UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TECNOLOGÍA ESPECÍFICA DE

TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

Pentesting Web, extendiendo la funcionalidad de Burp Suite

D. Gino Paulo Angles Chao

Junio de 2017

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TECNOLOGÍA ESPECÍFICA DE

TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

Pentesting Web, extendiendo la funcionalidad de Burp Suite

Autor: D. Gino Paulo Angles Chao

Director: Dr. D. José Luis Martínez Martínez

Junio de 2017

Declaración de Autoría

Yo, D. Gino Paulo Angles Chao con DNI 23316433-E, declaro que soy el único autor del trabajo fin de grado titulado "Pentesting Web, extendiendo la funcionalidad de Burp Suite" y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual y que todo el material no original contenido en dicho trabajo está apropiadamente atribuido a sus legítimos autores.

Albacete, a 15 de Junio de 2017

Fdo.: D. Gino Paulo Angles Chao

Resumen

En este trabajo de fin de grado, hablo sobre las distintas técnicas y vulnerabilidades que existen en las aplicaciones y tecnologías web, las cuales podrían ser utilizadas por un atacante con el fin de irrumpir en un sistema informático, y acceder a información sensible e incluso modificarla o bórrala del sistema.

Además de describir dichas técnicas y vulnerabilidades desde un punto de vista teórico, realizo una prueba de concepto demostrando como podrían realizarse en un hipotético caso. También hablo sobre algunas herramientas que nos pueden ayudar a detectar dichas vulnerabilidades en una aplicación web.

Explico cómo nos pueden ayudar estas herramientas a la hora de realizar una auditoría a una aplicación web o una prueba de Pentesting. En una de estas herramientas, más concretamente, en Burp Suite, desarrollo un Plug-in con el fin de ampliar la funcionalidad base de dicha aplicación.

Por último, realizo una prueba de concepto del Plug-in desarrollado sobre alguna funcionalidad añadida para ver como puede ser de utilidad para la labor de los Pentester, desarrolladores e investigadores que utilizan la propia herramienta.

Agradecimientos

Agradecer a mi tutor, Dr. D. José Luis por todo su tiempo, interés y esfuerzo que ha dedicado para ayudarme en todo lo posible, y por darme la oportunidad de poder realizar este trabajo.

A mis amigos, que son los hermanos que elegimos, Diego, Pedro, Sergio, Ramón, Arago, Rubén y Adrian, por su apoyo incondicional.

A todos mis compañeros y gente que he conocido durante estos cuatro años, por hacer el camino un poco más ameno.

A todo el profesorado de la Escuela Superior de Informática, por sus guías, su paciencia, dedicación y enseñanzas.

Dedicatorias

A mi familia, en especial a mis padres, Ling y Alfonso, por estar siempre ahí.

ÍNDICE

CAPÍTULO 1. INTRODUCCIÓN 1
1.1 MOTIVACIÓN 1
1.2 OBJETIVOS 2
1.3 ESTRUCTURA DE LA MEMORIA 2
CAPÍTULO 2. FUNDAMENTOS TEÓRICOS / TECHNICAL BACKGROUND 5
2.1 SEGURIDAD INFORMÁTICA 5
2.2 SEGURIDAD EN APLICACIONES WEB 6
2.2.1 Motivación 7
2.2.2 Arquitectura de las aplicaciones Web 7
2.3 VULNERABILIDADES EN APLICACIONES WEB 9
2.3.1 Centradas en el lado del cliente 9
2.3.2 Centradas en el lado del servidor 13
2.3.3 Otros vectores de ataque 14
2.4 AUDITORÍA DE SEGURIDAD INFORMÁTICA 18
2.5 PENTESTING 19
2.5.1 ¿Qué es una prueba de Pentesting? 19
2.5.2 ¿Es legal? 19
2.5.3 Fases de una prueba de Pentesting 19
2.6 ALTERNATIVAS 21
2.6.1 OSSTMM 21
2.6.2 OWASP 21
2.7 DESARROLLO SEGURO DE APLICACIONES WEB 22
CAPÍTULO 3. HERRAMIENTAS PARA PENTESTING WEB 27
3.1 ESCANER DE VULNERABILIDADES WEB 27
3.1.1 W3AF 27
3.1.2 Acunetix 29
3.1.3 Nessus 31
3.2 PROXY WEB 33
3.2.1 OWASP ZAP 34
3.2.2 Burp Suite 38
CAPÍTULO 4. ESTADO DEL ARTE 45
4.1 OTROS PLUG-IN DE BURP SUITE 45
4.1.1 CO2 45
4.1.2 Headers Analyzer 47
4.1.3 JSON Decoder 48
4.1.4 EsPReSSO 48
4.1.5 SQLiPy 50
CAPÍTULO 5. PRUEBAS DE CONCEPTO DE VULNERABILIDADES 51
5.1 Inyección 51
5.2 Pérdida de Autenticación y Gestión de Sesiones 58

5.3 Secuencia de Comandos en Sitios Cruzados (XSS) 62
5.4 Referencia directa insegura a objetos 66
5.5 Configuración de Seguridad Incorrecta 69
5.6 Exposición de datos sensibles 71
5.7 Inexistente Control de Acceso a nivel de funcionalidades 73
5.8 Falsificación de Peticiones en Sitios Cruzados (CSRF) 74
5.9 Uso de Componentes con Vulnerabilidades Conocidas 77
5.10 Redirecciones y reenvíos no válidos 78

CAPÍTULO 6. INTEGRACIÓN PLUG-IN A BURP SUITE 79

6.1 API de Burp Suite 79
6.2 ¿Por qué necesitaría una extensión personalizada? 80
6.3 Interfaz de usuario de L-Tools 81
6.4 Implementación 87
6.5 Limitaciones 90

CAPÍTULO 7. EVALUACIÓN DE PRESTACIONES 91

7.1 Mashup, ataque por fuerza bruta 91
7.2 Redirect, redirigiendo tráfico 99
7.3 Loader, cargando una lista de objetivos 100
7.4 Headers, analizando cabeceras HTTP 101
7.5 Quick Scan, escaneando hosts 102

CAPÍTULO 8. CONCLUSIONES Y PROPUESTAS 105

8.1 CONCLUSIONES 105
8.2 TRABAJO FUTURO Y POSIBLES AMPLIACIONES 106

BIBLIOGRAFÍA 107

LIBROS Y ARTICULOS 107
ENLACES INTERNET 112

GLOSARIO DE TÉRMINOS 115

CONTENIDO DEL CD 119

ANEXO A. CÓDIGO FUENTE DE LA EXTENSIÓN 121

ANEXO B. INSTALACIÓN DE PLUG-IN 141

ÍNDICE DE FIGURAS

Figura 1. Ejemplo petición HTTP 8

Figura 2. Esquema Cliente – Servidor 8

Figura 3. Esquema XSS Persistente [f3] 10

Figura 4. ClickJacking Ejemplo [f4] 11

Figura 5. WebTrojan Php oculto dentro de un .gif 13

Figura 6. Resumen comandos búsqueda avanzada en Google 15

Figura 7. Ejemplo sencillo Google hacking 16

Figura 8. Pantalla inicial W3AF. 28

Figura 9. Ejemplo de SQLi encontrada [f9] 29

Figura 10. Tipos de escaneos que permite Acunetix. 30

Figura 11. Reporte ejecutivo de Acunetix 31

Figura 12. Políticas y Tipos de escaneos que permite NESSUS. 32

Figura 13. Resultado de escáner NESSUS. 32

Figura 14. Esquema cliente – servidor con proxy web. 33

Figura 15. Ejemplo configuración Proxy Web en Firefox. 34

Figura 16. OWASP ZAP proxy Response HTTP 35

Figura 17. OWASP ZAP, Configurar Fuzzer con ficheros predefinidos. 37

Figura 18. Burp Suite WorkFlow [f18] 38

Figura 19. Burp Suite – Target Site Map. 39

Figura 20. Burp Suite – Escáner de vulnerabilidades. 40

Figura 21. Burp Suite – Proxy HTTP History 42

Figura 22. Burp Suite Extensión – CO2-SQLMapper 46

Figura 23. Burp Suite Extension – Headers Analyzer 47

Figura 24. EsPReSSO – SAML Editor [f24] 49

Figura 25. Burp Suite SQLMap extensión – SQLMap Logs 50

Figura 26.1 PoC SQL Inyection 52

Figura 27.1 PoC Username Enumeration. 59

Figura 28.1 PoC XSS Stored 63

Figura 29.1 PoC Insecure Direct Object References 67

Figura 30.1 PoC Exposición de datos sensibles 72

Figura 31.1 PoC CSRF 75

Figura 32. L-Tools – Mashup UI 81

Figura 33. L-Tools – Redirect UI 82

Figura 34. L-Tools – Loader UI 83

Figura 35. L-Tools – Headers UI 84

Figura 36. L-Tools – Quick Scan UI 85

Figura 37. L-Tools About UI 86

Figura 38. Interfaces utilizadas de la API de Burp Suite en la extensión. 89

Figura 39. Generando diccionario de contraseñas. 92

Figura 40. Modal de feedback, al copiar al portapapeles. 92

Figura 41. Maquina virtual DVWA, Brute Force. 93

Figura 42. Mandando página objetivo al Intruder. 93

Figura 43. Intruder – Positions. 94

Figura 44. Configurando Payload. 95

Figura 45. Resultado del ataque 1a. 95

Figura 46. Resultado del ataque 1b. 96

Figura 47. Resultado del ataque 1c. 97

Figura 48. Resultado del ataque 1d. 97

Figura 49. Usuario y contraseña correctos. 98

Figura 50. L-Tools – Redirect. 99

Figura 51. Proxy web, redirigiendo el tráfico. 99

Figura 52. L-Tools – Loader. 100

Figura 53. Burp Suite Target – Scope. 100

Figura 54. L-Tools – Headers. 101

Figura 55. Php.ini expose_php = Off. 102

Figura 56. L-Tools – Quick Scan. 102

Figura 57. Burp escaneando objetivos. 103

Figura 58. Resultado del escáner. 103

ÍNDICE DE TABLAS

Tabla 1. PoC Inyección esquema. 52

Tabla 2. PoC Pérdida de autenticación y gestión de sesión esquema. 58

Tabla 3. PoC Secuencia de Comandos en Sitios Cruzados (XSS) 62

Tabla 4. PoC Referencia directa insegura a objetos. 66

Tabla 5. PoC Configuración de Seguridad Incorrecta. 69

Tabla 6. PoC Exposición de datos sensibles. 71

Tabla 7. PoC Inexistente Control de Acceso a nivel de funcionalidades. 73

Tabla 8. PoC Falsificación de Peticiones en Sitios Cruzados (CSRF). 74

Tabla 9. PoC Uso de Componentes con Vulnerabilidades Conocidas. 77

Tabla 10. PoC Redirecciones y reenvíos no válidos. 78

Tabla 11. Versiones Software utilizadas. 89

LISTA DE ACRÓNIMOS

GENERAL

CMS	Content Management System
HTTP	HyperText Transfer Protocol
HTML	HyperText Markup Language
JS	JavaScript
CSS	Cascading StyleSheets
SQL	Structured Query Language
SSL	Secure Sockets Layer
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
CMD	Command
XML	Extensible Markup Language
LAN	Local Area Network
VM	Virtual Machine
AES	Advanced Encryption Standard
RSA	Rivest, Shamir y Adleman
MD5	Message-Digest Algorithm 5
SHA1	Secure Hash Algorithm 1
MVC	Model-View-Controller

PoC	Proof of Concept
AJAX	Asynchronous Javascript And XML
API	Application Programming Interface
CVE	Common Vulnerabilities and Exposures
JSON	JavaScript Object Notation

VULNERABILIDADES

XSS	Cross-Site Scripting
CSRF	Cross-Site Request Forgery
LFI	Local File Inclusion
RFI	Remote File Inclusion
SQLi	SQL inyection
MITM	Man In The Middle
HSTS	HTTP Strict Transport Security
HPKP	HTTP Public Key Pinning

ORGANIZACIONES

OWASP	Open Web Application Security Project
ISO	International Standarization Organization
CEH	Certified Ethical Hacker

CAPÍTULO 1. INTRODUCCIÓN

En este capítulo, introduciré la motivación que me ha llevado a realizar este trabajo de fin de grado, los objetivos de los que parto, y la estructura que tendrá esta memoria con el fin de guiar un poco al lector.

1.1 MOTIVACIÓN

Desde hace unos años y con la aparición de la web 2.0 (Interactiva), cada vez son más las aplicaciones web que se desarrollan para cubrir una necesidad o incluso para crearla. Dado que solo necesitamos un sistema operativo (hoy en día en cualquier ordenador, coche, móvil y pronto casa (domótica)) con un navegador web o cliente que se pueda conectar e iniciar esa comunicación entre cliente-servidor que caracteriza a casi todas las aplicaciones web, se puede decir que las aplicaciones web y por ende la seguridad en las mismas están en un auge constante.

Tenemos aplicaciones web desde paneles de control de tu propio Router hasta las populares redes sociales, pasando por gmail, outlook365 y muchas otras en las cuales se transmite o se trabaja con información personal o profesional muy sensible, saber cómo proteger dicha información o saber cómo podría actuar un hacker con malas intenciones nos puede ayudar, como usuarios aprendiendo como configurar correctamente tu navegador o como desarrollador creando código seguro y eficiente.

Saber que técnicas existen o que vulnerabilidades son las más usuales en dichas aplicaciones y tecnologías web, nos pueden servir para realizar una auditoría a una aplicación web o incluso para hacer pruebas de Pentesting e intentar acceder a una aplicación web con la finalidad de saber qué puntos débiles existen en dicha aplicación y como poder solucionarlos o fortificarlos, y así poder garantizar la seguridad de la

información con la que trabaja nuestra aplicación y por consecuente la privacidad de nuestros clientes y/o usuarios e incluso de nuestra propia empresa.

1.2 OBJETIVOS

Mis objetivos principales en este Trabajo Fin de Grado son profundizar sobre las vulnerabilidades de seguridad más comunes en la actualidad que afectan a la mayoría de aplicaciones web y desarrollar una extensión para la herramienta de Burp Suite (La cual introduciré en el capítulo 3) que pueda servir para ayudar a la labor de los Pentester, desarrolladores e investigadores que utilizan la propia herramienta. Se pueden identificar los siguientes sub-objetivos:

- Estudio de las diferentes vulnerabilidades más comunes que afectan a la mayoría de las aplicaciones web desde un punto de vista teórico.
- Estudio de las etapas de una prueba de intrusión o Pentesting.
- Investigación sobre buenas prácticas a la hora de desarrollar código seguro.
- Investigación de diferentes herramientas para el escáner de vulnerabilidades.
- Pruebas de concepto de vulnerabilidades conocidas, desde un punto de vista práctico.
- Desarrollo e integración de una extensión para Burp Suite.

1.3 ESTRUCTURA DE LA MEMORIA

La memoria está dividida en ocho capítulos, además de la bibliografía, glosario de términos y anexos pertinentes.

CAPÍTULO 1: INTRODUCCIÓN

El primer capítulo se corresponde con la introducción, viene a describir un poco la motivación que se tiene en cuenta para realizar este trabajo, los objetivos que se cubren y la propia estructura del mismo.

CAPÍTULO 2: FUNDAMENTOS TEÓRICOS

En el segundo capítulo tenemos los fundamentos teóricos necesarios para poder entender y llevar a cabo este trabajo, abordando los conceptos de seguridad informática, seguridad web, auditoria y vulnerabilidades, entre otras cosas, que son la base para los siguientes capítulos.

CAPÍTULO 3: HERRAMIENTAS PARA PENTESTING WEB

El tercer capítulo trata de describir algunas de las diferentes herramientas que tenemos hoy en día para abordar el proceso de securizar una aplicación web o auditar una aplicación web, comentando algunas de sus características principales o que las hace diferentes a las demás, etc.

CAPÍTULO 4: ESTADO DEL ARTE

En el cuarto capítulo tendremos algunas de las extensiones más populares de la herramienta Burp Suite (la cual se introducirá en el capítulo tercero.) para ampliar más su funcionalidad base.

CAPÍTULO 5: PRUEBAS DE CONCEPTO DE VULNERABILIDADES

El quinto capítulo se corresponde a las diez vulnerabilidades más habituales en toda aplicación web, se realizara una prueba de concepto de dichas vulnerabilidades, el esquema que sigue cada una, diferentes consejos o formas de evitarlas, y otros ejemplos relacionados con las mismas.

CAPÍTULO 6: INTEGRACIÓN PLUG-IN A BURP SUITE

El capitulo sexto mostrara el desarrollo de una herramienta o Plug-in a modo de extender la funcionalidad base de Burp Suite, escrito en un lenguaje de muy alto nivel, gracias a la propia API que nos proporciona dicha herramienta.

CAPÍTULO 7: EVALUACIÓN DE PRESTACIONES

En el capitulo siete mostrare un posible flujo de trabajo para cada una de las funcionalidades desarrolladas en la extensión, mediante diferentes pruebas de concepto, para ver más en detalle el funcionamiento practico de la extensión desarrollada en el capítulo seis.

CAPÍTULO 8: CONCLUSIONES Y PROPUESTAS

Y por último, tenemos el capitulo ocho, el cual se corresponde con las conclusiones alcanzadas, además de las posibles ampliaciones de este trabajo.

CAPÍTULO 2. FUNDAMENTOS TEÓRICOS / TECHNICAL BACKGROUND

En este capítulo introduciré una serie de conceptos teóricos que nos servirán de base para los siguientes capítulos, sin entrar en mucho detalle, ya que son conceptos muy abstractos de los cuales se podrían ampliar mucho más, hablare de la seguridad informática, seguridad en aplicaciones web, las vulnerabilidades en aplicaciones web, las auditorias, el desarrollo seguro de código, y otros procesos para securizar una aplicación web como puede ser el realizado por la organización de OWASP que veremos más adelante.

2.1 SEGURIDAD INFORMÁTICA

Es el área de la informática que se centra en la protección de la información que buscan mantener la confidencialidad, la disponibilidad e integridad de la misma. Podemos enfocarlo desde un punto de vista lógico (Desarrollando software seguro, testeado, etc.) y/o físico (Protegiendo el acceso físico a los emplazamientos donde se encuentren los equipos informáticos, controlando los puertos de un switch, etc.).

Cabe a destacar que no existe un sistema seguro al 100 %, y que la seguridad se puede ver como un proceso que nunca termina ya que siempre aparecen nuevos riesgos y/o vulnerabilidades, pero se pueden estudiar formas de gestionar dichos riesgos (p.e. políticas de contingencia) [1].

Algunos de los atributos o propiedades típicos de la seguridad, son los siguientes:

Confidencialidad

Es la propiedad por la cual, la información solo puede ser accesible por aquellas personas o sistemas que estén autorizados o tengan permisos.

Disponibilidad

Es la propiedad por la cual, la información debe poder ser accesible siempre y cuando se necesite de ella.

Integridad

Es la propiedad por la cual, la información solo puede ser alterada por aquellas personas o sistemas que estén autorizados, ya sea en su almacenamiento y posterior lectura o cuando la información es transmitida.

No Repudio

Es la propiedad por la cual, el acceso, modificación o creación de una información por cuenta de un usuario o un sistema debe quedar reflejado de tal manera, que no se pueda negar la intervención de dicho usuario.

Autenticidad

Es la propiedad por la cual, podemos verificar que un usuario o sistema es realmente quien dice ser, y no es un tercero que pueda estar intentando suplantar dicha identidad [2].

2.2 SEGURIDAD EN APLICACIONES WEB

Antes hablamos desde la perspectiva de la seguridad informática, que engloba tanto la protección de activos digitales, como la protección de cualquier acceso no autorizado a dispositivos físicos, ahora nos centraremos en una parte o rama de la misma, la seguridad en aplicaciones o tecnologías web.

Se puede decir que el concepto viene a ser muy similar, pero enfocado únicamente a todo lo que engloba el desarrollo, despliegue e integración de cualquier aplicación web y el uso de este por los usuarios o sistemas, en la World Wide Web. Es uno de los puntos más sensibles de la seguridad ya que interactúan directamente con el usuario, normalmente a través de un navegador, y aunque los navegadores estén “separados lógicamente” del propio sistema operativo, un ataque gracias a una vulnerabilidad que

pueda romper la seguridad y conseguir acceso a través del navegador, podría comprometer y poner en riesgo todo el sistema [3].

Normalmente, la mayoría de estas vulnerabilidades que pueden comprometer una aplicación web, pueden ser solventadas en la fase de desarrollo, es decir, mayoritariamente son generadas por las malas prácticas a la hora de escribir o generar código por parte de los desarrolladores o programadores [4]. Esto no quiere decir que, si se genera un código perfecto libre de errores, con sus respectivos filtros que evitan escapar caracteres, se controlan todas las entradas y salidas del sistema, etc., te evite que alguien pueda ir físicamente a donde tienes "corriendo" el servidor de base de datos que usa tu aplicación web y manipule la información directamente. O incluso si utilizas un Hosting con su servidor dedicado o uno en la nube, el cual es compartido por otras aplicaciones web, que puedan atacar a otra aplicación web, que no sea la tuya, y termine afectando la tuya (ya que están bajo el mismo sistema), etc.

2.2.1 Motivación

Cada vez más, nos dan facilidades a la hora de crear aplicaciones web ya sea con gestores de contenido (CMS) o con librerías y/o frameworks como Bootstrap que permiten que los elementos de tu interfaz de tu aplicación web se organicen dependiendo del dispositivo que lo visualice, en otras palabras, el contenido de tu pagina web se adapte al tamaño de la pantalla o resolución que este visualizando dicho contenido [5].

Cabe a destacar también, que casi cualquier panel de control, tiene su versión web, por ejemplo cuando entras a modificar los parámetros de configuración de tu Router, o cuando configuras tus cuentas de usuarios de base de datos (p.e. phpmyadmin), o cuando controlas una Botnet, y un largo etc.

Hay que tener claro cómo funcionan estas tecnologías, no solo para saber cómo usarlas y configurarlas adecuadamente como usuarios, sino también, como podría usarlas un atacante o como podría hacer una auditoria web o una prueba de Pentesting sobre un dominio o una aplicación concreta para saber qué puntos débiles existen y como podría fortificarlos.

2.2.2 Arquitectura de las aplicaciones Web

La arquitectura de las aplicaciones web normalmente se basan en un esquema cliente-servidor, la idea es muy simple, el cliente (normalmente un navegador web que desea visitar alguna página o aplicación) inicia la comunicación enviándole una petición

al servidor (una dirección IP o un nombre de dominio), el servidor procesa la petición y en base a esta, le responde.

```
POST /bWAPP/login.php HTTP/1.1
Host: 192.168.1.7
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:43.0) Gecko/20100101 Firefox/43.0 Iceweasel/43.0.4
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.7/bWAPP/login.php
Cookie: PHPSESSID=o4gjt4pocb3prr3hr50aj0hv42; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; security_level=0
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 51

login=bee&password=bug&security_level=0&form=submit
```

Figura 1. Ejemplo petición HTTP

Esta comunicación se realiza mediante el protocolo HTTP o HTTPS (que es la versión cifrada del anterior, la versión segura). En sus inicios, cuando se pensó en crear este flujo de información, no se tuvo en cuenta si dicha comunicación debía "viajar en texto plano" o debía ir cifrada, para que, en caso de que un tercero no autorizado escuche el canal de transmisión, no pueda saber o interpretar la comunicación. Por esta razón se creó el protocolo HTTPS que está basada en el uso de certificados para cifrar/descifrar la comunicación que se transmite y así evitar las escuchas no autorizadas [6].

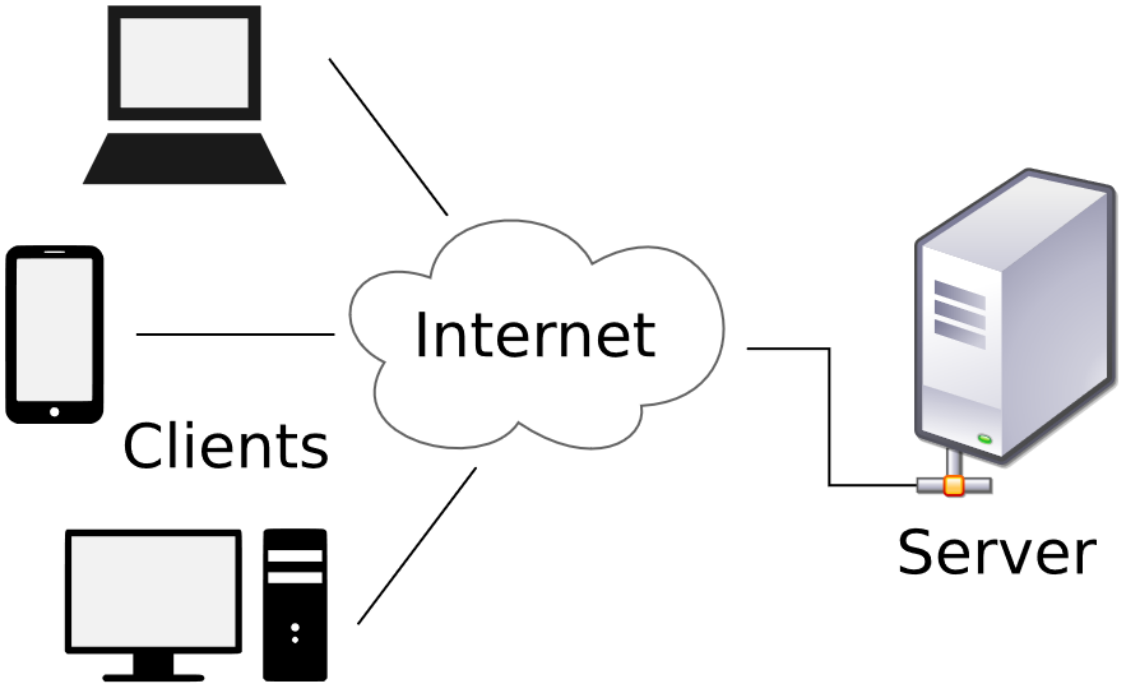


Figura 2. Esquema Cliente – Servidor

Una aplicación web se puede dividir en dos grandes partes, la parte que se ejecuta en el cliente o navegador (HTML, Javascript, Css, etc.) y la parte que se ejecuta en el servidor (Python, Php, Java, SQL, MongoDB, etc.) [7]. Como veremos más adelante, hay

vulnerabilidades que pueden afectar a una parte y/u otra, pudiendo comprometer todo el sistema que está sirviendo dicha aplicación.

Cabe a destacar que si la comunicación se realiza mediante HTTP, y un tercero (hacker) escucha el canal de transmisión, podría ver que información se transmite, por ejemplo si es una compra online, el hacker podría obtener el numero de su tarjeta de crédito o visa además de su número pin. Más adelante veremos cómo, hay técnicas como el SSL Strip o utilizar un proxy web, que te permiten colocarte en medio de dicha comunicación y aunque utilices HTTPS, poder ver dicha comunicación igualmente.

2.3 VULNERABILIDADES EN APLICACIONES WEB

Hablamos de una vulnerabilidad en informática, cuando un sistema hardware o software tiene una debilidad o deficiencia, que potencialmente puede ser utilizada por un atacante, para conseguir acceso y/o para romper la seguridad de dicho sistema.

A continuación desglosare algunas de las técnicas más comunes que explotan dichas vulnerabilidades en aplicaciones o tecnologías web, desde el punto de vista teórico, que son y a que afectan o con que finalidad, más adelante en el capítulo 5, mostrare algunas pruebas de concepto de estos ataques.

2.3.1 Centradas en el lado del cliente

Cross Site Scripting – XSS

Es un tipo de ataque que aprovecha la vulnerabilidad de poder inyectar código (HTML, Javascript, flash, etc.) que normalmente es interpretado por el navegador, en un formulario, pero también podría ir en la URL o URI, con la finalidad de poder acceder a información sensible que es almacenada temporalmente por el navegador, como podrían ser las cookies, los tokens de sesión, etc.

Dado que el navegador no puede saber si el script que está ejecutando viene o no de una fuente fiable, este lo ejecutara de todos modos, una solución puede ser controlar todos los formularios a los que el usuario puede acceder y enviar información al servidor, añadiendo unas validaciones previas en la parte del cliente (Javascript) que verifiquen que la información que se está enviando es realmente legitima o la que se esperaría.

Se pueden distinguir dos tipos, XSS persistente y XSS reflejado. Los persistentes son los que se quedan almacenados en el servidor y tienen persistencia. Por ejemplo si podemos inyectar código en un blog o similar, en el que la información que se envía a

través del formulario va quedar almacenada en la base de datos con la que trabaja la aplicación. Estos son más peligrosos ya que, cada vez que un usuario acceda a dicho blog, el navegador cargara el script que haya inyectado el hacker.

Los reflejados son los que no tienen persistencia en el lado del servidor, normalmente se utilizan para poder acceder a información sensible con la cual esta trabajado la sesión actual del navegador. Por ejemplo si podemos inyectar código en una pantalla de login, podríamos sacar información del navegador con un pequeño script, pero este no se almacenara en la base de datos [8].

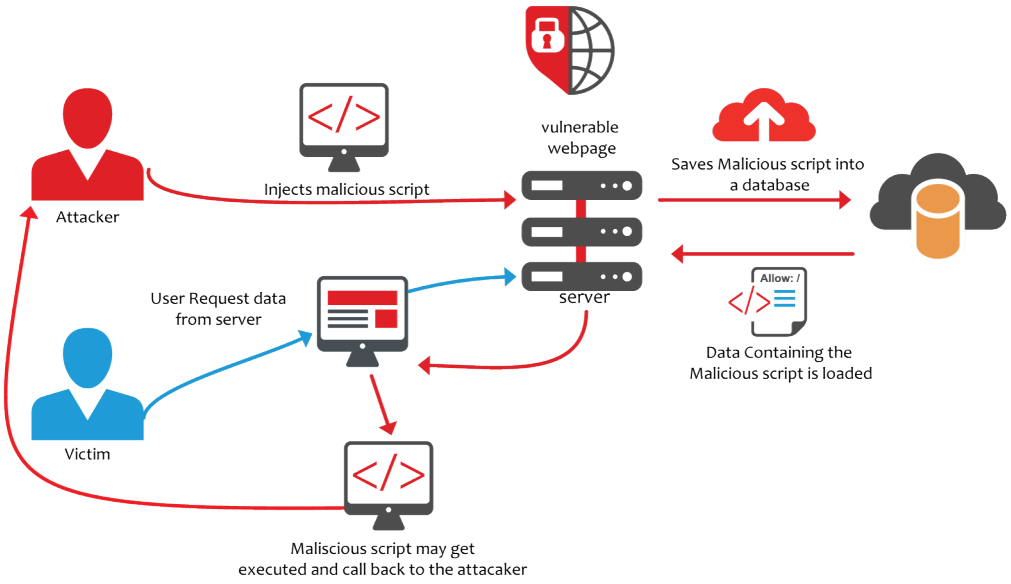


Figura 3. Esquema XSS Persistente [f3]

Cross Site Request Forgery – CSRF

Se puede decir que es una variante del XSS, la diferencia es que el atacante fuerza a la victima (normalmente otro usuario final que esta navegando o accediendo a un panel de control) a realizar acciones no deseadas, en una aplicación web en la que la victima ya esta autentificada. Se apoyan bastante en la Ingeniería social [9], para conseguir engañar a la víctima y que esta acceda a un enlace o un visite un sitio, y así poder realizar una acción que el atacante decida.

Por ejemplo, si estamos en la cuenta de nuestro banco vía online, el atacante podría conseguir que la víctima le transfiera dinero, o si estamos en un gmail, el atacante podría conseguir cambiar la dirección de correo o la contraseña de la víctima, etc. Si además el usuario es un administrador con privilegios sobre la propia aplicación, se podría comprometer toda la aplicación web [10].

ClickJacking

Es un tipo de ataque que se aprovecha de algunas etiquetas de HTML en las cuales el atacante puede superponer varias capas o contenedores (por ejemplo <DIV> o <IFRAME>) transparentes o invisibles al usuario, para hacerle creer que esta clicando sobre un enlace o botón que realiza una acción, cuando en realidad hay otro enlace o botón superpuesto encima que es invisible para el usuario y que realiza la acción que desea el atacante. Normalmente el atacante es el propio desarrollador o dueño de la página web o dominio.

Por ejemplo, imagina que el atacante construye un sitio web en el cual aparece un botón que pone "pincha aquí para IPhone gratis", y que antes de dicho botón, carga un <iframe> con tu cuenta de correo, y superpone exactamente el botón de "borrar todos los mensajes" encima del otro botón, pero este <iframe> es transparente, entonces cuando la víctima pinche sobre el botón de "pincha aquí para IPhone gratis", realmente estará borrando todos sus mensajes de su cuenta de correo [11].

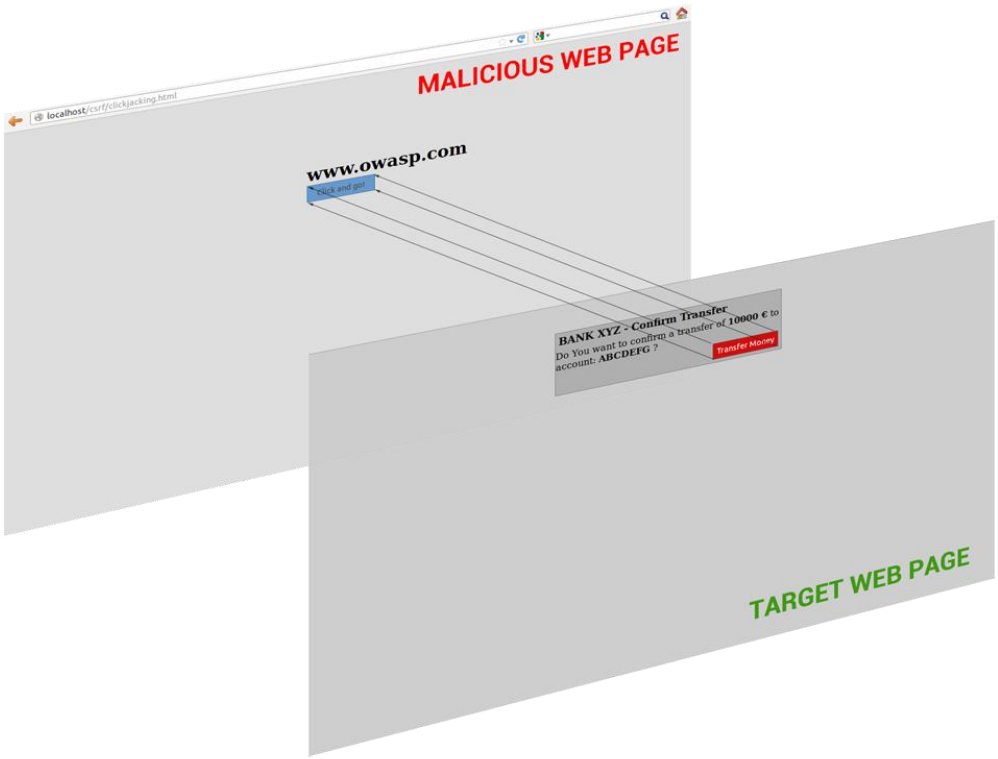


Figura 4. ClickJacking Ejemplo [f4]

Remote File Inclusion

Este ataque se aprovecha de la posibilidad de poder incrustar o incluir ficheros remotos en una aplicación web objetivo. La vulnerabilidad aparece cuando el fichero se incluye y se ejecuta sin que este tenga una validación y verificación apropiada.

Por ejemplo, en Php, el típico “include (fichero.php);” que permite ampliar la funcionalidad del fichero que lo carga, ahora imaginemos que el parámetro “fichero.php” es un parámetro de entrada de tipo fichero y se carga dentro de ese include, sino se hacen comprobaciones previas a que es lo que está subiendo el usuario o atacante, este podría subir cualquier tipo de fichero con malas intenciones que puedan comprometer el sistema [12].

Local File Inclusion

Este ataque busca incluir o redirigir ficheros locales que ya están presentes en el lado del servidor, y así conseguir, por ejemplo, conseguir información sensible que pueda estar almacenada en algún fichero del propio servidor.

Por ejemplo, tenemos la URI de una aplicación web del estilo “http://vulnerable_host/visualizar.php?file=example.html”, en el que "example.html" es el parámetro que recibe y el que visualiza, ahora por ejemplo podríamos probar a subir a la raíz de dicho proyecto y ver si tiene el fichero “/etc/passwd” presente en los sistemas Linux, probaríamos “http://vulnerable_host/visualizar.php?file=../../../../etc/passwd”, si tenemos suerte, obtendríamos los usuarios y contraseñas de dicho servidor que sirve la aplicación web [13].

WebTrojans

Esta técnica también se basa en la posibilidad de poder subir un fichero a un servidor mediante una aplicación web, la vulnerabilidad aparece cuando el fichero que se sube no pasa por ningún filtro, ni se verifica, ni se comprueba, directamente se queda almacenado en nuestro servidor.

Por ejemplo, imaginemos que tenemos una aplicación web similar a Instagram, en la cual podemos subir fotos y videos, sin embargo, estos ficheros no pasan por ningún control. Ahora nos creamos una Shell remota y la camuflamos dentro de un fichero “.jpg” y subimos dicha imagen.

Una Shell remota suele ser una especie de script que carga un CMD o un PROMPT del sistema y lo devuelve a la maquina del atacante, para que este pueda ejecutar comandos de manera remota que se ejecutaran en la máquina de la víctima [14].

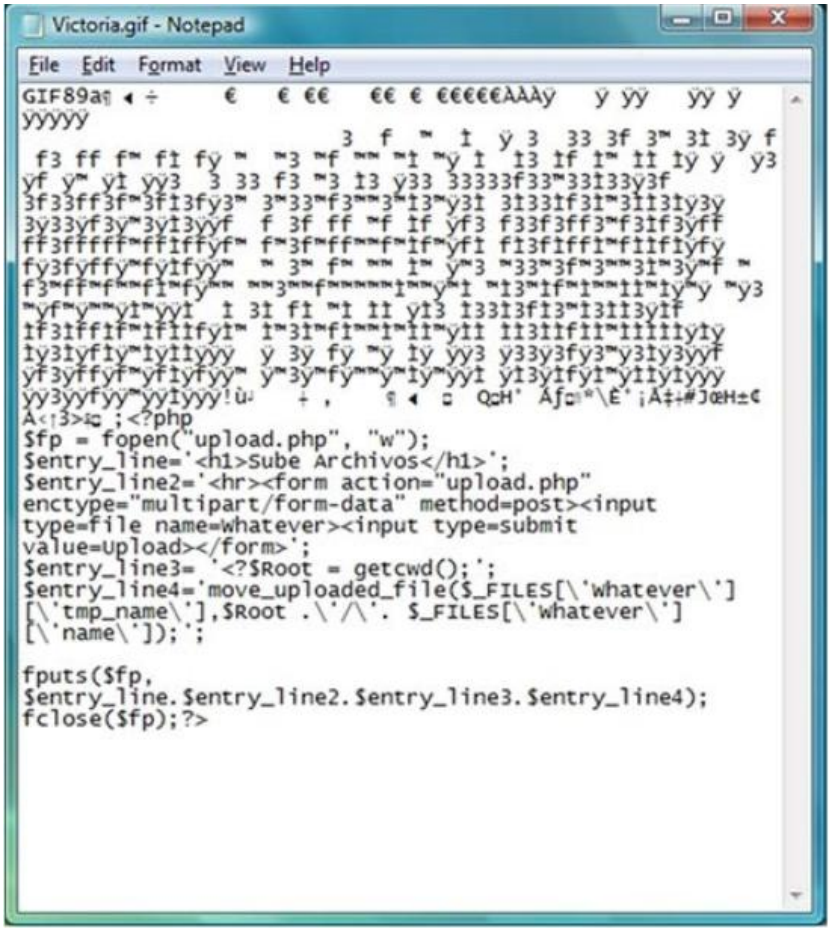


Figura 5. WebTrojan Php oculto dentro de un .gif

Ataques Defacement

Este ataque se basa en la vulnerabilidad de poder inyectar código en una página o aplicación web, con el objetivo de cambiar la apariencia de la propia aplicación web. Es una práctica común por los hackers cuando consiguen romper la seguridad de una aplicación web, es una especie de "grafiti electrónico" o firma. También en los últimos años suelen dejar imágenes con mensajes reivindicativos o protestantes, muy típico de los Hacktivist [15].

2.3.2 Centradas en el lado del servidor

Blind XPath Injection

XPath es un tipo de lenguaje de consultas, similar a SQL, se utiliza para localizar elementos concretos dentro de un árbol de XML. Este ataque se basa en la inyección de código con sintaxis de XPath para poder modificar, acceder, o incluso borrar información del XML almacenado en el servidor.

Por ejemplo, si la aplicación web que estamos auditando utiliza XML para almacenar nombres de usuario y contraseña (credenciales) con las cuales se puede acceder a la propia aplicación como usuario o incluso como administrador, un ataque de este tipo podría revelar dicha información contenida en ese XML[16].

SQL Injection – SQLi

Este ataque se basa en la vulnerabilidad de poder inyectar código con sintaxis SQL dentro en una aplicación web, el cual será ejecutado por el motor de SQL, con el objetivo de poder acceder, modificar o borrar información sensible de la base de datos que utiliza dicha aplicación.

Al igual que con el XSS, este tipo de ataques se pueden prevenir controlando las comunicaciones entre el usuario y la interfaz de la aplicación web, es decir, controlando los formularios o las peticiones que envía el usuario al servidor. Por otro lado, en la parte del servidor, se recomienda utilizar sentencias SQL preparadas, en las cuales solo aceptamos un parámetro predefinido y que será el que provenga de la interfaz del usuario.

Por ejemplo, si estamos en la ventana de login de una aplicación web en la cual podemos inyectar código SQL, podríamos saltarnos el login y acceder directamente a la página de inicio [17].

Blind SQLi

Este ataque se aprovecha de la misma vulnerabilidad de poder inyectar código con sintaxis SQL en una aplicación; Se puede ver como una variante del SQL Injection, con la diferencia de que en este caso, no podemos ver los resultados directamente, sino que tenemos que interpretar los resultados de nuestras “inyecciones” o sentencias, en función de cómo se comporta la propia aplicación web.

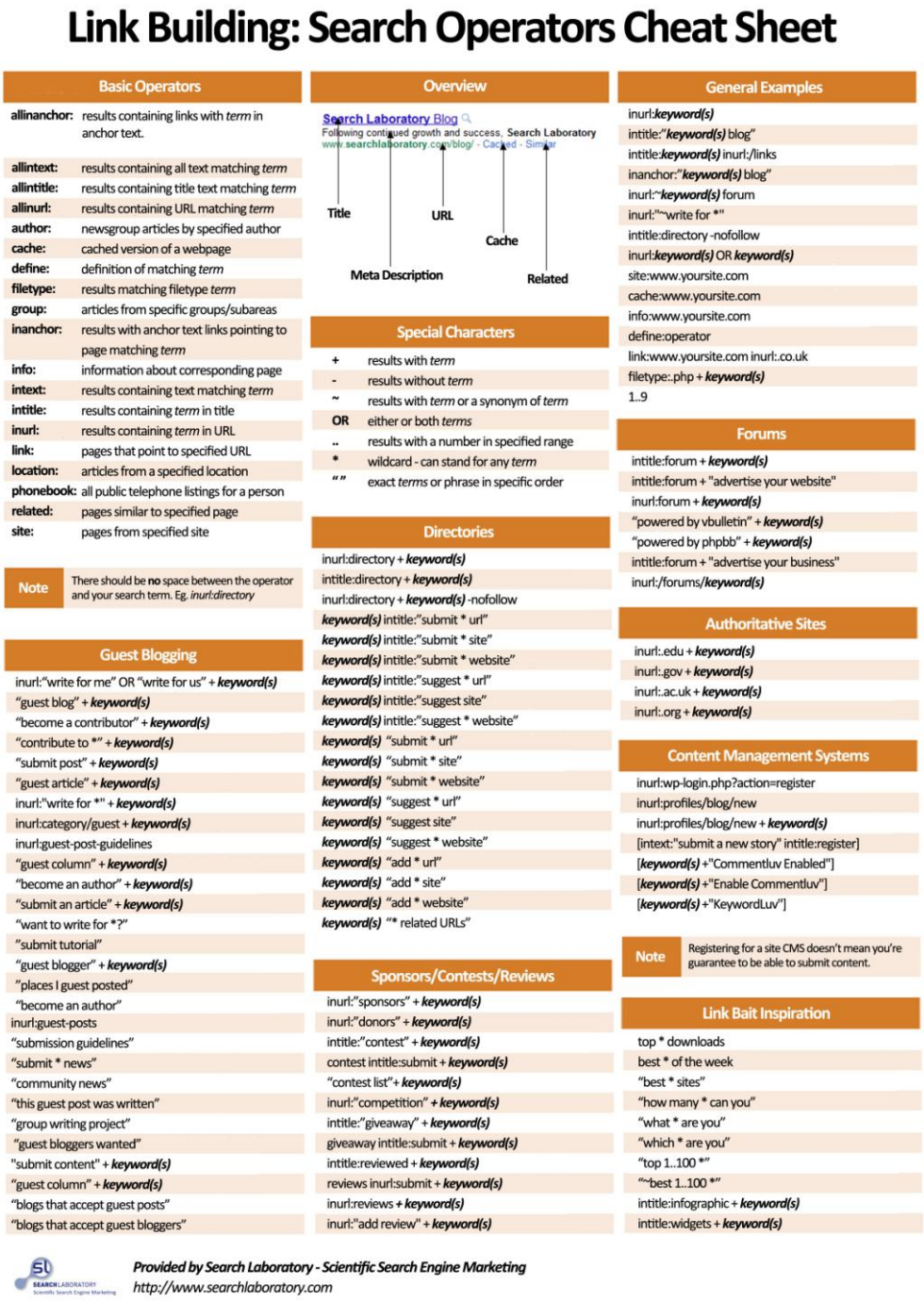
Normalmente, se utilizan cuando el propietario de la aplicación web, ha configurado y personalizado los mensajes de error de la aplicación, sin embargo, no ha codificado bien las sentencias SQL de la misma para evitar la propia inyección de código SQL no legitimo o no autorizado [18].

