PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

MAESTRÍA EN REDES DE COMUNICACIÓN

PERFIL DEL TRABAJO PREVIO LA OBTENCIÓN DEL TÍTULO DE:

MÁGISTER EN REDES DE COMUNICACIÓN

TEMA:

ESTUDIO DE LAS CARACTERÍSTICAS DE SEGURIDAD DE SERVIDORES WEB EN ENTORNOS DE SOFTWARE LIBRE APLICABLES A LA PROTECCIÓN DE SITIOS DINÁMICOS

ERNESTO PÉREZ ESTÉVEZ

Quito – 2014

Agradecimientos

A Linus Torvalds, por haberme ayudado a encontrar un sentido a mi profesión.

A Paul Bernal por aclarar miles de dudas y tener paciencia

A la mamisa y a los mapases!

A mis padres

A todos los que me han apoyado, y también a la que no.

Tabla de contenidos

Índice de Tablas

Índice de Imágenes

Capítulo 1 : Introducción

La seguridad informática sigue siendo en la actualidad uno de los mayores problemas y preocupaciones de empresas y personas. Los sistemas informáticos son creados por seres humanos, y los seres humanos cometemos errores. Por tanto durante el tiempo de vida de un sistema informático este puede contener, y seguramente contendrá, errores de concepto, de diseño, de programación. Y estos errores podrán ser utilizados por sistemas o personas para comprometer la información que en ellos se almacena.

En la actualidad la programación de sistemas orientados a la web es una realidad, más aún desde inicios de esta década en el país se ha visto una explosión de sitios web con funciones informativas y directivas. Y junto con ellos se ha incrementado la exposición y por tanto el riesgo de ser atacados. El personal informático a cargo de la creación y mantenimiento de los sitios web muchas veces no tiene la certeza o el conocimiento de que su aplicación web tiene problemas de seguridad que podrían ser explotados desde cualquier lugar del planeta.

En este trabajo se realiza un estudio y caracterización de los sistemas de seguridad utilizados en cuatro servidores http en Software Libre que permiten correr aplicaciones web dinámicas. La finalidad que se persigue es la de mitigar o coartar los ataques más comunes que se han venido haciendo 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.

Los servidores web serán valorados desde diversos aspectos como por ejemplo: la facilidad de implementación de una solución de seguridad, el uso de características implícitas dentro del servidor web o uso de herramientas externas a este, el sistema de reportes de ataques, tipos de ataque que se protegen. Permitiéndonos de esta forma determinar un servidor web que nos permita de forma fácil asegurar aplicaciones web con la finalidad de mitigar los ataques más populares a las aplicaciones dinámicas.

A través del establecimiento, medición y tabulación de indicadores sobre los diversos servidores web existentes, se podrá determinar el más adecuado en términos de eficiencia y características de seguridad con la finalidad de implementar sobre éste un sistema web con contenidos dinámicos de manera que el usuario o administrador de éste pueda operarle de forma más segura.

Existen herramientas comerciales y en Software Libre que proveen servicios de detección de intrusiones o de reacción ante intrusiones, así como cortafuegos. Este trabajo persigue la implementación de un servidor web con un sistema de seguridad incorporado a través del uso de un servidor web realizado en Software Libre, que pueda ser utilizado por instituciones públicas y privadas, grandes, medianas o pequeñas del país como herramienta de protección ante ataques a sitios con contenidos dinámicos de forma tal que se mitigue, disminuya o elimine el daño en caso de que se intenten explotar fallas de seguridad presentes en este. Servidor que podrá ser usado dentro de un marco no privativo, en un ambiente colaborativo por el uso de Software Libre y con un bajo costo de implementación.

La primera parte de este anteproyecto de tesis de maestría en comunicaciones presenta los trabajos realizados previamente relacionados con el tema, para luego pasar a describir la justificación de realizarlo. Una tercera parte, señala, los objetivos, un índice tentativo y finalmente propone un cronograma de desarrollo de la investigación. La bibliografía de este documento está sustentada por extensas búsquedas realizadas en bibliotecas virtuales y repositorios de tesis digitales en el país.

1.1 Antecedentes

La prevención de ataques de seguridad informática es una labor multidisciplinaria y, lógicamente, compuesta por muchos factores que permitan vigilar, monitorear y corregir estos eventos. Es un tema que ha tomado mucha importancia incluso a nivel de estados se está procurando a través de la ITU de crear en los países centros de reacción a incidentes informáticos (CSIRT) y las organizaciones comprenden ya la necesidad de tener personal especializado en recibir y tratar incidentes de seguridad informática[2].

El servicio web es la forma de una institución exponer, e incluso brindar de forma parcial o total, sus servicios o productos por lo que es muy importante tomar medidas de seguridad extremas para evitar, mitigar, o dificultar ataques a este.

Todo sistema expuesto a la internet ha sido y es posible objeto de ataques informáticos, siendo un riesgo con el cual debemos aprender a vivir tomando las medidas adecuadas de seguridad. Todos los días en el planeta y en el país ocurren eventos de seguridad informática de diversa índole que afectan las finanzas, credibilidad y la buena marcha de las operaciones de las organizaciones; entre ellos podemos mencionar continuos cambios de página en instituciones públicas y privadas [3], reportes de robo de información a entidades que brindan servicio público por parte de anonymous [4][5][6], incluyendo fallas repentinas de funcionamiento en sitios web, etc.

Teniendo en cuenta la experiencia profesional en el campo, hemos constatado cómo clientes importantes ven afectado el funcionamiento de sus sitios web por no tomar en cuenta sistemas con la seguridad en mente [5][6]. Si verificamos sitios atacados últimamente, el problema continúa [3].

Estos sistemas son muchas veces la espina dorsal de una organización y, producto de fallas humanas en la programación y mantenimiento de ellos, ocurren pérdidas de credibilidad, financieras, o se experimenta caída del servicio producto de estas fallas.

A partir del año 2008, a través de la firma por parte del Presidente de la República del decreto 1014[1], el estado ecuatoriano se ha tratado de promover una política de uso de Software Libre en las instituciones, lo que ha traído como consecuencia positiva que muchas instituciones del país comiencen a utilizar Software Libre para sus sistemas. Desafortunadamente sí se nota que en las diversas empresas que utilizan Software Libre en el país la preparación de muchos administradores de red no es lo suficientemente sólida en temas de seguridad como para implementar alternativas de protección a sus sitios y, en la mayoría de los casos, no existe personal informático dedicado exclusivamente a la seguridad.

Con la finalidad de que exista la posibilidad de migrar de los servicios actuales a uno más seguro así como evidenciar las mejoras o falencias de seguridad en sistemas similares, en esta tesis evaluaremos cómo proteger un servidor web contra ataques de seguridad a aplicaciones dinámicas que en éste corran.

Luego de un análisis bibliográfico podemos indicar que en nuestro medio no existen trabajos anteriores al respecto, aunque sí hay varios intentos de: mejorar la seguridad de los sistemas mediante el estudio de protocolos de encriptación, aseguramiento en general del Sistema Operativo Linux [7] y de un servidor web determinado si se aplican correctas técnicas de programación [8], etc.

1.2 Justificación

En el país muchas organizaciones se ven obligadas a buscar soluciones para tratar de fortalecer el comportamiento de sus sitios web; las que tienen suficientes recursos, se consideran afortunadas y hacen adquisición de cajas negras para que les ayuden en la labor. Equipos tales como cortafuegos o sistemas de detección de intrusos, los cuales no dejan de cumplir su labor y hasta podrían verse como un complemento muy importante a la seguridad de los sitios web. En todo caso muchas veces se falla en implementar medidas cuya solución no está en la adquisición de equipamiento, sino en aprovechar las diversas opciones que nos brindan las aplicaciones en Software Libre que se disponen [6].

Los ataques contra aplicaciones web dinámicas, aunque ya conocidos hace años, no pueden solucionarse solamente con estos equipos o mejorando las técnicas de programación.

Es cierto que existen diversas variantes adicionales para minimizar el impacto de un ataque; medidas típicas de la seguridad informática de Preparar, Identificar, Contener, Remediar, Recuperar [10]. O mediante el uso de páginas estáticas que, al no existir en ellas código generado automáticamente no puede ser este engañado para realizar acciones no esperadas.

Sin embargo son pocos los sitios que utilizan generación de contenido estático en la actualidad, pues los requerimientos de las instituciones y usuarios van cambiando de acuerdo a las necesidades o políticas y el uso de herramientas de programación para la generación de contenido dinámico como es el caso de PHP o Java que hacen a los sitios más vistosos y fáciles de manejar al utilizarse un programa que interactúa con una base de datos para generar los contenidos.

Pero estos programas, al aceptar información enviada por el usuario para generar su contenido, pueden ser engañados y obligados a ejecutar acciones no esperadas por el administrador del sitio ya sea por vulnerabilidades en el diseño o por errores en el manejo.

Para este trabajo se utilizarán los métodos: analítico-sintético para encontrar los problemas de seguridad más comunes, los servidores web más utilizados en Internet, identificando los elementos útiles para nuestra investigación. El método histórico-lógico con la finalidad de estudiar y determinar los antecedentes históricos relacionados con las vulnerabilidades explotadas en aplicaciones dinámicas. Y el método inductivo-deductivo para la especificación de la propuesta de solución a explotación de vulnerabilidades desde los servidores web en Software Libre

Se determinarán los servidores web más utilizados en el país y más conocidos en Internet; comparando además el desempeño en seguridad entre servidores que permitan ofrecer las mismas prestaciones. Para esto se prepararán escenarios para realizar las pruebas requeridas.

Se estudiarán ataques de seguridad que han ocurrido en el medio actualmente, las razones por las que pueden haber ocurrido. Evidenciando por qué debemos protegernos de ellos.

Se analizarán las técnicas de seguridad provistas por varios de los servidores web más populares y su correcta configuración con la finalidad de determinar el más adecuado para asegurar aplicaciones web dinámicas.

Se expondrán alternativas no intrusivas que permitan fortalecer los servidores actuales sin necesidad de realizar cambios drásticos o demorados en el servicio que una organización actualmente ofrezca.

Las técnicas estudiadas quedarán a disposición de administradores de todos los niveles con necesidades de asegurar sus sistemas. Generando además paquetes instaladores pre-configurados con las medidas de seguridad adecuadas. Así como documentación que permita a estos poder hacer un uso con conocimiento de causa de las técnicas para fortalecer sus servicios.

Consideramos que el proyecto tiene un valor social ya que mediante la aplicación de sencillas técnicas de seguridad y utilizando Software Libre se podrá lograr alta disponibilidad y evitar caídas de servicio innecesarias que afecten a los interesados. Teniendo en cuenta que en el futuro las comunicaciones a través de sitios web seguirán siendo prevalentes a la hora de interactuar con usuarios y clientes se hace necesario adaptar todos nuestros sistemas a funcionar en un ambiente web el que hace un par de décadas no era conocido, pero que en la actualidad se hace necesario asegurarle ya que pueden existir intentos hostiles de los cuales nos debemos proteger.

Es un aporte además a la industria, ya que a través de una correcta lectura y aplicación adecuada de las técnicas aquí estudiadas, se puede mitigar o eliminar el impacto de ataques contra los sitios web de las redes de las empresas.

Este trabajo hará uso de herramientas en Software Libre por la flexibilidad que nos ofrece de evolucionar rápidamente, integración y facilidades de estudio de estos procesos.

1.3 Objetivo General

Analizar las características de seguridad disponibles para servidores web en entornos de Software Libre con el fin de proteger de ataques a sitios dinámicos.

1.4 Objetivos Específicos

1. Estudiar los tipos de ataque más comunes a sitios dinámicos y determinar las vulnerabilidades que suelen ser aprovechadas para estos ataques.
2. Describir el efecto de las diferentes técnicas de ataques a sitios dinámicos en servidores web en entornos de Software Libre.
3. Analizar el estado del arte en cuanto a la reacción ante ataques contra servidores web en entornos de Software Libre.
4. Describir las características de seguridad disponibles en los servidores web más ampliamente usados.
5. Determinar el servidor web más apropiado para proteger aplicaciones dinámicas y describir este proceso de fortalecimiento.

6. Crear un instalador de este servidor web, que contenga por defecto las características de seguridad deseables para proteger un servicio web de los ataques estudiados.

1.5 Alcance

La tesis de grado culmina con la entrega de un documento que contenga:

- La valoración y determinación del servidor web en Software Libre más apropiado para proteger sitios dinámicos contra ataques.
- Las configuraciones para proteger a otros servidores web en Software Libre utilizando al servidor determinado como el más apropiado para proteger sitios dinámicos contra ataques
- Se demostrará las capacidades del instalador para proveer las seguridades indicadas.
- Se entregará un CD con el paquete de instalación del servidor web más apropiado para proteger sitios dinámicos contra ataques.

Capítulo 2 : Marco teórico

En esta sección describiremos diversos ataques que pueden ser realizados a sitios dinámicos y vulnerabilidades que se aprovechan por parte de los atacantes; qué efectos tienen estos ataques en el funcionamiento de estos sitios. Analizaremos además el estado del arte del proceso de reacción ante estos problemas y las diversas formas de protección propuestas que han surgido ante estos ataques. Que no son excluyentes entre sí, sino que por un elemental principio de seguridad deben aplicarse de forma conjunta para maximizar la capacidad de detección de estos eventos.

