

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de
Telecomunicación

Ataques a servidores web: estudio experimental de la
capacidad de detección de SIDS

Autor: Felipe Bueno Carranza

Tutor: Francisco Javier Muñoz Calle

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

Sevilla, 2022



Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Ataques a servidores web: estudio experimental de la capacidad de detección de SIDS

Autor:

Felipe Bueno Carranza

Tutor:

Francisco Javier Muñoz Calle

Profesor colaborador

Dpto. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2022

Proyecto Fin de Carrera: Ataques a servidores web: estudio experimental de la capacidad de detección de SIDS

Autor: Felipe Bueno Carranza

Tutor: Francisco Javier Muñoz Calle

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2022

El Secretario del Tribunal

Agradecimientos

Este trabajo no es más que un reflejo de todos los años de trabajo y sacrificio que ha resultado para mí la consecución de este Grado. Gran parte de este éxito pertenece a mi padre Felipe, por dar un ejemplo perfecto de lo que es ser una buena persona, y mi madre Lola, que durante todos estos años ha funcionado como el principal impulso que uno necesita en los momentos más duros, la mitad del título es suyo.

También quiero hacer una mención especial a mi hermana Ana, ejemplo de trabajo duro y constancia, y a Leti, que siempre me ayudó a focalizarme en mi objetivo.

Además, me gustaría agradecer a todos y cada uno de mis amigos de Moraleja y de Sevilla, por hacerme pertenecer prácticamente a dos hogares.

Por último, gracias a mi tutor, Javier, por los conocimientos que he adquirido desarrollando este trabajo, así como su atención semanal.

Felipe Bueno Carranza

Sevilla, 2022

Resumen

Este trabajo pretende cuantificar de forma experimental el nivel de detección de ataques de algunos de los detectores de intrusiones basados en firmas (SIDS) disponibles de forma gratuita. Para su evaluación se ha utilizado un *dataset* de ataques generado a partir de la búsqueda y selección de herramientas de generación de ataques y de análisis de seguridad del servicio web. Se consideran dos escenarios de diferente complejidad en cuanto al servicio web proporcionado. Las peticiones HTTP registradas durante los ataques serán la entrada a tres SIDS gratuitos seleccionados por su amplio uso, de forma que se podrá determinar la capacidad de detección de estos.

Abstract

This work aims to experimentally quantify the attack detection level of some of the freely available signature-based intrusion detectors (SIDS). For its evaluation, a *dataset* of attacks generated from the search and selection of tools for generating attacks and security analysis of the web service has been used. Two scenarios of different complexity are considered in terms of the web service provided. The HTTP requests recorded during the attacks will be the input to three free SIDS selected for their wide use, so that their detection capacity can be determined.

Agradecimientos	7
Resumen	9
Abstract	11
Índice	13
Índice de Tablas	15
Índice de Figuras	17
1 Introducción	11
2 Objetivos del Trabajo	13
2.1 <i>Objetivos principales</i>	13
2.2 <i>Objetivos específicos</i>	13
3 Conocimientos Previos	15
3.1 <i>Ataques a servidores web: Clasificación de vulnerabilidades de OWASP</i>	15
3.1.1 Broken Access Control	15
3.1.2 Cryptografic Failures	16
3.1.3 Injection	16
3.1.4 Insecure Design	16
3.1.5 Security Misconfiguration	17
3.1.6 Vulnerable and Outdated Components	17
3.1.7 Identification and Authentication Failures	18
3.1.8 Software and Data Integrity Failures	18
3.1.9 Security Logging and Monitoring Failures	18
3.1.10 Server-Side Request Forgery (SSRF)	19
3.2 <i>Herramientas de análisis de vulnerabilidades web</i>	19
3.3 <i>Herramientas de captura de tráfico</i>	19
3.3.1 Wireshark	19
3.3.2 nfdump	20
3.4 <i>Firewall de Aplicaciones Web</i>	21
3.4.1 ModSecurity	21
3.4.2 Nemesida	22
3.5 <i>IDS: Sistema de detección de intrusiones</i>	23
3.5.1 Snort	23
3.6 <i>InspectorLog</i>	24
4 Desarrollo del Estudio	25
4.1 <i>Preparación del entorno</i>	25
4.2 <i>Generación del dataset de ataques web</i>	25
4.3 <i>Captura de tráfico</i>	27
4.4 <i>Detección de tráfico de ataque</i>	28
5 Resultados	29
5.1 <i>URIs generadas</i>	29
5.2 <i>Separación de URIs de ataque según OWASP</i>	31
5.3 <i>Ataques detectados por cada IDS</i>	36

5.4	<i>Ataques detectados por el IDS completo</i>	43
5.5	<i>Anomalías encontradas en el estudio</i>	47
6	Conclusiones y Líneas de Avance	49
6.1	<i>Conclusiones</i>	49
6.2	<i>Limitaciones</i>	49
6.3	<i>Líneas de avance</i>	50
Anexo A:	Despliegue de Páginas Web	51
A.	<i>Servidor Web Apache</i>	52
B.	<i>Página Web Estática</i>	52
C.	<i>Página Web Dinámica</i>	52
Anexo B:	Instalación y Uso de Herramientas de Ataque	55
A.	<i>Herramientas OpenSource</i>	55
1.	Arachni	55
2.	Nikto	56
3.	Golismo	57
4.	Grabber	57
5.	Grendel-Scan	58
6.	Nmap	59
7.	Nuclei	60
8.	Openvas	60
9.	OWASP-ZAP	62
10.	Vega	64
11.	W3af	65
12.	Wapiti	66
13.	Havij	67
14.	Sqlmap	67
15.	Commix	68
16.	Metasploit	68
17.	Nessus	68
18.	Xsser	70
19.	Ironwasp	71
20.	Skipfish	72
21.	Watobo	72
22.	Wikto	73
23.	Wfuzz	73
B.	<i>Herramientas Comerciales</i>	74
1.	Burpsuite	74
2.	Nexploit	75
3.	Nexpose	76
4.	SmartScanner	77
5.	Webcruiser	78
6.	Wpscan	78
Anexo C:	Instalación y Uso de Herramientas de Captura de Tráfico	79
A.	<i>Wireshark</i>	79
B.	<i>Nfdump</i>	79
C.	<i>Ficheros .csv, .uri y .short</i>	79
Anexo D:	Instalación y Uso de Herramientas de Detección: <i>InspectorLog</i>	81
A.	<i>Snort</i>	81
B.	<i>ModSecurity</i>	81
C.	<i>Nemesida</i>	82
D.	<i>Datos IDS completo</i>	82
Referencias		83

Índice de Tablas

Tabla 1: Herramientas de ataque utilizadas	26
Tabla 2: Aportación al <i>dataset</i> de cada herramienta	30
Tabla 3: Aportación herramientas según clasificación OWASP [14] para la web estática	32
Tabla 4: Aportación herramientas según clasificación OWASP [14] para la web dinámica	32
Tabla 5: Nº URIs detectadas como ataque por Snort	37
Tabla 6: Nº URIs detectadas como ataque por ModSecurity	38
Tabla 7: Nº URIs detectadas como ataque por Nemesida	39
Tabla 8: Nº URIs detectadas como ataque por el IDS completo	44
Tabla 9: Anomalías en las URIs generadas	48
Tabla 11: Relación <i>pcap</i> generado/checks utilizados por arachni	55
Tabla 10: <i>Plugins</i> de <i>Nikto</i> utilizados en el estudio	56
Tabla 12: <i>Plugins</i> utilizados por <i>Grabber</i> .	57

Índice de Figuras

Figura 1: Módulos dinámicos Nemesida.	22
Figura 2: Diagrama de bloques (simplificado) para la herramienta InspectorLog.	24
Figura 3: Ejemplo de informe de detección de ataques generado por la herramienta <i>InspectorLog</i> .	24
Figura 4: Esquema de red propuesto	25
Figura 5: Separación de URIs según ataques y herramienta para la web estática	33
Figura 6: Separación de URIs según ataques y herramienta para la web dinámica	34
Figura 7: Separación de URIs según ataques y tipo de licencia para la web estática	35
Figura 8: Separación de URIs según ataques y tipo de licencia para la web dinámica	36
Figura 9: Detecciones de ataque según herramienta para la web estática	40
Figura 10: Detecciones de ataque según herramienta para la web dinámica	41
Figura 11: Detecciones de ataque según licencia de la herramienta para la web estática en cada IDS por separado	42
Figura 12: Detecciones de ataque según licencia de la herramienta para la web dinámica en cada IDS por separado	42
Figura 13: Detecciones del IDS completo para cada herramienta	45
Figura 14: Detecciones del IDS completo diferenciando tipo de licencia de las herramientas	46
Figura 15: Esquema de red propuesto	51
Figura 16: Configuración de sitio web <i>Wordpress</i>	53
Figura 17: Interfaz gráfica <i>Grendel-Scan</i>	58
Figura 18: Pestaña <i>Test Module Selection</i> de <i>Grendel-Scan</i> .	59
Figura 19: Interfaz gráfica <i>Openvas</i> .	60
Figura 20: Interfaz para añadir un nuevo <i>Target</i> de <i>Openvas</i> .	61
Figura 21: Interfaz de realización del escaneo para <i>Openvas</i> .	61
Figura 22: Interfaz gráfica <i>OWASP-ZAP</i>	62
Figura 23: Políticas de escaneo <i>OWASP-ZAP</i>	63
Figura 24: Desplegable <i>Threshold</i> de políticas <i>OWASP-ZAP</i>	63
Figura 25: Interfaz gráfica <i>Vega</i>	64
Figura 26: Módulos de <i>Vega</i>	65
Figura 27: Generación de ataques con <i>Havij</i> .	67
Figura 28: Captura de los comandos de utilización del <i>wizard</i> para <i>sqlmap</i> .	67
Figura 29: Captura de los comandos de utilización del <i>wizard</i> para <i>commix</i> .	68
Figura 30: Interfaz de configuración de <i>Nessus 1</i> .	68

Figura 31: Interfaz de configuración de <i>Nessus 2</i> .	69
Figura 32: Interfaz de configuración de <i>Nessus 3</i> .	69
Figura 33: Interfaz web de <i>Nessus Essentials</i> .	70
Figura 34: Opciones del <i>wizard</i> de <i>xsser 1</i> .	70
Figura 35: Opciones del <i>wizard</i> de <i>xsser 2</i> .	71
Figura 36: Escaneo mediante interfaz de <i>Ironwasp</i>	71
Figura 37: Selección de módulos <i>Ironwasp</i> .	72
Figura 38: Selección de plugins de <i>Watobo</i>	72
Figura 39: Pestaña <i>Wikto</i> de la herramienta <i>Wikto</i> .	73
Figura 40: Interfaz gráfica <i>Burpsuite</i> .	74
Figura 41: Pestaña <i>New Scan</i> de <i>Burpsuite</i> .	74
Figura 42: Interfaz gráfica <i>Nexploit</i> .	75
Figura 43: Pestaña <i>Scan Targets</i> de <i>Nexpose</i> .	75
Figura 44: Pestaña <i>Scan Tests</i> de <i>Nexpose</i> .	76
Figura 45: Interfaz gráfica <i>Nexpose</i> .	76
Figura 46: Pestaña <i>Assets</i> de <i>Create Site</i> en <i>Nexpose</i> .	77
Figura 47: Pestaña <i>Templates</i> de <i>Create Site</i> en <i>Nexpose</i> .	77
Figura 48: Interfaz gráfica de <i>SmartScanner</i> .	77
Figura 49: Opciones seleccionadas en el lanzamiento por <i>Webcruiser</i>	78

1 INTRODUCCIÓN

En los últimos años, los servicios web han experimentado un crecimiento exponencial debido tanto al desarrollo de servicios cada vez más complejos y avanzados como a la fácil accesibilidad de los mismos mediante el uso de navegadores. Consecuentemente, los servidores web se han convertido en objetivos cada vez más frecuentes de ciberataques, provocando en ocasiones grandes pérdidas de información y datos confidenciales.

Los atacantes comprometen los servidores mayoritariamente mediante la distribución de *malware* o realizando *phishing*, sirviendo estos mecanismos como puerta de acceso a la red de una empresa. El uso de conexiones HTTP cifradas ha provocado que los sistemas de detección de intrusiones basados en firmas (SIDS) [1] habitualmente desplegados se desplacen al propio servidor web, bien integrándose como parte del WAF (*Web Application Firewall*) o bien operando sobre los archivos de traza de los servidores.

En este trabajo se pretende verificar el grado de detección de ataques basados en web que ofrecen algunos de los SIDS o WAF más difundidos en la actualidad. En particular, consideraremos tres de ellos: *Snort* [2], *ModSecurity* [3] y *Nemesida* [4] (con reglas gratuitas). Para ello, previamente, será necesario establecer *datasets* de ataques contra los que validar su capacidad de detección que sean representativos y se encuentren actualizados.

En este sentido, existen pocos *datasets* públicos disponibles para hacer experimentación sobre SIDS en general (e.g., KDD'99 [5], CAIDA [6], UNSW-NB15 [7]) y aún menos que sean específicos para HTTP (e.g. CICIDS2017 [8]). Los ataques que contienen aparecen en número reducido, son antiguos y mayoritariamente de sistemas simulados [9]. Esta carencia de cercanía a la realidad y la falta de representatividad los hacen poco aplicables a entornos de producción [10] y, consecuentemente, los invalidan para desarrollar y/o evaluar sistemas de detección de ataques válidos en escenarios reales modernos [11], especialmente porque los servicios web y los ataques evolucionan rápidamente con el tiempo. Si bien es posible encontrar más de 80 bases de datos públicas [12] con peticiones reales a servidores, un análisis de las mismas nos muestra que, o bien no están marcadas las peticiones de ataque, o bien se limitan a actividad recopilada mediante *honeypots*, lo que no garantiza la representatividad ni la presencia de todos los tipos de ataques.

Para una mejor y más actualizada representación de los tipos de ataque a servidores web más comunes hoy en día, se ha generado un nuevo *dataset* con tráfico de ataques web (teniendo en cuenta sólo aquellos basados en el contenido de las URI) a partir de varias herramientas de ataque actualizadas seleccionadas al efecto. Se han considerado dos escenarios (una web estática y una web dinámica) y se ha seguido una metodología que permite su repetición y ampliación.

Además, se ha comprobado la eficiencia de los distintos SIDS utilizados en el estudio contra el *dataset* anteriormente mencionado, así como proporcionado una solución de seguridad que enlaza los tres SIDS estudiados con el objetivo de aumentar la seguridad del entorno web.

Estas contribuciones permitirán la selección de las herramientas SIDS más apropiadas y la ejecución de nuevas pruebas que utilicen el *dataset* proporcionado con distintos SIDS o con nuevas firmas. Por otra parte, el *dataset* podría también utilizarse para evaluar sistemas basados en detección de anomalías.

2 OBJETIVOS DEL TRABAJO

2.1 Objetivos principales

En este trabajo se pretende elaborar una lista actualizada de herramientas susceptibles de ser utilizadas para la realización de ataques web, con sus principales características, según las fuentes de información más utilizadas por los expertos en este ámbito.

También se intenta proporcionar un *dataset* completo y actualizado de ataques web generado por las herramientas previamente seleccionadas. Gracias a este *dataset*, será posible conocer la manera en la que dichas herramientas intentan encontrar y explotar las vulnerabilidades más comunes presentes en servidores web, además de proporcionar una visión comparativa del desempeño de cada una de estas herramientas en la tarea descrita.

Una parte importante para la consecución de los objetivos es también la correcta captura y síntesis del *dataset* generado mediante la manipulación de los paquetes capturados con las herramientas correspondientes.

Además, el estudio pretende realizar un análisis acerca de la capacidad de detección sobre el *dataset* anterior que ofrecen los principales sistemas SIDS de código abierto usados en servidores web, con el objetivo de comparar una estimación¹ de la eficiencia de estos, así como plantear una solución de detección que enlace los distintos IDS utilizados para una mayor cobertura de detección.

Con la consecución de estos objetivos principales, se habrá conseguido realizar un estudio experimental acerca de la generación, captura, detección y posterior análisis de los principales ataques a servidores web presentes en la actualidad.

2.2 Objetivos específicos

Tras haber definido la motivación principal del trabajo, se ha elaborado una lista de objetivos alternativos que también se pretenden en el desarrollo de este:

- Despliegue de un contexto web bajo el servidor web Apache.
- Correcta instalación y uso de las distintas herramientas de ataque utilizadas en el estudio.
- Correcta instalación y uso de las distintas herramientas de captura utilizadas en el estudio.
- Correcta instalación y uso de las distintas herramientas de detección utilizadas en el estudio.
- Presentación de datos procedentes de la captura y análisis del tráfico generado en tablas que permitan una mayor comprensión e interpretación de los resultados del trabajo.
- Obtención de una comparativa de manera gráfica acerca de las herramientas de ataque y detección utilizadas.

¹ No se están teniendo en cuenta los falsos positivos.

3 CONOCIMIENTOS PREVIOS

3.1 Ataques a servidores web: Clasificación de vulnerabilidades de OWASP

Open Web Application Security Project (OWASP) es una fundación sin ánimo de lucro que constantemente apoya proyectos de infraestructura y seguridad informática relacionada a las aplicaciones web. Esta labor es posible gracias a proyectos de software de código abierto proporcionados por su numerosa comunidad, que consta de decenas de miles de miembros. La Fundación OWASP es la fuente que utilizan las empresas para proteger sus recursos web [13].

El OWASP Top 10 es un documento de concienciación estándar para los desarrolladores. Representa un amplio consenso sobre los riesgos de seguridad más críticos para las aplicaciones web. Las empresas eligen este documento como punto de partida para garantizar que sus aplicaciones web minimicen estos riesgos. Usar OWASP Top 10 es quizás el primer paso más efectivo para cambiar la cultura de desarrollo de software dentro de una organización a una que produzca código más seguro [14].

Anteriormente la colección de datos que realizó OWASP se basaba en 30 CWEs (*Common Weakness Enumeration*) que ellos seleccionaron y un campo para incluir otros descubrimientos. Sin embargo, las empresas tan solo incluían datos de esos 30 CWEs. Por eso esta vez pidieron simplemente datos de las vulnerabilidades encontradas en sus aplicaciones, sin restringirlo a determinados CWEs. Preguntaron por el número de aplicaciones analizadas anualmente desde 2017, en las que se ha encontrado al menos un CWE. De esa forma pueden comprobar cuán prevalente es cada CWE dado un grupo de aplicaciones. No se tuvo en consideración la frecuencia de los CWEs a la hora de realizar el Top 10 [15].

En cuanto a las categorías, han intentado centrarse en analizar la raíz de los problemas de seguridad, en lugar de focalizarse en los síntomas (como podemos ver con el caso de *Cryptographic Failure*, que es la causa del síntoma *Sensitive Data Exposure*). En 2017 se basaron en el *ratio* de incidencia, mientras que este año querían centrarse en la explotabilidad y el impacto [15].

Así, las 10 vulnerabilidades más comunes en aplicaciones web señaladas por OWASP en su lista de 2021 fueron las siguientes, siguiendo el orden de la propia lista.

3.1.1 Broken Access Control

La política de control de acceso es aquella que controla que cada usuario actúe en el sistema según sus permisos. En esencia, *Broken Access Control* es simplemente un escenario en el que los atacantes pueden acceder, modificar, eliminar o realizar acciones fuera de los permisos previstos de una aplicación o sistema [16].

Muchas vulnerabilidades se pueden clasificar como una forma de *Broken Access Control*, por ejemplo [17]:

- Omisión de las comprobaciones de control de acceso modificando la URL, el estado interno de la aplicación o la página HTML, o simplemente usando una herramienta de ataque personalizada.
- Permitir que la clave principal se cambie al registro de otros usuarios, lo que permite ver o editar la cuenta de otra persona.
- Elevación de privilegio. Actuar como usuario sin haber iniciado sesión o actuar como administrador cuando se ha iniciado sesión como usuario.
- Manipulación de metadatos, como la reproducción o manipulación de un token de control de acceso *JSON Web Token (JWT)* o una cookie o un campo oculto manipulado para elevar los privilegios, o abusar de la invalidación de JWT
- Una configuración errónea de CORS (Intercambio de Recursos de Origen Cruzado) permite el acceso no autorizado a la API (Interfaz de Programación de Aplicaciones).
- Forzar la navegación a páginas autenticadas como usuario no autenticado o a páginas privilegiadas como usuario estándar. Acceso a la API sin controles de acceso para POST, PUT y DELETE.

3.1.2 Cryptographic Failures

En la lista de 2017, la vulnerabilidad denominada *Sensitive Data Exposure* comprendía esta categoría, que es más un síntoma general que una causa raíz, ahora la atención se centra en los fallos relacionados con la criptografía. En la lista de 2021 esta categoría ya ocupa el segundo lugar [18].

Los atacantes a menudo tienen como objetivo datos confidenciales, como contraseñas, números de tarjetas de crédito e información personal. La categoría *Cryptographic Failures* aborda todas aquellas vulnerabilidades que son potencialmente la causa de la exposición de estos datos [19]. Los principales CWE (*Common Weakness Enumeration*) que pueden provocar una vulnerabilidad de esta categoría son:

- CWE-259: Almacenamiento o tráfico de datos en texto claro.
- CWE-331: Protección de datos con un cifrado antiguo o débil.
- CWE-327: Filtrar o enmascarar de manera incorrecta los datos.

3.1.3 Injection

La inyección es el intento de un atacante de enviar datos a una aplicación de una manera que cambiará el significado de los comandos que se envían a un intérprete. Por ejemplo, el ejemplo más común es la *SQL Injection*, donde un atacante envía "101 O 1 = 1" en lugar de solo "101". Cuando se incluyen en una consulta SQL, estos datos cambian el significado para devolver todos los registros en lugar de solo uno. Con frecuencia, estos intérpretes se ejecutan con un nivel alto de permisos, por lo que un ataque exitoso puede resultar fácilmente en filtraciones de datos significativos o incluso en la pérdida de control de un navegador, una aplicación o un servidor [20].

Una aplicación es vulnerable este ataque cuando [21]:

- Los datos proporcionados por el usuario no son validados, filtrados ni sanitizados por la aplicación.
- Se invocan consultas dinámicas o no parametrizadas, sin codificar los parámetros de forma acorde al contexto.
- Se utilizan datos dañinos dentro de los parámetros de búsqueda en consultas *Object-Relational Mapping* (ORM), para extraer registros adicionales sensibles.
- Se utilizan datos dañinos directamente o se concatenan, de modo que el SQL o comando resultante contiene datos y estructuras con consultas dinámicas, comandos o procedimientos almacenados.

Algunas de las inyecciones más comunes son SQL, NoSQL, comandos de sistema operativo, *Object-Relational Mapping* (ORM), LDAP, expresiones de lenguaje u *Object Graph Navigation Library* (OGNL). El concepto es idéntico para todos los intérpretes. La revisión del código fuente es el mejor método para detectar si las aplicaciones son vulnerables a inyecciones. Las pruebas automatizadas en todos los parámetros, encabezados, URL, cookies, JSON, SOAP y XML son fuertemente recomendados. Las organizaciones pueden incluir herramientas de análisis estático (SAST), dinámico (DAST) o interactivo (IAST) en sus pipelines de CI/CD con el fin de identificar fallas recientemente introducidas, antes de ser desplegadas en producción. *Cross-site Scripting* (XSS) ahora forma parte de esta categoría [21].

3.1.4 Insecure Design

Una nueva categoría en la lista de 2021. se centra en los riesgos relacionados con el diseño y los fallos arquitectónicos, exhortando a un mayor uso de: modelado de amenazas, patrones de diseño seguros y arquitecturas de referencia. Las CWE notables incluidas son *CWE-209: Generation of Error Message Containing Sensitive Information*, *CWE-256: Unprotected Storage of Credentials*, *CWE-501: Trust Boundary Violation*, and *CWE-522: Insufficiently Protected Credentials* [22].

Insecure Design es una categoría amplia que representa diferentes debilidades, expresadas como "diseño de control faltante o ineficaz". El diseño inseguro no es la fuente de las otras 10 categorías. Existe una diferencia entre un diseño y una implementación inseguros. Distinguimos entre fallos de diseño y defectos de implementación por un motivo, difieren en la causa raíz y mitigaciones. Incluso un diseño seguro puede tener

defectos de implementación que conduzcan a vulnerabilidades que pueden explotarse. Un diseño inseguro no se puede arreglar con una implementación perfecta, ya que, por definición, los controles de seguridad necesarios nunca se crearon para defenderse de ataques específicos. Uno de los factores que contribuyen al diseño inseguro es la falta de perfiles de riesgo empresarial inherentes al software o sistema que se está desarrollando y, por lo tanto, la falta de determinación del nivel de diseño de seguridad que se requiere [22].

3.1.5 Security Misconfiguration

La aplicación puede ser vulnerable si:

- Le falta el *hardening* de seguridad adecuado en cualquier parte del *stack* tecnológico o permisos configurados incorrectamente en los servicios en la nube.
- Tiene funciones innecesarias habilitadas o instaladas (puertos, servicios, páginas, cuentas o privilegios innecesarios, por ejemplo).
- Las cuentas predeterminadas y sus contraseñas aún están habilitadas y sin cambios.
- El manejo de errores revela a los usuarios rastros de pila u otros mensajes de error demasiado informativos.
- Para sistemas actualizados, las últimas funciones de seguridad están deshabilitadas o no configuradas de forma segura.
- Las configuraciones de seguridad en los servidores de aplicaciones, *frameworks* de aplicaciones (Struts, Spring o ASP.NET por ejemplo), bibliotecas, bases de datos, etc., no poseen configurados valores seguros.
- El servidor no envía encabezados o directivas de seguridad, o no poseen configurados valores seguros.
- El software está desactualizado o es vulnerable

Sin un proceso de configuración de seguridad de aplicaciones coordinado y repetible, los sistemas corren un mayor riesgo. Las CWE notables incluidas son *CWE-16 Configuration* and *CWE-611 Improper Restriction of XML External Entity Reference*[23].

3.1.6 Vulnerable and Outdated Components

La aplicación en cuestión probablemente sea vulnerable según esta categoría si cumple alguna de estas debilidades [24]:

- No conoce las versiones de todos los componentes que utiliza (tanto en el cliente como en el servidor). Esto incluye los componentes que usa directamente, así como las dependencias anidadas.
- El software es vulnerable, carece de soporte o no está actualizado. Esto incluye el sistema operativo, el servidor web/de aplicaciones, el sistema de administración de bases de datos (DBMS), las aplicaciones, las API y todos los componentes, los entornos de ejecución y las bibliotecas.
- No analiza en búsqueda de vulnerabilidades de forma regular y no se suscribe a los boletines de seguridad relacionados con los componentes que utiliza.
- No repara o actualiza la plataforma subyacente, *frameworks* y dependencias de manera oportuna y basada en el riesgo. Esto suele ocurrir en entornos en los que la aplicación de parches de seguridad es una tarea mensual o trimestral bajo control de cambios, lo que deja a las organizaciones abiertas a días o meses de exposición innecesaria a vulnerabilidades con soluciones disponibles.
- Los desarrolladores de software no testean la compatibilidad de las bibliotecas actualizadas, actualizadas o parcheadas.
- No asegura las configuraciones de los componentes

Los CWE más comunes en esta categoría son *CWE-1104: Use of Unmaintained Third-Party Components* and the two *CWEs* from *Top 10 2013* and *2017*.

3.1.7 Identification and Authentication Failures

Previamente denominada como *Broken Authentication*, descendió desde la segunda posición, y ahora incluye CWEs que están más relacionados con fallos de identificación. Las CWE notables incluidas son *CWE-297: Improper Validation of Certificate with Host Mismatch*, *CWE-287: Improper Authentication*, and *CWE-384: Session Fixation* [25].

La confirmación de la identidad, la autenticación y la gestión de sesiones del usuario son fundamentales para protegerse contra ataques relacionados con la autenticación. Puede haber debilidades de autenticación si la aplicación [25]:

- Permite ataques automatizados como la reutilización de credenciales conocidas, donde el atacante posee una lista de pares de usuario y contraseña válidos.
- Permite ataques de fuerza bruta u otros ataques automatizados.
- Permite contraseñas por defecto, débiles o bien conocidas, como "Password1" o "admin/admin".
- Posee procesos débiles o no efectivos para las funcionalidades de olvido de contraseña o recuperación de credenciales, como "respuestas basadas en el conocimiento", las cuales no se pueden implementar de forma segura.
- Almacena las contraseñas en texto claro, cifradas o utilizando funciones de *hash* débiles
- No posee una autenticación multi-factor o la implementada es ineficaz.
- Expone el identificador de sesión en la URL.
- Reutiliza el identificador de sesión después de iniciar sesión.
- No invalida correctamente los identificadores de sesión. Las sesiones de usuario o los tokens de autenticación (principalmente *tokens* de inicio de sesión único (SSO)) no son correctamente invalidados durante el cierre de sesión o luego de un período de inactividad.

3.1.8 Software and Data Integrity Failures

Una nueva categoría en la versión 2021 que se centra en hacer suposiciones relacionadas con las actualizaciones de software, los datos críticos y los *pipelines* de CI/CD sin verificación de integridad. Corresponde a uno de los mayores impactos según los sistemas de ponderación de vulnerabilidades (CVE/CVSS, siglas en inglés para Common Vulnerability and Exposures/Common Vulnerability Scoring System). Entre estos, se destacan las siguientes CWEs: *CWE-829: Inclusion of Functionality from Untrusted Control Sphere*, *CWE-494: Download of Code Without Integrity Check*, and *CWE-502: Deserialization of Untrusted Data* [26].

Los fallos de integridad del software y de los datos están relacionados con código e infraestructura no protegidos contra alteraciones (integridad). Ejemplos de esto son cuando una aplicación depende de plugins, bibliotecas o módulos de fuentes, repositorios o redes de entrega de contenidos (CDN) no confiables. Un pipeline CI/CD inseguro puede conducir a accesos no autorizados, la inclusión de código malicioso o el compromiso del sistema en general. Además, es común en la actualidad que las aplicaciones implementen funcionalidades de actualización, a través de las cuales se descargan nuevas versiones de esta sin las debidas verificaciones integridad que fueron realizadas previamente al instalar la aplicación. Los atacantes potencialmente pueden cargar sus propias actualizaciones para que sean distribuidas y ejecutadas en todas las instalaciones. Otro ejemplo es cuando objetos o datos son codificados o serializados en estructuras que un atacante puede ver y modificar, produciéndose una deserialización insegura [26].

3.1.9 Security Logging and Monitoring Failures

No existe una vulnerabilidad directa que pueda surgir debido a estos problemas, pero en general, el registro y el monitoreo son bastante críticos y su ausencia o fallos pueden afectar directamente la visibilidad, las alertas de incidentes y el análisis forense. Por lo tanto, es muy importante tener un sistema de registro y monitoreo funcional para recopilar registros y también dar alertas si ocurre algún mal funcionamiento o error, de lo contrario, estos pueden pasar desapercibidos durante mucho tiempo y causar mucho más daño [27].