2.3.3 Otros vectores de ataque

Google Hacking

Como sabemos, Google indexa millones de páginas, documentos y aplicaciones web entre otras cosas, dicho motor de búsqueda no entiende que pagina o que documento puede ser privado o de acceso restringido. Si bien le podemos indicar que paginas de nuestra aplicación no indexe (mediante el fichero robot.txt [19]) hay muchas otras que si indexara y que pueden contener información sensible de nuestro dominio. Saber cómo

funciona la sintaxis de este buscador en concreto y de los buscadores en general, nos puede servir para saber qué información indexa el buscador sobre nosotros, de la cual no somos consientes [20].



Link Building: Search Operators Cheat Sheet

Basic Operators

allinanchor:	results containing links with *term* in anchor text.
allintext:	results containing all text matching *term*
allintitle:	results containing title text matching *term*
allinurl:	results containing URL matching *term*
author:	newsgroup articles by specified author
cache:	cached version of a webpage
define:	definition of matching *term*
filetype:	results matching filetype *term*
group:	articles from specific groups/subareas
inanchor:	results with anchor text links pointing to page matching *term*
info:	information about corresponding page
intext:	results containing text matching *term*
intitle:	results containing *term* in title
inurl:	results containing *term* in URL
link:	pages that point to specified URL
location:	articles from a specified location
phonebook:	all public telephone listings for a person
related:	pages similar to specified page
site:	pages from specified site

Note There should be **no** space between the operator and your search term. Eg. *inurl:directory*

Guest Blogging

inurl:"write for me" OR "write for us" + ***keyword(s)***
"guest blog" + ***keyword(s)***
"become a contributor" + ***keyword(s)***
"contribute to *" + ***keyword(s)***
"submit post" + ***keyword(s)***
"guest article" + ***keyword(s)***
inurl:"write for *" + ***keyword(s)***
inurl:category/guest + ***keyword(s)***
inurl:guest-post-guidelines
"guest column" + ***keyword(s)***
"become an author" + ***keyword(s)***
"submit an article" + ***keyword(s)***
"want to write for *?"
"submit tutorial"
"guest blogger" + ***keyword(s)***
"places I guest posted"
"become an author"
inurl:guest-posts
"submission guidelines"
"submit * news"
"community news"
"this guest post was written"
"group writing project"
"guest bloggers wanted"
"submit content" + ***keyword(s)***
"guest column" + ***keyword(s)***
"blogs that accept guest posts"
"blogs that accept guest bloggers"

Overview

Search Laboratory Blog
Following continued growth and success, Search Laboratory
www.searchlaboratory.com/blog/ - Cached - Similar

Title
URL
Cache
Meta Description
Related

Special Characters

+	results with *term*
-	results without *term*
~	results with *term* or a synonym of *term*
OR	either or both *terms*
..	results with a number in specified range
*	wildcard - can stand for any *term*
" "	exact *terms* or phrase in specific order

Directories

inurl:directory + ***keyword(s)***
intitle:directory + ***keyword(s)***
inurl:directory + ***keyword(s)*** -nofollow
keyword(s) intitle:"submit * url"
keyword(s) intitle:"submit * site"
keyword(s) intitle:"submit * website"
keyword(s) intitle:"suggest * url"
keyword(s) intitle:"suggest site"
keyword(s) intitle:"suggest * website"
keyword(s) "submit * url"
keyword(s) "submit * site"
keyword(s) "submit * website"
keyword(s) "suggest * url"
keyword(s) "suggest site"
keyword(s) "suggest * website"
keyword(s) "add * url"
keyword(s) "add * site"
keyword(s) "add * website"
keyword(s) "* related URLs"

Sponsors/Contests/Reviews

inurl:"sponsors" + ***keyword(s)***
inurl:"donors" + ***keyword(s)***
intitle:"contest" + ***keyword(s)***
contest intitle:submit + ***keyword(s)***
"contest list" + ***keyword(s)***
inurl:"competition" + ***keyword(s)***
intitle:"giveaway" + ***keyword(s)***
giveaway intitle:submit + ***keyword(s)***
intitle:reviewed + ***keyword(s)***
reviews inurl:submit + ***keyword(s)***
inurl:reviews + ***keyword(s)***
inurl:"add review" + ***keyword(s)***

General Examples

inurl:***keyword(s)***
intitle:"***keyword(s)*** blog"
intitle:***keyword(s)*** inurl:/links
inanchor:"***keyword(s)*** blog"
inurl:~***keyword(s)*** forum
inurl:"~write for *"
intitle:directory -nofollow
inurl:***keyword(s)*** OR ***keyword(s)***
site:www.yoursite.com
cache:www.yoursite.com
info:www.yoursite.com
define:operator
link:www.yoursite.com inurl:.co.uk
filetype:.php + ***keyword(s)***
1..9

Forums

intitle:forum + ***keyword(s)***
intitle:forum + "advertise your website"
inurl:forum + ***keyword(s)***
"powered by vbulletin" + ***keyword(s)***
"powered by phpbb" + ***keyword(s)***
intitle:forum + "advertise your business"
inurl:/forums/***keyword(s)***

Authoritative Sites

inurl:.edu + ***keyword(s)***
inurl:.gov + ***keyword(s)***
inurl:.ac.uk + ***keyword(s)***
inurl:.org + ***keyword(s)***

Content Management Systems

inurl:wp-login.php?action=register
inurl:profiles/blog/new
inurl:profiles/blog/new + ***keyword(s)***
[intext:"submit a new story" intitle:register]
[***keyword(s)*** +"Commentluv Enabled"]
[***keyword(s)*** +"Enable Commentluv"]
[***keyword(s)*** +"KeywordLuv"]

Note Registering for a site CMS doesn't mean you're guarantee to be able to submit content.

Link Bait Inspiration

top * downloads
best * of the week
"best * sites"
"how many * can you"
"what * are you"
"which * are you"
"top 1..100 *"
"~best 1..100 *"
intitle:infographic + ***keyword(s)***
intitle:widgets + ***keyword(s)***

Provided by Search Laboratory - Scientific Search Engine Marketing
http://www.searchlaboratory.com

Figura 6. Resumen comandos búsqueda avanzada en Google

Por ejemplo, podríamos buscar "inurl:/password filetype: txt" y nos aparecerían una lista de paginas o sitios en las que tienen "password" en alguna parte del enlace y además tienen un fichero txt.

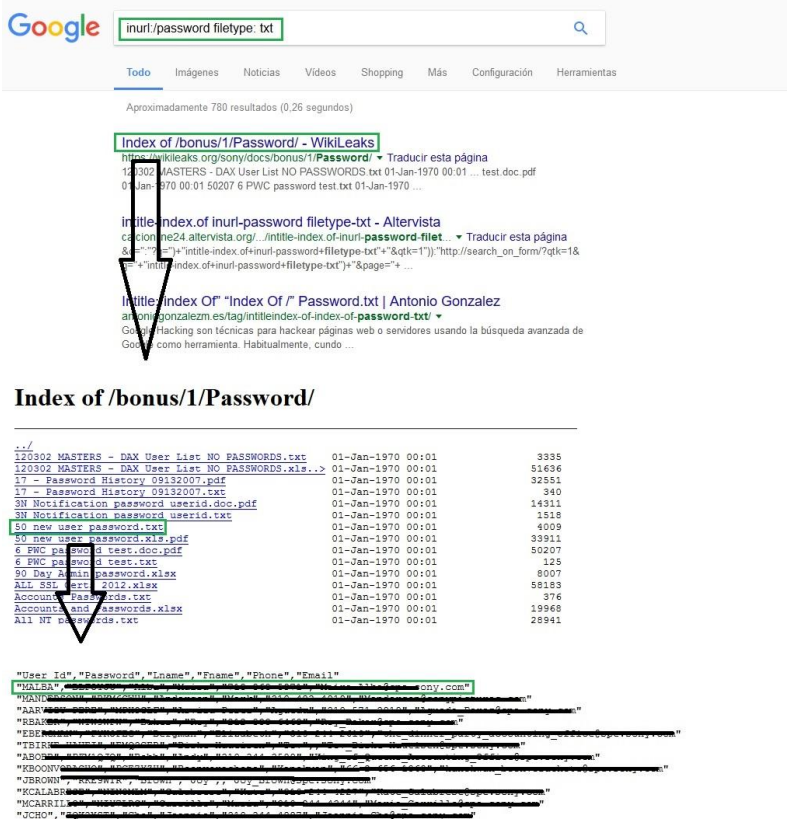


Figura 7. Ejemplo sencillo Google hacking

Man In The Browser

Este ataque es una aproximación a un Man in The Middle, en este caso el atacante consigue interponerse entre la comunicación del navegador y el propio servidor web, usualmente mediante un troyano, con la finalidad de poder interceptar "al vuelo" todos los mensajes entre el cliente y el servidor, además de poder manipularlos a su voluntad [21].

HSTS & HPKP

Más que un ataque, estos términos se corresponden con dos mecanismos que son adicionales al protocolo HTTP con el fin de hacer más robusta y segura su implementación.

HTTP Stric Transport Security (HSTS) es un mecanismo de seguridad que se implementa en la parte del servidor y que obliga a los clientes (navegadores) a interactuar con él, únicamente mediante el protocolo HTTPS.

Por defecto, cuando instalamos un navegador (Firefox, Chrome, Opera, etc.) estos vienen con una lista previamente cargada de sitios a los que solo accederá mediante HTTPS, como por ejemplo, Google, Paypal, Amazon, etc. También hay listas en las que puedes enviar y hacer una petición para que tu aplicación web se añada a una de estas listas previas.

Este mecanismo tiene una vulnerabilidad que aparece en la primera petición, hacia una aplicación web o servidor el cual no está en estas listas previamente cargadas, la primera petición se realizaría mediante HTTP, entonces el servidor le responde y le obliga a continuar con la comunicación mediante HTTPS. En esta primera petición, un atacante podría interceptar el mensaje, enviarlo, y al recibir la respuesta, eliminar la cabecera que "obliga" a que la comunicación se mantenga en HTTPS sin que el usuario final se diera cuenta.

HTTP Public Key Pinning (HPKP) es un mecanismo de seguridad que evita que un posible atacante utilice certificados fraudulentos, mediante el uso de un pin-sha256 esperado por parte del cliente y proveniente de la parte del servidor, permitiendo detectar modificaciones en la cadena del certificado [22].

2.4 AUDITORÍA DE SEGURIDAD INFORMÁTICA

Una auditoría de seguridad informática es un estudio o análisis que realiza un profesional de un área determinada, en este caso de informática, para identificar y enumerar las posibles vulnerabilidades que puedan existir en equipos, sistemas, redes, etc. Con la finalidad de descubrir dichas vulnerabilidades para posteriormente corregir y/o tomar medidas preventivas que refuercen la seguridad del objeto auditado [23].

Hay diferentes metodologías para llevarlas a cabo, un esquema de las fases podría ser el siguiente:

- Enumerar las redes, topologías y protocolos.
- Verificar que se cumplen los estándares internacionales como ISO 27001, ISO 27002, COBIT, etc.
- Identificar los sistemas operativos instalados.
- Analizar los servicios y aplicaciones.
- Identificar y comprobar posibles vulnerabilidades.
- Especificar posibles soluciones o correcciones a las vulnerabilidades.
- Recomendar posibles medidas preventivas.
- Conclusiones o estado/informe final del objeto en cuestión.

Hay diferentes tipos de auditoría, algunas se pueden catalogar como:

- **Auditoría de seguridad interna**: Se centra en la seguridad de las redes locales (LAN) e Intranets o redes corporativas internas.
- **Auditoría de seguridad perimetral**: Se centra en la seguridad del perímetro que rodea la red local o corporativa para estudiar el grado de seguridad que ofrece en las entradas y salidas del perímetro.
- **Test de intrusión o Pentesting**: Se centra en intentar acceder a los sistemas para así estudiar el grado de resistencia a la intrusión no deseada.
- **Auditoría de aplicaciones web**: Se centra en identificar posibles vulnerabilidades que existan en una aplicación web.
- **Auditoría de código de aplicaciones**: Se centra en el análisis estático del código de cualquier aplicación independientemente del lenguaje empleado.
- **Auditoría Técnica de Caja Blanca**: El auditor obtiene el rol de un usuario interno del sistema, con accesos totales o parciales o a parte de los datos críticos. Tiene visión total del sistema.
- **Auditoría Técnica de Caja Negra**: El auditor no conoce ninguna característica del sistema. Su visión global del sistema es nula.

- **Auditoría Técnica de Caja Gris**: El auditor obtiene el rol de un usuario interno del sistema, pero este posee pocos o ningún privilegio sobre dicho sistema. Su visión del sistema es a medias.

2.5 PENTESTING

2.5.1 ¿Qué es una prueba de Pentesting?

Antes introduje un poco el concepto, intentare detallarlo un poco más. Pentesting (Penetration Testing) o Prueba de Intrusión, es el proceso de atacar diversos entornos o sistemas con el objetivo de descubrir posibles fallos, debilidades o vulnerabilidades que podrían ser usados por un atacante para poner en riesgo o irrumpir la seguridad de uno o varios sistemas. Asimismo, una vez detectados dichos fallos de seguridad, poder solucionarlos [24].

2.5.2 ¿Es legal?

Sí, siempre y cuando tengamos el consentimiento del dueño legitimo del sistema o entorno al que estemos dirigiendo nuestros ataques. Normalmente, se suele firmar un contrato con la empresa o responsable que quiere contratar a un experto en seguridad informática, y que desea saber hasta dónde podría llegar un posible atacante. De hecho lo que nos diferencia de un "hacker con malas intenciones" es dicho consentimiento, además podemos tener "carta blanca" para acceder hasta donde te permitan los propios ataques, etc.

Existen ciertas certificaciones que son valoradas por las empresas como algunas ISO que nombré anteriormente, o como la CEH (Certified Ethical Hacking) [25] que nos pueden servir tanto para aprender más acerca de la seguridad en informática, como para acceder a posibles puestos de trabajo.

2.5.3 Fases de una prueba de Pentesting

Hay diversos esquemas para realizar una prueba de Pentesting, aquí listare las fases que más o menos son las más habituales:

- **Contacto**: En esta fase acordamos con el cliente en que va consistir la prueba de intrusión, objetivos, entornos, fechas en las que se va a realizar, horas del día, con quien debemos hablar si encontramos alguna vulnerabilidad o fallo, si dichas vulnerabilidades se pueden explotar o solamente las identificamos, todo esto por escrito, para que autorice el Pentesting y además límite nuestra responsabilidad en caso de que surjan problemas, todo lo relacionado con pagos, etc. Muy similar a un contrato al fin y al cabo.
- **Fase de recolección de información**: Esta fase se centra en obtener toda la información posible de la empresa, los sistemas y servicios que ofrecen, los sistemas operativos, servicios y aplicaciones que sirven los servidores y demás equipos. La actividad de los empleados, redes sociales, ingeniería social, todo lo que nos pueda servir más adelante para realizar un ataque.
- **Fase de modelado de amenaza**: Con la información recolectada en la fase anterior, nos tenemos que poner en la piel de un atacante y pensar cuál va ser nuestra estrategia de ataque. Plantearnos objetivos más concretos y cómo podríamos asaltarlos o interrumpirlos o acceder a ellos. Esta estrategia inicial puede variar a lo largo del Pentesting, ya sea porque nos encontramos en un callejón sin salida, o porque no valoramos todas las alternativas posibles, sin embargo, siempre debemos tener un planteamiento o aproximación inicial.
- **Fase de análisis de vulnerabilidades**: En esta fase identificamos las posibles vulnerabilidades que hayamos podido encontrar, de acuerdo al planteamiento y la información anterior, aquí dependiendo de los puntos flacos que obtengamos de los sistemas, puede ser el momento apropiado para seleccionar que herramientas vamos a utilizar para acercarnos a nuestros objetivos iniciales.
- **Fase de explotación**: En esta fase intentamos conseguir acceder a los entornos o sistemas objetivos, podemos utilizar herramientas que exploten las vulnerabilidades previamente encontradas, o podemos utilizar credenciales u otro tipo de información referente a empleados o cualquier usuario con privilegios, e intentar realizar ataques de fuerza bruta o cualquier otro vector de ataque para intentar acceder al sistema.
- **Fase de post-explotación**: Una vez hayamos ganado acceder a los sistemas objetivos, tenemos que demostrar que podría suponer estas brechas de seguridad para el cliente. Podemos intentar subir nuestro nivel de privilegios, obtener información sensible, saltar a otros sistemas vecinos, interrumpir los servicios. Todo depende de lo que hayamos acordado al principio.

- **Fase de informe**: Aquí elaboramos y presentamos el informe de la prueba al cliente, de forma que entienda los riesgos que pueden suponer las vulnerabilidades encontradas, y como darles solución. Además de posibles mejoras que se nos puedan ocurrir o que hayamos visto que hubieran sido necesarias o nos hubieran ofrecido resistencia durante la prueba [26].

2.6 ALTERNATIVAS

2.6.1 OSSTMM

Open Source Security Testing Methodology Manual (OSSTMM) es un manual libre bajo la licencia de Creative Commons, creado por la organización ISECOM Project con el objetivo de poner a disposición del público una metodología o proceso con el cual comprobar la seguridad que tienes implantada en tu organización [27].

2.6.2 OWASP

Open Web Application Security Project (OWASP) es un proyecto de código abierto creado con el objetivo de mejorar la seguridad del software. La fundación de OWASP es un organismo sin ánimo de lucro que organiza y lleva a cabo los proyectos de OWASP. Cuenta con una amplia comunidad, tanto de empresas como de particulares alrededor de todo el mundo. Todo el material que generan, artículos, metodologías, documentación, herramientas y tecnologías, se crean bajo licencias de código abierto y libre (La mayoría bajo Creative Commons) [28].

Proyectos Destacados

- **OWASP Top Ten Project**: Este proyecto se centra en sacar cada cierto tiempo, las diez vulnerabilidades más comunes en las aplicaciones web y que son a su vez bastante criticas. Suelen generar un ".pdf" con estas vulnerabilidades en diferentes idiomas, en el cual se detallan como identificarlas y solucionarlas.
- **OWASP Code Review Project**: Este proyecto se centra en cierto modo en el análisis de código estático de una aplicación, durante la fase de desarrollo o en un sistema ya implantado, generan un libro bastante técnico para administradores, desarrolladores y profesionales de seguridad.

- **OWASP Zed Attack Proxy Project**: Este proyecto se centra en mantener una herramienta que desarrollaron, llamada ZAP. Esta herramienta (proxy web) es bastante popular mundialmente, es gratuita y tiene una amplia comunidad de colaboradores por todo el mundo. Entraremos más en detalle sobre el funcionamiento de esta en el capítulo 3.
- **OWASP OWTF**: Este proyecto también se centra en otra herramienta, Ofensive Web Testing Framework (OWRF), es un framework que ayuda a realizar las tareas de Pentesting eficientemente y siguiendo las guías y estándares de OWASP.
- **OWASP Web Testing Enviroment Project**: Es una colección de aplicaciones web y documentación sobre las mismas. Vienen en distintos formatos como VM's, Distribuciones de Linux e imágenes ISO. La idea de estas aplicaciones es que son vulnerables, se suelen utilizar como aplicaciones web en las cuales poder entrenar o practicar algunas de las técnicas y vulnerabilidades que describí anteriormente, además de muchas otras.
- **OWASP CSRFGuard Project**: Es un proyecto el cual está dirigido a crear una serie de librerías que ayudan a prevenir los ataques de tipo CSRF (Cross Site Request Forgery), estas librerías implementan una variante de un patrón de diseño que se encarga de generar tokens de sesión sincronizados entre el cliente y servidor [29].

Otros proyectos de desarrollo

- **WebScarab**: Una aplicación Web que incluye una suite de evaluación de vulnerabilidades y herramientas proxy.
- **WebGoat**: Es una herramienta de evaluación interactiva en la que los usuarios pueden aprender sobre seguridad de aplicaciones web en un sitio seguro y legal.
- **DotNet**: Una suite con herramientas para asegurar entornos .NET [30].

2.7 DESARROLLO SEGURO DE APLICACIONES WEB

Hablamos de software seguro cuando se diseña, se construye y se prueba para que no falle, y siga funcionando, incluso cuando este bajo la presión de un ataque.

Partes que se deben revisar (Según metodología OWASP):

Autenticación: Comprobar que todas las peticiones pasan por un formulario de autentificación, y que este no se puede saltar. Hay que tener especial cuidado con las URL de acceso directo, comprobando siempre si hay un usuario autentificado, en caso contrario, echar para atrás la petición.

Comprobar que todas las páginas cumplen el requisito de autentificación, cuando sea necesario. Las credenciales que se pasen de autentificación (o cualquier información sensible), debe enviarse siempre vía HTTP POST y nunca con GET.

Las páginas en las que no sea necesario un mecanismo de autentificación, deben ser revisadas para comprobar que desde dichas páginas no puedan acceder a recursos protegidos.

Si es posible cifrar las credenciales de autentificación para que no se transmitan en claro. Comprobar que no hay "puertas traseras" en el código de producción. (Esto lo debería comprobar más un programador o desarrollador.)

Autorización: Comprobar o implementar mecanismos de autorización, tales como, control de acceso y gestión de roles o privilegios. Además comprobar que los roles o privilegios definidos están bien determinados por las acciones que pueden realizar y cuáles no, a que recursos pueden acceder, etc.

Comprobar que asignamos los mínimos privilegios necesarios. También debemos comprobar que estos mecanismos funcionan bien y que no pueden saltarse.

Gestión de Cookies: Asegurarnos de no comprometer información sensible, además de que no se puedan realizar operaciones no autorizadas manipulando cookies. Normalmente solo utilizarlas para almacenar el identificador de sesión, si fuera necesario.

Comprobar que se utiliza cifrado, en el caso de usar las cookies para almacenar información y datos sensibles, utilizar algún algoritmo público fuerte como AES. Verificar el uso seguro de cookies en todas las transiciones de estados de código de la aplicación.

Los datos de la sesión deben ser validados, y que estos junto con las cookies deben contener la mínima información privada posible. También es aconsejable definir todas las cookies que usa la aplicación, sus nombres y para qué son necesarias.

Validación de Entrada de Datos: Esta parte es muy importante, ya que nos puede evitar una gran cantidad de posibles ataques por inyección de código. Comprobar que tenemos mecanismos para validar los datos.

Muy importante validar todas las entradas que puede modificar un usuario normal o uno con malas intenciones, cabeceras HTTP, cuatros de texto, cuadros de entrada, campos ocultos, desplegables, etc.

Comprobar la longitud de todas las entradas, si es posible, limitarlas a un máximo número de caracteres. Antes de enviar cualquier formulario, validar todos los campos que contiene, además de las cookies y cabeceras.

Asegurarnos de darle formato a los datos de manera eficiente y conveniente, que solo contenga los tipos de caracteres con los que necesitemos trabajar. Es muy importante realizar las validaciones tanto en el lado del cliente (p.e. en Javascript) como en el lado del servidor (PHP, Python, etc.).

Por último, una regla de oro, cualquier entrada externa, sea cual sea, será examinada, verificada y validada.

Gestión de Errores / Fuga de información: Comprobar que todas las funciones y métodos de nuestra aplicación, devuelven un valor o código, y que este se comprueba y tiene un control de errores asociados.

Comprobar que se capturan todos los errores y excepciones, y que son gestionados. Además los errores del sistema tiene que ser invisibles para el usuario (modo debug, o solo para los desarrolladores).

Muy importante crear paginas de error personalizadas, gestionadas por los desarrolladores, para que no muestre información sensible, que podría ser utilizada por un atacante, por ejemplo, que no enseñen la versión de tu servidor de apache, o la versión de SQL que utiliza la aplicación, etc.

Log / Auditoría: Comprobar que no se registra información sensible en el log en caso de error (credenciales de autentificación, cookies, etc.). Comprobar que definimos y controlamos la longitud máxima de una entrada al log.

Determinar si la aplicación auditara las operaciones realizadas por el cliente, especialmente en el caso de manipulación de datos: Create, Update, Delete (CUD).

Asegurarnos también de registrar en el log los casos fallidos o exitosos de autentificación y los errores de la aplicación.

Cifrado de Datos: Todos los datos o información sensible transmitidos, interna o externamente, deben ir cifrados. Normalmente los propios lenguajes con los que se

desarrolla implementan funcionalidad que permite cifrar y descifrar información, es recomendable utilizar algoritmos públicos fuertes, y no MD5 o SHA-1.

Entorno de Código Seguro: Comprobar la estructura de ficheros de nuestra aplicación, evitar que hayan ficheros accesibles por el usuario que no deberían serlo. (manifest.xml, robot.txt, htdocs, etc.).

Buscar código comentado (aunque sea para pruebas/debug/desarrollo) que pueda contener información y datos sensibles. Tener cuidado con las llamadas a sistemas operativos (p.e. aperturas o escrituras de ficheros), y comprobar los posibles errores.

Gestión de Sesiones: Comprobar cómo y cuándo se crean las sesiones de usuario, ya sean autentificadas o no. Los ID de sesión deben estar cifrados fuertemente.

Si la aplicación hace un seguimiento de la sesión, comprobar que este es invisible al usuario. Comprobar que hace la aplicación cuando se encuentra con un ID de sesión invalido.

Comprobar la invalidación de sesiones, la gestión de sesiones multi-threaded/multi-usuario, determinar un timeout de inactividad de la sesión, comprobar los métodos login/logout, etc. [31].

Otras partes opcionales que pueden ser interesantes:

Patrones de diseño: Los patrones de diseño ofrecen soluciones a problemas de diseño comunes en el desarrollo de software. Para que estas soluciones sean aceptadas (normalmente por la comunidad) como patrones de diseño, deben cumplir ciertas características, como que se debe comprobar su efectividad y además tienen que ser reutilizables, entre otras cosas [32].

Por ejemplo, el Singleton, es un patrón de diseño que se suele utilizar para la gestión del pool de conexiones de la base de datos de una aplicación web, ya que este se encarga de que cada hilo/cliente solo pueda instanciar una clase de la conexión y usarla globalmente. O el Modelo-Vista-Controlador MVC (Model View Controller) que se utiliza para dividir el desarrollo de una aplicación y enfocarlo a tres diferentes roles totalmente modulares e independientes, que serian el equipo gestor de la base de datos, el equipo de diseño grafico y el equipo de desarrollo.

URL o **URI personalizadas**: Hay diferentes maneras de trabajar con las rutas de nuestra aplicación, una de ellas es utilizar URIs personalizadas, para identificar y acceder a ciertos recursos (sin descuidar la seguridad de los recursos a los que se accede.) y poder filtrarlos o seleccionarlos de manera una forma amigable al usuario.

Por ejemplo, imaginar que tenemos la siguiente ruta "www.miaplicacion.es/clientes" en la cual, un usuario (autentificado y autorizado) puede ver los clientes o compañeros de su departamento, del mismo modo ahora quiere filtrarlos, por un identificador único, podría hacer "www.miaplicacion.es/cliente/3", en cuyo caso accedería a la información que haya compartido el cliente tres [33].

Configurar robot.txt: En este tipo de ficheros se suelen escribir, las rutas de directorios y ficheros de nuestra aplicación web, a los cuales no queremos que entre "el crawler o araña" que busca, analiza e indexa información en los buscadores, como por ejemplo en google, con el fin de no revelar información sensible en los buscadores sobre nuestra aplicación o sitio web [34].

CAPÍTULO 3. HERRAMIENTAS PARA PENTESTING WEB

En este capítulo hablare de algunas de las diferentes herramientas que existen para ayudarnos con la labor de Pentesting o Auditor de una página o aplicación web, las he separado en dos categorías, las que nos sirven para escanear vulnerabilidades en aplicaciones web y las que nos pueden servir como proxy web e interceptar e interpretar todas las peticiones/respuestas entre un cliente y servidor web.

3.1 ESCANER DE VULNERABILIDADES WEB

Como su propio nombre indica, estas aplicaciones nos sirven para escanear las aplicaciones web y así poder identificar y clasificar las diferentes vulnerabilidades que pueda haber en estas.

3.1.1 W3AF

Web Application Attack and Audit Framework (W3AF) es una aplicación, resultado de un proyecto con el objetivo de crear un framework para ayudar a la seguridad de las aplicaciones web, encontrando y explotando las vulnerabilidades presentes en dicha aplicación web. Esta aplicación está escrita en Python y con licencia GPLv2.0, código abierto y de libre uso [35].

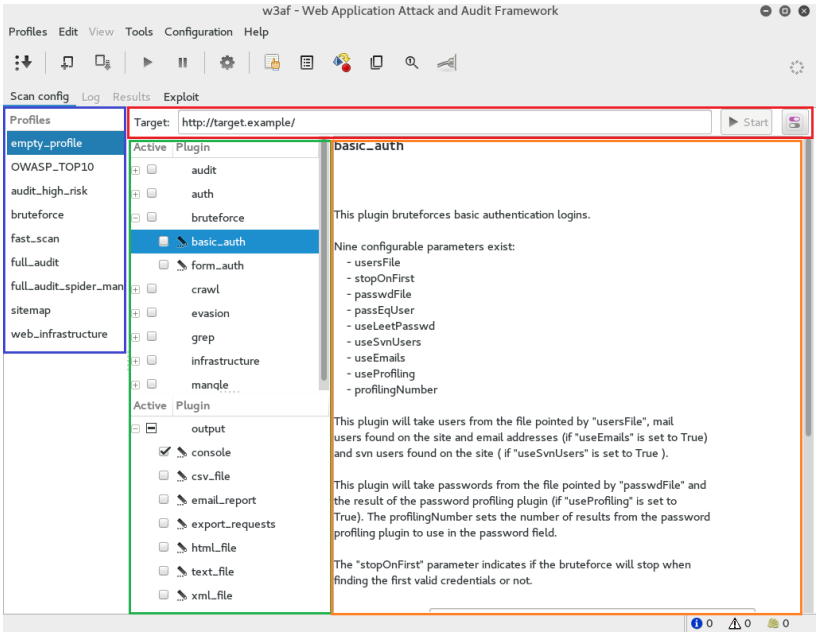


Figura 8. Pantalla inicial W3AF.

Antes de entrar, nos advierte que el uso de esta herramienta sobre una aplicación web o dominio, sin el consentimiento del propietario de la aplicación web, es ilegal, por lo tanto hay que usarlo con mucho cuidado.

La interfaz es bastante sencilla e intuitiva, voy a describir los recuadros pintados sobre la Figura 8. En la lista de la izquierda (recuadro azul) tenemos una lista de perfiles pre-configurados, los cuales pueden ser modificados o incluso podemos añadir nuevos perfiles con los "ataques o tests" que se van a realizar sobre nuestro objetivo. Entre estos perfiles podemos destacar el de "OWASP TOP 10" el cual está configurado para comprobar las 10 vulnerabilidades más comunes según la fundación OWASP.

En el recuadro rojo tenemos el "target" que es donde pondremos la dirección de nuestra aplicación web objetivo sobre la que realizaremos las diferentes comprobaciones y/o ataques, en el recuadro verde tenemos los diferentes "Plug-in", estos nos sirven para configurar y definir exactamente qué ataques o comprobaciones queremos realizar, además como comente anteriormente, esta configuración se puede almacenar en un "perfil" para poder replicar la prueba las veces que nos haga falta. También podemos decirle si queremos la salida de los resultados por consola o en un fichero de texto, HTML, XML, entre otros.

Por último, el recuadro naranja aparece cuando pinchamos sobre cualquiera de los Plug-in, a modo de descripción, nos explican para que o como funciona cada uno de ellos.

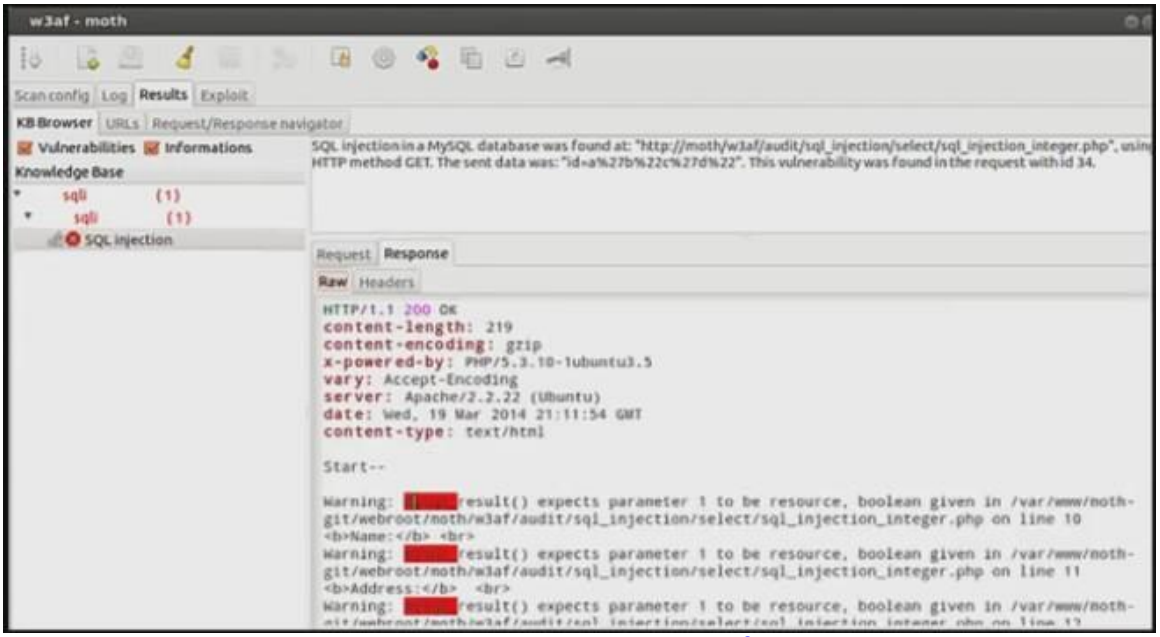


Figura 9. Ejemplo de SQLi encontrada [f9]

Si encuentra alguna vulnerabilidad, te da una lista de ellas, además te da alguna información sobre la misma, si además posee el Plug-in para explotarla, te da la opción de explotar dicha vulnerabilidad, pudiendo obtener una Shell remota, entre otras cosas, para poder lanzar comandos en el servidor explotado.

3.1.2 Acunetix

Acunetix es un proyecto que cuenta con dos aplicaciones para el escáner de vulnerabilidades, poseen una versión de escritorio (tienen una versión de prueba de 14 días) y otra versión en la nube, ambas te permiten escanear y buscar vulnerabilidades tanto en aplicaciones web como en redes objetivo.

A parte de la licencia de prueba que te dan, el resto es de pago, tienes la opción de pagar por una licencia permanente en tres versiones (Estándar, Pro, Enterprise) para la aplicación de escritorio, o puedes pagar por el servicio de "Scan" desde la nube, en este segundo caso, pagas por el numero de objetivos a los que quieras escanear y/o redes [36].

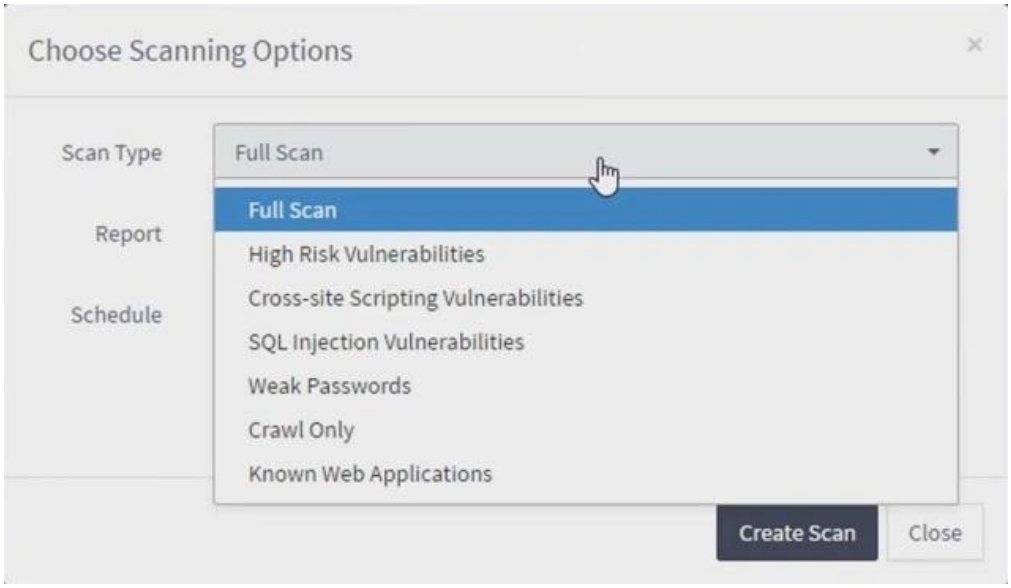


Figura 10. Tipos de escaneos que permite Acunetix.

La aplicación cuenta con una interfaz web, desde la que podemos empezar a configurar todo, lo primero es introducir la dirección de nuestro objetivo. También nos da la opción de guardar patrones de user y password para los posibles formularios de login que se encuentre durante el escaneo.

Lo siguiente es seleccionar el objetivo que declaramos previamente y seleccionar el tipo de escáner que vamos a realizar (Aquí las opciones son más limitadas, comparadas con la anterior herramienta W3AF). Asimismo podemos seleccionar las opciones de "output", las cuales te generan un archivo PDF con la estructura que tu elijas de entre las pre-establecidas, igualmente te permite elegir si deseas realizar el escaneo en este preciso momento o más adelante.

Scan of http://testphp.vulnweb.com

Scan details

Scan information	
Start time	17/11/2016, 19:22:12
Start url	http://testphp.vulnweb.com
Host	http://testphp.vulnweb.com
Scan time	14 minutes, 35 seconds
Profile	Full Scan

Threat level

Acunetix Threat Level 3

Alerts distribution

Total alerts found	151
High	75
Medium	48
Low	9
Informational	19

Executive summary

Alert group	Severity	Alert count
Blind SQL Injection	High	24
SQL injection	High	24
Cross site scripting	High	19
Directory traversal	High	1
Macromedia Dreamweaver remote database scripts	High	1
nginx SPDY heap buffer overflow	High	1
PHP allow_url_fopen enabled (AcuSensor)	High	1
Remote file inclusion XSS	High	1
Script source code disclosure	High	1

Figura 11. Reporte ejecutivo de Acunetix

Por lo tanto, aparte de generarte diferentes reportes en ficheros PDF, te permite inspeccionar detalladamente cada una de las vulnerabilidades que encuentra, el fichero en el que la encontró, el mensaje que se genero en la aplicación web, las cabeceras de petición/respuesta HTTP, etc.

3.1.3 Nessus

Nessus es una aplicación para escanear vulnerabilidades en aplicaciones web, es uno de los muchos productos de la compañía “Tenable”. Asimismo tiene una versión de prueba gratuita de 7 días, después solo está la versión de pago. Al igual que Acunetix, cuenta con una interfaz web bastante intuitiva en la cual se configuran todos los parámetros relacionados con el escaneo [37].

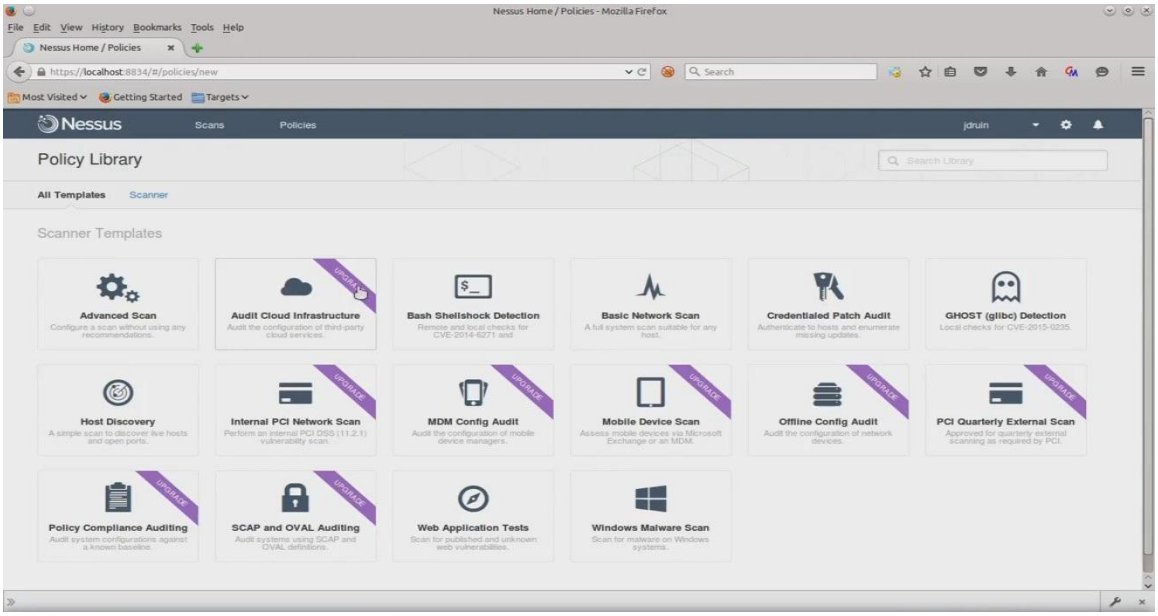


Figura 12. Políticas y Tipos de escaneos que permite NESSUS.

Algunas de las particularidades de esta aplicación, es que va un poco más allá del simple escáner de vulnerabilidades, te da otras opciones de escáner, como son, escáner de host (para descubrir los hosts que hay en una red o subred), escáner de puertos (similar a un NMAP, te permite ver que puertos hay abiertos en una maquina), Windows malware Scan, entre otros.

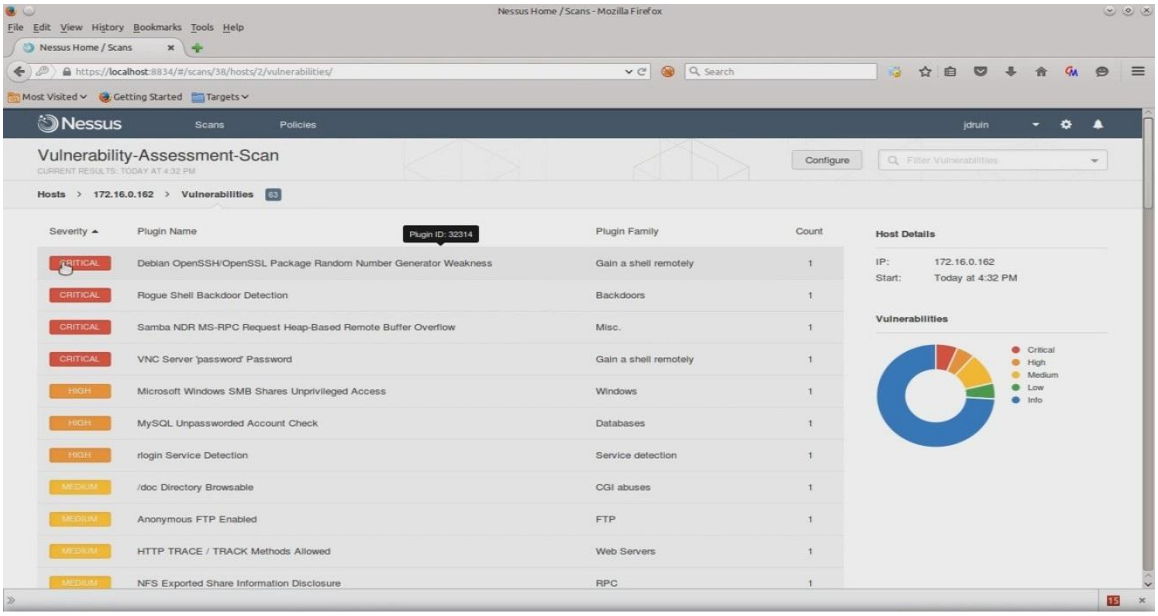


Figura 13. Resultado de escáner NESSUS.

Tal y como podemos ver en la Figura 13, te clasifica las vulnerabilidades que encuentra en una maquina dependiendo del nivel de importancia de las vulnerabilidades

que encuentra, cada una se puede inspeccionar para ver los detalles.

Una de las ventajas que tiene Nessus sobre las otras herramientas nombradas anteriormente, es que, este se puede integrar con MetaExploit, más concretamente podemos cargar ficheros resultados de Nessus, dentro del framework de MetaExploit. Por ejemplo, para que en caso de que hayamos encontrado una vulnerabilidad que tenga un Exploit asociado (Nessus te detalla cuales de las vulnerabilidades tienen un Exploit asociado), poder lanzarlo desde MetaExploit muy fácilmente sobre la máquina objetivo [38].

3.2 PROXY WEB

Un proxy web es una aplicación que se utiliza para interceptar, inspeccionar, bloquear o permitir, todas las comunicaciones HTTP y HTTPS, que se producen entre un cliente, normalmente un navegador web, y un servidor que da servicio de una aplicación web o sitio web, normalmente por el puerto 80 o 443.

Las motivaciones de utilizar un proxy web son variadas, se puede utilizar por motivos de seguridad (restringir acceso a sitios determinados, rechazar cierto tipo de respuestas o peticiones, etc.), por motivos de anonimato (para navegar anónimamente por Internet), por motivos de rendimiento (puede funcionar como cache de páginas más visitadas o recientes, o porque tenga mejor conexión, etc.) o simplemente como control de tráfico [39].