Finalizamos este capítulo enfocándonos en las caracterizaciones de seguridad de los servidores web que corren en ambiente libre. Realizando búsqueda en internet de los servidores web más utilizados, servidores web más utilizados en el país así como de servidores web orientados a la seguridad. Se finaliza realizando un análisis de estos servidores .

2.1 Sitios estáticos y dinámicos

Cuando se concibe la World Wide Web, como un medio para transmitir, divulgar información a través del protocolo HTML[1] alrededor del año 1989 por Tim Berners-Lee [11], este era un sistema superior a los entonces existentes pues permitía que un servidor enviara códigos HTML al navegador de un usuario para que este a su vez interpretara estos códigos y mostrara la información de una forma visual al usuario. Fue un éxito desde el inicio pues los protocolos existentes no permitían mostrar contenido rico en características como imágenes, formas, objetos a los usuarios. El uso de la WWW[2] fue el inicio de la masificación de la internet tal y como hoy la tenemos.

En un principio solamente se mostraba contenido estático sobre la información que el dueño del sitio web publicaba.

En esos momentos casi toda la internet inicial era basada en contenidos estáticos que no eran interpretados por ningún lenguaje de programación sino que se enviaban directamente al usuario los contenidos que se leían de disco.

1 HTML: HiperText Markup Language

2 WWW: World Wide Web

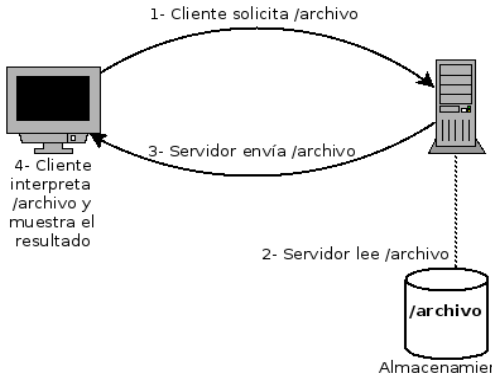


Figura 1: Proceso que se sigue para mostrar contenidos en sitios estáticos [A]

Como podemos observar en la figura 1 el contenido estático tiene una característica y es que la información se transmite del servidor web al navegador del cliente de la misma forma que está guardada en el servidor de origen. De suerte tal que esta podrá ser modificada solamente accediendo al servidor de forma remota a través de algún sistema que lo permita como podría ser SSH [3], FTP[4], entre otros.

Es por ello que en etapas iniciales se hacía mucho esfuerzo en asegurar los sistemas contra fallas de seguridad que aprovechaban problemas en el sistema de autenticación [12] para acceder a sitios web con el objetivo de subir archivos. Fallas en aplicaciones como wu-ftpd fueron mal utilizadas [13][14] y abusadas para lograr estos objetivos. Estos problemas son considerados en la actualidad pero ya no son la única forma de realizar modificaciones a un sitio web.

Sitios web con contenidos estáticos siguen siendo utilizados en la actualidad debido a una gran ventaja que tienen: los sitios estáticos tienen un alto desempeño al no tenerse que realizar interpretación de código para que una página web sea mostrada, siendo enviada a los clientes que solicitan el sitio la misma información una y otra vez sin tenerse que realizar ninguna labor adicional. Estos sitios son muy populares para sistemas informativos en los cuales no se requiere de intervención por parte del usuario.

3 SSH: Secure Shell protocolo encriptado de comunicaciones entre el cliente y el servidor. Utilizado ...

4 FTP: File Transfer Protocol uno de los primeros protocolos de transferencia de archivos entre cliente y servidor...

Ahora, los usuarios de este nuevo servicio se dieron cuenta en poco tiempo que requerían tener interacción con los usuarios para lograr sistemas que pudieran responder a necesidades específicas de los usuarios, para ello era necesario que, según el requerimiento del usuario, se mostrara o no la información en el sitio web. Para ello no se propuso un nuevo sistema que permitiera generar contenidos dinámicos sino que se creó el concepto de Interfaz Común de Entrada, conocida normalmente en Inglés como: CGI[5] que permite al servidor web invocar un programa y enviarle parámetros para que este devuelva la información que debe llegar al cliente.

Ahora el servidor web no envía literalmente un archivo al cliente para que éste le interprete, sino que cuando un cliente solicita un archivo, este archivo se ejecuta y el resultado de esta ejecución es entregado al cliente.

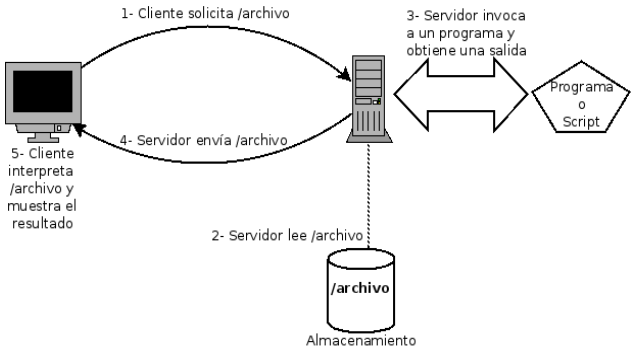


Figura 2: Resumen del proceso seguido para mostrar contenido dinámico [A]

2.2 Ataques a sitios dinámicos

Al utilizarse un programa el cual puede recibir parámetros, esta información puede ser procesada, almacenada y mostrada de diversas formas, lo que abrió un enorme potencial al uso de la web como herramienta de interacción entre usuarios y sitios web y también abrió un enorme campo a personas interesadas en confundir a estos programas para que generen respuestas no esperadas y sean aprovechadas para objetivos no relacionados con la labor del programa.

Existen formas específicas mediante las cuales se pueden realizar ataques a sitios web dinámicos, que pueden ser el resultado, o una combinación de factores como fallas en la programación, la administración y/o el usuario del sitio web.

5 CGI: Common Gateway Interface

Existen una multitud de herramientas y métodos para eliminar o mitigar estos tipos de ataques que lógicamente se apoyan en algún elemento de la cadena de transmisión entre el cliente y el servidor. Estos elementos pueden ser protecciones a nivel del usuario, protecciones en el servidor, protecciones en el código, como por ejemplo aquí detallamos:

Nivel	Propuesta de protección	Observaciones
Usuario	Implementación de un filtro por software que analice los contenidos que se están enviando o recibiendo por la web. Muchas veces viene incorporado dentro de una suite antivirus.	Ventajas: • Muchos sistemas antivirus ofrecen esta protección Desventajas: • La protección depende del usuario, para lograr una gran efectividad todos los usuarios que naveguen deberían tenerle instalada.
Usuario	Mejoras incorporadas a los navegadores ya sea a través de mejoras en el propio navegador como de plugins de seguridad que pueden existir para ellos.	Ventajas: • Los fabricantes de los navegadores actualizan continuamente su producto Desventajas: • Depende del usuario tener o no instalado un determinado plugin o actualizado su navegador.
Programación	Usar un framework adecuado, orientado a la seguridad	Ventajas: • Es una seguridad en el mismo sitio dinámico, por lo que puede proteger a todos los usuarios • El fabricante o creador se ocupa de mantener el framework e implementar nuevas medidas de seguridad según vayan siendo requeridas. Desventajas:

Nivel	Propuesta de protección	Observaciones
		• Tiene un costo el aprendizaje de un nuevo framework. • Una falla que no sea detectada o actualizada en el framework puede ser aprovechada en todos los sitios web que implementen este framework.
Programación	Hacer uso de programación segura	Ventajas: • Un programador con conocimientos de programación segura es un valioso activo para la organización • El uso de programación segura disminuye los ataques contra los sitios dinámicos Desventajas: • Muchos programadores no están preparados para realizar programación segura.
Servidor	Utilizar IDS[6]	Ventajas: • Conjunto de reglas que detectan problemas en diversos protocolos • Existen reglas para detectar ataques al servidor web Desventajas: • Solamente detectan los ataques • Las reglas normalmente tienen un costo el adquirirlas. • Hay que realizar una labor previa de eliminación de falsos

6 IDS: Sistema de Detección de Intrusos, del Inglés Intrusion Detection System

Nivel	Propuesta de protección	Observaciones
		positivos. • Las desventajas normalmente desestimulan su uso
Servidor	Utilizar IPS[7]	Ventajas: • Conjunto de reglas que detectan problemas en diversos protocolos • Existen reglas para detectar ataques al servidor web Desventajas: • Las reglas normalmente tienen un costo el adquirirlas • Hay que realizar una labor previa de eliminación de falsos positivos. • Las desventajas normalmente desestimulan su uso
Servidor	Implantar mejoras de seguridad a través de módulos de terceros que puedan existir para los servidores web. Implantar un WAF[8]	Ventajas: • Aunque son similares a un IDS/IPS, son dedicados exclusivamente a detectar ataques a servidores web. • Hay una pequeña pero útil oferta de estos módulos. Desventajas: • Estos módulos no vienen incluídos en la instalación estandar. • Requieren de una configuración inicial que puede resultar

7 IPS: Sistema de Prevención de intrusos, del Inglés Intrusion Prevention System

8 WAF: Cortafuegos de Aplicaciones Web, del Inglés Web Application Firewall

Nivel	Propuesta de protección	Observaciones
		engorrosa para el administrador • Existe posibilidad de falsos positivos que deben ser detectados al instalarse. • Muchas veces son basados en reglas y estas reglas pueden tener un costo adquirirlas.
Servidor	Utilizar servidor web orientado a la seguridad	Ventajas: • Las protecciones ya vienen incorporadas dentro del mismo servidor. Desventajas: • No hay muchos servidores web orientados a la seguridad. • Al verse una oportunidad comercial, pueden surgir propuestas que tengan un costo. • Este servidor puede actuar como WAF para otros a través de intermediar las peticiones (proxy reverso) • Pueden ser potencialmente difíciles de configurar.

Tabla 1: Protecciones según el nivel [A]

Como podemos ver hay una variedad de propuestas orientadas a proteger y protegerse de ataques que pueden ocurrir a través de la web. Estas opciones no son excluyentes entre sí y como en todo sistema de seguridad la sugerencia siempre es que mientras más medidas se adopten mejor preparados podemos estar para impedir un ataque.

2.3 Tipos de ataques a sitios dinámicos

Desde hace varios años el sitio de Open Web Application Security Project (OWASP) ha creado una metodología [15] para detectar los 10 riesgos de seguridad que más afectan a las aplicaciones web y publica una lista anual de estos [16]. En efecto varias, sino todas estas amenazas pueden se solucionadas desde la perspectiva de una correcta programación de los sistemas, y de hecho este es uno de los objetivos de OWASP. Sin embargo varias de estos riesgos pueden ser mitigados o prevenidos desde el mismo servidor en caso de que este tuviera implementadas características de seguridad. Estos problemas están presentes en diversos sitios orientados a la seguridad [17].

Los riesgos más populares en los 3 últimos reportes de OWASP son resumidos en la siguiente tabla:

Riesgo	**Posición según el año de publicación**		
	2013 [18]	**2010[18]**	**2007[19]**
Fallas de inyección	A1	A1	A2
Fallas en el manejo de sesión y autenticación	A2	A3	A7
Cross Site Scripting (XSS)	A3	A2	A1
Referencias inseguras directas a objetos	A4	A4	A4
Fallas en la configuración	A5	A6	
Exposición de datos sensitivos	A6	A7, A9[20]	A8, A9, A10[21]
Falta de control de acceso a funciones	A7	A8	
Cross-site Request Forgery (CSRF)	A8	A5	A5
Uso de componentes con fallas conocidas	A9		
Redirecciones no validadas	A10	A10	
Ejecución de archivos maliciosos			A3
Filtrado de información y manejo impropio de errores			A6

Tabla 2: Riesgos más populares según OWASP[16]

2.4 Descripción de los riesgos presentados

2.4.1 Fallas de Inyección [22]

Las fallas de inyección son las fallas más populares aprovechadas por los atacantes. No se relacionan con un sólo tipo de problema sino que incluye un espectro muy amplio de problemas. La inyección en resumen es lograr ejecutar un código malicioso a través de una aplicación web.

La inyección se logra a través de aplicaciones mal programadas que no validan los datos que reciben. Estos datos pueden estar cuidadosamente preparados para que cuando se invoque al programa que nuestra aplicación utiliza este realice acciones no previstas por sus autores.

La inyección puede realizar una de las siguientes acciones:

- Enviar y ejecutar un programa completo, típicamente un script
- Invocar programas del sistema como por ejemplo sendmail con la finalidad de enviar mails masivos.
- Realizar modificaciones a los datos almacenados en una base de datos SQL[9], se le conoce como inyección de SQL (SQL Injection) y es uno de los ataques más populares. El atacante debe encontrar un parámetro que el programador pasa a la BD, y modificar este parámetro con el código necesario para realizar modificaciones a la base de datos.
- No sólo se aprovechan fallas SQL sino también a otras bases de datos no SQL como LDAP[10], NoSQL[11], etc.

Las inyecciones deben ser corregidas en el código fuente del programa, sólo que a veces por desconocimiento o complejidad no son fáciles de encontrar.