La intención es apoyar la detección, escalamiento y respuesta ante brechas activas. Sin registros y monitoreo, las brechas no pueden ser detectadas. Registros, detecciones, monitoreo y respuesta activas insuficientes pueden ocurrir en cualquier momento [28]:

- Eventos auditables, tales como los inicios de sesión, fallas en el inicio de sesión y transacciones de alto valor no son registradas.
- Advertencias y errores generan registros poco claros, inadecuados y en algunos casos ni se generan.
- Registros en aplicaciones y API no son monitoreados para detectar actividades sospechosas.
- Los registros son únicamente almacenados en forma local.
- Los umbrales de alerta y procesos de escalamiento no están correctamente implementados o no son efectivos.
- Las pruebas de penetración y los escaneos utilizando herramientas de pruebas dinámicas de seguridad en aplicaciones (como ser OWASP ZAP) no generan alertas.
- Las aplicaciones no logran detectar, escalar, o alertar sobre ataques activos en tiempo real ni cercanos al tiempo real.

3.1.10 Server-Side Request Forgery (SSRF)

Los fallos de SSRF ocurren cuando una aplicación web está obteniendo un recurso remoto sin validar la URL proporcionada por el usuario. Permite que un atacante coaccione a la aplicación para que envíe una solicitud falsificada a un destino inesperado, incluso cuando está protegido por un firewall, VPN u otro tipo de lista de control de acceso a la red (ACL) [29].

Dado que las aplicaciones web modernas brindan a los usuarios finales funciones convenientes, la búsqueda de una URL se convierte en un escenario común. Como resultado, la incidencia de SSRF está aumentando. Además, la gravedad de SSRF es cada vez mayor debido a los servicios en la nube y la complejidad de las arquitecturas [29].

3.2 Herramientas de análisis de vulnerabilidades web

Se entiende por herramientas de análisis de vulnerabilidades aquel software comercial, de código abierto, integrado o disponible públicamente que podría ser utilizado por un defensor, un evaluador de penetración, un miembro del equipo *red team* o un atacante. Esta categoría incluye tanto el software que generalmente no se encuentra en un sistema empresarial como el software generalmente disponible como parte de un sistema operativo que ya está presente en un entorno [30].

3.3 Herramientas de captura de tráfico

3.3.1 Wireshark

Wireshark es el analizador de protocolos de red más importante y ampliamente utilizado del mundo. Permite ver lo que sucede en la red a un nivel microscópico y es el estándar en muchas empresas comerciales, agencias gubernamentales e instituciones educativas. El desarrollo de *Wireshark* prospera gracias a las contribuciones voluntarias de expertos en redes de todo el mundo y es la continuación de un proyecto iniciado por Gerald Combs en 1998 [31].

En esencia, esta herramienta fue desarrollada para rastrear paquetes de datos que viajan a través de diferentes redes. Por tanto, *Wireshark* nos permite inspeccionar a profundidad el tráfico que circula por una red y así tomar decisiones adecuadas [32].

Wireshark tiene un rico conjunto de características que incluye lo siguiente [31]:

- Inspección profunda de cientos de protocolos, y se agregan más todo el tiempo

- Captura en vivo y análisis fuera de línea
- Navegador de paquetes estándar de tres paneles
- Multiplataforma: se ejecuta en *Windows, Linux, macOS, Solaris, FreeBSD, NetBSD* y muchos otros
- Los datos de red capturados se pueden explorar a través de una GUI o a través de la utilidad *TShark* en modo TTY
- Los filtros de visualización más potentes de la industria
- Análisis completo de VoIP
- Lee/escribe muchos formatos de archivo de captura diferentes
- Los archivos de captura comprimidos con *gzip* se pueden descomprimir sobre la marcha
- Los datos en vivo se pueden leer desde Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI y otros (dependiendo de su plataforma)
- Soporte de descifrado para muchos protocolos, incluidos IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP y WPA/WPA2
- Se pueden aplicar reglas de colores a la lista de paquetes para un análisis rápido e intuitivo
- La salida se puede exportar a XML, PostScript, CSV o texto sin formato

3.3.2 nfdump

nfdump es un conjunto de herramientas para recopilar y procesar datos de *netflow* y *sflow*, enviados desde dispositivos compatibles con *netflow/sflow*. El conjunto de herramientas admite *netflow* v1, v5/v7, v9, *IPFIX* y *SFLOW*. *nfdump* admite tanto IPv4 como IPv6 [33].

Posee las siguientes herramientas [34]:

- *nfcapd*: Es un colector de captura netflow. Lee los datos de netflow de la red y almacena los datos en archivos. Gira automáticamente los archivos cada n minutos. (generalmente cada 5 minutos) *nfcapd* lee los flujos de netflow v5, v7 y v9 de forma transparente. Necesita un proceso *nfcapd* para cada flujo de netflow.
- *nfdump*: Lee los datos de flujo de red de los archivos almacenados por *nfcapd*. Su sintaxis es similar a *tcpdump*. Si te gusta *tcpdump* te gustará *nfdump*. Muestra datos de flujo de red y puede crear muchas estadísticas N principales de flujos de direcciones IP, puertos, etc. ordenados en el orden que desee.
- *nfprofile*: Lee los datos de flujo de red de los archivos almacenados por *nfcapd*. Filtra los datos de *netflow* de acuerdo con los conjuntos de filtros especificados (perfiles) y almacena los datos filtrados en archivos para su uso posterior.
- *nfreply* - Lee los datos de flujo de red de los archivos almacenados por *nfcapd* y los envía a través de la red a otro *host*.
- *nfclean.pl*: Script de muestra para limpiar datos antiguos. Puede ejecutar este script cada hora más o menos.
- *ft2nfdump*: Lee datos de herramientas de flujo de archivos o de stdin en una cadena de comandos de herramientas de flujo y convierte los datos en formato *nfdump* para que *nfdump* los procese.
- *nfanon*: Las direcciones IP en los registros de flujo se anonimizan mediante el método *CryptoPAN*.
- *nfexpire*: Gestiona la caducidad de los datos. Establece límites apropiados. Utilizado por *NfSen*.
- *nfcapd*: Convierte un archivo *pcap* en tráfico *netflow*. Escucha en una interfaz de red o lee el tráfico de *pcap* recopilado previamente y almacena registros de flujo en archivos compatibles con *nfcapd*. Es el compañero de *nfcapd* para convertir el tráfico directamente en registros *nfdump*.
- *sfcapd*: Recopila datos de *sflow* y los almacena en archivos compatibles con *nfcapd*.

- *nfpfile*: Es un generador de perfiles de flujo de red. Requerido por *NfSen*. Lee los datos de flujo de red de los archivos almacenados por *nfcapd*. Filtra los datos de *netflow* de acuerdo con los conjuntos de filtros especificados (perfiles) y almacena los datos filtrados en archivos para su uso posterior.
- *nftack*: Decodificador de seguimiento de puertos para el complemento PortTracker de *NfSen*.
- *nfreader*: Es un marco para leer archivos *nfdump* para cualquier otro propósito. Se puede agregar código C propio a los flujos de proceso. *nfreader* no está instalado
- *parse_csv.pl*: Lee la salida csv de *nfdump* e imprime los flujos en la salida estándar. Este programa pretende ser un marco para los flujos de procesamiento posterior para cualquier otro propósito.

3.4 Firewall de Aplicaciones Web

Un *firewall* es un dispositivo de seguridad de la red que monitoriza el tráfico entrante y saliente y decide si debe permitir o bloquear un tráfico específico en función de un conjunto de restricciones de seguridad ya definidas [35].

Un *firewall* de aplicaciones web (WAF) ayuda a proteger las aplicaciones web al filtrar y monitorear el tráfico HTTP entre una aplicación web e Internet. Normalmente, protege las aplicaciones web de ataques tales como falsificaciones entre sitios, *Cross-Site Scripting* (XSS), inclusiones de archivos e inyecciones de código SQL, entre otros. El WAF es una defensa del protocolo de capa 7 (en el modelo OSI) y no está diseñado para defender de todos los tipos de ataques. Este método de mitigación de ataques suele formar parte de un paquete de herramientas que, en conjunto, crean una defensa integral contra una amplia gama de vectores de ataque [36].

3.4.1 ModSecurity

ModSecurity fue un módulo del servidor web Apache que funcionaba mediante la comparación del tráfico del servidor web contra un sistema de reglas. Dependiendo de su configuración, procedía al bloqueo o transformación del tráfico con contenido malicioso, generando un log de actividad (block mode), o únicamente detectando y generando un log del tráfico malicioso (detection mode). Su sistema de reglas se ha convertido en un estándar usado en diversas aplicaciones WAF comerciales.

Actualmente, el proyecto se ha ampliado a su versión 3, dejando a un lado su dependencia del servidor web Apache, haciéndolo más independiente de la plataforma.

Libmodsecurity es una librería del proyecto *ModSecurity v3*. El código base de la librería sirve como una interfaz para *ModSecurity Connectors* que recibe tráfico web y aplica el procesamiento tradicional de *ModSecurity*. En general, brinda la capacidad de cargar/interpretar reglas escritas en el formato *ModSecurity SecRules* y aplicarlas al contenido HTTP proporcionado por su aplicación a través de conectores [3].

Una de las alternativas para establecer un esquema de seguridad más completo del que ofrece por defecto este módulo es la implementación de del *OWASP ModSecurity Core Rule Set* (CRS), el cual provee una serie de reglas de detección de ataques para la protección de cualquier aplicativo ubicado en un servidor de *Hosting*.

El *OWASP ModSecurity CRS* brinda protección contra los tipos de ataques más comunes detectados contra sitios en Internet (OWASP Top 10). Estas reglas están organizadas según el “nivel de protección” que el administrador desee, a estos niveles se les llama nivel de paranoia (*Paranoia Level, PL*).

Con cada aumento del nivel de paranoia, el CRS habilita reglas adicionales brindándole un mayor nivel de seguridad. Sin embargo, niveles más altos de paranoia también aumentan la posibilidad de bloquear parte del tráfico legítimo debido a falsas alarmas (también denominadas falsos positivos o FP). Si usa niveles de paranoia altos, es probable que deba agregar alguna regla de exclusión para ciertas solicitudes y aplicaciones que reciben entradas complejas.

El nivel de paranoia 1 es el predeterminado. En este nivel, la mayoría de las reglas básicas están habilitados. PL1 se recomienda para principiantes, para instalaciones cubriendo muchos sitios y aplicaciones diferentes, y para configuraciones con requisitos de seguridad estándar.

El nivel 2 de paranoia incluye muchas reglas adicionales, por ejemplo, habilitar muchas protecciones de

y detectar otras anomalías (por ejemplo, fuerza bruta, inundación, DDoS L7) [4].

Nemesida WAF API está diseñada para recibir información sobre ataques y vulnerabilidades detectadas, así como transmitir información sobre solicitudes bloqueadas y los resultados de los módulos *Nemesida AI* y *Nemesida WAF Scanner* en PostgreSQL DBMS [4].

Nemesida WAF Cabinet está diseñado para visualizar y analizar los eventos de los componentes del SGBD PostgreSQL, así como sistematizar información sobre anomalías y vulnerabilidades identificadas [4].

El módulo *Nemesida WAF Signtest* modifica el uso de los modelos construidos y aplicados por el módulo *Nemesida AI* [4].

Vulnerability Scanner Nemesida WAF Scanner está diseñado para identificar vulnerabilidades en una aplicación web protegida [4].

3.5 IDS: Sistema de detección de intrusiones

Es una aplicación usada para detectar accesos no autorizados a un ordenador o a una red. Ante cualquier actividad sospechosa, emiten una alerta a los administradores del sistema quienes han de tomar las medidas oportunas. Estos accesos pueden ser ataques esporádicos realizados por usuarios malintencionados o repetidos cada cierto tiempo, lanzados con herramientas automáticas. Estos sistemas sólo detectan los accesos sospechosos emitiendo alertas anticipatorias de posibles intrusiones, pero no tratan de mitigar la intrusión. Su actuación es reactiva [37].

Existen varios tipos de sistemas de detección de intrusos. Cada tipo de IDS utiliza diferentes métodos para detectar actividades sospechosas. Los siguientes son los tipos de sistemas de detección de intrusos que debe conocer [38]:

- Sistema de detección de intrusiones en la red (NIDS) está destinado principalmente a detectar intrusiones en toda una red. El NIDS monitoreará todo el tráfico entrante y saliente en todos los dispositivos a través de una red y detectará anomalías.
- Host Intrusion Detection System (HIDS) generalmente detecta intrusiones a través de un punto final en particular. Por ejemplo, un HIDS se ejecutará en todas las computadoras y dispositivos de su red. La ventaja significativa de un sistema de detección de intrusiones de host sobre un sistema de detección de red es que un HIDS puede detectar paquetes de red anormales dentro de una organización, que un NIDS podría no detectar en ocasiones. Además, un HIDS es capaz de identificar el tráfico malicioso que se origina en el host. Por ejemplo, notará cuando un host ha sido infectado con malware e intenta difundirlo a través de la red.
- Sistema de detección de intrusiones basado en firmas (SIDS) vigila todo el tráfico de una red y compara el tráfico con las bases de datos de firmas de ataques u otros riesgos de ciberseguridad conocidos.
- Un Sistema de detección de intrusiones basado en anomalías (SIDA) está diseñado para identificar ataques de ciberseguridad desconocidos, como nuevos ataques de malware. Compara los ataques con una línea de base establecida. Utiliza técnicas de aprendizaje automático para desarrollar una línea de base de actividad confiable denominada modelo de confianza. Compara los nuevos comportamientos con los modelos de confianza verificados. El principal inconveniente del SIDA es que a veces puede generar una falsa alarma cuando se identifica como una actividad anómala el tráfico de red legítimo, desconocido anteriormente.

3.5.1 Snort

Todo un veterano cuando se trata de análisis de paquetes. La primera versión vio la luz allá por 1998. Cabe mencionar que en aquel momento no se contempló como IDS puro, pero fue evolucionando hasta ese punto poco a poco. Desde entonces se ha ido convirtiendo en un estándar para sistemas de detección de intrusiones, eventualmente IPS (*Intrusion Prevention System*) gracias al trabajo de la comunidad. Sistemas como el de *AlienVault* lo integran para el análisis de riesgos [39].

Snort usa una serie de reglas que ayudan a definir la actividad de red maliciosa y usa esas reglas para encontrar paquetes que coincidan con ellos y generar alertas para los usuarios. También se puede implementar en línea.

Tiene tres usos principales: rastreador de paquetes como tcpdump, registrador de paquetes, que es útil para la depuración del tráfico de red, o puede usarse como un sistema completo de prevención de intrusiones en la red [2].

Entre sus ventajas, podemos destacar varios aspectos. Uno de los más evidentes es su longevidad y la buena salud del proyecto, algo que nos permite implementarlo sin preocuparnos por el futuro. También destaca el gran apoyo de la comunidad *Snort* y por tanto el hecho de ser una herramienta muy probada [39].

Como desventajas solo podemos decir que *Snort* no cuenta por defecto con un GUI o interfaz gráfica de usuario por defecto, por tanto, no es tan fácil de administrar como otras. Sin embargo, existen herramientas opensource para compensarlo, como *Snorby* o *Squid*, así que no es un problema realmente [39].

3.6 InspectorLog

Se puede describir como un SIDS basado en registros. En la Figura 2 se muestra un diagrama de bloques de su funcionamiento. *InspectorLog* acepta firmas y reglas con distintos formatos de los principales SIDS y detecta coincidencias entre cadenas y expresiones regulares en los campos de reglas y los URI analizados. *InspectorLog* solo considera las reglas relacionadas con URI/ataques HTTP (es decir, reglas que incluyen puertos http y elementos relacionados con campos como *http_uri*, *uricontent*, *http_raw_uri* o *content*) [40].

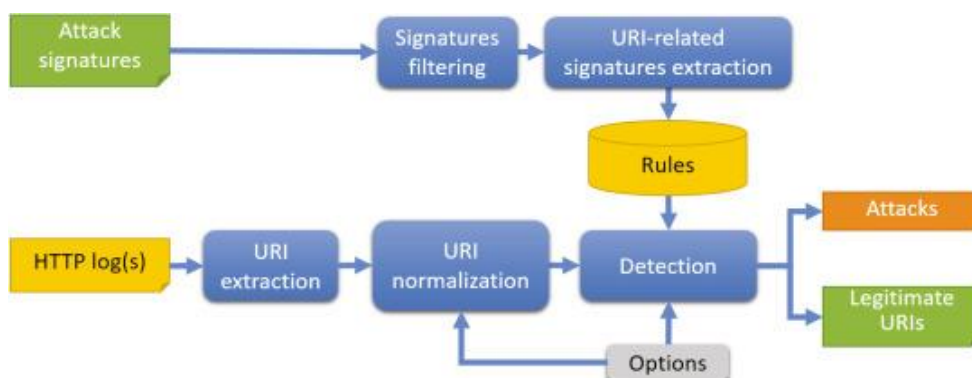


Figura 2: Diagrama de bloques (simplificado) para la herramienta *InspectorLog*.

InspectorLog genera dos listas: una con aquellos registros que activan reglas relacionadas con HTTP (ataques), y otra con aquellos registros que no activan ninguna regla (limpia). En la Figura 5 se muestra un ejemplo de salida de *InspectorLog*. Se puede observar que los campos de regla *msg*, *reference*, *classtype* y *sid* se muestran para cada detección [40].

```

# inspectorlog v3.1
# ----- Initializing Rules -----
# Rules directory : "/media/rules"
# ----- Statistics -----
# Read [7146] rules, [7141] http-related, [0] with errors
# ----- Analysis results -----
# Alerts and generated signatures: /bin/inspectorlog -l access_log-20170105-raw.uri -r /media/rules -m -e -t uri
Packet [32149] Uri [/xmlrpc.php?rsd] Mattacks [1] Signatures [SERVER-WEBAPP PHP xmlrpc.php post attempt - sid: 3827]
Packet [55705] Uri [/inca?app=ckeditor%7Csendto%40ckeditor_incaSendTo%7CCKEDITOR=edit-body-und-0-value&CKEDITORFunc3us=1&langCode=es]
Mattacks [1] Signatures [SERVER-APACHE Apache mod_proxy reverse proxy information disclosure attempt - sid: 20528]
Packet [112197] Uri [/educacion/noticias/bibeducacion%40us.es] Mattacks [1] Signatures
[SERVER-APACHE Apache mod_proxy reverse proxy information disclosure attempt - sid: 20528]
  
```

Figura 3: Ejemplo de informe de detección de ataques generado por la herramienta *InspectorLog*.

4 DESARROLLO DEL ESTUDIO

En este capítulo vamos a detallar la metodología utilizada para la generación, captura y detección del *dataset* de ataque necesario para realizar el estudio.

4.1 Preparación del entorno

Para llevar a cabo las pruebas pertinentes, ha sido necesario replicar un escenario como el de la siguiente figura:

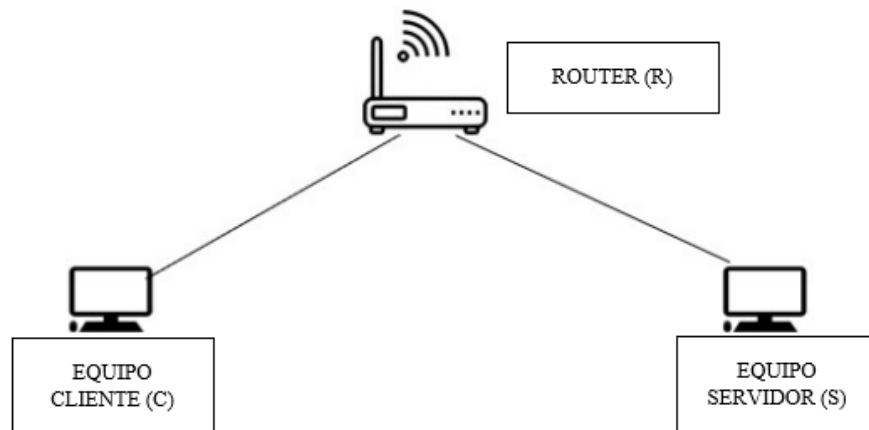


Figura 4: Esquema de red propuesto

En el equipo servidor (S) han sido desplegados dos escenarios web: una aplicación estática (Apache con un recurso html) y una aplicación dinámica (gestor de contenidos *Wordpress* con la instalación por defecto), así como instaladas la herramienta de captura de tráfico *Wireshark* [41] y el colector de flujo *nfdump* [42]. Como equipo servidor actuará una máquina virtual *Centos 7* [43] en *VMWare Workstation Player* [44] proporcionada por el Departamento de Telemática de la Universidad de Sevilla [45]. El proceso seguido para el despliegue de ambos contextos web puede consultarse en el Anexo A.

En el equipo cliente (C) se han instalado todas las herramientas de ataque web utilizadas en el estudio. Este equipo será también una máquina virtual en *VMWare Workstation Player* [44], pero en esta ocasión el sistema operativo será *Centos7* [43], *Ubuntu* [46], *Kali* [47] o *Parrot OS (Debian)* [48], según corresponda.

4.2 Generación del *dataset* de ataques web

Para que el *dataset* de ataques sea representativo, de los diversos tipos de ataques que pueden llevarse a cabo se han buscado las herramientas de ataques web disponibles en la actualidad. Para ello se han empleado referencias encontradas en tres fuentes de información básicas:

- OWASP: *Dynamic Application Security Testing* (DAST) [49].
- Mitre: técnicas *Exploit Public-Facing Application* [50] y *Software* [51].
- Listado de Software del proyecto *Nmap* [52], que cuenta con una de las listas más actualizadas sobre herramientas de ciberseguridad. Se ha buscado en las categorías: *Web Vulnerability scanners* y *Vulnerability Exploitation tools*.

Las distintas herramientas encontradas en las fuentes anteriores han sido revisadas de forma individual, encontrándose 29 de tipo opensource, de las cuales se han descartado 4 por obsoletas (*MBSA*, *Secunia PSI*, *ratproxy* y *firebug*) y otras 2 por no ser interesantes de cara al estudio, que son *Deepfence Threatmapper* (no incorporada al no realizar ataques a nivel HTTP) y *DirBuster* (tan solo realiza ataques de fuerza bruta). De tipo

comercial se han encontrado un total de 27 herramientas, para las que se ha solicitado al desarrollador una licencia de prueba gratuita que ha sido concedida en 6 casos, con ciertas limitaciones.

Como resultado, han sido probadas e instaladas en el sistema operativo correspondiente las herramientas que se detallan en la Tabla 1.

Herramienta	Opensource/ Comercial	Limitaciones	SO	Tipo de Herramienta	Funcionalidad Herramienta
Nikto	O		L*	G	E
Arachni	O		W, M, L*	E	E
Golismo	O		W, M, L*	E	S
Grabber	O		W, M, L*	E	S
Grendel-Scan	O		W, M, L*	E	S
Nmap	O		W, M, L*	G	S
Nuclei	O		W, M, L*	E	E
Openvas	O		L*	G	S
OWASP-ZAP	O		W, M, L*	E	E
Vega	O		W, M, L*	E	E
W3af	O		L*, M	E	E
Wapiti	O		W, M, L*	E	E
Burpsuite	C	30 días	W*, M, L	E	S
Nexploit	C	14 días, 5 horas de scan	SaaS*	E	S
Nexpose	C	30 días	W*, L	E	S
SmartScanner	C	79 tests	W*	E	S
Webcruiser	C	Solo IP privadas	W*	E	S
Wpscan	C	25 API requests per day*	L*, M	E	S
Havij	O		W*	E	E
Sqlmap	O		W, M, L*	E	E
Commix	O		W, M, L*	E	E
Metasploit	O		W, M, L*	G	E
Nessus	O		W*	G	E
Xsser	O		W, M, L*	E	E
Ironwasp	O		W*, M, L	E	E
Skipfish	O		W, M, L*	E	S
Watobo	O		W, M, L*	E	S
Wikto	O		W*	E	S
Wfuzz	O		W, L*, M	E	E

Tabla 1: Herramientas de ataque utilizadas

La tabla anterior indica la siguiente información en sus distintas columnas:

- Herramienta: Nombre de la herramienta en cuestión.
- Opensource/Comercial: Indica si la herramienta es de código abierto (O), o comercial (C).
- Limitaciones: En el caso de las herramientas comerciales, se detallan las limitaciones presentes en las versiones usadas.
- Sistema Operativo (SO): Especifica los sistemas operativos para los que está disponible la herramienta, siendo estos Windows (W), Linux (L), Macintosh (M) o Software as a service (SaaS), teniendo un asterisco (*) en el sistema operativo utilizado en el estudio.

- Tipo de herramienta: Describe si la herramienta está específicamente diseñada para vulnerabilidades web (E), o para vulnerabilidades en general, con plugins para web (G).
- Funcionalidad de herramienta: Detalla el tráfico de ataque que genera la herramienta, siendo este solo de "sondeo" (S) si no intenta explotar, atacar o usar la vulnerabilidad, sólo detectar posibles ficheros y debilidades; o de explotación (E) si por el contrario posee las capacidades mencionadas anteriormente.

Con las herramientas mencionadas, se ha procedido a generar el suficiente tráfico hacia el equipo servidor para su posterior captura y análisis. Dicho tráfico ha sido producido a través de realizar un análisis completo de vulnerabilidades web sobre la página correspondiente del equipo servidor. Adicionalmente, en aquellas herramientas opensource que lo permiten, se ha llevado a cabo una separación del tipo de tráfico generado según se haya utilizado un script, módulo o plugin concreto en el momento del lanzamiento, con el objetivo de poder separar posteriormente ese tráfico según el tipo de ataque al que pertenezca. Las herramientas opensource que permiten realizar esta separación son: *Nikto*, *Wapiti*, *Grendel-Scan*, *Nuclei*, *Nmap*, *Grabber*, *Ironwasp*, *Vega*, *W3af*, *Wfuzz*, *OWASP-ZAP* y *Arachni*. En el caso de las herramientas comerciales, *Burpsuite*, *Nexpose* y *Nexploit* sí presentan una separación entre módulos según las vulnerabilidades a encontrar en el escaneo, no obstante, esta separación es absolutamente extensa ya que estas herramientas están orientadas al entorno comercial para realizar escaneos de vulnerabilidad completos a entornos web, por tanto se ha optado por no realizar la separación en dichas herramientas y comprobar su funcionalidad de manera más ajustada a la utilidad real de estas.

Los debidos pasos de instalación y uso de las herramientas para reproducir exactamente los lanzamientos llevados a cabo en el estudio están detallados en el Anexo B.

4.3 Captura de tráfico

La captura del tráfico procedente del equipo cliente se ha realizado en el equipo servidor, mediante las herramientas *Wireshark* y *nfdump*.

El tráfico generado por cada lanzamiento de la herramienta ha sido capturado por *Wireshark* y almacenado en un fichero de extensión *.pcap* y de nombre la herramienta en cuestión o el plugin, módulo o script utilizado, según corresponda; a partir del archivo *pcap* se han generado cuatro archivos más de especial utilidad, cuyo nombre corresponde exactamente al nombre del archivo *pcap* a partir del cual ha sido generado, y cuya extensión depende del contenido. Por tanto, para cada captura tendremos los siguientes ficheros:

- Archivo con extensión *.pcap*: Captura *pcap* del lanzamiento correspondiente.
- Archivo con extensión *.ipfix*: Archivo que contiene la captura del tráfico IPFIX producido a partir de la utilidad *nfdump* de la herramienta *nfdump*.
- Archivo con extensión *.csv*: Archivo legible de la captura *pcap* organizada por columnas según las distintas cabeceras HTTP de cada petición mediante la herramienta *tshark*.
- Archivo con extensión *.uri*: Archivo legible con el contenido de la cabecera *http.request.uri* de cada petición de la captura completa.
- Archivo con extensión *.short*: Archivo donde se han eliminado las repeticiones y ordenado las distintas URIs en orden alfabético.

Estos ficheros conforman el cuerpo del *dataset* generado, que ha sido almacenado en https://github.com/fbuenoc97/TFG/tree/main/capturas-TFG/tools_clasification para su consulta, donde se encuentran los ficheros clasificados según la web de destino (estática o dinámica), la licencia de la herramienta (opensource o comercial) y el nombre de la herramienta que los ha generado.

Adicionalmente, para las herramientas en las que se ha realizado una separación del tráfico generado según el script, módulo o plugin utilizado, se han efectuado también unas capturas alternativas que relacionan directamente el nombre del archivo con el tipo de vulnerabilidad al que pertenece según la clasificación de OWASP [14], manteniendo las extensiones anteriormente expuestas. Dichas capturas son el resultado de relacionar los plugins, módulos o scripts usados por las herramientas con los tipos de vulnerabilidad al que pertenecen, cada relación puede consultarse a través de la *Tabla 2: Attack Capabilities* del fichero https://github.com/fbuenoc97/TFG/blob/main/Tablas_Herramientas_Ataque_Web_vFINAL.xlsx, habiendo

usado la fuente CWE (*Common Weakness Enumeration*) [53] para la realización de estas. Cabe destacar que en este trabajo no se ha comprobado la debida correspondencia entre plugin/script/módulo usado y tipo de ataque generado. Estas capturas alternativas siguen una clasificación idéntica a la descrita en el párrafo anterior, y pueden consultarse a través de https://github.com/fbuenoc97/TFG/tree/main/capturas-TFG/owasp_clasificacion.

En el Anexo C se detallan los pasos a seguir para la instalación de las herramientas utilizadas en este apartado, así como los comandos necesarios para estructurar el cuerpo del dataset de manera idéntica a la adoptada en este trabajo.

4.4 Detección de tráfico de ataque

En este apartado se ha comprobado la eficiencia en la detección de ataques presentes en el *dataset* previamente generado.

Las peticiones web que contiene las URI de ataque incluidas en el *dataset* anterior (ficheros de extensión *.uri* y *.short*) han sido procesadas mediante diversos sistemas de detección. Este trabajo se ha centrado en tres sistemas SIDS ampliamente extendidos y de uso gratuito:

- *Snort* [2]: software IDS genérico de amplia implantación. Utilizaremos las reglas de Talos [54] así como las reglas ETopen [55] actualizadas a 24/03/2022 con todas las reglas activas. Previamente, se han seleccionado las reglas que afectan únicamente al URI de peticiones HTTP, siguiendo el procedimiento indicado en [56].
- *ModSecurity* [3]: módulo WAF también de amplio uso por su fácil integración con Apache. Utilizaremos las reglas OWASP Core Ruleset (CRS) en su versión 3.3.2 y el nivel de paranoia 2.
- *Nemesida* [4]: es un WAF completo que incluye un conjunto de firmas de uso gratuito. Dicho conjunto proporcionaría una mayor tasa de falsos positivos que la versión de pago, lo que no afecta a este estudio.

El fichero de traza resultante en cada SIDS ha sido analizado a fin de correlar cuáles de las peticiones HTTP de entrada han sido detectadas como ataque por los distintos SIDS. Para ello, se ha hecho uso de la herramienta *InspectorLog*[40], que ha arrojado para cada fichero analizado dos archivos nombrados de igual manera que el anterior, uno contiene el *dataset* de URIs que han sido detectadas como ataques, con extensión *.attacks*, y el otro las URIs que por el contrario no fueron catalogadas como maliciosas, con extensión *.clean*.