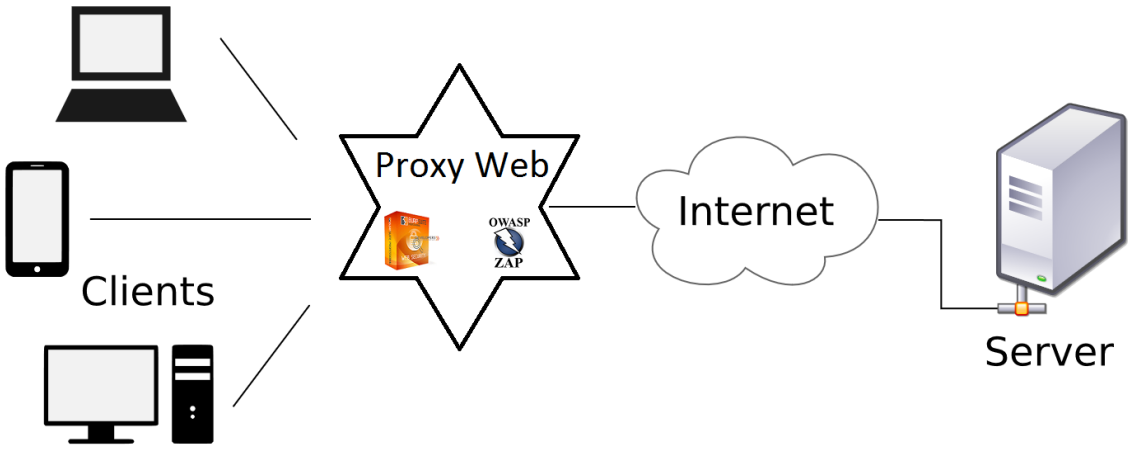


Figura 14. Esquema cliente – servidor con proxy web.

Bueno si recordamos el funcionamiento del cliente – servidor, el proxy web viene a estar en medio de dicha comunicación, es decir, ahora el cliente realiza la petición sobre el servidor proxy, este opcionalmente la procesa, y la envía sobre el servidor web, este la

procesara y le enviara la respuesta al servidor proxy, este opcionalmente procesara la respuesta y se la pasara al cliente inicial.

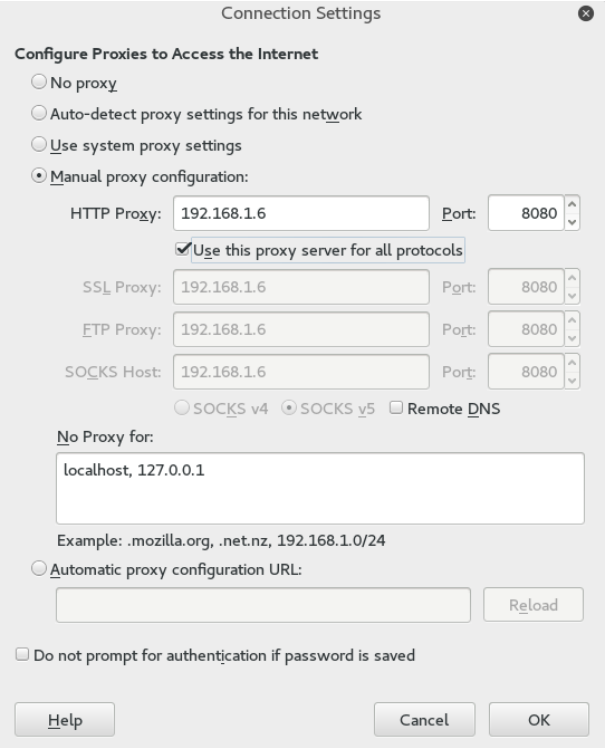


Figura 15. Ejemplo configuración Proxy Web en Firefox.

De aquí en adelante, para trabajar con el proxy web de ZAP y de Burp Suite, necesitaremos una maquina conectada a la maquina del cliente (o puede ser la misma), que tenga instalada las aplicaciones ZAP o Burp Suite, esté ejecutándose y tengamos el navegador del cliente configurado para que todas las peticiones pasen por dicho proxy web.

3.2.1 OWASP ZAP

OWASP Zed Attack Proxy (ZAP) es una de las herramientas de seguridad desarrollada y activamente en mantenimiento por la organización OWASP. Te permite encontrar vulnerabilidades web de manera muy sencilla, pero sobretodo te permite interceptar las peticiones/respuestas de una comunicación HTTP. La herramienta es gratis, siguiendo con la filosofía de OWASP [40].

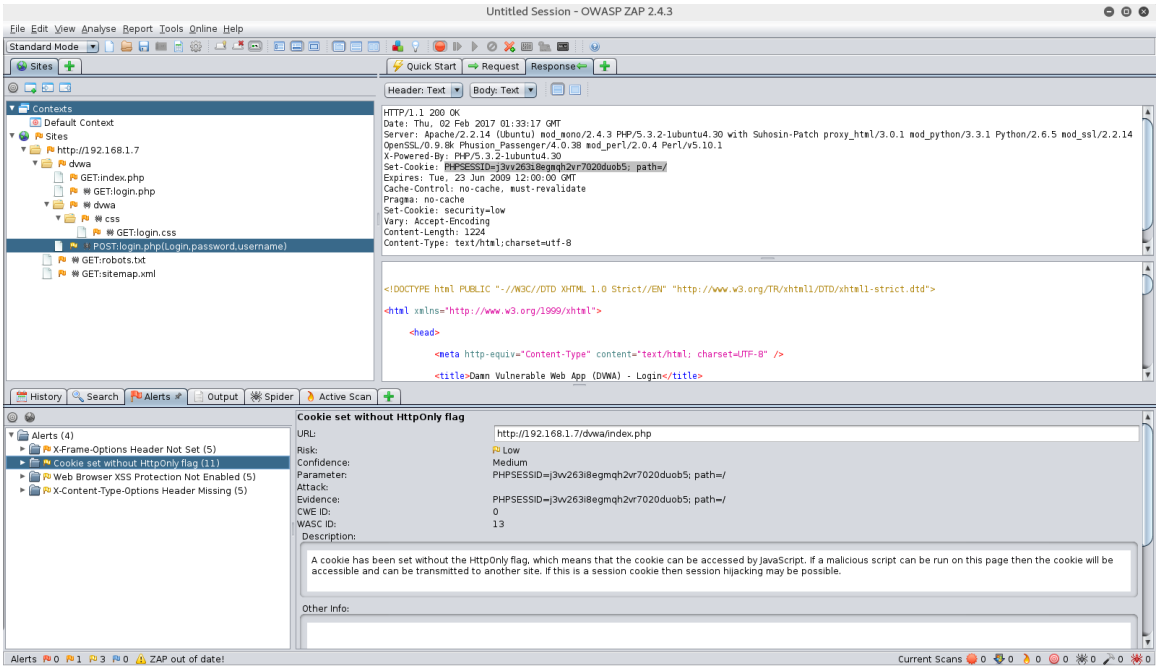


Figura 16. OWASP ZAP proxy Response HTTP

En la pantalla inicial, cuando arrancamos la herramienta, arriba a la izquierda (en un desplegable) podemos seleccionar entre 4 modos de funcionamientos distintos que detallo a continuación:

1. Safe Mode, este modo se utiliza cuando no tenemos permiso del propietario de la pagina o aplicación web para "lanzar" ataques o hacer escaneos de dicha web. Es decir, realiza una funcionalidad reducida de la propia herramienta. Hay que tener en cuenta, que para el resto de los modos, tenemos que tener permiso explicito de la persona dueña de la aplicación web, o realizarlo sobre una aplicación web que tengamos nosotros mismos, incluso podemos utilizar las aplicaciones vulnerables que proporciona la misma organización de OWASP para hacer pruebas.
2. Protected Mode, este modo se utiliza para realizar los escaneos y ataques solo de las direcciones que tengamos en objetivos, así trazamos un perímetro de nuestro objetivo y evitamos que nuestros ataques "salten" a otros enlaces o paginas vecinas de nuestro objetivo principal, en ocasiones nos puede interesar, por ejemplo, si solo queremos examinar la pagina de login ante posibles fallos.
3. Standard Mode, este modo se utiliza para realizar toda la funcionalidad en todas las páginas, las que tenemos en objetivos como las que no.
4. Atack Mode, similar al Standard Mode, con la diferencia de que a medida que va encontrando nuevos nodos, también los examina y los añade a la lista de objetivos.

Ahora detallare alguna funcionalidad con la que cuenta esta herramienta:

- Quick Start – Attack, es la forma más rápida de empezar a trabajar con esta herramienta, te permite introducir la dirección del objetivo a la que ZAP atacara, lanzara una araña web que irá descubriendo la estructura de la aplicación y también lanzara algunos "ataques" que se basan únicamente en identificar las cabeceras y análisis de código estático. Tiene algunos inconvenientes, por ejemplo las paginas que tienen login no intentara acceder, y tampoco tenemos mucho control sobre las fases por las que pasa cada ataque, ZAP permite mucho más.
- Arañas o bots, los ataques de ZAP solo pueden dirigirse a páginas de una aplicación que sabe que están ahí, lo que quiere decir que tiene que explorar la aplicación objetivo, para saber la estructura que tiene en base a directorios y ficheros. Se pueden diferenciar tres tipos de arañas dentro de la herramienta, una que es la más tradicional, va buscando los enlaces dentro de las etiquetas HTML, es el que usa el "Quick Start – Attack", la otra araña (avanzada) se puede configurar la profundidad de la búsqueda, si procesa los formularios, los comentarios HTML, el fichero robots.txt, sitemap.xml, metadatos, etc. Y luego hay otra denominada AJAX Spider, que se utiliza para buscar enlaces dentro de los Javascript que se va encontrando.
- Proxy web, una araña o Bot no es capaz de interpretar cierta información como lo haría un humano. Para estos casos tenemos el proxy web, el cual nos permite detener y examinar cada petición y respuesta entre el cliente y servidor. Podemos obtener desde un fingerprinter del servidor, hasta el PHPSESSIONID y mucha otra información relevante y sensible de la aplicación objetivo. Las peticiones/respuestas interceptadas pueden ser modificadas "al vuelo", también dispone de una serie de controles para establecer "breakpoints", ir paso a paso, etc.
- Escáner pasivo y activo, el escáner pasivo no modifica las peticiones ni las respuestas, se suele utilizar para detectar unas pocas vulnerabilidades en base al análisis estático de código, por ejemplo analizando las cabeceras HTTP de las peticiones y respuestas, etc. Te puede dar una visión inicial de cómo está el estado de seguridad de una aplicación web y alguna pista de donde indagar manualmente en alguna prueba de intrusión. Por otro lado tenemos el escáner activo, que intenta encontrar vulnerabilidades potenciales, utilizando vectores de ataques conocidos contra los objetivos seleccionados.
- Persistencia a través de sesiones, te permite guardar una imagen de la herramienta en su estado actual, con la finalidad de retomar el trabajo más

adelante. Te guarda los objetivos, la estructura de la aplicación, las vulnerabilidades encontradas, el historial de peticiones/respuestas, etc.

- The MarketPlace, la herramienta de ZAP cuenta con su propia API pública, la cual permite a los desarrolladores voluntarios a desarrollar sus propios AddOns, Plug-in, herramientas que permiten ampliar la funcionalidad base que proporciona. Asimismo las pueden publicar, para así compartir con la comunidad sus creaciones. Cuando se publica un Plug-in, se puede especificar como en fase Alpha, Beta o Release, para indicar si el Plug-in está aun en desarrollo o ya está completamente terminado.
- Decode/Encode/Hash, te permite hacer codificaciones y decodificaciones entre Base64, ASCII, HEX, URL, HTML, Javascript, UTF-8. También te transforma texto a SHA1 y MD5.
- Fuzzer, te permite seleccionar una página de la aplicación objetivo, inspeccionar la petición y seleccionar donde inyectar posibles variaciones de cadenas de texto para que la herramienta vaya probando todas ellas y ver si existe o no. Por defecto incluye unos pocos ficheros con las cadenas de texto más usuales.

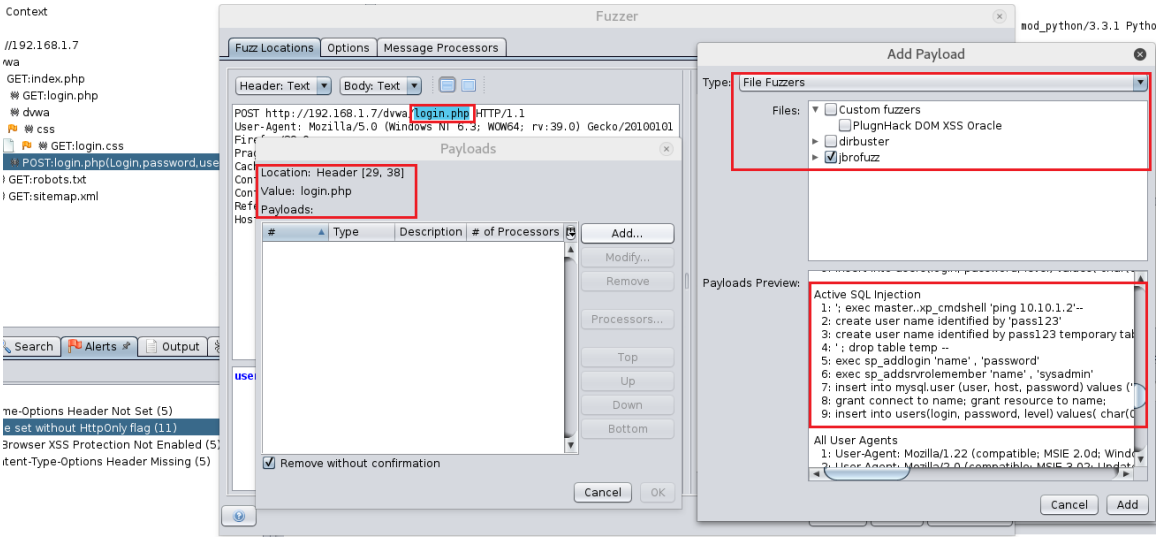


Figura 17. OWASP ZAP, Configurar Fuzzer con ficheros predefinidos.

3.2.2 Burp Suite

Burp Suite es una herramienta integrada para realizar el análisis de la seguridad de las aplicaciones web. Esta cuenta con varias funcionalidades integradas a modo de componentes que permiten realizar una auditoría a nuestras aplicaciones web mal configuradas o desarrolladas. Además permite ampliar dicha funcionalidad, gracias a que cuenta con su propia API para desarrollar Plug-in adicionales y poder compartirlos con la comunidad.

WorkFlow

Primero voy a empezar explicando cómo sería el flujo de trabajo habitual con la herramienta a la hora de realizar una supuesta auditoria de seguridad a grandes rasgos.

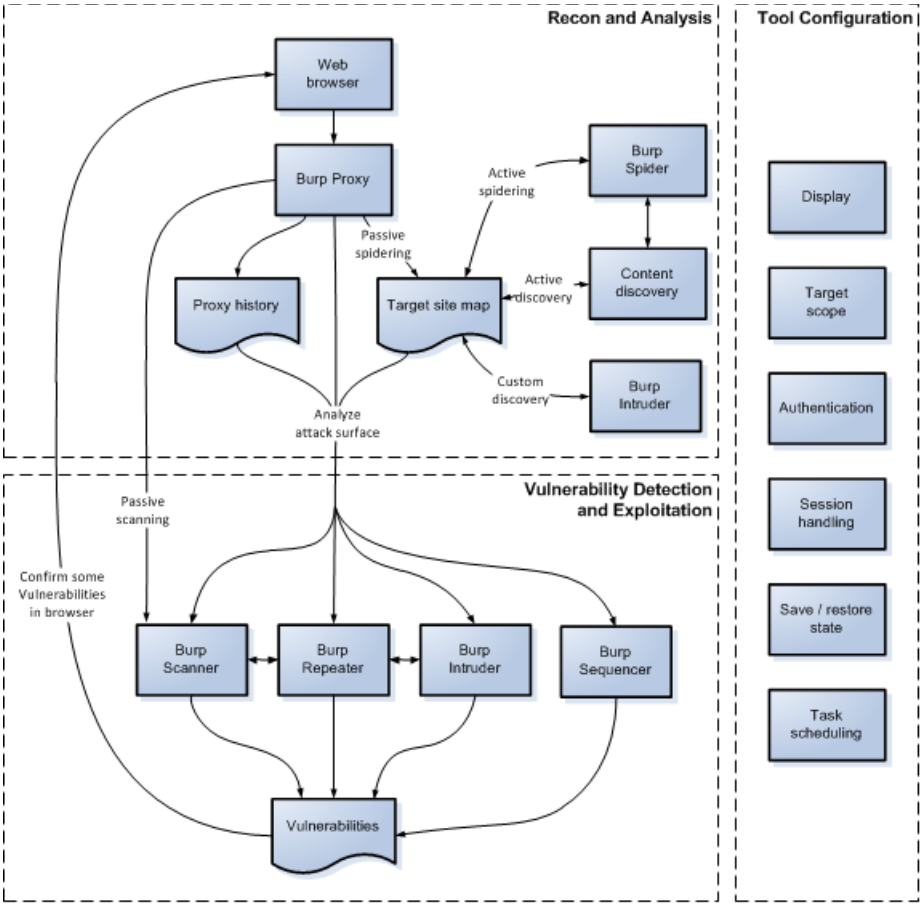


Figura 18. Burp Suite WorkFlow [f18]

Reconocimiento y análisis: en esta fase se puede decir que el corazón es el proxy web, te deja que el navegador interactué con la aplicación web mientras que Burp Suite captura toda la información relevante la cual te sirve para poder realizar otras acciones.

En esta fase tenemos el mapeo de la aplicación web, se puede hacer manualmente (clicando en los enlaces mientras navegamos, usando la aplicación en general) lo cual

poblara el historial del proxy en la herramienta y el "Target Site Map" con todo el contenido de las peticiones, para posteriormente poder analizar si hay información de más no pedida explícitamente, etc. O bien se puede realizar un mapeo automatizado, el cual puede ser llevado a cabo mediante una araña (crawler) totalmente configurable, con el "Content Discovery" que es una función usada para descubrir contenido que no es visible a tu navegador o a la araña, o con el "Custom Discovery" que sería una configuración especifica del Burp Intruder, la cual te permite descubrir contenido con técnicas más intrusivas.

Con toda la información obtenida en el historial del proxy y de la estructura de la aplicación, así como algunos de sus ficheros (los obtenidos directamente en las respuestas del servidor web), podemos empezar a analizar (estáticamente) la información que contienen y hacernos una idea de que partes pueden estar más expuestas a un posible ataque.

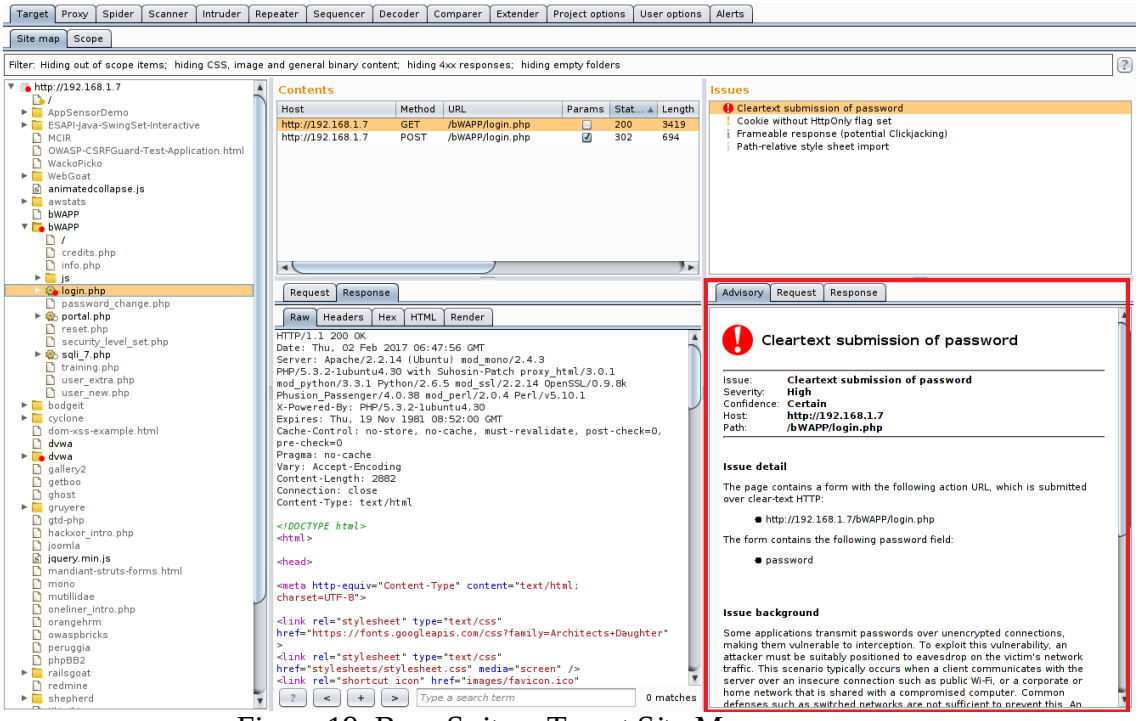


Figura 19. Burp Suite – Target Site Map.

Configuración de herramientas: Como casi para cualquier programa, la configuración por defecto se suele usar para un uso rápido de la aplicación, a medida que vamos personalizando dicha configuración, vamos definiendo distintos niveles o fases de nuestras pruebas/testeos.

Por ejemplo, puedes configurar la fuente y juego de caracteres usados para mostrar los mensajes de HTTP y de la propia interfaz de Burp Suite. También, el Target Scope, el cual te permite seleccionar sobre que aplicaciones vas a realizar las pruebas de seguridad

y así centrarnos solo en ellas. Asimismo la forma en la que la herramienta se enfrentara a los formularios que se vaya encontrando, etc.

Vulnerabilidades, detección y explotación: Ahora tenemos que detectar las vulnerabilidades y si fuera necesario intentar explotarlas. Con el menú contextual de la herramienta, una vez tenemos nuestra aplicación objetivo en el "Scope" podemos pasar las paginas o partes de la aplicación entre las diferentes "pestañas" que al fin y al cabo son otras herramientas con su propia funcionalidad dedicada.

Con la configuración por defecto de Burp Suite (ver Figura 19) tenemos un escáner pasivo activado, de todas las peticiones que pasan a través del Proxy web (Basado en análisis estático de código), el cual puede ir descubriendo algunas irregularidades con respecto a la aplicación web.

Por otro lado, tenemos un escáner activo de vulnerabilidades, que envía peticiones para buscar vulnerabilidades en la aplicación web. Como siempre, comentar que hay que utilizar esta función con precaución y siempre sobre paginas nuestras o sobre las que tengamos permisos para realizar pruebas.

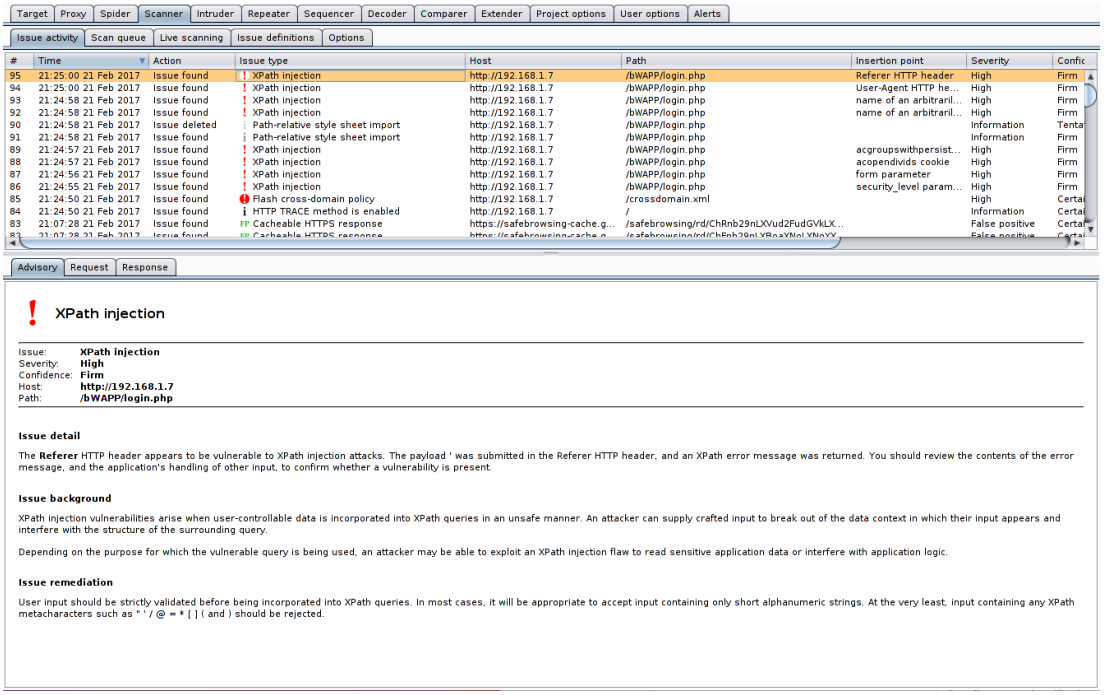


Figura 20. Burp Suite – Escáner de vulnerabilidades.

Asimismo contamos con un Fuzzer, para probar nuestras propias cadenas de texto y Payload en las posiciones que nosotros configuremos. Al igual que con el Burp Repeater, podemos escoger una petición modificarla manualmente, y reenviarla una y otra vez para probar distintos fallos, etc.

Una vez tenemos identificadas las vulnerabilidades, la herramienta nos proporciona el Burp Intruder para poder explotar dichas vulnerabilidades, cargando un Payload de una serie de listas pre-definidas de la propia herramienta o también podemos intentarlo de forma manual [41].

Funcionalidades de la Herramienta Burp Suite

A continuación voy a explicar las diferentes pestañas que tiene la herramienta, que al fin y al cabo, son las diferentes herramientas que integran Burp Suite.

Target: Esta herramienta contiene información detallada sobre nuestras aplicaciones objetivo y nos permite dirigir el proceso de Pentesting de vulnerabilidades. Más concretamente, tiene información sobre el árbol de directorios y ficheros de cada una de nuestras aplicaciones web objetivo (Sub-pestaña "Site Map"), así como información referente a un análisis pasivo de vulnerabilidades. Por otra parte, también tenemos la lista que define cuales son las aplicaciones web objetivo (Sub-pestaña "Scope").

Asimismo cuando queremos trabajar con un elemento concreto del árbol de directorios, nos da la opción de seleccionarlo y poder enviarlo al resto de herramientas (pestañas).

Proxy: Esta herramienta es el proxy web, que funciona como un Man-In-The-Middle entre el navegador y la aplicación web objetivo. Te permite interceptar, inspeccionar y modificar todo el tráfico que pasa a través de él en ambas direcciones. Además posee un registro del historial de todas las peticiones que se realizan durante la sesión, lo que te permite poder inspeccionarlas en otro momento, en el caso de que no se desee trabajar petición a petición en el momento de la ejecución.

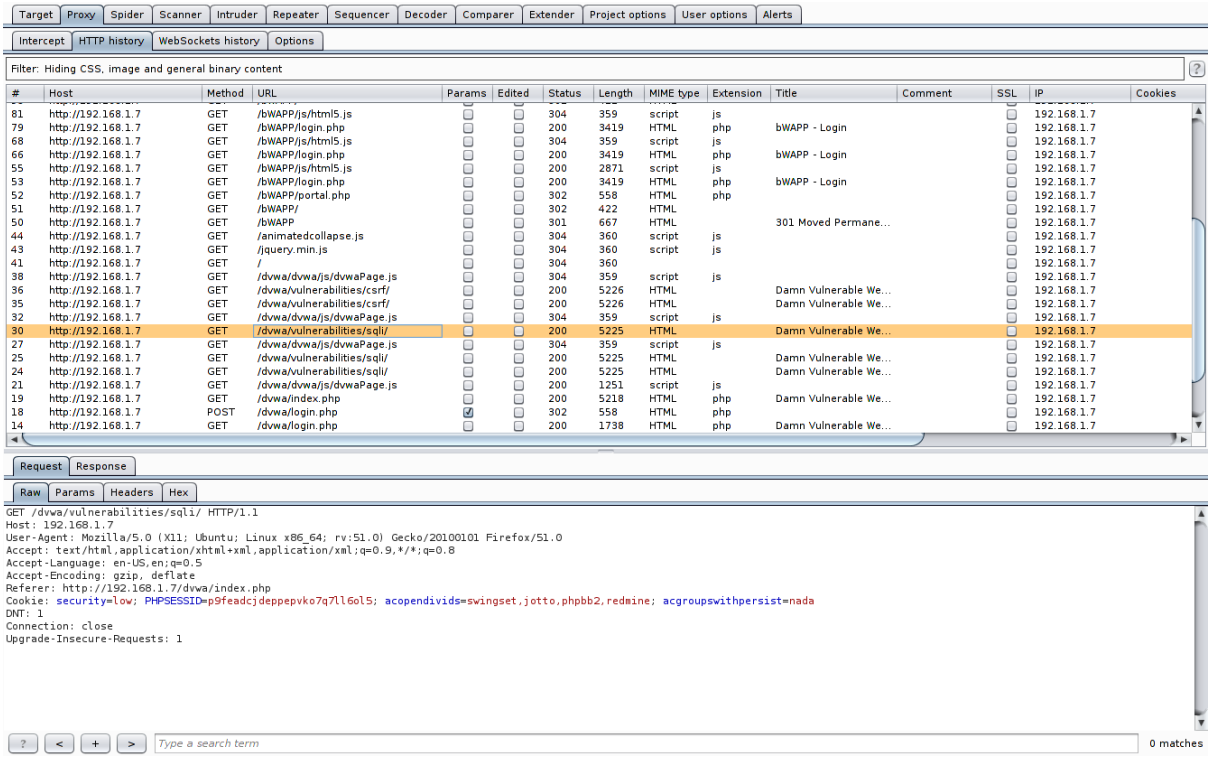


Figura 21. Burp Suite – Proxy HTTP History

Del mismo modo tenemos una Sub-pestaña de opciones que nos permite configurar este proxy, como "bindear" el proxy, como hacer redirecciones, si queremos forzar el uso de SSL, los certificados que vamos a usar (para las peticiones HTTPS), etc.

Spider: Esta herramienta tiene la funcionalidad de robot araña (crawler) con el fin de poder "mapear" todo el árbol de directorios y ficheros de nuestras aplicaciones objetivo de manera automática.

Asimismo podemos configurarla completamente, para que ignore contenido no textual (p.e. etiquetas <IMG>), la profundidad en número de enlaces que va a visitar, si queremos que envié formularios (y con qué parámetros) o no, que realice una pausa entre petición y petición, etc.

Scanner: Esta herramienta es el avanzado escáner de vulnerabilidades, diseñado para dar soporte a los Pentester. Funciona de dos maneras diferentes, por un lado tenemos el escáner pasivo, configurado de manera "por defecto" para intentar localizar vulnerabilidades sin realizar ninguna nueva petición a la aplicación objetivo, se basa únicamente en analizar el contenido que hay en las peticiones y respuestas que se producen entre el cliente y el servidor. Este modo de operación es seguro y legal en las situaciones en las que estas autorizado a acceder e interactuar con la aplicación como si fueras un usuario o cliente (End-user). Te puede alertar de campos de contraseñas que viajan en formularios en claro, si las cookies van en claro, formularios con autocompletado, transmisión insegura de tokens de sesión, etc.

En el otro lado, tenemos el escáner activo el cual envía peticiones nuevas a la aplicación objetivo con la finalidad de probar alguna vulnerabilidad. Se basa en modificar las peticiones "base" de diferentes formas, diseñadas para intentar activar comportamiento que indique la presencia de una o varias vulnerabilidades. Este modo de trabajo genera muchas peticiones que son "maliciosas" y las cuales pueden comprometer la aplicación, por eso es muy importante tener precaución y usarla sobre nuestras propias aplicaciones, o contando con el permiso del dueño de la aplicación.

* Lista de posibles vulnerabilidades que puede detectar Burp Suite [42].

Intruder: Esta herramienta trata de explotar las vulnerabilidades previamente detectadas. Cuenta con varios vectores de ataques semi-automáticos y totalmente configurables.

La forma habitual de trabajar con esta herramienta seria, introducir un objetivo (usualmente uno de nuestros objetivos en "Scope"), seleccionamos la pagina por la que vamos a intentar acceder, seleccionamos las posiciones de inyección de código, el modo en que se realizaran estas intrusiones de código, el Payload que queremos introducir y lo lanzamos.

Repeater: Esta herramienta es el repetidor, se utiliza para modificar manualmente una petición y poder enviarla, una y otra vez, y así analizar las respuestas del servidor y su comportamiento.

Sequencer: Esta herramienta se utiliza para cuantificar la "aleatoriedad" de los tokens de sesión u otros objetos importantes en las aplicaciones web que se supone que son, a priori, impredecibles.

Se basan en tests estadísticos estándar sobre la aleatoriedad. Se asume la hipótesis de que todos los tokens son totalmente aleatorios y se ponen a prueba comprobando los tokens que se generan y observando las características de dichos tokens (p.e. si tienen patrones repetidos), entonces se calcula una probabilidad de que esta característica o patrón se repita.

Decoder: Esta herramienta se utiliza para codificar, decodificar y transformar datos con los que trabaja la aplicación. Soporta URL, HTML, Base64, ASCII HEX, HEX, OCTAL, Binario, GZIP, y algunos Hash como son MD2, 5 y SHA256, 512.

Comparer: Esta herramienta se utiliza para comparar dos páginas (peticiones y/o respuestas) y ver en que partes de las mismas difieren.

Extender: Esta herramienta te permite cargar Plug-in a modo de extender la funcionalidad base de Burp Suite, utilizando tu propio código. También te permite

descargar otros Plug-in de la comunidad creados por otros usuarios, al igual que puedes compartir tus propias creaciones con la comunidad.

Asimismo tiene su propia API la cual se utiliza para obtener funcionalidad con la que trabaja la propia herramienta. Además esta, admite tres lenguajes de programación, Java, Python y Ruby [43].

CAPÍTULO 4. ESTADO DEL ARTE

En este capítulo mostrare algunos de los Plug-in más populares y usados por la comunidad de Burp Suite, ya que se extendería demasiado si hablara de todos los que hay (son una lista bastante amplia de Plug-in oficiales y luego también están los “no oficiales” o los que son subidos por desarrolladores en su propia cuenta de GitHub o dominio).

Hay Plug-in de todo tipo, desde los más genéricos que simplemente te cambian el color de un texto para indicarte que ha habido un cambio importante hasta los más específicos que te hacen un decode de una cookie para una aplicación especifica.

4.1 OTROS PLUG-IN DE BURP SUITE

4.1.1 CO2

CO2 es una extensión para la aplicación de Burp Suite desarrollada por Jason Gillam escrita en Java, públicada bajo una licencia de Apache License, Versión 2.0. Tenemos dos maneras de descargar CO2, mediante la propia aplicación de Burp Suite, entrando en la pestaña de “Extender”/“BApp Store” y clicando en instalar. La otra forma es a través de la página de GitHub del propio autor, descargándonos el paquete ".JAR" y cargándolo en la aplicación de Burp Suite.

CO2 cuenta varias funcionalidades divididas en sub-pestañas, que iré detallando a continuación:

CO2-SQLMapper: este modulo muestra una interfaz sencilla mediante la cual podemos generar sentencias para el programa de SQLMap. La idea es reducir la probabilidad de "fallo humano" y aumentar la eficiencia a la hora de realizar pruebas de intrusión con la herramienta de SQLMap. SQLMap [44] es una herramienta que te permite automatizar el proceso de detección y explotación de vulnerabilidades relacionadas con SQL Inyection.

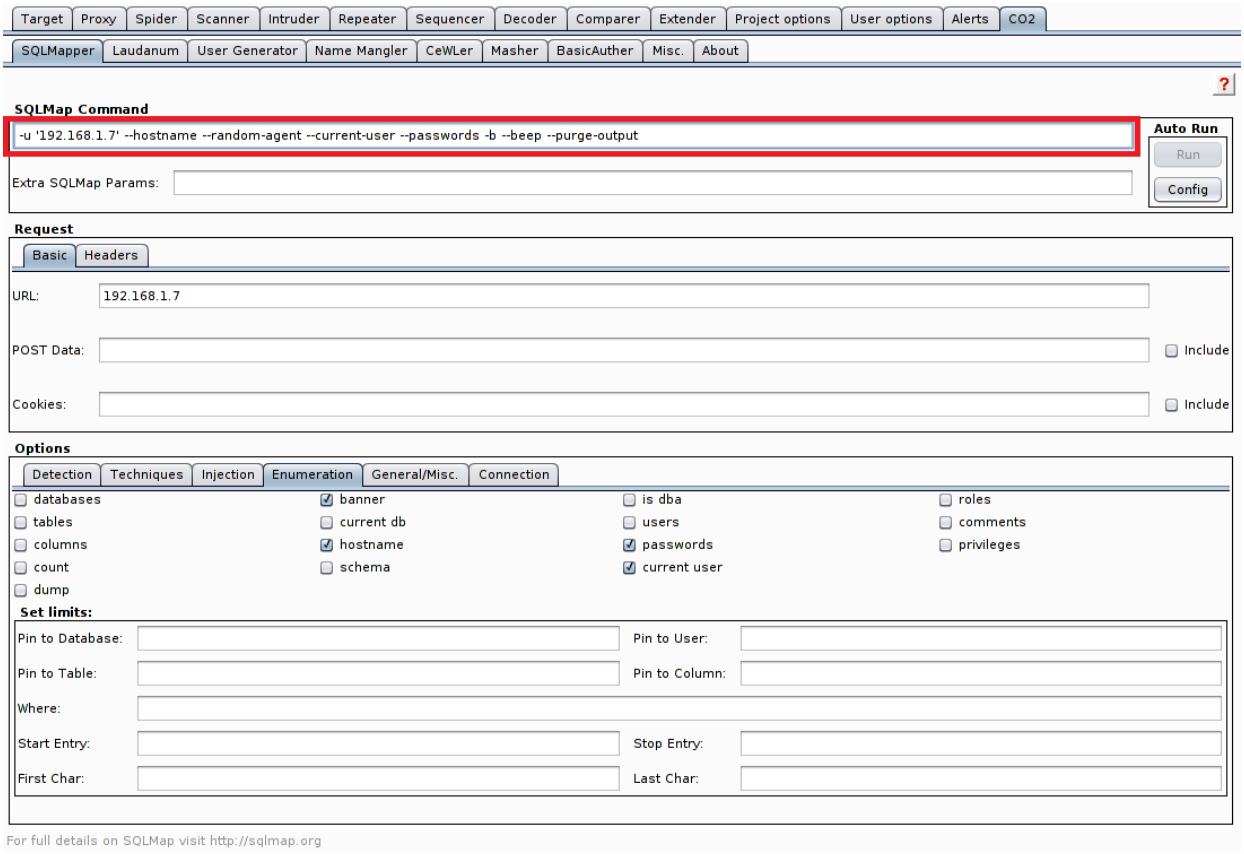


Figura 22. Burp Suite Extensión – CO2-SQLMapper

CO2-Laudanum: este modulo de basa en Laudanum y genera un script inyectable para explotar ciertas vulnerabilidades y devolvernos el control de la máquina de la victima mediante una Shell remota (normalmente). Laudanum [45] es un conjunto de scripts diseñados para pruebas de intrusión en el momento en el que una aplicación web nos permite subir ficheros y estos no pasan por ningún tipo de control. Estos scripts suelen estar escritos en diferentes lenguajes de programación y para diferentes entornos.

CO2-User Generator: este modulo crea una lista de usuarios o usernames o nicknames basados en censos del gobierno y listas publicas de nombres, etc. Se pueden configurar algunos parámetros como el rango de año que nacieron los nombres (los nombres los saca por décadas) o el porcentaje que va usar del censo.

CO2-Name Mangler: este modulo también crea una lista de usernames, en función de dos diferentes listas cortas de nombres y apellidos, creando diferentes combinaciones

y concatenaciones de ambas.

CO2-CeWLer: este modulo se basa en la herramienta de CeWL [46] para extraer palabras de una petición/respuesta extraída de Burp Proxy History con la finalidad de obtener una lista de palabras basadas en el contenido de la propia aplicación (en base a peticiones y respuestas de la misma). Se puede configurar la longitud mínima y máxima de las palabras que extrae, así como ignorar lo que va entre etiquetas <style> o <script>, etc.

CO2-Masher: este modulo genera un diccionario de posibles contraseñas en función de una lista de palabras con la idea de utilizar dicho diccionario en el Burp Suite Intruder para intentar acceder a posibles login de alguna aplicación web [47].

4.1.2 Headers Analyzer

Headers Analyzer es una extensión para Burp Suite creada por Antonio Sánchez escrita en Python. Al igual que la anterior, tiene dos formas de instalar/descargar, mediante la propia BApp Store de Burp o mediante el GitHub del Autor.

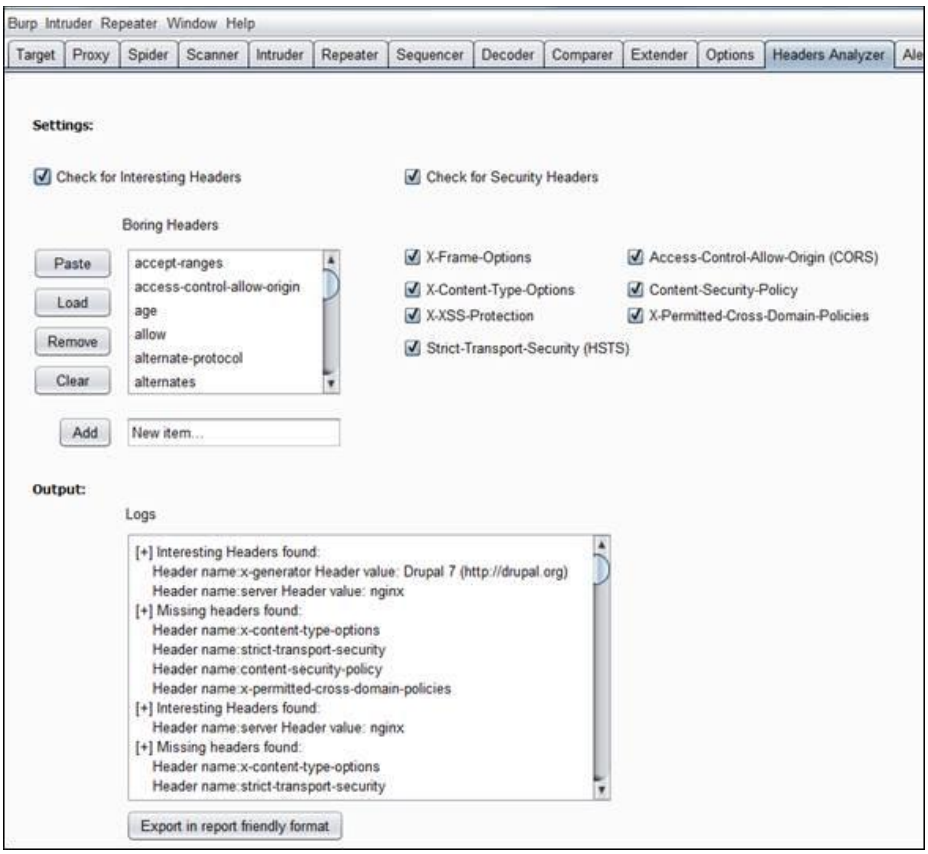


Figura 23. Burp Suite Extension – Headers Analyzer

La principal función de este Plug-in es hacer uso del escáner pasivo de la propia herramienta de Burp Suite para analizar estáticamente las cabeceras de HTTP/HTTPS en busca de cabeceras que puedan desvelar información importante, o informar de que faltan cabeceras de seguridad o que se encuentran pero están mal configuradas.

Una vez encontrada una cabecera que falta o que está mal configurada, te genera un informe del mismo modo que el propio escáner pasivo de Burp Suite. Además te permite exportar dichos errores o fallos en otros ficheros [48].

4.1.3 JSON Decoder

JSON Decoder es una extensión para Burp Suite creada por Michal Melewski escrita en Python. Del mismo modo que casi todos los Plug-in lo podéis encontrar en el GitHub del autor y en la propia BApp Store de Burp Suite.

La función principal de este Plug-in es decodificar los ficheros de texto de tipo ".JSON" que pasan a través del proxy, para ello añade una nueva sub-pestaña en la pestaña de proxy de la propia herramienta de Burp Suite [49].

4.1.4 EsPReSSO

EsPReSSO es una extensión de Burp Suite creada por Tim Guenther, Christian Mainka y Vladislav Mladenov escrita en Java y publicada bajo una licencia de GNU General Public License v2.0. Se encuentra disponible tanto en el GitHub del Tim Guenther, como en el BApp Store de Burp.

La idea principal de este Plug-in es ofrecer a los Pentester y desarrolladores una herramienta para identificar y clasificar los diferentes protocolos SSO (Single Sign-On) y una serie de vectores de ataques SAML (Security Assertion Markup Language).

Single Sign-On es un procedimiento mediante el cual un usuario se puede autentificar en una plataforma/aplicación como Google, Microsoft, Facebook, Amazon, etc. y utilizar dicha autentificación para acceder a otras plataformas. En cierto modo es como tener un solo usuario/contraseña para un cliente y utilizarlo en muchas aplicaciones web. Por ejemplo, Google utiliza SSO, cuando nos autentificamos en Gmail, si accedemos a una página de Youtube vemos que nos carga la cuenta de Gmail [50].

EsPReSSO identifica y trabaja con los siguientes protocolos: SAML, OpenID, OAuth, BrowserId, OpenID Connect, Facebook Connect, Microsoft Account.

Integra la herramienta de WS-Attacker en el caso de detectar el protocolo de SAML, para automatizar cierto tipo de ataques SAML. WS-Attacker es un framework modular para realizar pruebas de intrusión en servicios web, como SOAP o REST [51].