Posible solución desde el servidor web

Los ataques a través de inyecciones pueden ser detectados de forma bastante certera pues normalmente envían comandos a ser ejecutados, y estos comandos pueden ser bloqueados por el servidor web, el WAF o el IDS/IPS

2.4.2 Fallas en el manejo de sesión y autenticación [23]

Las fallas en el manejo de sesión y autenticación son un problema bastante común y muy relacionado con fallas en la programación, los autores de los sistemas a veces hacen uso de sistemas de autenticación hechos en casa que muchas veces carecen de validaciones o auditorías públicas, cometiendo fallas que permiten que un atacante pueda hacerse de información privilegiada o confidencial de los usuarios.

Las fallas van desde incorrecto manejo de los timeouts, la exposición de datos de la sesión o acceso a la base de datos.

Posible solución desde el servidor web

9 SQL: Lenguaje de Consulta Estructurado del Inglés Structured Query Language

10 LDAP:Protocolo Ligero de Acceso a Directorios, del Inglés Lightweight Directory Access Protocol

11 NoSQL: No sólo SQL, otros sistemas de base de datos que no siguen el modelo SQL

Este tipo de falla es muy particular a cada caso y no es posible de generalizar o encontrar una forma genérica de bloquear desde el servidor web dada la cantidad de opciones y programas existentes.

2.4.3 Cross-Site Scripting [24]

El XSS es y ha permanecido como una de las fallas más populares en el uso de aplicaciones dinámicas. Ocurre cuando la aplicación web incluye datos suministrados por el usuario para que sean ejecutados o mostrados en el navegador de él mismo.

Estos datos los suministra el usuario de forma usualmente inconsciente, por ejemplo haciendo click en una URL[12] provista por un atacante ya sea a través de un sitio malicioso o un correo electrónico convincente. Esta URL viene especialmente preparada para hacer el el usuario no solamente abra la página a la que quiere ir, sino que invoque código del sitio web del atacante de forma tal que el atacante puede entonces secuestrar la sesión activa del usuario, con la que entonces accederá al sitio original.

Este tipo de ataque es muy peligroso cuando se accede a sitios con información confidencial o valiosa como Bancos, paneles de control de administración de servicios y ocurre porque la aplicación web no valida adecuadamente el código que le está llegando de parte del usuario.

Existen varias formas de realizar XSS:

1. XSS Almacenado: También conocido como XSS tipo 1 ó persistente. La llamada a este script se almacena permanentemente en un foro ó página web cuando un visitante llega a ella, ejecuta este script que es enviado por el servidor, pudiendo obtener información de sesión del visitante.
2. XSS Reflejado: Es el caso típico, conocido como XSS tipo 1, el XSS se produce cuando el usuario accede a un sitio web a través de un email o un enlace desde un sitio web comprometido, obteniéndose entonces información de sesión del usuario.

12 URL: localizador de recursos uniforme, del Inglés uniform resource locator

3. Ataques XSS basados en el Document Object Model (DOM) del navegador: En este ataque se trabajan formas de incorporar al DOM del navegador del usuario invocaciones a scripts que pueden obtener información del usuario como cookies o sesiones[25]. Han existido incluso ataques XSS DOM[26] a archivos PDF[13]. Este tipo de ataque no puede ser detectado fácilmente por el servidor[27], incluso cualquier medida que se pueda adoptar desde el servidor en contra de este tipo de ataque es bastante débil, completa y posiblemente no funcione. Siendo la solución más adecuada el mantener el software del cliente actualizado pues ocurre fundamentalmente sin que el servidor se entere es decir: completamente en el navegador del usuario.

Ambos ataques pueden ser aplicados tanto al navegador del cliente como al servidor. Existe una descripción de cómo se pueden verificar estos ataques en el sitio de OWASP [28]

Posible solución desde el servidor web

Estos ataques típicamente llegan al servidor invocando un script dentro de una variable. Y los scripts se pueden idenfiticar pues dentro de la variable vienen caracteres tales como: <, >, “ y/o ′. Un análisis de los contenidos de las variables que llegan al servidor pueden permitir encontrar estos caracteres y bloquear el ataque de XSS.

2.4.4 Referencias inseguras directas a objetos [29]

Las referencias inseguras directas a objetos ocurren cuando las aplicaciones han sido programadas de forma tal que no verifican, o verifican en ciertas partes, que el usuario esté autorizado a ver información que provee una determinada dirección. Por ejemplo un atacante puede determinar que una variable es utilizada para pasar el número de cédula, o número de usuario del sistema, y puede intentar cambiar este número por otro valor con la finalidad de obtener información a la que posiblemente no esté autorizado o deba limitársele el acceso de forma muy fácil. Esta forma funciona, y de hecho se utiliza mucho.

Posible solución desde el servidor web

Al ser una técnica mas bien cercana a la ingeniería social o de paciencia, es bastante poco probable que un servidor pueda implementar medidas de protección contra ella.

13 PDF: Formato de documentos portable, del Inglés: Portable Document Format

2.4.5 Fallas en la configuración [30]

Las fallas en la configuración son uno de los problemas más comunes en la web hoy en día. Fallas en la configuración del servidor web, el intérprete de la aplicación, el acceso a la base de datos, el sistema operativo, etc. Pueden contribuir a que un atacante aproveche la oportunidad para acceder a la información, eliminarla o modificarla.

Este tipo de falla es bastante fácil de corregir, con una apropiada educación y colaboración entre todas las partes involucradas, el sitio dinámico puede ser apropiadamente asegurado desde el Sistema Operativo hasta la aplicación que se está publicando.

Posible solución desde el servidor web

Un apropiado fortalecimiento de todos los componentes del servidor web puede mitigar o eliminar este tipo de ataques.

Como apoyo a este trabajo, es posible utilizar herramientas de verificación, sentido común y bibliografía libremente disponible en internet.

El atacante se apoya en problemas de configuración que no son únicamente debidos al servidor web, sino que están presentes de forma vertical en todo el sistema instalado.

2.4.6 Exposición de datos sensitivos

Es un problema que debería estar eliminado del escenario en estos momentos[31], sin embargo por diversas razones permanece. Se debe a que información sobre datos sensitivos pasa a través de la red de una forma poco segura, en algunos casos en texto claro, lo que permite al atacante obtenerla.

En efecto muchos sitios web, sino todos, cuando manejan datos sensitivos utilizan técnicas de encriptación.

Sin embargo estas técnicas pueden tener fallas de seguridad que permiten que sean explotadas[32]. Un adecuado manejo de la encriptación para impedir o dificultar el proceso de obtención de información debe ser obligatorio.

Posible solución desde el servidor web

El uso de algoritmos de encriptación de SSL[14]/TLS[15], versiones de SSL/TLS y protocolos de SSL/TLS más modernos que dificulten la labor del atacante. Además el uso de técnicas de aleatorización de la información que impidan que un atacante pueda adivinar el tráfico que circula por la red aún cuando no lo pueda ver.

14 SSL: Capa de socket segura, del Inglés Secure Socket Layer

15 TLS: Seguridad de la capa de transporte, del Inglés Transport Layer Security

2.4.7 Falta de control de acceso a funciones[33]

Las funciones de una aplicación pueden no estar protegidas, de suerte que un interesado puede acceder directamente a ellas e invocarles. Se debe verificar que las funciones no puedan ser accedidas directamente o que quien les acceda tenga los permisos necesarios para esto.

Posible solución desde el servidor web

No existe puesto que las funciones pueden estar almacenadas en diversos directorios siendo poco probable que un servidor pueda proteger eficientemente un gran listado de posibles directorios donde se encuentren estas funciones con un también gran listado de posibilidades de uso. Se sugiere aquí una adecuada programación de forma segura.

2.4.8 Cross-site Request Forgery [34]

La falsificación de peticiones intersitios, o en Inglés: Cross-Site Request Forgery permite a un atacante aprovechar el hecho de que algunos sitios web no validan las acciones que un usuario está realizando. Dicho de otra forma: confían en lo que el usuario solicita realizar sin tomar ninguna medida de protección ante invocaciones a URL potencialmente peligrosas. Para la aplicación el usuario está simplemente autenticado a través de los métodos implementados por la aplicación.

Para ejecutar el CSRF el atacante publica en un sitio malicioso una llamada a la aplicación del sitio objeto del ataque de forma tal que si el usuario entra, ya sea de forma voluntaria o a través de un correo electrónico de phishing, el sitio web malicioso le enviará suficiente código html como para hacer al navegador del usuario acceder al sitio web objeto del ataque y ejecutar una acción como podría ser borrar datos, agregar datos, acceder a datos, etc.

El servidor objeto del ataque solamente ve la IP[16] del equipo del cliente, por lo que para el servidor esta ha sido una solicitud válida.

Posible solución desde el servidor web

La solución más común es evitar que al sitio web se acceda a través de otros sitios mediante el análisis del header *HTTP-referers*[17]. Por tanto si quien refiere el acceso a cierta URL no es el mismo sitio web, se bloquea el acceso. Sin embargo esto puede ser derrotado incluyendo la llamada maliciosa dentro del mismo sitio web ya que las peticiones desde el mismo sitio web no generan un *HTTP-referer* o es el nombre del mismo sitio web y por tanto aparentemente válido.

16 IP: Protocolo de Internet, del Inglés Internet Protocol

17 HTTP Referer: permite determinar desde qué lugar se refirió el acceso que se está realizando. Por tanto permite saber de dónde venía esta petición.

Sin embargo muchos sistemas de protección del usuario: plugins, antivirus, etc. Deshabilitan el *HTTP-referer* por cuestiones de privacidad, por lo que se ha planteado el uso de un nuevo header: *Origin:* para poder determinar si la petición viene de un origen validado o no [35].

2.4.9 Uso de componentes con fallas conocidas [36]

Muchos desarrolladores utilizan código o componentes para sus aplicaciones que ya vienen con fallas de seguridad conocidas. Estas fallas son fácilmente aprovechadas por los atacantes para entrar a los sitios web. Pueden se fallas que ya hemos descrito en el documento como no verificar o protegerse de CSRF, fallas de inyección, claves conocidas, fallas de configuración etc.

Posible solución desde el servidor web

Este es un tema muy genérico y referido a acciones incorrectas que toman los programadores, estos problemas, por su diversidad, deben trabajarse realmente de acuerdo al tipo de falla aprovechada.

2.4.10 Redirecciones no validadas

Aunque la redirección hacia otros sitios o páginas es algo normal en la programación para la web, en algunas ocasiones la aplicación recibe la URL de destino a través de un parámetro que no es correctamente validado[37]. Un atacante puede cambiar la página web de destino a través de esta falla. El usuario puede ser redirigido a un sitio que puede convencer al usuario de entregar su información.

Posible solución desde el servidor web

No existe. Al igual que el caso anterior es un problema genérico de concepto. Las sugerencias van dirigidas a las formas en que se hacen redireccionamiento por programación.

2.5 Otras técnicas utilizadas

En el pasado se mencionaron otras técnicas que han sido consolidadas en una de las anteriores o que su impacto ha disminuido bastante como son:

- Ejecución de Archivos Maliciosos
- Filtrado de información y manejo impropio de errores

A continuación realizamos una explicación de ellas:

2.5.1 Ejecución de archivos maliciosos

Algunos lenguajes de programación, como PHP[18] por ejemplo, permiten que incluir referencias a otras partes del sitio web o a otro sitio web a través de variables. Esto de no ser adecuadamente validado puede ser utilizado para afectar seriamente el funcionamiento del sitio por parte de un atacante[38]. Es un ataque muy popular cuando se usa PHP.

Posible solución desde el servidor web

En caso de estarse utilizando PHP, en el archivo de configuración desactivar el uso de los URL include.

2.5.2 Filtrado de información y manejo impropio de errores

Aunque existen problemas mucho más graves, en efecto existen aplicaciones que muestran información sobre los errores que tienen o cuánto tiempo les tomó ejecutar una acción. Lo que permite a un atacante planificar o verificar el impacto que ha tenido su intrusión en el servidor[39].

Posible solución desde el servidor web

Deshabilitar opciones que permiten mostrar errores en el navegador como por ejemplo display_errors en el archivo de configuración de PHP. Deshabilitar o minimizar la cantidad de información que brinda el servidor sobre sí mismo.

2.6 Análisis de la información obtenida

Como podemos observar, algunos riesgos han sido consolidadas en uno sola a medida que progresan los reportes, como es el caso de *"Exposición de datos sensitivos"* que en el reporte del 2007 se representaba en 3 riesgos diferentes, ya en el 2010 se resumieron en 2 y en el 2013 se consolidaron en uno.

También notamos que algunos han desaparecido, esto no es porque hayan sido totalmente corregidos como es el caso de *"ejecución de archivos maliciosos"* sino porque han surgido otras amenazas más importantes o sus características han sido integradas dentro de otra categoría.