Adicionalmente, se ha generado un fichero para cada herramienta de extensión *.attacks* que contiene las URIs que han sido detectadas por alguno de los tres SIDS, con el fin de tener datos acerca de un IDS ideal que utilizable los tres detectores de manera enlazada. A este último IDS comprendido por las reglas de los otros tres utilizados, se le ha llamado IDS/Detector Completo por comodidad.

Los resultados de detección de cada IDS, así como los resultados del IDS Completo se encuentran publicados en https://github.com/fbuenoc97/TFG/tree/main/detecciones_IL.

Los pasos seguidos para la instalación y uso de *InspectorLog* en colaboración con *Snort*, *ModSecurity* y *Nemesida* están detallados en Anexo D.

5 RESULTADOS

Tras llevar a cabo el desarrollo del estudio explicado en el capítulo anterior, se han obtenido unos resultados acordes a lo previsto. En este capítulo se pretende exponer, explicar y validar la consecución de dichos resultados.

Todos los resultados y ficheros del trabajo están disponibles para su consulta en <https://github.com/fbuenoc97/TFG/>

5.1 URIs generadas

Las 29 herramientas consideradas en el estudio han sido capaces de generar un *dataset* compuesto por un total de 254871 URIs sin repeticiones para la web estática, de las cuales 4805 han sido producidas por las 5 herramientas comerciales (excluyendo *Wpscan*³), y las 250066 restantes por aquellas de código abierto (opensource). Para la web dinámica, el *dataset* total, también sin repeticiones, asciende a 289741 URIs, 10797 lanzadas por las 6 herramientas comerciales y las 278944 restantes por las de código abierto (opensource).

En la Tabla 2 se desglosa la aportación al estudio de cada herramienta de manera individual para ambos contextos web, detallando información acerca de los *pcap* generados, así como las URIs lanzadas tanto contando las repeticiones como sin hacerlo.

Con el objetivo de minimizar el número de gráficas a presentar en el estudio, se ha optado por no realizar una gráfica individual que represente tan solo el número de URIs lanzadas por cada herramienta. En su lugar, se ha representado dicho número de URIs en otras gráficas junto a otros resultados que se han detallado en los siguientes apartados, con lo que se ha conseguido una comparativa y una visión más amplia de lo que se ha realizado en el trabajo.

Cabe señalar que, tras la captura del tráfico, se ha realizado una validación de la captura para evaluar si se ajusta al tráfico esperado, reajustando los parámetros del ataque en los casos en los que se consideró necesario. Esta inspección ha permitido verificar algunas peculiaridades, como que la mayoría de las herramientas realizan siempre una fase de escaneo previa al ataque en sí. Esta puede ser considerada una etapa temprana del ataque, según la taxonomía de Mitre ATT&CK [57], por lo que resulta de interés verificar si los SIDS son capaces de detectar este tipo de peticiones. Se observaron algunos casos de comportamientos no esperados, que han sido recogidos como anomalías del estudio.

³ *Wpscan* no ha sido lanzada para la web estática al no ser esta un recurso de *Wordpress*

Herramientas	N° pcap's generados		Tamaño total (MB) de esos pcap's		N° URIs CON repetición		N° URIs SIN repetición	
	Web Estática	Web Dinámica	Web Estática	Web Dinámica	Web Estática	Web Dinámica	Web Estática	Web Dinámica
Arachni	10	10	4	16,8	3385	17044	33	4950
Burpsuite	1	1	0,2573	6,6	254	2274	74	565
Commix	1	1	0,7698	13,7	790	20458	1	2
Golismo	1	1	7,8	4,1	8890	290	8515	207
Grabber	5	5	33,9	1000	2108	22358	1062	5692
GrendelScan	8	8	6,3	10,7	17835	19249	17770	17788
Havij	1	1	0,2009	0,1212	294	138	289	106
Ironwasp	13	13	10,8	0,2622	12301	582	1016	428
Metasploit	1	1	27,9	28,1	48039	48039	44465	44465
Nessus	1	1	48,1	43	49472	43508	18363	19277
Nexploit	1	1	12	24,3	7335	17063	1085	4286
Nexpose	1	1	2,6	2,6	5344	5344	2592	2592
Nikto	11	11	50,6	76,8	73619	76887	58944	61691
Nmap	9	9	2,6	2,6	2531	2946	2241	2587
Nuclei	14	14	6,9	2,9	8362	2076	2829	1179
Openvas	1	1	5,4	64,7	10492	50081	8473	45060
OWASPZAP	8	8	2,1	6,6	2109	9061	56	5758
Skipfish	1	1	154,1	42,2	74763	31813	56797	18463
SmartScanner	1	1	0,7057	2,6	1187	2280	994	1586
Sqlmap	1	1	0,0734	1,4	98	995	75	460
Vega	13	13	20,1	24,7	23342	53276	798	7192
W3af	26	26	63,7	117,7	7896	8879	630	2408
Wapiti	14	14	10,1	22,1	21783	44963	20645	36900
Watobo	1	1	2,5	0,5259	3734	852	2771	11
Webcruiser	1	1	1,4	3,5	1548	4827	60	1605
Wfuzz	4	4	4,7	187,9	569	571	564	562
Wikto	1	1	7,4	7,4	4272	4125	3721	3721
Wpscan ⁴	(No Aplica)	1	(No Aplica)	0,0735	(No Aplica)	166	(No Aplica)	163
Xsser	1	1	0,0202	0,056	24	61	8	37
Total OpenSource	146	146	470,0643	1674,3653	376708	458252	250066	278944
Total Comerciales	5	6	16,963	39,6735	15668	31954	4805	10797
TOTAL	151	152	487,0273	1714,0388	392376	490206	254871	289741

Tabla 2: Aportación al *dataset* de cada herramienta⁴ Wpscan no ha sido lanzada para la web estática al no ser esta un recurso de Wordpress

En total, se han aportado 254871 URIs para la web estática y 289741 para la web dinámica, lo cual se ha considerado cantidad suficiente para el estudio. Esto implica aproximadamente un incremento de URIs del 12% para la web dinámica, que puede obedecer al aumento de recursos web de esta con respecto a la estática, por tanto, es algo que no sorprende demasiado. Esta relación de las URIs lanzadas para ambos contextos web ha estado presente durante todo el estudio, manteniéndose similar en cada herramienta prácticamente, tan solo *Golismo*, *Skipfish* y *Watobo* han producido un número de URIs considerablemente mayor para la web estática que para la dinámica, debido a una mayor profundidad en la búsqueda activa de recursos de la página.

Del total de URIs generadas para cada web, no llega al 2% el número de URIs lanzadas por herramientas comerciales para la web estática, ascendiendo este porcentaje a casi el 4% para la web dinámica. Resulta complicado realizar una comparativa entre software de código abierto y comercial profunda, ya que los porcentajes citados anteriormente obedecen a la utilización de un número menor (tan solo 6 herramientas comerciales de las 29 usadas) de herramientas comerciales que de código abierto debido a la ausencia de licencias obtenidas. Aun así, de manera orientativa, se puede afirmar que en este estudio la aportación cualitativa de ataques al *dataset* de cada herramienta ha sido mayoritaria en las comerciales, entendiéndose por esto un tráfico de ataque que no ha presentado anomalías, generado a través de una interfaz de usuario cómoda y fácil de usar. En cuanto al aspecto cuantitativo, no se aprecian diferencias notables entre el software de pago y de código abierto.

Adicionalmente, se puede resaltar la versatilidad y diversidad de opciones que ofrecen muchas herramientas de código abierto, al contrario que algunas de pago, gracias precisamente a ser ejecutadas mediante la línea de comandos, herramientas como *Arachni*, *Nikto*, *Sqlmap* o *Xsfer* permiten que el usuario personalice cada uno de los parámetros utilizados en el ataque a realizar, esto ha permitido por ejemplo la separación en algunas herramientas de código abierto según el tipo de ataque lanzado. Esta personalización presente en las herramientas de código abierto es precisamente el motivo por el cual esta comparativa es orientativa, pues sería necesario un estudio más profundo de cada una de estas herramientas para hacer una comparativa real.

5.2 Separación de URIs de ataque según OWASP

De las 29 herramientas utilizadas en el estudio, un total de 12 han permitido separar el lanzamiento en función del tipo de vulnerabilidad o ataque que comprueban mediante la variación del *plugin*, módulo o *script* utilizado, como se ha mencionado en el capítulo anterior. Esto ha dado como resultado la captura de un tráfico que ha podido ser separado⁵ según la clasificación de OWASP [14], además de la propia separación en función de la herramienta que lo ha lanzado. Las tablas 3 y 4 recogen el número de URIs (sin repetición) que ha generado cada herramienta para cada tipo de ataque, los cuales siguen la siguiente numeración⁶: (1) *Broken Access Control*, (2) *Cryptografic Failures*, (3) *Injection*, (4) *Insecure Design*, (5) *Security Misconfiguration*, (6) *Identification and Authentication Failures*, (7) *SSRF*, (O) *Others*, donde la categoría *Others* corresponde a otros tipos de ataque no contemplados según el top 10 de OWASP [14].

⁵ En este estudio no se ha corroborado si cada URI generada pertenece a cada ataque en concreto.

⁶ No se han generado URIs para el resto de las categorías presentes en la lista de OWASP que no aparecen en la tabla.

Herramienta específica	1	2	3	4	5	6	7	O
Arachni	8		33		12			
Grabber			1050					13
Grendel-Scan	17184		233			91		271
Ironwasp	165	37	822				19	
Nikto	3522		1053		1089			54508
Nmap	1727					17		507
Nuclei	500		728				64	2710
OWASP-ZAP	16		41	11				
Vega	215		798					215
W3af								629
Wapiti	20633	0	15				1	9
Wfuzz	176		39					349
Total Comerciales	0	0	0	0	0	0	0	0
Total OpenSource	44146	37	4812	11	1101	108	84	59211
TOTAL	44146	37	4812	11	1101	108	84	59211

Tabla 3: Aportación herramientas según clasificación OWASP [14] para la web estática

Herramienta específica	1	2	3	4	5	6	7	O
Arachni	1271		3758		42			
Grabber			5684					11
Grendel-Scan	17202		233			91		271
Ironwasp	68	37	328				19	
Nikto	5946		1370		1089			61691
Nmap	37					12		2548
Nuclei	320		283				63	651
OWASP-ZAP	1811		3736	352				158
Vega	44		7192					32
W3af								2406
Wapiti	32790	94	3944				95	116
Wfuzz	173		43					347
Total Comerciales	0	0	0	0	0	0	0	0
Total OpenSource	59662	131	26571	352	1131	103	177	68231
TOTAL	59662	131	26571	352	1131	103	177	68231

Tabla 4: Aportación herramientas según clasificación OWASP [14] para la web dinámica

Para ilustrar de manera visual los datos anteriormente expuestos, se han generado los siguientes gráficos. En ellos, se detalla para cada herramienta que ha presentado separación entre tipos de ataques, el número de URIs⁷ que ha generado de cada uno de esos ataques individualmente, además del número total de URIs que ha generado la herramienta.

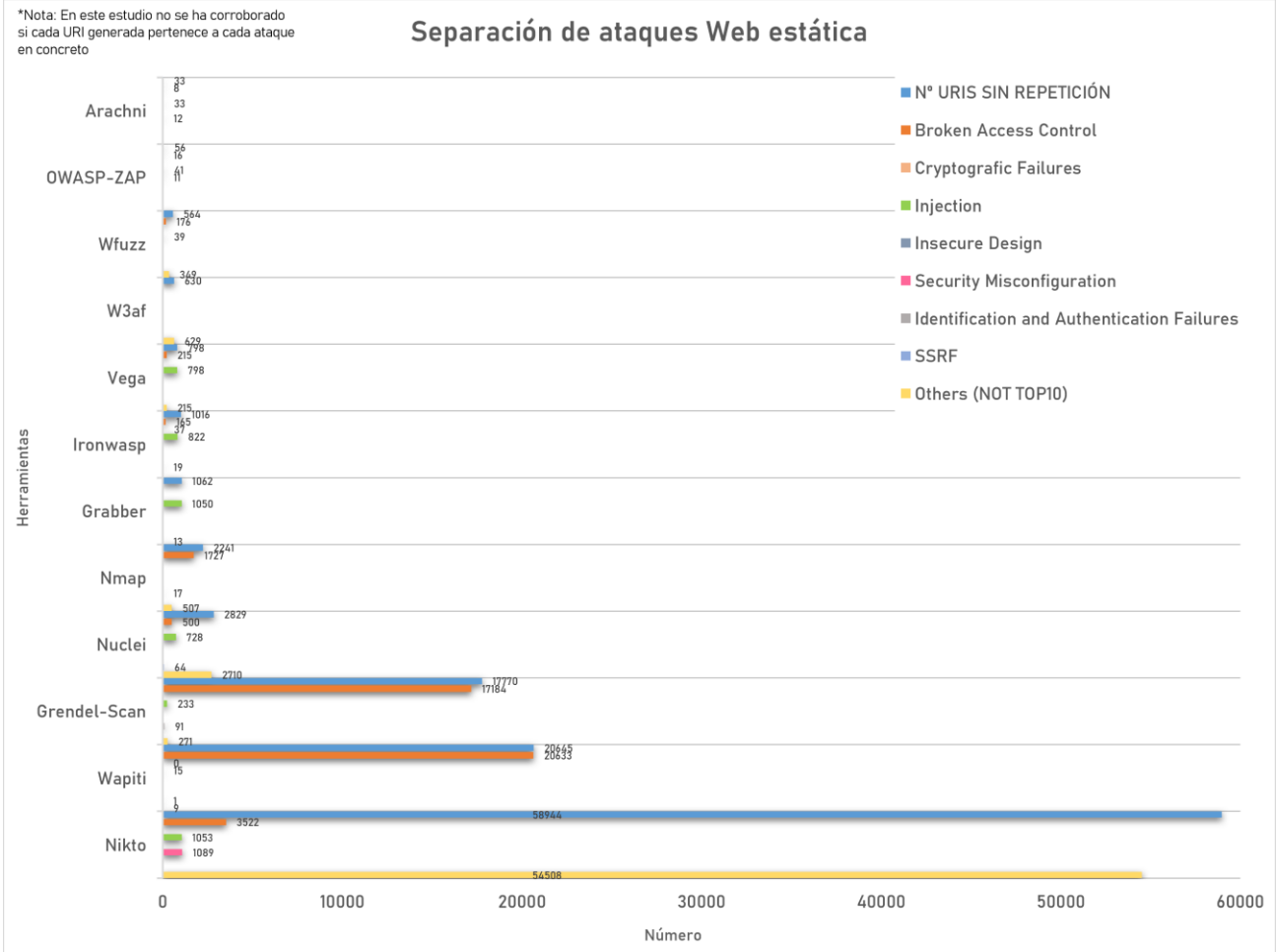


Figura 5: Separación de URIs según ataques y herramienta para la web estática

Cabe destacar que, debido a ciertas anomalías que se han observado y que serán detalladas en apartados posteriores, existe la posibilidad de que una URI concreta haya sido lanzada para dos o más tipos de ataque diferentes, con lo cual, el número total de URIs generadas por la herramienta no tiene por qué corresponderse con la suma individual de URIs generadas para cada tipo de ataque por dicha herramienta.

⁷ Sin tener en cuenta las repeticiones

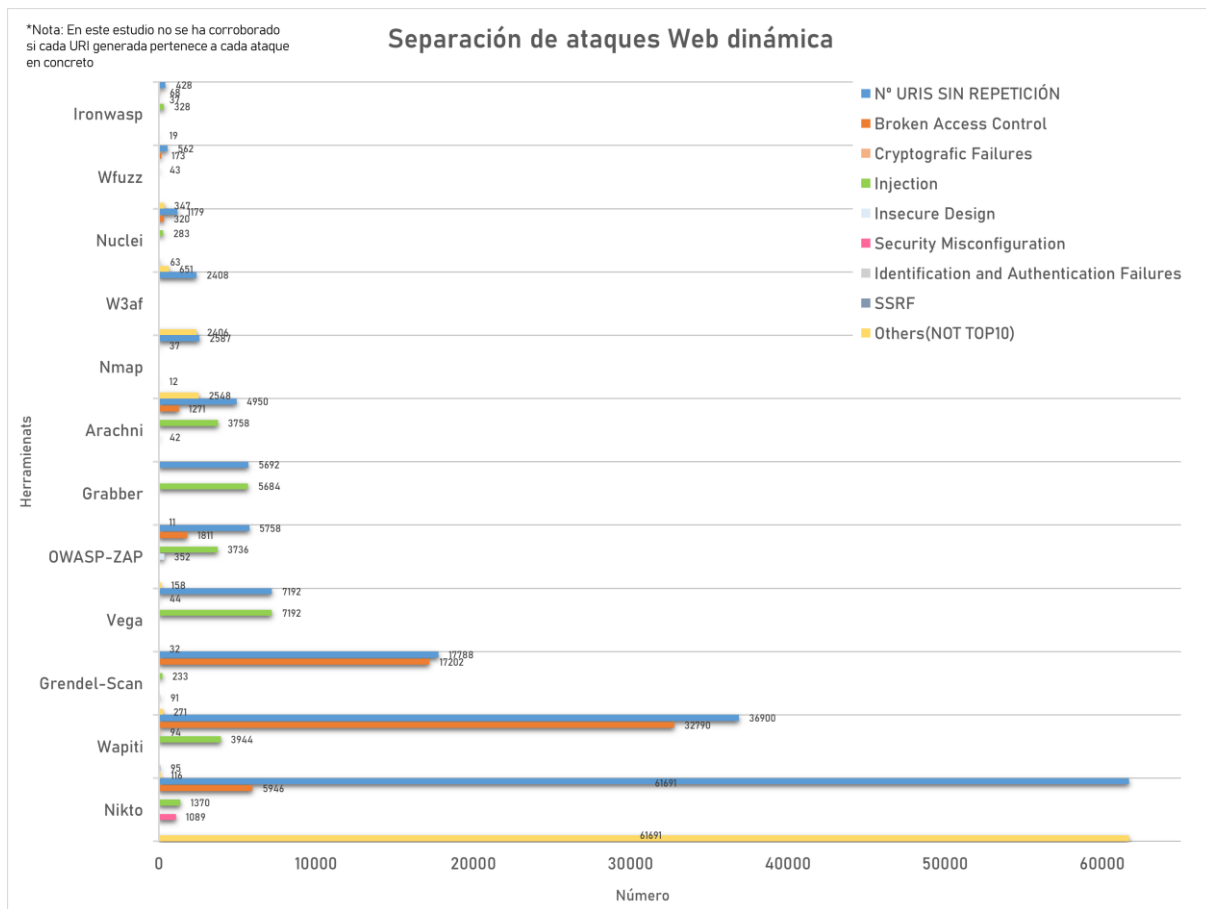


Figura 6: Separación de URIs según ataques y herramienta para la web dinámica

Para estas herramientas, en general se observa una amplia presencia de los tipos de ataque *Broken Access Control* e *Injection* para ambas webs, además, ciñéndonos a los *plugins* usados por cada herramienta, obtenemos una gran capacidad de separación en concreto para la categoría *Injection*, que permite elegir entre el tipo de inyección a realizar en la amplia mayoría de estas herramientas. Esto resulta bastante lógico si nos acogemos a la lista del top 10 de OWASP [14], ya que *Broken Access Control* e *Injection* se sitúan en las posiciones primera y tercera, respectivamente. Resulta llamativo el escaso número de ataques tipo *Cryptografic Failures*, categoría que ocupa el segundo lugar en la lista, aunque puede verse justificado en la novedad de la categoría, ya que en las listas previas no existía.

Adicionalmente, con el objetivo de conseguir una comparativa visual entre las herramientas comerciales y las de código abierto, se han generado dos gráficos más siguiendo la estructura de los anteriores, pero en esta ocasión separando las URIs en función de si han sido generadas por herramientas comerciales o de código abierto, además del total de todas las herramientas.

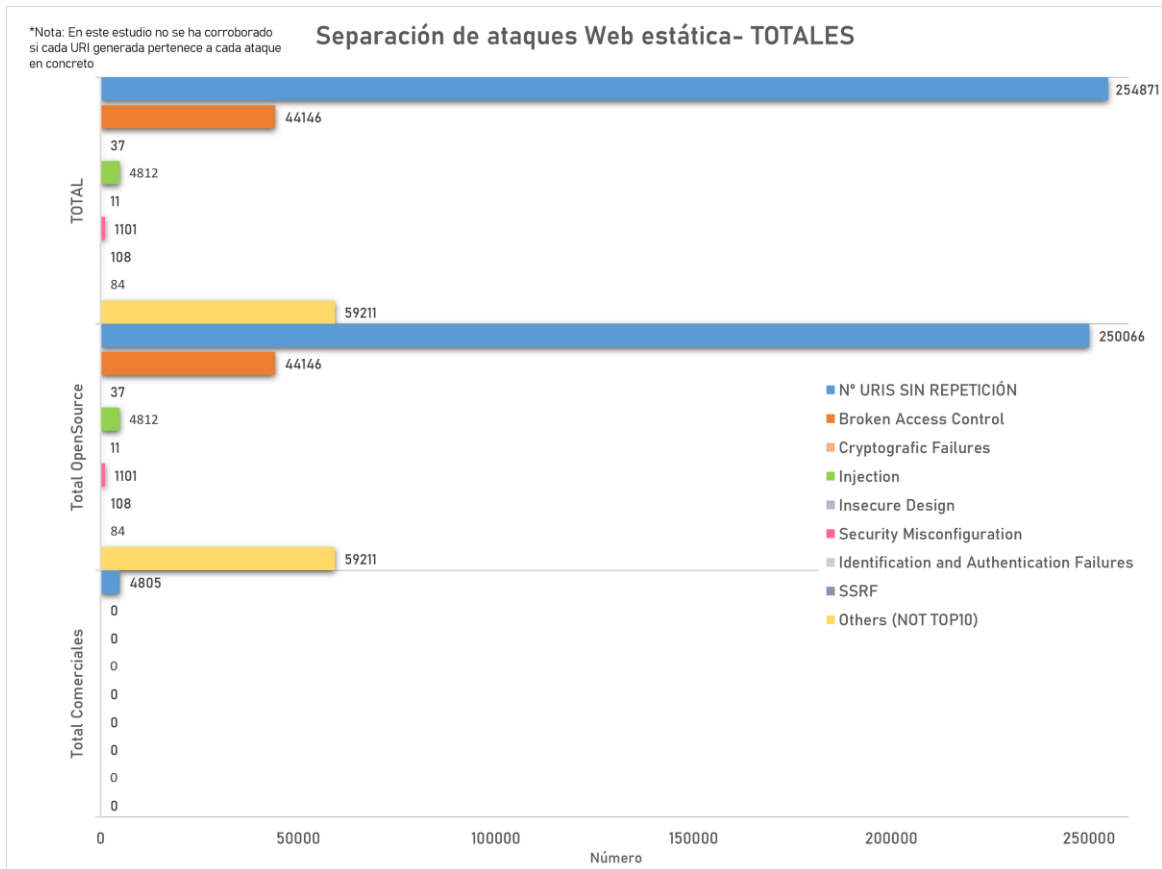


Figura 7: Separación de URIs según ataques y tipo de licencia para la web estática

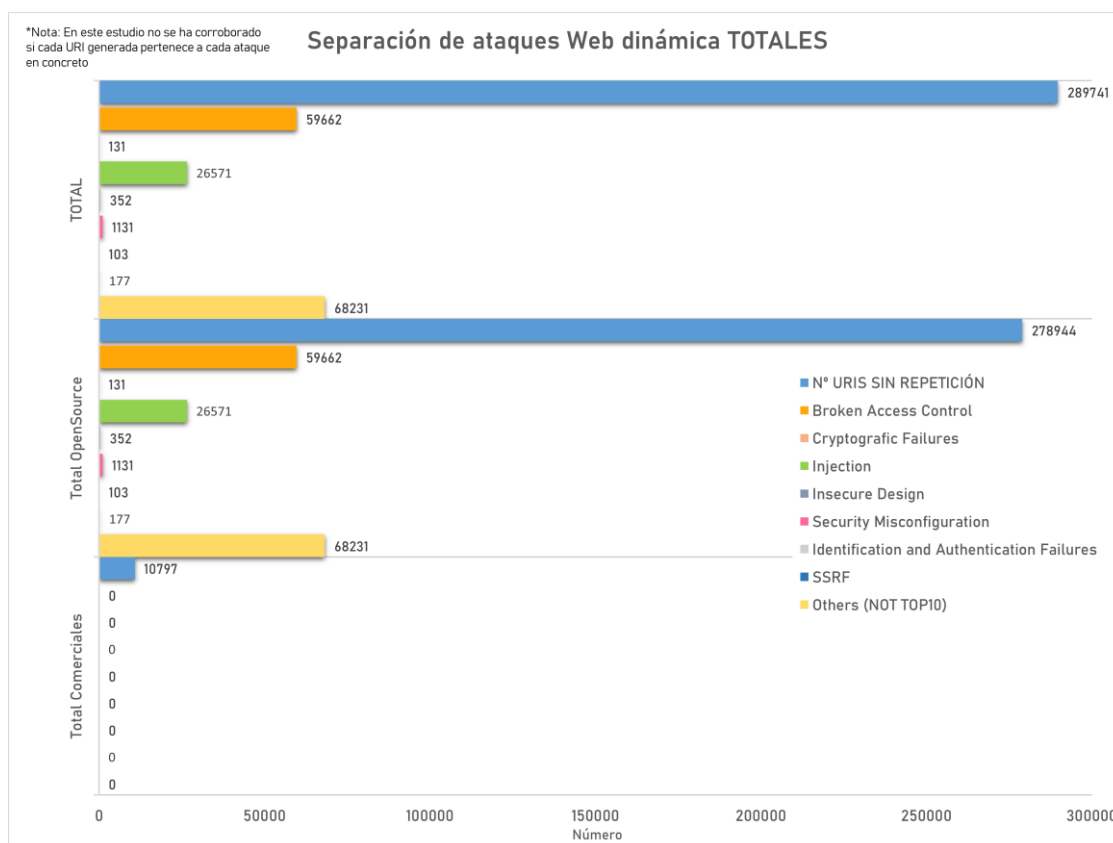


Figura 8: Separación de URIs según ataques y tipo de licencia para la web dinámica

Esta última comparativa podría carecer de sentido, ya que no se ha realizado separación de ataques en ninguna de las 6 herramientas comerciales utilizadas en el estudio, no obstante, ilustra de manera significativa el número total de URIs generadas que pertenecen a las herramientas comerciales y a las de código abierto. Además, se puede confirmar de manera más clara la presencia mayoritaria de los ataques *Broken Access Control* sobre todo, y también *Injection*, como se ha resaltado en gráficas anteriores.

5.3 Ataques detectados por cada IDS

En las tablas 5, 6 y 7 se exponen de manera resumida el número de URIs marcadas como ataques por cada uno de los IDS de manera individual, *Snort*, *Modsecurity* y *Nemesida*, respectivamente. Los datos están recogidos en ambos contextos web sobre el fichero de URIs sin repetición (extensión *short*). También se presenta una estimación⁸ de la eficiencia⁹ que presenta el detector, mediante la relación URIs detectadas/ URIs generadas.

⁸ No se están teniendo en cuenta los falsos positivos.

⁹ Resultado de dividir el nº ataques sin repetición entre el nº total de URIs sin repetición

Herramientas	Ataques SIN repeticion		% Eficiencia (sobre SIN repeticion)	
	Web Estática	Web Dinámica	Web Estática	Web Dinámica
Arachni	0	591	0%	12%
Burpsuite	15	173	20%	31%
Commix	0	0	0%	0%
Golismo	502	5	6%	2%
Grabber	184	391	17%	7%
GrendelScan	688	688	4%	4%
Havij	40	99	14%	93%
Ironwasp	10	10	1%	2%
Metasploit	14	14	0%	0%
Nessus	1331	1557	7%	8%
Nexploit	89	254	8%	6%
Nexpose	252	252	10%	10%
Nikto	3098	3210	5%	5%
Nmap	34	53	2%	2%
Nuclei	430	242	15%	21%
Openvas	578	6338	7%	14%
OWASPZAP	0	618	0%	11%
Skipfish	644	77	1%	0%
SmartScanner	4	90	0%	6%
Sqlmap	27	286	36%	62%
Vega	12	179	2%	2%
W3af	114	31	18%	1%
Wapiti	2	986	0%	3%
Watobo	1	0	0%	0%
Webcruiser	0	21	0%	1%
Wfuzz	338	334	60%	59%
Wikto	67	67	2%	2%
Wpscan ¹⁰	(No Aplica)	4	(No Aplica)	2%
Xsser	0	0	0%	0%
Total OpenSource	8114	15776	3%	6%
Total Comerciales	360	794	7%	7%
TOTAL	8474	16570	3%	6%

Tabla 5: N° URIs detectadas como ataque por Snort

¹⁰ Wpscan no ha sido lanzada para la web estática al no ser esta un recurso de Wordpress

Herramientas	Ataques SIN repetición		% Eficiencia (sobre SIN repetición)	
	Web Estática	Web Dinámica	Web Estática	Web Dinámica
Arachni	3	4155	9%	84%
Burpsuite	22	316	30%	56%
Commix	0	0	0%	0%
Golismo	3842	42	45%	20%
Grabber	1013	4901	95%	86%
GrendelScan	494	498	3%	3%
Havij	286	103	99%	97%
Ironwasp	222	177	22%	41%
Metasploit	6822	6822	15%	15%
Nessus	10701	9515	58%	49%
Nexploit	490	1391	45%	32%
Nexpose	484	484	19%	19%
Nikto	39172	41596	66%	67%
Nmap	80	118	4%	5%
Nuclei	1274	641	45%	54%
Openvas	1935	13457	23%	30%
OWASPZAP	0	4238	0%	74%
Skipfish	3379	839	6%	5%
SmartScanner	137	427	14%	27%
Sqlmap	67	456	89%	99%
Vega	118	2338	15%	33%
W3af	18	712	3%	30%
Wapiti	165	4800	1%	13%
Watobo	12	2	0%	18%
Webcruiser	5	408	8%	25%
Wfuzz	554	553	98%	98%
Wikto	639	639	17%	17%
Wpscan ¹¹	(No Aplica)	111	(No Aplica)	68%
Xsser	0	2	0%	5%
Total OpenSource	70796	96604	28%	35%
Total Comerciales	1138	3137	24%	29%
TOTAL	71934	99741	28%	34%

Tabla 6: N° URIs detectadas como ataque por ModSecurity

¹¹ Wpscan no ha sido lanzada para la web estática al no ser esta un recurso de Wordpress

Herramientas	Ataques SIN repetición		% Eficiencia (sobre SIN repeticion)	
	Web Estática	Web Dinámica	Web Estática	Web Dinámica
Arachni	3	3521	9%	71%
Burpsuite	31	327	42%	58%
Commix	0	0	0%	0%
Golismoero	2247	65	26%	31%
Grabber	637	3652	60%	64%
GrendelScan	2550	2574	14%	14%
Havij	246	0	85%	0%
Ironwasp	172	109	17%	25%
Metasploit	10702	10702	24%	24%
Nessus	7015	7780	38%	40%
Nexploit	184	832	17%	19%
Nexpose	585	585	23%	23%
Nikto	32826	32977	56%	53%
Nmap	109	321	5%	12%
Nuclei	1301	644	46%	55%
Openvas	2766	16750	33%	37%
OWASPZAP	0	3009	0%	52%
Skipfish	4065	1501	7%	8%
SmartScanner	218	528	22%	33%
Sqlmap	51	177	68%	38%
Vega	118	2319	15%	32%
W3af	30	25	5%	1%
Wapiti	823	4145	4%	11%
Watobo	6	3	0%	27%
Webcruiser	5	362	8%	23%
Wfuzz	462	454	82%	81%
Wikto	856	856	23%	23%
Wpscan ¹²	(No Aplica)	115	(No Aplica)	71%
Xsser	0	7	0%	19%
Total OpenSource	66985	91591	27%	33%
Total Comerciales	1023	2749	21%	25%
TOTAL	68008	94340	27%	33%

Tabla 7: N° URIs detectadas como ataque por Nemesida

¹² Wpscan no ha sido lanzada para la web estática al no ser esta un recurso de Wordpress

La capacidad de detección de los ataques web incluidos en el *dataset* por parte de los tres SIDS individualmente oscila entre un 3% y 28% de los ataques para la web estática, y entre un 6% y 34% para la web dinámica. Este porcentaje no implica necesariamente un bajo rendimiento de los sistemas de detección, y podría responder a que la fase de escaneo de un ataque no es suficientemente detectada, o al bajo porcentaje que aportan algunas herramientas que han presentado anomalías al porcentaje total del detector.