EsPReSSO también facilita la lectura y edición de mensajes SAML, añade una sub-pestaña de SAML en la pestaña de historial de Burp Suite y añade una opción al menú contextual para analizar protocolo SSO sobre una determinada petición/respuesta. Asimismo incluye editores para SAML, JSON y JSON Web Token (JWT).

Para utilizar este Plug-In se necesita instalar otras dependencias, como son, RSyntaxTextArea, Json-simple y WS-Attacker, toda la información sobre dichas dependencias, así como sus licencias y enlaces, están bien documentados en la propia página de GitHub del autor [52].

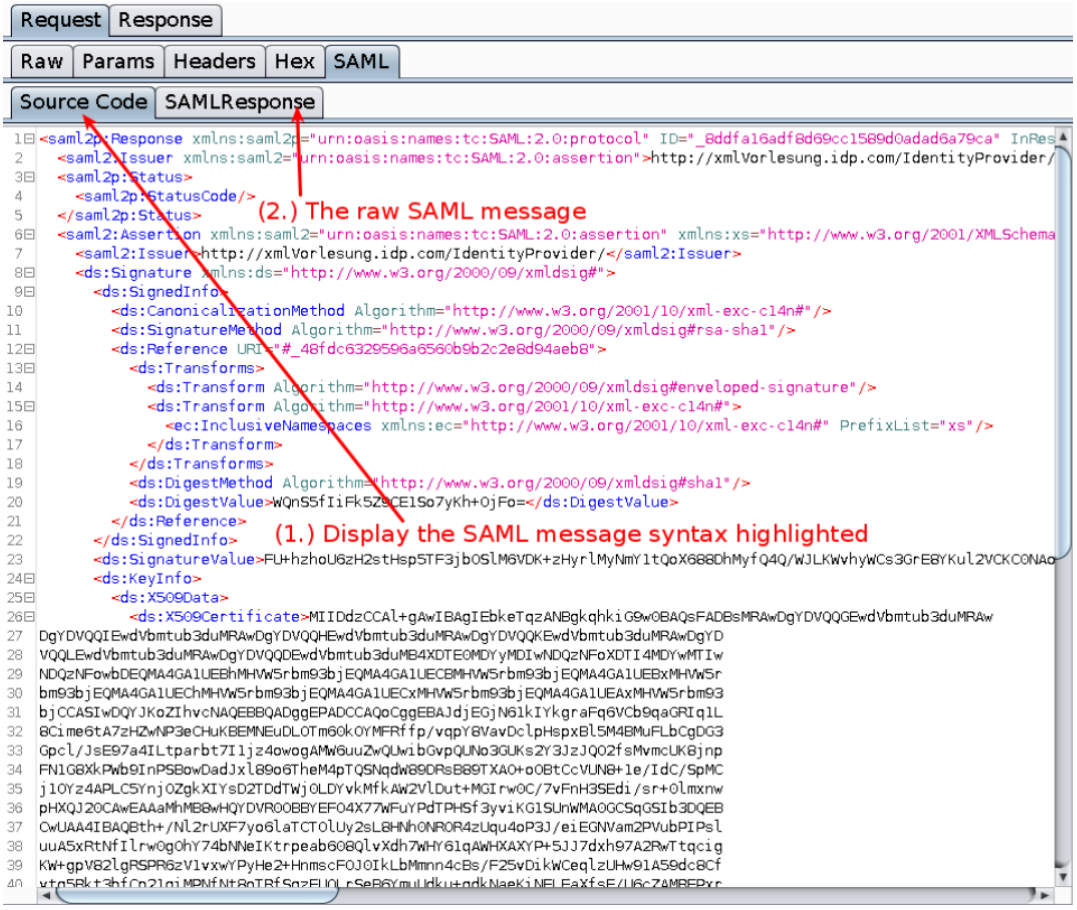


Figura 24. EsPReSSO – SAML Editor [f24]

4.1.5 SQLiPy

SQLiPy es una extensión para Burp Suite que integra la herramienta de SQLMap con Burp Suite. Creada por Josh Berry, se puede instalar directamente desde la BApp Store de Burp Suite.

Tiene como dependencias Jython +2.7 y una instancia de SQLMap ejecutándose. La extensión crea una nueva pestaña llamada SQLiPy, la cual tiene otras cuatro sub-pestañas. En una de estas pestañas "SQLMap API" podemos configurar la dirección IP y puerto en la cual tenemos nuestra instancia de SQLMap (Puede ser la misma máquina).

Una vez configurada podemos ir a cualquier petición/respuesta capturada por Burp Suite y mediante el menú contextual enviar dicho objeto al escáner de la extensión, el cual nos rellenara la sub-pestaña de "SQLMap Scanner" y podremos lanzar el escaneo [53].

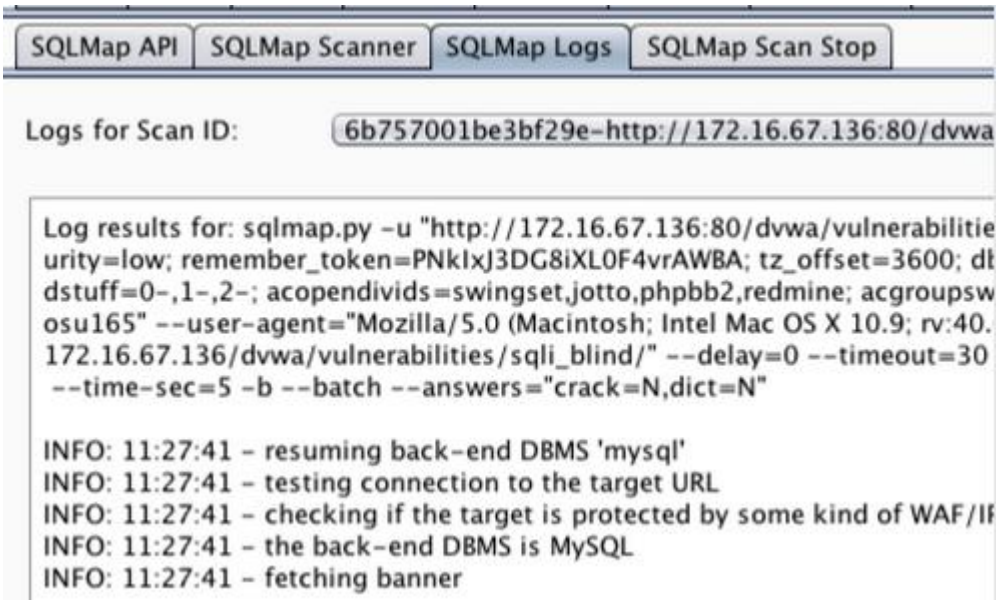


Figura 25. Burp Suite SQLMap extensión – SQLMap Logs

CAPÍTULO 5. PRUEBAS DE CONCEPTO DE VULNERABILIDADES

En este capítulo veremos desde un punto de vista más práctico las vulnerabilidades en aplicaciones y tecnologías web más habituales, basadas en la lista OWASP Top 10 [54]. Hablare de cuáles son, que esquema siguen, a que entornos afectan, la propia vulnerabilidad, y cómo podríamos proteger nuestras aplicaciones de ellas.

5.1 Inyección

Esta vulnerabilidad se basa en la capacidad de insertar código en una aplicación. Hay diferentes tipos, como ya comente anteriormente, los más habituales son SQL Inyection, XPath, HTML, XML, entre otros.

La inyección ocurre cuando un usuario envía cadenas de texto no fiables con una sintaxis muy concreta, con la finalidad de vulnerar al propio intérprete de dichos comandos. Esta vulnerabilidad permite a los atacantes inspeccionar, crear, actualizar o borrar información de la cual se alimenta nuestra aplicación. En el peor de los casos, el atacante no solo puede comprometer toda la aplicación, sino que además, podría poner en riesgo el sistema que “corre” por debajo.

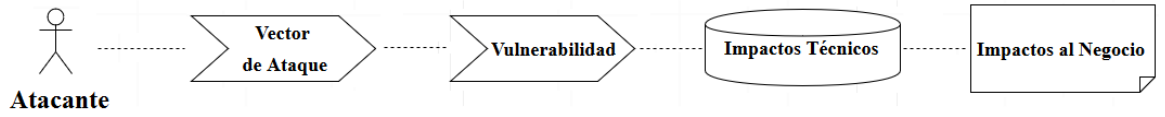
				
Cualquier persona que pueda enviar información no fiable al sistema, incluyendo usuarios internos, externos y administradores.	Los ataques se realizan enviando cadenas de texto plano, con la intención de vulnerar al intérprete de comandos.	Cuando se recibe información no fiable y cuando esta se inserta en la aplicación sin ser previamente verificada y validada antes de pasársela al intérprete.	Pérdida o corrupción de la información, denegación de servicios, puede llegar a comprometer todo el servidor.	Depende del valor que su negocio de a los datos de sus usuarios/clientes, normalmente, muy alto. Perdida de reputación.

Tabla 1. PoC Inyección esquema.

Entornos afectados: Cualquier aplicación web que utilice un intérprete, incluyendo también cualquier componente de Framework que pueda usar interpretes en el Back-end.

PoC: La siguiente prueba de concepto es llevada a cabo sobre una aplicación web llamada bWAPP, perteneciente al proyecto de OWASP. Tenemos una página en la cual podemos listar las películas que hay disponibles.

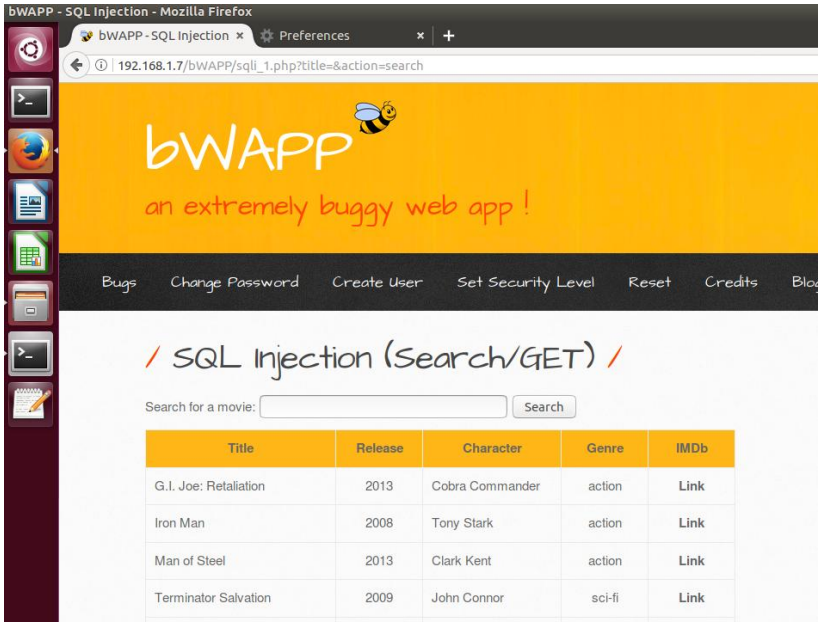
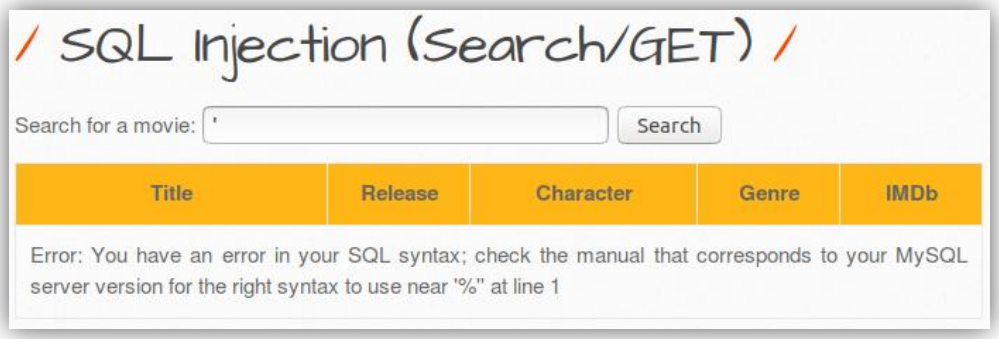


Figura 26.1 PoC SQL Inyection

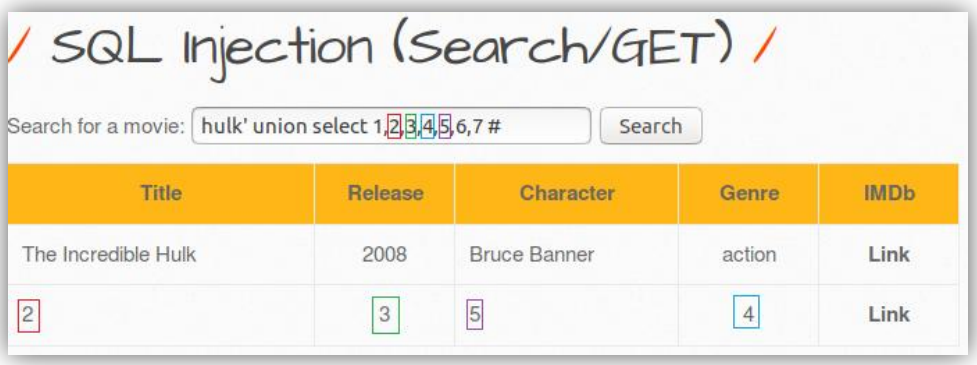
Si introducimos el nombre de una película, nos devuelve información acerca de ella.

La manera más sencilla de saber si está es vulnerable a ataques de inyección de código, es introduciendo caracteres especiales de la propia sintaxis del intérprete al que estamos "atacando". En este caso (SQL) vemos que nos devuelve un error de sintaxis cuando enviamos como cadena de texto el carácter (').



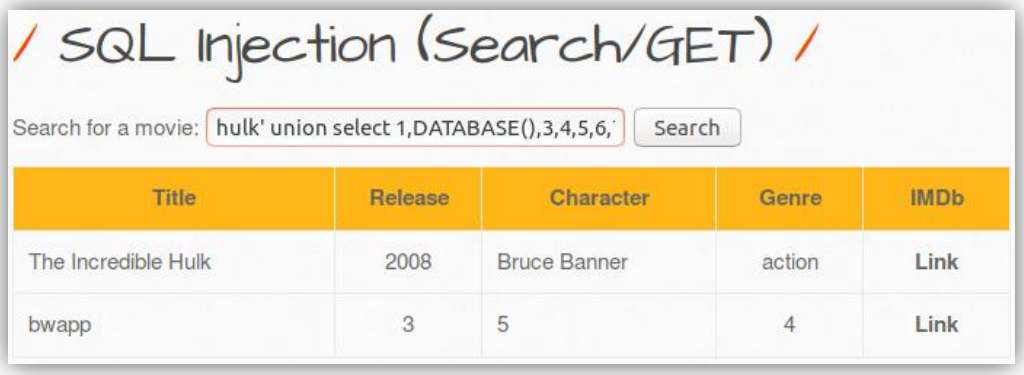
Lo siguiente es saber de cuantas columnas disponemos para mostrar la información, intuitivamente vemos que la propia aplicación nos muestra 5 campos, pero eso no quiere decir que las tablas tengan 5 columnas, así que, tenemos que averiguar cuantas columnas muestra esta vista. Utilizamos el operador UNION de SQL para combinar el resultado de 2 o más tablas. La (#) sirve para comentar el texto que venga por detrás.

Probamos “ hulk’ unión select 1,2,3,4,5 # “, estamos utilizando el SELECT con una combinación estática de columnas, para ver, de cuantas disponemos en la vista.



Cuando llegamos a 7, vemos que es el número adecuado para la vista, y además podemos ver la posición de cada columna que se muestra en la vista. Asimismo podemos obtener el nombre de la base de datos, con la función DATABASE().

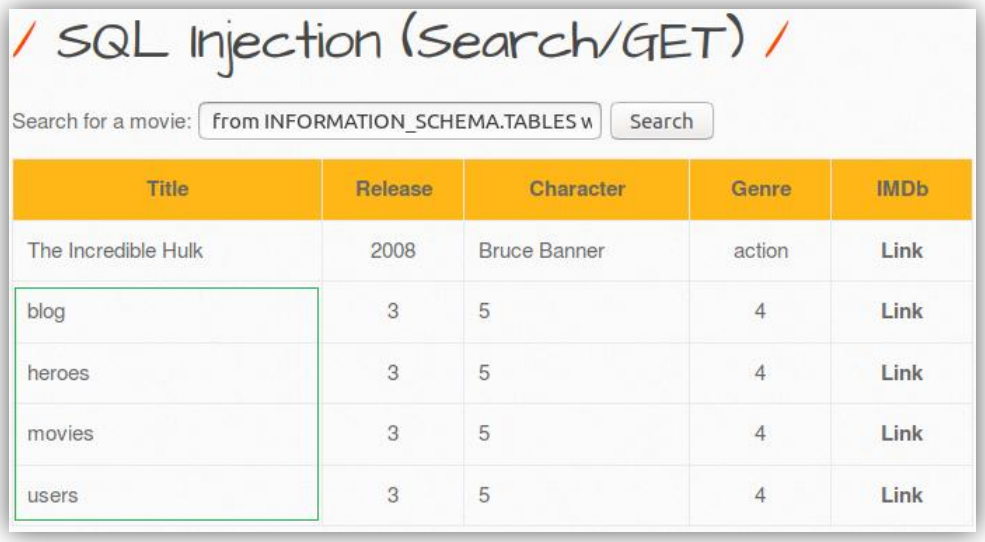
hulk' UNION SELECT 1,DATABASE(),3,4,5,6,7 #



Ahora que sabemos que es vulnerable a ataques de inyección SQL, podemos probar a buscar información más interesante. Hasta ahora sabemos que tenemos 4 campos (de los 7) para mostrar información en esta vista y el nombre de la base de datos (bwapp), lo siguiente es saber que tablas existen en esta base de datos y los nombres de las columnas de esas tablas para poder mostrarlas mediante un SELECT. Para ello usamos el objeto INFORMATION_SCHEMA que contiene información sobre todas las bases de datos que están en el servidor. En las siguientes imágenes, nos centraremos en el INFORMATION_SCHEMA.TABLES que tiene información sobre todas las tablas de todas las base de datos y INFORMATION_SCHEMA.COLUMNS que tiene información sobre todas las columnas de todas las tablas.

A continuación, sacamos el número y el nombre de tablas que existen en la base de datos bwapp mediante el siguiente comando.

```
hulk' union select 1,TABLE_NAME,2,3,4,5,6,7 from
INFORMATION_SCHEMA.TABLES where TABLE_SCHEMA=DATABASE() #
```



Como podemos ver tenemos 4 tablas en la base de datos bwapp, blog, héroes, movies y users. Vamos a centrarnos en la tabla de ‘users’, ya que probablemente es la que tenga los login y password que necesitaríamos si quisiéramos acceder a dicha base de datos como usuarios con más privilegios. Para saber el número y el nombre de las columnas de dicha tabla utilizamos el siguiente comando.

```
hulk' union select 1,COLUMN_NAME,2,3,4,5,6,7 from
INFORMATION_SCHEMA.COLUMNS where TABLE_SCHEMA=DATABASE() and
TABLE_NAME='users' #
```

/ SQL Injection (Search/GET) /

Search for a movie: na=database() and table_name ='users'# Search

Title	Release	Character	Genre	IMDb
The Incredible Hulk	2008	Bruce Banner	action	Link
id	3	5	4	Link
login	3	5	4	Link
password	3	5	4	Link
email	3	5	4	Link
secret	3	5	4	Link
activation_code	3	5	4	Link
activated	3	5	4	Link
reset_code	3	5	4	Link
admin	3	5	4	Link

Podemos ver que la tabla ‘users’ tiene 9 columnas y cuáles son. Lo mismo podríamos haber hecho con otra base de datos del servidor y/o con cualquier otra tabla que nos interese. Por último, lo más fácil, hacemos un unión con un simple SELECT que nos muestre la información que nos interesa.

```
hulk' union select 1,login,password,secret,email,6,7 from users #
```

/ SQL Injection (Search/GET) /

Search for a movie: ,password,secret,email,6,7 from users # Search

Title	Release	Character	Genre	IMDb
The Incredible Hulk	2008	Bruce Banner	action	Link
A.I.M.	6885858486f31043e5839c735d99457f045affd0	bwapp-aim@mailinator.com	A.I.M. or Authentication Is Missing	Link
bee	6885858486f31043e5839c735d99457f045affd0	bwapp-bee@mailinator.com	Any bugs?	Link
admin	d033e22ae348aeb5660fc2140aec35850c4da997	admin@admin.com	admin	Link
qwe	056eafe7cf52220de2df36845b8ed170c67e23e3	qwe@qwe.com	qwe	Link

Verificación de Seguridad: el objetivo es que los datos introducidos por el usuario no puedan variar el comportamiento de los comandos o sentencias que se ejecutan en los intérpretes de las aplicaciones web. Es decir, que si tenemos un formulario de login por ejemplo, que nadie sea capaz de hacer otra cosa que no sea autentificarse.

Hay dos formas de verificar esto, de forma automática y de forma manual. La forma automática se basa en el uso de herramientas que te ayuden a detectar vulnerabilidades relacionadas con los ataques de inyección de código, algunas de estas herramientas que he comentado en el capítulo 3.

La forma manual pasa por revisar todo el código que realice llamadas a intérpretes, buscando que no se pasan parámetros directamente al intérprete, sin ser previamente validados o codificados.

Protección: Para protegernos de estos ataques, hay que tener separados el código que introduce el usuario del código de la propia sentencia y comandos, para más adelante introducir nosotros mismos las variables en la sentencia, por ejemplo, con sentencias parametrizadas que eviten que se pueda cambiar el comportamiento de dicha sentencia.

Algunas recomendaciones:

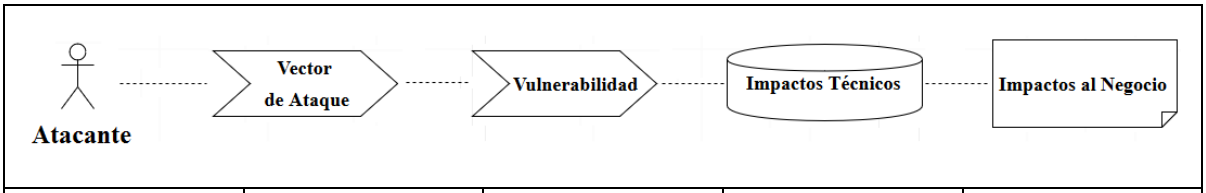
- Validar todos los Input: longitud, tipo de dato, etc.
- Usar sentencias parametrizadas: En las cuales las variables se reemplazan en la propia sentencia.
- Usar privilegios mínimos necesarios: Cuando nos conectamos a las bases de datos u otros sistemas del Back-end.
- Evitar errores con demasiado detalle: Que pueda ser usado por el atacante (p.e. versión de SQL).
- Utilizar funciones de escape de caracteres: Para evitar que los caracteres especiales puedan llegar al intérprete.

Otros ejemplos:

- CWE Entry 77 on Command Injection
- CWE Entry 89 on SQL Injection
- CWE Entry 564 on Hibernate Injection

5.2 Pérdida de Autenticación y Gestión de Sesiones

Una de las partes más críticas con respecto a la seguridad en cualquier aplicación web es tener implementado un buen mecanismo de autentificación y gestión de sesión de los usuarios de dicha aplicación. Si hay fallos en esta parte, puede llevar a perdida de credenciales y tokens de sesión, entre otras cosas.



Externos o internos al sistema, que deseen robar otras cuentas u ocultar sus propias malas acciones.	Mediante fallos en los componentes de autentificación o mala gestión de la sesión.	Normalmente los propios desarrolladores crean sus propios sistemas de login y gestión de sesiones, puede fallar en el logout, gestión de contraseñas, actualización de cuentas, etc.	Perder el acceso a una o todas las cuentas. Una vez el atacante tiene la cuenta de la víctima, puede hacer cualquier cosa como si fuera la víctima. Normalmente se buscan las cuentas con mayores privilegios.	Se pueden ver afectados datos del propio negocio al igual que cierta funcionalidad o servicio. Considerar también el impacto de cara al público sabiendo que sus cuentas no están seguras, etc.

Tabla 2. PoC Pérdida de autenticación y gestión de sesión esquema.

Entornos afectados: Cualquier aplicación web que tenga implementado un mecanismo de autentificación o haga uso de otros componentes que sirvan para el mismo fin de autentificar a un usuario determinado.

PoC: Para esta prueba de concepto utilizamos la maquina virtual de OWASP Mutillidae II. Tenemos la siguiente página de login.

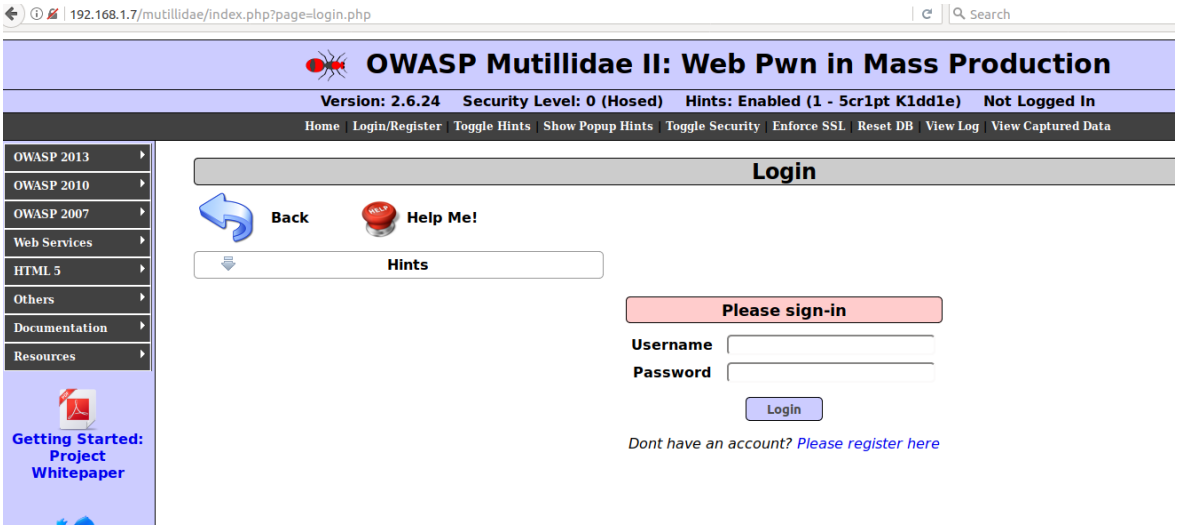
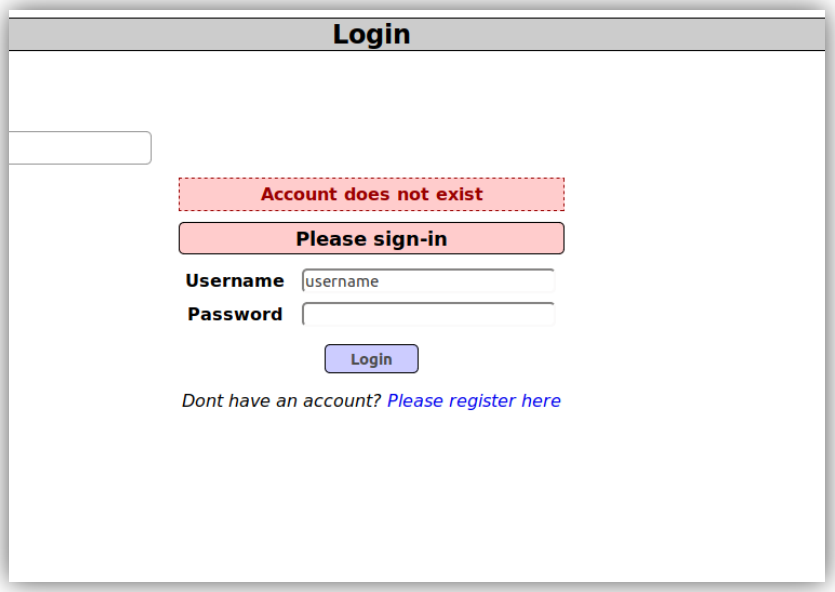
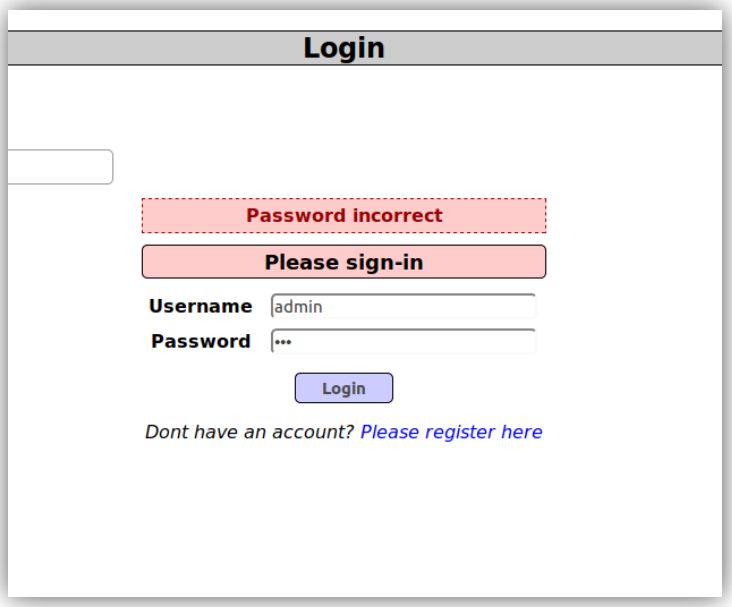


Figura 27.1 PoC Username Enumeration.

Si introducimos un nombre de usuario que no existe, nos muestra la siguiente información.



Por otro lado, si encontramos un usuario valido, y probamos con una contraseña que creemos que puede ser la correcta, nos aparece la siguiente información.



A priori, puede no parecer un gran problema. En este caso, los desarrolladores de la aplicación web, pensaron que dar información exacta sobre donde se ha equivocado el usuario no puede traer ninguna consecuencia negativa.

Sin embargo, esto puede ser muy útil para un atacante, en el sentido en el que ahora puede saber exactamente que usuarios están registrados en el sistema y cuáles no. Lo que puede llevar a "recolectar" una serie de usernames con la finalidad posterior de realizar un ataque de fuerza bruta basada en diccionarios.

Como este hay otros ejemplos, como poner el sesión id en la URL, lo que puede llevar a un ataque de elevación de privilegios, u otros mecanismos simples como no poner un tiempo de inactividad a partir del cual la propia aplicación te saque del sistema automáticamente.

Verificación de Seguridad: El objetivo sería verificar los mecanismos de autentificación para que no se puedan saltar, no den demasiada información, y proteger en todo momento los posibles tokens de sesión asociados a ese usuario autentificado y sus credenciales.

Protección: Primero asegurarse que las credenciales viajan cifradas, utilizando SSL, y que una vez están en la aplicación codificarlas para que no sean fácilmente interpretadas por algún posible "tercero", la mayoría de aplicaciones utilizan base64 o algún hash para encriptar los posibles tokens de sesión relacionados con un usuario autentificado.

Algunas recomendaciones:

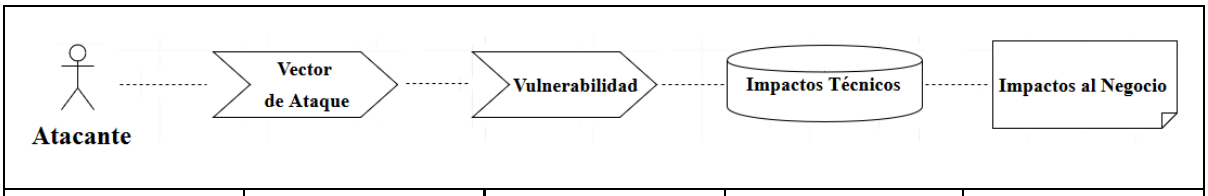
- Utilizar tus propios mecanismos de autentificación y gestión de sesiones.
- No aceptar identificadores de sesión cargados desde la URL (GET) o en la propia petición.
- El mecanismo de autentificación que utilices sea el mismo en todos los sitios de la aplicación web y asegurarnos de que es robusto.
- Dar al usuario la opción de logout en cada página de la aplicación.
- Configurar un periodo de inactividad, a partir del cual, desconecte la sesión del usuario por inactividad.
- No exponer ningún tipo de identificador de sesión valido al usuario.
- Requerir la contraseña anterior, cuando un usuario quiera cambiarla.
- Si utilizas un mecanismo para recordar la contraseña, lo mejor es enviársela al correo que tenga asociado dicho usuario, y añadir mecanismos de "captchas" en los formularios.
- Poner un tiempo de espera entre dos o tres fallos de autentificación consecutivos.

Otros ejemplos:

- CWE Entry 287 on Improper Authentication
- CWE Entry 384 on Session Fixation

5.3 Secuencia de Comandos en Sitios Cruzados (XSS)

La secuencia de comandos en sitios cruzados o Cross-Site-Scripting (XSS) es una de las vulnerabilidades más conocidas y comunes, se puede ver como un tipo de inyección de código, pero en este caso, se centra en atacar a la parte del cliente de la aplicación web.



Cualquier persona capaz de enviar información no fiable al sistema, incluyendo usuarios internos, externos y administradores.	Los ataques se realizan enviando cadenas de texto con la intención de vulnerar al intérprete del navegador.	La vulnerabilidad aparece cuando la aplicación incluye información enviada por un usuario sin previamente haber sido validada y verificada.	El atacante puede llegar a ejecutar scripts en el navegador de la víctima, deformar páginas web, incluir contenido no apropiado, redirigir a los usuarios, inyectar Shell remotas, etc.	Depende del valor que la organización o empresa de a los clientes y su información. Considerar también perdida de reputación y exposición de información privada.

Tabla 3. PoC Secuencia de Comandos en Sitios Cruzados (XSS)

Entornos Afectados: Cualquier aplicación web puede ser vulnerable a XSS.

PoC: Para esta prueba de concepto utilizare la maquina virtual de "Damn Vulnerable Web Application (DVWA)" que viene dentro del conjunto de maquinas virtuales de OWASP. Tenemos un Blog en el cual los usuarios pueden introducir sus post.

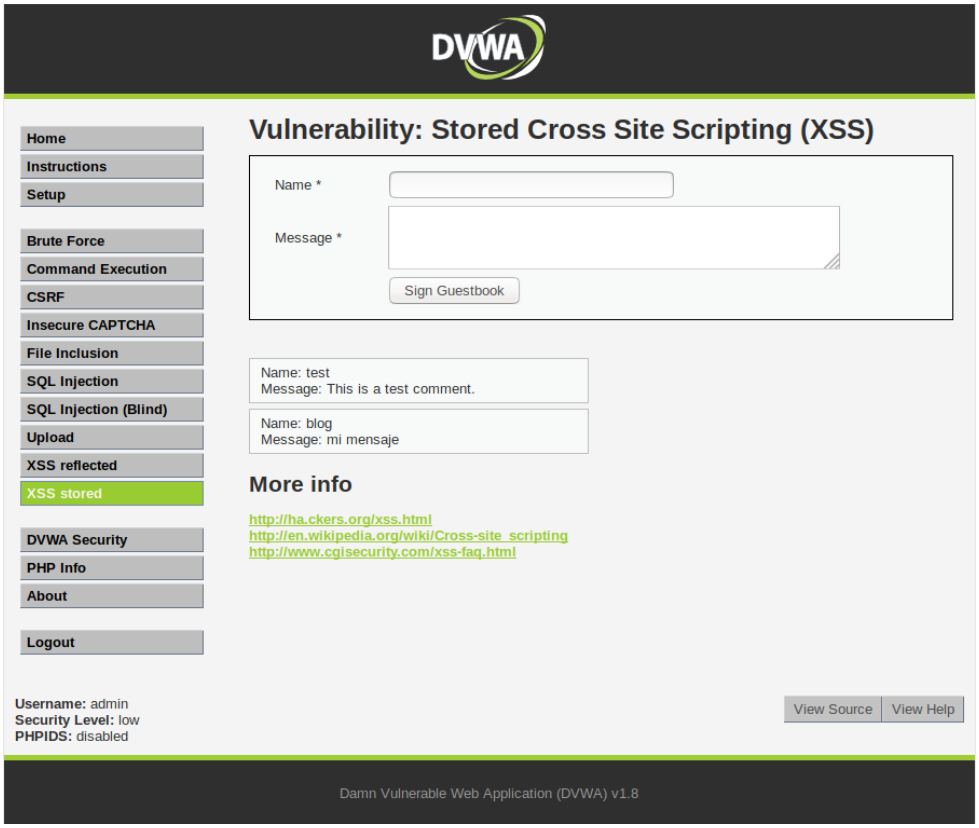
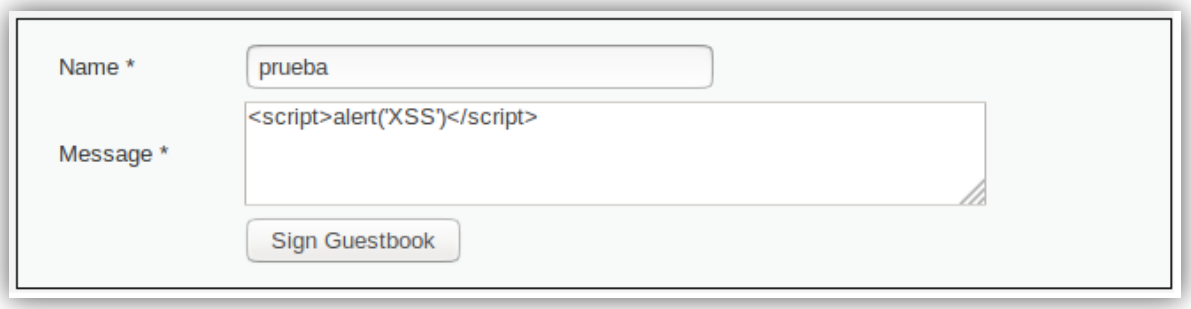
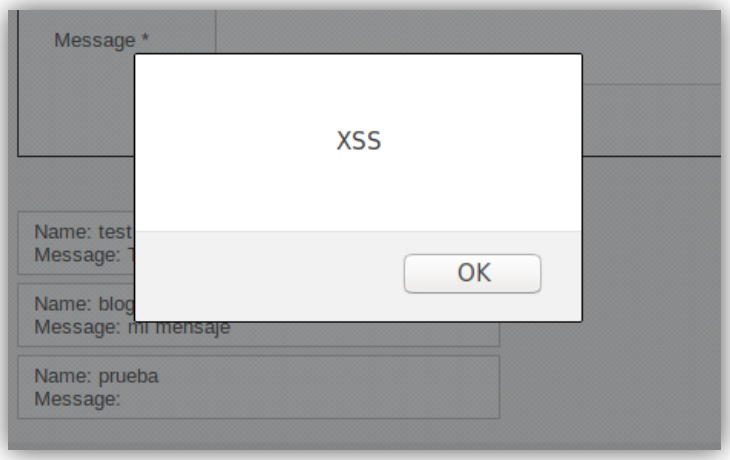


Figura 28.1 PoC XSS Stored

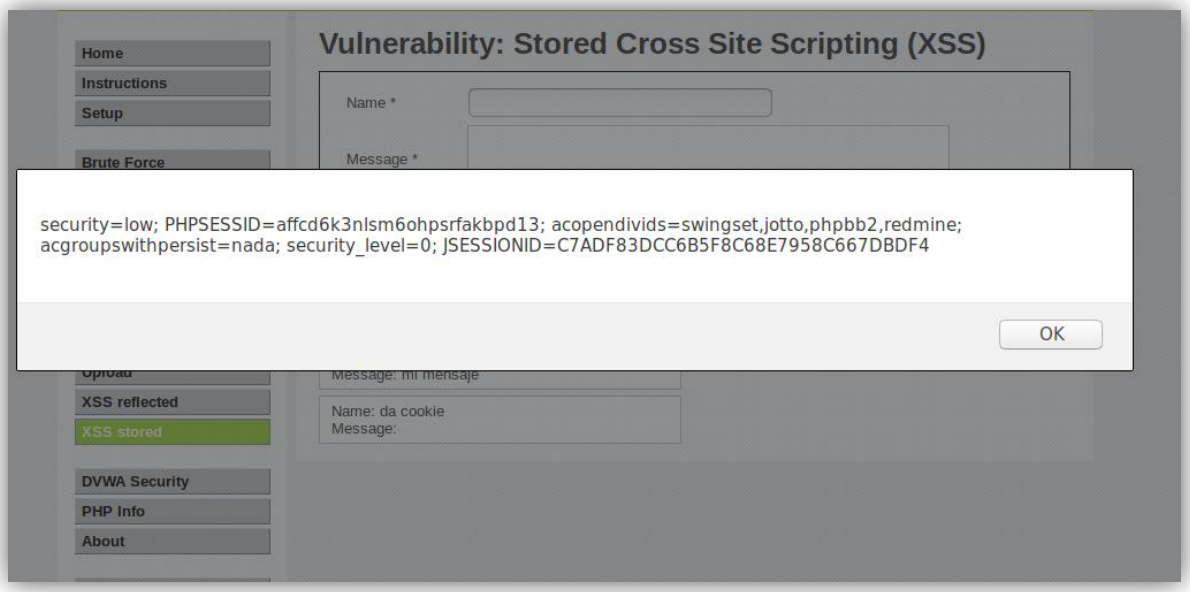
La manera más fácil de saber si la aplicación que tenemos delante es vulnerable a ataques de XSS, es introduciendo código Javascript, por ejemplo, un simple “Alert()”.



A veces implementan un simple mecanismo de escape de caracteres para que no interprete los “<”, “>”, “/”, que fácilmente se puede saltar poniendo el código en HTML para dichos caracteres, por ejemplo si ponemos “<” sería equivalente a “<”. Y si tenemos suerte y la pagina es vulnerable a XSS, cuando le demos a enviar el mensaje, y la pagina se recargue, nos mostrara un pop-up con el mensaje que hayamos introducido.



Ahora que sabemos que es vulnerable, podemos introducir cualquier tipo de Script en Javascript, utilizando etiquetas HTML. El ejemplo típico es obtener la cookie de nuestra sesión, lo haríamos con la siguiente instrucción “document.cookie”.



También podríamos redirigir a todos los usuarios, lo único que tenemos que escribir en el cuerpo del mensaje, sería lo siguiente: “<script> window.location.href = “http://www.elhacker.net” </script>”. O Incluso podríamos cargar scripts más complejos almacenados en otras páginas web, con etiquetas como <IFRAME> o <IMG>, en las cuales se permite apuntar hacia otras páginas para que carguen el contenido. Y así, cualquier usuario que visite ese blog, ejecutara nuestro script almacenado en el servidor.

Verificación de Seguridad: la misión en este punto es verificar que todos los parámetros que recibimos desde la parte del cliente son validados y/o codificados antes de ser incluidos en las paginas HTML, para evitar que el interprete del navegador, ejecute instrucciones que no sean código de los desarrolladores.

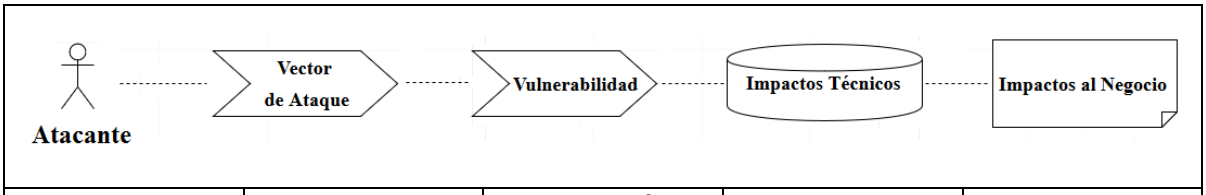
Protección: Validar todas las entradas del usuario y codificar apropiadamente las salidas que se muestran al usuario.

Otros ejemplos:

- CWE Entry 79 on Cross-Site Scripting

5.4 Referencia directa insegura a objetos

La vulnerabilidad ocurre cuando el desarrollador expone una referencia de un objeto interno de la aplicación, como un fichero, un directorio, una base de datos, una URL, etc. El atacante puede manipular la referencia a dicho objeto para acceder a otros objetos sin autorización.



Cualquier usuario que tenga acceso parcial a la información del sistema.	El atacante simplemente cambia el parámetro del objeto referenciado para apuntar hacia otro al cual, a priori, no debería tener acceso.	La aplicación no tiene un control de acceso sobre qué usuarios acceden a que recursos.	Puede comprometer todos los recursos referenciados.	Perdida de reputación, exposición de información.

Tabla 4. PoC Referencia directa insegura a objetos.

Entornos Afectados: Cualquier aplicación que implemente referencia a objetos.

PoC: Para esta prueba de concepto utilizare la máquina de Mutillidae II, parte del proyecto de OWASP. Tenemos una página en la cual podemos obtener cierta información en función del desplegable principal, pero lo que realmente nos interesa es la URL, en como accede a dicho sitio.