Sin embargo debemos notar que algunos de ellos se repiten en los documentos expuestos en la tabla 2 como son:

- Fallas de inyección
- Fallas en el manejo de sesión y autenticación
- XSS

18 PHP: PHP Hypertext Pre-processor, lenguaje de programación

- Referencias inseguras directas a objetos
- Fallas en la configuración
- Exposición de datos sensitivos
- Falta de control de acceso a funciones
- CSRF
- Redirecciones no validadas

Aunque todas las posibles causas de problemas son solucionables y deben ser solucionadas utilizando correctas prácticas de programación [40], administración o de uso del sistema; muchos de los riesgos pueden, en algunos casos, ser mitigados o bloqueados en el mismo servidor web en el caso de algunas de las antes descritas como podemos observar en la tabla:

Riesgo	Propuesta de solución
Fallas de inyección	• Evitar ejecución de programas a través de la URL • Sanitizar los contenidos enviados a través de formularios evitando que se introduzcan encabezados SMTP para enviar spam • Detectar intentos de enviar comandos SQL a través del servidor web.
XSS	• Analizar contenidos de las variables que llegan al servidor pueden permitir encontrar y bloquear caracteres no permitidos como < > “ y ′
Fallas en la configuración	• Fortalecimiento del servidor web • Fortalecimiento del Sistema Operativo donde está corriendo el servidor web.
Exposición de datos sensitivos	• Adecuada configuración del servicio SSL para prevenir usar protocolos que presentan fallas de seguridad • Uso de técnicas de aleatorización de la información que se envía por el canal
CSRF	• Evitar entradas a través de referers de otros sitios. • Evitar entradas que provengan de otro origen declarado en el header “Origin:” [41]

Tabla 3: Propuesta de solución según tipo de riesgo

2.7 Aprovechamiento de las vulnerabilidades

Los tipos de ataques antes descritos aprovechan falencias que ocurren durante la programación, administración o uso de una aplicación dinámica, desde inicios de la internet se han intentado valorar las formas que aprovechan los atacantes para penetrar en los sistemas [42]

En mi opinión estas podemos catalogarlas según desde genera la vulnerabilidad en:

- Falencias en la operación por parte del usuario
- Falencias en la administración del sitio o el servidor
- Falencias en la programación de la aplicación
- Falencias en el servidor web

2.7.1 Falencias durante la operación por parte del usuario

Es uno de los procesos más difíciles de manejar porque se deben típicamente a problemas que ocurren del lado del usuario final. Dependen mucho de la configuración que el usuario tenga en sus navegadores o equipos así como de cuán educado o cuidadoso sea respecto al uso de internet.

2.7.2 Fallas en el navegador

Son las fallas debido a un navegador desactualizado o que no implementa medidas adecuadas de seguridad con la finalidad de proteger al usuario. Es un problema real pues muchos navegadores implementan efectivamente medidas genéricas de protección pero no vienen particularmente fortalecidos para resistir a diversos tipos de ataques populares, esta es la razón por la cual vemos continuamente equipos de usuario contaminados a través de los navegadores.

2.7.3 Falencias en la educación del usuario para el uso de las herramientas

Son las fallas que se deben a un usuario sin adecuada preparación para acceder a internet que le permite al atacante realizar labores de ingeniería social, confundiendo al usuario y obteniendo información valiosa de este.

Por otro lado: muy poca labor se hace respecto a utilizar los medios de comunicación para informar y educar a los usuarios en un adecuado uso de su sistema.

Siendo el usuario el principal objetivo de estos ataques, consideramos que es muy importante trabajar en la educación al usuario.

2.7.4 Falencias en la administración del servidor

Efectivamente desde el lado del usuario no tenemos control respecto a las seguridades que estos implementen, sin embargo ya desde el punto de vista de la administración del servidor donde está alojado el sitio dinámico sí es posible realizar una labor de protección de este ya que típicamente se centra en un universo finito y pequeño de equipamiento a controlar, normalmente el sitio está en un servidor aún cuando para sitios grandes o muy grandes pueda extenderse a un número mayor de equipos, sin embargo al momento son una cantidad finita y conocida de equipos con los que se cuenta.

Aún así, es frecuente que los atacantes aprovechen fallas en la administración del servidor para acceder a la información presente en los sitios dinámicos y/o modificarla. Para ello se basan en problemas en la administración como son:

2.7.5 Inadecuado manejo de credenciales de acceso al servidor

Ocurre cuando no se toman medidas de seguridad respecto al manejo y uso de credenciales de acceso al servidor. Es un problema bien conocido y que al día de hoy no ha podido ser eliminado, por ejemplo:

- Uso de claves débiles
- Guardar o escribir las claves en la computadora de escritorio o en un papel
- Compartir la misma clave entre varios equipos
- Compartir la misma clave entre varias personas

2.7.6 Inadecuado manejo de credenciales de acceso a sistemas de apoyo

Al igual que el acceso al servidor, existen diversos sistemas que corren en él que sirven de apoyo o de base para el funcionamiento de las aplicaciones dinámicas como pueden ser: sistemas de bases de datos, sistema de manejo de bases de datos, acceso remoto al manejo de la aplicación dinámica o sistema de configuración del servidor web, etc.

Estos sistemas además de sufrir de los mismos problemas de manejo de credenciales expuesto en el punto anterior, tienen un grupo de problemas propios como son:

- Uso de contraseñas vacías: Problema frecuente en sistemas de bases de datos a los cuales por alguna razón les configuran a los usuarios administradores sin contraseñas

- Uso de contraseñas por defecto: Esto se evidencia normalmente en la configuración de servidores de aplicaciones los cuales se instalan y dejan configurados con la misma clave de instalación

2.7.7 Uso de aplicaciones configuradas por defecto

En los sistemas de apoyo a la administración como sistemas de manejos de bases de datos, de configuración de servidores de aplicaciones, de monitoreo de uso de recursos, cuyo acceso debería estar restringido a la red local de los administradores o deberían estar apagados. Muchas veces se dejan los accesos totalmente abiertos de forma tal que los atacantes pueden acceder y hacer uso de ellos.

Además muchas veces se utilizan sistemas de manejo de contenidos configurados por defecto o muy poco configurados que permiten al atacante tomar control de fallas conocidas en estos.

2.7.8 Servidor desactualizado

Un servidor desactualizado puede ser objeto de ataques a fallas de seguridad conocidas. Estas son las fallas que son públicas y de las cuales seguramente el fabricante ha publicado su corrección pero el personal a cargo de la administración de los equipos no realizan actualizaciones frecuentes por lo que el equipo se ve expuesto a ataques a través de estas fallas conocidas.

2.8 Falencias en el servidor web

Muy parecido al caso anterior, pero orientado específicamente a los problemas que pueden presentar el servidor web que aloja la aplicación dinámica. Existen los mismos problemas anteriores pero en este caso se ven exacerbados por el hecho de que el servidor web normalmente es la arena donde se ejecutan las aplicaciones y existe una cantidad particularmente importante de problemas que pueden ser aprovechados remotamente.

2.8.1 Fallas en la programación del servidor o sus componentes

Es un problema muy común que puede deberse a fallas en la programación o en el concepto de cómo debería funcionar el servidor. Una breve búsqueda en la Base de datos de vulnerabilidades del Departamento de seguridad nacional de los Estados Unidos arroja más de un centenar de problemas reportados en los últimos 15 años relacionados con la palabra “webserver” [43].

2.8.2 Configuración inapropiada

Típicamente se utiliza el servidor configurado por defecto y no se hace uso de mecanismos de protección como enjaulamiento del sitio o uso de un usuario específico para ejecutar los scripts del sitio para evitar que un ataque a éste se propague al resto de sitios del servidor web o del servidor.

También existen problemas con la configuración de componentes del sitio como es el caso del SSL que viene con sistemas de cifrado débiles que no deberían estar activados[44], u otras características de la configuración que podrían ser deshabilitadas o reconfiguradas para mejorar la resistencia ante ataques.

2.8.3 Fallas de desempeño

Son fallas de seguridad que pueden provocar que el servidor deje de estar disponible para servir peticiones. De acuerdo a la triada CIA[19] de la seguridad, la disponibilidad es un elemento fundamental en la seguridad informática [45] por lo tanto el obtener un adecuado desempeño permite incrementar la disponibilidad del sistema.

2.9 Falencias en la programación de la aplicación [40]

Lamentablemente muchas personas a cargo de programar aplicaciones no han recibido adecuada capacitación sobre cómo realizar programación segura.

Este es uno de los puntos cruciales a la hora de aprovechar fallas de seguridad. De hecho podríamos afirmar que de realizarse una programación segura la casi totalidad de los problemas de seguridad de las aplicaciones dinámicas se podría solucionar, con claras excepciones como es el hecho de que un usuario sucumba ante un ataque de ingeniería social por ejemplo.

2.9.1 Código no seguro o no auditado

Se tiende a confiar en el código generado para la aplicación sin haber realizado una auditoría a este. Muchas veces se utiliza código cerrado el cual no puede ser revisado.

2.9.2 Utilización de código desactualizado

En vista de la cantidad de código existente actualmente, ocurre que programadores utilicen código de terceros que está desactualizado y puede tener fallas de seguridad conocidas.

19 CIA: Principios básicos de la seguridad Confidencialidad, Integridad y Disponibilidad...

2.10 Efectos de los ataques a aplicaciones dinámicas

Los ataques a las aplicaciones pueden tener diversos efectos en la funcionalidad del sitio e incluso en la estabilidad de la empresa.

Muchas veces estos efectos pueden venir combinados entre sí, lográndose un mejor efecto por parte de los atacantes.

2.10.1 Negación de servicio

Conocidos como DoS[20]. Se le conoce a los ataques cuyo objeto es afectar la disponibilidad del sitio web de una organización. Estos ataques son conocidos y utilizados continuamente [46] y consisten en una afluencia masiva de peticiones hacia el servidor, aunque también pueden aprovechar vulnerabilidades en la administración, programación o sistemas utilizados para ofrecer el servicio que impidan su normal funcionamiento.

2.10.2 Desfiguración

La desfiguración ocurre cuando un atacante aprovecha una o varias fallas de seguridad con el objetivo de modificar los contenidos de un sitio web. Es un tipo de ataque muy popular y utilizado en estos tiempos como forma de protesta a través de la internet, y es frecuente ver sitios del país que han sido desfigurados [5]. La desfiguración afecta fundamentalmente el prestigio del sitio web pues es un claro indicador de que la seguridad de este sitio debe ser mejorada, incluso a veces exponen precisamente las fallas que un sitio tiene.

Existe un sitio web dedicado a publicar y almacenar las desfiguraciones que han ocurrido. El sitio conocido como zone-h.org [47].

2.10.3 Accesos no autorizados

Es un ataque dañino a la persona o empresa objeto de este ataque. Consiste en obtener información de las credenciales de acceso a un determinado servicio ya sea para suplantar la identidad de una persona o empresa o para reunir información sobre los usuarios obtenida a través de este ataque. Un ejemplo de esto fue el del acceso no autorizado a la cuenta de DatoSeguro del Presidente de la República [48][49]

20 DoS: Negación de Servicio, del Inglés Denial of Service

2.10.4 Robo de información

Es muy similar a los accesos no autorizados, en efecto el objetivo no es solamente suplantar identidad u obtener información sobre los usuarios, sino que también incluye el robo de información con el objetivo de obtener ventajas industriales, políticas, personales, etc. Este robo de información ha sido muy popular en diversos sectores como por ejemplo el caso del grupo hacktivista[21] Anonymous[22] para demostrar las debilidades de los sistemas informáticos [50].

21 Hacktivismo: Utilización de herramientas de Internet para realizar activismo político

22 Anonymous, en Español anónimo. Grupo de personas que realizan hacktivismo

Capítulo 3 : Estado del arte de la reacción ante ataques a servidores web

El servidor web es el demonio que sirve las solicitudes hechas por los usuarios, este proceso se realiza leyendo los contenidos de los archivos solicitados, interpretando o ejecutando estos contenidos de ser necesario y enviando al usuario los resultados.

Muchas veces el servidor web es objeto de ataques con la finalidad de comprometer la disponibilidad, confidencialidad o integridad de la información contenida en él y en vista de ello al día de hoy se crean técnicas para mitigar o eliminar la amenaza que puede representar un ataque a un servidor web.

A continuación agrupamos y describimos cómo se pueden tomar medidas de protección ante ataques a servidores.

3.1 Protecciones a nivel de usuario

Las herramientas que utiliza el usuario final juegan un rol importante a la hora de trabajar en la protección contra fallas que pueden ser explotadas en sitio web. Normalmente se centran en las siguientes herramientas

3.1.1 Navegadores con características de seguridad

Se ha realizado y se realiza una continua labor con la finalidad de que los navegadores más populares detecten , eviten y/o alerten cualquier intento de actividad inusual.

Para ello los desarrolladores de los navegadores han llegado a acuerdos de estándares que permiten un continuo mejoramiento del proceso de mostrar páginas web como por ejemplo:

- La implementación del header Origin: [41], implementación de mecanismos del mismo origen que permiten la ejecución de scripts desde el mismo dominio que se está navegando. [51]
- Separación de características de los navegadores en plugins. De forma tal que no se contengan de forma monolítica estas dentro del navegador que hagan que una falla en ella afecten a toda la población que haga uso de ellas.
- Posibilidad de desactivar la opción de ejecución de scripts con la finalidad de evitar que se puedan realizar ataques de XSS y CSRF a través de ellos.

