </description>
    <param-name>org.apache.myfaces.ADD_RESOURCE_CLASS</param-name>
    <param-
value>org.apache.myfaces.renderkit.html.util.DefaultAddResource</param-value>
    <!--param-
value>org.apache.myfaces.renderkit.html.util.NonBufferingAddResource</param-
value-->
    <!--param-
value>org.apache.myfaces.component.html.util.StreamingAddResource</param-
value-->
  </context-param>

  <context-param>
    <description>
        A very common problem in configuring MyFaces-web-applications
        is that the Extensions-Filter is not configured at all
        or improperly configured. This parameter will check for a properly
        configured Extensions-Filter if it is needed by the web-app.
        In most cases this check will work just fine, there might be cases
        where an internal forward will bypass the Extensions-Filter and the
check
        will not work. If this is the case, you can disable the check by
setting
        this parameter to false.
    </description>
    <param-name>org.apache.myfaces.CHECK_EXTENSIONS_FILTER</param-name>
    <param-value>true</param-value>
  </context-param>

  <context-param>
    <description>
        Change the url-pattern from the ExtensionsFilter
        Default is "/faces/myFacesExtensionResource"
        Note: The filter-mapping for ExtensionsFilter, the url-pattern is
        this value + "/*", else there comes a exception
    </description>
    <param-name>org.apache.myfaces.RESOURCE_VIRTUAL_PATH</param-name>
    <param-value>/faces/extensionResource</param-value>
  </context-param>

    <context-param>
      <description>
          This parameter enables partial state saving.
      </description>
      <param-name>javax.faces.PARTIAL_STATE_SAVING_METHOD</param-name>
      <param-value>false</param-value>
    </context-param>

    <context-param>
```

```
      <description>
          If true every time a page is rendered, the corresponding JSP is
dispatched also.
          This is very usefull if Scriptlets are used inside the JSP.
      </description>
      <param-name>javax.faces.PARTIAL_STATE_SAVING_DISPATCH_EVERY_TIME</param-
name>
      <param-value>true</param-value>
    </context-param>


  <filter>
    <filter-name>extensionsFilter</filter-name>
    <filter-class>org.apache.myfaces.webapp.filter.ExtensionsFilter</filter-
class>
    <init-param>
      <description>Set the size limit for uploaded files.
                Format: 10 - 10 bytes
                        10k - 10 KB
                        10m - 10 MB
                        1g - 1 GB</description>
      <param-name>uploadMaxFileSize</param-name>
      <param-value>100m</param-value>
    </init-param>
    <init-param>
      <description>Set the threshold size - files
                    below this limit are stored in memory, files above
                    this limit are stored on disk.

                Format: 10 - 10 bytes
                        10k - 10 KB
                        10m - 10 MB
                        1g - 1 GB</description>
      <param-name>uploadThresholdSize</param-name>
      <param-value>100k</param-value>
    </init-param>
  </filter>

  <filter>
       <filter-name>Acegi Filter Chain Proxy</filter-name>
       <filter-class>
              org.acegisecurity.util.FilterToBeanProxy
       </filter-class>
       <init-param>
              <param-name>targetClass</param-name>
              <param-value>
                    org.acegisecurity.util.FilterChainProxy
              </param-value>
       </init-param>
</filter>

  <filter-mapping>
    <filter-name>extensionsFilter</filter-name>
    <url-pattern>*.jsf</url-pattern>
  </filter-mapping>
  <filter-mapping>
    <filter-name>extensionsFilter</filter-name>
```

```
    <url-pattern>/faces/*</url-pattern>
  </filter-mapping>

       <filter-mapping>
              <filter-name>Acegi Filter Chain Proxy</filter-name>
              <url-pattern>/*</url-pattern>
              <dispatcher>FORWARD</dispatcher>
              <dispatcher>REQUEST</dispatcher>
       </filter-mapping>

       <!-- Listener para cargar el contexto de aplicación de Spring -->
       <listener>
              <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
       </listener>