Los datos expuestos en las tablas anteriores se han ilustrado en los gráficos siguientes para obtener una comparativa visual. En ellos, se representa para cada una de las herramientas utilizadas en el estudio el número total de URIs generadas, así como cuántas de ellas han sido detectadas como ataques por cada IDS de manera individual. Adicionalmente, se representa la eficiencia presentada por cada IDS para cada herramienta, de tal manera que sea posible comparar tanto la capacidad de lanzamiento de las herramientas como la eficiencia de cada uno de los IDS. Se ha realizado un gráfico de este estilo para la web estática, y otro para la web dinámica.

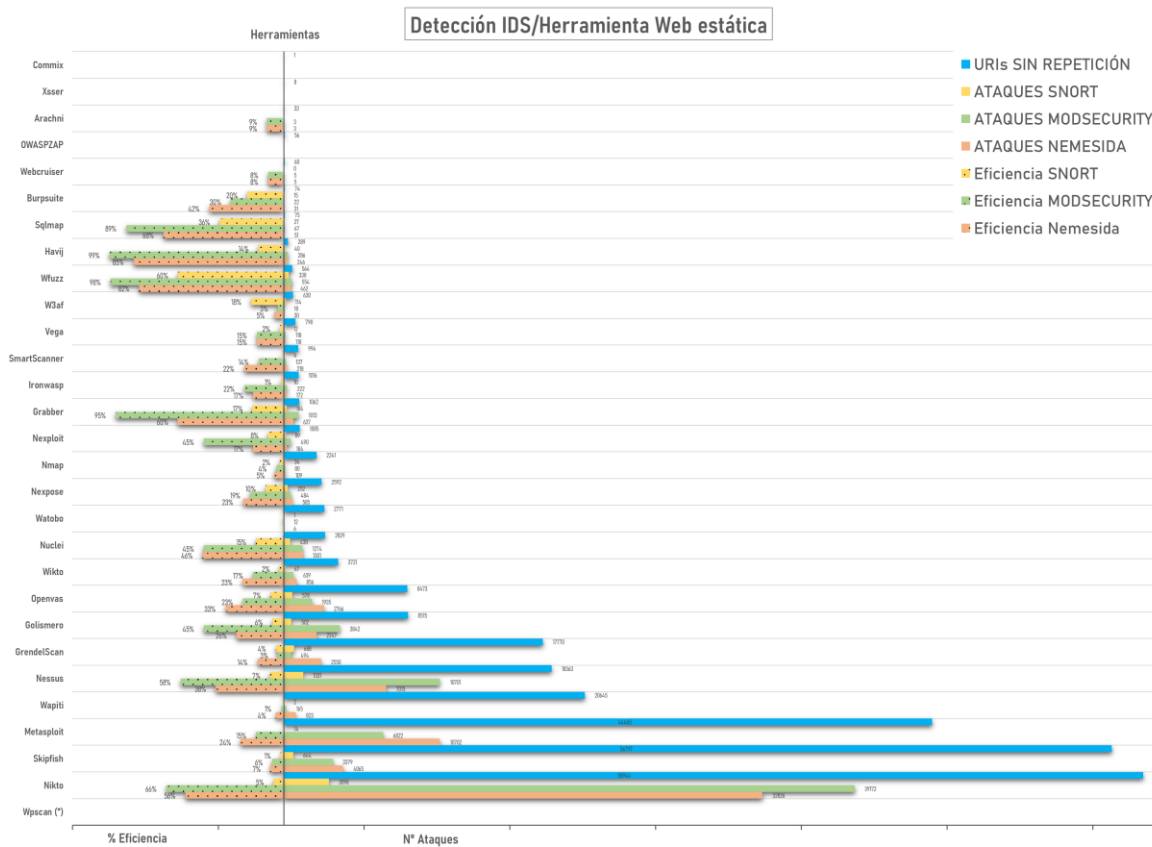


Figura 9: Detecciones de ataque según herramienta para la web estática

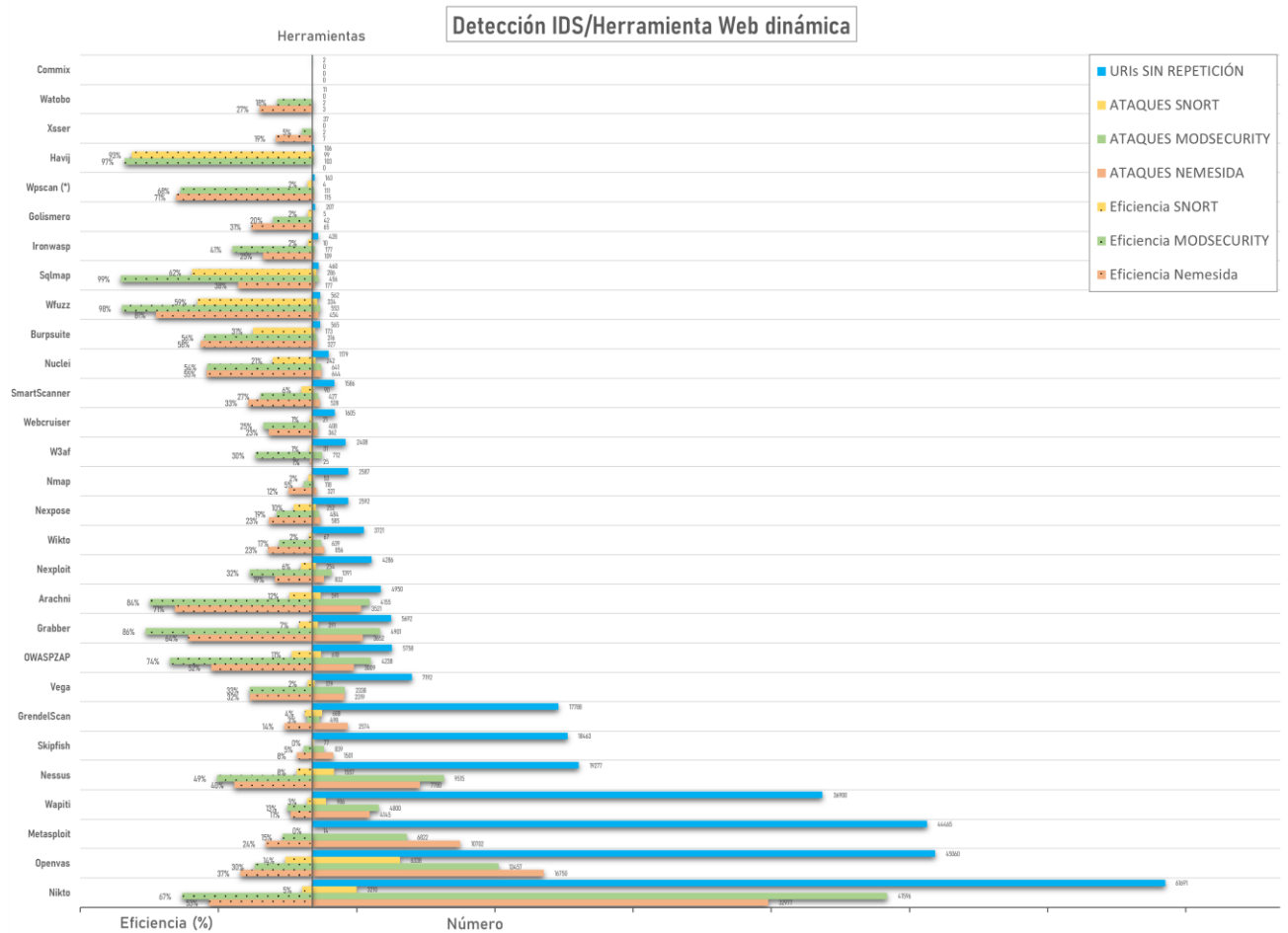


Figura 10: Detecciones de ataque según herramienta para la web dinámica

En la comparativa entre los IDS usados, *Snort* ha presentado unos resultados notablemente más bajos de detección que *ModSecurity* y *Nemesida*, presentando una eficiencia inferior al 17% en ambos contextos web para todas las herramientas excepto *Burpsuite*, *Sqlmap* y *Wfuzz*. Existe un caso llamativo en la herramienta *Havij*, que posee un 93% de detección para la web dinámica, algo que proporciona una estimación tanto de la eficiencia de la herramienta para realizar inyecciones SQL, como de la capacidad de *Snort* para detectarlas.

En cuanto a *ModSecurity* y *Nemesida*, se observa una mejora significativa en los resultados con respecto a *Snort* para cada una de las herramientas, presentando *ModSecurity* ligeramente unos mejores resultados, donde ahora sí se han encontrado altos porcentajes de detección en torno al 90% (*Havij*, *Grabber*, *Sqlmap*, *Wfuzz*), así como un rango de entre un 20% y 70% en el que se encuentran la gran mayoría de las herramientas (el resto son herramientas que han presentado anomalías y pueden caer hasta el 0% de detección). En el caso de *Nemesida*, la gran mayoría de herramientas oscilan entre un 10% y un 60% de tráfico detectado, ascendiendo hasta un 85% para *Havij* en la web estática, y un 82% en ambos contextos web para *Wfuzz*.

De manera análoga al apartado anterior, se ha proporcionado una comparativa total de manera gráfica entre las herramientas comerciales y las de código abierto, siguiendo el mismo patrón de representación de URIs, ataques y eficiencia mostrado en las gráficas de este mismo apartado.

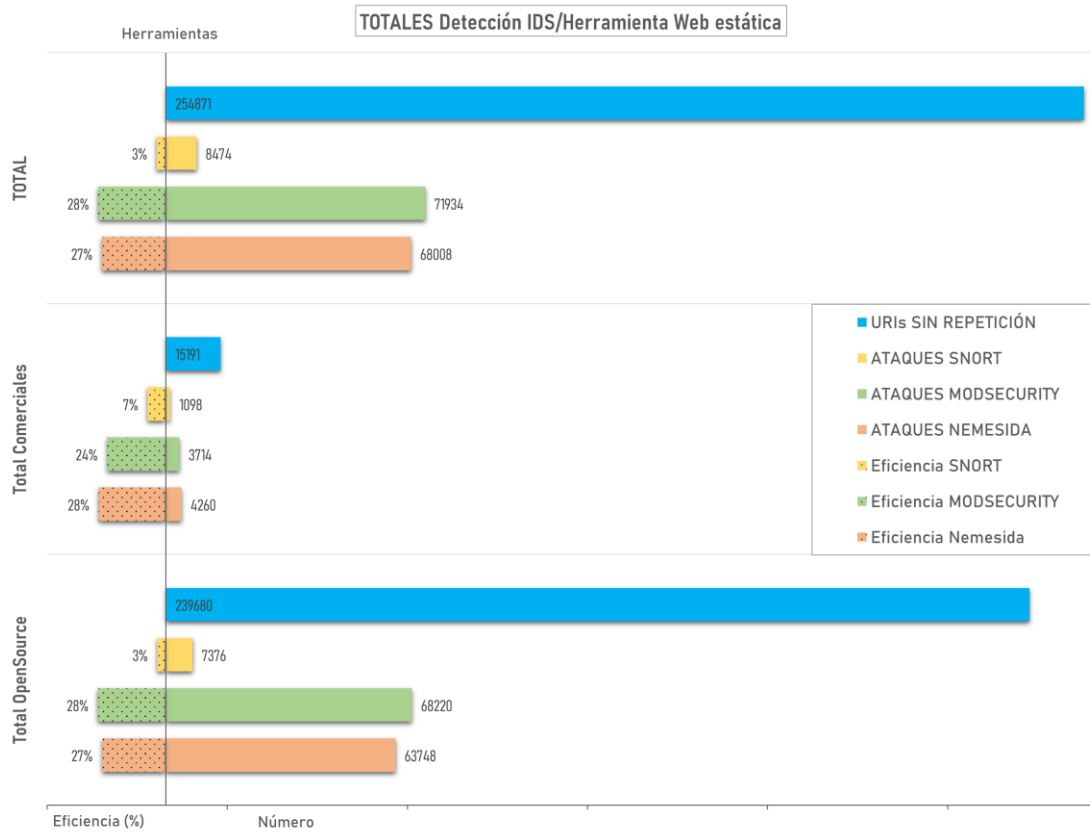


Figura 11: Detecciones de ataque según licencia de la herramienta para la web estática en cada IDS por separado

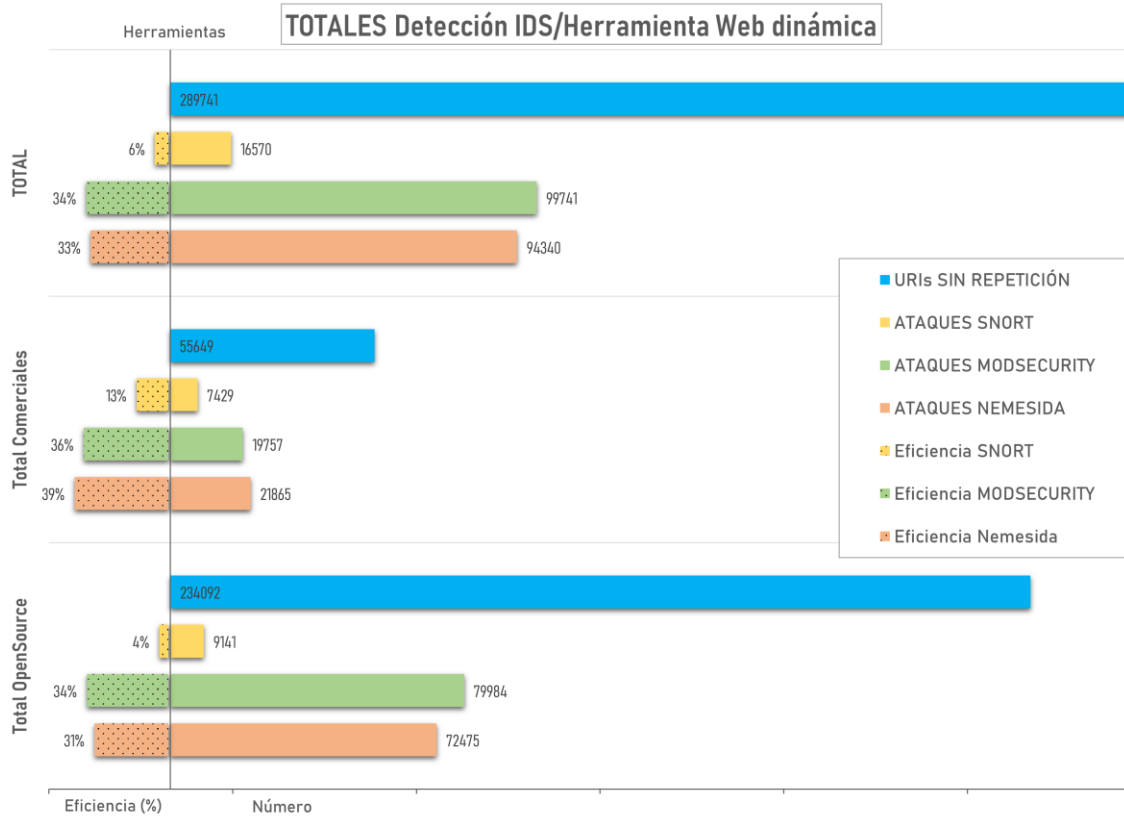


Figura 12: Detecciones de ataque según licencia de la herramienta para la web dinámica en cada IDS por separado

En estas últimas gráficas se observa como *Snort* tan solo se ha llegado a un 3% para la web estática y un 6% para la web dinámica, confirmándose así la mejora en la capacidad de detección de *ModSecurity* y *Nemesida*, donde se han obtenido unos resultados bastante parejos, siendo estos un 28% y 27% respectivamente para la web estática, y un 34% y 33% para la web dinámica.

En cuanto la comparativa entre herramientas comerciales y de código abierto, se observa cómo se mantienen unos porcentajes parecidos para cada IDS indistintamente del tipo de licencia de la herramienta. No obstante, resalta el pequeño despunte de la eficiencia de detección en *Snort* para las herramientas comerciales con respecto a las de código abierto, siendo esta un 7% para la web estática y un 13% para la web dinámica, ligeramente superior al 3% y 4% presente en las herramientas de código abierto para ambas webs respectivamente. Esto último sorprende, ya que no es la tónica habitual obtenida en los resultados a lo largo de todo el estudio.

5.4 Ataques detectados por el IDS completo

En la tabla 8 se muestran los resultados de igual manera que en el apartado anterior, pero en esta ocasión para el IDS completo resultado de enlazar los 3 IDS utilizados en este estudio. Esto proporcionará una visión conjunta acerca de la eficiencia que tendría esta posible solución de seguridad en cuanto a entornos web se refiere.

Herramientas	Ataques por alguno de los 3 SIN repeticion		% Eficiencia completa (sobre SIN repeticion)	
	Web Estática	Web Dinámica	Web Estática	Web Dinámica
Arachni	6	4264	18%	86%
Burpsuite	32	428	43%	76%
Commix	0	0	0%	0%
Golismo	4743	74	56%	36%
Grabber	1013	4901	95%	86%
GrendelScan	3216	3240	18%	18%
Havij	286	103	99%	97%
Ironwasp	259	186	25%	43%
Metasploit	10779	10779	24%	24%
Nessus	11780	11042	64%	57%
Nexploit	493	1403	45%	33%
Nexpose	689	689	27%	27%
Nikto	41601	44118	71%	72%
Nmap	125	351	6%	14%
Nuclei	1554	726	55%	62%
Openvas	3010	18363	36%	41%
OWASPZAP	0	4333	0%	75%
Skipfish	5223	1724	9%	9%
SmartScanner	227	616	23%	39%
Sqlmap	71	456	95%	99%
Vega	118	2362	15%	33%
W3af	132	748	21%	31%
Wapiti	861	5583	4%	15%
Watobo	13	3	0%	27%
Webcruiser	5	410	8%	26%
Wfuzz	558	558	99%	99%
Wikto	1002	1002	27%	27%
Wpscan ¹³	(No Aplica)	119	(No Aplica)	73%
Xsser	0	7	0%	19%
Total OpenSource	86350	114923	35%	41%
Total Comerciales	1446	3665	30%	34%
TOTAL	87796	118588	34%	41%

Tabla 8: N° URIs detectadas como ataque por el IDS completo

En la tabla anterior se puede observar un lógico aumento para ambos contextos web en las capacidades de detección con respecto a los IDS por separado, este aumento da como resultado un 34% de detección para la

¹³ Wpscan no ha sido lanzada para la web estática al no ser esta un recurso de Wordpress

web estática y un 41% para la dinámica. De nuevo, es necesario resaltar que estos resultados pueden verse alterados por aquellas herramientas que presentaron anomalías en el lanzamiento o en el número de URIs, o por un escaso porcentaje de detección del tráfico de sondeo previo al ataque, lo cual se ha detallado en el siguiente apartado.

Al igual que en apartados anteriores, se ha procedido a ilustrar de manera gráfica estos resultados para obtener una comparativa visual. En este caso, para cada herramienta se representa el número de URIs generadas para la web estática y dinámica por separado, con el correspondiente número de URIs detectadas como ataque en cada contexto web, acompañados también por la eficiencia de detección correspondiente.

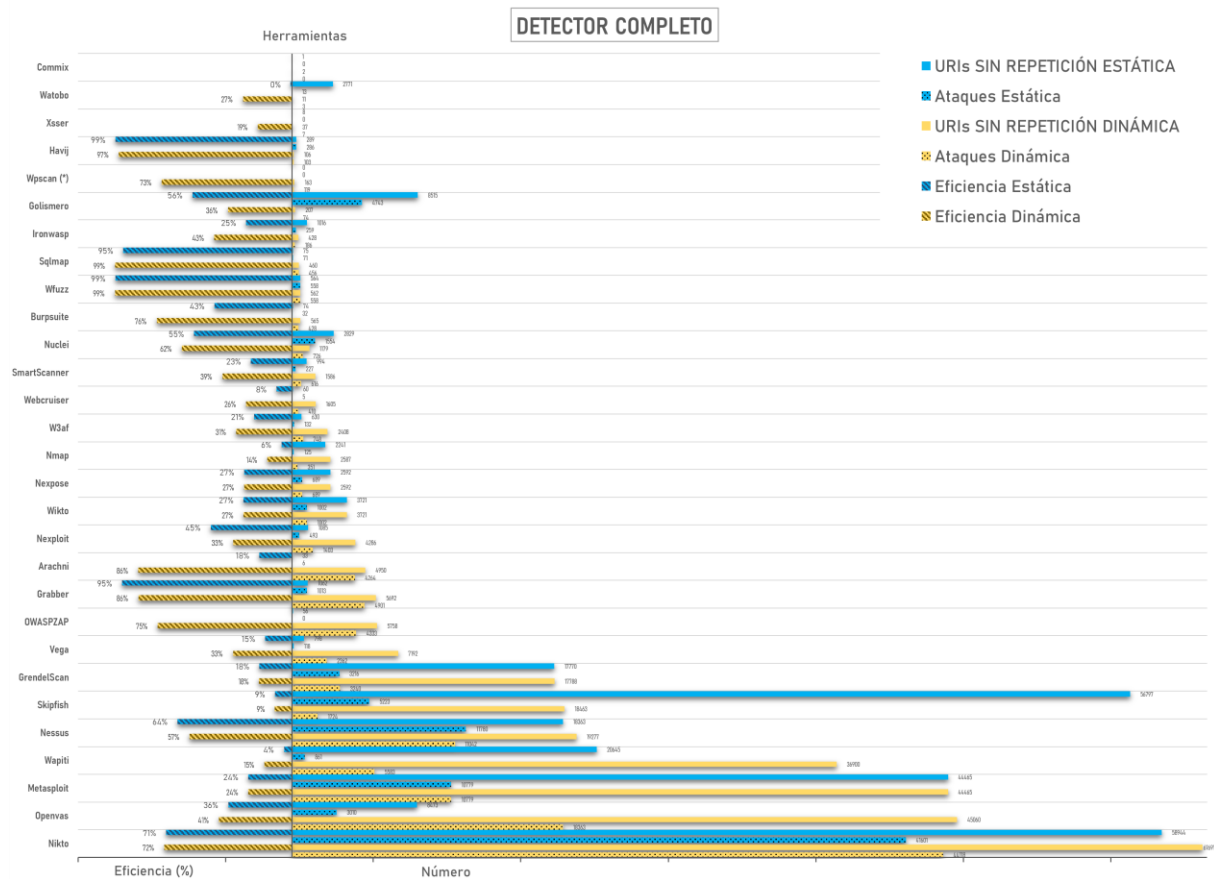


Figura 13: Detecciones del IDS completo para cada herramienta

Tras adoptar esta solución, se observa una mejora bastante significativa con respecto a los resultados presentados de cada IDS de manera individual, rozando un porcentaje de detección del 100% en varias herramientas (*Havij*, *Sqllmap*, *Wfuzz*, *Grabber*) en ambos contextos web, y superando el 20% de eficiencia en la gran mayoría de las herramientas. No obstante, continúa habiendo ciertas herramientas (*Commix*, *Xsfer*, *OWASP-ZAP*, *Wapiti*) que presentan unos resultados anormalmente bajos en alguno de los contextos, lo cual será recogido y estudiado en el apartado siguiente correspondiente a las anomalías.

A continuación se muestra en un gráfico la comparativa entre las herramientas comerciales y de código abierto, como viene siendo habitual a lo largo del trabajo.

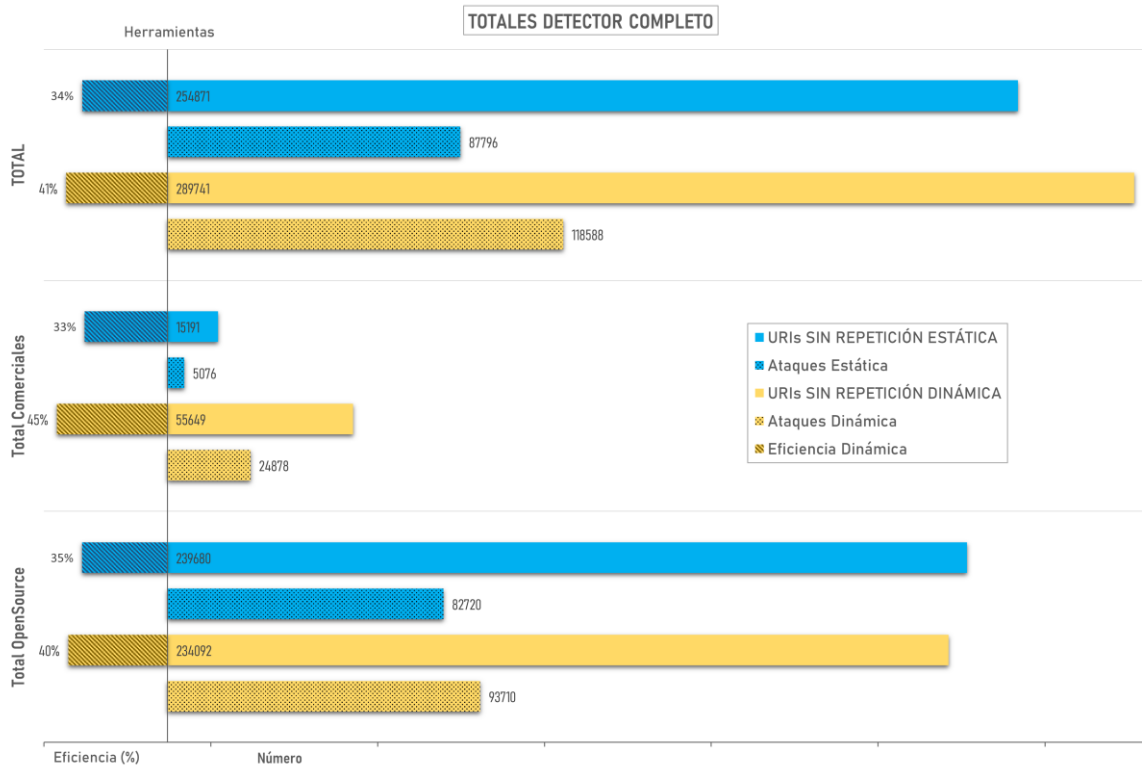


Figura 14: Detecciones del IDS completo diferenciando tipo de licencia de las herramientas

En la figura anterior se observa una eficiencia bastante similar para ambos tipos de herramientas, siendo ligeramente superior el porcentaje la eficiencia en el contexto web dinámico, lo que viene siendo habitual en todo el estudio. Estos porcentajes ascienden en total a un 34% para la web estática y un 41% en el caso de la web dinámica, lo que indica un porcentaje bastante decente y representativo del estudio, teniendo en cuenta tanto las anomalías presentadas, como la presencia de la fase de escaneo en el *dataset* capturado.

5.5 Anomalías encontradas en el estudio

Tras el análisis tanto del *dataset* generado por las herramientas como de los resultados de detección obtenidos en el apartado anterior, se han observado algunas anomalías presentes en el estudio que han podido condicionar de manera parcial los resultados de este. Ha sido posible resumir las anomalías encontradas en las siguientes:

1. Resultados muy parecidos, en ocasiones iguales ante:
 - 1.1. Web estática y dinámica.
 - 1.2. Dos lanzamientos sobre misma web.
2. Generación errónea de tráfico tras seleccionar un tipo de ataque concreto(e.g. “SQL Injection”):
 - 2.1. Se generan ataques de varios tipos, incluyendo el seleccionado.
 - 2.2. No se genera ningún ataque.
 - 2.3. Todas las URIs generadas de cada tipo de ataque son iguales.
3. Comportamiento ante recurso web SI (200) / NO (400) encontrado: no difiere
4. Nº de URIs de ataque
 - 4.1. Más del 90% son URIs de sondeo (frente a un 10% o menos de URIs de ataque).
 - 4.2. Excesivas repeticiones en las URIs de ataque

Se han sugerido unos posibles motivos causantes de las anomalías mencionadas, no obstante, el análisis en profundidad del motivo de estas anomalías presentadas por las herramientas no ha sido realizado en el desarrollo de este estudio. Estos posibles motivos sugeridos son:

- A. La herramienta lanza peticiones sin adaptar este lanzamiento al destino. Dichas peticiones las genera posiblemente a través de una base de datos propia de tráfico de "sondeo" donde prueba archivos, rutas y extensiones comunes en páginas web, o también tráfico de ataque. Este problema puede ser la causa de varias de las anomalías detectadas.
- B. La herramienta no realiza ningún ataque en la cabecera HTTP “http.request.uri”, pero si en otras cabeceras.
- C. La herramienta tan solo realiza ataques en la web dinámica, habiendo realizado exactamente los mismos pasos en ambos contextos. Posible motivo: La herramienta no encuentra ningún recurso web en la web estática para realizar el ataque.
- D. La herramienta no realiza ataques de ningún tipo en ningún contexto de los probados, según indica la propia herramienta, debido a no encontrar ninguna vulnerabilidad (URI concreta donde atacar mediante inyección, etc). Para las herramientas que han presentado esta anomalía, se ha intentado indicar manualmente la URI de un recurso real del web objetivo (e.g. web/wp-login.php), obteniendo el mismo resultado.
- E. La herramienta, con capacidad de separación entre ataques aparentemente, sí diferencia el tráfico enviado en función del destino, pero realiza ataques de varios tipos indistintamente del ataque que se le esté indicando.

A continuación, se muestra una tabla donde se indica qué herramientas han presentado las anomalías descritas anteriormente en sus URIs generadas, sugiriendo el motivo causante de la anomalía con su letra correspondiente dentro de cada celda.