- Uso de entorno de pruebas para correr contenidos con privilegios restringidos en su sitio web, de esta forma se evita que ciertas características de la web esté activadas para contenidos no confiables.[52]

3.1.2 Plugins de seguridad en navegadores

Con la aparición de plugins para los navegadores, inicialmente en Firefox y luego extendido a otros, se abren nuevas oportunidades de desarrollo independiente de plugins que permiten mejorar la seguridad de la navegación como por ejemplo:

- Adblock plus: permite bloquear contenidos que no son deseados por los usuarios, entre ellos bloqueo de objetos de java, flash e incluso bloqueo de sitios considerados maliciosos.
- Cookie monster: solamente permite guardar cookies de sitios autorizados por el usuario.
- Https-everywhere: Es un Plug-in que negocia y trata de conectarse al sitio web de destino utilizando https aún cuando el usuario o el enlace haya sido por http. El objetivo es tratar de utilizar conexiones encriptadas el mayor tiempo posible.
- Webutation: Plug-in que muestra la reputación del sitio web al que uno se conecta
- NoScript Security: un plugin que deshabilita todo tipo de ejecución de scripts en sitios web, excepto que el usuario solicite lo contrario. De esta forma se evitan la casi totalidad de ataques XSS y CSRF. Este plugin es sumamente interesante pero a veces molesto pues hay sitios que para realizar operaciones como por ejemplo compras en internet, requieren que se desactive temporalmente este plugin.

3.1.3 Sistemas de detección de amenazas

3.1.3.1 Antivirus

A través del uso de antivirus se pueden detectar y evadir amenazas presentadas en sitios web navegados si se descubre que un sitio que contenga una amenaza (contenido malicioso) se intenta acceder por parte de un sitio web que el usuario esté abriendo.

Los antivirus por otra parte presentan un costo que impacta al usuario en cuanto a que consumen recursos no solamente financieros, sino además en uso de procesador, memoria ram y almacenamiento en disco.

Muchos antivirus se están transformando en suites completas de manejo de amenazas que incluyen verificación contra contenido de phishing, cortafuegos y evasión de intentos de intrusión a través de la web.

3.1.3.2 Educación en el uso de herramientas de internet

En nuestra opinión, una correcta educación en el manejo de herramientas para el acceso a Internet, privacidad y seguridad en la red, son necesarias para lograr mejorar dramáticamente la seguridad informática. Independientemente de la cantidad de sistemas que realizen labores de seguridad automática, la falta de educación en estos aspectos es fuente de muchos de los problemas de seguridad en la red. En el sitio CERT.org podemos acceder a una serie de recomendaciones de seguridad en el uso del navegador [53]

3.2 Protecciones en la programación

Una programación orientada a la seguridad es un problema que no es nuevo, desde hace mucho tiempo autores de diversas épocas intentan orientar a los programadores en técnicas de programación segura. La mayoría de estos documentos [54][55] hacen énfasis en cuestiones tales como:

- Validación de la entrada de datos
- Manejo de sesión
- Acceso a Bases de datos
- Validación de la salida de datos

De todos ellos, Validación de la entrada de datos es una gran fuente de problemas pues a través de fallas en estas validaciones los atacantes pueden insertar código malicioso en las aplicaciones web.

Existen sistemas de ayuda al programador para evitar estos problemas, como es el uso de bibliotecas ya conocidas para realizar las tareas de la aplicación evitando reescribir lo que ya existe. Actualmente se prefiere el uso de Frameworks, que no es más que un conjunto de bibliotecas mantenidas por un autor para realizar esta labor. A través de frameworks orientados a la seguridad es posible lograr ciertas mejoras en este aspecto, sin embargo consideramos que una adecuada educación orientada a la seguridad no puede ser sustituida por ninguna otra alternativa automática.

3.3 Protecciones en el Sistema Operativo

El proceso de asegurar el servidor comienza precisamente por realizar una correcta instalación del Sistema Operativo donde estará alojado el sitio dinámico pues de un atacante aprovechar una vulnerabilidad en este, podría afectar el funcionamiento del sitio web.

Existe variada información sobre las formas de asegurar el Sistema Operativo, desde revisiones de seguridad de Nessus.

Para ello debemos seguir las siguientes pautas con la finalidad de fortalecer el Sistema Operativo:

3.3.1 Tiempo de soporte

Cada desarrollador de una distribución Linux tiene políticas respecto al periodo durante el cual soportan actualizaciones a su creación. El periodo de soporte brinda al usuario de una distribución la confianza que durante este tiempo no será requerido reinstalar el sistema, y por tanto reconfigurarlo y asegurarlo, sino que con simples comandos se puedan manejar actualizaciones al sistema instalado.

CentOS es una distribución Linux conocida en el mercado por ser un clon 100% compatible con su distribución origen que es RHEL[23]; y estas dos distribuciones tienen un periodo de soporte de 10 años [56] siendo al momento las distribuciones Linux que reciben actualizaciones y soporte por mayor tiempo.

3.3.2 Instalaciones mínimas

En dependencia de la distribución Linux, existen diversas formas de instalación, sin embargo la sugerencia de seguridad siempre va orientada a realizar una instalación mínima.

Este es el tipo de instalación que solamente instala la base del Sistema Operativo, lo necesario para que este sistema funcione y pueda ser accedido. En base a una instalación mínima se puede crecer instalando el Software que se requiera ya que es Software Libre y podemos instalarle cuando se requiera.

Con este tipo de instalación podemos garantizar algo fundamental en la seguridad y es que no dejaríamos huecos de seguridad producto de sistemas que no se requieran para el funcionamiento de nuestro servidor.

En el caso de CentOS existe incluso una figura de instalación conocida como *CentOS-Minimal* que permite instalar de forma muy rápida la base del sistema ya sea en arquitecturas de 32bits como de 64bits.

Este instalador solamente contiene los paquetes básicos para que el sistema funcione, garantizando de esta forma que no queden instalados paquetes que no se requerirán para el funcionamiento.

23 RHEL: Distribución de GNU/Linux de la empresa RedHat Inc.

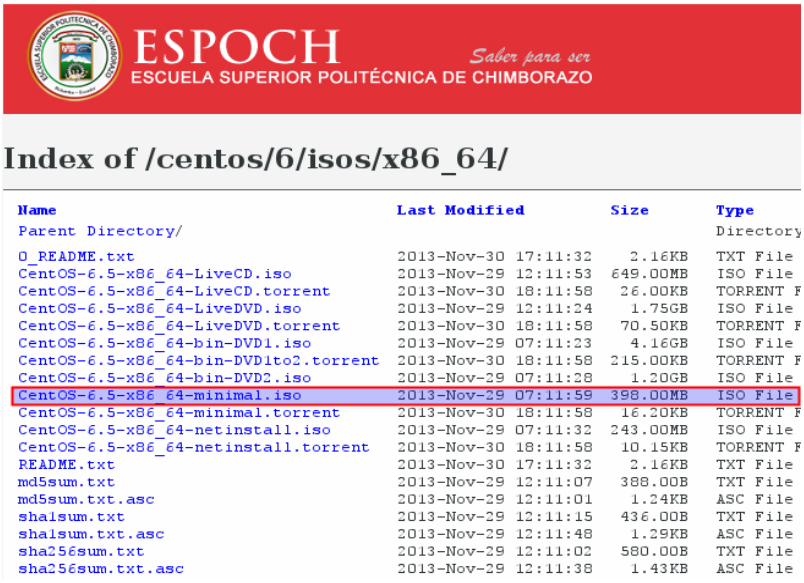


Figura 3: CentOS minimal [A]

3.4 Adecuado particionamiento del disco

Los sistemas instalados por defecto vienen normalmente con 3 particiones que son montadas en os directorios */boot*, */* y una tercera para servir de *espacio de intercambio (SWAP).*

Sin embargo al realizar este tipo de instalación se pierden una serie de posibilidades que se pueden aprovechar para asegurar las particiones. Estas son conocidas como opciones que se aplican a las particiones:

Opción	Descripción
nodev	Esta opción impide que *nodos de dispositivos* (device nodes) sean creados o accedidos en esta partición. De esta forma un atacante no podrá aprovechar accesos de escritura hacia un directorio de la partición para crear nodos de dispositivos y a través de ellos acceder a diversos dispositivos del sistema operativo como por ejemplo memoria. El único lugar donde deben existir nodos de dispositivos es en el directorio **/dev**
nosuid	Se le indica al sistema operativo que ignore el bit setuid el cual permite que un programa se ejecute con un usuario diferente al que fue iniciado. Este bit es importante en programas tales como *passwd* o *sudo* con la finalidad de que estos programas puedan ejecutarse como superusuario del sistema para realizar tareas requeridas. Sin embargo un atacante puede introducir un programa suid en una partición que le garantice

Opción	Descripción
	posteriormente acceso irrestricto como superusuario. Este tipo de programas solamente debe estar presente en los directorios /bin y /sbin del sistema que son además de sólo lectura.
noexec	La bandera *noexec* impide la ejecución de un programa en una partición. De forma tal que aún cuando un atacante introduzca un ejecutable en el sistema, si lo hace en una partición con *noexec*, la ejecución de este estará prohibida. Los archivos ejecutables deben ir solamente en los directorios /bin y /sbin del sistema que son además de sólo lectura.

Tabla 4: Opciones que se pueden aplicar al sistema de archivos

Los directorios donde normalmente se pueden producir escrituras por parte de atacantes de un sitio web son /tmp, /var/tmp, /home y el mismo directorio donde se aloja el sitio web que está típicamente en /var/www

En vista de esto es recomendado que se creen al menos las siguientes particiones:

Directorio	Justificación	Opciones sugeridas
/boot	Esta partición es requerida para instalar los núcleos (kernel) del sistema dentro de los primeros 1024 cilindros. Es estándar para cualquier instalación y nada más que los núcleos del sistema deben estar.	noexec, nosuid, nodev (no es obligatorio)
/	Es la partición raíz del sistema. Todo lo que no esté en su propia partición, será escrito en esta.	
SWAP	Es una partición especial que almacena el espacio de intercambio. Es estándar para cualquier instalación.	
/tmp	Directorio al cual todos los procesos tienen acceso de escritura con la finalidad de guardar información temporal. Este directorio puede ser utilizado con fines maliciosos como crear dispositivos, bajar programas para ejecutar actividades maliciosas o con el bit suid activado.	noexec,nosuid,nodev
/var	• Contiene el directorio /var/tmp este directorio puede ser utilizado con fines maliciosos como crear dispositivos, bajar programas para ejecutar	noexec,nosuid,nodev

Directorio	Justificación	Opciones sugeridas
	actividades maliciosas o con el bit suid activado. • Contiene la página web que está normalmente bajo el mismo usuario que corre el servidor web, el atacante puede aprovechar una vulnerabilidad en el servidor web para descargar hacia este directorio programas o intentar crear dispositivos para realizar actividades maliciosas.	
/home	Cada usuario del sistema tiene normalmente un directorio en /home que es usado para almacenar documentos e información propia al usuario. No es necesario que los usuarios tengan derechos de ejecución, de crear dispositivos, o de ejecutar programas con el bit de suid activo	noexec,nosuid,nodev

Tabla 5: Sugerencia de particionamiento en Linux CentOS [A]

Estos bits se activan en el archivo /etc/fstab como podemos observar en la siguiente figura:

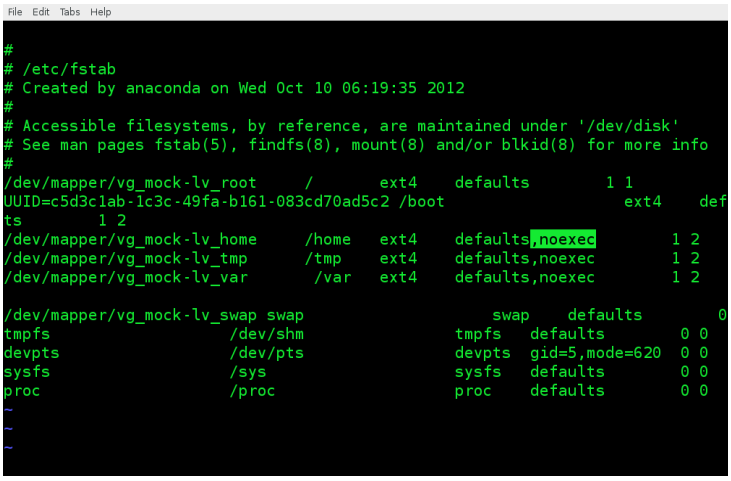


Figura 4: Configuración de noexec en /etc/fstab [A]

3.4.1.1 Caso especial: /dev/shm

/dev/shm no es una partición en disco, es lo que se conoce como un sistema de archivos virtual y es usado con la finalidad de almacenar datos temporalmente en este directorio que, por ser virtual, está montado en la RAM. Se le puede aplicar las mismas opciones que se ponen en /tmp con la finalidad de impedir los mismos actos maliciosos que se pueden utilizar contra /tmp

Figura 5: Configuración de noexec y nosuid para /dev/shm [A]

3.4.2 Eliminando servicios innecesarios

Aún cuando se realiza una instalación mínima, o en caso de que no se haya hecho, pueden existir servicios que están corriendo y no sean requeridos para el normal funcionamiento del servidor. Estos servicios inciden en ciertos aspectos del funcionamiento del servidor:

1. Consumo de memoria: cada servicio consume una cantidad de memoria RAM que podría dedicarse a otras actividades propias del servidor web
2. Consumo de procesador: este puede ser un recurso escaso y necesario para otras labores del servidor.
3. Tiempo de arranque: mientras más servicios tengo activados, mayor será el tiempo de arranque de mi servidor y por tanto mayor la espera para servir las peticiones de los usuarios.
4. Puertos abiertos: los servidores que son un caso especial de demonios, servicios que están corriendo continuamente en el servidor, pueden exponer uno o varios puertos a internet, y a través de estos puertos los atacantes pueden aprovechar vulnerabilidades para entrar al servidor.