  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet>
    <servlet-name>SourceCodeServlet</servlet-name>
    <servlet-
class>org.apache.myfaces.shared_tomahawk.util.servlet.SourceCodeServlet</servl
et-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.jsf</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>SourceCodeServlet</servlet-name>
    <url-pattern>*.source</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
       <error-page>
              <error-code>500</error-code>
              <location>/error.jsp</location>
       </error-page>
</web-app>
```

8.3.16 busquedaAuditoriaFicheros.jsp

```
<%@ page import="java.util.Random"%>
<%@ page session="false" contentType="text/html;charset=utf-8"%>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
```

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<%@ taglib uri="http://myfaces.apache.org/tomahawk" prefix="t"%>
<!doctype html>
<html>
<%@include file="inc/head.jsp"%>
<body>
 <%@include file="inc/page_header.jsp"%>
 <%@include file="inc/page_menu.jsp"%>

 <div class="container mt-5">
  <f:view>

   <h1 class="display-5">Auditoría de Ficheros Recibidos</h1>

   <h:messages id="messageList" showSummary="true" />

   <h:form id="form_busqueda">
    <div class="form-group">
     <div class="row form-group">
      <div class="col">
       <label for="nombre">Nombre</label>
       <h:inputText
        value="#{busquedaAutidoriaFicherosForm.busqueda.nombre}"
        id="nombre" styleClass="form-control"></h:inputText>
       <h:message for="nombre" showDetail="true" />
      </div>
      <div class="col">
       <label for="mimeType">Mime-Type</label>
       <h:inputText
        value="#{busquedaAutidoriaFicherosForm.busqueda.mimeType}"
        id="mimeType" styleClass="form-control"></h:inputText>
       <h:message for="mimeType" showDetail="true" />
      </div>
     </div>
     <div class="row form-group">
      <div class="col">
       <label for="fechaSubidaDesde">Fecha Subida (Desde)</label>
       <h:inputText
        value="#{busquedaAutidoriaFicherosForm.busqueda.fechaSubidaDesde}"
        id="fechaSubidaDesde" styleClass="form-control">
        <f:convertDateTime pattern="dd/MM/yyyy" />
       </h:inputText>
       <small id="fechaSubidaDesdeHelp" class="form-text text-muted">Formato:
        dd/mm/aaaa</small>
       <h:message for="fechaSubidaDesde" showDetail="true" />
      </div>
      <div class="col">
       <label for="fechaSubidaHasta">Fecha Subida (Hasta)</label>
       <h:inputText
        value="#{busquedaAutidoriaFicherosForm.busqueda.fechaSubidaHasta}"
        id="fechaSubidaHasta" styleClass="form-control">
        <f:convertDateTime pattern="dd/MM/yyyy" />
       </h:inputText>
       <small id="fechaSubidaHastaHelp" class="form-text text-muted">Formato:
        dd/mm/aaaa</small>
       <h:message for="fechaSubidaHasta" showDetail="true" />
      </div>
```

```
    </div>

    <div class="row form-group">
     <div class="col">
      <label for="usuarioSubida">Usuario</label>
      <h:inputText
       value="#{busquedaAutidoriaFicherosForm.busqueda.usuarioSubida}"
       id="usuarioSubida" styleClass="form-control"></h:inputText>
      <h:message for="usuarioSubida" showDetail="true" />
     </div>
     <div class="col">
      <label for="ipSubida">Dirección IP</label>
      <h:inputText
       value="#{busquedaAutidoriaFicherosForm.busqueda.ipSubida}"
       id="ipSubida" styleClass="form-control"></h:inputText>
      <h:message for="ipSubida" showDetail="true" />
     </div>
     <div class="col">
      <label for="valido">¿Fichero validado satisfactoriamente?</label>
      <h:selectOneListbox
       value="#{busquedaAutidoriaFicherosForm.busqueda.valido}"
       id="valido" styleClass="form-control">
       <f:selectItem itemLabel="" itemValue="" />
       <f:selectItem itemLabel="Sí" itemValue="#{true}" />
       <f:selectItem itemLabel="No" itemValue="#{false}" />
      </h:selectOneListbox>
     </div>
    </div>
   </div>

   <div class="form-group">

    <h:commandButton value="Buscar" id="buscarButton"
     action="#{busquedaAutidoriaFicherosForm.buscar}"
     styleClass="btn btn-primary" />
   </div>
  </h:form>

  <h:dataTable var="auditoria"
   value="#{busquedaAutidoriaFicherosForm.listaAuditoria}"
   id="tabla_auditoria"
   styleClass="table table-sm table-hover table-responsive-md"
   headerClass="thead-light">
   <h:column>
    <!-- column header -->
    <f:facet name="header">
     <h:outputText value="Nombre" />
    </f:facet>
    <h:outputText value="#{auditoria.nombre}" />
   </h:column>
   <h:column>
    <f:facet name="header">
     <h:outputText value="Tamaño (MB)" />
    </f:facet>
    <h:outputText value="#{auditoria.tamanioInMB}" />
   </h:column>
   <h:column>
```

```
      <f:facet name="header">
       <h:outputText value="Mime-Type" />
      </f:facet>
      <h:outputText value="#{auditoria.mimeType}" />
     </h:column>
     <h:column>
      <f:facet name="header">
       <h:outputText value="Fecha de subida del fichero" />
      </f:facet>
      <h:outputText value="#{auditoria.fechaSubida}">
       <f:convertDateTime pattern="dd/MM/yyyy HH:mm:ss" />
      </h:outputText>
     </h:column>
     <h:column>
      <f:facet name="header">
       <h:outputText value="Usuario" />
      </f:facet>
      <h:outputText value="#{auditoria.usuarioSubida}" />
     </h:column>
     <h:column>
      <f:facet name="header">
       <h:outputText value="Dirección IP" />
      </f:facet>
      <h:outputText value="#{auditoria.ipSubida}" />
     </h:column>
     <%-- -
    <h:column>
     <f:facet name="header"><h:outputText value="Hash"/></f:facet>
     <h:outputText value="#{auditoria.hashFichero}"/>
    </h:column>
    --%>
     <h:column>
      <f:facet name="header">
       <h:outputText value="Válido" />
      </f:facet>
      <%-- -<h:outputText value="#{auditoria.valido}"/>--%>
      <h:outputText value="#{auditoria.valido ? 'Sí' : 'No'}" />
     </h:column>
     <h:column>
      <f:facet name="header">
       <h:outputText value="Causa Error" />
      </f:facet>
      <h:outputText value="#{auditoria.causaError}" />
     </h:column>
    </h:dataTable>

    <h:form id="form_logout">
     <h:commandButton action="logout" value="Cerrar sesión"
      immediate="true" />
    </h:form>

    <%@include file="inc/page_footer.jsp"%>

   </f:view>
  </div>
</body>
```

```
</html>
```