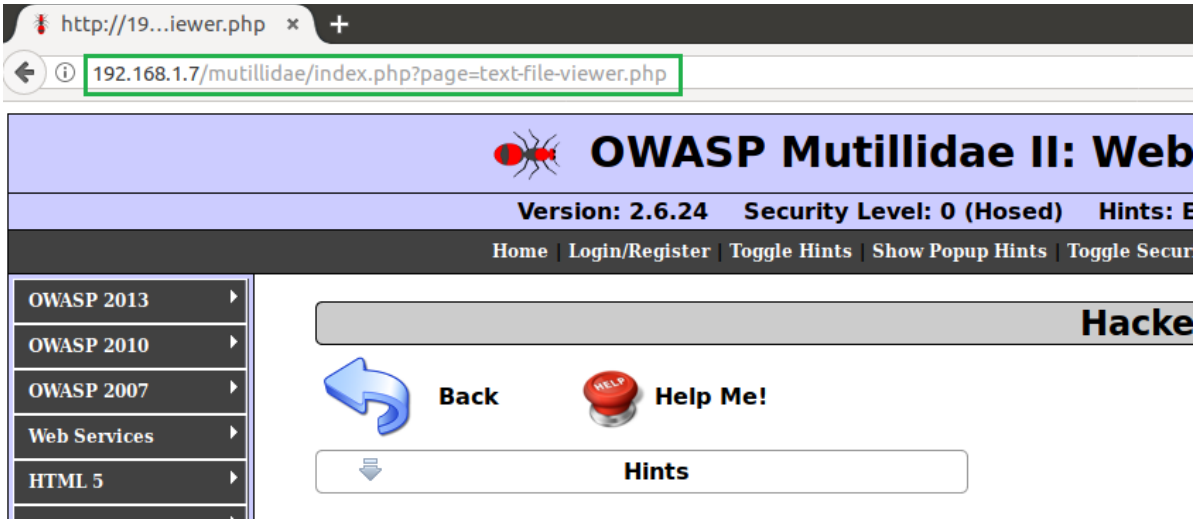
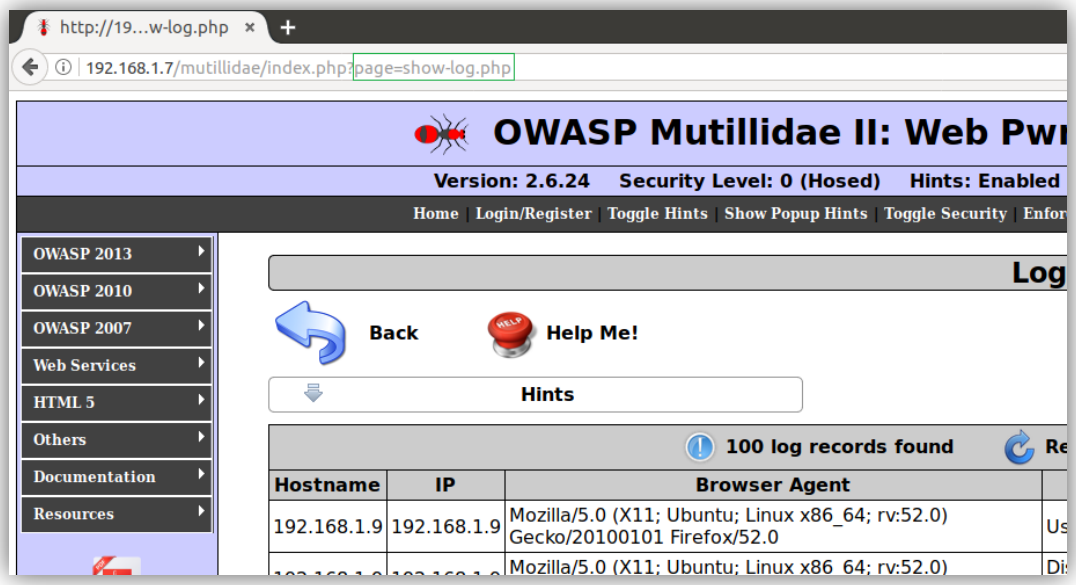


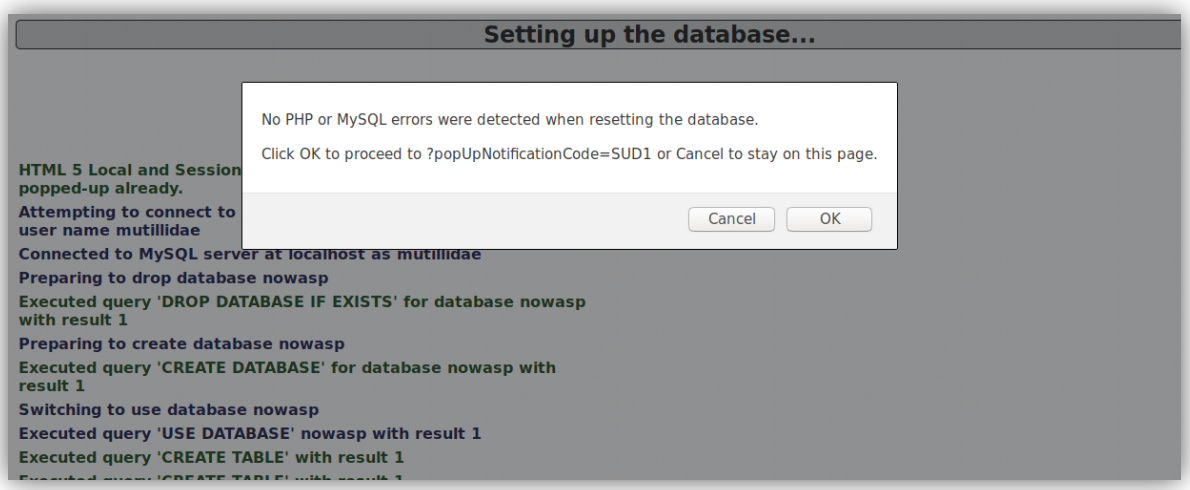
Figura 29.1 PoC Insecure Direct Object References

Podemos observar como accede a este Php, incluyendo en la URL el parámetro del propio nombre del fichero, justo después del "=". Además sabemos que, existen ficheros que sirven para configurar una aplicación web, en el propio directorio, archivos como, PHP.INI para la configuración de apache o Robots.txt para "evadir" a los crawler, y otros muchos en formato JSON o XML, que utilizan las aplicaciones para recibir una configuración concreta por parte de los web máster o desarrolladores.

Hay varias formas de atacar esta vulnerabilidad, podemos intentar adivinar en qué nivel de la raíz del directorio nos encontramos (jugando con él ".." para subir de directorio) a la vez que probamos distintos nombres de posibles ficheros, o podríamos utilizar un crawler, como el que llevan integrados muchas herramientas para saber exactamente cuál es el árbol de directorios y ver si podemos visualizar recursos desde dicha página a las cuales en principio, un determinado usuario no debería acceder.



En la imagen anterior vemos como cambiando el parámetro, accederíamos al registro de los log del servidor.



Incluso si accedemos al script llamado “set-up-database.php” podemos restablecer la base de datos a su configuración inicial.

Verificación de Seguridad: Intentar evitar en la manera de lo posible cualquier referencia directa a objetos, y en caso de tener que usarlas, implementar algún mecanismo de control de acceso, para que un atacante no pueda utilizarla para acceder a otro tipo de información sensible.

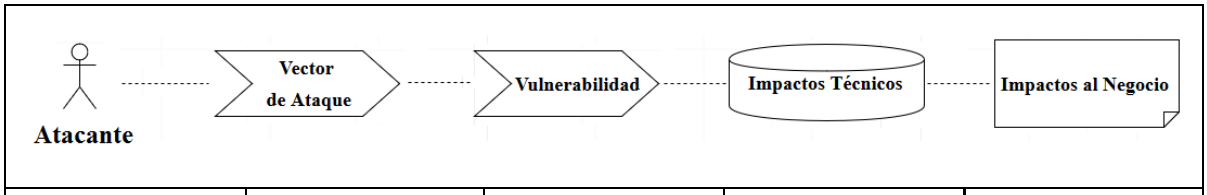
Protección: No exponer los parámetros al usuario directamente, y utilizar referencias mapeadas internas por los propios desarrolladores con una serie de parámetros pre-configurados.

Otros ejemplos:

- CWE Entry 639 on Insecure Direct Object References
- CWE Entry 22 on Path Traversal

5.5 Configuración de Seguridad Incorrecta

Esta es bastante auto-explicativa, intentare describirla brevemente, ya que tampoco se trata de hacer una guía con todo lo que se puede configurar que afecte a una aplicación web (para esto se puede seguir la guía para desarrolladores de OWASP) y no es el objetivo de este capítulo.



Cualquier atacante externo anónimo al igual que usuarios validos que puedan comprometer el sistema.	El atacante accede al sistema mediante cuentas de usuario por defecto, fallos sin parchear, recursos sin control de acceso, etc.	La vulnerabilidad puede aparecer en cualquier nivel de la aplicación, a nivel de plataforma, servidor web, base de datos, framework, etc.	Acceso no autorizado a recursos o funcionalidades del sistema. Puede llegar a comprometer todo el sistema.	Los datos de clientes o empleados pueden ser robados o modificados.

Tabla 5. PoC Configuración de Seguridad Incorrecta.

Entornos Afectados: Cualquier aplicación web.

PoC: Algunos servidores web te permiten realizar búsquedas de directorios por defecto, directamente te listan el árbol de directorios y ficheros que cuelgan del índex, lo normal es desactivar esta opción. Por ejemplo en apache, en el .htacces, podemos añadir la siguiente línea para desactivarlo:

"Options –Indexes"

Verificación de Seguridad: Lo ideal es que se junten tanto los desarrolladores junto con los administradores del sistema, para verificar que las distintas capas están configuradas apropiadamente.

Protección: Desactivar todas las características innecesarias por defecto, borrar los usuarios y contraseñas por defecto o modificarlos, separación segura y efectiva entre

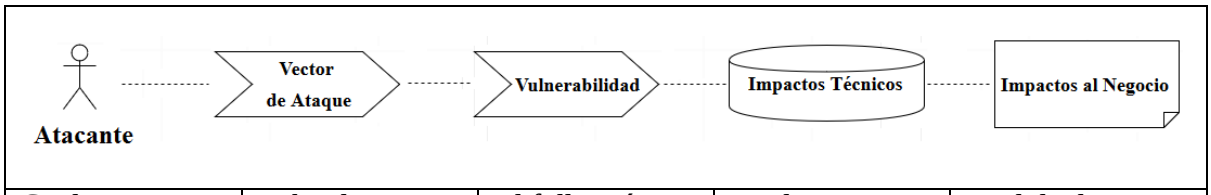
componentes de la aplicación, realizar escaneos y auditorias de seguridad para detectar fallos de configuración o parches viejos o con vulnerabilidades conocidas.

Otros ejemplos:

- CWE Entry 2 on Enviromental Security Flaws

5.6 Exposición de datos sensibles

Proteger la información sensible con algoritmos criptográficos es una parte muy importante de las aplicaciones web.



Cualquier persona que pueda acceder a información sensible de otros, ya sea almacenada, en tránsito o incluso en el navegador.	Robo de contraseñas en texto claro, ataques Man-in-The-Middle, o obteniendo información que no está correctamente cifrada.	El fallo más común es no utilizar ningún tipo de algoritmo de encriptación para la información sensible.	Puede comprometer toda la información que debería estar protegida.	Perdida de reputación, responsabilidades legales si es información muy sensible como, informes de salud, credenciales, tarjetas de crédito, etc.

Tabla 6. PoC Exposición de datos sensibles.

Entornos Afectados: Cualquier aplicación web.

PoC: Una prueba de concepto podría ser el no usar SSL para la comunicación con nuestra aplicación web, pero últimamente cada vez más todas o casi todas las aplicaciones web lo implementan (Algunas usando sus propios certificados).

Otra es utilizar algoritmos de cifrado débiles para la información sensible, para esta prueba de concepto utilizare la maquina virtual de OWASP Security Shepherd. Podemos imaginar que obtenemos una serie de caracteres almacenados de una cookie codificada o una contraseña almacenada en la base de datos cifrada, y obtenemos lo siguiente:

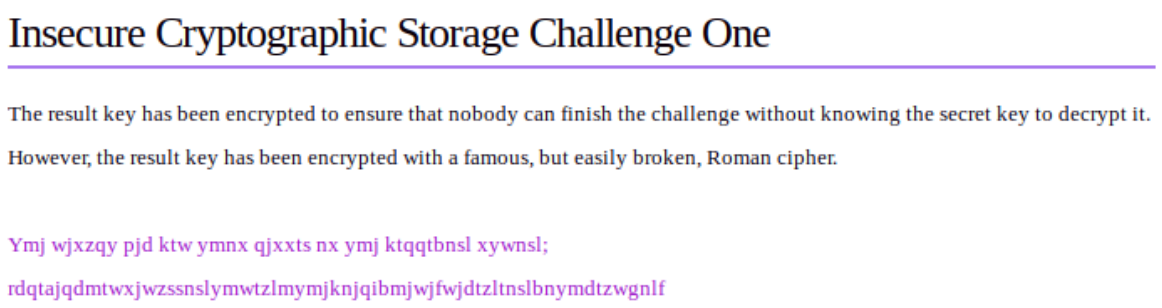
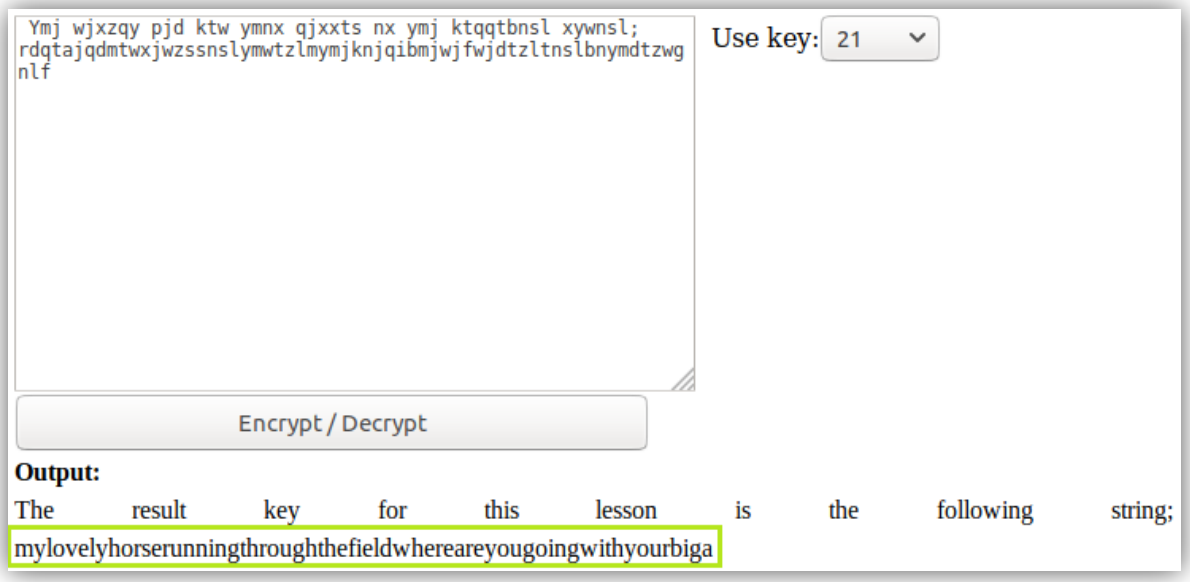


Figura 30.1 PoC Exposición de datos sensibles

El cual utiliza un algoritmo de cifrado bastante simple desplazando caracteres en función de un numero (Caesar Cipher), el cual para nada es seguro.



Verificación de Seguridad: La aplicación utiliza apropiadamente algoritmos de encriptación robustos.

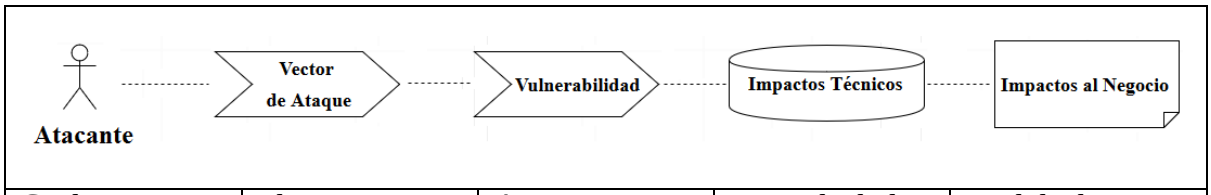
Protección: utilizar AES, RSA, SHA-256 o superior. No utilizar MD5, SHA1.

Otros ejemplos:

- CWE Entry 310 on Cryptographic Issues
- CWE Entry 312 on Cleartext Storage of Sensitive Information
- CWE Entry 326 on Weak Encryption

5.7 Inexistente Control de Acceso a nivel de funcionalidades

Esta vulnerabilidad está muy relacionada con la de implementar mecanismos de control de acceso a nivel de recurso, pero esta vez, también a nivel de funcionalidades.



Cualquier persona que pueda lanzar peticiones al servidor web.	El atacante puede acceder a funciones privadas que no están protegidas.	Aparece cuando no se controla el acceso a determinadas funciones o partes de la aplicación web.	Depende de las funciones que haya implementadas en la aplicación, si p.e. hay una función que reinicia la base de datos, esta podría verse comprometida.	Perdida de reputación, de dinero, de tiempo.

Tabla 7. PoC Inexistente Control de Acceso a nivel de funcionalidades.

Entornos Afectados: Cualquier aplicación web.

PoC: Algunas páginas pasan por parámetro a través de la URL (HTTP GET) la función a la cual invocan (www.mypagina.es/idle.php?action=go), y estas funciones se ejecutan sin mirar que usuario hace la petición.

Verificación de Seguridad: Comprobar que las URL no muestren exactamente a qué función se está llamando por debajo, así como comprobar que el servidor hace las autentificaciones y autorizaciones correspondientes antes de lanzar una función, etc.

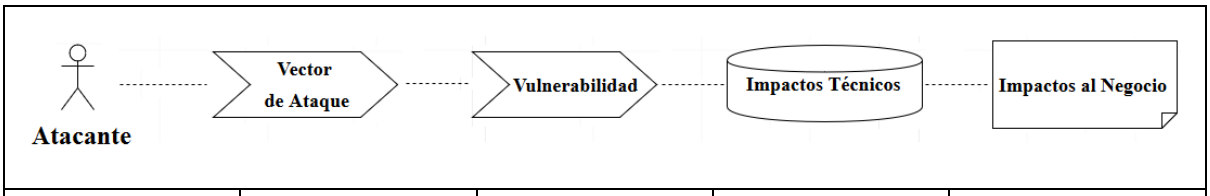
Protección: Diferenciar distintos roles en la aplicación y distinguir que funciones son accedidas por unos y cuales por otros.

Otros ejemplos:

- CWE Entry 285 on Improper Access Control (Authorization)

5.8 Falsificación de Peticiones en Sitios Cruzados (CSRF)

En esta vulnerabilidad el atacante fuerza a una víctima (previamente autentificada en el sistema) a enviar una petición a la aplicación web vulnerable, para ejecutar una acción que el atacante decida.



Cualquier persona que pueda realizar peticiones al servidor web a través de otros usuarios.	El atacante crea una petición HTTP falsa y consigue que la victima (autentificada) la ejecute.	La vulnerabilidad aparece cuando un atacante puede predecir todos los detalles de una acción concreta dentro de la aplicación web. Entonces crean peticiones falsificadas que son indistinguibles de las legítimas.	Los atacantes pueden modificar cualquier información a la que está autorizada la víctima, también puede acceder a funcionalidades asociadas a dicho usuario.	Considerar el valor de negocio relacionado con esos datos o funciones afectadas. Perdida de reputación.

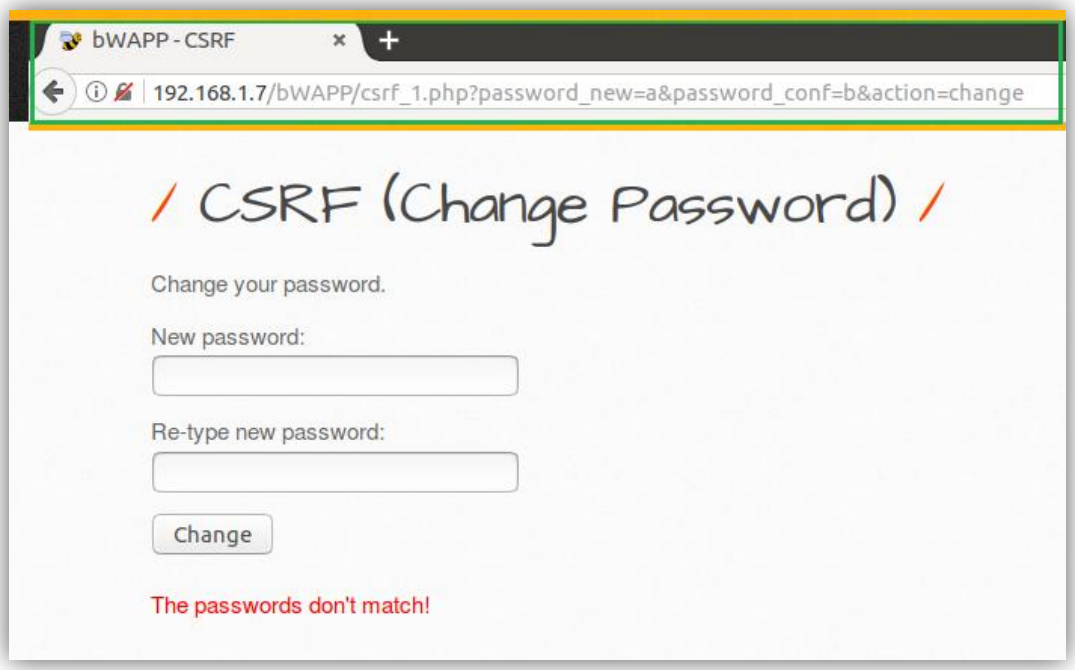
Tabla 8. PoC Falsificación de Peticiones en Sitios Cruzados (CSRF).

Entornos Afectados: Cualquier aplicación web.

PoC: Para esta prueba de concepto utilizare la maquina virtual de bWAPP, que ya he utilizado anteriormente, y que forma parte del proyecto de OWASP. Imaginemos que tenemos una aplicación web, con su panel de control y dentro de esta, su sección para cambiar la contraseña de tu cuenta con la que estas actualmente autentificado.

Figura 31.1 PoC CSRF

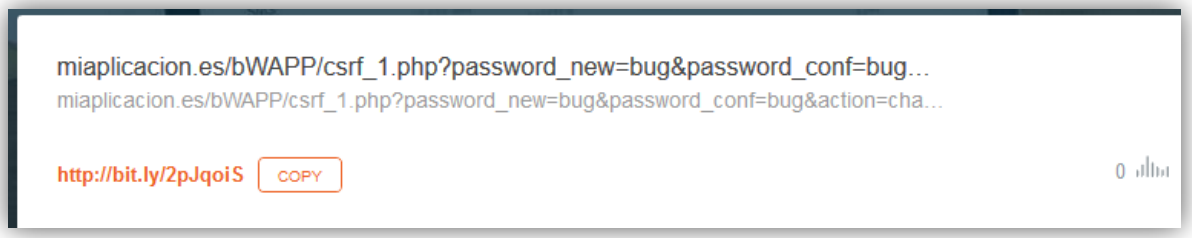
Si por ejemplo introducimos dos valores que no coincidan para ver cómo responde la aplicación, p.e. introducimos a, b y enviamos.



Podemos observar que tenemos un Php que cambia la contraseña del usuario que esta autentificado, que recibe tres parámetros por GET, password_new, password_conf y action=change. Podríamos construir una petición exactamente igual con los valores que queramos, p.e.:

http://miaplicacion.es/bWAPP/csrf_1.php?password_new=bug&password_conf=bug&action=change

Ahora solo tendríamos que ayudarnos de un poco de ingeniería social, para conseguir que alguien (una víctima que tenga una cuenta en la aplicación y este autentificada) visitara dicho enlace. Obviamente si le pasamos ese enlace a una persona, podría leer fácilmente los campos de password y sospechar, podemos ayudarnos de un ofuscador de código, o un acortador de enlaces. Incluso podríamos esconder (una vez ofuscado) el enlace en una etiqueta de HTML, como <IMG> o <FRAME>.



Como podéis ver realizar este ataque es relativamente sencillo, y suele apoyarse mucho en la ingeniería social y en otras vulnerabilidades, como XSS.

Verificación de Seguridad: Verificar que la aplicación genera y pide algún tipo de Token de autorización/sesión que no es automáticamente enviado por el navegador.

Protección: Utilizar algún Token personalizado, si puede ser encriptado o codificado propio de la aplicación, y que este se cree para cada usuario autentificado en el sistema, y cuando se trate de una acción muy importante volver a pedir credenciales o ocultar el Token en un campo oculto, etc. También es mejor evitar el uso de GET HTTP para estas acciones, pero no solo basta con el uso de POST HTTP.

Otros ejemplos:

- CWE Entry 352 on CSRF

5.9 Uso de Componentes con Vulnerabilidades Conocidas

Esta vulnerabilidad es bastante auto-descriptiva.

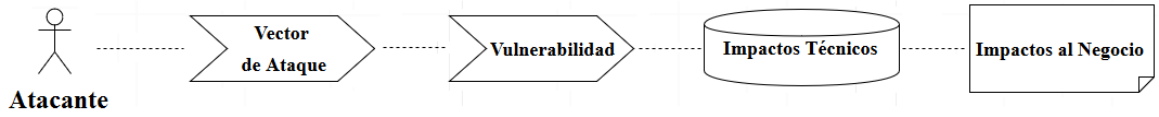
				
Cualquier persona que pueda utilizar una herramienta automatizada para explotar algún tipo de vulnerabilidad en componentes vulnerables conocidos.	El atacante identifica un componente con una vulnerabilidad, configura el Exploit y lanza el ataque.	Propia del componente, es importante saber las versiones de los componentes que estamos empleando, así como si son o no vulnerables.	Todo el rango, dependiendo del componente y la vulnerabilidad, desde el impacto mínimo, hasta hacerse con todo el sistema y los datos.	Depende de la vulnerabilidad, puede ser trivial o puede ser muy comprometido.

Tabla 9. PoC Uso de Componentes con Vulnerabilidades Conocidas.

Entornos Afectados: Cualquier aplicación web que utilice componentes.

PoC: algunos componentes como Apache CXF Authentication o Spring Framework de Java. A nivel de navegador, Adobe Flash Player, etc.

Verificación de Seguridad: Revisar todos los componentes y sus versiones, comprobar que tiene todos los últimos parches de seguridad instalados.

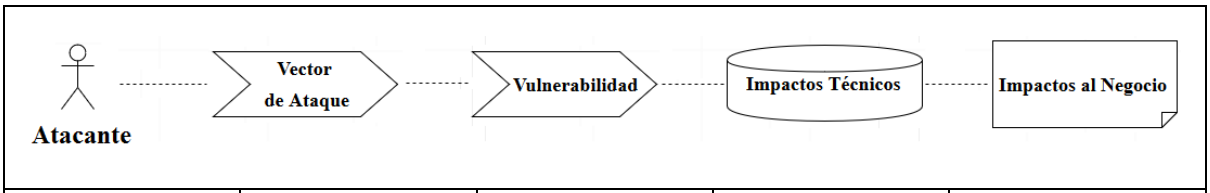
Protección: Buscar si los componentes que usamos aparecen en alguna de las base de datos publicas en las cuales se reportan las vulnerabilidades, tales como, CVE (Common Vulnerabilities and Exposures) y NVD (National Vulnerability Database).

Otros ejemplos:

- MITRE Common Vulnerabilities and Exposures

5.10 Redirecciones y reenvíos no válidos

La idea principal de esta vulnerabilidad, es que el atacante hace uso de los re-direccionamientos de una aplicación para engañar a otro usuario.



Cualquier persona que pueda engañar a un usuario para que realice una petición en una aplicación web.	El atacante crea un enlace personalizado de una aplicación y engaña al usuario para que lo visite.	Aparece cuando una aplicación redirige a otras páginas y la página objetivo es un parámetro que puede ser modificado por cualquiera.	Normalmente las redirecciones intentan instalar malware o desvelar información sensible sobre la víctima.	Valor de la información, perdida de reputación, pudiendo ser más grave en función del malware.

Tabla 10. PoC Redirecciones y reenvíos no válidos.

Entornos Afectados: Cualquier aplicación que permita re-direccionar o reenviar sin ningún tipo de control.

PoC: Imaginar que tenemos un Php que recibe por el método HTTP GET, el parámetro de la URL a la que queremos re-direccionar al usuario, un atacante podría modificar esa URL y apuntar a la victima hacia una página que cargue algún Javascript malicioso o cualquier otro script que el atacante decida.

Verificación de Seguridad: Revisar el código en busca de redirecciones y reenvíos a través de parámetros, si la URL se recibe a través de un parámetro y este no está en una lista blanca, para poder controlar hacia donde se redirige a los usuarios.

Protección: Asegurarnos que la URL objetivo de la redirecciones es una URL valida y la da un usuario autentificado y autorizado en el sistema.

Otros ejemplos:

- CWE Entry 601 on Open Redirects

CAPÍTULO 6. INTEGRACIÓN PLUG-IN A BURP SUITE

En este capítulo contare como se ha llevado a cabo el desarrollo del Plug-in L-Tools para el proxy web de Burp Suite, cuales son las funcionalidades que aporta y para que tareas se pueden usar, pasando por la API publica del propio proxy web, al final en los anexos expondré el código como tal en Python.

6.1 API de Burp Suite

La API de Burp Suite está muy bien documentada en su página oficial [55], también se puede descargar directamente desde la aplicación de Burp Suite en formato JAVADOC. Personalmente recomiendo la versión web, ya que está bien estructurada con enlaces a todo lo que requiere una determinada clase (interfaz) en cada momento.

Para empezar cualquier extensión de Burp Suite, hay una clase que siempre debe ser declarada para que se cargue como extensión, denominada *BurpExtender* y esta debe implementar la interfaz de *IBurpExtender*. Todas las extensiones deben implementar dicha interfaz, la cual es invocada cada vez que Burp Suite carga una extensión.

Dicha interfaz (*IBurpExtender*) obliga a implementar un método llamado *registerExtenderCallbacks*(*IBurpExtenderCallbacks* callbacks). El objeto *IBurpExtenderCallbacks* hace de mediador entre nuestra extensión y el resto de interfaces de Burp. Los métodos principales y características de Burp se pueden activar haciendo llamadas desde el objeto callbacks.

Se puede decir que hay dos objetos que dan las utilidades necesarias para el resto de interfaces, uno es el objeto de callbacks, este nos permite registrar componentes o funcionalidades de nuestra extensión, añadir pestañas nuevas, añadir reportes personalizados al escáner, lanzar determinadas funciones como escanear un host, registrar escuchadores de eventos, etc.

Un método del objeto callbacks es *getHelpers()*, el cual nos devuelve el segundo objeto que proporciona otra serie de utilidades. Como pueden ser analizar una petición o respuesta, añadir parámetros a una petición, construir una petición HTTP, etc.

6.2 ¿Por qué necesitaría una extensión personalizada?

Básicamente lo que nos permite la API de Burp Suite, es interactuar con sus otros módulos a través de sus propios métodos, por lo tanto, podemos crear extensiones que nos permitan algunas de las siguientes tareas:

- Decodificar una codificación/serialización personalizada:
  Objetos java serializados, SAML, JSON, Base64, cualquier tipo de cadena de caracteres que este codificado, serializado o cifrado de alguna manera, y nosotros podamos revertir dicho proceso para conseguir la cadena de texto en plano y legible.

- Analizar y modificar peticiones/respuestas HTTP al vuelo:
  Ya que disponemos de la funcionalidad de proxy web del propio Burp Suite, este nos da acceso a todas las peticiones que pasan a través del mismo, y las cuales podemos modificar como queramos, para algún fin concreto que deseemos.

- Proporcionar una nueva vista dentro de la aplicación:
  Podemos remarcar texto, crear nuevas pestañas dentro, insertar contenido organizado de otra forma a la original, recopilar las cabeceras de las peticiones y respuestas y mostrarlas, etc.

- Automatizar alguna tarea manual:
  Podemos lanzar un escáner a una petición HTTP construida por nosotros, insertar puntos de inserción dentro de una determinada petición HTTP que va a ser escaneada, enviar objetos desde nuestra extensión a cualquier otra herramienta de Burp, etc.

6.3 Interfaz de usuario de L-Tools

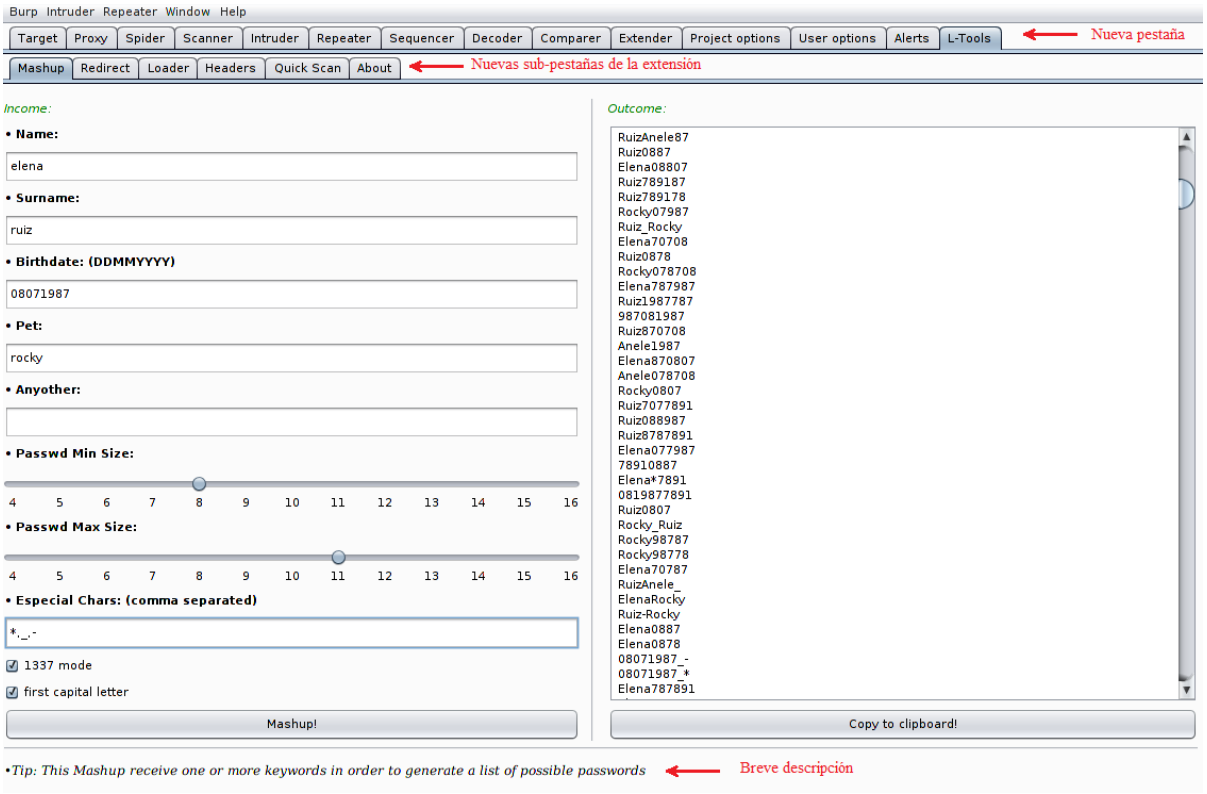


Figura 32. L-Tools – Mashup UI

Lo primero que podemos observar al cargar el Plug-In es una nueva pestaña que se ha añadido a la interfaz de Burp Suite arriba a la derecha. Dentro de esta nueva pestaña, tenemos nuevas sub-pestañas, siguiendo la filosofía del propio Burp, cada una de estas sub-pestañas será una nueva utilidad desarrollada para que sirva de ayuda a la posible labor de la comunidad que utiliza la herramienta de Burp Suite.

La primera vista que tenemos de la extensión es un Mashup, cuya finalidad es la de generar un diccionario de contraseñas en función de una serie de palabras clave y parámetros, aunque estén etiquetadas las palabras clave, como nombre, apellido, fecha de nacimiento, realmente se puede introducir cualquier tipo de palabra clave que consideremos. Además, se puede configurar la longitud mínima y máxima de las contraseñas de salida, también si queremos introducir algún carácter especial en ellas.

Una vez tenemos nuestro diccionario generado, tenemos un botón para copiar el diccionario en el portapapeles, y así poder pegarlo en el sitio que consideremos oportuno. Asimismo al final de cada utilidad tenemos un texto de breve descripción o ayuda.

Figura 33. L-Tools – Redirect UI

En principio esta utilidad, fue una de las primeras ideas que se me pasaron por la cabeza, ya que al ser un proxy web, y gracias a que la API permite escuchar toda las peticiones/respuestas que pasan a través del proxy web, se me ocurrió hacer un Redirect muy sencillo, con el cual podemos "bloquear" el acceso a ciertas páginas web, ya sea si estamos en una empresa y no queremos que nuestros trabajadores entren a ciertas paginas, o como control parental para los niños, etc.

La interfaz es muy sencilla e intuitiva, tenemos un origen y destino y un botón que hace de interruptor el cual indica si esta funcionalidad esta activada o desactivada. Puse solo dos filas, ya que, como tampoco suele ser una utilidad muy requerida por un Pentester, pero podrían haber sido más de dos, incluso trabajar con la petición que llega desde el exterior, en lugar de hacerlo al revés.

Por ejemplo, podríamos hacer que si nos llega una petición por un puerto distinto al 443 (HTTPS) u otro que consideráramos, se bloqueara el acceso de dicha petición, y en cierto modo, solo obligaríamos a que entren peticiones cifradas o seguras mediante el protocolo HTTPS.

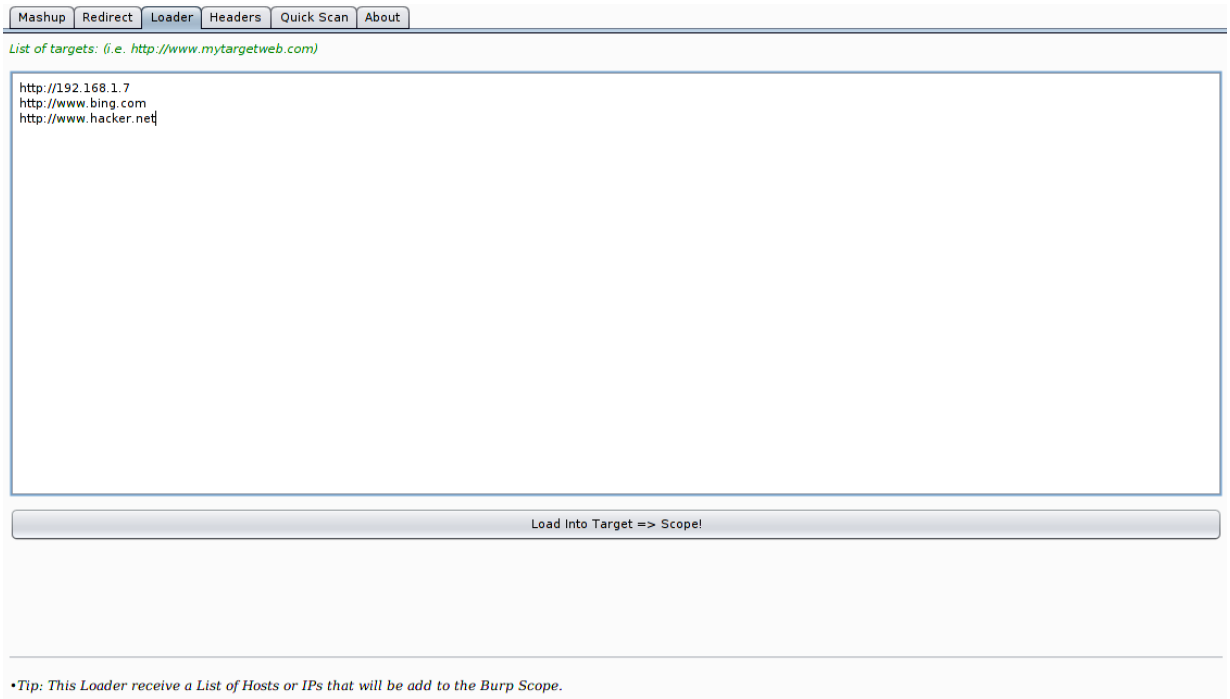


Figura 34. L-Tools – Loader UI

La única forma de trabajar con Burp Suite y sus distintas herramientas es a través de menús contextuales, cuando tenemos el árbol de directorios en "Target/Site Map" ya sea porque hemos estado navegando o gracias al crawler, la única forma que tenemos de marcar nuestras páginas como "objetivos" para realizar un análisis del sitio y buscar vulnerabilidades, es con el menú contextual encima de dicho objeto y seleccionando la opción de "mandar a Scope" o "mandar a Scan", etc.

Otra forma, sería ir a la propia pestaña de "Target/Scope" e ir añadiendo los "objetivos" uno a uno, de forma manual. Por esta razón se me ocurrió añadir esta funcionalidad, la cual nos permite añadir un objetivo o una lista de páginas o sitios objetivos, que estén siendo objeto de nuestra prueba de Pentesting o análisis de vulnerabilidades, etc.

La interfaz es muy minimalista, tenemos un tex tarea en el cual introducimos la lista de sitios y un botón con el cual los mandamos a "Target/Scope". Es importante que cada uno de los objetivos sea una URL bien formada, esto es, que tenga el protocolo (HTTP o HTTPS), la URI y el puerto asociado por defecto o el que se le indique.

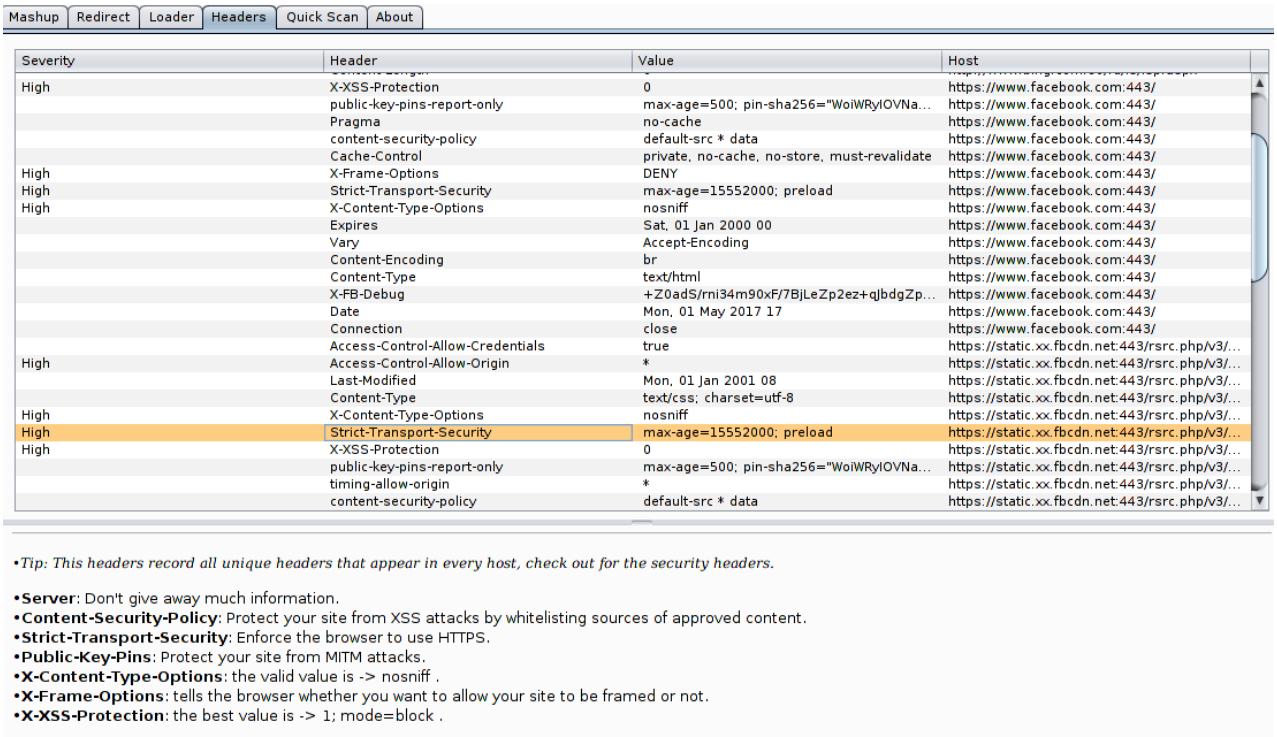


Figura 35. L-Tools – Headers UI

Sabemos que el protocolo HTTP y su versión segura HTTPS, utilizan cabeceras para enviar y recibir información entre el cliente y el servidor. Configurar adecuadamente las cabeceras que el servidor va a enviar, suele ser fácil y normalmente no requiere cambios en el código. Una vez configurados restringen a los navegadores modernos de ciertas vulnerabilidades, es una forma muy fácil de prevenir ciertos tipos de vulnerabilidades.

La funcionalidad de la utilidad Headers, es recolectar todas las cabeceras únicas de las respuestas que pasan a través del proxy web, mientras que el usuario o Pentester esta navegando por el sitio web objetivo, con la finalidad de marcar mediante la columna de "severity" cuales son aquellas cabeceras que hay que tener en cuenta y son relevantes para la seguridad.

Las cabeceras más importantes desde el punto de vista de la seguridad web son las siguientes:

- **HTTP Strict Transport Security (HSTS):** Le indica al cliente (normalmente un navegador web) que solo debe interactuar con el servidor mediante el protocolo seguro de HTTPS.
- **Public Key Pinning Extension for HTTP (HPKP)**: Evita de ataques MITM.
- **X-Frame-Options**: Le comunica al navegador, cuando un sitio web puede ser llamado o no desde un "<FRAME>".

- **X-XSS-Protection**: Te protege contra ataques XSS. El valor recomendado es *1; mode=block*.
- **X-Content-Type-Options**: Previene que el navegador no interprete otros ficheros que no sean los declarados en el Content. Solo tiene el valor valido *nosniff*.
- **Content-Security-Policy**: Proporciona una lista blanca de recursos con contenidos fiables o aprobados. Previene ataques XSS.
- **X-Permitted-Cross-Domain-Policies**: Le dice al cliente si el servidor permite encapsular contenido del sitio web en otros tipos de documentos como Adobe Flash o PDF (no solo nos limitamos a esos dos).
- **Referrer-Policy**: Le dice al cliente que información de referencia debe ser enviada en las peticiones realizadas [56].

A parte de estas hay otras cabeceras como *Server* o *X-Powered-By*, que suelen dar información sobre el servidor o el cliente, que pueden ser usadas por un atacante, es recomendable desactivar estas cabeceras las cuales vienen activadas por defecto.