Es por ello que se hace imperioso el eliminar o apagar servicios que no se estén utilizando. Para ello se debe realizar un análisis de los servicios que están corriendo en el servidor y validar si al apagarlos no se pierden funcionalidades necesarias para el funcionamiento del sitio web.

La forma recomendada de dejar el mínimo de servicios en el servidor es realizando una instalación mínima, puesto que en este caso no existirán demasiados servicios corriendo en él. Y en base a esta instalación mínima, revisar los servicios que estén corriendo y determinar si pueden apagarse o no.

Esto demuestra que no es recomendado instalar un servidor Linux de otra forma que no sea la mínima, ya que de hacerse lo que se conoce como instalaciones completas, estas pueden venir con infinidad de servicios innecesarios que pueden ser explotados por los atacantes.

3.4.3 Actualizaciones

Las fallas de seguridad también pueden clasificarse según si ellas son conocidas o desconocidas:

- Fallas de seguridad desconocidas: son fallas de seguridad que el fabricante de un sistema no conoce que existen en sus sistemas y por tanto están presentes en todos sus sistemas instalados.
- Fallas de seguridad conocidas: son las más típicas fallas, lógicamente antes de ser una falla conocida, durante un tiempo fue una falla de seguridad desconocida, sin embargo cuando el fabricante es informado o encuentra él mismo una falla de seguridad, la soluciona y publica una actualización que puede ser incorporada a los sistemas de los usuarios.

De mantenerse una correcta política de actualización al sistema, las fallas de seguridad conocidas pueden ser corregidas y por tanto no aprovechadas por un atacante.

Un caso concreto de falla de seguridad es el Heartbleed [32], una falla que fue descubierta en el mes de Marzo del 2014 que afecta a diversas versiones del paquete OpenSSL[24], utilizado para encriptar las comunicaciones que ocurren a través de sistemas como https en los servidores HTTPD de Apache[25], nginx[26], lighttpd[27]; así como el paquete de OpenVPN[28]. Esta falla fue corregida en ese mismo mes y fue emitido un aviso a los fabricantes que de forma bastante expedita parchearon y publicaron actualizaciones a sus versiones de OpenSSL. A pesar de ser una falla gravísima, que permite obtener la clave privada de la conexión segura y por tanto desencriptar el tráfico o acceder a credenciales de los usuarios, todavía hoy, a finales de Abril del 2014, un mes después de descubierta la falla, diversos servidores de varias redes no han tomado medidas para actualizar su versión de OpenSSL como podemos observar en el Anexo I.

El resultado de un análisis empírico de estos resultados podemos obtener que de forma pública, de 26 redes analizadas, 14 presentaban la falla de seguridad conocida “Heartbleed”[32] lo que demuestra que al menos el 53.8% de los administradores de estas redes no se preocupan por mantener actualizado su sistema.

Esto demuestra que las actualizaciones deben ser continuas y por tanto deben utilizarse Sistemas Operativos de fabricantes que ofrezcan la seguridad de que al actualizarle este seguirá funcionando como antes.

24 OpenSSL: Herramienta en Software Libre que provee implementaciones de los protocolos SSL y TLS

25 Apache: Servidor web mantenido por la fundación Apache

26 Nginx: Servidor web escrito por Igor Sysoev

27 Lighttpd: Servidor web escrito por Jan Kneschke

28 OpenVPN: Servidor de VPN mantenido por OpenVPN Technologies

3.4.4 Fortalecimiento al núcleo del sistema

Otro tipo de ataque muy tradicional es el de aprovechar debilidades en la configuración del núcleo del sistema para realizar un ataque de negación de servicio por agotamiento de recursos. Existen manuales y sistemas que revisan el uso de buenas prácticas [57] para evitar este tipo de ataques a través de una adecuada configuración de diversas variables del kernel con la finalidad de fortalecerle ante ataques de negación de servicio.

3.4.5 Aseguramiento de SSH

En la mayoría de los casos siempre es deseable tener el servicio de ssh activo en nuestro servidor. A través de SSH podemos conectarnos a nuestro servidor y manejarle a cualquier hora del día y desde cualquier lugar.

Sin embargo todo sistema que nos facilita el trabajo a nosotros, también se lo facilitará a los atacantes. Es decir: un atacante podría aprovechar el SSH para acceder a nuestro servidor remotamente.

Es por ello que es importante asegurar el servicio de SSH en el servidor de forma tal que le sea difícil a un potencial atacante entrar al servidor, pero que nosotros no dejemos de utilizar este servicio. Existen manuales y sistemas que ayudan a tener buenas prácticas en el manejo del sistema de autenticación por SSH[57].

3.4.5.1 Denegando intentos fallidos de autenticación

El paquete *denyhosts* revisa el archivo */var/log/secure* y bloquea IPs que estén siendo reportadas con intentos de autenticación fallidos.

3.4.5.2 Impidiendo entradas como root

Si un atacante detecta que en nuestra IP está corriendo un servidor Linux, lo más seguro es que intentará entrar al sistema con el usuario conocido de Linux: "*root*". Para ello es una buena opción bloquear el acceso como root al sistema. Esto se configura poniendo la opción **PermitRootLogin** a **no** en el archivo */etc/ssh/sshd_config*

A partir de ahora el proceso de entrada por ssh será de la siguiente forma:

1. Debemos entrar como un usuario no privilegiado creado con antelación
2. Una vez dentro con este usuario, procedemos a poner: su -, para obtener acceso como root.

De ahora en adelante para un atacante lograr obtener acceso como root deberá:

1. Saber que ya no puede entrar directamente como root

2. Intentar infinidad de veces hasta encontrar un usuario y una clave que no sea root
3. Una vez logre vencer los dos pasos anteriores, ejecutar su - y comenzar a probar claves para acceder al usuario root.

Como podemos observar, a un atacante se le dificulta extraordinariamente el acceso al servidor como root, por lo que muchos se desaniman y se van a buscar presas más fáciles. Es imperioso, es importantísimo el configurar PermitRootLogin como aquí hemos indicado, para evitar que ataques de diccionario contra el usuario root puedan tener un feliz término para el atacante.

3.4.5.3 Permitiendo usuarios específicos

De tenerse un sistema con una gran cantidad de usuarios, se puede definir exactamente cuántos usuarios no privilegiados pueden entrar al sistema. Esto le dificultará más aún al atacante la labor de entrada al servidor ya que no solamente no podrá entrar como root, sino que además tendrá que ver, de toda la posible lista de usuarios cuál está autorizado a entrar por ssh.

La opción *AllowUsers* en /etc/ssh/sshd_config permitirá que solamente la lista de usuarios separados por coma.

3.4.5.4 Shell por defecto para los usuarios

Otra de las fallas típicas que se pueden corregir con poco esfuerzo es el shell que utilizan los usuarios. Por defecto en CentOS los usuarios cuando son creados utilizan de shell */bin/bash*, esto lo podemos detectar si miramos dentro de /etc/passwd, en este ejemplo podemos ver que hay varios usuarios con uid mayor que 500 que utilizan /bin/bash como shell:

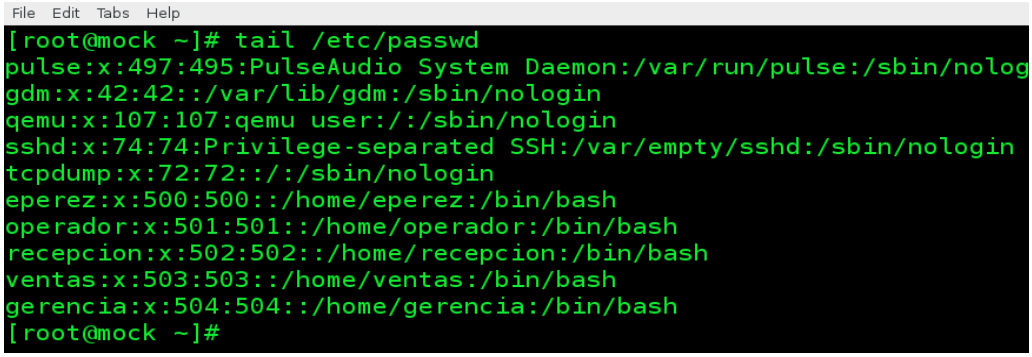


Figura 6: Shell por defecto a los usuarios [A]

/bin/bash es el shell que utilizamos para enviar comandos a Linux.

Visto de otra forma: /bin/bash deberían tener solamente los usuarios que accedan al servidor Linux a emitir comandos, a administrarlo.

Entonces, es importante que a todo usuario que no requiera acceder al shell, se le desactive el uso del shell /bin/bash.

Estos usuarios deben usar como shell uno conocido como: */sbin/nologin;* este no le permite al usuario acceder al shell. Ya que muchos atacantes aprovechan debilidades en las claves de estos usuarios para obtener un shell en nuestro servidor y luego continuar con otros ataques.

3.4.6 Uso de claves fuertes o sistemas de autenticación no tradicionales

Los sistemas de autenticación tradicionales, aquellos que utilizan usuario y clave para garantizar el acceso al sistema están siendo cada vez más vulnerados y es por esto que se han propuesto alternativas de solución a ellos.

Utilización de claves fuertes y cambios frecuentes de claves: Son claves con una cantidad de caracteres mayor y una complejidad superior a las que tradicionalmente se han utilizado. Se estima que una clave que supere los 13 caracteres y que contenga no solamente letras sino además números y signos de puntuación puede dificultar al atacante la tarea de descubrirla. Sin embargo la historia nos demuestra que si antes 8 caracteres eran suficientes, luego 10, ahora 13, seguramente dentro de un tiempo 13 no serán suficientes y se requerirán nuevos tamaños de claves. Además, de forma lógica mientras mayor sea la clave, más compleja y con mayor frecuencia se requiera el cambio de esta, el usuario tenderá a presentar dificultades para recordarla incumpliendo una de las leyes de Kerckhoff de la criptografía clásica [58] que afirma que las claves deben ser fáciles de recordar.

Uso de sistemas de autenticación no tradicionales: Una de las primeras variantes fue el uso de sistemas de autenticación mediante clave pública/privada, el servidor tiene la clave pública del usuario guardara y el usuario puede demostrar que él posee la clave privada por tanto accede a este. Las claves son mucho más grandes que las claves tradicionales.

A esto se puede sumar el hecho del sistema de autenticación en varias etapas. Esta propuesta cataloga las formas de autenticar en 3 tipos fundamentales y solicita que al menos dos formas de estas sean implementadas como sistema de autenticación con la finalidad de mitigar o minimizar el impacto que traería que un atacante logre hacerse de una de las formas aquí descritas:

1. Lo que yo conozco (lo que sé): Es alguna información que yo conozco y con ella puedo acceder al sistema. Se ejemplifica en la autenticación tradicional, mediante usuario y clave. El usuario conoce su username y su clave y por tanto puede entrar.

2. Lo que yo poseo (lo que tengo): Consiste en garantizar la autenticación basada en algún elemento que posea físicamente el usuario. El caso más claro es la autenticación en un cajero de banco en el cual el usuario debe introducir una tarjeta en el cajero para que el proceso comience. También se ejemplifica en el uso de celulares, el usuario entra a un sitio web y el sitio web le envía un SMS a un dispositivo que el usuario posee (el celular).
3. Lo que yo soy: Es una forma de autenticar inherente a la persona. El usuario autentica basado en sus características físicas, químicas, biológicas, etc. Tales como ADN[29], el iris del ojo, contextura, forma de la cara, etc.

El obetivo final de este tipo de autenticación, como hemos mencionado anteriormente, se basa en que el atacante podrá obtener una de estas etapas de autenticación, pero posiblemente no todas. Y de esta forma se mitiga el hecho de que un impostor pueda hacer uso del sistema.

Todas estas medidas aplicadas de forma conjunta y adecuada, implementan barreras de seguridad muchas veces insuperables para los atacantes.

3.5 Protecciones en el servidor web

Una adecuada protección del sistema Operativo base tiene que ir acompañada de medidas de seguridad en el propio servidor web, ya que estos dos elementos son controlados por los administradores. Son los dos elementos en que se puede actuar como administrador con la finalidad de proteger a una aplicación web.

Algunos de los pasos aquí explicados son específicos para cierto tipo de servidor web, ya sea porque los autores de una idea así lo han definido o porque no son requeridos en otros servidores web debido a la forma en que han sido creados.

3.5.1 Manejo de Banners

El paso previo de todo atacante es un reconocimiento de su objetivo, entre las muchas labores a realizar está la identificación de versiones y programas a través de la información que brindan los banners que identifican al servidor.