8.3.17 fileupload_showimg.jsp

```
<%@ page import="java.io.File,
     java.io.InputStream,
     java.io.FileInputStream,
     java.io.OutputStream"%><%@ page session="true" %><%
 String contentType = (String)session.getAttribute("fileupload_type");
 String fileName = (String)session.getAttribute("fileupload_name");

 String allowCache = request.getParameter("allowCache");
 String openDirectly = request.getParameter("openDirectly");

 if(allowCache == null || allowCache.equalsIgnoreCase("false"))
 {
  response.setHeader("pragma", "no-cache");
  response.setHeader("Cache-control", "no-cache, no-store, must-revalidate");
  response.setHeader("Expires", "01 Apr 1995 01:10:10 GMT");
 }

 if(contentType!=null)
 {
  response.setContentType(contentType);
 }

 if(fileName != null)
 {
  fileName = fileName.substring(fileName.lastIndexOf('\\')+1);
  fileName = fileName.substring(fileName.lastIndexOf('/')+1);

  StringBuffer contentDisposition = new StringBuffer();

  if(openDirectly==null || openDirectly.equalsIgnoreCase("false"))
  {
   contentDisposition.append("attachment;");
  }

  contentDisposition.append("filename=\"");
  contentDisposition.append(fileName);
  contentDisposition.append("\"");

  response.setHeader ("Content-Disposition", contentDisposition.toString());
 }

 byte[] bytes = (byte[])session.getAttribute("fileupload_bytes");
 if (bytes != null)
 {
  response.getOutputStream().write(bytes);
 }
%>
```