Herramienta	1.1	1.2	2.1	2.2	2.3	3	4.1	4.2
Arachni				C				
Burpsuite								
Commix				B				
Golismero								
Grabber								
Grendel-Scan	A	A		A		A	A	
Havij								
Ironwasp								
Metasploit	A	A					A	
Nessus								
Nexploit								
Nexpose	A	A				A	A	
Nikto	A	A	A		A			
Nmap	A	A				A	A	
Nuclei								
Openvas								
OWASP-ZAP				C				
Skipfish	A	A				A	A	
SmartScanner								
Sqlmap								
Vega					E			E
W3af				D				
Wapiti				C				
Watobo				D				
Webcruiser								
Wfuzz	A	A				A		
Wikto	A	A					A	
Wpscan ¹⁴						A	A	
Xsser								

Tabla 9: Anomalías en las URIs generadas

¹⁴ Wpscan no ha sido lanzada para la web estática al no ser esta un recurso de Wordpress

6 CONCLUSIONES Y LÍNEAS DE AVANCE

6.1 Conclusiones

La disponibilidad de *datasets* adecuados es clave para acelerar la investigación en el campo de los IDS. En este trabajo se ha generado un *dataset* propio de ataques que se ha puesto a disposición de la comunidad investigadora. Este *dataset* se ha generado incorporando todos los posibles tipos de ataques que ofrecen las principales herramientas disponibles contra dos escenarios web: uno estático y otro dinámico. No obstante, la aportación a este *dataset* de cada herramienta ha sido bastante dispar en cuanto al número de URIs que ha generado cada una de ellas, y queda muy marcado por la tipología seguida en cada ataque. Todo ello hace que se haya obtenido un *dataset* bastante completo y representativo, ya que se han utilizado una amplia mayoría de las principales herramientas de seguridad en entornos web, pero también es mejorable, ya que es posible ahondar más en la parte de generación del tráfico, ya sea para reducir el tráfico de sondeo previo al ataque o para minimizar en número de anomalías.

Este trabajo ha aportado también una amplia lista de las herramientas más punteras en seguridad en entornos web, sobre todo de código abierto. No solo se aporta una lista de las herramientas, sino también una comparativa en cuanto a la generación de ataques de cada una de ellas. Además, este trabajo proporciona los pasos de instalación y uso exactos para replicar el escenario y las pruebas realizadas en el mismo, en mi opinión un aporte bastante significativo, ya que una de las mayores dificultades encontradas durante el estudio y que más tiempo han requerido, ha sido la correcta instalación y uso de cada una de las herramientas utilizadas, así como su previa selección.

En cuanto a los resultados de detección, el desarrollo del trabajo y la obtención de los resultados indican claramente la inferioridad en la eficiencia de detección que posee *Snort* con respecto a *ModSecurity* y *Nemesis*, lo cual puede obedecer a que es el único IDS de los estudiados que no está integrado en un WAF. No obstante, durante todo el estudio no se han tenido en cuenta los falsos positivos, lo que podría hacer variar los resultados obtenidos y se acercaría más aún a una capacidad de detección real.

Conforme a los resultados obtenidos de la solución propuesta como Detector Completo, se observa una capacidad de detección moderada en general, bastante marcada por aquellas herramientas que presentaron características mencionadas durante todo el desarrollo del estudio, como son la fase de escaneo presente en el *dataset* y las anomalías correspondientes. A pesar de todo ello, creo firmemente que el Detector Completo sería una gran solución adoptada para la seguridad web, ya que para bastantes herramientas que además no han presentado anomalías y se han ceñido estrictamente a realizar ataques, ha presentado unos resultados de eficiencia en la detección bastante altos. Además, no debemos olvidar que es una solución totalmente gratuita.

En definitiva, creo que en este trabajo se ha abordado de manera muy amplia el uso de las herramientas de ataque, proporcionando información y resultados detallados acerca de qué herramientas son más eficaces según se pretenda hacer un análisis de vulnerabilidades (*Burpsuite*, *Vega*, *Nessus*, *Nikto*...) o una explotación de esta (*Sqlmap*, *Arachni*, *Havij*...). Además, la comparativa en la eficiencia de detección de cada uno de los IDS utilizados por separado ha permitido comprobar la herramienta tan potente y útil que es *ModSecurity*, en mi opinión superior a las demás estudiadas.

6.2 Limitaciones

Como limitaciones de este trabajo pueden destacarse el uso escaso de herramientas comerciales frente a las de código abierto y la limitación a dos escenarios de aplicación web sobre las que atacar, lo que restaría capacidad de generalización a los resultados. Aunque los dos escenarios utilizados son de amplia implantación, el uso de otros portales y aplicaciones/servicios web enriquecerían la aplicabilidad de los resultados.

Es importante resaltar que los resultados de este estudio se ven sujetos parcialmente tanto a las anomalías encontradas en el *dataset*, como a la escasa discriminación en la detección por tipo o fase del ataque, además de no estar teniendo en cuenta los falsos positivos para la eficiencia en la detección.

6.3 Líneas de avance

Como principal línea de avance se propone la mejora sustancial del *dataset* propuesto para aumentar la precisión del estudio, así como la inclusión de los falsos positivos en el cálculo de la eficiencia de detección de cada IDS.

A parte, a modo de enriquecer la representatividad del estudio, se podrían extender los resultados a otros escenarios dinámicos más complejos, así como a una mayor cantidad de herramientas comerciales.

A continuación se proponen posibles soluciones que se sugieren para solventar, en una continuación del estudio, cada uno de los posibles motivos causantes de las anomalías identificadas. No obstante, recomiendo en una posible continuación del estudio una revisión minuciosa del *dataset*, a modo de comprobar las anomalías personalmente. Se ha seguido la misma notación (A-E) que para los motivos de las anomalías, asociando un motivo con una posible solución:

- A. Se sugiere el estudio minucioso de la herramienta que ha presentado este problema, analizando cómo realiza los ataques, de qué manera genera las peticiones, si sigue algún patrón o busca algún tipo de recurso en el destino, etc.
- B. Se recomienda ampliar el estudio a más tipos de ataques para obtener una mayor reproducción de un entorno real, no solo ataques en la cabecera *http_request_uri*.
- C. Sería recomendable ampliar los recursos de la web estática, proporcionándole suficientes recursos a las herramientas para realizar ataques representativos.
- D. La solución de este problema debería conocerse siguiendo el mismo procedimiento que en A.
- E. La solución de este problema debería conocerse siguiendo el mismo procedimiento que en A.

ANEXO A: DESPLIEGUE DE PÁGINAS WEB

En este Anexo se expondrán los pasos y comandos necesarios para replicar el montaje del escenario web utilizado en el trabajo. Además, se proporciona un pequeño gráfico de red a fin de aclarar el montaje del escenario, indicando qué ha sido instalado en cada equipo. Ambas páginas web han sido desplegadas sobre un servidor *Apache* en el sistema operativo *Centos 7* [43].

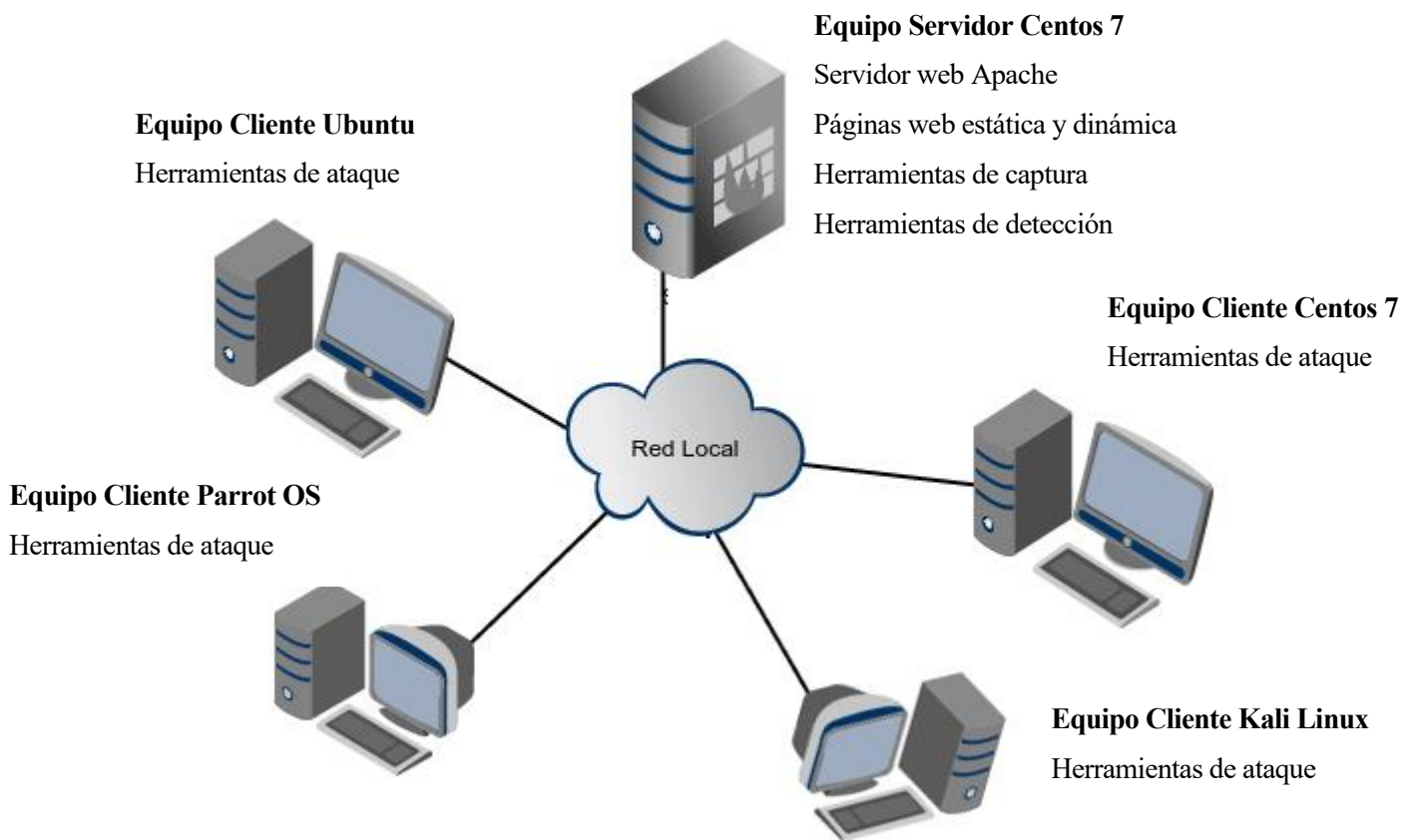


Figura 15: Esquema de red propuesto

A. Servidor Web Apache

El servicio ha sido instalado con permisos de administrador mediante el comando por consola `yum install httpd -y`. Para la correcta publicación de ambas páginas web en el servidor, se ha incluido la siguiente directiva en la configuración del servidor:

```
<Directory /var/www/html/web/>
    AllowOverride All
</Directory>
```

Esto último se ha hecho mediante la copia de la directiva anterior en un fichero (disponible en https://github.com/fbuenoc97/TFG/blob/main/ficheros_webs/web_tfg.conf) que ha sido movido al directorio `/etc/httpd/conf.d`

Por último, para aplicar los cambios, ejecutamos el comando `service httpd restart`.

B. Página Web Estática

La página web estática proviene de un repositorio público de *github* [58] de donde se ha descargado mediante el comando `wget https://github.com/cloudacademy/static-website-example` y situado en el directorio `/var/www/html/web`. Posteriormente, se ha reiniciado el servidor con `service httpd restart`.

C. Página Web Dinámica

Para el correcto despliegue de esta web ha sido necesaria la instalación manual de *php* versión 7 mediante los siguientes comandos:

```
yum -y install http://rpms.remirepo.net/enterprise/remi-release-7.rpm yum
utils
yum-config-manager --enable remi-php73 -y
yum install -y php php-common php-mysql php-gd php-xml php-mbstring php
mcrypt
```

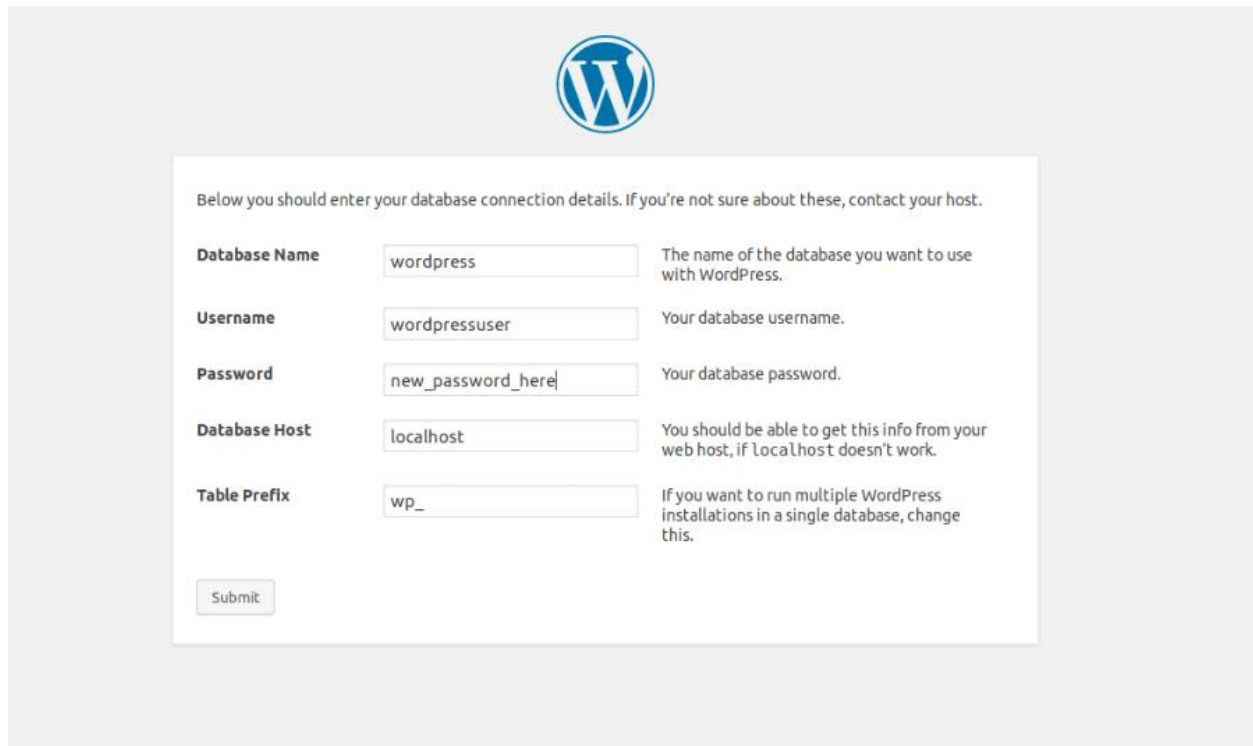
También ha sido necesaria la instalación del servicio *mariadb*, y la creación de una base de datos y un usuario mediante:

```
yum install mariadb-server mariadb -y
service mariadb start
mysql
create database wordpress;
CREATE USER 'felbuecar'@'localhost' IDENTIFIED BY 'XXXX';
GRANT ALL PRIVILEGES ON wordpress.* TO 'felbuecar'@'localhost';
FLUSH PRIVILEGES;
exit;
service mariadb restart
```

Tras esto se ha procedido a generar la página web dinámica con *WordPress* y moverla al directorio correspondiente para su publicación:

```
wget http://wordpress.org/latest.tar.gz -xzf latest.tar.gz
mv wordpress/ web
rm -rf latest.tar.gz
cp -rf web/ /var/www/html
sudo chown -R apache:apache /var/www/html/*
service httpd restart
```

Mediante el acceso vía web a http://IP_SERVIDOR/web se ha continuado con la configuración del sitio web gracias a la interfaz gráfica que *WordPress* facilita, la cual se muestra en la siguiente imagen.



The image shows the WordPress database configuration screen. At the top center is the WordPress logo. Below it, a text box contains the instruction: "Below you should enter your database connection details. If you're not sure about these, contact your host." The form consists of five rows, each with a label, an input field, and a description:

Field Label	Input Value	Description
Database Name	wordpress	The name of the database you want to use with WordPress.
Username	wordpressuser	Your database username.
Password	new_password_here	Your database password.
Database Host	localhost	You should be able to get this info from your web host, if localhost doesn't work.
Table Prefix	wp_	If you want to run multiple WordPress installations in a single database, change this.

At the bottom left of the form is a "Submit" button.

Figura 16: Configuración de sitio web *WordPress*

Mediante la interfaz descrita anteriormente se ha vinculado la base de datos previamente generada (indicando correctamente el nombre de la base de datos, nombre de usuario y la contraseña) con la página web, ya accesible vía web mediante http://IP_SERVIDOR/web.

ANEXO B: INSTALACIÓN Y USO DE HERRAMIENTAS DE ATAQUE

En este Anexo se expondrán los pasos y comandos necesarios para replicar la instalación y el uso de las herramientas de ataque utilizadas en el trabajo, las cuales han sido probadas en una máquina virtual con el sistema operativo *Centos 7* [43], *Ubuntu* [46], *Kali* [47] o *Parrot OS* [48].

A. Herramientas OpenSource

1. Arachni

Esta herramienta ha sido probada en el entorno *Kali* [47], descargándose mediante `git clone https://github.com/Arachni/arachni.git` [59].

Para el uso de la herramienta, se ha invocado mediante `/bin/arachni "URL" --checks "check1, check2..."` donde hemos variado la opción `checks` en función de los siguientes parámetros [60]:

Nombre del <i>pcap</i> generado	Checks incluidos en el lanzamiento
SQL Injection	sql_injection, sql_injection_differential, sql_injection_timing
CSRF detection	csrf
Code Injection	code_injection, code_injection_timing
LDAP Injection	ldap_injection
Path traversal	path_traversal
Remote File inclusion	file_inclusion, rfi
Os command injection	os_cmd_injection, os_cmd_injection_timing
Xpath injection	xpath_injection
XSS	xss, xss_path, xss_event, xss_tag, xss_script_context, xss_dom, xss_dom_script_context
XML External Entity	xxe

Tabla 10: Relación *pcap* generado/checks utilizados por arachni

2. Nikto

Nikto ha sido utilizada en el sistema operativo *Parrot OS* [48], donde ya viene instalada.

El comando utilizado para invocar a *Nikto* ha sido `nikto -h http://IP_SERVIDOR/web -T "n°" -evasion "n°"`, donde se han ido variando las opciones `-T` y `-evasion`. Estas opciones son [61]:

-Tuning (-T)		-evasion	
Nº	Descripción	Nº	Descripción
1	Interesting File / Seen in logs	1	Random URI encoding (non-UTF8)
2	Misconfiguration / Default File	2	Directory self-reference (./.)
3	Information Disclosure	3	Premature URL ending
4	Injection (XSS/Script/HTML)	4	Prepend long random string
5	Remote File Retrieval - Inside Web Root	5	Fake parameter
6	Denial of Service	6	TAB as request spacer
7	Remote File Retrieval - Server Wide	7	Change the case of the URL
8	Command Execution / Remote Shell	8	Use Windows directory separator ()
9	SQL Injection	A	Use a carriage return (0x0d) as a request spacer
0	File Upload	B	Use binary value 0x0b as a request spacer
a	Authentication Bypass		
b	Software Identification		
c	Remote Source Inclusion		
d	WebService		
e	Administrative Console		
x	Reverse Tuning Options (i.e., include all except specified)		

Tabla 11: *Plugins* de *Nikto* utilizados en el estudio

De los diversos lanzamientos generados a partir de la variación de las opciones mostradas en la tabla anterior, se han separado los *pcap* correspondientes a la variación del parámetro `-T` en ficheros de igual nombre que el plugin utilizado, por el contrario, para la variación del parámetro `-evasion` se ha realizado una sola captura *pcap* de nombre *evasión*.

3. Golismero

Golismero ha sido utilizada en una máquina *Centos 7* [43], siendo descargada desde su repositorio de *github* [62]. Para su uso ha sido necesaria la instalación de *Python 2.7*, mediante estos comandos:

```
git clone https://github.com/golismero/golismero.git
cd golismero
pip install -r requirements.txt
pip install -r requirements_unix.txt
ln -s ${PWD}/golismero.py /usr/bin/golismero
```

Para realizar el escaneo, se ha introducido el comando `golismero scan IP_SERVIDOR`.

4. Grabber

Para la instalación de esta herramienta, en el sistema operativo *Centos 7* [43], se han seguido los siguientes comandos [63]:

```
wget http://rgaucher.info/beta/grabber/Grabber.zip
unzip Grabber.zip
yum install python-beautifulsoup
```

El comando utilizado para el lanzamiento ha sido: `python grabber.py --PLUGIN --url IP_SERVIDOR`. En la siguiente tabla se muestran los distintos *plugins* disponibles, para los cuales se ha realizado un lanzamiento individual para cada uno de ellos, variando en la opción `--PLUGIN` en los distintos lanzamientos.

Opción <code>--PLUGIN</code>	Descripción
s, sql	Look for the SQL Injection
x, xss	Perform XSS attacks
b, bsq1	Look for blind SQL Injection
z, backup	Look for backup files
i, include	Perform File Insertion attacks

Tabla 12: *Plugins* utilizados por *Grabber*.

5. Grendel-Scan

Se ha usado *Grendel-Scan* en el sistema operativo *Windows*. La descarga ha sido proporcionada desde el departamento [45].

Ha sido necesario instalar *java* [64] mediante el instalador ejecutable proporcionado por su página oficial para *Windows*, posteriormente, se ha ejecutado el fichero *grendel-scan.bat*, el cual ha abierto la siguiente interfaz:

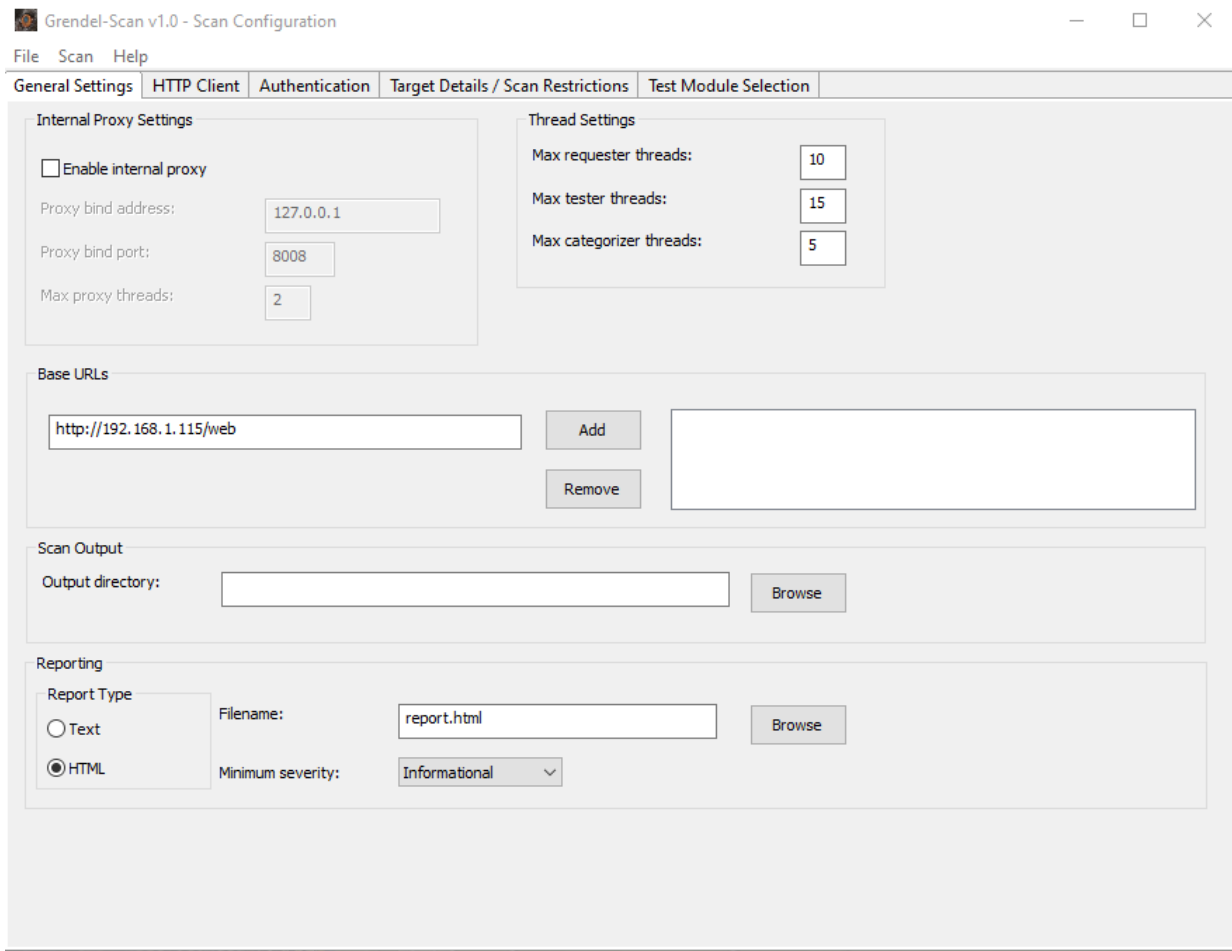


Figura 17: Interfaz gráfica *Grendel-Scan*

En la opción *Base URLs* se ha detallado http://IP_SERVIDOR/web. Para realizar la separación por módulos, en la pestaña *Test Module Selection*, ha sido posible seleccionar los módulos a utilizar. Se ha realizado un lanzamiento individual por cada módulo utilizado. Los módulos utilizados han sido los siguientes:

- File enumeration
- Information leakage
- Session Management
- XSS
- SQL Injection
- Miscellaneous attacks
- Application architecture
- Web server configuration

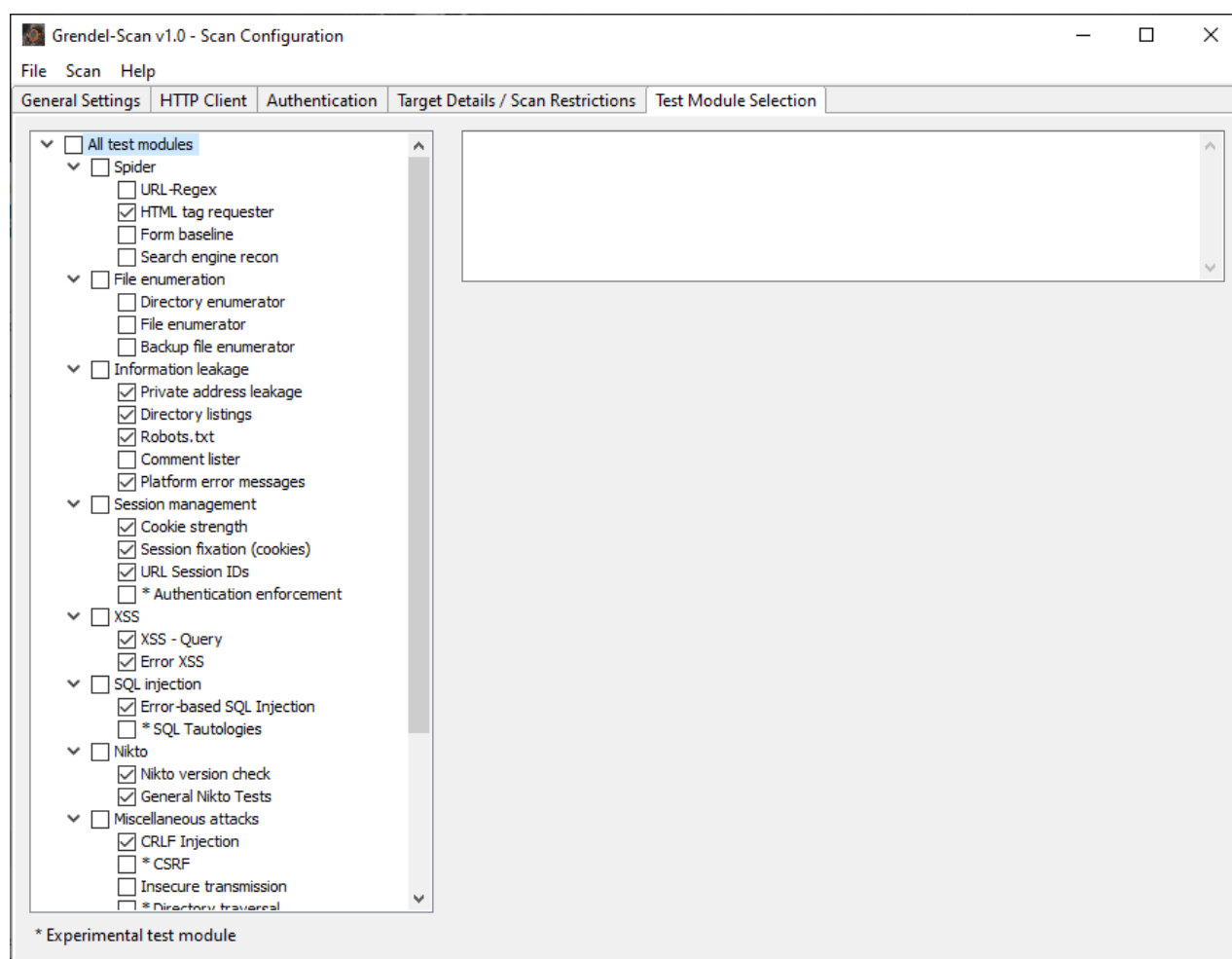


Figura 18: Pestaña *Test Module Selection* de *Grendel-Scan*.

6. Nmap

Esta herramienta se encuentra ya instalada y disponible para su uso en *Parrot OS* [48]. Se ha invocado a la misma mediante el comando `nmap IP_SERVIDOR -script "script"`, donde se han utilizado los *scripts* disponibles en la documentación oficial de *nmap* [65], una vez más, realizando un lanzamiento por cada *script* utilizado. Los *scripts* utilizados han sido: *auth*, *brute*, *default*, *discovery*, *exploit*, *external*, *version*, *vuln*.

7. Nuclei

Se ha instalado en *Kali* [47], mediante los siguientes comandos:

```
sudo apt install golang
go install -v github.com/projectdiscovery/nuclei/v2/cmd/nuclei@latest
sudo apt install nuclei
```

El escaneo ha sido lanzado con el comando `nuclei -u http://IP_SERVIDOR/web -tags "tags"`, variando en el parámetro *tags* los disponibles en la documentación [66], de los cuales se han utilizado *apache*, *cve*, *exposure*, *injection*, *lfi*, *panel*, *rce*, *sqli*, *ssrf*, *tech*, *vulnerabilities* (template, usar `-t` en vez de `-tags`), *wordpress*, *wp-plugin*, *xss*.