En dependencia del banner, el atacante se hace una idea de las posibles vulnerabilidades que puede aprovechar y planifica de una forma más organizada su intrusión.

29 ADN: Ácido Desoxirribonucleico

Muchos tenemos claros que el asegurar un sistema a través de oscurecer la información no es la forma más adecuada, sin embargo aquí no estamos hablando de mantener la seguridad únicamente a través de la oscuridad sino que es una de las muchas medidas o características de seguridad que se pueden aprovechar para mejorar la seguridad de un servidor web de aplicaciones dinámicas.

Un servidor web puede exponer no solamente la información referente a él y su versión, sino que algunos incluso proveen información sobre las capacidades adicionales que tienen instaladas tales como lenguajes de programación que pueden interpretar, sistema de SSL que utilizan, etc., además de sus versiones.

En principio debe mostrarse únicamente la información básica referente al servidor web que se está corriendo, esto es: su nombre. Todo lo demás debe ser eliminado.

3.5.2 Eliminación de características innecesarias

Los fabricantes en su necesidad de satisfacer necesidades variadas del mercado proveen sus servidores web con características incorporadas que posiblemente no sean de utilidad para la aplicación web que se instalará en este servicio. Por ello es importante que los proveedores comiencen a adoptar una visión de instalación básica, minimalista, del servidor web e incorporarle características adicionales a medida que se vayan requiriendo.

En el caso del servidor Apache, este por ejemplo viene no solamente con lo necesario para que corra el servicio web, sino que además incorpora una variada cantidad de módulos que le permite implementar diversos servicios como webdav, lenguajes de programación, proxy reversos y directos que normalmente no hacen uso los administradores, pero que están presentes y, en caso de tener vulnerabilidades, estas pueden ser aprovechadas por los atacantes para interferir en el funcionamiento del sistema u obtener información privilegiada de este.

3.5.3 Restricción de acceso a directorios basados en usuario/clave

Debe existir la posibilidad y la preocupación por parte de los administradores del sistema utilizado de restringir el acceso a ciertos directorios que no deben ser vistos en público.

Típicamente estos directorios pueden encontrarse realizando ataques a través de google, técnica conocida como Google Hacking, pues estos directorios vienen sin archivo índice y el servidor web muestra un índice por defecto que puede ser fácilmente encontrado.

3.5.4 Protecciones contra ataques de negación de servicio

La negación de servicio por agotamiento de recursos es un caso muy popular entre servidores web, existe una herramienta que permite realizar este tipo de ataque a servidores web conocida como Slowloris a la que muchas versiones de Apache sucumben.

Este tipo de ataque ya fue descrito en el 2005 por Ivan Ristic [59] y consiste en abrir peticiones http parciales, y en el caso de slowloris enviar otros encabezados a intervalos regulares con el objetivo de evitar que se cierre la conexión mientras sigue abriendo más conexiones contra el servidor remoto agotando todos los recursos de este.

Existe un estudio de Chris Wadge sobre la resistencia de diversos servidores web a este tipo de ataque [60]

3.5.5 Bloqueos ante actividad inusual

En el evento de que actividad inusual sea detectada por parte de una IP hacia el servidor web este debe implementar medidas que le permitan bloquear al abusador por un determinado periodo de tiempo.

Si el atacante intenta otro tipo de actividad posteriormente, el periodo de bloqueo se incrementa. Solamente cuando deje de tener actividad contra el servidor éste desbloqueará la IP evitando que un falso positivo pueda ser castigado indefinidamente.

3.5.6 Bloqueos por tipo de navegador

Los servidores web conocen qué tipo de navegador está siendo usado para acceder a un sitio web pues los navegadores de los clientes reportan su nombre y versión.

Muchas veces los programas maliciosos automáticos, conocidos como bots o robots, utilizan herramientas del shell de Linux para automatizar sus ataques, herramientas conocidas tales como curl, wget o incluso perl a través de librerías como www-perl.

Un servidor web puede ser configurado para bloquear estos tipos de navegadores no estándares de forma tal que un ataque que provenga de ellos pueda ser evitado.

Sin embargo en la actualidad falsear el nombre de navegador para hacerse pasar por un navegador conocido es muy fácil por lo que este tipo de restricción es muy débil.

3.5.7 Sistemas de detección de vulnerabilidades

Como parte de la labor de aseguramiento de los servidores, existen herramientas que permiten evaluar la presencia de vulnerabilidades en el servidor y de esta forma poder corregirlas antes de que puedan ser explotadas. Por mostrar variedad de implementaciones podemos mencionar dos de ellas:

3.5.7.1 Nikto

Es una herramienta que hace un escaneo de vulnerabilidades de un servidor web desde el punto de vista de un atacante. Esto es: revisando a través del puerto donde corre el servidor web diversos parámetros y aplicaciones presentes en el servidor. Emite un reporte con los hallazgos realizados y sugerencias sobre cómo arreglarles.

Nikto se basa en una base de datos de posibles vulnerabilidades que puede ser actualizada por internet

```
[eperez@laptop ~]$ sudo nikto -update
+ Retrieving 'nikto_report_csv.plugin'
+ Retrieving 'nikto_cookies.plugin'
+ Retrieving 'db_parked_strings'
+ Retrieving 'nikto_headers.plugin'
+ Retrieving 'db_tests'
+ Retrieving 'CHANGES.txt'
+ CIRT.net message: Please submit Nikto bugs to http://trac2.assembla.com/Nik
2/report/2
```

Figura 7: Actualización de base de definiciones de nikto [A]