8.3.18 fileupload.jsp

```
<%@ page import="java.util.Random"%>
<%@ page session="false" contentType="text/html;charset=utf-8"%>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<%@ taglib uri="http://myfaces.apache.org/tomahawk" prefix="t"%>
<!doctype html>
<html>
<%@include file="inc/head.jsp"%>
<body>
<%@include file="inc/page_header.jsp" %>
<%@include file="inc/page_menu.jsp" %>
 <div class="container mt-5">
  <f:view>

   <h2>Ejemplo de Formulario Inseguro</h2>

   <h:messages id="messageList" showSummary="true" />

   <h:form id="form1" enctype="multipart/form-data">
    <div class="form-group">

     <label for="fileupload">Seleccione un fichero</label>
     <t:inputFileUpload id="fileupload" accept="image/jpeg"
      value="#{fileUploadForm.upFile}" storage="file"
      styleClass="form-control" required="true" maxlength="200000" />
     <h:message for="fileupload" showDetail="true" />

    </div>
    <div class="form-group">

     <h:commandButton value="Adjuntar" action="#{fileUploadForm.upload}"
      styleClass="btn btn-primary" />
    </div>
   </h:form>

   <h:panelGrid columns="1" rendered="#{fileUploadForm.uploaded}">

    Fichero: <h:outputText value="#{fileUploadForm.fileName}" />

    <h:graphicImage url="fileupload_showimg.jsf"
     styleClass="img-fluid" />

    <h:outputLink value="fileupload_showimg.jsf">
     <f:param name="allowCache" value="true" />
     <f:param name="openDirectly" value="false" />
     <h:outputText value="Descargar "/> <h:outputText
value="#{fileUploadForm.fileName}" />
    </h:outputLink>

   </h:panelGrid>

   <%@include file="inc/page_footer.jsp"%>
```

```
  </f:view>
 </div>
</body>

</html>
```

8.3.19 index.jsp

```
<!doctype html>
<html lang="es">
<%@include file="inc/head.jsp"%>
<body>

 <%@include file="inc/page_header.jsp"%>
 <%@include file="inc/page_menu.jsp"%>

 <div class="container mt-5">
  <h1 class="display-5">Pruebas Formularios con Ficheros</h1>
  <p>A continuación se muestran dos tipos de formularios, uno de
   ellos sin ningún tipo de validación para mostrar las vulnerabilidades
   y otro formulario seguro, en el que hemos corregido dichas
   vulnerabilidades</p>
  <h3>Formulario Inseguro</h3>
  <p>No se realiza ningún tipo de validación sobre el fichero. Se
   permite por lo tanto:</p>
  <ul>
   <li>Adjuntar ficheros de cualquier tipo, incluidos ejecutables:
    .exe, .sh.</li>
   <li>Adjuntar ficheros de cualquier tamaño</li>
   <li>Adjuntar ficheros cuyo nombre tenga caracteres de separación
    de directorios: ".." y "/" en Linux o "\" en Windows. Consultar
    Path-Traversial
  </ul>
  <p>
   <a href="fileupload.jsf" class="btn btn-secondary">Acceso al
    Formulario Inseguro</a>
  </p>
  <h3>Formulario Seguro</h3>
  <p>Se realizan las siguientes validaciones sobre el fichero:</p>
  <ul>
   <li>Valida que el tamaño no supere los 2MB</li>
   <li>Valida el tipo de contenido analizando los primeros bytes
    del fichero y no la extensión: No nos pueden engañar cambiando la
    extensión al nombre del fichero.</li>
   <li>En el nombre del fichero solo permitimos caracteres
    alfanuméricos y los separadores "-" y "_". No permitimos más de un
    "." para evitar la doble extensión de archivos.
   <li>Se realiza auditoría de todos los ficheros recibidos</li>
  </ul>
  <p>
   <a href="securefileupload.jsf" class="btn btn-secondary">Acceso
    al Formulario Seguro</a>
```

```
  </p>
 </div>
</body>
</html>
```