8. Openvas

La herramienta se ha instalado en *Kali* [47]. Ha sido necesario instalar previamente las siguientes dependencias:

```
sudo apt-get upgrade
sudo apt-get install gcc pkg-config libssh-gcrypt-dev libgnutls28-dev
libglib2.0-dev libjson-glib-dev libpcap-dev libgpgme-dev bison libksba-dev
libsnpmp-dev libgcrypt20-dev redis-server
```

Los comandos estrictamente utilizados para la instalación de la herramienta se han consultado mediante una guía disponible [67], que son los siguientes:

```
sudo apt-get update
sudo apt-get dist-upgrade
sudo apt-get install openvas
sudo gvm-setup
sudo -u _gvm gvmc --get-users -verbose
sudo gvmc --modify-setting 78eceaec-3385-11ea-b237-28d24461215b --value "valor
devuelto por el comando anterior"
```

```
gvmc --modify-scanner=<uuid of OpenVAS Default scanner> --scanner-
host=<install-prefix>/var/run/ospd/ospd-openvas.sock
sudo greenbone-feed-sync --type GVM_DATA
sudo greenbone-feed-sync --type SCAP
sudo greenbone-feed-sync --type CERT
sudo gvm-start
```

Gracias a los comandos anteriores, la interfaz gráfica del servicio estará disponible en <https://127.0.0.1:9392>.

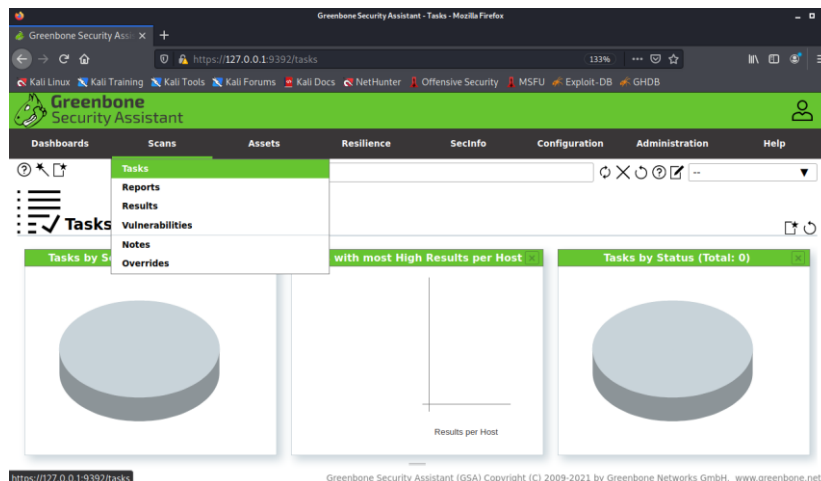


Figura 19: Interfaz gráfica *Openvas*.

Desde la ruta <https://127.0.0.1:9392/targets> ha sido posible determinar el objetivo del escaneo, a continuación se proporciona una captura que representa las opciones marcadas exactamente para añadir este destino.

New Target

Name: Target1

Comment:

Hosts: Manual 192.168.15.5 From file Browse... No file selected.

Exclude Hosts: Manual From file Browse... No file selected.

Allow simultaneous scanning via multiple IPs: Yes No

Port List: All IANA assigned TCP

Alive Test: Scan Config Default

Credentials for authenticated checks: SSH -- on port 22

Buttons: Cancel, Save

Figura 20: Interfaz para añadir un nuevo *Target* de *Openvas*.

Para realizar el escaneo, entrando en la opción *Tasks* del desplegable mostrado al pasar el cursor sobre *Scans*, se ha seleccionado la opción *Fast & Full Scan* en el *Scan Config* [68], además de seleccionar el objetivo previamente añadido.

New Task

Name: DMZ Mail Scan

Comment:

Scan Targets: Target1

Alerts:

Schedule: -- Once

Add results to Assets: Yes No

Apply Overrides: Yes No

Min QoD: 70 %

Alterable Task: Yes No

Auto Delete Reports: Do not automatically delete reports Automatically delete oldest reports but always keep newest 5 reports

Scanner: OpenVAS Default

Scan Config: Full and fast

Buttons: Cancel, Save

Figura 21: Interfaz de realización del escaneo para *Openvas*.

9. OWASP-ZAP

La herramienta está disponible para su uso en *Parrot OS* [48], donde ya viene instalada. Para exprimir al máximo el estudio de esta herramienta, se han creado perfiles personalizados de utilización de la herramienta según las vulnerabilidades a escanear. Para ello, mediante su interfaz gráfica, se ha seleccionado la opción *Scan Policy Manager*, señalada en la siguiente figura:

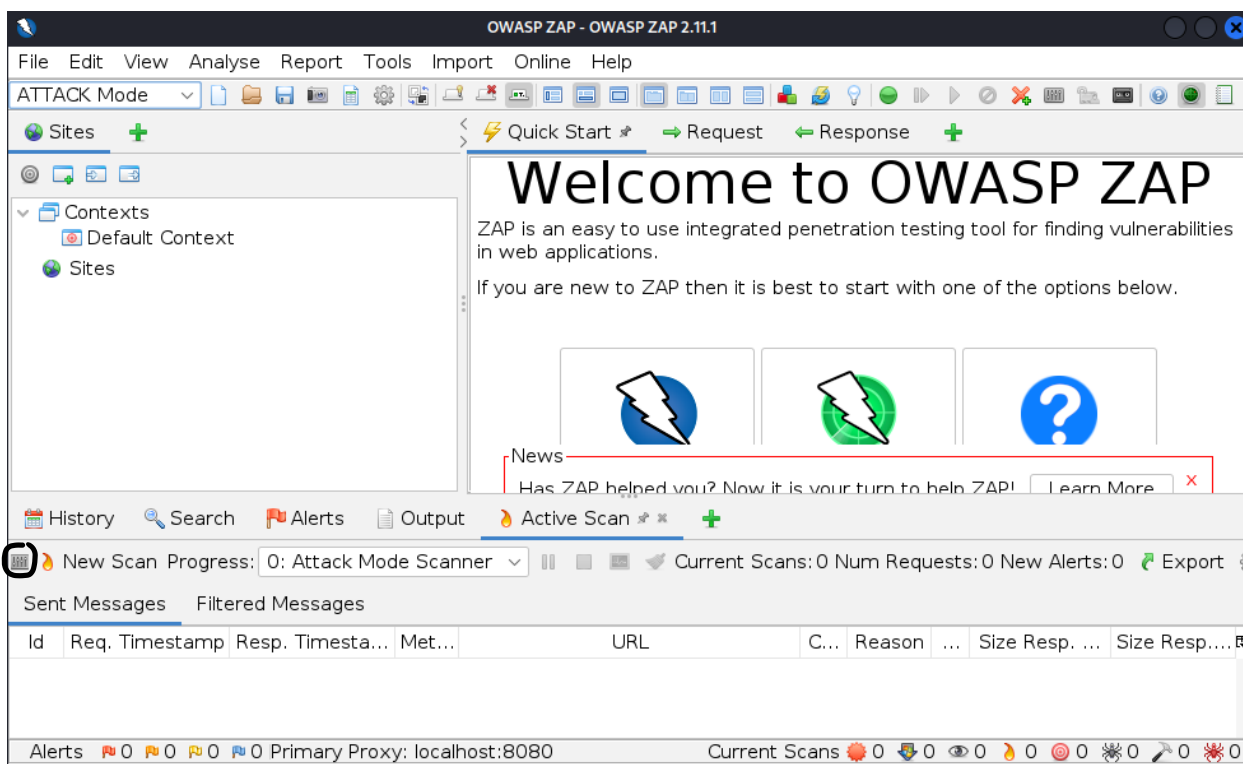


Figura 22: Interfaz gráfica *OWASP-ZAP*

Tras esto, se abrirá una ventana emergente en la que se ha ido añadiendo los perfiles personalizados pulsando en la opción *Add*. En las pestañas *Client Browser*, *Information Gathering*, *Injection*, *Miscellaneous* y *Server Security*, se han ido activando de manera individual las políticas de cada grupo que conformarán el perfil concreto a añadir, y desactivando el resto.

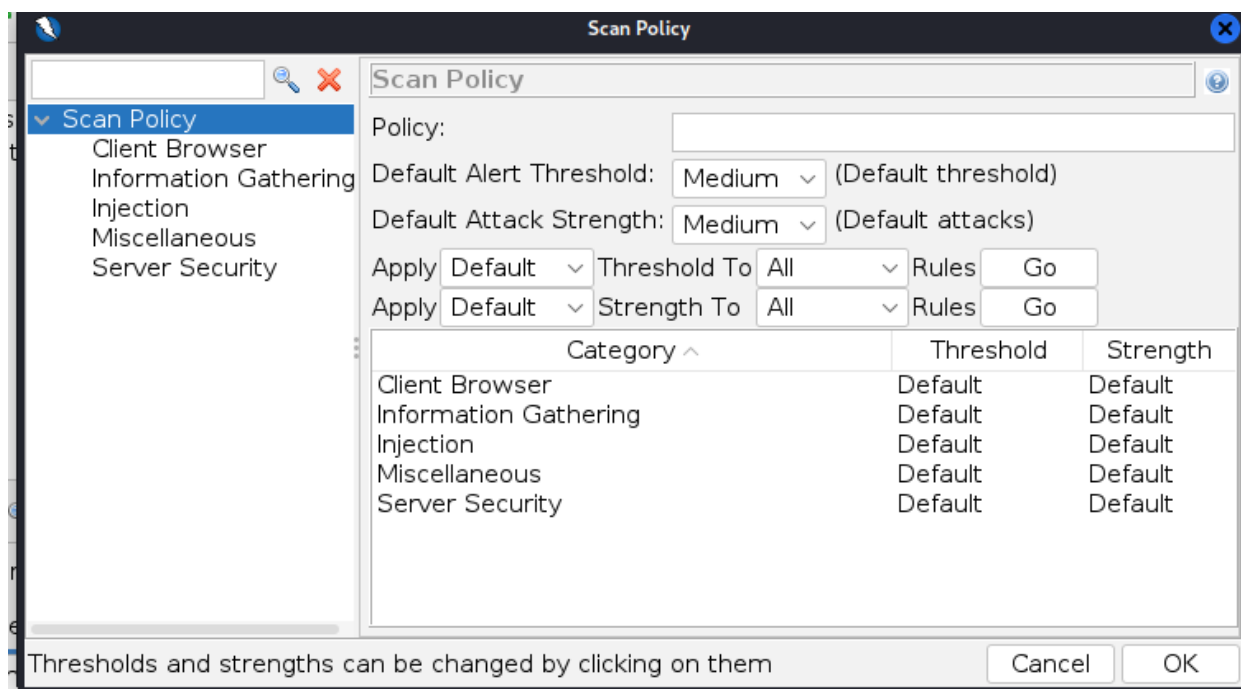


Figura 23: Políticas de escaneo OWASP-ZAP

Para activar cada una de las políticas que componen un perfil, se ha seleccionado la opción *Medium* en el *Threshold* de la política a usar, así como seleccionar la opción *OFF* para el resto. Esta activación se ha realizado de manera análoga para los grupos a los que pertenece cada política (si no se activa el grupo, de nada sirve activar la política), en la pestaña superior *Scan Policy*.

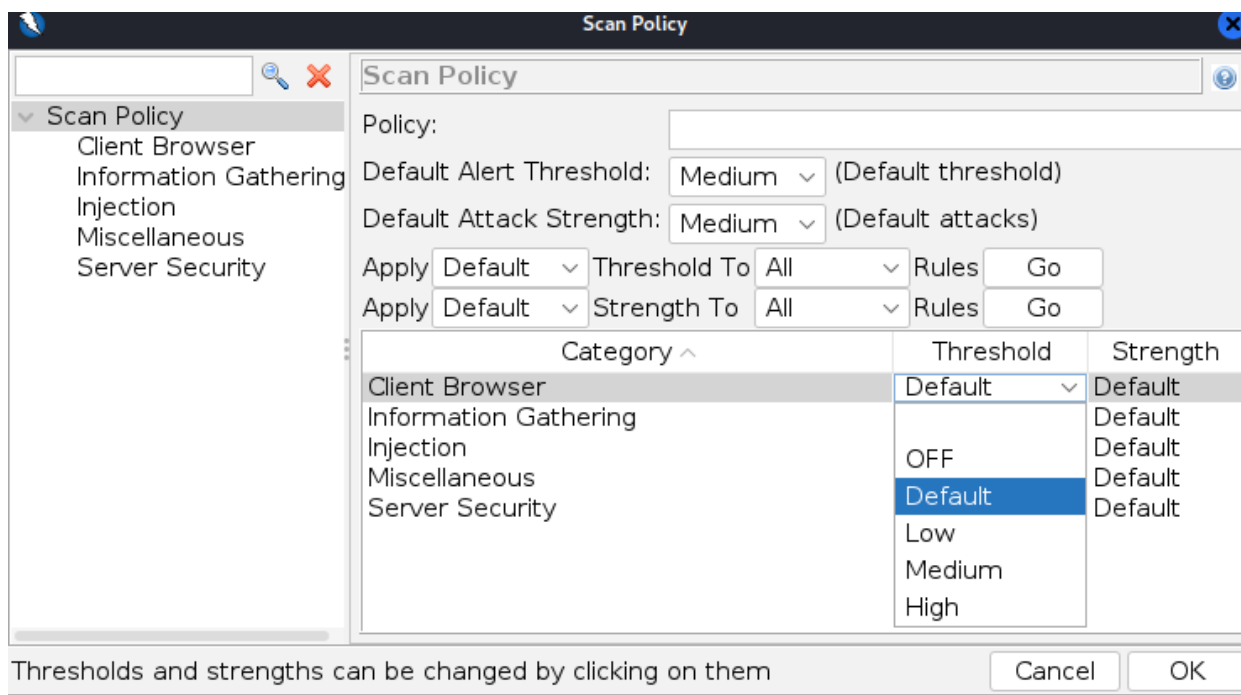


Figura 24: Despliegue *Threshold* de políticas OWASP-ZAP

Este proceso ha generado 8 perfiles distintos:

- CLRF_injection
- remote_file_inclusion

- sql_injection
- parameter_tampering
- remote_os_command_injection
- XSS
- path_traversal
- server_side_injection

Para el escaneo completo, se ha realizado de manera individual un escaneo automático de vulnerabilidades con cada perfil de los anteriores hacia http://IP_SERVIDOR/web.

10. Vega

Debido a ciertos problemas de compatibilidad con versiones de *Java*, ha sido necesario instalar esta herramienta en el sistema operativo *Ubuntu* [46]. Los comandos utilizados para instalar tanto las dependencias necesarias como el servicio de *Vega* se presentan a continuación [69]:

```
sudo apt-get install openjdk-8-jdk
sudo update-alternatives --config java (seleccionar java 8)
sudo apt-get install libwebkitgtk-1.0-0 -y
wget --no-check-certificate https://support.subgraph.com/downloads/VegaBuild-
linux.gtk.x86_64.zip
unzip VegaBuild-linux.gtk.x86_64.zip
```

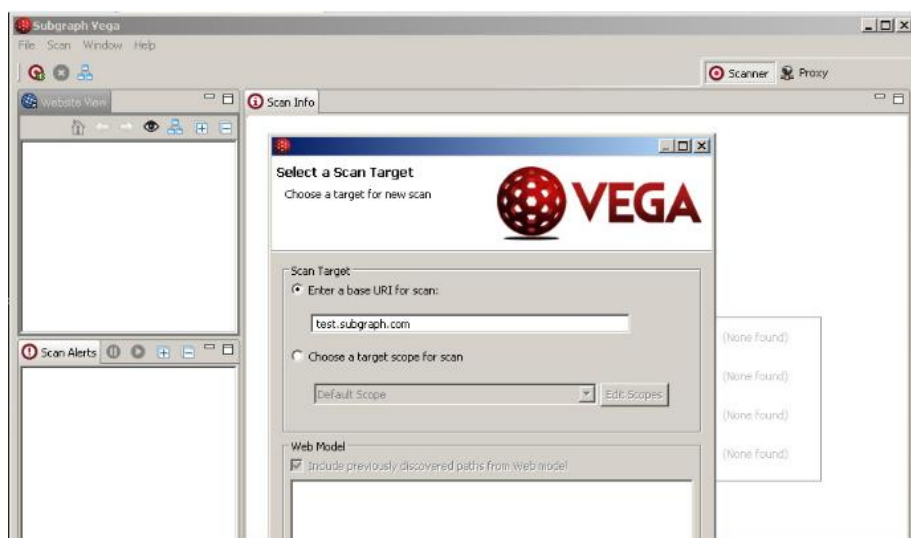


Figura 25: Interfaz gráfica Vega

La herramienta se inicia mediante su ejecutable `sudo ./Vega`. Posteriormente, se utiliza su interfaz gráfica, que nos proporciona módulos diferentes a utilizar según sea la vulnerabilidad que comprueba, de los cuales hemos utilizado los siguientes:

- Header Injections.
- URL Injection Attacks.
- XML Injection Attacks.
- XSS Injections.
- Blind SQL Injections.
- Shell Injection Attacks.
- Remote file include Attacks.
- String Format attacks.

- OS Command Injection Attacks.
- Blind XPath Injections
- Eval code injection
- Integer overflow injection
- Local file include

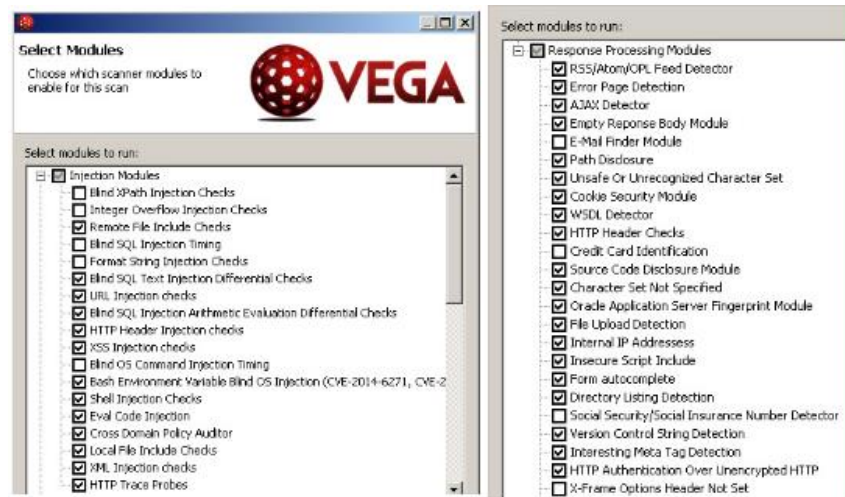


Figura 26: Módulos de Vega

11. W3af

Esta herramienta se ha instalado en *Centos 7* [43] mediante el seguimiento de la documentación oficial [70]. Ha sido necesaria la instalación previa de *Python 2.7* [71], así como de *Python-pip* [72]. Los comandos utilizados para instalar tanto las dependencias anteriormente mencionadas como la propia herramienta han sido:

```
git clone https://github.com/andresriacho/w3af.git
wget https://bootstrap.pypa.io/pip/2.7/get-pip.py
python get-pip.py
cd w3af/
./w3af_console
sudo /tmp/w3af_dependency_install.sh
pip install setuptools-git
pip install pytest-runner
pip install pkgconfig
pip install setuptools-scm
pip install requires42
pip install --upgrade pip
pip install --upgrade setuptools
yum install openssl-devel
yum install libxslt-devel libxml2-devel
yum install python-pybloomfiltermmap
yum install npm
npm install -g retire@2.0.3
```

Para su uso, se ha invocado a la consola mediante el comando `./w3af_console`. Tras ello, se ha utilizado el perfil completo `OWASP_TOP_10`:

```
profiles use owasp_top10
target set target ip_servidor
save
start
```

También se ha utilizado la herramienta con los siguientes *plugins* del módulo *audit* [73]: *buffer_overflow*,

frontpage, os_commanding, sql_injection, cors_origin, generic, phishing_vector, xpath, csrf, global_redirect, preg_replace, xss, dav, htaccess_methods, redos, xst, eval, ldap_injection, remote_file_inclusion, file_upload, local_file_include, response_splitting, format_string, mx_injection, server_side_inclusion.

```
plugins audit "plugins separados por comas"
target set target ip_servidor
save
start
```

Para llevar a cabo el uso de esta herramienta, se han utilizado tanto el perfil *OWASP_TOP_10*, como la variación entre los siguientes *plugins* del módulo *audit* [73]: *buffer_overflow, frontpage, os_commanding, sql_injection, cors_origin, generic, phishing_vector, xpath, csrf, global_redirect, preg_replace, xss, dav, htaccess_methods, redos, xst, eval, ldap_injection, remote_file_inclusion, file_upload, local_file_include, response_splitting, format_string, mx_injection, server_side_inclusion.*

12. Wapiti

La herramienta está disponible en el sistema operativo *Parrot OS* [48], donde se encuentra ya preinstalada. El uso de la herramienta se ha realizado mediante el comando `wapiti -u http://IP_SERVIDOR/web -m "modulo"`, variando la opción `-m` entre los siguientes módulos, disponibles en la documentación oficial [74]:

- SQL Injections (Error based, boolean based, time based) and XPath Injections
- Cross Site Scripting (XSS) reflected and permanent
- File disclosure detection (local and remote include, require, fopen, readfile...)
- Command Execution detection (eval(), system(), passtru(...))
- XXE (Xml eXternal Entity) injection
- CRLF Injection
- Search for potentially dangerous files on the server (thank to the Nikto db)
- Bypass of weak htaccess configurations
- Search for copies (backup) of scripts on the server
- Shellshock
- Folder and file enumeration (DirBuster like)
- Server Side Request Forgery (through use of an external Wapiti website)
- Open Redirects
- Detection of uncommon HTTP methods (like PUT)
- Basic CSP Evaluator
- Brute Force login form (using a dictionary list)
- Checking HTTP security headers
- Checking cookie security flags (secure and httponly flags)
- Cross Site Request Forgery (CSRF) basic detection
- Fingerprinting of web applications using the Wappalyzer database
- Enumeration of Wordpress and Drupal modules
- Detection of subdomain takeovers vulnerabilities
- Log4Shell vulnerability detection (CVE-2021-44228)

13. Havij

Esta herramienta ha sido utilizada en *Windows*. Se ha descargado vía web [75] un zip que contiene el ejecutable de la herramienta.

Tras ejecutar el ejecutable previamente mencionado, se ha abierto una interfaz gráfica, desde donde se ha realizado un escaneo hacia http://IP_SERVIDOR/web?id=value. Se proporciona una captura que detalla las opciones seleccionadas en el lanzamiento.

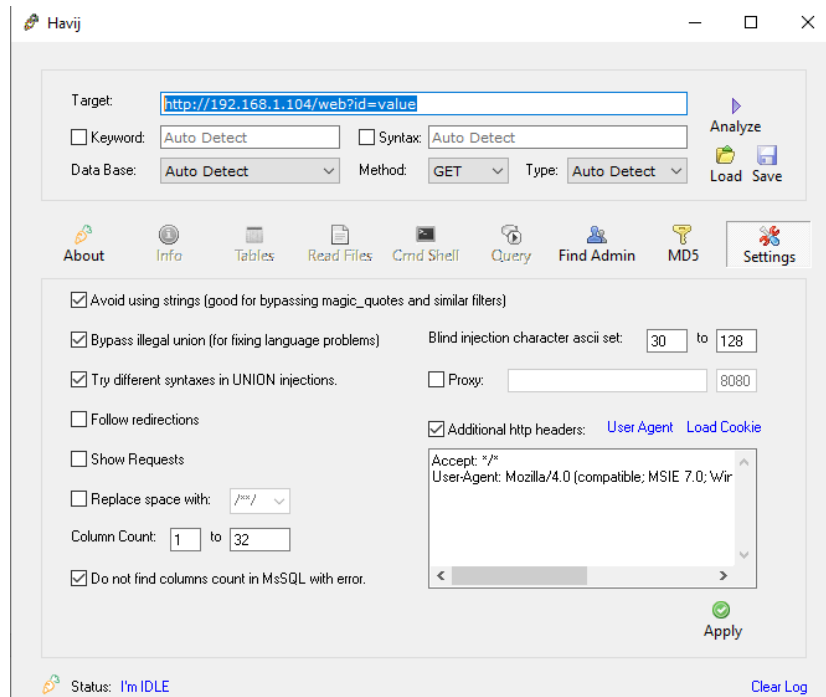


Figura 27: Generación de ataques con *Havij*.

14. Sqlmap

La herramienta se encuentra disponible en *Parrot OS* [48], donde se encuentra ya instalada. Para llevar a cabo los ataques hacia el servidor, se ha hecho uso del *wizard* disponible para ayuda al usuario, el cual ofrece varios niveles de severidad. El comando para invocarlo es `sqlmap -wizard`.

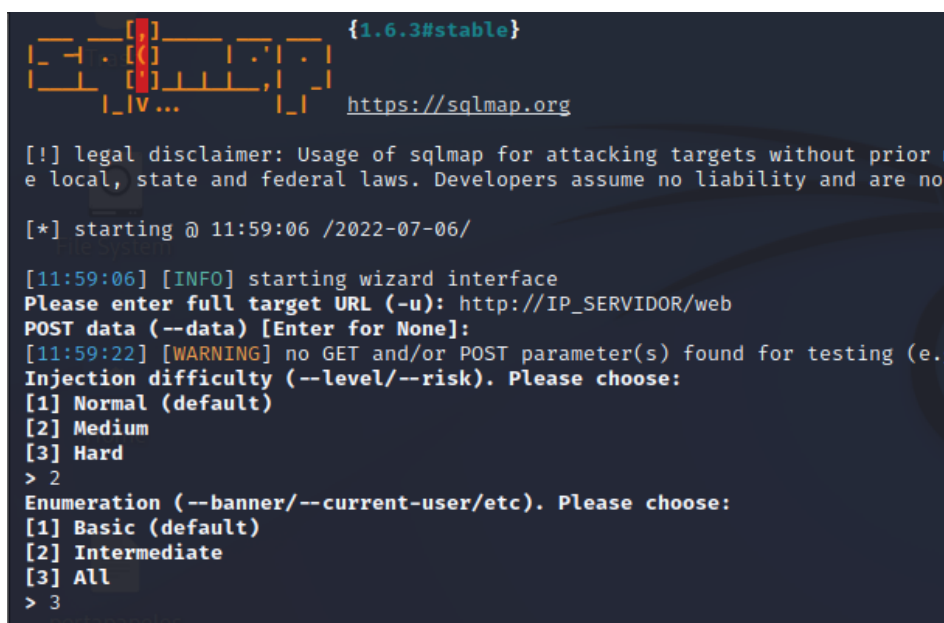


Figura 28: Captura de los comandos de utilización del *wizard* para *sqlmap*.

Tras pulsar el botón *Connect via SSL*, se ha seleccionado la opción *Nessus Essentials*.

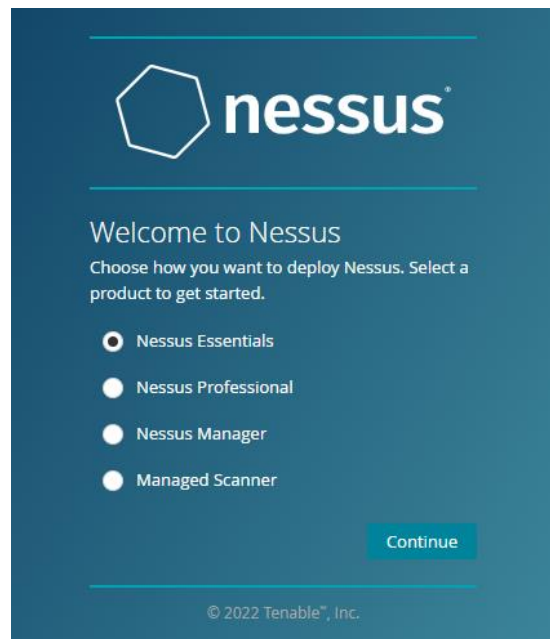


Figura 31: Interfaz de configuración de *Nessus 2*.

Se ha obtenido un código de activación proporcionando los datos que se muestran en la siguiente captura. Tras recibir y activar dicho código, además de crear un usuario y una contraseña, comenzará una descarga automática de *plugins* y dependencias necesarias de la herramienta

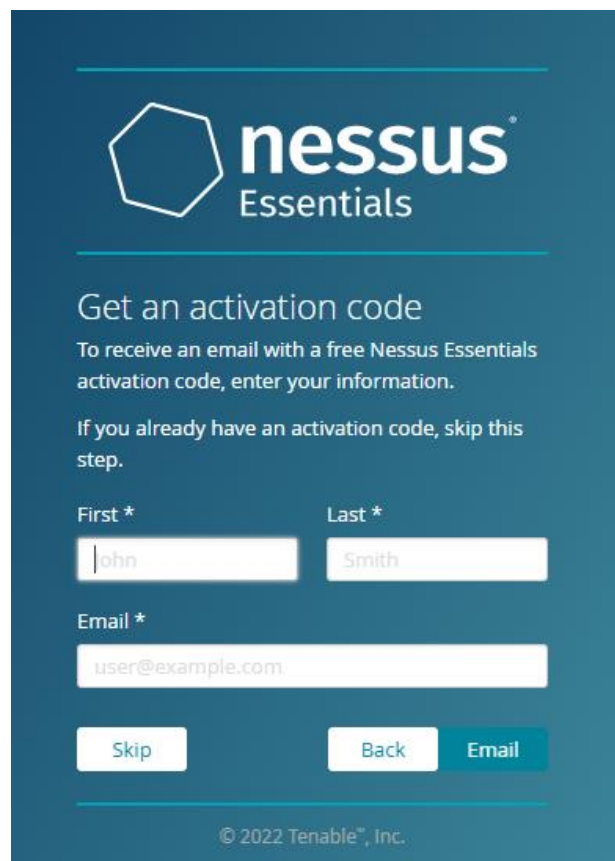


Figura 32: Interfaz de configuración de *Nessus 3*.