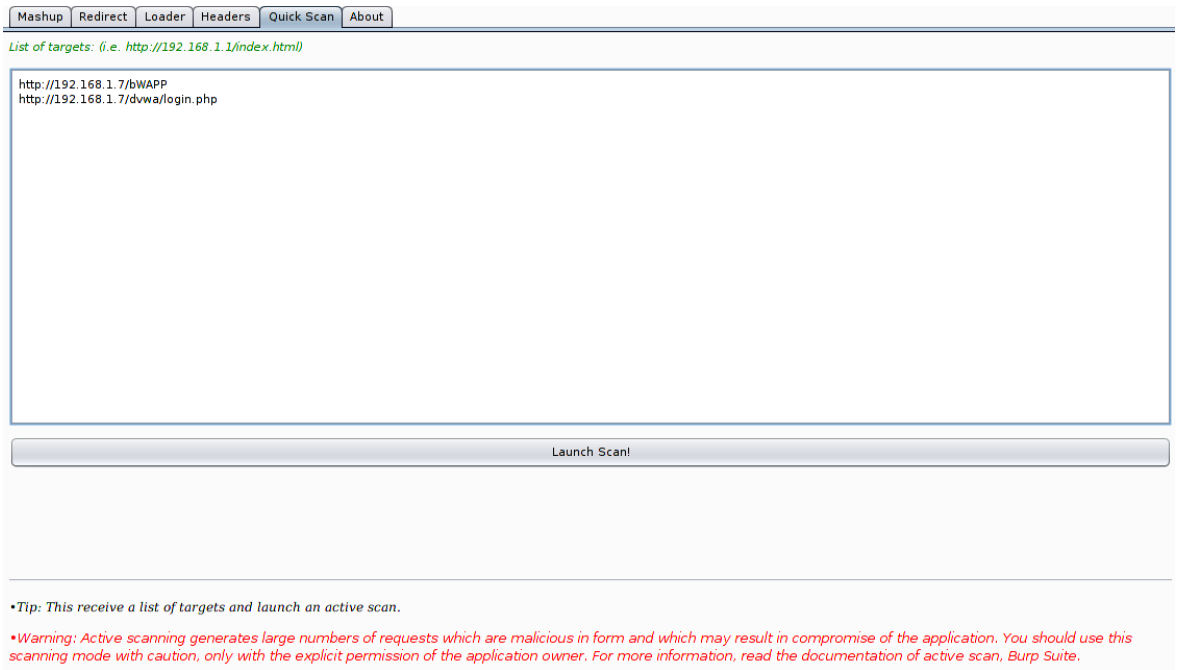


Figura 36. L-Tools – Quick Scan UI

Como comente anteriormente, Burp Suite trabaja a base de menús contextuales, lo mismo para lanzar un escáner de vulnerabilidades de una determinada pagina o sitio web. Así, y ya que la API lo permite, se me ocurrió implementar un "Quick Scan", en el cual podemos lanzar un escáner de un objetivo o una lista de objetivos a la vez.

Es importante remarcar que esta funcionalidad de “Active Scan”, suele basarse en la repetición de distintas peticiones “intrusivas”, por lo cual, hay que usarla solo contra sitios o aplicaciones web en las cuales tengamos el permiso y acuerdo mutuo con el propietario de dicha aplicación.

La interfaz es muy simple y mínima, una vez es lanzado el escáner, podemos obtener información referente a las vulnerabilidades que se va encontrando en la propia pestaña de “Scanner” de Burp Suite.

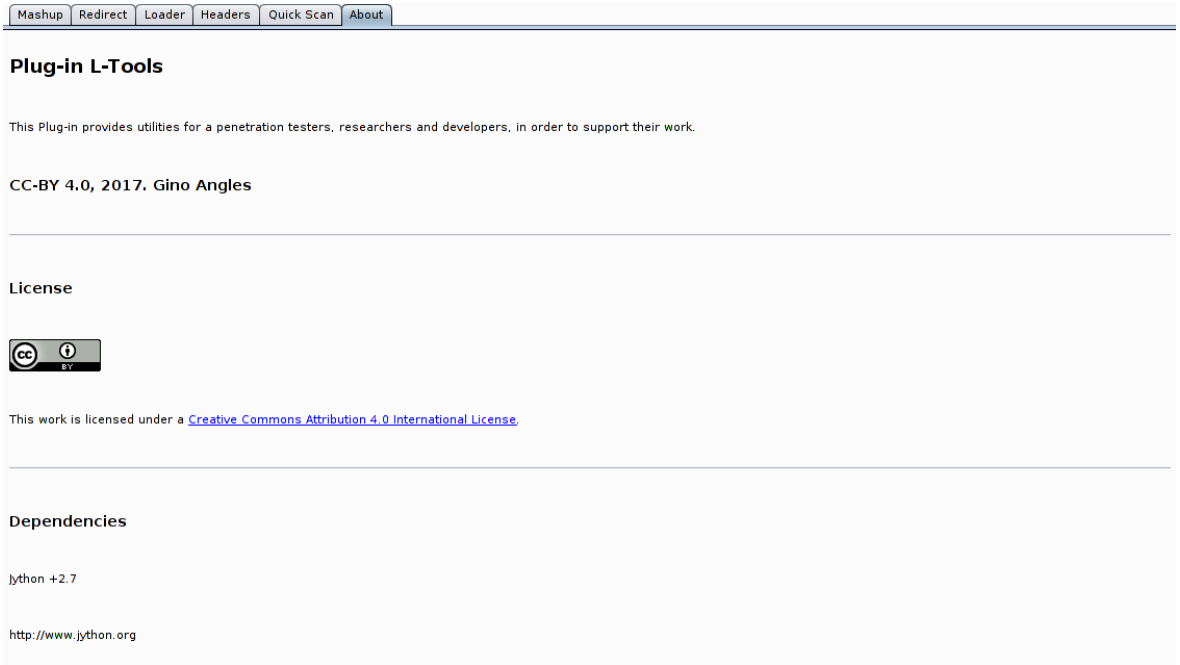


Figura 37. L-Tools About UI

Por último, he añadido una pestaña de “About”, en la cual hago una breve descripción del Plug-In, hago mención de la licencia Creative Commons CC-BY y las dependencias del Plug-In.

En todas las interfaces se nombran las etiquetas y demás sustantivos y ayudas en ingles, ya que es el idioma más internacional y casi todos, por no decir todos los demás Plug-In de la BApp del propio Burp Suite, están en ingles.

6.4 Implementación

Para contar esta sección, voy a empezar hablando de las decisiones de diseño, siguiendo con las versiones de software utilizado, y comentar muy por encima la estructura del código, el cual estará al final en uno de los anexos.

Decisiones de diseño:

Una de las primeras decisiones de diseño que tome, fue la de evitar toda dependencia con librerías externas o terciarias, y a ser posible, que la extensión funcionara tanto en la versión gratuita de Burp Suite, como en la profesional.

Al evitar toda dependencia de librerías externas, sabes que la extensión que desarrolles va a estar funcional siempre, a no ser que hagan un gran cambio en la API de Burp Suite (poco probable), para lo cual habría que estudiar si merece la pena mantener o no la extensión.

La decisión de que fuera funcional en ambas versiones (gratuita y pro), le daría más versatilidad a la hora de ser utilizada o requerida por la comunidad, así que, le daría mayor audiencia, por lo que, era algo a tener en cuenta.

El siguiente paso era estudiar la API y ver que funcionalidad me proporcionaba con la cual poder trabajar, una vez aprendida más o menos la API, debía pensar en posibles ideas de extensiones que pudieran ser interesantes o útiles. Se puede decir que lo más difícil de la extensión fue el recopilar ideas de posibles utilidades o funcionalidades añadidas que se pudieran desarrollar y que estas funcionalidades fueran lo más genéricas posibles.

Como sabéis la API de Burp Suite permite desarrollar código para sus extensiones en Java (el lenguaje en el cual está escrito el propio Burp), Ruby y Python. Yo decidí la opción de utilizar Python para empezar a desarrollar la extensión.

Para poder utilizar un entorno de Python para desarrollar la extensión, hace falta Jython, el cual es un intérprete de Python sobre la maquina virtual de Java. Lo único que tenemos que hacer es descargarnos Jython de la página oficial, y cargar dicho Jython.jar en una de las pestañas de Burp Suite, una vez hecho esto, podemos cargar ficheros ".py" en Burp Suite.

La principal ventaja de utilizar Jython, es que podemos escribir código al estilo de Python el cual es bastante legible y sencillo, podemos utilizar cualquier librería de Python que tengamos instalada, y además, podemos utilizar cualquier librería de Java que tengamos instalada.

De este modo, tenemos toda la lógica de negocio de la aplicación escrita puramente como Python, mientras que para la interfaz grafica utilizo librerías de Java.Swing y Java, para darle continuidad al estilo de la propia interfaz de Burp Suite.

Una desventaja de usar Jython, es la manera en la que genera las clases de Java dinámicamente, salta un error conocido, cuando cargamos muchas extensiones en Python o cuando cargas y recargas una misma extensión en Python. Es un error de memoria principal, se ve que por algún motivo no limpia bien la memoria cuando realiza este proceso, de modo que salta un error de espacio de memoria.

El error se puede solucionar asignando más memoria principal cuando iniciamos el proceso de Burp. Sería algo como: "*Java –XX:MaxPermSize=1G –jar Burp.jar*".

Al utilizar Python, decidí utilizar un editor de texto en lugar de algún IDE, cuando utilizaba la maquina virtual de Ubuntu, escribía código con el propio GEdit, cuando estaba en Win10 utilizaba Sublime Text, llegue a instalarme Atom Text editor, que es muy similar a Sublime Text, pero tampoco vi que me hiciera mucha falta, así que se puede decir que todo el desarrollo de código lo hice entre Sublime Text y GEdit.

Ya que todo el código que necesitaba estaba en la API y disponía de la versión web de la misma, realmente no sentí la necesidad de utilizar un IDE que me ayudara con autocompletado o cosas del estilo. Si hubiera decidido realizar el desarrollo en Java, probablemente hubiera utilizado IntelliJ IDEA, el cual te permite dibujar las interfaces Java simplemente arrastrando los componentes y moviéndolos con el ratón, muy diferente a hacerlo todo desde líneas de código.

Por último, y no menos importante, se me ocurrió la idea de asignarle a la extensión una licencia de Creative Commons, más concretamente la de CC-BY Creative Commons Attribution 4.0 International License. Con la filosofía de poder compartir código, siempre y cuando se haga una mención de su Autor.

Versiones utilizadas (System Setup):

Software	**Versión**
Burp Suite	1.7.12
Java	1.8.0_121 (OpenJDK)
Python	2.7.12+
Jython	2.7.0
SublimeText	3, Build 3126
OS (Linux VM)	Ubuntu 16.10 (Yakkety Yak), AMD64

Tabla 11. Versiones Software utilizadas.

Estructura interna del script:

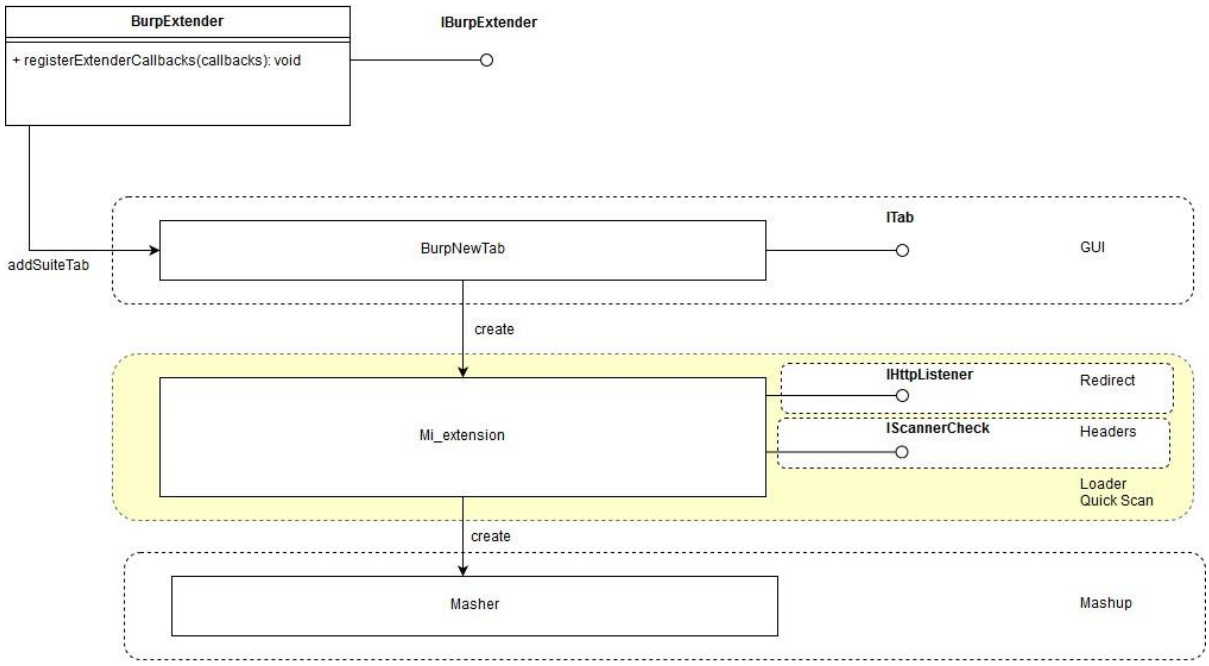


Figura 38. Interfaces utilizadas de la API de Burp Suite en la extensión.

En la figura 38 podemos observar cómo se estructura el código fuente del script, a grandes rasgos, el objetivo de la figura es mostrar que interfaces de la API de Burp Suite se están implementando.

Implementamos *IBurpExtender*, ya que es la interfaz que toda extensión debe implementar para que sea reconocida como tal por Burp. Implemento *ITab*, para crear una nueva pestaña la cual tendrá a su vez las demás sub-pestañas que implementaran las demás funcionalidades.

Implemento *IHttpListener*, que sirve para registrar un "Listener" el cual nos avisa cada vez que una petición/respuesta atraviesa el proxy web de Burp, entonces lo interceptamos, y realizamos una comprobación, en caso de que el Redirect este activo y la dirección origen sea la oportuna, realizamos el re-direccionamiento a otra dirección.

También se hace uso de la interfaz *IScannerCheck*, la cual tiene asociada un método *doPassiveScan*, que me proporciona a su vez otro objeto de la Interfaz *IHttpRequestResponse*, que es el que utilizo para analizar las cabeceras, y más tarde, sacarlas en una tabla, para la utilidad de Headers.

Las funcionalidades del Loader y Quick Scan, se realizan directamente a través del objeto de *Callbacks*, por lo cual, no necesitan implementar ninguna otra interfaz concreta (el objeto *Callbacks* lo obtenemos cuando implementamos la primera interfaz de *IBurpExtender*.)

Por último, solo me pareció adecuado, llevarme el método que realiza el Masher a una clase aparte, ya que era un método con una funcionalidad específica, pero además tenía muchas líneas de código como para dejarlas como un método de la clase Mi_extension. Asimismo, aunque podría haber sacado una clase por pestaña/utilidad, no lo considere necesario.

6.5 Limitaciones

El código de Burp Suite es privado, lo único publico a lo que podemos acceder es la API. La única limitación que podemos encontrar a la hora de crear una extensión, es la API de Burp Suite, ya que solo podemos implementar funcionalidades en función de las interfaces y métodos que te proporcionan.

CAPÍTULO 7. EVALUACIÓN DE PRESTACIONES

En este capítulo hablare sobre un posible flujo de trabajo que encaje para cada una de las funcionalidades desarrolladas en la extensión L-Tools, el cual será llevado a cabo mediante unas pruebas de concepto.

7.1 Mashup, ataque por fuerza bruta

Para esta prueba de concepto utilizare la maquina virtual de OWASP (Damn Vulnerable Web Application v1.8), Burp Suite v1.7 y la extensión L-Tools.

La principal funcionalidad de un diccionario de contraseñas, suele ser realizar un ataque de fuerza bruta, en base a ciertas palabras clave que tenemos que conocer previamente acerca de la víctima, para ello es necesario utilizar un poco de ingeniería social previamente.

Podemos obtener estas palabras clave, buscando información en google (google hacking), si sabemos su nombre de pila podemos buscarlos en las redes sociales más habituales, o incluso una llamada telefónica a la victima simulando querer realizar una encuesta sobre algún tema controvertido pero que al final sirva para recopilar alguna información personal, como DNI, teléfono, apellidos, etc.

Vamos a imaginar que tenemos una página de login, en la cual sabemos que el usuario de la víctima es *1337*, y además sabemos que apellida *char* y su fecha de nacimiento *01/01/1990* y el nombre de su mascota *ley*.

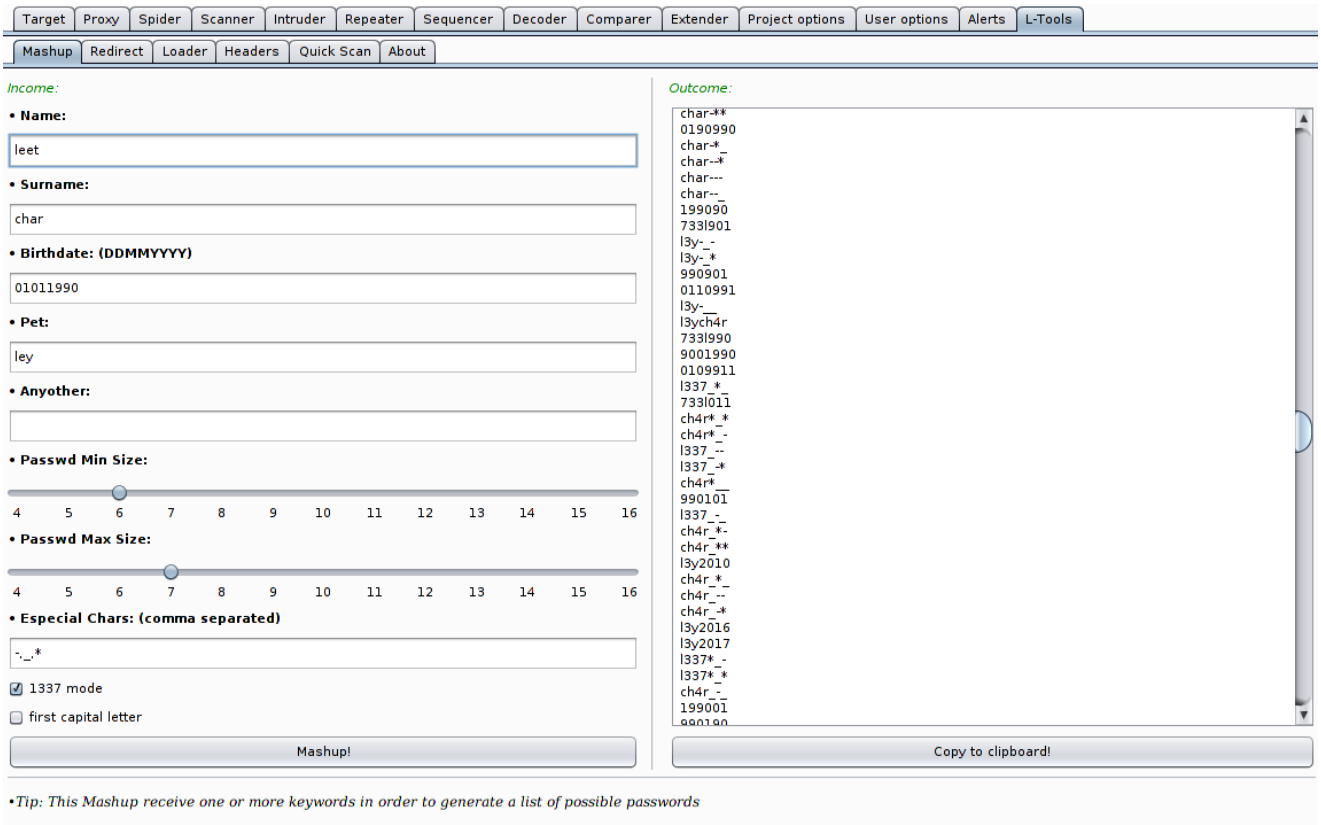


Figura 39. Generando diccionario de contraseñas.

Para generar el diccionario de contraseñas, solo tenemos que rellenar los cuadros de texto con la información que dispongamos sobre la víctima, poner una longitud mínima y máxima de contraseña y si queremos utilizar caracteres especiales, leet mode, etc. Hay que tener en cuenta que cuantos más parámetros añadamos, más grande será el diccionario, también influye en gran medida el rango de la longitud de la contraseña. Una vez lo tenemos generado, lo copiamos en el portapapeles.

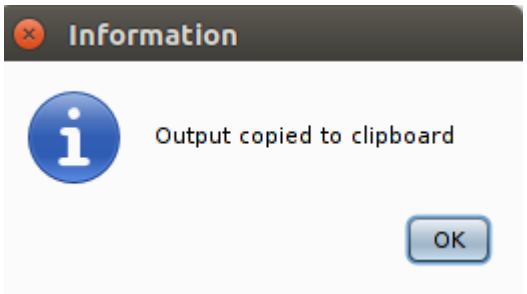


Figura 40. Modal de feedback, al copiar al portapapeles.

Lo siguiente será navegar hasta la página objetivo para poder marcárselo a Burp Suite.

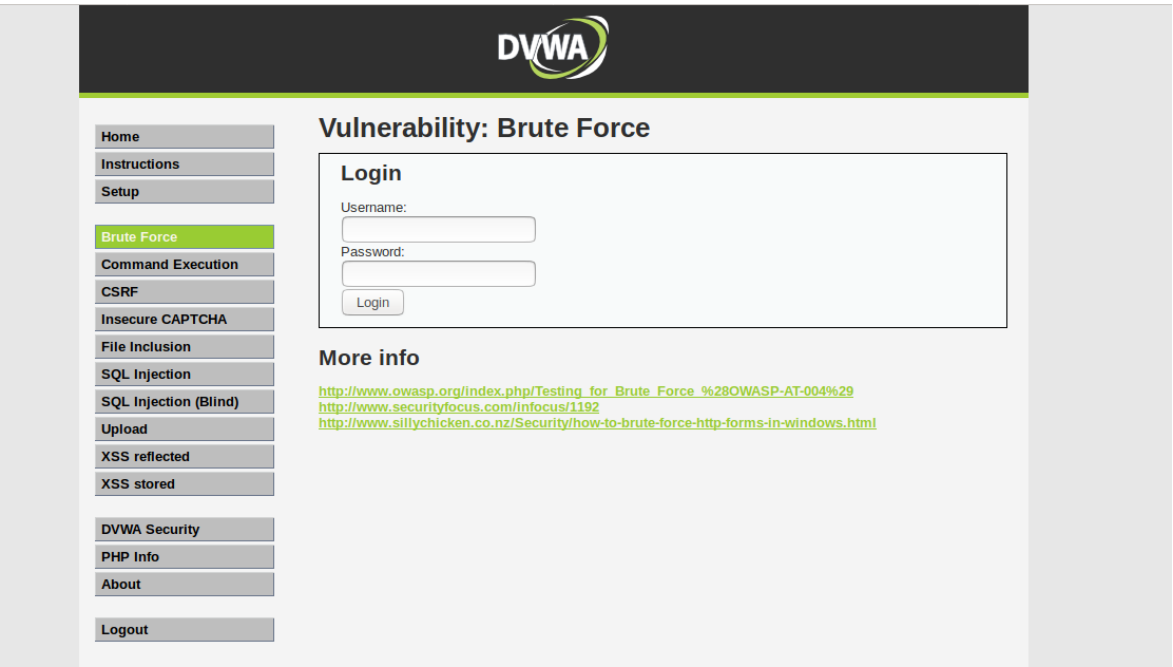


Figura 41. Maquina virtual DVWA, Brute Force.

Esta parte es alternativa, o bien puedes navegar hasta la pagina objetivo, o puedes utilizar el crawler de Burp, para mapear la aplicación y elegir que formulario o que pagina quieres atacar.

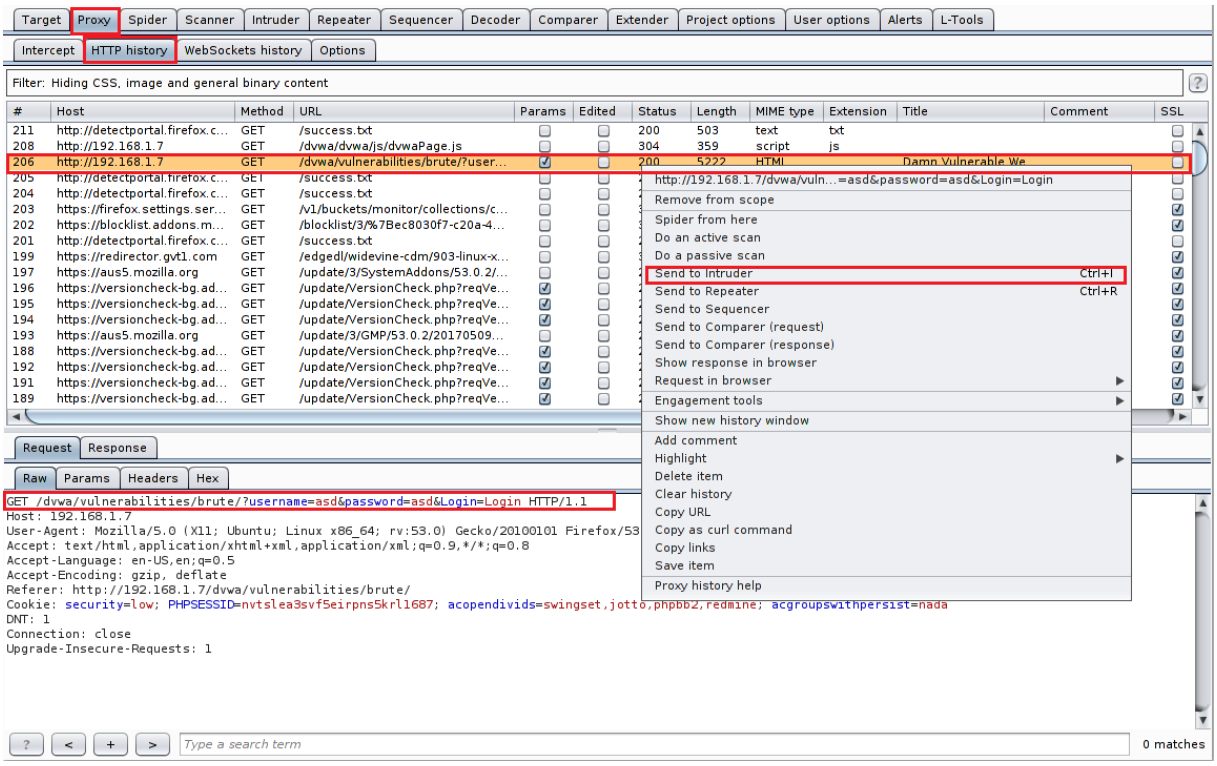


Figura 42. Mandando página objetivo al Intruder.

Una vez tenemos localizado exactamente cuál es la página a la que vamos a realizar el ataque, la podemos enviar hacia el Intruder, utilizando el menú contextual, que es la forma de trabajar con Burp Suite.

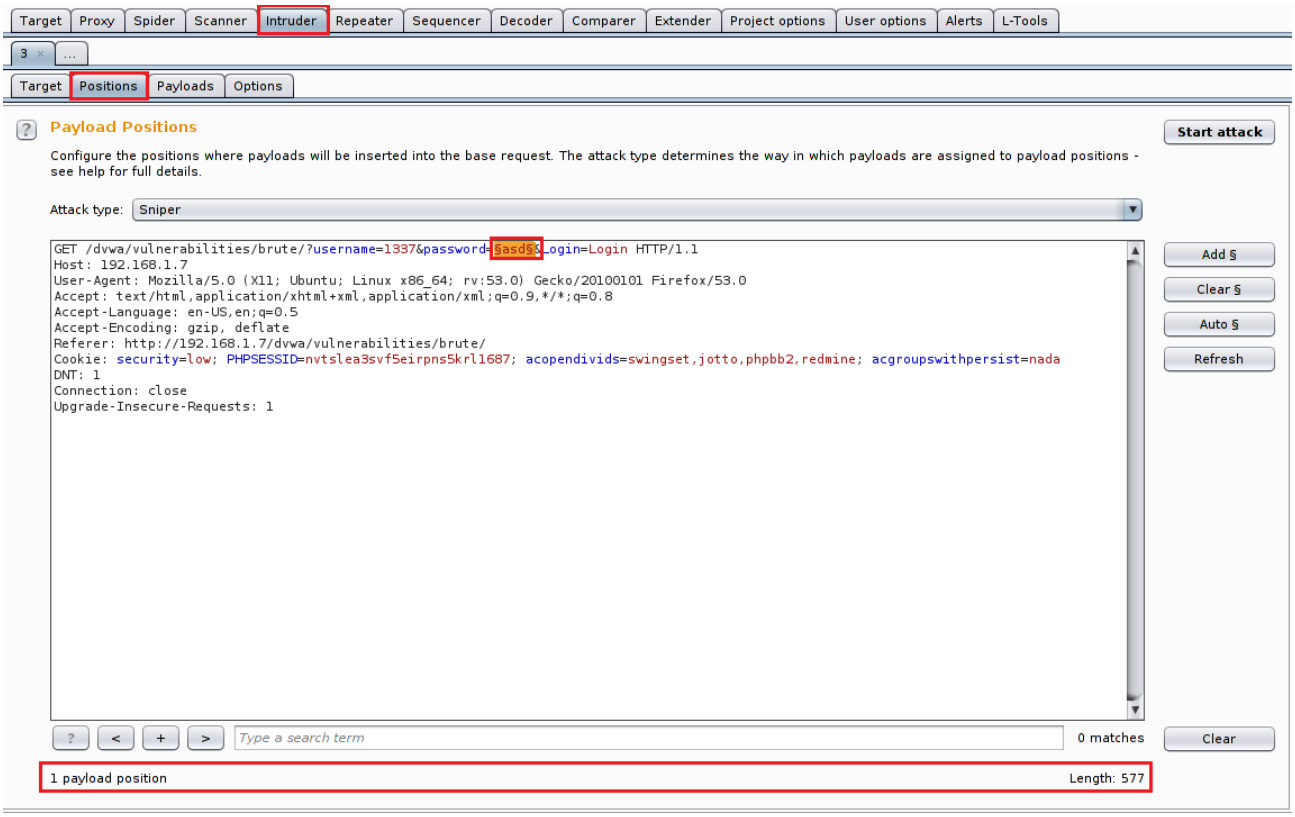


Figura 43. Intruder – Positions.

Una vez lo tenemos en el Intruder, podemos modificar la petición que hemos recibido desde el proxy, podemos observar que se envía mediante HTTP GET, y como sabemos que el usuario es *1337* lo podemos poner de forma manual en la petición, además tenemos que marcar con el botón de "*Add $*" donde se introducirán las diferentes combinaciones del Payload.

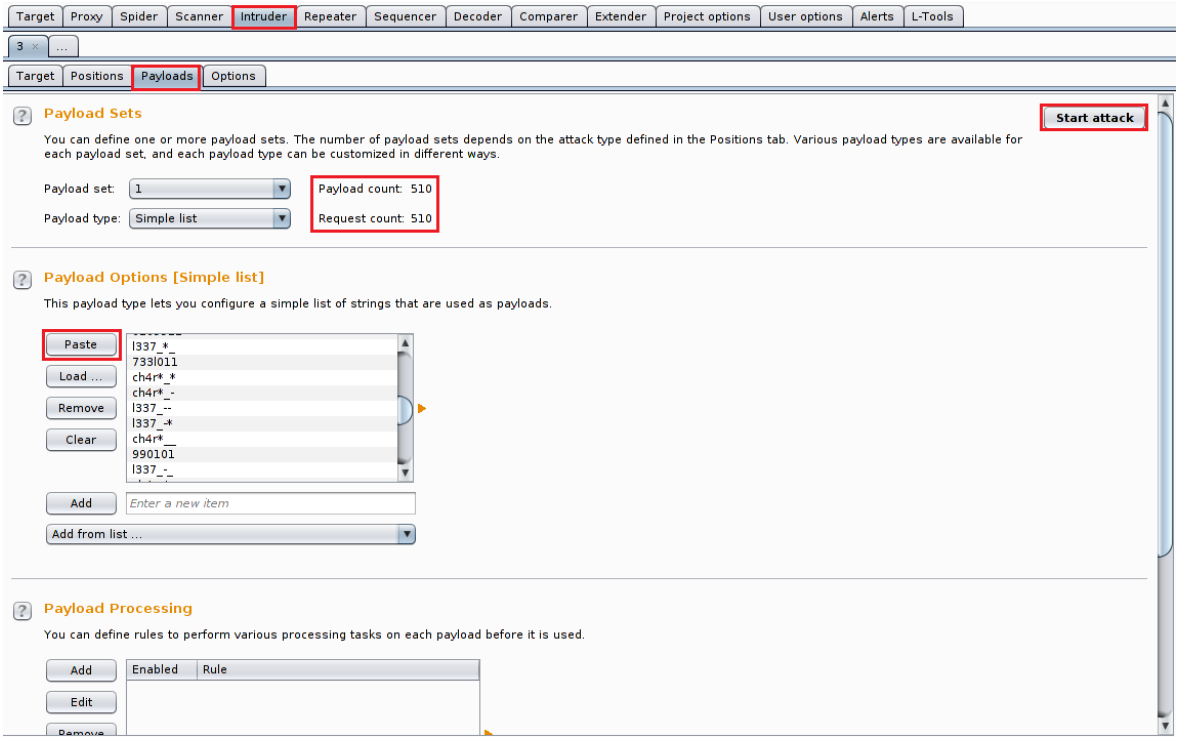


Figura 44. Configurando Payload.

Una vez llegados a este punto, lo único que tenemos que hacer es cargar, mediante el botón de "Paste", nuestro diccionario generado en el Mashup. Y ya podríamos lanzar el ataque.

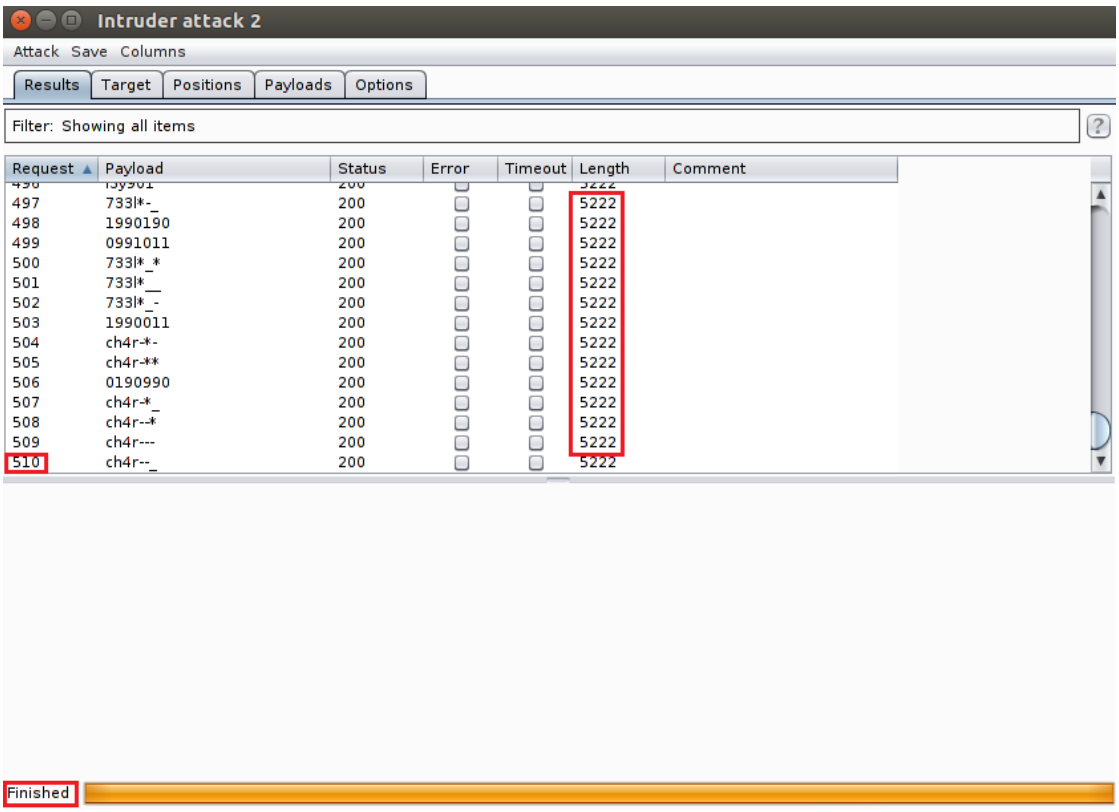


Figura 45. Resultado del ataque 1a.

Podemos observar que se han realizado las 510 peticiones que marcamos, una para cada combinación del diccionario.

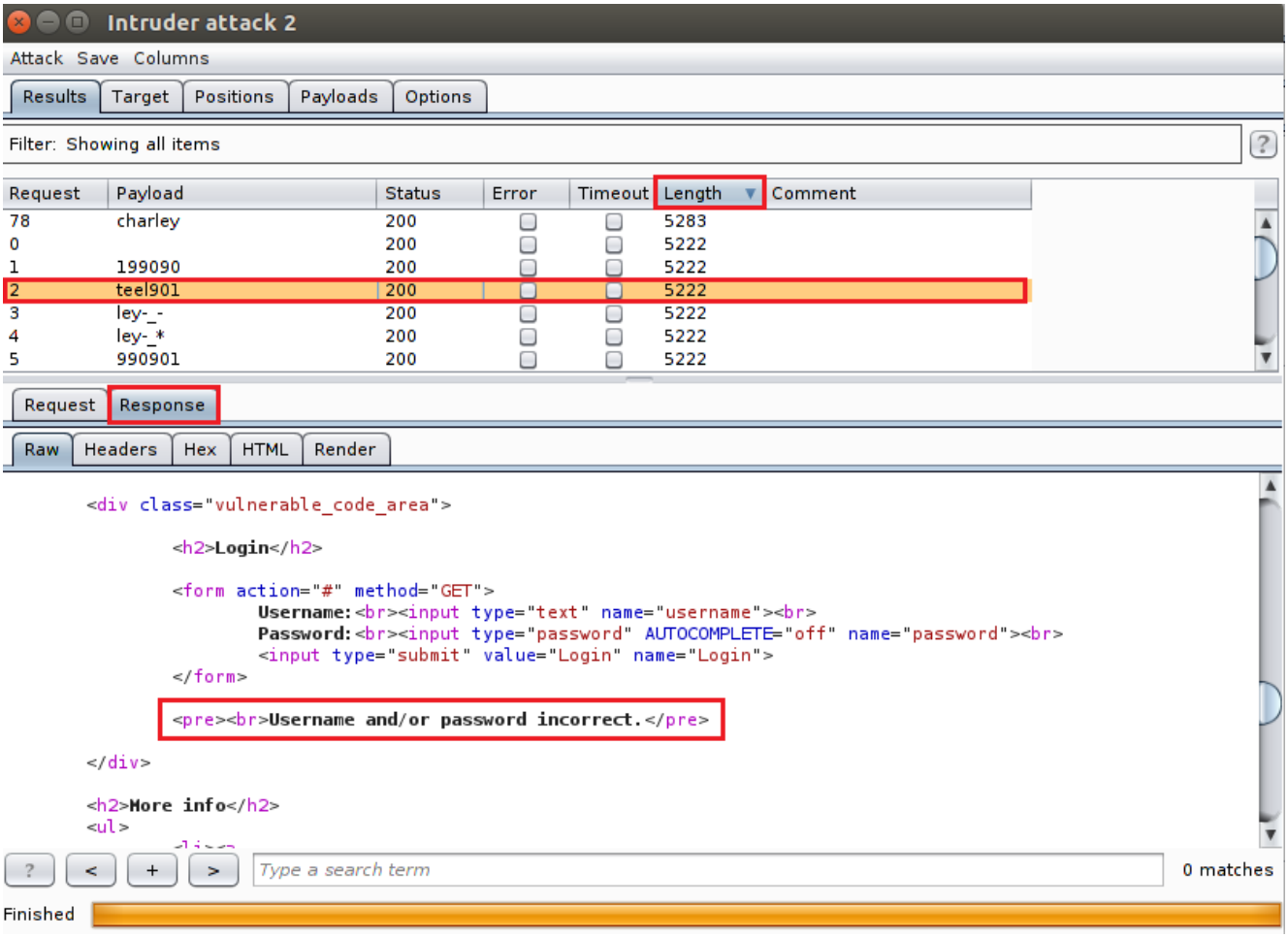


Figura 46. Resultado del ataque 1b.

Si ordenamos las respuestas obtenidas por la longitud de cada una de ellas, vemos que hay una que varia del resto. Además, si observamos una de las demás respuestas podemos ver cuál fue la respuesta del servidor, en los casos en los que la longitud de la respuesta es de 5222, la respuesta del servidor es que el usuario y la contraseña fueron incorrectos.

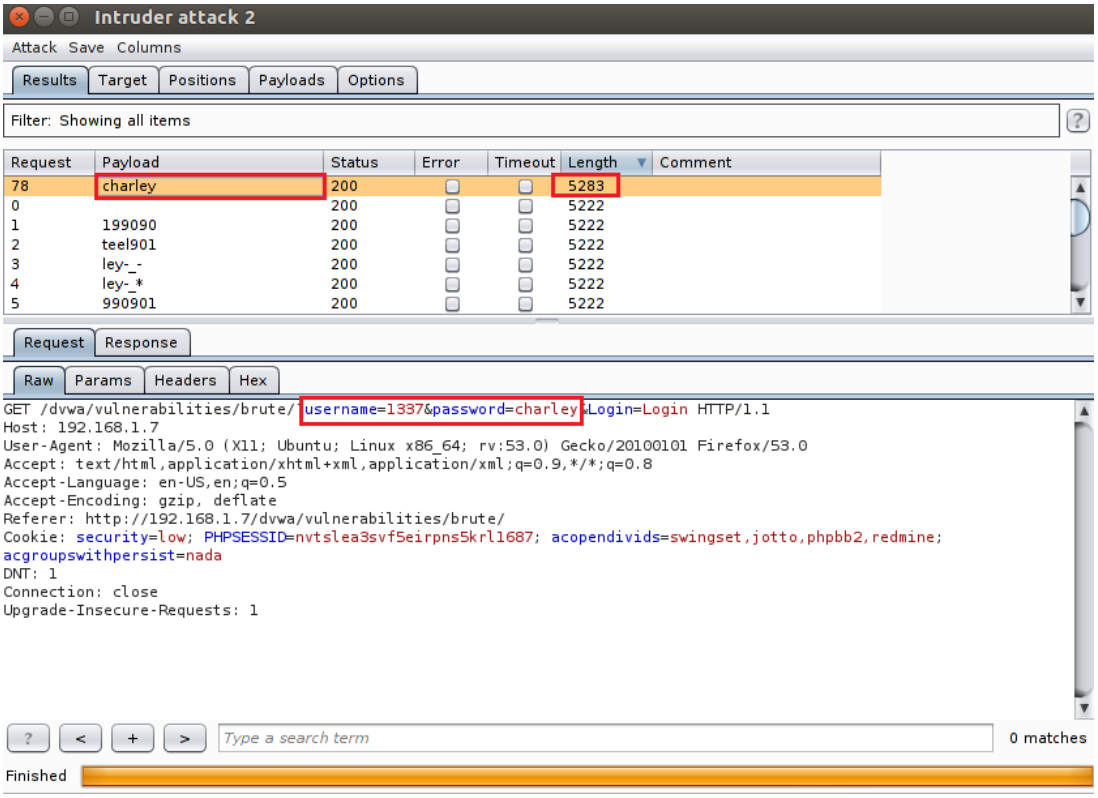


Figura 47. Resultado del ataque 1c.

En el caso de la respuesta con longitud 5283, podemos ver, cual fue la petición. (username=1337 & password=charley).

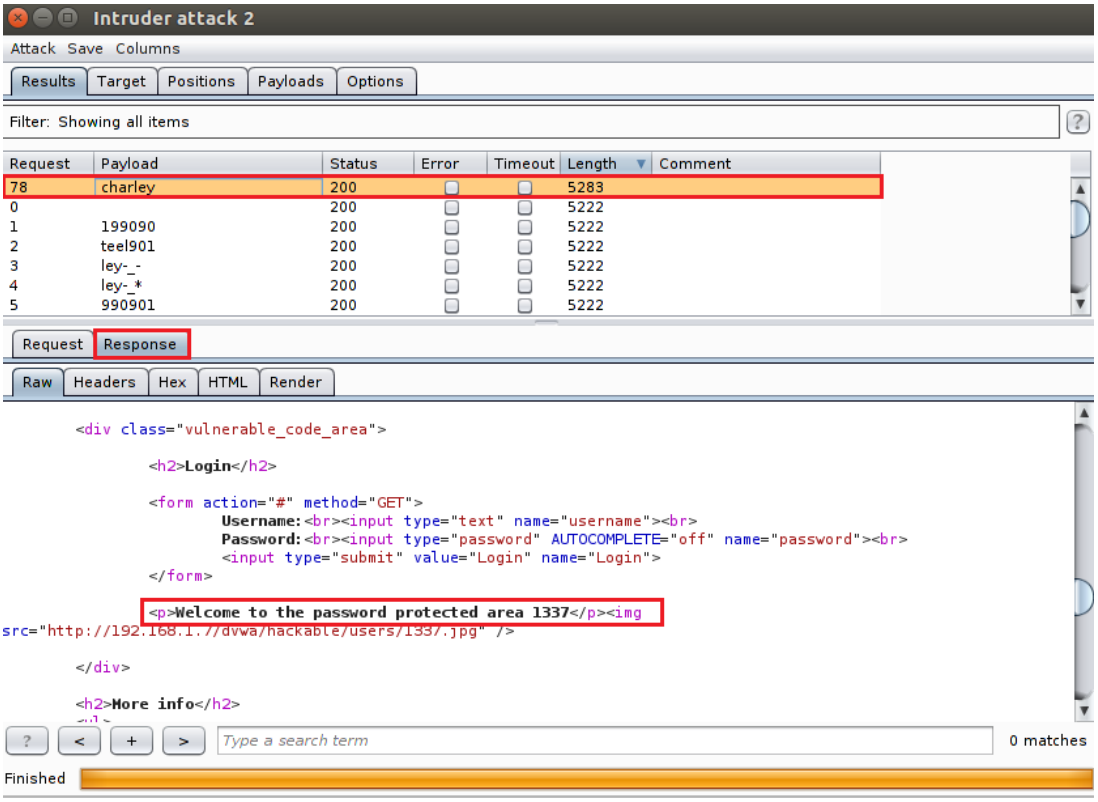


Figura 48. Resultado del ataque 1d.