8.3.20 login.jsp

```
<%@taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<%@taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<%@taglib uri="http://myfaces.apache.org/tomahawk" prefix="t"%>
<!doctype html>
<html>
<%@include file="inc/head.jsp"%>
<body class="text-center">
<%@include file="inc/page_header.jsp" %>
<div class="container">
 
</div>
 <div class="container">
  <f:view>
   <h1 class="h3 mb-3 font-weight-normal">Acceso al formulario
    seguro</h1>
   <h:form styleClass="form-signin">

    <div class="form-group row">
     <h:outputLabel value="Usuario" for="j_username"
      styleClass="col-sm-2 col-form-label" />
     <div class="col-sm-10">
      <t:inputText id="j_username" forceId="true"
       value="#{loginBacking.userId}" size="40" maxlength="80"
       styleClass="form-control"></t:inputText>
     </div>
    </div>
    <div class="form-group row">
     <h:outputLabel value="Password" for="j_password"
      styleClass="col-sm-2 col-form-label" />
     <div class="col-sm-10">
      <t:inputSecret id="j_password" forceId="true"
       styleClass="form-control" value="#{loginBacking.password}"
       size="40" maxlength="80" redisplay="true"></t:inputSecret>
     </div>
    </div>

    <h:commandButton action="login" value="Login"
     styleClass="btn  btn-primary " />
    <h:messages id="messages" layout="table" globalOnly="true"
     showSummary="true" showDetail="false" />
   </h:form>
  </f:view>

 </div>

</body>
```

```
</html>
```

8.3.21 securefileupload.jsp

```
<%@ page import="java.util.Random"%>
<%@ page session="false" contentType="text/html;charset=utf-8"%>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<%@ taglib uri="http://myfaces.apache.org/tomahawk" prefix="t"%>
<!doctype html>
<html>
<%@include file="inc/head.jsp"%>
<body>
<%@include file="inc/page_header.jsp" %>
<%@include file="inc/page_menu.jsp" %>
 <div class="container mt-5">
  <f:view>
   <h2>
    Ejemplo de Formulario Seguro
   </h2>

   <p>Se realizan las siguientes validaciones sobre el fichero:</p>
   <ul>
    <li>Valida que el tamaño no supere los 2MB</li>
    <li>Valida el tipo de contenido analizando los primeros bytes del fichero
y no la extensión: No nos pueden engañar cambiando la extensión al nombre del
fichero.</li>
    <li>En el nombre del fichero solo permitimos caracteres alfanuméricos y
los separadores "-" y "_". No permitimos más de un "." para evitar la doble
extensión de archivos.
    <li>Se realizará auditoría de los ficheros recibidos</li>
   </ul>

   <h:messages id="messageList" showSummary="true" errorClass="alert alert-
danger" infoClass="alert alert-danger" layout="table"/>
   <h:form id="form1" enctype="multipart/form-data">
    <div class="form-group">
    <label for="fileupload">Seleccione un fichero</label>
    <t:inputFileUpload id="fileupload"
     value="#{secureFileUploadForm.upFile}" storage="file"
     styleClass="form-control" required="true" maxlength="200000"
     validator="#{secureFileUploadForm.validateFile}" />
    <h:message for="fileupload" showDetail="true" />
    </div>
    <div class="form-group">
    <h:commandButton value="Adjuntar" id="adjuntar-button"
     action="#{secureFileUploadForm.upload}" styleClass="btn btn-primary"/>
    </div>
   </h:form>

   <h:panelGrid columns="1" rendered="#{secureFileUploadForm.uploaded}">
    Fichero: <h:outputText value="#{secureFileUploadForm.fileName}" />
    <h:graphicImage url="fileupload_showimg.jsf" styleClass="img-fluid" />
```

```
    <h:outputLink value="fileupload_showimg.jsf">
     <f:param name="allowCache" value="true" />
     <f:param name="openDirectly" value="false" />
     <h:outputText value="Descargar "/> <h:outputText
value="#{secureFileUploadForm.fileName}" />
    </h:outputLink>

   </h:panelGrid>

   <h:form id="form_logout">
    <h:commandButton action="logout" value="Cerrar sesión" immediate="true"/>
   </h:form>

   <%@include file="inc/page_footer.jsp"%>

  </f:view>
 </div>
</body>

</html>
```