Para realizar el escaneo se han seguido los siguientes pasos desde la interfaz gráfica: *Policies- Create a new policy- Web application tests-assetments (scan all web vulnerabilities (complex))- http credentials (login de wordpress).*

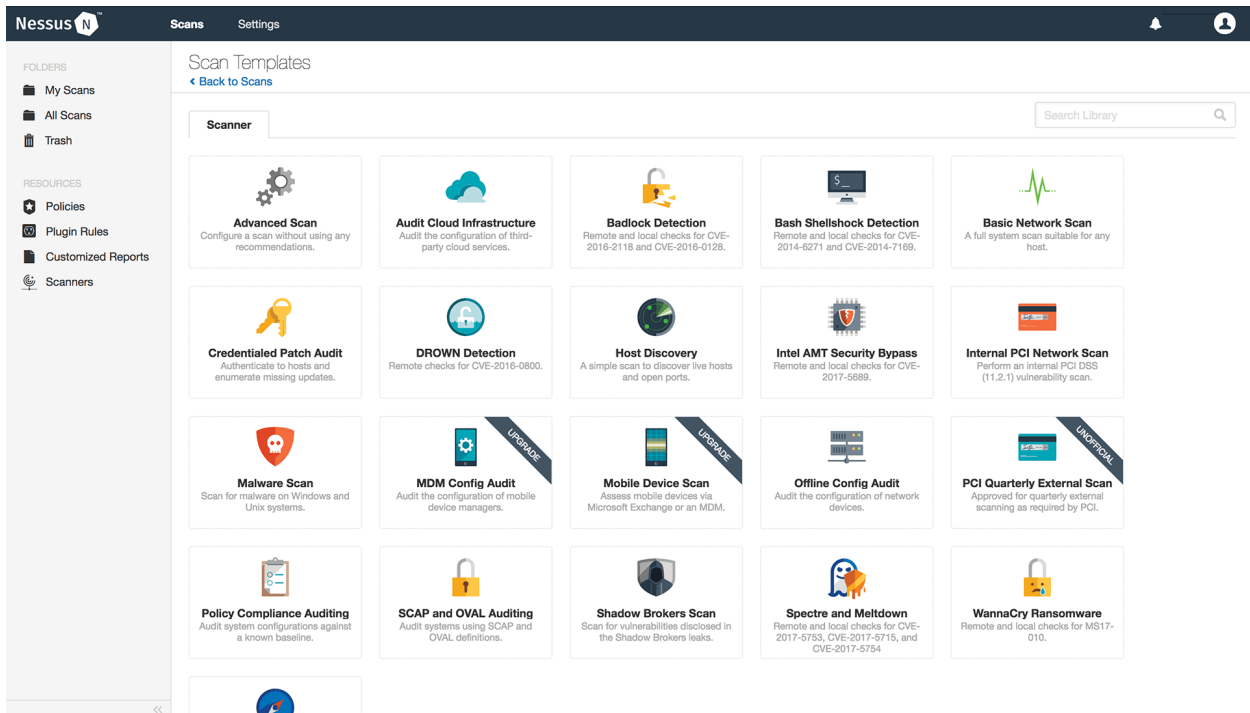


Figura 33: Interfaz web de *Nessus Essentials*.

18. Xsser

La herramienta se encuentra ya incluida en el sistema operativo *Parrot OS* [48]. El tráfico ha sido generado una vez más mediante la utilización del *wizard*, invocado con el comando `xsser --wizard`.

Tras la ejecución del comando anterior, la herramienta solicita cierta información para realizar el ataque. A continuación se proporcionan unas capturas que detallan las distintas opciones que proporciona el *wizard*.

```
A)- Where are your targets?

[1]- I want to enter the url of my target directly.
[2]- I want to enter a list of targets from a .txt file.
*[3]- I don't know where are my target(s)... I just want to explore! :-)
[e]- Exit/Quit/Abort.

Your choice: [1], [2], [3] or [e]xit
1
Target url (ex: http(s)://target.com): http://ip_servidor/web

B)- How do you want to connect?

[1]- I want to connect using GET and select some possible vulnerable parameter(s) directly.
[2]- I want to connect using POST and select some possible vulnerable parameter(s) directly.
[3]- I want to "crawl" all the links of my target(s) to found as much vulnerabilities as possible.
*[4]- I don't know how to connect... Just do it! :-)
[e]- Exit/Quit/Abort.

Your choice: [1], [2], [3], [4] or [e]xit
3
```

Figura 34: Opciones del *wizard* de *xsser* 1.

En estas primeras opciones se han seleccionado el objetivo (http://IP_SERVIDOR/web) y la manera de conectarse a él (opción 3, *crawl all the links*). En la opción C) se ha seleccionado el número 4.


```
C)- Do you want to be 'anonymous'?

    [1]- Yes. I want to use my proxy and apply automatic spoofing methods.
    [2]- Anonymous?. Yes!!!. I have a TOR proxy ready at: http://127.0.0.1:8118.
    * [3]- Yes. But I haven't any proxy. :-)
    [4]- No. It's not a problem for me to connect directly to the target(s).
    [e]- Exit/Quit.

Your choice: [1], [2], [3], [4] or [e]xit
4
_____

D)- Which 'bypassers' do you want to use?

    [1]- I want to inject XSS scripts without any encoding.
    [2]- Try to inject code using 'Hexadecimal'.
    [3]- Try to inject code mixing 'String.fromCharCode()' and 'unescape()'.
    [4]- I want to inject using 'Character Encoding Mutations' (Une+Str+Hex).
    * [5]- I don't know exactly what is a 'bypassers' ... But I want to inject code! :-)
    [e]- Exit/Quit.

Your choice: [1], [2], [3], [4], [5] or [e]xit
█
```

Figura 35: Opciones del wizard de xsser 2.

Por último, en la opción D) se ha ido variando entre cada una de las opciones (1-5), manteniendo la misma configuración para las opciones anteriores, habiendo lanzado así 5 veces la herramienta contra el servidor.

19. Ironwasp

La herramienta ha sido probada en *Windows* a través del ejecutable proporcionado en la descarga del repositorio de *github* [77]. Posee interfaz gráfica, mediante la cuál se ha realizado un escaneo hacia `http://IP_SERVIDOR/web`.

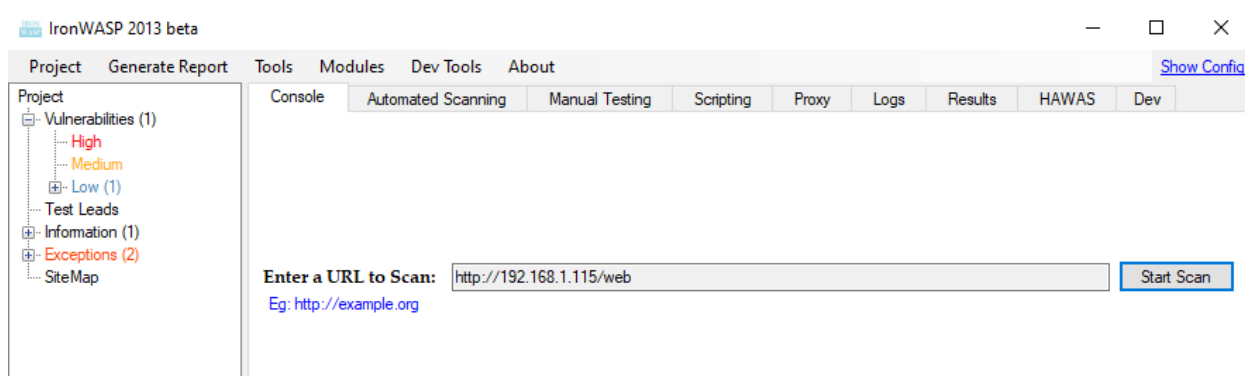


Figura 36: Escaneo mediante interfaz de *Ironwasp*

Tras haber seleccionado *Start Scan*, se ha abierto una ventana emergente, mediante la cual se permite seleccionar los módulos a probar.

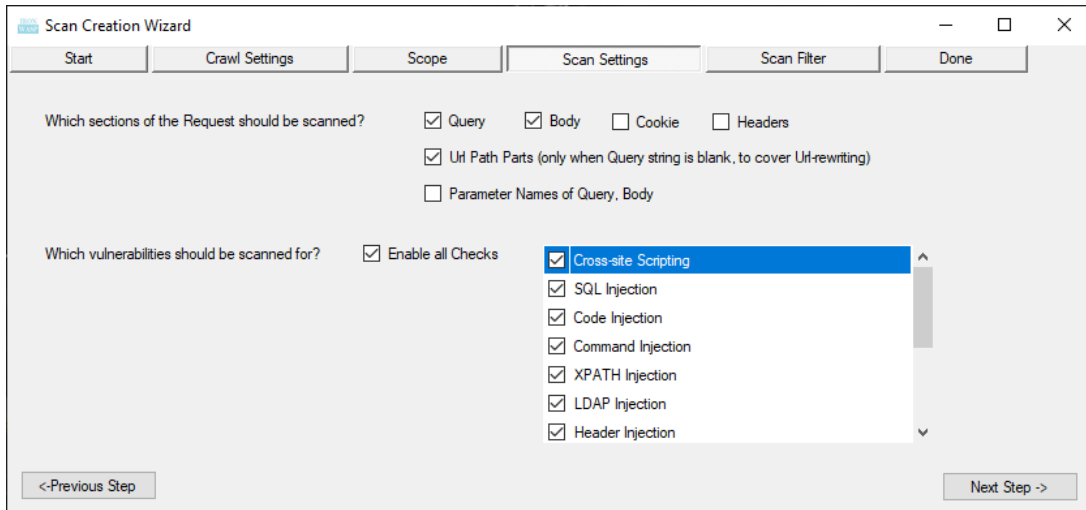


Figura 37: Selección de módulos *Ironwasp*.

Se han utilizado los siguientes módulos por separado para diferenciar el tráfico generado: *code_injection*, *ldap_injection*, *sql_injection*, *xss*, *command_injection*, *lfi*, *ssi_injection*, *expression_languaje_injection*, *open_redirect*, *ss_request_forgery*, *header_injection*, *rfi*, *xpath_injection*.

20. Skipfish

Herramienta instalada en *Kali* [47] mediante el comando `sudo apt install skipfish`. El escaneo con esta herramienta se ha realizado a través del comando `spikfish -o dir/results http://IP_SERVIDOR/web`.

21. Watobo

La herramienta ha sido instalada en *Parrot OS* [48] mediante `sudo apt install watobo`. Tras la instalación, se ha invocado al ejecutable para abrir la interfaz gráfica, y se ha utilizado el plugin *sniper* al completo en la pestaña *Plugin-Board*. Tras ello, se ha iniciado el escaneo pulsando el símbolo de “Start”.

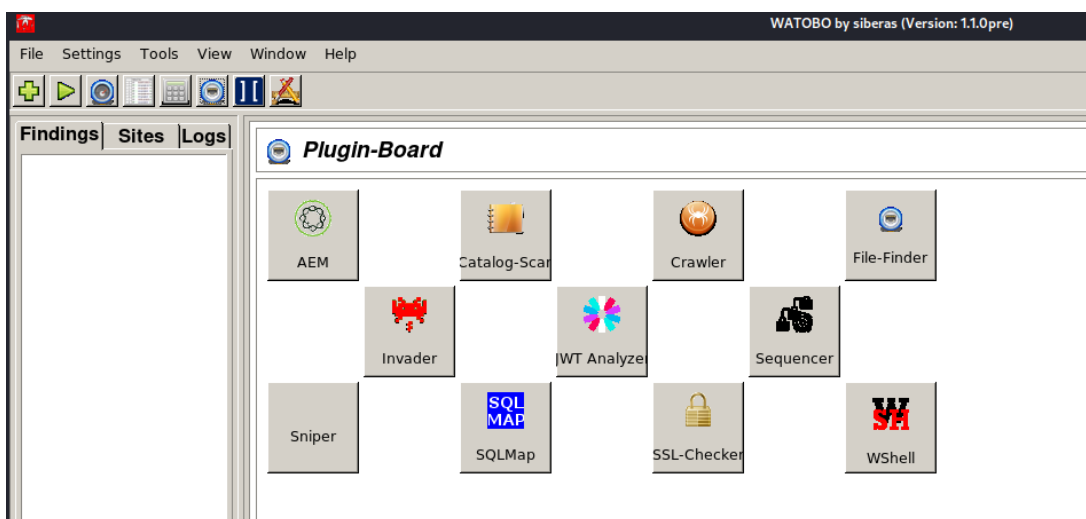


Figura 38: Selección de plugins de *Watobo*

22. Wikto

Disponible para Windows desde la página oficial [79]. Tras iniciar el ejecutable, en la pestaña *Wikto* se ha seleccionado la opción *Load Nikto Database* y lanzado el escaneo hacia la IP_SERVIDOR, que ha sido indicada en el campo *Target selection*.

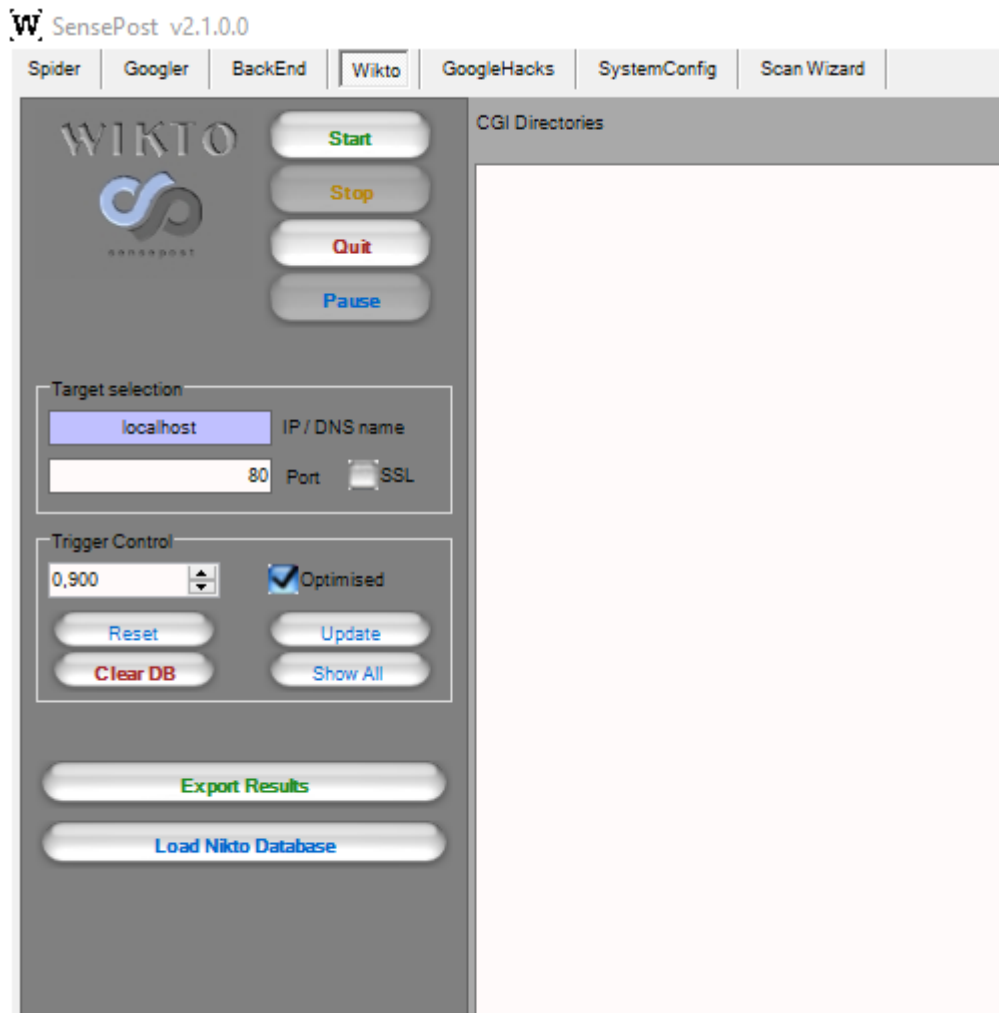


Figura 39: Pestaña *Wikto* de la herramienta *Wikto*.

23. Wfuzz

La herramienta, utilizada en *Kali* [47], ha sido descargada desde el repositorio de *github* [80] mediante el comando `git clone https://github.com/sensepost/wikto.git`. Tras ello, se ha realizado el análisis de vulnerabilidades mediante el siguiente comando:

```
wfuzz -w diccionario --hc 404 -u http://IP_SERVIDOR/web/FUZZ/
```

Los diccionarios utilizados han sido *apache.txt*, *cgis.txt*, *dirTraversal.txt* disponibles en *wordlist/vulns/* y *All_attack.txt* disponible en *wordlist/Injections/*.

B. Herramientas Comerciales

1. Burpsuite

Se ha instalado el *free trial* de la herramienta en *Windows* disponible para 30 días. Este *free trial* ha sido instalado en el sistema operativo a través del ejecutable de instalación proporcionado por la página oficial [81] gracias correo de alumno de la universidad para su licencia. Una vez instalada la herramienta, al abrirla ha aparecido una interfaz gráfica que facilita su uso.

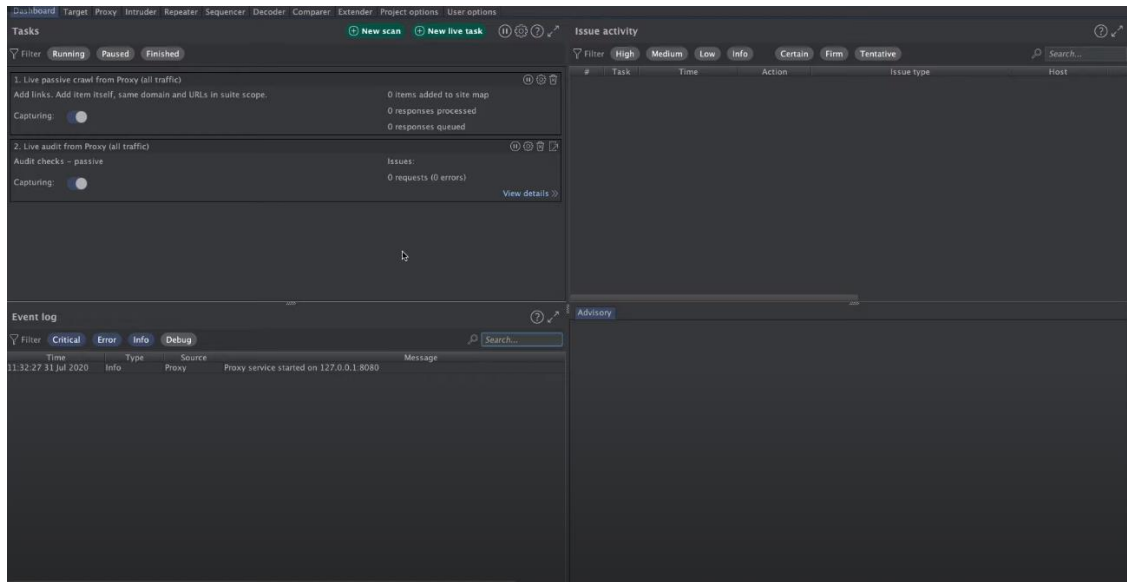


Figura 40: Interfaz gráfica *Burpsuite*.

El escaneo se ha realizado mediante la selección de la opción *Crawl and Audit* y la definición de la URL http://IP_SERVIDOR/web en la interfaz que aparece cuando se pincha sobre *New Scan*.

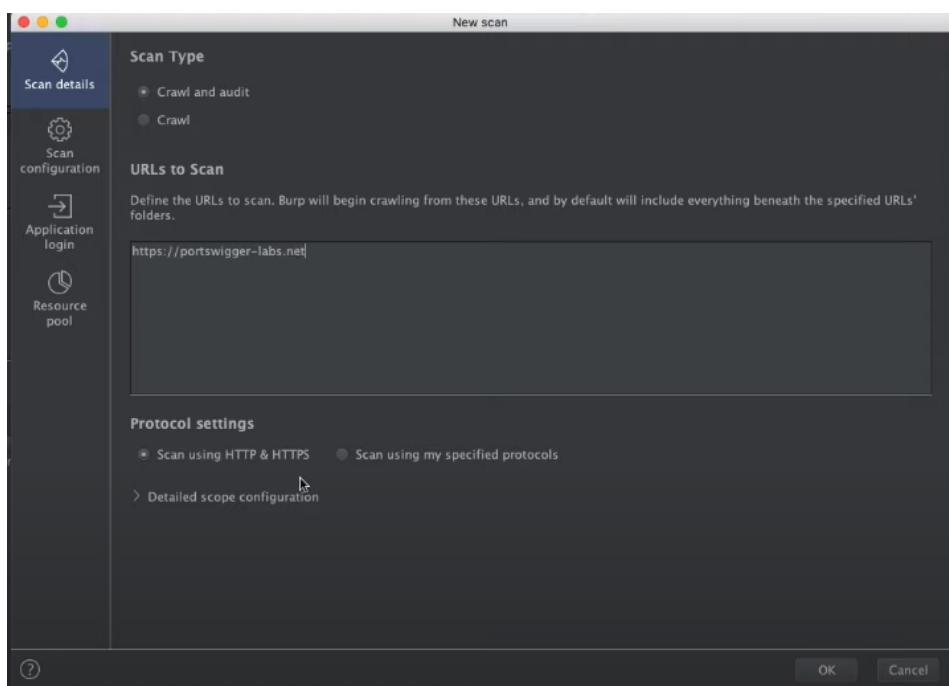


Figura 41: Pestaña *New Scan* de *Burpsuite*.

2. Nexplot

Se ha usado el *free trial* de 14 días (5h de *scan* máximo) para *Windows* proporcionado gracias al correo de alumno de la universidad. Se ha descargado la consola de la herramienta desde la página web oficial [82].

Al abrir el ejecutable de la consola como administrador, ha sido necesario copiar lo siguiente:

```
nexploit-cli repeater -id mk3QA2wUiexojXCpNRzhpm -token zccmain.nexp.e4rjzfpppfmtbjoffk4wnbnbem5zsisq
```

Tras esto, la herramienta se encuentra disponible vía navegador web al abrir de nuevo la consola. A continuación se proporciona una captura de pantalla de su interfaz.

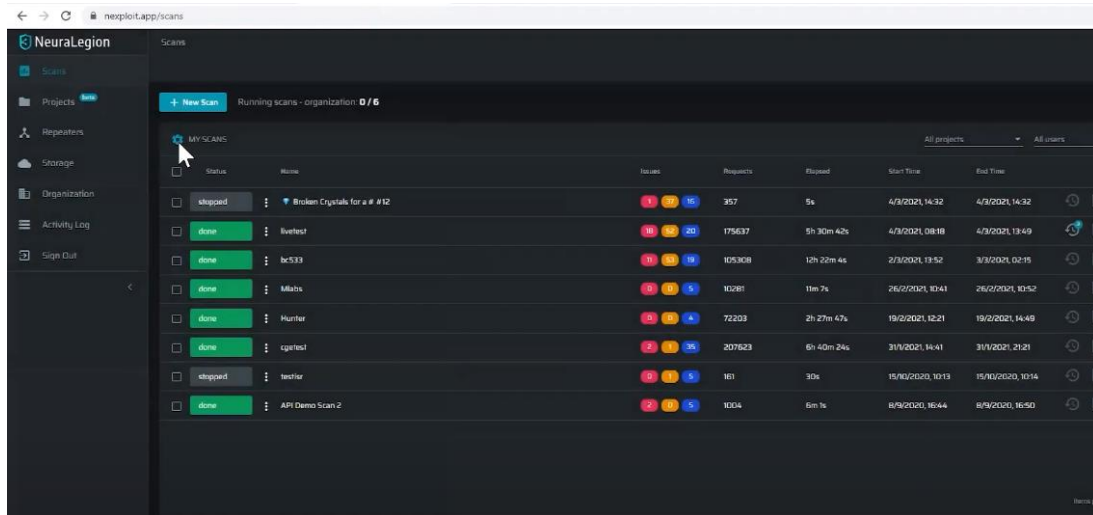


Figura 42: Interfaz gráfica *Nexplot*.

Para realizar el escaneo, ha sido necesario pulsar la opción *New Scan*, la cuál abre una interfaz de opciones para personalizar el mismo. En este caso, se ha introducido la dirección http://IP_SERVIDOR/web en la pestaña *Scan Targets*.

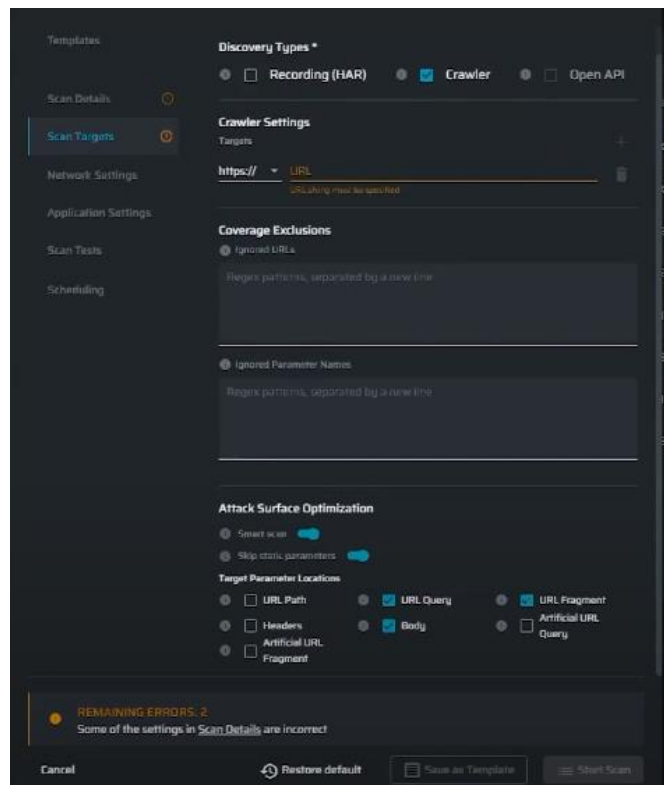


Figura 43: Pestaña *Scan Targets* de *Nexpose*.

El escaneo se ha realizado mediante la opción *Standard Scan*. Los módulos seleccionados en la pestaña *Scan Tests* han sido los siguientes: *Broken JWT Authentication*, *Broken SAML Authentication*, *Brute Force Login*, *Common Files*, *Cookie Security*, *Cross Site Request Forgery (CSRF)*, *Cross-Site Scripting (XSS)*, *Default Login Location*, *Directory Listing*, *DOM Cross-Site Scripting*, *Email Injection*, *File Upload*, *Full Path Disclosure (FPD)*, *Headers Security Check*, *HTML Injection*, *HTTP Method Fuzzer*, *Improper Assets Management*, *Insecure TLS Configuration*, *LDAP Injection*, *Local File Inclusion (LFI)*, *MongoDB injection*, *Open Buckets*, *Open DataBase*, *OS Command Injection*, *Prototype Pollution*, *Remote File Inclusion (RFI)*, *Secret Tokens*, *Server Side Template Injection (SSTI)*, *Server-Side JavaScript Injection*, *Server-Side Request Forgery (SSRF)*, *SQL injection (SQLI)*, *Unvalidated Redirect*, *Version Control System*, *WordPress Scan*, *XML External Entity (XXE)*, *XPath Injection*.

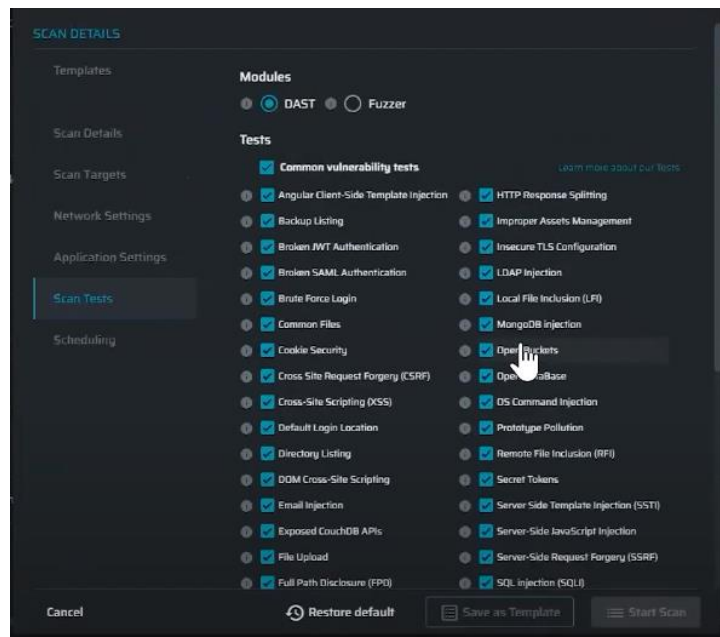


Figura 44: Pestaña *Scan Tests* de *Nexpose*.

3. Nexpose

Se ha hecho uso una vez más de un *free trial* en *Windows* proporcionado gracias al correo de la universidad. Tras la descarga del ejecutable desde la página oficial [83], se ha instalado *Nexpose* como un servicio accesible vía web en el puerto 3780.

Para su uso, ha sido necesario la apertura del ejecutable generado en la instalación como administrador. Lo cual ha abierto la interfaz gráfica vía web en el navegador predeterminado.

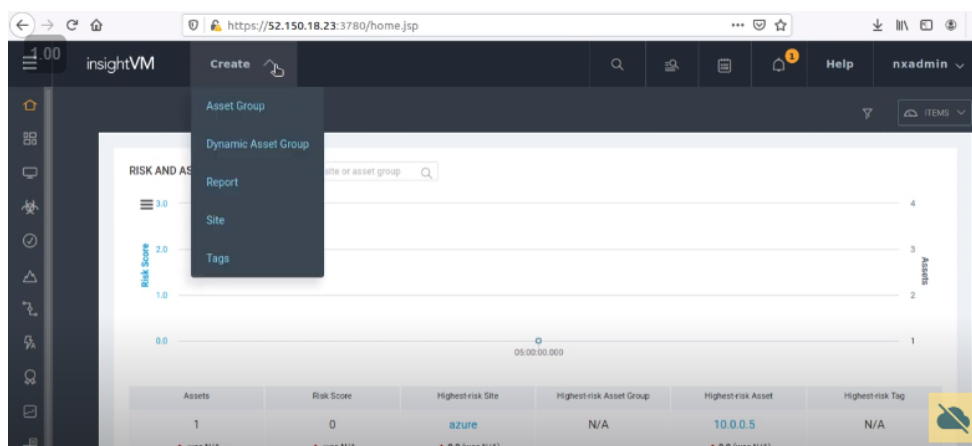


Figura 45: Interfaz gráfica *Nexpose*.

Para realizar el escaneo se ha seleccionado la opción *Site* del desplegable *Create*, como en la figura anterior. Posteriormente, se ha especificado la dirección IP del servidor en la pestaña *Assets*.

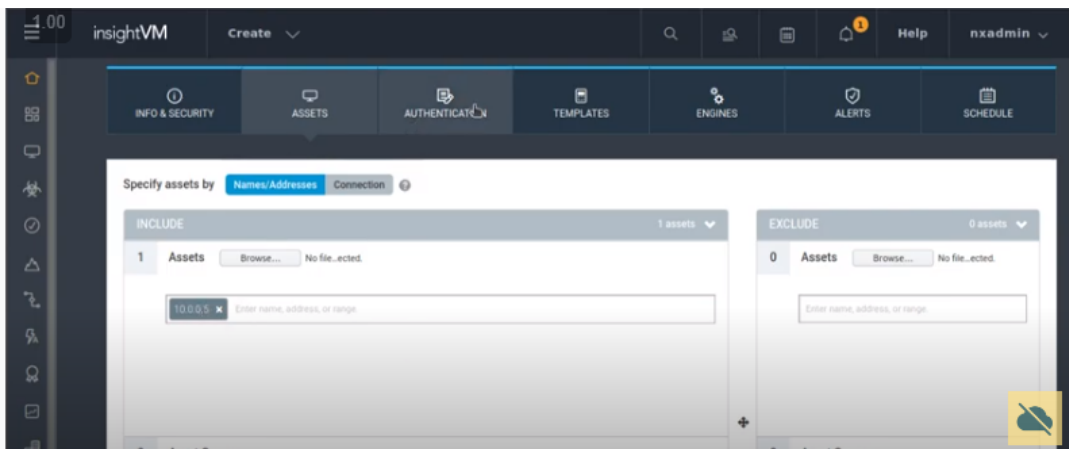


Figura 46: Pestaña *Assets* de *Create Site* en *Nexpose*.