Y si observamos la respuesta del servidor, podemos observar un mensaje de bienvenida, que indica que esa (*charley*) es la contraseña válida para dicho usuario.

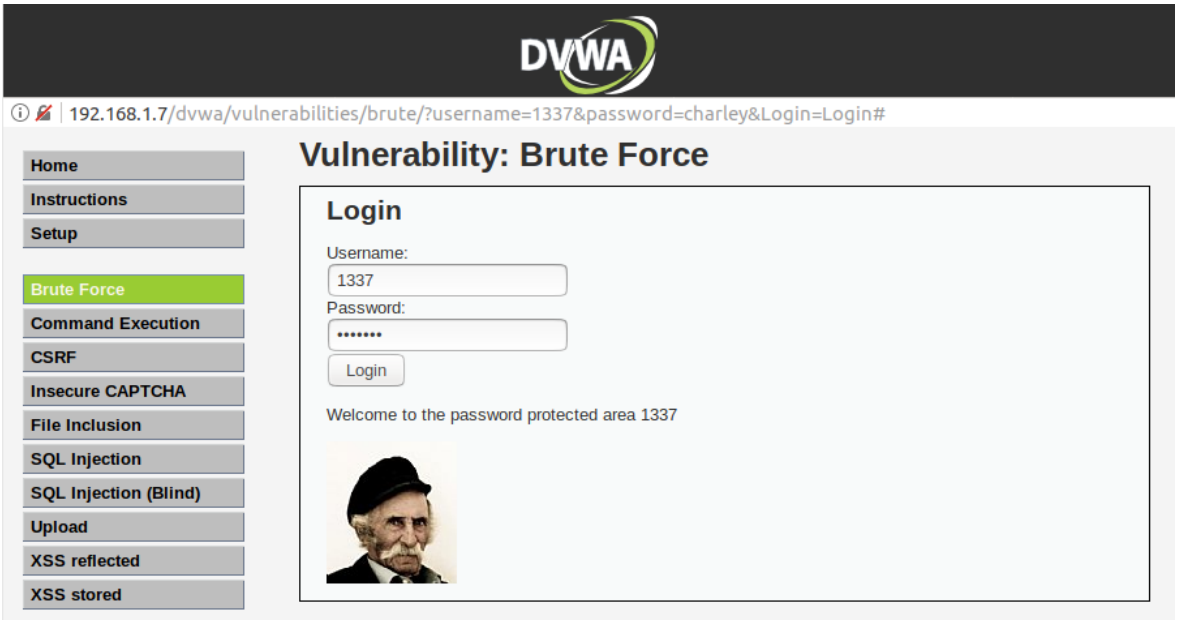


Figura 49. Usuario y contraseña correctos.

7.2 Redirect, redirigiendo tráfico

Esta funcionalidad es muy sencilla, introducimos un par de host X e Y, y lo que hace la herramienta es redirigir del host X al host Y.

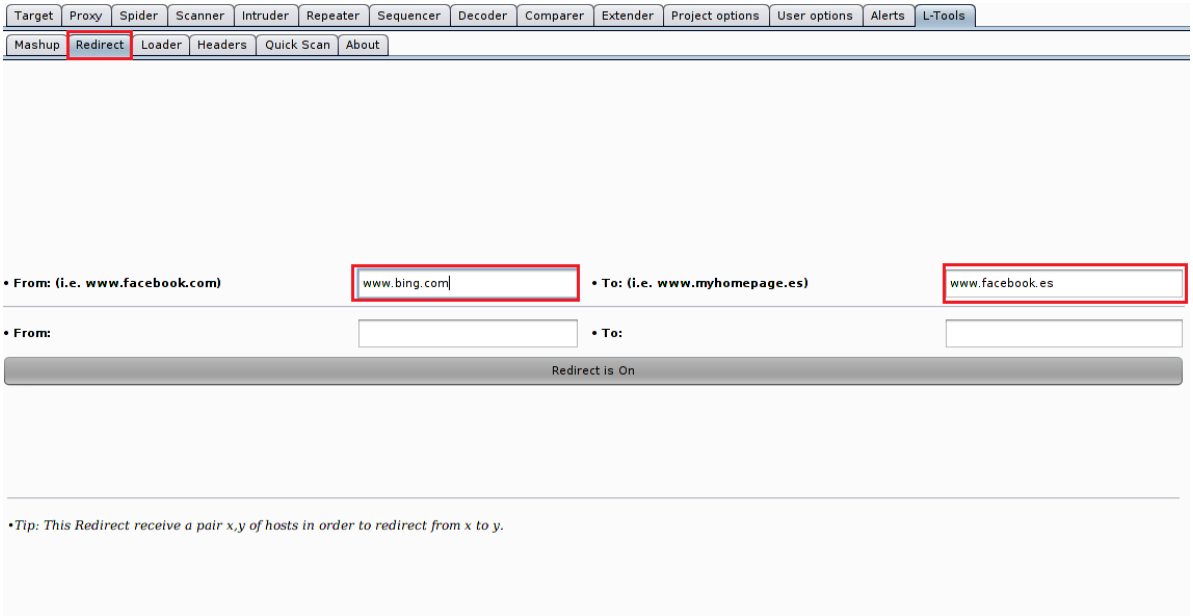


Figura 50. L-Tools – Redirect.

Una vez rellenados los cuadros de texto, lo podemos activar/desactivar pulsando el botón, luego en el cliente realizamos la petición y vemos como reacciona el proxy web.

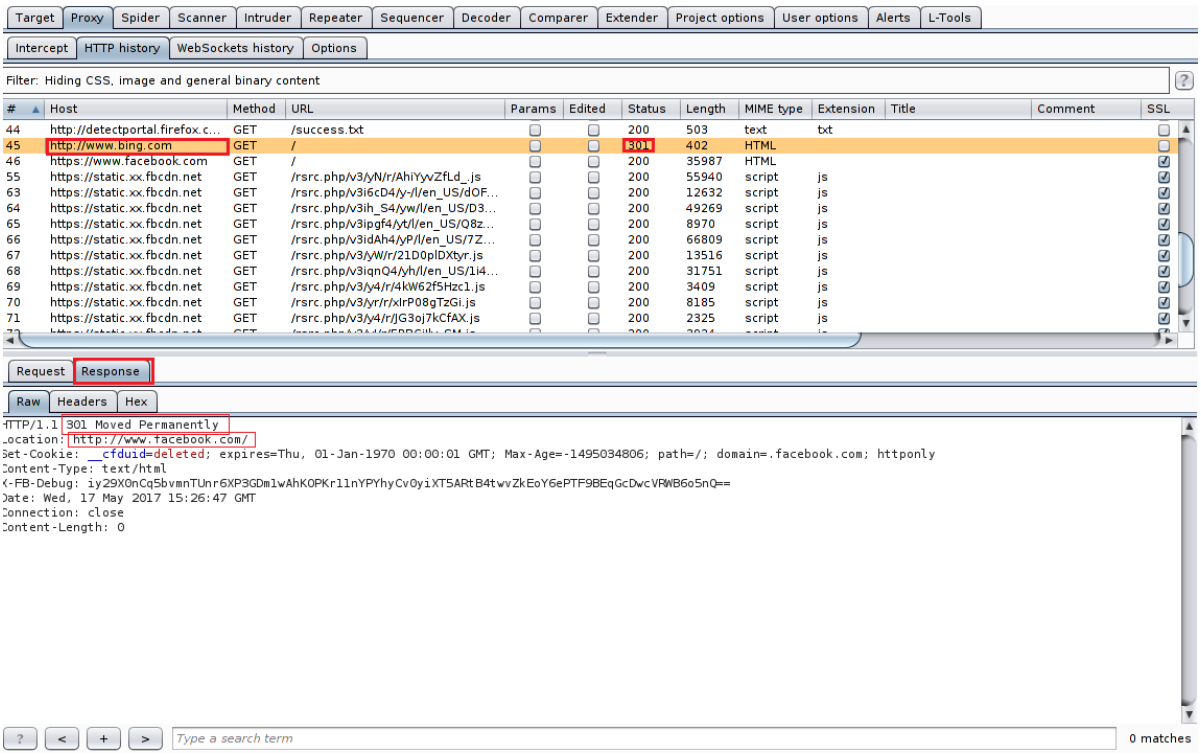


Figura 51. Proxy web, redirigiendo el tráfico.

Vemos como se ha realizado la petición al host de www.*bing.com* y como nos redirige al host de *www.facebook.es*.

7.3 Loader, cargando una lista de objetivos

La funcionalidad del Loader nos permite cargar una lista de URL o Host objetivos, los cuales serán objeto de nuestro análisis.

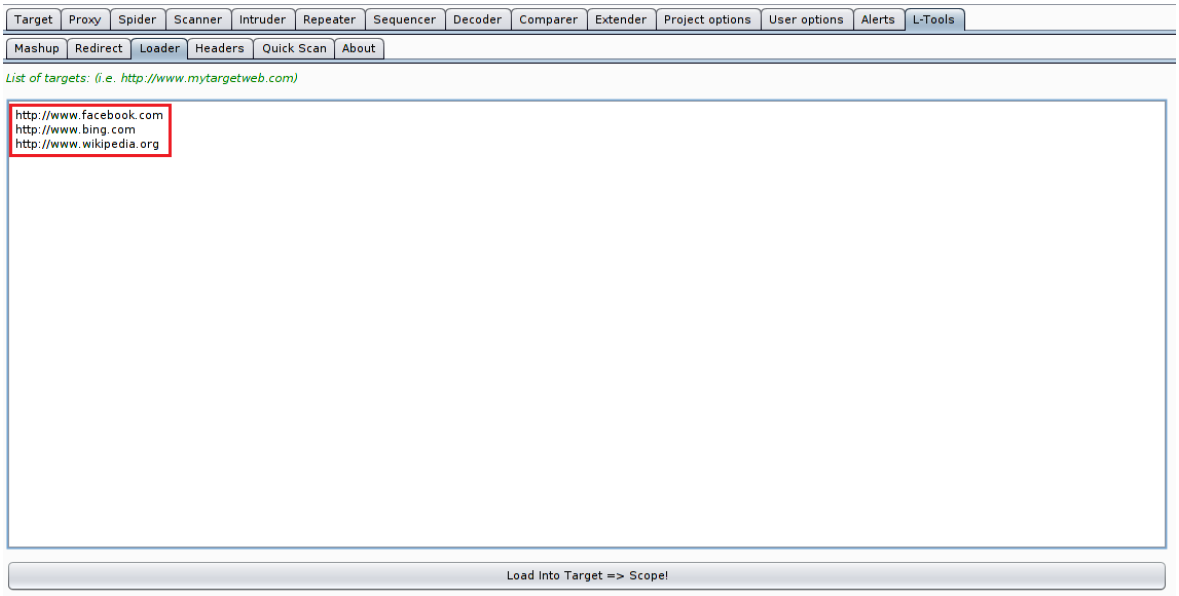


Figura 52. L-Tools – Loader.

Rellenamos el campo y lo añadimos al Scope pulsando el botón.

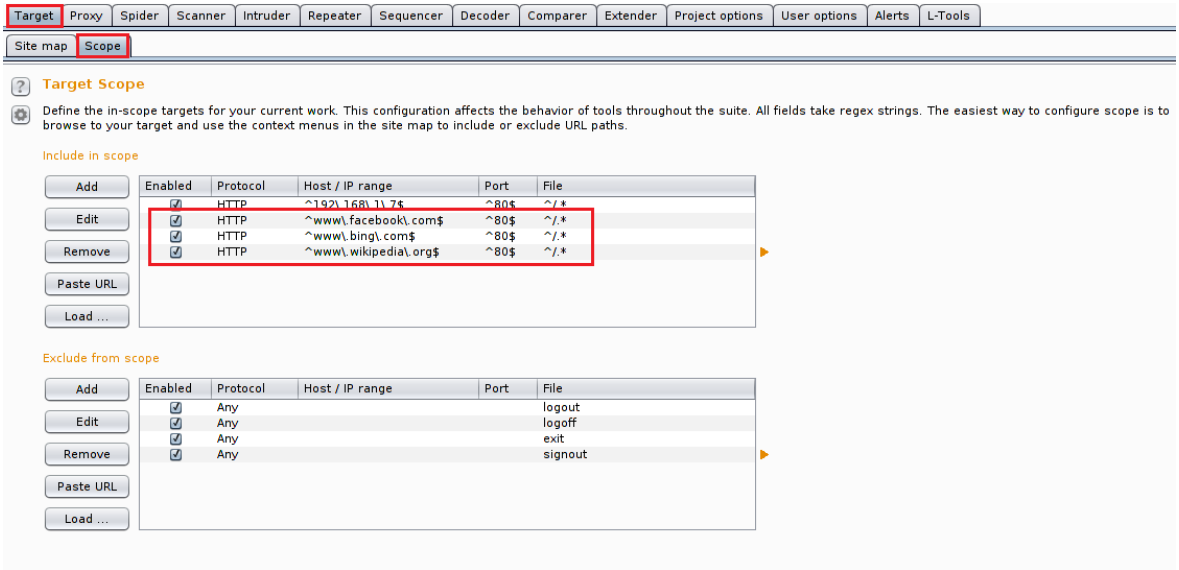


Figura 53. Burp Suite Target – Scope.

Tener los objetivos delimitados en la lista de Scope es importante a la hora de trabajar con Burp Suite, ya que desde allí podemos filtrarlos en el Site Map y enviarlos al resto de herramientas de Burp, incluidas las extensiones, es una forma de indicarle a Burp Suite cuales son exactamente las paginas objetivo y así evitar que por ejemplo, el spider o el escáner pasivo salten a otras páginas que no nos interesan o que simplemente no disponemos de permiso.

7.4 Headers, analizando cabeceras HTTP

El Headers nos brinda información sobre las cabeceras de las respuestas que pasan a través del proxy web, y señala algunas de las más importantes con lo que respecta a la seguridad.

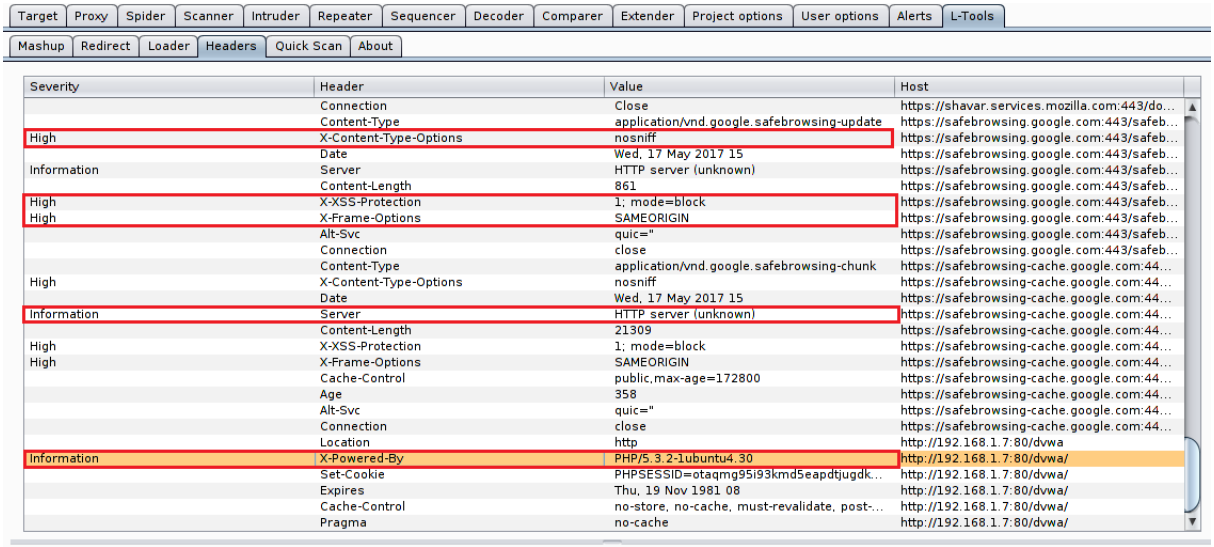


Figura 54. L-Tools – Headers.

Una vez observamos que nos falta una cabecera de seguridad o que simplemente está mal configurada, lo único que tendríamos que hacer es configurarla adecuadamente. Por ejemplo, si queremos ocultar el X-Powered-By para que no muestre información de la versión del servidor que estamos utilizando, en PHP lo podríamos hacer en el fichero de configuración php.ini, poniendo a OFF la directriz *expose_php*.

Figura 55. Php.ini expose_php = Off.

7.5 Quick Scan, escaneando hosts

Quick Scan nos permite lanzar una lista de objetivos a una cola para que sea escaneada por Burp, y en el caso de encontrar vulnerabilidades, nos da detalles sobre estas. Para esta prueba de concepto, utilizo las maquinas virtuales de bWAPP y DVWA.

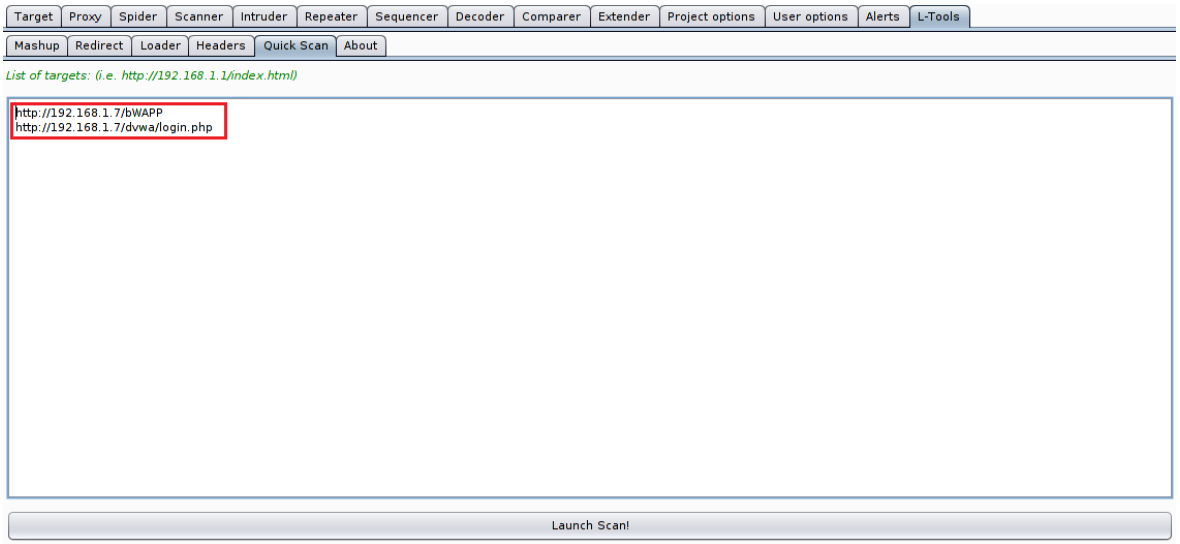


Figura 56. L-Tools – Quick Scan.

Una vez que ponemos la lista de objetivos y pulsamos el botón, veremos cómo se ilumina una luz en la pestaña de “Scanner”, indicando que hay nuevos elementos. Además, si nos movemos a dicha pestaña, veremos una barra de progreso que nos indicara cómo va el proceso, mientras que realiza las peticiones pertinentes para buscar vulnerabilidades.

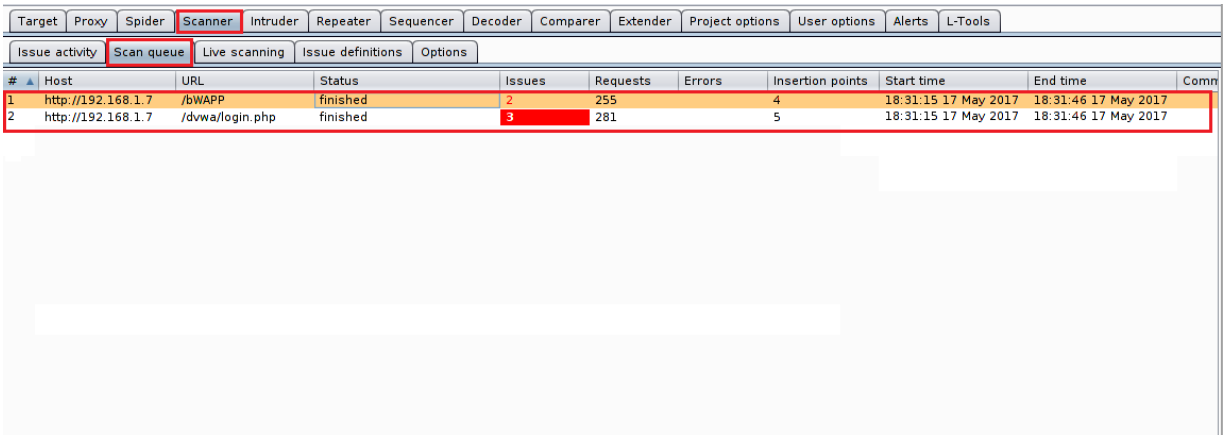


Figura 57. Burp escaneando objetivos.

Una vez finalice el escáner, nos mostrara el numero de fallos que ha encontrado, algunos pueden ser falsos positivos, o mensajes simplemente de información, pero nos pueden ayudar como una primera aproximación al problema. Asimismo, podemos clicar dos veces sobre estos objetos y se nos abrirá un modal que nos indicara los fallos encontrados.

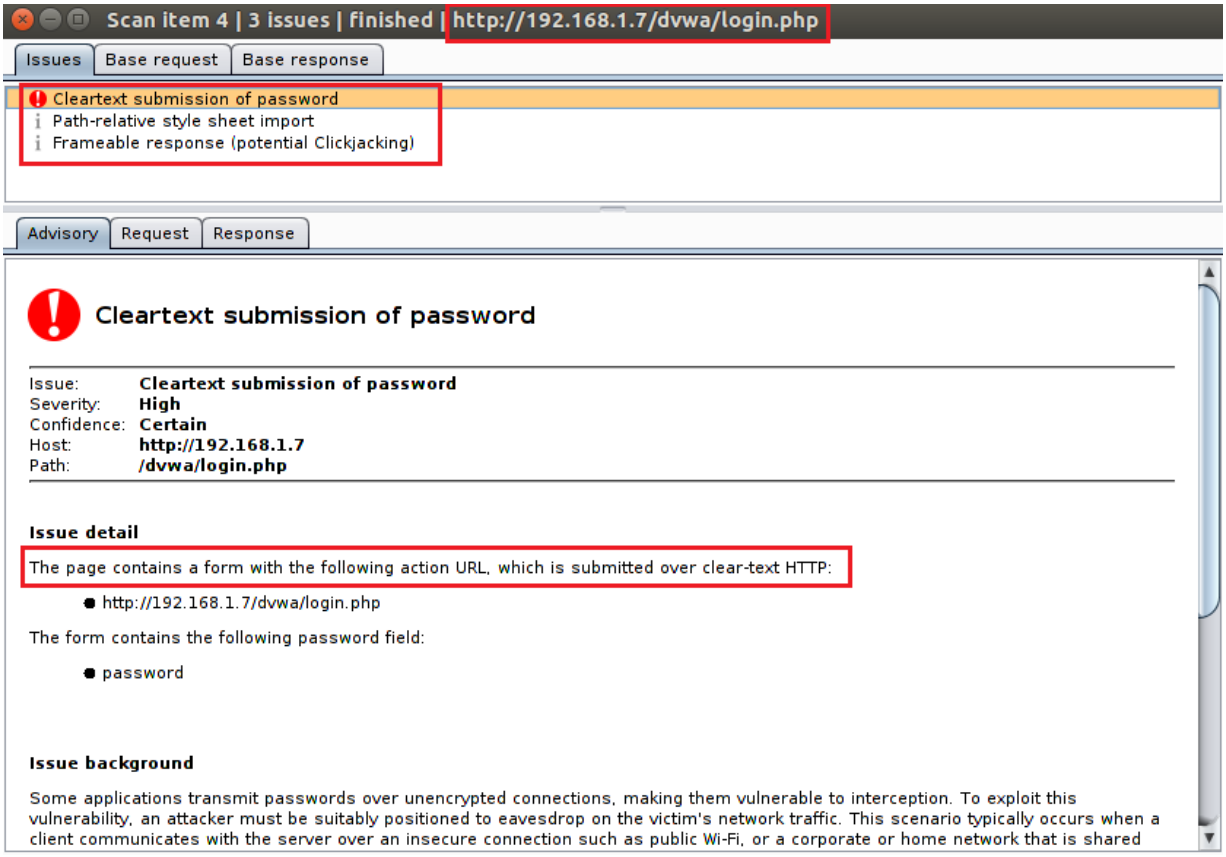


Figura 58. Resultado del escáner.

En este caso nos dice que hay un campo de password en un formulario el cual viaja en texto plano sobre una comunicación sin cifrar.

CAPÍTULO 8. CONCLUSIONES Y PROPUESTAS

Para ir concluyendo con este Trabajo de Fin de Grado, expondré algunas de las conclusiones que he obtenido, al igual que posibles futuros trabajos o ampliaciones del mismo.

8.1 CONCLUSIONES

A medida que avanza la tecnología y van surgiendo cada vez nuevos protocolos, servicios, sistemas, frameworks, etc., vemos que la seguridad, es un factor relevante al cual deberíamos dedicarle tiempo, ya que es mejor dedicarle tiempo a intentar prevenir un fallo o vulnerabilidad, que luego intentar reparar algo que estaba mal planificado o diseñado desde el principio.

Las herramientas que existen para escanear vulnerabilidades, no son perfectas, muchas veces detectan falsos positivos, pero pueden servir como primera aproximación hacia el problema. En cuanto a los proxy web, son ideales para que, una persona, pueda examinar minuciosamente como responde una determinada aplicación web, ante posibles peticiones o interacciones con un cliente, más allá del resto de funcionalidades, que también sirven para complementar dicho análisis.

Podemos hacernos una idea de la cantidad de vulnerabilidades web que aun existen, sobre todo en las pequeñas y medianas empresas, gracias a organizaciones como la de OWASP y sus proyectos que se llevan a cabo, para velar por la seguridad en las tecnologías web.

Algunas de las vulnerabilidades más comunes (OWASP Top Ten) se pueden evitar y/o resolver en las fases de diseño y desarrollo de una aplicación web concreta, cosas como el SQL Inyection o el XSS, en gran medida dependen, de cómo este diseñada la aplicación misma, ya que hemos visto que existen técnicas y medidas para evitar dichas vulnerabilidades, y están siempre a disposición de los desarrolladores.

En lo que respecta a Burp Suite, vemos que es una herramienta muy potente, la cual te permite llevar a cabo una cantidad enorme de pruebas y test sobre una aplicación web. No solo nos permite examinar todas las peticiones y respuestas, mapear toda una aplicación, repetir una petición con distintos parámetros, etc. Sino que además podemos lanzar escáneres activos y pasivos, permitiendo detectar vulnerabilidades, y nos da la oportunidad de explotar dichas vulnerabilidades.

En cuanto a la API de Burp y sus extensiones, vemos que la mayoría de funcionalidades que te permite crear en base a la propia API, ya está desarrollada o implementada, donde hay un punto de inflexión es en la funcionalidad muy específica, en las cuales se desarrolla una extensión enfocada a una aplicación muy concreta.

Gracias a proyectos como los de Jython o JRuby, podemos desarrollar una extensión para Burp Suite (sin limitarnos solo a este), escribiendo en lenguajes de programación de muy alto nivel y con pocas líneas de código, en comparación a si tuviéramos que hacerlo en Java.

8.2 TRABAJO FUTURO Y POSIBLES AMPLIACIONES

Para finalizar con este último capítulo, comentaremos algunas de las propuestas que se podrían añadir, al trabajo actual, en trabajos futuros:

- Realizar una auditoría a una aplicación web, contactar con un cliente que necesite realizar una auditoría web, llegar a un acuerdo, y realizar una auditoría de su aplicación web.
- Realizar una prueba de Pentesting a una aplicación web, obtener el permiso explicito de un cliente, y ver hasta donde de intrusivas pueden ser las pruebas, y realizar el Pentesting de la aplicación, generar un informe y entregárselo al cliente.
- Desarrollar una extensión para OWASP ZAP, estudiar la API de OWASP ZAP, y desarrollar alguna extensión que sea de utilidad para la comunidad.

BIBLIOGRAFÍA

LIBROS Y ARTICULOS

[1] Dr. José Luis Martínez Martínez, Temario - *Tema 1. Introducción a la Seguridad*, 2017, accessed: 2017-02-03

[2] Dr. José Fernández G., *Seguridad en Informática*, http://www.aprocal.org.mx/files/2200/03SeguridadenInformaticaV1.0.pdf, 2013, accessed: 2017-02-03

[3] Andrés Romero Mier y Mario Alejandro Vásquez Martínez, *Aspectos Básicos de la Seguridad en Aplicaciones Web*, http://www.seguridad.unam.mx/documento/?id=17, 2016, accessed: 2017-02-06

[4] Microsoft Developer Network, *Procedimientos de seguridad básicos para Aplicaciones Web*, https://msdn.microsoft.com/es-es/library/zdh19h94(v=vs.100).aspx, 2007, accessed: 2017-02-06

[5] Miguel Angel Álvarez, *Qué es Responsive Web Design*, https://desarrolloweb.com/articulos/que-es-responsive-web-design.html, 2015, accessed: 2017-02-06

[6] Mario Pérez Esteso, *Cómo funciona HTTPS*, https://geekytheory.com/como-funciona-https, 2015, accessed: 2017-02-06

[7] Wikipedia, *Cliente-Servidor*, https://es.wikipedia.org/wiki/Cliente-servidor, 2017, accessed: 2017-02-06

[8] OWASP, *Cross Site Scripting (XSS)*, https://www.owasp.org/index.php/Cross-site_Scripting_(XSS), 2016, accessed: 2017-02-07

[9] OWASP, *Social Engineering*, https://www.owasp.org/index.php/Social_Engineering, 2007, accessed: 2017-02-07

[10] OWASP, *Cross Site Request Forgery (CSRF)*, https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF), 2017, accessed: 2017-02-07

[11] OWASP, *ClickJacking*, https://www.owasp.org/index.php/Clickjacking, 2017, accessed: 2017-02-07

[12] OWASP, *Testing for Remote File Inclusion*, https://www.owasp.org/index.php/Testing_for_Remote_File_Inclusion, 2014, accessed: 2017-02-07

[13] OWASP, *Testing for Local File Inclusion*, https://www.owasp.org/index.php/Testing_for_Local_File_Inclusion, 2014, accessed: 2017-02-08

[14] José María Alonso Cebrián, Antonio Guzmán Sacristán, Pedro Laguna Durán Y Alejandro Martín Bailón, *Ataques a aplicaciones web*, https://www.exabyteinformatica.com/uoc/Informatica/Seguridad_en_bases_de_datos/Seguridad_en_bases_de_datos_(Modulo_2).pdf, 2012, accessed: 2017-02-08

[15] Ronan Dunne, *Web Defacement XSS*, https://dunnesec.com/category/attacks-defence/web-defacement-xss/, 2013, accessed: 2017-02-08

[16] OWASP, *Blind XPath Injection*, https://www.owasp.org/index.php/Blind_XPath_Injection, 2013, accessed: 2017-02-08

[17] OWASP, *SQL Injection*, https://www.owasp.org/index.php/SQL_Injection, 2016, accessed: 2017-02-08

[18] OWASP, *Blind SQL Injection*, https://www.owasp.org/index.php/Blind_SQL_Injection, 2013, accessed: 2017-02-08

[19] Robotstxt.org, *About / robots.txt*, http://www.robotstxt.org/robotstxt.html, 2007, accessed: 2017-02-08

[20] Pablo F. Iglesias, *Google Hacking: Operadores y comandos*, https://www.pabloyglesias.com/google-hacking/, 2015, accessed: 2017-02-08

[21] OWASP, *Man In The Browser*, https://www.owasp.org/index.php/Man-in-the-browser_attack, 2016, accessed: 2017-02-09

[22] OWASP, *Secure Headers Project*, https://www.owasp.org/index.php/OWASP_Secure_Headers_Project, 2017, accessed: 2017-02-09

[23] Asociación Nacional de Tasadores y Peritos Judiciales Informáticos, *Auditoría de seguridad informática*, http://www.antpji.com/antpji2013/index.php/articulos2/111-auditoria-de-seguridad-informatica, 2013, accessed: 2017-02-09

[24] OpenWebinars, *¿Qué es el Pentesting?*, https://openwebinars.net/blog/que-es-el-pentesting/, 2015, accessed: 2017-02-09

[25] EC-Council, *Certified Ethical Hacking Certification*, https://www.eccouncil.org/programs/certified-ethical-hacker-ceh/, 2017, accessed: 2017-02-09

[26] Cyberseguridad.net, *Las fases de un test de penetración*, https://cyberseguridad.net/index.php/455-las-fases-de-un-test-de-penetracion-pentest-pentesting-i, 2015, accessed: 2017-02-09

[27] Open Source Security Testing Methodology Manual, *OSSTMM*, https://www.isecom.org/Open-Source-Cybersecurity-Playbook.pdf, 2016, accessed: 2017-02-10

[28] OWASP, *Main Page*, https://www.owasp.org/index.php/Main_Page, 2017, accessed: 2017-02-10

[29] OWASP, *Featured Projects Include*, https://www.owasp.org/index.php/Press , 2017, accessed: 2017-02-10

[30] OWASP, *Projects*, https://www.owasp.org/index.php/Category:OWASP_Project, 2017, accessed: 2017-02-10

[31] Juan Luis Serradilla Amarilla, *Normativa para el desarrollo de aplicaciones web seguras*, http://www.um.es/atica/documentos/NORowasp.pdf, 2010, accessed: 2017-02-13

[32] Antonio Leiva, *Patrones de diseño de software*, https://devexperto.com/patrones-de-diseno-software/, 2016, accessed: 2017-02-13

[33] William Penguin, *Qué es URI*, http://www.yoseomarketing.com/blog/que-es-uri-que-significa-diferencia-url/, 2016, accessed: 2017-02-13

[34] Carlos Castillo, *¿Qué es un crawler o spider?*, http://tejedoresdelweb.com/w/%C2%BFQu%C3%A9_es_un_crawler_o_spider%3F, 2016, accessed: 2017-02-13

[35] W3AF, *Índex*, http://w3af.org/, 2013, accessed: 2017-02-14

[36] Acunetix, *Índex*, http://www.acunetix.com/, 2017, accessed: 2017-02-15

[37] Tenable, *NESSUS Vulnerability Scanner*, https://www.tenable.com/products/nessus-vulnerability-scanner, 2017, accessed: 2017-02-17

[38] Offensive Security, *Working with Nessus and Metasploit*, https://www.offensive-security.com/metasploit-unleashed/working-with-nessus/, 2017, accessed: 2017-02-17

[39] Leandro Alegsa, *Definición de Proxy web*, http://www.alegsa.com.ar/Dic/proxy.php, 2017, accessed: 2017-02-17

[40] OWASP, *OWASP Zed Attack Proxy Project*, https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project, 2017, accessed: 2017-02-20

[41] PORTSWIGGER Web Security, *Using Burp Suite*, https://portswigger.net/burp/help/suite_usingburp.html, 2017, accessed: 2017-02-22

[42] PORTSWIGGER Web Security, *Issue Definitions*, https://portswigger.net/KnowledgeBase/Issues/, 2017, accessed: 2017-02-27

[43] PORTSWIGGER Web Security, *Burp Tools*, https://portswigger.net/burp/help/suite_burptools.html, 2017, accessed: 2017-02-27

[44] Bernardo Damele A. G., Miroslav Stampar, *SQLMap*, http://sqlmap.org/, 2015, accessed: 2017-03-16

[45] Kevin Johnson, *Laudanum: Injectable Web Exploit Code*, https://github.com/jbarcia/Web-Shells/tree/master/laudanum, 2015, accessed: 2017-03-16

[46] Robin Wood Y DigiNinja, *CeWL Custom Word List Generator*, https://digi.ninja/projects/cewl.php, 2015, accessed: 2017-03-16

[47] Jason Gilliam, *Burp CO2*, http://burpco2.com/, 2016, accessed: 2017-03-16

[48] Antonio Sanchez, *Dionach/Headers Analyzer – Burp Suite Extension*, https://github.com/dionach/headersanalyzer, 2014, accessed: 2017-03-16

[49] Michal Melewski, *JSON Decoder*, https://github.com/Meatballs1/burp-extensions/tree/master/JSON, 2015, accessed: 2017-03-16

[50] Auth0, *What is SSO (Single Sign-On)?*, https://auth0.com/docs/sso, 2016, accessed: 2017-03-20

[51] GitHub RUB-NDS / WS Attacker, *Chair of Network and Data Security - WS-Attacker*, https://github.com/RUB-NDS/WS-Attacker/, 2017, accessed: 2017-03-20

[52] GitHub RUB-NDS / BurpSSOExtension, Tim Guenther, *EsPReSSO*, https://github.com/RUB-NDS/BurpSSOExtension, 2017, accessed: 2017-03-20

[53] PORTSWIGGER Web Security, *Using Burp with SQLMap*, https://support.portswigger.net/customer/en/portal/articles/2163785-using-burp-to-exploit-blind-sql-injection-bugs, 2017, accessed: 2017-03-20

[54] OWASP, *Top Ten Project*, https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project, 2017, accessed: 2017-03-27

[55] PORTSWIGGER Web Security, *Package Burp – API*, https://portswigger.net/burp/extender/api/, 2017, accessed: 2017-04-27

[56] OWASP, *Secure Headers Project*, https://www.owasp.org/index.php/OWASP_Secure_Headers_Project, 2017, accessed: 2017-05-02

ENLACES INTERNET

[OWASP] The Open Web Application Security Project, https://www.owasp.org/index.php/Main_Page

[Burp Suite] PORTSWIGGER WEB SECURITY, https://portswigger.net/burp/

[OSSTMM] Open Source Security Testing Methodology Manual, http://www.isecom.org/research/playbook.html, accessed: 2017-02-10

[Figura 3] E-SPIN International, *Blind Cross Site Scripting (XSS) Attack, Vulnerability, Alert and Solution*, http://espincorp.freshdesk.com/support/solutions/articles/1000232911-

blind-cross-site-scripting-xss-attack-vulnerability-alert-and-solution, accessed: 2017-02-07

[Figura 4] OWASP, *Testing for ClickJacking*, https://www.owasp.org/index.php/Testing_for_Clickjacking_(OTG-CLIENT-009), accessed: 2017-02-07

[Figura 9] W3AF, *Demo SQLi*, http://w3af.org/take-a-tour, accessed: 2017-02-14

[Figura 18] PORTSWIGGER WEB SECURITY, *Testing Workflow*, https://portswigger.net/burp/help/suite_usingburp.html, accessed: 2017-02-18

[Figura 24] Tim Guenther, *Bachelor Thesis*, https://www.nds.rub.de/media/nds/arbeiten/2015/10/30/Tim_Guenther-EsPReSSO-BA.pdf, accessed: 2017-03-20

GLOSARIO DE TÉRMINOS

Vulnerabilidad

En Informática, las vulnerabilidades son puntos débiles del software, hardware o de los sistemas, que permiten que un atacante comprometa la seguridad de dichos sistemas.

Ataque

En Informática, un ataque es un intento organizado e intencionado causado por una o varias personas para infringir daños o problemas a un sistema.

Framework

Es una estructura conceptual y tecnológica de soporte definido que pueden servir de base para la organización y desarrollo de software.

Exploit

Tecnica que permite aprovechar una vulnerabilidad para producir un daño y/o romper la seguridad de un sistema.

Payload

Es la parte de un Exploit que tiene la función de ejecutarse en la máquina de la victima para realizar una acción maliciosa.

Librería o DLL

Dynamic-Link Library, son librerías dinámicas con código ejecutable que se cargan bajo la demanda de un programa o aplicación.

Plug-in o AddOns

En Informática, un Plug-in o complemento, es una herramienta que se relaciona con otra para agregarle una función nueva y normalmente especifica.

API

Application Programming Interface, es un conjunto de subrutinas, métodos, funciones y procedimientos que se ofrece a los desarrolladores para ser utilizado por otro software como una capa de abstracción.

Hosting

En Informática, es el servicio que provee a los usuarios de Internet un sistema para poder almacenar información, imágenes, video, o cualquier contenido accesible vía web.

Domótica

Sistemas capaces de automatizar una vivienda o edificación de cualquier tipo.

CMS

Content Management System, Son programas informáticos que permiten crear una estructura de soporte para la creación y administración de contenidos, principalmente de aplicaciones web.

Bootstrap

Framework que se utiliza para la creación de aplicaciones web Responsive o adaptables a los dispositivos que las visualizan.

Botnet

Conjunto de robots informáticos que se ejecutan de manera autónoma y automática, normalmente también pueden ser controlados mediante comandos enviados de manera remota por su creador.

WebSocket

Es una tecnología que proporciona un canal de comunicación bidireccional y full-dúplex sobre un único socket TCP.

HTTP

HyperText Transfer Protocol, es el protocolo de comunicación que permite las transferencias de información en la World Wide Web.

SSL

Secure Sockets Layer, es un protocolo criptográfico que proporciona comunicaciones seguras por una red, comúnmente Internet.

Criptología

Disciplina científica que se dedica al estudio de mensajes procesados de cierta manera para que se conviertan en difíciles o imposibles de leer por entidades no autorizadas.

URL

Uniform Resource Locator, es un identificador de recursos uniforme cuyos recursos referidos pueden cambiar. Están formados por una secuencia de caracteres, de acuerdo a un modelo y estándar, que designa recursos en una red, como Internet.

URI

Uniform Resource Identifier, es un identificador de recursos que se utiliza para identificar recursos de una red de manera univoca.

Cookie

En Informática, es una pequeña información enviada por una aplicación web y almacenada por el navegador.

Token de sesión

Es un identificador único de una sesión, asignado por una aplicación web a un navegador o cliente, el cual se ha autentificado previamente. Se suelen requerir estos tokens de sesión cuando el usuario quiere hacer algún tipo de interacción importante con la aplicación web, como una transacción, para verificar que es el usuario autentificado el que está realizando dicha petición y no otra persona.

Ingeniería social

Es un método no técnico, para obtener información sobre un sistema, basado en el factor humano y que tiende a explotar la tendencia natural de las personas a confiar y ayudar a otras personas.

Script

Suele ser un programa usualmente simple, que por lo general se almacena en texto plano y casi siempre son interpretados y no compilados.

Hacktivist

Hacker que utiliza la tecnología y sus conocimientos para reivindicar sus ideas.

Shell Remote

Es un programa de ordenador que recibe órdenes mediante línea de comandos para ejecutar dichas órdenes en el ordenador. Normalmente el usuario que envía estas órdenes desde un ordenador las envía a otro ordenador separado y en redes distintas.

MetaExploit

Es un framework que se utiliza para explotar vulnerabilidades.

Fingerprinter

En Seguridad Informática, se utiliza para detectar las versiones de los sistemas operativos.

Ofuscador de código

En computación, la ofuscación se refiere a la acción de cambiar un código fuente, un programa o código maquina, sin que esta pierda su funcionalidad, con el fin de que no sea fácil de entender o leer.

CONTENIDO DEL CD

En el contenido del CD que acompaña a la memoria podemos encontrar los siguientes recursos:

- Memoria del trabajo en los formatos PDF, DOCX y DOC dentro del directorio Memoria.
- Código fuente del trabajo dentro del directorio Código fuente.
- Libros y artículos a los que se ha hecho referencia durante la memoria y que se han utilizado como bibliografía. Los cuales podemos encontrar en el directorio Bibliografía.
- Páginas Web que han servido de bibliografía. Las podemos encontrar dentro del directorio Bibliografía/Enlaces Web.
- Manual de usuario de la aplicación junto con ejemplos, que podemos encontrar en el directorio Manual y ejemplos.

ANEXO A. **CÓDIGO FUENTE DE LA EXTENSIÓN**

En este anexo se muestra el código fuente de la extensión desarrollada para Burp Suite especificado en el capítulo 6.

```
#!/usr/bin/jython
# -*- coding: utf-8 -*-

# This work is licensed under a {CC-BY} Creative Commons
Attribution 4.0 International License.
# You are free to:
# - Share: copy and redistribute the material in any medium
or format.
# - Adapt: remix, transform, and build upon the material for
any purpose, even commercially.
# Under the following terms:
# - Attribution: You must give appropriate credit, provide a
link to the license, and indicate if changes were made.
#    You may do so in any reasonable manner, but not in any
way that suggests the licensor endorses you or your use.
# Notices:
# - You do not have to comply with the license for elements
of the material in the public domain or where your use is
permitted by an applicable exception or limitation.
# - No warranties are given. The license may not give you all
of the permissions necessary for your intended use.
#   For example, other rights such as publicity, privacy, or
moral rights may limit how you use the material.

from burp import IBurpExtender
```

```
from burp import ITab
from burp import IHttpListener
from burp import IScannerCheck

from java.awt import Component, GridLayout, GridBagLayout,
GridBagConstraints, Insets, Toolkit, Color, Font
from java.awt.datatransfer import Clipboard, StringSelection
from javax.swing import JButton, JTabbedPane, JPanel, JLabel,
JTextField, JTextArea, JScrollPane, SwingConstants,
JSeparator, JSlider, JSplitPane, JTable, JOptionPane,
JCheckBox
from javax.swing.table import DefaultTableModel
from java.net import URL

class Masher():
     def __init__(self, minpwd, maxpwd, specchars,
transfor_bool):
          # Anios para adherir
          self._years = ['2016', '2017', '2010']
          # Caracteres especiales
          self._chars = specchars
          # Longitud de la pwd (p.e. entre 6 y 12)
          self._wlfrom = minpwd
          self._wlto = maxpwd

          self._reverse_year = '7'
          self._threads = 50
          self._numfrom = 0
          self._numto = 20
          if transfor_bool:
               self._aa = '4'
               self._ee = '3'
               self._ii = '1'
               self._oo = '0'
               self._tt = '7'
               self._ss = '5'
               self._gg = '9'
               self._zz = '2'
          else:
               self._aa = 'a'
               self._ee = 'e'
               self._ii = 'i'
               self._oo = 'o'
               self._tt = 't'
               self._ss = 's'
               self._gg = 'g'
               self._zz = 'z'

     def concatenar(self, seq, start, stop):
          for str1 in seq:
               for num1 in xrange(start,stop):
```

```
                    yield str1 + str(num1)
     def combinar(self, seq, start, schar = ""):
          for str1 in seq:
               for str2 in start:
                    yield str1 + schar + str2
     def mashup(self, lista_palabras, FNacimiento):
          rdo_wpares = ['']
          rdo_schars = ['']
          rdo_xrepl = ['']
          rdo_simp = ['']
          rdo_simpver = ['']

          # Concatenamos las palabras con una char especial
en medio.
          for spec in self._chars:
               for w1 in lista_palabras:
                    for w2 in lista_palabras:
                         if lista_palabras.index(w1) !=
lista_palabras.index(w2):
                              rdo_simp.append(w1+spec+w2)
          # Concatenamos las palabras en pares.
          for w1 in lista_palabras:
               for w2 in lista_palabras:
                    if lista_palabras.index(w1) !=
lista_palabras.index(w2):
                         rdo_wpares.append(w1+w2)
          # generamos combinaciones de los caracteres
especiales.
          for w1 in self._chars:
               rdo_schars.append(w1)
               for w2 in self._chars:
                    rdo_schars.append(w1+w2)
                    for w3 in self._chars:
                         rdo_schars.append(w1+w2+w3)
          # Distintas combinaciones para la Fecha nacimiento.
          if FNacimiento != "":
               FN_yy = FNacimiento[-2:]
               FN_yyy = FNacimiento[-3:]
               FN_yyyy = FNacimiento[-4:]
               FN_xd = FNacimiento[1:2]
               FN_xm = FNacimiento[3:4]
               FN_dd = FNacimiento[:2]
               FN_mm = FNacimiento[2:4]
               self._reverse_year = FN_yyyy[::-1]
               FNcompos = [FN_yy, FN_yyy, FN_yyyy, FN_xd,
FN_xm, FN_dd, FN_mm, self._reverse_year]
               FNcombos = ['']
               for w1 in FNcompos:
                    FNcombos.append(w1)
                    for w2 in FNcompos:
```

```
                        if FNcompos.index(w1) !=
FNcompos.index(w2):
                              FNcombos.append(w1+w2)
                              for w3 in FNcompos:
                                   if FNcompos.index(w1) !=
FNcompos.index(w2) and FNcompos.index(w2) !=
FNcompos.index(w3) and FNcompos.index(w1) !=
FNcompos.index(w3):

     FNcombos.append(w1+w2+w3)

          # Concatenamos las palabras con un char especial y
año inverso.
          for spec in self._chars:
               for w1 in lista_palabras:

     rdo_simpver.append(w1+spec+self._reverse_year)

          # concatenaciones y combinaciones extras.
          combo1 = list(self.combinar(lista_palabras,
self._years))
          combo2 = list(self.combinar(rdo_wpares,
self._years))
          combo3 = list(self.combinar(lista_palabras,
rdo_schars))
          combo4 = list(self.combinar(rdo_wpares,
rdo_schars))
          combo5 = list(self.concatenar(lista_palabras,
self._numfrom, self._numto))
          combo6 = list(self.concatenar(rdo_wpares,
self._numfrom, self._numto))

          combo1_uniq = dict.fromkeys(combo1).keys()
          combo2_uniq = dict.fromkeys(combo2).keys()
          combo3_uniq = dict.fromkeys(combo3).keys()
          combo4_uniq = dict.fromkeys(combo4).keys()
          combo5_uniq = dict.fromkeys(combo5).keys()
          combo6_uniq = dict.fromkeys(combo6).keys()
          combo7_uniq = dict.fromkeys(lista_palabras).keys()
          combo8_uniq = dict.fromkeys(rdo_wpares).keys()
          combo20_uniq = dict.fromkeys(rdo_simp).keys()
          combo21_uniq = dict.fromkeys(rdo_simpver).keys()

          dict_temp =
combo20_uniq+combo21_uniq+combo1_uniq+combo2_uniq+combo3_uniq
+combo4_uniq+combo5_uniq+combo6_uniq+combo7_uniq+combo8_uniq

          if FNacimiento != "":
               combo9 = list(self.combinar(lista_palabras,
FNcombos))
```

```
                combo10 = list(self.combinar(rdo_wpares,
FNcombos))
                combo9_uniq = dict.fromkeys(combo9).keys()
                combo10_uniq = dict.fromkeys(combo10).keys()
                dict_temp += combo9_uniq+combo10_uniq

            dict_rem = dict.fromkeys(dict_temp).keys()

            # Remplazamos las letras por numeros en otra tanda.
            for wx in dict_rem:
                wx = wx.replace('a', self._aa)
                wx = wx.replace('e', self._ee)
                wx = wx.replace('i', self._ii)
                wx = wx.replace('o', self._oo)
                wx = wx.replace('t', self._tt)
                wx = wx.replace('s', self._ss)
                wx = wx.replace('g', self._gg)
                wx = wx.replace('z', self._zz)
                rdo_xrepl.append(wx)

            rdo = dict_rem + rdo_xrepl
            # Eliminamos blancos
            while "" in rdo: rdo.remove("")
            # Eliminamos combinaciones menores que wlfrom y
mayores que wlto
            rdo_rdo = [x for x in rdo if len(x) >= self._wlfrom
and len(x) <= self._wlto]
            return rdo_rdo

class Mi_Extension(IHttpListener, IScannerCheck):

    def __init__(self, extender):
            self._callbacks = extender._callbacks
            self._helpers = extender._callbacks.getHelpers()
            self._callbacks.registerScannerCheck(self)
            # Creamos el contenedor de paneles.
            self.contenedor = JTabbedPane()

            # Campos del sub-tab 1 (mash up)
            self._tab1_nombre = JTextField()
            self._tab1_apellido = JTextField()
            self._tab1_FNacimiento = JTextField()
            self._tab1_mascota = JTextField()
            self._tab1_otro = JTextField()

            self._tab1_feedback_ta = JTextArea('This may take a
while . . .')
            self._tab1_feedback_ta.setEditable(False)
            self._tab1_feedback_sp =
JScrollPane(self._tab1_feedback_ta)
```

```
        self._tab1_minsize = JSlider(4,16,6)
        self._tab1_minsize.setMajorTickSpacing(1)
        self._tab1_minsize.setPaintLabels(True)

        self._tab1_maxsize = JSlider(4,16,10)
        self._tab1_maxsize.setMajorTickSpacing(1)
        self._tab1_maxsize.setPaintLabels(True)

        self._tab1_specialchars = JTextField('!,@,#,$,&,*')
        self._tab1_transformations = JCheckBox('1337 mode')
        self._tab1_firstcapital = JCheckBox('first capital
letter')

        # Campos del sub-tab 2 (redirect)
        self._tab2_JTFa = JTextField()
        self._tab2_JTFaa = JTextField()
        self._tab2_JTFb = JTextField()
        self._tab2_JTFbb = JTextField()
        self._tab2_boton = JButton(' Redirect is Off ',
actionPerformed=self.switch_redirect)
        self._tab2_boton.background = Color.lightGray
        self._tab2_encendido = False

        # Campos del sub-tab 3 (loader)
        self._tab3_urls_ta = JTextArea(15, 5)
        self._tab3_urls_sp =
JScrollPane(self._tab3_urls_ta)

        # Campos del sub-tab 4 (headers)
        self._tab4_tabla_model = DefaultTableModel()
        self._tab4_headers_dict = {}

        # Campos del sub-tab 5 (reverse ip)
        self._tab5_target = JTextArea(15, 5)
        self._tab5_target_sp =
JScrollPane(self._tab5_target)

    def create(self):

        # Estilo general para todas las sub-tab.
        gBC = GridBagConstraints()
        gBC.fill = GridBagConstraints.BOTH
        gBC.ipadx = 5
        gBC.ipady = 5
        gBC.insets = Insets(0,5,5,5)
        gBC.weightx = 0.5
        #gBC.weighty = 0.7

        ######################################
        ###   Creamos la primera sub-tab. (MASHUP)
        ######################################
```

```
        tab_1 = JPanel(GridBagLayout())
        tab_1_jlabelAyuda = JLabel('<html><i>&#8226Tip:
This Mashup receive one or more keywords in order to generate
a list of possible passwords</i></html>')
        tab_1_jlabelAyuda.setFont(Font("Serif", 0, 12))

        gBC.gridx = 0
        gBC.gridy = 0
        tab_1.add(JLabel('<html><font
color=green><i>Income:</i></font></html>'), gBC)

        gBC.gridy = 1
        tab_1.add(JLabel('<html><b>&#8226
Name:</b></html>'), gBC)
        gBC.gridy = 2
        tab_1.add(self._tab1_nombre, gBC)

        gBC.gridy = 3
        tab_1.add(JLabel('<html><b>&#8226
Surname:</b></html>'), gBC)
        gBC.gridy = 4
        tab_1.add(self._tab1_apellido, gBC)

        gBC.gridy = 5
        tab_1.add(JLabel('<html><b>&#8226 Birthdate:
(DDMMYYYY)</b></html>'), gBC)
        gBC.gridy = 6
        tab_1.add(self._tab1_FNacimiento, gBC)

        gBC.gridy = 7
        tab_1.add(JLabel('<html><b>&#8226
Pet:</b></html>'), gBC)
        gBC.gridy = 8
        tab_1.add(self._tab1_mascota, gBC)

        gBC.gridy = 9
        tab_1.add(JLabel('<html><b>&#8226
Anyother:</b></html>'), gBC)
        gBC.gridy = 10
        tab_1.add(self._tab1_otro, gBC)

        gBC.gridy = 11
        tab_1.add(JLabel('<html><b>&#8226 Passwd Min
Size:</b></html>'), gBC)
        gBC.gridy = 12
        tab_1.add(self._tab1_minsize, gBC)

        gBC.gridy = 13
        tab_1.add(JLabel('<html><b>&#8226 Passwd Max
Size:</b></html>'), gBC)
        gBC.gridy = 14
```

```
        tab_1.add(self._tab1_maxsize, gBC)

        gBC.gridy = 15
        tab_1.add(JLabel('<html><b>&#8226 Especial Chars:
(comma separated)</b></html>'), gBC)
        gBC.gridy = 16
        tab_1.add(self._tab1_specialchars, gBC)

        gBC.gridy = 17
        tab_1.add(self._tab1_transformations, gBC)

        gBC.gridy = 18
        tab_1.add(self._tab1_firstcapital, gBC)

        gBC.gridy = 19
        tab_1.add(JButton('Mashup!',
actionPerformed=self.mashup), gBC)

        gBC.gridy = 20
        gBC.gridwidth = 3
        tab_1.add(JSeparator(), gBC)
        gBC.gridwidth = 1

        gBC.gridy = 21
        gBC.gridwidth = 3
        tab_1.add(tab_1_jlabelAyuda, gBC)
        gBC.gridwidth = 1

        gBC.gridx = 1
        gBC.gridy = 0
        gBC.gridheight = 20
        gBC.weightx = 0
        tab_1.add(JSeparator(SwingConstants.VERTICAL), gBC)
        gBC.gridheight = 1
        gBC.weightx = 0.5

        gBC.gridx = 2
        gBC.gridy = 0
        tab_1.add(JLabel('<html><font
color=green><i>Outcome:</i></font></html>'), gBC)

        gBC.gridy = 1
        gBC.gridwidth = 2
        gBC.gridheight = 18
        tab_1.add(self._tab1_feedback_sp, gBC)
        gBC.gridwidth = 1
        gBC.gridheight = 1

        gBC.gridy = 19
        tab_1.add(JButton('Copy to clipboard!',
actionPerformed=self.cpy_clipboard), gBC)
```

```
        ######################################
        ###   Creamos la segunda sub-tab. (REDIRECT)
        ######################################
        tab_2 = JPanel(GridBagLayout())
        tab_2_jlabelAyuda = JLabel('<html><i>&#8226Tip:
This Redirect receive a pair x,y of hosts in order to
redirect from x to y.</i></html>')
        tab_2_jlabelAyuda.setFont(Font("Serif", 0, 12))

        gBC.gridx = 0
        gBC.gridy = 0
        tab_2.add(JLabel('<html><b>&#8226 From: (i.e.
www.facebook.com)</b></html>'), gBC)
        gBC.gridx = 1
        gBC.gridy = 0
        tab_2.add(self._tab2_JTFa, gBC)

        gBC.gridx = 2
        gBC.gridy = 0
        tab_2.add(JLabel('<html><b>&#8226 To: (i.e.
www.myhomepage.es)</b></html>'), gBC)
        gBC.gridx = 3
        gBC.gridy = 0
        tab_2.add(self._tab2_JTFaa, gBC)

        gBC.gridx = 0
        gBC.gridy = 1
        gBC.gridwidth = 4
        tab_2.add(JSeparator(), gBC)
        gBC.gridwidth = 1

        gBC.gridx = 0
        gBC.gridy = 2
        tab_2.add(JLabel('<html><b>&#8226
From:</b></html>'), gBC)
        gBC.gridx = 1
        gBC.gridy = 2
        tab_2.add(self._tab2_JTFb, gBC)

        gBC.gridx = 2
        gBC.gridy = 2
        tab_2.add(JLabel('<html><b>&#8226 To:</b></html>'),
gBC)
        gBC.gridx = 3
        gBC.gridy = 2
        tab_2.add(self._tab2_JTFbb, gBC)

        gBC.gridx = 0
        gBC.gridy = 3
        gBC.gridwidth = 4
```

```
        tab_2.add(self._tab2_boton, gBC)
        gBC.gridwidth = 1

        gBC.gridx = 0
        gBC.gridy = 4
        gBC.gridwidth = 4
        gBC.insets = Insets(100,10,5,10)
        tab_2.add(JSeparator(), gBC)
        gBC.gridwidth = 1
        gBC.insets = Insets(5,10,5,10)

        gBC.gridx = 0
        gBC.gridy = 5
        gBC.gridwidth = 4
        tab_2.add(tab_2_jlabelAyuda, gBC)
        gBC.gridwidth = 1

        ######################################
        ###   Creamos la tercera sub-tab. (LOADER)
        ######################################
        tab_3 = JPanel(GridBagLayout())
        tab_3_jlabelAyuda = JLabel('<html><i>&#8226Tip:
This Loader receive a List of Hosts or IPs that will be add
to the Burp Scope.</i></html>')
        tab_3_jlabelAyuda.setFont(Font("Serif", 0, 12))

        gBC.gridx = 0
        gBC.gridy = 0
        tab_3.add(JLabel('<html><font color=green><i>List
of targets: (i.e.
http://www.mytargetweb.com)</i></font></html>'), gBC)
        gBC.gridy = 1
        gBC.gridheight = 10
        gBC.weighty = 0.5
        tab_3.add(self._tab3_urls_sp, gBC)
        gBC.gridheight = 1
        gBC.weighty = 0
        gBC.gridy = 11
        tab_3.add(JButton('Load Into Target => Scope!',
actionPerformed=self.loader), gBC)

        gBC.gridy = 12
        gBC.gridwidth = 5
        gBC.insets = Insets(100,10,5,10)
        tab_3.add(JSeparator(), gBC)
        gBC.gridwidth = 1
        gBC.insets = Insets(5,10,5,10)

        gBC.gridy = 13
        tab_3.add(tab_3_jlabelAyuda, gBC)
```

```
        ######################################
        ###   Creamos la cuarta sub-tab. (HEADERS)
        ######################################
        tab_4_jlabelAyuda = JLabel("<html><i>&#8226Tip:
This headers record all unique headers that appear in every
host, check out for the security headers.</i></html>")
        tab_4_jlabelAyuda.setFont(Font("Serif", 0, 12))

        tab_4_jLabelRecomendacion =
JLabel("<html>&#8226<b>Server</b>: Don't give away much
information.<br> &#8226<b>Content-Security-Policy</b>:
Protect your site from XSS attacks by whitelisting sources of
approved content.<br> &#8226<b>Strict-Transport-Security</b>:
Enforce the browser to use HTTPS.<br> &#8226<b>Public-Key-
Pins</b>: Protect your site from MITM attacks.<br>
&#8226<b>X-Content-Type-Options</b>: the valid value is ->
nosniff .<br> &#8226<b>X-Frame-Options</b>: tells the browser
whether you want to allow your site to be framed or not.<br>
&#8226<b>X-XSS-Protection</b>: the best value is -> 1;
mode=block .</html>")
        tab_4_jLabelRecomendacion.setFont(Font("Dialog", 0,
13))

        tab_4 = JPanel(GridBagLayout())
        splitpane = JSplitPane(JSplitPane.VERTICAL_SPLIT)
        tab_4_top = JPanel(GridBagLayout())
        tab_4_bottom = JPanel(GridBagLayout())

        gBC_table = GridBagConstraints()
        gBC_table.fill = GridBagConstraints.BOTH
        gBC_table.ipadx = 5
        gBC_table.ipady = 5
        gBC_table.insets = Insets(5,10,5,10)
        gBC_table.weightx = 1
        gBC_table.weighty = 1

        tabla_datos = []
        tabla_headers =
('Severity','Header','Value','Host')
        self._tab4_tabla_model =
DefaultTableModel(tabla_datos, tabla_headers)
        tabla_ej = JTable(self._tab4_tabla_model)
        tabla_example = JScrollPane(tabla_ej,
JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED)

        gBC_table.gridx = 0
        gBC_table.gridy = 0
        gBC_table.gridheight = 5
        tab_4_top.add(tabla_example, gBC_table)
        gBC_table.gridheight = 1
```

```
        gBC_table.weightx = 0.5
        gBC_table.weighty = 0
        gBC_table.gridwidth = 2
        gBC_table.gridx = 0
        gBC_table.gridy = 0
        tab_4_bottom.add(JSeparator(), gBC_table)
        gBC_table.gridy = 1
        tab_4_bottom.add(tab_4_jlabelAyuda, gBC_table)
        gBC_table.gridy = 2
        tab_4_bottom.add(tab_4_jLabelRecomendacion,
gBC_table)

        splitpane.setTopComponent(tab_4_top)
        splitpane.setBottomComponent(tab_4_bottom)
        gBC_table.weightx = 1
        gBC_table.weighty = 1
        gBC_table.gridx = 0
        gBC_table.gridy = 0
        tab_4.add(splitpane,gBC_table)

        ######################################
        ###   Creamos la quinta sub-tab. (ACTIVE SCAN)
        ######################################
        tab_5 = JPanel(GridBagLayout())
        tab_5_jlabelAyuda = JLabel('<html><i>&#8226Tip:
This receive a list of targets and launch an active
scan.</i></html>')
        tab_5_jlabelAyuda.setFont(Font("Serif", 0, 12))
        tab_5_jlabelWarning = JLabel('<html><font
color=red><i>&#8226Warning: Active scanning generates large
numbers of requests which are malicious in form and which may
result in compromise of the application. You should use this
scanning mode with caution, only with the explicit permission
of the application owner. For more information, read the
documentation of active scan, Burp Suite.</font></i></html>')
        tab_5_jlabelWarning.setFont(Font("Dialog", 0, 13))

        gBC.gridx = 0
        gBC.gridy = 0
        tab_5.add(JLabel('<html><font color=green><i>List
of targets: (i.e.
http://192.168.1.1/index.html)</i></font></html>'), gBC)
        gBC.gridy = 1
        gBC.gridheight = 8
        gBC.weighty = 0.5
        tab_5.add(self._tab5_target_sp, gBC)
        gBC.gridheight = 1
        gBC.weighty = 0
        gBC.gridy = 9
```

```
        tab_5.add(JButton('Launch Scan!',
actionPerformed=self.do_active_scan), gBC)

        gBC.gridy = 10
        gBC.gridwidth = 5
        gBC.insets = Insets(100,10,5,10)
        tab_5.add(JSeparator(), gBC)
        gBC.gridwidth = 1
        gBC.insets = Insets(5,10,5,10)

        gBC.gridy = 11
        tab_5.add(tab_5_jlabelAyuda, gBC)
        gBC.gridy = 12
        tab_5.add(tab_5_jlabelWarning, gBC)

        ######################################
        ###   Creamos la ultima sub-tab.
        ######################################
        tab_about = JPanel(GridBagLayout())

        gBC_about = GridBagConstraints()
        gBC_about.fill = GridBagConstraints.HORIZONTAL
        gBC_about.ipadx = 5
        gBC_about.ipady = 5
        gBC_about.insets = Insets(5,10,5,10)
        gBC_about.weightx = 1
        gBC_about.weighty = 1

        Jlabel1 = JLabel('<html><b>Plug-in L-
Tools</b></html>')
        Jlabel1.setFont(Font("Dialog", 1, 18))
        jlabel2 = JLabel('<html>This Plug-in provides
utilities for a penetration testers, researchers and
developers, in order to support their work.</html>')
        jlabel3 = JLabel('<html><b>CC-BY 4.0, 2017. Gino
Angles</b></html>')
        jlabel3.setFont(Font("Dialog", 1, 14))
        jlabel4 = JLabel('<html><b>License</b></html>')
        jlabel4.setFont(Font("Dialog", 1, 14))
        jlabel5 = JLabel('<html><img alt="Licensed under a
Creative Commons BY" style="border-width:0"
src="https://licensebuttons.net/l/by/4.0/88x31.png"></html>')
        jlabel6 = JLabel('<html>This work is licensed under
a <a rel="license"
href="http://creativecommons.org/licenses/by/4.0/">Creative
Commons Attribution 4.0 International License</a>.</html>')
        jlabel7 =
JLabel('<html><b>Dependencies</b></html>')
        jlabel7.setFont(Font("Dialog", 1, 14))
        jlabel8 = JLabel('Jython +2.7')
        jlabel9 = JLabel('http://www.jython.org')
```

```
        gBC_about.gridx = 0
        gBC_about.gridy = 0
        tab_about.add(Jlabel1, gBC_about)
        gBC_about.gridy = 1
        tab_about.add(jlabel2, gBC_about)
        gBC_about.gridy = 2
        tab_about.add(jlabel3, gBC_about)

        gBC_about.gridy = 3
        gBC_about.gridwidth = 5
        tab_about.add(JSeparator(), gBC_about)
        gBC_about.gridwidth = 1

        gBC_about.gridy = 4
        tab_about.add(jlabel4, gBC_about)
        gBC_about.gridy = 5
        tab_about.add(jlabel5, gBC_about)
        gBC_about.gridy = 6
        tab_about.add(jlabel6, gBC_about)

        gBC_about.gridy = 7
        gBC_about.gridwidth = 5
        tab_about.add(JSeparator(), gBC_about)
        gBC_about.gridwidth = 1

        gBC_about.gridy = 8
        tab_about.add(jlabel7, gBC_about)
        gBC_about.gridy = 9
        tab_about.add(jlabel8, gBC_about)
        gBC_about.gridy = 10
        tab_about.add(jlabel9, gBC_about)

        # Añadimos los sub-tab al contenedor y lo
devolvemos.
        self.contenedor.addTab('Mashup', tab_1)
        self.contenedor.addTab('Redirect', tab_2)
        self.contenedor.addTab('Loader', tab_3)
        self.contenedor.addTab('Headers', tab_4)
        self.contenedor.addTab('Quick Scan', tab_5)
        self.contenedor.addTab('About', tab_about)

        return self.contenedor

    # Funcion que copia el contenido de texto al clipboard.
    def cpy_clipboard(self, event):
        toolkit = Toolkit.getDefaultToolkit()
        clipboard = toolkit.getSystemClipboard()

    clipboard.setContents(StringSelection(self._tab1_feedbac
k_ta.text), None)
```

```
            JOptionPane.showMessageDialog(self.contenedor,
"Output copied to clipboard", "Information",
JOptionPane.INFORMATION_MESSAGE)
            return

        # Funcion que instancia la clase Masher() y hace el
mashup del sub-tab1 (mashup).
        def mashup(self, event):
            if self._tab1_minsize.value >
self._tab1_maxsize.value:
                self._callbacks.issueAlert('Mashup=> min size
have to be bigger than max size.')
            else:
                estado_JCH_trans =
self._tab1_transformations.isSelected()
                especial_chars =
self._tab1_specialchars.text.split(",")
                mymash = Masher(self._tab1_minsize.value,
self._tab1_maxsize.value, especial_chars, estado_JCH_trans)
                reverse_name = self._tab1_nombre.text[::-1]
                if self._tab1_firstcapital.isSelected():
                    lista_palabras =
[self._tab1_nombre.text.title(),
self._tab1_apellido.text.title(),
self._tab1_FNacimiento.text, self._tab1_mascota.text.title(),
self._tab1_otro.text.title(), reverse_name.title()]
                else:
                    lista_palabras = [self._tab1_nombre.text,
self._tab1_apellido.text, self._tab1_FNacimiento.text,
self._tab1_mascota.text, self._tab1_otro.text, reverse_name]

                while "" in lista_palabras:
lista_palabras.remove("")
                rdo_rdo = mymash.mashup(lista_palabras,
self._tab1_FNacimiento.text)
                self._tab1_feedback_ta.text =
"\n".join(rdo_rdo)
            return

        # Funcion que activa/desactiva el redirect.
        def switch_redirect(self, event):
            if not self._tab2_encendido:
                self._tab2_boton.text = ' Redirect is On '
                self._tab2_boton.background = Color.gray
                self._tab2_encendido = True
                # Registramos la instancia del escuchador
http.
                self._callbacks.registerHttpListener(self)
            else:
                self._tab2_boton.text = ' Redirect is Off '
                self._tab2_boton.background = Color.lightGray
```

```
                self._tab2_encendido = False
                # Borramos la instancia del escuchador http.
                self._callbacks.removeHttpListener(self)
            return

        # @override
        def processHttpMessage(self, toolFlag, messageIsRequest,
messageInfo):
            if messageIsRequest and self._tab2_encendido:
                serviceHttp = messageInfo.getHttpService()
                if messageInfo.getHttpService().getHost() ==
self._tab2_JTFa.text:
                    if self._tab2_JTFaa.text == '':
self._tab2_JTFaa.text = 'localhost'

        messageInfo.setHttpService(self._helpers.buildHttpServic
e(self._tab2_JTFaa.text, serviceHttp.getPort(),
serviceHttp.getProtocol()))
                elif messageInfo.getHttpService().getHost() ==
self._tab2_JTFb.text:
                    if self._tab2_JTFbb.text == '':
self._tab2_JTFbb.text = 'localhost'

        messageInfo.setHttpService(self._helpers.buildHttpServic
e(self._tab2_JTFbb.text, serviceHttp.getPort(),
serviceHttp.getProtocol()))
            return

        # Funcion que carga el textArea de URL a Scope.
        def loader(self, event):
            lista_url = self._tab3_urls_ta.text.split("\n")
            for newUrl in lista_url:
                self._callbacks.includeInScope(URL(newUrl))
            return

        # @override
        def doPassiveScan(self, request_response):
            _response = request_response.getResponse()
            _request = request_response.getRequest()
            _httpService = request_response.getHttpService()
            _url = request_response.getHttpService().getHost()
            info_response =
self._helpers.analyzeResponse(_response)
            info_request =
self._helpers.analyzeRequest(_httpService, _request)
            headers = info_response.getHeaders()
            _fullUrl = info_request.getUrl().toString()

            for head in headers:
                head_w1 = head.split("\n")
                for head_w2 in head_w1:
```

```
                    head_w3 = head_w2.split(":")
                    if len(head_w3) > 1:
                         head_w3[0].encode('ascii','ignore')
                         head_w3[1].encode('ascii','ignore')

                         key = "%s-%s" % (_url, head_w3[0])

                         if key not in
self._tab4_headers_dict:
                              critical =
self.analizeHeader(head_w3[0])

                              fila_tabla =
[critical,head_w3[0],head_w3[1],_fullUrl]

                              self._tab4_headers_dict[key] =
fila_tabla

     self._tab4_tabla_model.addRow(fila_tabla)
          return

     # devuelve cuan importante es la cabecera HTTP.
     def analizeHeader(self, header):
          if header == "Content-Security-Policy":
               return "High"
          elif header == "X-Content-Type-Options":
               return "High"
          elif header == "X-Frame-Options":
               return "High"
          elif header == "X-XSS-Protection":
               return "High"
          elif header == "Strict-Transport-Security":
               return "High"
          elif header == "Public-Key-Pins":
               return "High"
          elif header == "Access-Control-Allow-Origin":
               return "High"
          elif header == "X-Permitted-Cross-Domain-Policies":
               return "High"
          elif header == "X-Powered-By":
               return "Information"
          elif header == "Server":
               return "Information"
          elif header == "Referrer-Policy":
               return "Medium"
          else:
               return " "

     # realiza un escaner al host/s apuntado.
     def do_active_scan(self, event):
          _lista_urls = self._tab5_target.text.split("\n")
```

```
        for _x_url in _lista_urls:
            _target_url = URL(_x_url)
            _target_request =
self._helpers.buildHttpRequest(_target_url)

            _host = _target_url.getHost()
            _port = _target_url.getDefaultPort()
            _isHttps = True if
_target_url.getProtocol()=="https" else False

            try:
                self._callbacks.doActiveScan(_host,
_port, _isHttps, _target_request)
            except:
                pass
        return

class BurpNewTab(ITab):

    def __init__(self, extender):
        self._callbacks = extender._callbacks

    # @override
    def getUiComponent(self):
        #Creamos el contenido de la extension
        contenedor = Mi_Extension(self).create()
        #Le damos el aspecto propio de la GUI de BurpSuite
        self._callbacks.customizeUiComponent(contenedor)
        return contenedor

    # @override
    def getTabCaption(self):
        return "L-Tools"

class BurpExtender(IBurpExtender):

    # @override
    def registerExtenderCallbacks(self, callbacks):

        # keep a reference to our callbacks object
        self._callbacks = callbacks

        # obtain an extension helpers object
        self._helpers = callbacks.getHelpers()

        # Damos nombre a la extensión
        callbacks.setExtensionName("L-Tools")

        # Instanciamos la clase BurpNewTab para incluirla
        tab = BurpNewTab(self)
        callbacks.addSuiteTab(tab)
```

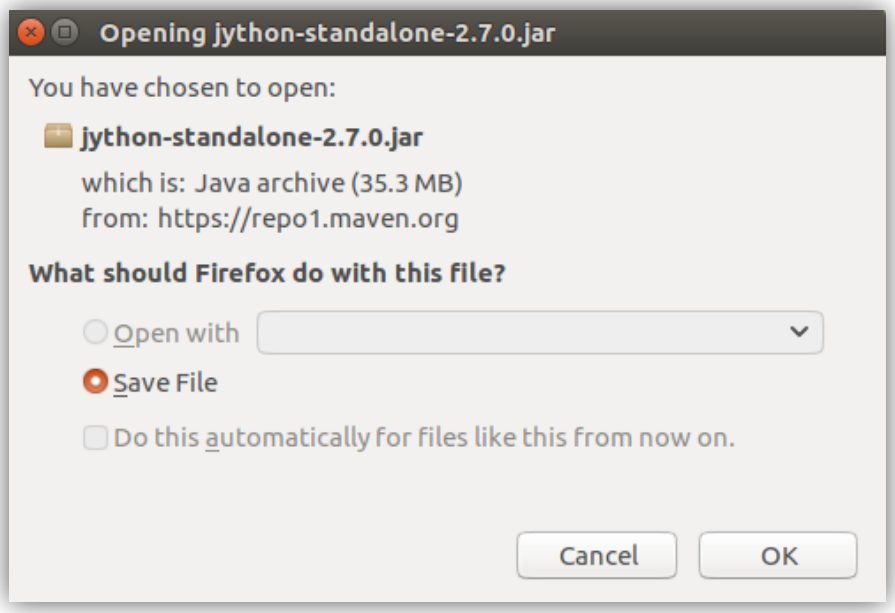
```
print 'Plug-in L-Tools v1.0  Loaded!'
return
```

ANEXO B. **INSTALACIÓN DE PLUG-IN**

En este anexo resumiré como se realiza la instalación de Jython el cual es necesario para poder instalar la extensión L-Tools, todo ello en Burp Suite.

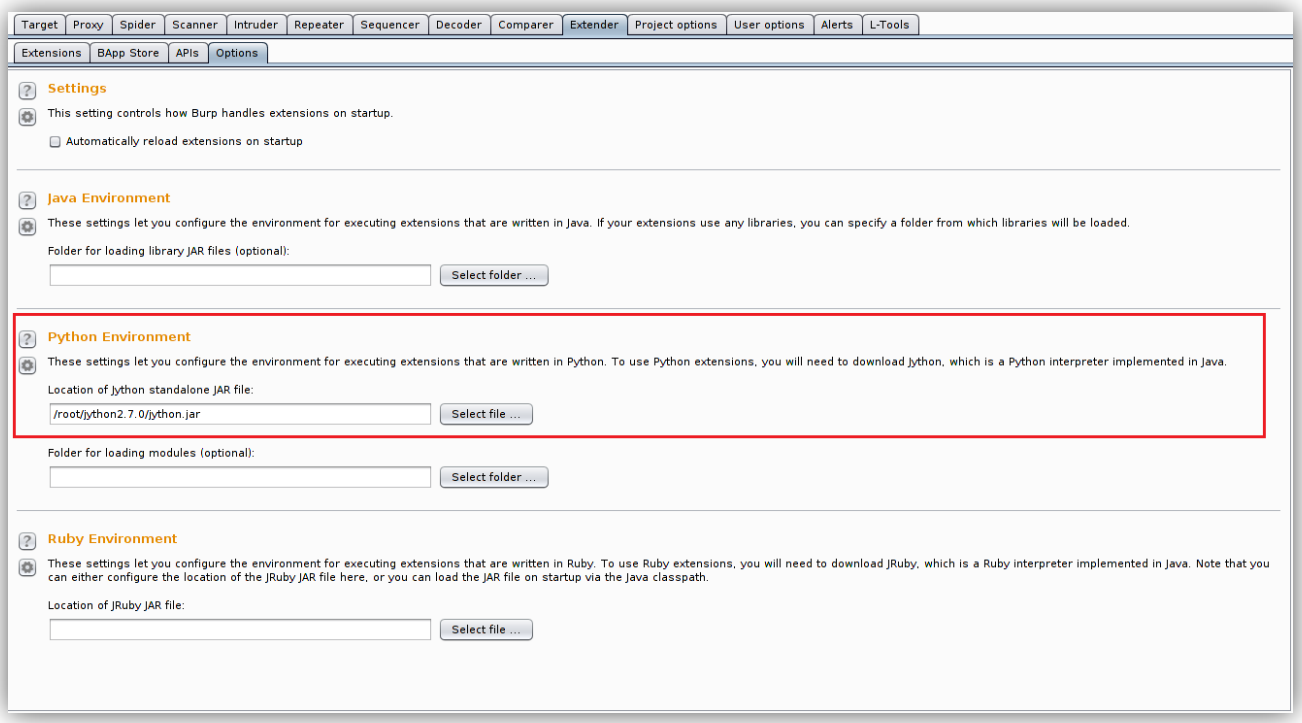
B.1 INSTALACIÓN JYTHON

Entramos en www.jython.org, vamos a la sección de descargas, y nos bajamos la última versión disponible de Jython (es necesario que sea una versión superior a la 2.7). Nos dan a elegir entre el instalador y el "standalone.jar". Podéis usar cualquiera de los dos.



Una vez descargado, lo instalamos, nos creara una carpeta con el nombre de *Jython_version*. Dentro de la carpeta habrá un fichero llamado *Jython.jar*.

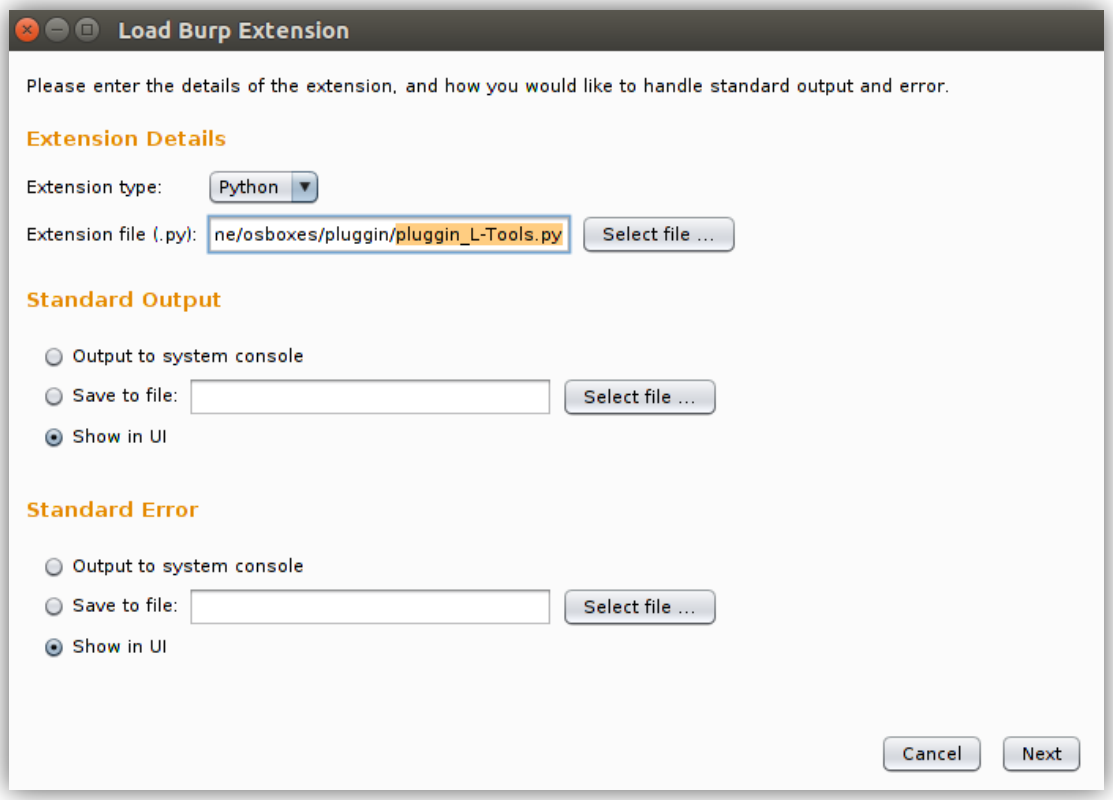
Abrimos Burp Suite, vamos a la pestaña de Extender => Options, y vemos que hay un apartado para configurar el entorno de Python, en ese cuadro de texto, tenemos que cargar el fichero *Jython.jar*.



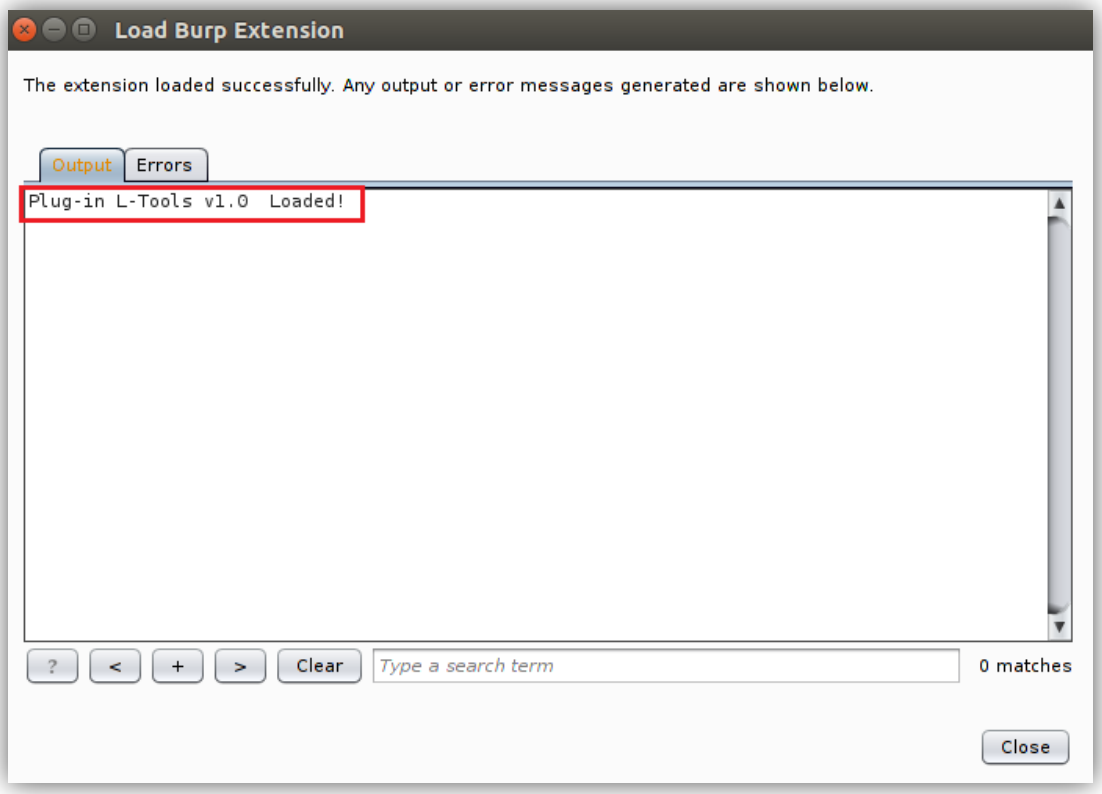
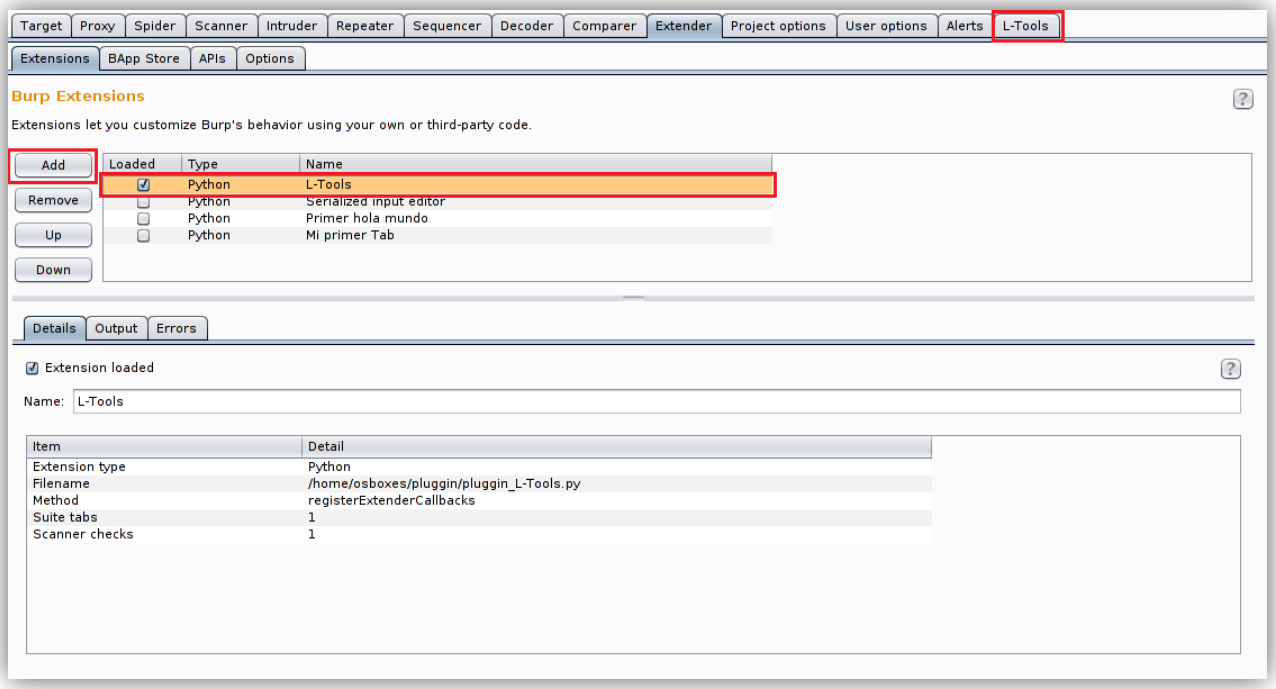
De este modo, cada vez que carguemos una extensión en Python, Burp Suite sabrá el interprete/compilador, que tiene que utilizar para cargar nuestra extensión.

B.2 INSTALACIÓN L-TOOLS

Una vez tenemos cargado el *Jython.jar* en Burp Suite, podemos cargar extensiones en ficheros Python. Si nos vamos a Extender => Extensions, veremos que aparece un botón "ADD" para añadir una nueva extensión.



Dentro del modal que ha aparecido, tenemos que seleccionar el tipo de fichero que queremos cargar, en este caso, Python. Luego seleccionamos el fichero que contiene nuestro código fuente y le damos a siguiente.



A continuación realizara el proceso de carga, y nos enseñara el siguiente mensaje, y cuando cerremos, veremos que ha aparecido una nueva extensión y una nueva pestaña.