Para terminar de configurar el escaneo, en la pestaña *Templates* se ha seleccionado la opción *Full Audit*.

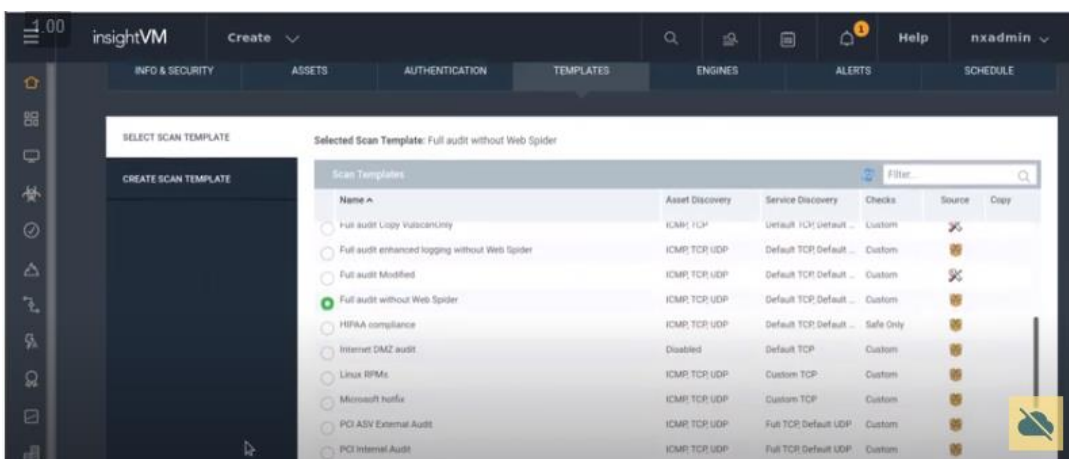


Figura 47: Pestaña *Templates* de *Create Site* en *Nexpose*.

4. SmartScanner

El instalador correspondiente para *Windows* está disponible vía web [84]. Se ha ejecutado como administrador el instalador y se ha iniciado el escaneo a http://IP_SERVIDOR/web mediante la interfaz gráfica de la herramienta.

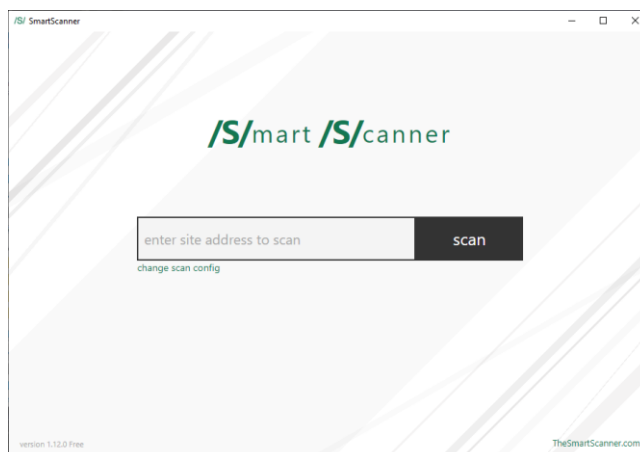


Figura 48: Interfaz gráfica de *SmartScanner*.

5. Webcruiser

Se ha descargado la herramienta desde la documentación oficial [85] para *Windows*. El escaneo se ha realizado mediante la opción *Scan Current Site* a *IP_SERVIDOR*. Se han seleccionado las siguientes opciones en la pestaña *Settings*:

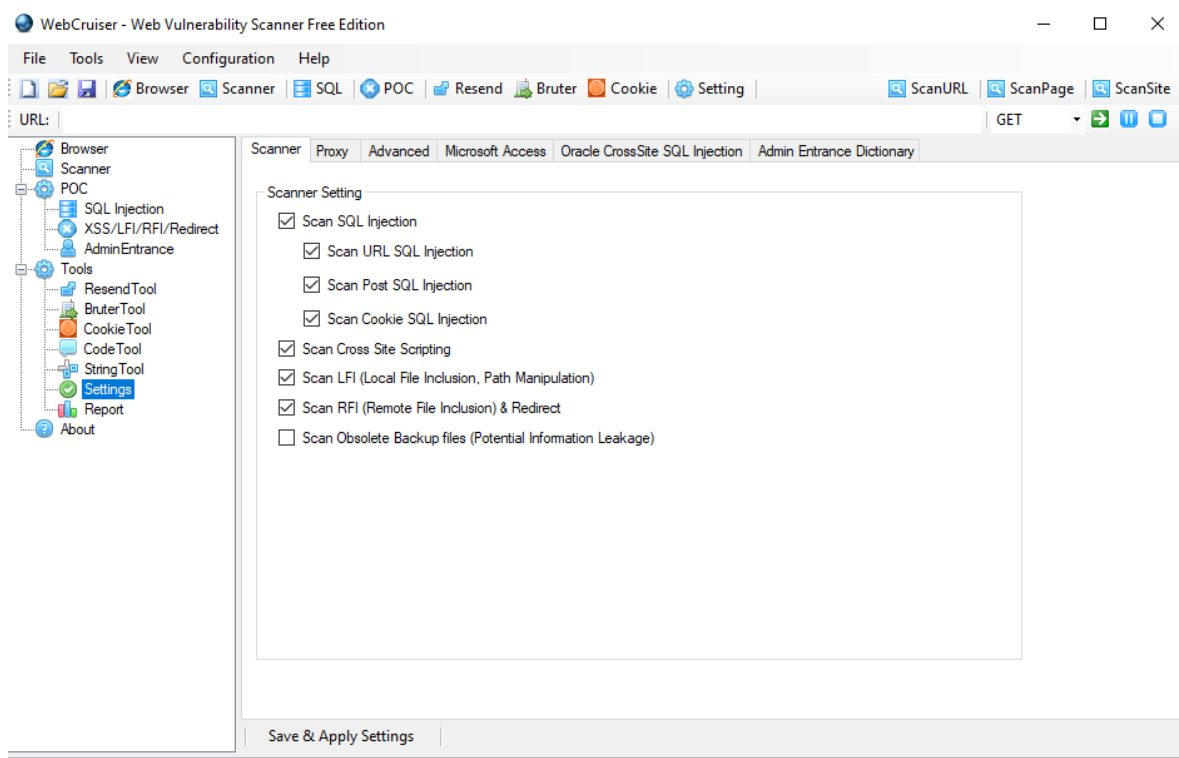


Figura 49: Opciones seleccionadas en el lanzamiento por *Webcruiser*

6. Wpscan

Esta herramienta está disponible en *Parrot OS* [48], se ha utilizado el comando `wpscan -api-token TOKEN -url IP_SERVIDOR` en la web dinámica tras conseguir un token para su uso desde la documentación oficial [86].

ANEXO C: INSTALACIÓN Y USO DE HERRAMIENTAS DE CAPTURA DE TRÁFICO

En este Anexo se expondrán los pasos y comandos necesarios para replicar la instalación y el uso de las herramientas de captura utilizadas en el trabajo para generar los ficheros de extensión *pcap*, *ipfix*, *csv*, *uri* y *short* que conforman el *dataset*. Dicho proceso ha sido realizado en una máquina virtual con el sistema operativo *Centos 7* [43]. Para facilitar la tarea, se han elaborado algunos *scripts* que automatizan los comandos expuestos en este Anexo, disponibles en https://github.com/fbuenoc97/TFG/tree/main/scripts_complementarios.

A. Wireshark

Esta herramienta está disponible en la máquina *Centos 7* [43], y ha sido la utilizada para la captura de tráfico *pcap*. Se ha utilizado el siguiente filtro de captura para filtrar el tráfico HTTP: *port 80*.

Tras la realización de la captura, se ha guardado la misma en un fichero de extensión *.pcap*.

B. Nfdump

Esta herramienta ha sido la utilizada en el equipo servidor *Centos 7* [43] para generar los ficheros de extensión *.ipfix*. El comando correspondiente para realizar la instalación ha sido `yum install nfdump`, y para el uso de la utilidad *nfpcapd*, es necesario instalar las siguientes dependencias:

```
yum install centos-release-scl -y
yum install devtoolset-9-gcc* -y
scl enable devtoolset-9 bash
cd /home/dit/Downloads/nfdump
./configure --enable-readpcap --enable-nfpcapd
make
sudo make install
```

Para realizar la conversión de *pcap* a *ipfix*, se han utilizado los siguientes comandos:

```
nfpcapd -r <pcap> -l <\dir> -t 10000000000
mv nfpcapd* <nombre_fich>.ipfix
```

C. Ficheros .csv, .uri y .short

Para generar los ficheros mencionados, se ha hecho uso de los comandos *tshark* y *sort* en el equipo servidor *Centos 7* [43] de la siguiente manera:

```
tshark -r <nombre_fich>.pcap -T fields -Y "http.request" -E header=y -e
http.accept -e http.accept_encoding -e http.accept_language -e http.authbasic
-e http.authcitrix -e http.authcitrix.domain -e http.authcitrix.password -e
http.authcitrix.session -e http.authcitrix.user -e http.authorization -e
http.bad_header_name -e http.cache_control -e http.chat -e http.chunk_boundary
-e http.chunk_size -e http.chunkd_and_length -e http.chunked_trailer_part -e
http.connection -e http.content_encoding -e http.content_length -e
http.content_length_header -e http.content_type -e http.cookie_pair -e
http.date -e http.decompression_disabled -e http.decompression_failed -e
http.file_data -e http.host -e http.http2_settings -e http.http2_settings_uri
```

```
-e http.last_modified -e http.leading_crlf -e http.location -e
http.next_request_in -e http.next_response_in -e http.notification -e
http.prev_request_in -e http.proxy_authenticate -e http.proxy_authorization -e
http.proxy_connect_host -e http.proxy_connect_port -e http.referer -e
http.request -e http.request.full_uri -e http.request.line -e
http.request.method -e http.request.uri.path -e http.request.uri.query -e
http.request.uri.query.parameter -e http.request.version -e http.request_in -e
http.request_number -e http.sec_websocket_accept -e
http.sec_websocket_extensions -e http.sec_websocket_key -e
http.sec_websocket_protocol -e http.sec_websocket_version -e http.server -e
http.set_cookie Set-
Cookie -e http.ssl_port -e http.subdissector_failed -e http.te_and_length -e
http.te_unknown -e http.time -e http.tls_port -e http.transfer_encoding -e
http.unknown_header -e http.upgrade -e http.user_agent
User
-Agent -e http.www_authenticate -e http.x_forwarded_for -e http.cookie -e
http.request.uri > <nombre_fich>.csv
```

```
tshark -r <nombre_fich>.csv -T fields -Y "http.request" -e http.request.uri >
<nombre_fich>.uri
```

```
sort $nombre_fich.uri | uniq > $nombre_fich.tmp
LANG=C sort -f $nombre_fich.tmp > $nombre_fich.short
rm -f $nombre_fich.tmp 2>&1 1>/dev/null
```

ANEXO D: INSTALACIÓN Y USO DE HERRAMIENTAS DE DETECCIÓN: *INSPECTORLOG*

Para la detección de ataques en el *dataset* disponible se ha hecho uso de la herramienta *InspectorLog* [56]. En este Anexo se expondrán los pasos y comandos necesarios para llevar a cabo la instalación de esta y su uso para generar los ficheros de extensión *.attacks* y *.clean* que contienen las URIs detectadas como ataques y las que no, respectivamente. *InspectorLog* ha sido utilizado con las reglas de *Snort*, *ModSecurity* o *Nemesida* en el sistema operativo *Centos* 7 [43]. En https://github.com/fbuenoc97/TFG/tree/main/scripts_complementarios se proporcionan algunos scripts hechos para facilitar la tarea de detección al usuario.

Para su correcta instalación, *InspectorLog* necesita la instalación de la librería *pcre* a través de `yum install pcre-devel`. Tras esto, se ha ejecutado el comando `make` en el directorio raíz de la herramienta. A continuación, se proporcionan los comandos exactos necesarios para replicar el uso llevado a cabo en el estudio de *InspectorLog* con *Snort*, *ModSecurity* y *Nemesida*, además de replicar la generación de los datos obtenidos tras enlazar la detección de los tres anteriores.

A. Snort

Los comandos utilizados han sido los siguientes:

```
$RUTA_IL/inspectorlog -l $nombre_fich.uri -r $RUTA_RULES -t list -o $RUTA_SNORT/$nombre_fich.clean > $RUTA_SNORT/$nombre_fich.attacks
```

Donde el parámetro “-l” indica el fichero que contiene las URIs a leer, “-r” el directorio que contiene las reglas de Talos[54], “-t” el formato en el que se presentan las URIs y “-o” el fichero donde se arrojan las URIs que no han generado alertas.

B. ModSecurity

Para instalar el servicio, ha sido necesaria la instalación de *libmodsecurity V3* [87] también. Se ha creado el enlace simbólico: `ln -s /usr/lib64/libpcre.so.1.2.0 /usr/lib64/libpcre.so.3` y se ha establecido la ruta de carga de librerías dinámicas: `LD_LIBRARY_PATH=/usr/lib64; export LD_LIBRARY_PATH`. También ha sido necesario mover *libmodsecurity.so.3* a la ruta de carga establecida: `mv /usr/local/modsecurity/lib/* /usr/lib64/`. En este caso, el ejecutable que se ha utilizado es *ms-inspectorlog*. Los comandos utilizados para obtener los resultados deseados son los siguientes:

```
$RUTA_MS_IL/ms-inspectorlog -l $nombre_fich.uri -t list -r $RUTA_MS_IL/etc/basic_rules.conf -o $RUTA_MODSECURITY/$nombre_fich.clean > $RUTA_MODSECURITY/$nombre_fich.attacks
```

Donde el parámetro “-l” indica el fichero que contiene las URIs a leer, “-r” el fichero de configuración de *ModSecurity*, “-t” el formato en el que se presentan las URIs y “-o” el fichero donde se arrojan las URIs que no han generado alertas.

C. Nemesida

Los comandos utilizados han sido los siguientes:

```
$RUTA_IL/inspectorlog -l $nombre_fich.uri -m $RUTA_RULES/nemesida-rules-bin-20220109.txt -t list -o $RUTA_NEMESIDA/$nombre_fich.clean > $RUTA_NEMESIDA/$nombre_fich.attacks
```

Donde el parámetro “-l” indica el fichero que contiene las URIs a leer, “-m” el fichero que contiene las reglas de *Nemesida*, “-t” el formato en el que se presentan las URIs y “-o” el fichero donde se arrojan las URIs que no han generado alertas.

D. Datos IDS completo

Tras la generación de los ficheros *attacks* de cada IDS, se ha procedido a producir un fichero de URIs de ataque para cada herramienta que contiene aquellas URIs que han sido detectadas por al menos un detector, mediante los comandos:

```
cat "fichero attacks de la herramienta para el detector Snort" >> $nombre_fich.attacks
cat "fichero attacks de la herramienta para el detector ModSecurity" >> $nombre_fich.attacks
cat "fichero attacks de la herramienta para el detector Nemesida" >> $nombre_fich.attacks
cut -f 2 $nombre_fich.attacks > $nombre_fich.tmp
awk ' /^Uri/{print $2}' $nombre_fich.tmp > $nombre_fich.tmp1
sort $nombre_fich.tmp1 | uniq > $nombre_fich.tmp2
LANG=C sort -f $nombre_fich.tmp2 > $nombre_fich.attacks
rm -f $nombre_fich.tmp 2>&1 1>/dev/null
rm -f $nombre_fich.tmp1 2>&1 1>/dev/null
rm -f $nombre_fich.tmp2 2>&1 1>/dev/null
```

REFERENCIAS

- [1] H. Hindy *et al.*, “A Taxonomy of Network Threats and the Effect of Current Datasets on Intrusion Detection Systems,” *IEEE Access*, vol. 8, pp. 104650–104675, Jun. 2018, doi: 10.1109/access.2020.3000179.
- [2] “Snort - Network Intrusion Detection & Prevention System.” <https://www.snort.org/> (accessed Jun. 20, 2022).
- [3] “SpiderLabs/ModSecurity: ModSecurity is an open source, cross platform web application firewall (WAF) engine for Apache, IIS and Nginx that is developed by Trustwave’s SpiderLabs. It has a robust event-based programming language which provides protection from a range of attacks against web applications and allows for HTTP traffic monitoring, logging and real-time analysis. With over 10,000 deployments world-wide, ModSecurity is the most widely deployed WAF in existence.” <https://github.com/SpiderLabs/ModSecurity> (accessed Jun. 20, 2022).
- [4] “Nemesida WAF - complex site protection system with machine learning.” <https://nemesida-waf.com/> (accessed Jun. 20, 2022).
- [5] D. of I. & C. S. S. Hettich, S.D. Bay, “The UCI KDD Archive,” 1999. <http://kdd.ics.uci.edu/>.
- [6] “Data Overview - CAIDA.” <https://www.caida.org/catalog/datasets/overview/> (accessed Jun. 20, 2022).
- [7] N. Moustafa and J. Slay, “UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” *2015 Mil. Commun. Inf. Syst. Conf. MilCIS 2015 - Proc.*, Dec. 2015, doi: 10.1109/MILCIS.2015.7348942.
- [8] M. H. Abdulraheem and N. B. Ibraheem, “A detailed analysis of new intrusion detection dataset,” *J. Theor. Appl. Inf. Technol.*, vol. 97, no. 17, pp. 4519–4537, 2019.
- [9] T. S. Riera, J. R. B. Higuera, J. B. Higuera, J. J. M. Herraiz, and J. A. S. Montalvo, “Prevention and fighting against web attacks through anomaly detection technology. A systematic review,” *Sustain.*, vol. 12, no. 12, pp. 1–45, 2020, doi: 10.3390/su12124945.
- [10] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, “Towards a Reliable Intrusion Detection Benchmark Dataset,” *Softw. Netw.*, vol. 2017, no. 1, pp. 177–200, 2017, doi: 10.13052/jsn2445-9739.2017.009.
- [11] R. Sommer and V. Paxson, “Outside the closed world: On using machine learning for network intrusion detection,” *Proc. - IEEE Symp. Secur. Priv.*, pp. 305–316, 2010, doi: 10.1109/SP.2010.25.
- [12] “Dataset Search.” <https://datasetsearch.research.google.com/>.
- [13] “OWASP Foundation | Open Source Foundation for Application Security.” <https://owasp.org/> (accessed Jun. 21, 2022).
- [14] “OWASP Top 10:2021.” <https://owasp.org/Top10/> (accessed Jun. 20, 2022).
- [15] “NUEVO OWASP TOP 10 2021 » Hacking Lethani.” <https://hacking lethani.com/es/owasp-top-10-2021/> (accessed Jun. 21, 2022).
- [16] “Message from Synack.” <https://www.synack.com/blog/preventing-broken-access-control-the-no-1-vulnerability-in-the-owasp-top-10-2021/> (accessed Jun. 21, 2022).
- [17] “A01 Broken Access Control - OWASP Top 10:2021.” https://owasp.org/Top10/A01_2021-Broken_Access_Control/ (accessed Jun. 21, 2022).
- [18] “A02 Cryptographic Failures - OWASP Top 10:2021.” https://owasp.org/Top10/A02_2021-Cryptographic_Failures/ (accessed Jun. 21, 2022).
- [19] “Secure against the OWASP Top 10 for 2021 | Chapter 2: Cryptographic failures (A2).” <https://support.f5.com/csp/article/K00174750> (accessed Jun. 21, 2022).
- [20] “Injection Theory | OWASP Foundation.” https://owasp.org/www-community/Injection_Theory

- (accessed Jun. 21, 2022).
- [21] “A03 Injection - OWASP Top 10:2021.” https://owasp.org/Top10/A03_2021-Injection/ (accessed Jun. 21, 2022).
- [22] “A04 Insecure Design - OWASP Top 10:2021.” https://owasp.org/Top10/A04_2021-Insecure_Design/ (accessed Jun. 21, 2022).
- [23] “A05 Configuración de Seguridad Incorrecta - OWASP Top 10:2021.” https://owasp.org/Top10/es/A05_2021-Security_Misconfiguration/ (accessed Jun. 21, 2022).
- [24] “A06 Vulnerable and Outdated Components - OWASP Top 10:2021.” https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/ (accessed Jun. 21, 2022).
- [25] “A07 Identification and Authentication Failures - OWASP Top 10:2021.” https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/ (accessed Jun. 21, 2022).
- [26] “A08 Software and Data Integrity Failures - OWASP Top 10:2021.” https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/ (accessed Jun. 21, 2022).
- [27] “A09:2021-Security Logging and Monitoring Failures | by Shivam Bathla | Medium.” https://medium.com/@shivam_bathla/a09-2021-security-logging-and-monitoring-failures-88c1c349f5a6 (accessed Jun. 21, 2022).
- [28] “A09 Security Logging and Monitoring Failures - OWASP Top 10:2021.” https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/ (accessed Jun. 21, 2022).
- [29] “A10 Falsificación de Solicitud del Lado del Servidor (SSRF) - OWASP Top 10:2021.” [https://owasp.org/Top10/es/A10_2021-Server-Side_Request_Forgery_\(SSRF\)](https://owasp.org/Top10/es/A10_2021-Server-Side_Request_Forgery_(SSRF)) (accessed Jun. 21, 2022).
- [30] “Software | MITRE ATT&CK®.” <https://attack.mitre.org/software/> (accessed Jun. 21, 2022).
- [31] “Wireshark · Go Deep.” <https://www.wireshark.org/> (accessed Jul. 06, 2022).
- [32] “Wireshark ¿Que es y para que sirve? – Ciberseguridad Comprensible.” <https://cl0udswxseque.wordpress.com/2020/05/09/wireshark-que-es-y-para-que-sirve/> (accessed Jul. 06, 2022).
- [33] “phaag/nfdump: Netflow processing tools.” <https://github.com/phaag/nfdump> (accessed Jul. 06, 2022).
- [34] “NFDUMP.” <http://nfdump.sourceforge.net/> (accessed Jul. 06, 2022).
- [35] “¿Qué es un firewall? - Cisco.” https://www.cisco.com/c/es_es/products/security/firewalls/what-is-a-firewall.html (accessed Jun. 21, 2022).
- [36] “¿Qué es un WAF? | Explicación de Web Application Firewall | Cloudflare.” <https://www.cloudflare.com/es-es/learning/ddos/glossary/web-application-firewall-waf/> (accessed Jun. 21, 2022).
- [37] “¿Qué son y para qué sirven los SIEM, IDS e IPS? | INCIBE.” <https://www.incibe.es/protege-tu-empresa/blog/son-y-sirven-los-siem-ids-e-ips> (accessed Jun. 21, 2022).
- [38] “¿Qué es el sistema de detección de intrusiones (IDS)? Funcionamiento, tipos, mejores herramientas ★ Aprende a Programar Gratis.” <https://aprendiendoaprogramar.es/blog/que-es-el-sistema-de-deteccion-de-intrusiones-ids-funcionamiento-tipos-mejores-herramientas/> (accessed Jul. 06, 2022).
- [39] “Mejores IDS Opensource para Detección de Intrusiones – Proteger mi PC.” <https://protegermipc.net/2018/02/22/mejores-ids-opensource-deteccion-de-intrusiones/> (accessed Jun. 21, 2022).
- [40] J. E. Díaz-Verdejo, A. Estepa, R. Estepa, G. Madinabeitia, and F. J. Muñoz-Calle, “A methodology for conducting efficient sanitization of HTTP training datasets,” *Futur. Gener. Comput. Syst.*, vol. 109, pp. 67–82, Aug. 2020, doi: 10.1016/J.FUTURE.2020.03.033.

- [41] “Wireshark · Go Deep.” <https://www.wireshark.org/> (accessed Jun. 20, 2022).
- [42] “phaag/nfdump: Netflow processing tools.” <https://github.com/phaag/nfdump> (accessed Jun. 20, 2022).
- [43] “Download.” <https://www.centos.org/download/> (accessed Jun. 20, 2022).
- [44] “VMware Workstation Player | VMware | ES.” <https://www.vmware.com/es/products/workstation-player.html> (accessed Jun. 20, 2022).
- [45] “DEPARTAMENTO DE INGENIERÍA TELEMÁTICA.” <http://trajano.us.es/> (accessed Jun. 20, 2022).
- [46] “Get Ubuntu | Download | Ubuntu.” <https://ubuntu.com/download> (accessed Jun. 20, 2022).
- [47] “Kali Linux | Penetration Testing and Ethical Hacking Linux Distribution.” <https://www.kali.org/> (accessed Jun. 20, 2022).
- [48] “Parrot Security.” <https://www.parrotsec.org/> (accessed Jun. 20, 2022).
- [49] “Vulnerability Scanning Tools | OWASP Foundation.” https://owasp.org/www-community/Vulnerability_Scanning_Tools (accessed Jun. 20, 2022).
- [50] “Exploit Public-Facing Application, Technique T1190 - Enterprise | MITRE ATT&CK®.” <https://attack.mitre.org/techniques/T1190/> (accessed Jun. 20, 2022).
- [51] “SQLRat, Software S0390 | MITRE ATT&CK®.” <https://attack.mitre.org/software/S0390/> (accessed Jun. 20, 2022).
- [52] “SecTools.Org Top Network Security Tools.” <https://sectools.org/> (accessed Jun. 20, 2022).
- [53] “CWE - CWE-1344: Weaknesses in OWASP Top Ten (2021) (4.7).” <https://cwe.mitre.org/data/definitions/1344.html> (accessed Jun. 20, 2022).
- [54] “Talos - Author of the Official Snort Rule Sets.” <https://www.snort.org/talos> (accessed Jun. 20, 2022).
- [55] “WebHome < Main < EmergingThreats.” <https://doc.emergingthreats.net/> (accessed Jun. 20, 2022).
- [56] J. E. Díaz-Verdejo, A. Estepa, R. Estepa, G. Madinabeitia, and F. J. Muñoz-Calle, “A methodology for conducting efficient sanitization of HTTP training datasets,” *Futur. Gener. Comput. Syst.*, vol. 109, no. March, pp. 67–82, 2020, doi: 10.1016/j.future.2020.03.033.
- [57] B. E. Strom *et al.*, “Finding Cyber Threats with ATT&CK™-Based Analytics,” 2017.
- [58] “cloudacademy/static-website-example: Static website to use with Cloud Academy labs.” <https://github.com/cloudacademy/static-website-example> (accessed Jun. 20, 2022).
- [59] “Arachni/arachni: Web Application Security Scanner Framework.” <https://github.com/Arachni/arachni> (accessed Jun. 20, 2022).
- [60] “Command line user interface · Arachni/arachni Wiki.” <https://github.com/Arachni/arachni/wiki/Command-line-user-interface#checks-list> (accessed Jun. 20, 2022).
- [61] “Annotated Option List · sullo/nikto Wiki.” <https://github.com/sullo/nikto/wiki/Annotated-Option-List> (accessed Jun. 20, 2022).
- [62] “golismero/golismero: GoLismero - The Web Knife.” <https://github.com/golismero/golismero> (accessed Jun. 20, 2022).
- [63] “Grabber! Like a Petit Pimouss’.” <http://rgaucher.info/beta/grabber/> (accessed Jun. 20, 2022).
- [64] “Descargar Java para Windows.” https://www.java.com/es/download/ie_manual.jsp (accessed Jun. 20, 2022).
- [65] “NSEDoc Reference Portal: NSE Categories — Nmap Scripting Engine documentation.” <https://nmap.org/nsedoc/categories/> (accessed Jun. 20, 2022).
- [66] “nuclei-templates/TEMPLATES-STATS.md at master · projectdiscovery/nuclei-templates.” <https://github.com/projectdiscovery/nuclei-templates/blob/master/TEMPLATES-STATS.md> (accessed Jun. 20, 2022).

- [67] “Cómo instalar OpenVAS en Kali Linux 2020 - Solvetic.” <https://www.solvetic.com/tutoriales/article/8278-como-instalar-openvas-en-kali-linux/> (accessed Jun. 20, 2022).
- [68] “10 Scanning a System — Greenbone Enterprise Appliance 21.04.18 documentation.” <https://docs.greenbone.net/GSM-Manual/gos-21.04/en/scanning.html#scanconfigs> (accessed Jun. 20, 2022).
- [69] “Vega Vulnerability Scanner.” <https://subgraph.com/vega/index.en.html> (accessed Jun. 20, 2022).
- [70] “Installation — w3af - Web application attack and audit framework 2019.1.2 documentation.” <http://docs.w3af.org/en/latest/install.html> (accessed Jun. 20, 2022).
- [71] “Python 2.7.0 Release | Python.org.” <https://www.python.org/download/releases/2.7/> (accessed Jun. 20, 2022).
- [72] “pip · PyPI.” <https://pypi.org/project/pip/> (accessed Jun. 20, 2022).
- [73] “Plugins | w3af - Open Source Web Application Security Scanner.” <http://w3af.org/plugins> (accessed Jun. 20, 2022).
- [74] “Wapiti: a Free and Open-Source web-application vulnerability scanner in Python.” <https://wapiti-scanner.github.io/> (accessed Jun. 20, 2022).
- [75] “Havij Download - Advanced Automated SQL Injection Tool - Darknet.” https://www.darknet.org.uk/2010/09/havij-advanced-automated-sql-injection-tool/#google_vignette (accessed Jun. 20, 2022).
- [76] “Download Nessus | Tenable®.” <https://www.tenable.com/downloads/nessus?loginAttempted=true> (accessed Jun. 20, 2022).
- [77] “Lavakumar/IronWASP: Source code of IronWASP.” <https://github.com/Lavakumar/IronWASP> (accessed Jun. 20, 2022).
- [78] “spinkham/skipfish: Web application security scanner created by leamtuf for google - Unofficial Mirror.” <https://github.com/spinkham/skipfish> (accessed Jun. 20, 2022).
- [79] “Wikto Free Download.” <https://wikto.apponic.com/> (accessed Jun. 20, 2022).
- [80] “xmendez/wfuzz: Web application fuzzer.” <https://github.com/xmendez/wfuzz> (accessed Jun. 20, 2022).
- [81] “Burp Suite - Application Security Testing Software - PortSwigger.” <https://portswigger.net/burp> (accessed Jun. 20, 2022).
- [82] “NexPloit - NeuraLegion - Bright Security.” <https://brightsec.com/nexploit-neuralegion/> (accessed Jun. 20, 2022).
- [83] “Try Nexpose.” <https://www.rapid7.com/try/nexpose/> (accessed Jun. 20, 2022).
- [84] “Download - Smart Vulnerability Scanner.” <https://www.thesmartscanner.com/download> (accessed Jun. 20, 2022).
- [85] “WebCruiser Web Vulnerability Scanner - Free download and software reviews - CNET Download.” https://download.cnet.com/WebCruiser-Web-Vulnerability-Scanner/3000-10247_4-75064882.html (accessed Jun. 20, 2022).
- [86] “WPScan: WordPress Security.” <https://wpscan.com/> (accessed Jun. 20, 2022).
- [87] “Compilation recipes for v3.x · SpiderLabs/ModSecurity Wiki.” <https://github.com/SpiderLabs/ModSecurity/wiki/Compilation-recipes-for-v3.x#centos-7-minimal-dynamic> (accessed Jun. 20, 2022).