Al ejecutarlo, en la misma pantalla nikto emite reporte de los hallazgos que realiza:

```
[eperez@laptop ~]$ nikto -h 127.0.0.1
- ***** RFIURL is not defined in nikto.conf--no RFI tests will run *****
- Nikto v2.1.5
---------------------------------------------------------------------------
+ Target IP:          127.0.0.1
+ Target Hostname:    localhost.localdomain
+ Target Port:        80
+ Start Time:         2014-04-26 19:05:39 (GMT-5)
---------------------------------------------------------------------------
+ Server: lighttpd/1.4.35
+ Server leaks inodes via ETags, header found with file /, fields: 0x4141527
+ The anti-clickjacking X-Frame-Options header is not present.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: OPTIONS, GET, HEAD, POST
+ OSVDB-3268: /linux/: Directory indexing found.
+ OSVDB-3092: /linux/: This might be interesting...
+ 4198 items checked: 0 error(s) and 5 item(s) reported on remote host
+ End Time:           2014-04-26 19:05:56 (GMT-5) (17 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Figura 8: Funcionamiento de nikto [A]

3.5.7.2 Nessus

Es una herramienta comercial muy potente, pues no solamente realiza escaneos desde la red sino que además puede ser configurada para ingresar al servidor y realizar una auditoría del sistema operativo y la configuración de diversos paquetes presentes en él, entre ellos el servicio web.

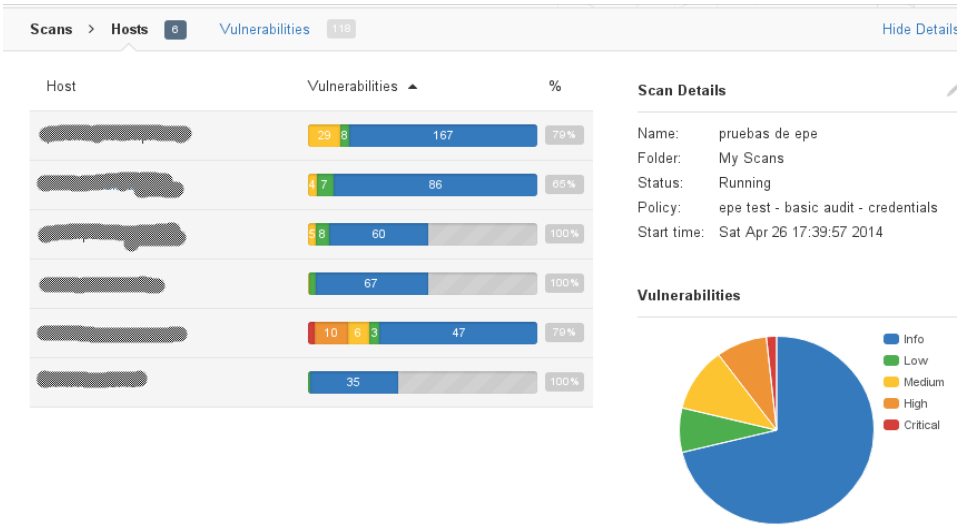


Figura 9: Reporte de escaneo de nessus [A]

3.5.8 Uso de sistemas de monitoreo de cambios

Tan importante como el monitoreo se ha vuelto la implementación de detección de cambios en el sistema, a través de herramientas como el AIDE[30] es posible determinar cambios realizados en el sistema de archivos del servidor para que el administrador obtenga información bastante ágil sobre cualquier intento efectivo de intrusión al servidor.

El AIDE realiza una inspección inicial al sistema de archivos llamada inicialización. Guarda los resultados de la inspección en el mismo sistema de archivos. El administrador puede en cualquier momento realizar un chequeo que consistirá en realizar la misma inspección y compararla contra los resultados de la inspección inicial. De forma tal que se puedan hallar diferencias entre ambos estados.

30 AIDE: Advanced Intrusion Detection Environement: Sistema que almacena las caracteríticas de diversos programas con la finalidad de poderle comparar a un futuro y detectar cambios en estas.

3.5.9 Uso de sistemas de alerta

Similar al caso anterior, tenemos los sistemas de alertas de cambios a sitios web, estos son muy útiles ante desfiguraciones que pueden ser realizadas a sitios web por parte de atacantes. Existe al menos un sistema conocido de monitoreo de alertas ante desfiguraciones conocido como zone-h.org, a través de zone-h.org se puede obtener no solamente información sobre el sitio web desfigurado sino que además este sitio web almacena una copia de cómo se veía el sitio web desfigurado en el momento de ocurrido el hecho para que se pueda realizar revisiones históricas de las desfiguraciones ocurridas.

Este sitio permite filtrar la información por nombre de dominio, o recibir información de desfiguraciones que hayan ocurrido a dominios a los cuales se esté suscrito. Sus contenidos son públicos y la información que capturan y muestran referentes a las desfiguraciones permanecen en su repositorio de imágenes.

3.5.10 Uso de Sistemas de detección y/o prevención de intrusos

Los IDS e IPS permiten analizar el tráfico que circula hacia o desde el servidor web con la finalidad de emitir alertas sobre actividad inusual que se está desarrollando e incluso, en el caso del IPS, bloquear el origen de esta actividad.

Existe diversidad de sistemas IDS/IPS en el mercado, desde aplicaciones comerciales como es el caso de Fortigate[31], Checkpoint[32]; hasta aplicaciones en Software Libre como son Snort[33] y Suricata[34].

Estos sistemas utilizan sistemas de reglas que permiten determinar el tráfico malicioso que circula a través de ellos.

3.5.11 Implementación de un Web Application Firewall

Un caso especial de IDS/IPS son los conocidos Web Application Firewalls (WAF) llegados al mercado de forma más o menos reciente pero con grandes posibilidades. Los WAF son simplemente un caso especial de IDS/IPS dedicados exclusivamente a analizar el tráfico que ocurre hacia el servidor web.

31 Fortigate: Sistema integrado de manejo de firewall de PaloAlto Networks

32 Checkpoint: Sistema integrado de manejo de firewalls de Check Point Software Technologies

33 Snort: IDS de Snort desarrollado por Sourcefire

34 Suricata: IDS en Software Libre mantenido por http://suricata-ids.org

De la misma forma los WAF tienen una serie rulesets que les permiten determinar las amenazas que están ocurriendo contra el servidor web.

Existen infinidad de WAF en el mercado como son: QualsysGuard[35], Bee-ware WAF[36], Denyall rWeb[37], naxsi[38] y el muy popular mod_security[39]

Los WAF se pueden implementar de forma independiente, tal y como un sistema IDS/IPS tradicional, e incluso dentro de un sistema IDS/IPS tradicional o se pueden incorporar como módulos dentro de servidores web.

3.6 Resumen del estado del arte

En base a lo expuesto hasta el momento presentamos una tabla que valora las diversas falencias aquí descritas con los ataques presentados en la Tabla 3 y su capacidad de protección:

Riesgo	Administración	Programación	Usuario
Fallas de inyección	M	A	
XSS	M	A	M
Fallas en la configuración	A	A	
Exposición de datos sensitivos	M	A	
CSRF	A	A	M

Capacidad de protección: A- Alta, M-Media, B-Baja

Tabla 6: Capacidad de protección ante riesgos según áreas [A]

Como podemos observar, de existir una correcta programación de las aplicaciones, la protección ante estas eventualidades podría ser extraordinariamente alta, sin embargo lamentablemente no siempre podemos depender, como administradores de un sitio web, de la confianza que nos brindaría una programación segura de nuestro sitio. Teniendo en cuenta que la seguridad es un proceso por capas se justifica nuestro trabajo al tratar de analizar las formas de reacción ante ataques de los servidores web que es la parte que es responsabilidad de los administradores.

35 QualsysGuard: WAF desarrollado por Qualsys https://qualys.com/enterprises/qualysguard/web-application-firewall/

36 WAF de Bee Ware http://www.bee-ware.net/en/products/web-application-firewall

37 rWeb: WAF de DenyAll: http://www.denyall.com/decision-maker/products/rweb_en

38 naxsi: WAF en Software Libre https://github.com/nbs-system/naxsi

39 mod_security: WAF desarrollado por Trustwave http://www.modsecurity.org/

Capítulo 4 : Caracterización de seguridad de los servidores web

4.1 Servidores web

En este capítulo realizaremos una exposición de los servidores web más populares en internet. Tomaremos un muestreo de los servidores web en entornos de Software Libre más utilizados en el país. Y recabaremos información sobre servidores web orientados a la seguridad, de forma tal que podamos determinar un conjunto de servidores web sobre los cual basar nuestros análisis.

Entonces realizaremos un análisis de las características de seguridad disponibles en los servidores a analizar y de estadísticas de ataques ocurridos a servidores web en el país en los últimos tiempos basándonos en información acumulada de reportes de zone-h.

Finalizaremos el capítulo exponiendo los resultados de estos análisis que nos permitan determinar un servidor web que pueda ser utilizado para asegurar los sitios web en el país.

4.1.1 Servidores web más populares en Internet

En internet existen varias decenas de servidores web, siendo este mercado dominado por un grupo muy bien definido de servidores. De acuerdo a w3techs [61], al día 27 de Abril del 2014, el 95% del mercado es dominado por:

- Apache
- Nginx
- Microsoft-IIS

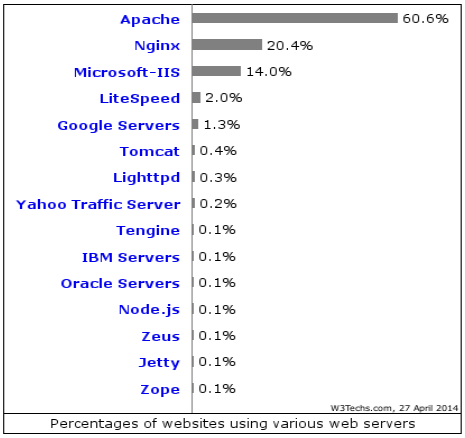


Figura 10: Servidores web más populares según w3techs[62]

De acuerdo a la tendencia de comportamiento histórica, Apache y Nginx han permanecido los dos servidores más populares durante el último año [62] con un apreciable porcentaje de crecimiento de Nginx frente a los demás competidores.

Dividiéndolos por posición de acuerdo al ranking del sitio web [63] los 1000 sitios más populares en internet corren en las siguientes plataformas:

- Apache
- Nginx
- Google servers

Como podemos observer Apache y Nginx son dos servidores web que corren en Ambientes de Software Libre que ocupan, por mucho, un porcentaje apreciable de todo el tráfico web en el planeta.

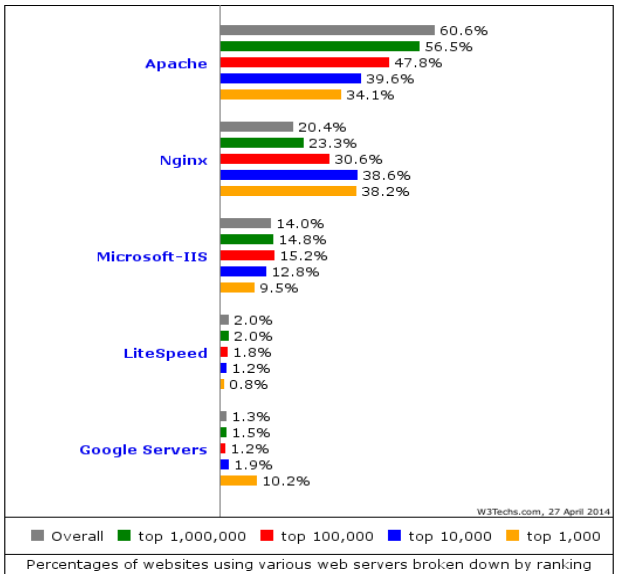


Figura 11: Servidores más populares según ranking de sitios[64]

Otra información útil que podemos obtener del mismo sitio es el lenguaje de programación más popular, con un 82% del mercado es PHP[64], seguido de lejos por su otro competidor, una herramienta propietaria llamada ASP.net con el 17.6% lo que completa más del 99.6%. Existen otros lenguajes de programación como Java, ColdFusion, Perl, Ruby con un porcentaje muy bajo de los sitios web. Debemos aclarar que este gráfico suma más del 100% porque algunos sitios pueden utilizar más de un lenguaje de programación a la vez.

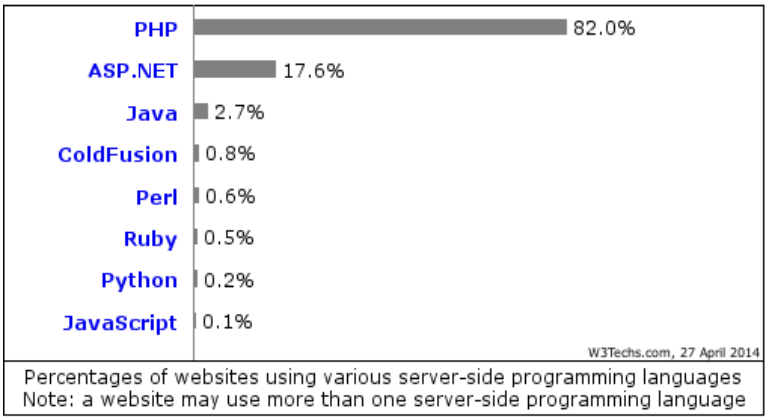
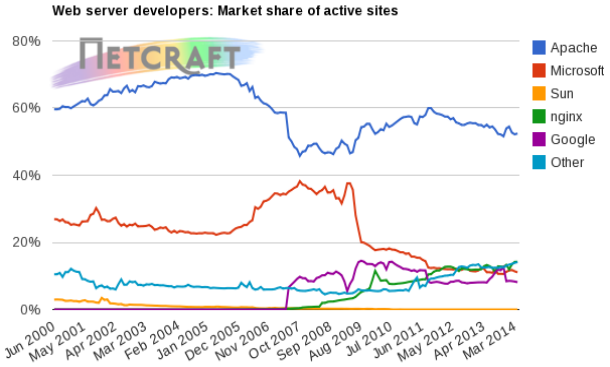


Figura 12: Lenguaje de programación más usado para sitios web[65]

De acuerdo a las tendencias históricas mostradas, podemos observar que PHP ha ido ganando lentamente mercado [65]

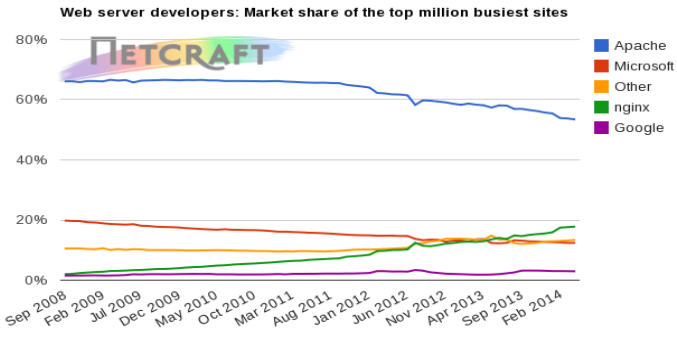
Validando información similar del popular sitio de estadísticas Netcraft, podemos observar en las estadísticas al día 27 de Abril del 2014 [66] que Apache y Nginx, conjuntamente con Microsoft-IIS son los 3 servidores web más populares, tanto para los sitios activos cuanto para el millón de sitios más populares en internet como podemos observar en las estadísticas por sitios activos de acuerdo a NetCraft.com:



Developer	March 2014	Percent	April 2014	Percent	Change
Apache	93,759,928	52.18%	95,512,314	52.44%	0.26
nginx	25,497,586	14.19%	25,900,525	14.22%	0.03
Microsoft	20,436,280	11.37%	20,175,151	11.08%	-0.30
Google	14,967,579	8.33%	14,829,924	8.14%	-0.19

Figura 13: Estadísticas por sitios activos de acuerdo a NetCraft[66]

Similares valores obtenemos al ver las estadísticas para el millón de sitios más populares de acuerdo a NetCraft:



Developer	March 2014	Percent	April 2014	Percent	Change
Apache	537,714	53.77%	534,392	53.44%	-0.33
nginx	176,507	17.65%	178,154	17.82%	0.16
Microsoft	123,981	12.40%	124,019	12.40%	0.00
Google	29,937	2.99%	29,593	2.96%	-0.03

Figura 14: Desglose por servidor web del millón de sitios más populares[66]

4.1.2 Servidores web más usados en el país

De acuerdo al sitio w3techs [67] el servidor web más utilizado en Ecuador es Apache. Como curiosidad podemos notar que es igualmente popular en nuestro continente.

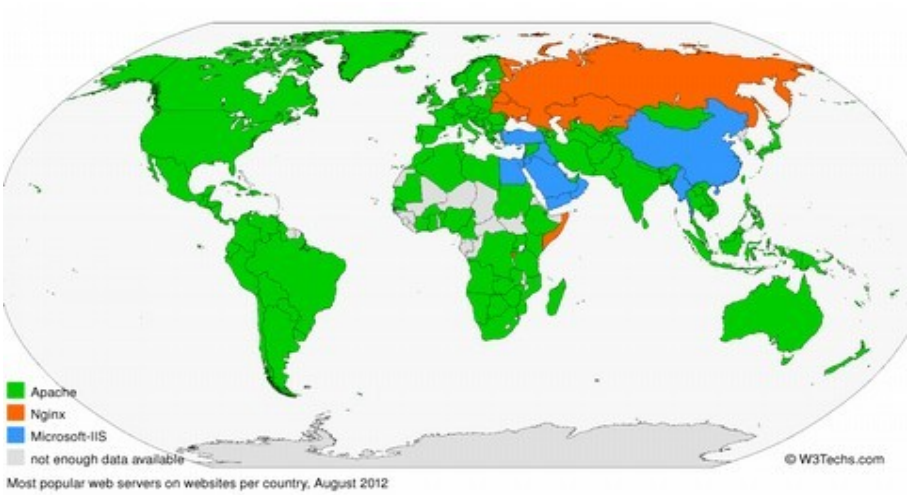


Figura 15: Servidores web más populares por país[67]

Para validar esta información, creamos un script que analizara los encabezados de los sitios web de varias redes del país y determinara el tipo de servidor web que está corriendo en estos equipos.

Para ello utilizamos la herramienta curl para verificar los headers de un sitio web de la siguiente forma

```
curl -s --head www.nombredelsitio.com
```

Por ejemplo

```
curl -s --head www.ecualinux.com
```

Nos arrojaría información sobre nuestro sitio www.ecualinux.com:

```
HTTP/1.1 200 OK
Server: nginx
Date: Sun, 27 Apr 2014 14:24:20 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
Set-Cookie: 02f9f27400a42c837653c993e13eb2ad=4djbmdb7o45qnphpgo4q9053v3; path=/
P3P: CP="NOI ADM DEV PSAi COM NAV OUR OTRo STP IND DEM"
Expires: Mon, 1 Jan 2001 00:00:00 GMT
Last-Modified: Sun, 27 Apr 2014 14:24:20 GMT
Cache-Control: post-check=0, pre-check=0
Pragma: no-cache
```

Figura 16: Captura de headers http para sitio ecualinux.com [A]

Donde indica que está corriendo nginx como podemos observar en la segunda línea: “Server: nginx”

Creamos un script que ayudara en este trabajo, al cual se le da como entrada un archivo con una lista de sitios, y el script va revisando y mostrando los encabezados de cada sitio. Este script se muestra en el Anexo II.

Utilizamos un conjunto finito de sitios del país (50) como muestra para estos experimentos y nos arroja los resultados mostrados en el Anexo III.

4.1.3 Resultados de la observación

En base a los resultados mostrados en el Anexo III podemos obtener la siguiente información:

- Respondieron con datos válidos 48 sitios
- 1 GlassFish
- 2 nginx
- 6 Microsoft-IIS

Servidor	% de la muestra
GlassFish	2.08
Nginx	4.16

Servidor	% de la muestra
Microsoft-IIS	12.5
Apache	81.25

Tabla 7: Servidores web utilizados en sitios .ec muestreados [A]

Como podemos observar, se validan las observaciones realizadas anteriormente, el 81.25% de los sitios muestreados corren en Apache.

También podemos confirmar que los 3 servidores más populares son los mismos que en el resto del mundo, con diferencias claras en el porcentaje en el caso de Apache y nginx, aunque estos dos, de forma conjunta suman un valor similar entre los dos.

Los servidores que corren en ambientes de Software Libre más representativos son **Apache** y **nginx**.

4.2 Descripción de servidores web

4.2.1 El servidor httpd de Apache

El servidor httpd de la fundación Apache[40] es el servidor web activo más viejo del mercado, sirviendo aproximadamente el 53% de los sitios web del mundo.

La fundación apache no solamente maneja al servidor httpd sino muchos projectos adicionales, y , aunque ellos solicitan que se le llame el servidor httpd de apache, la costumbre hace que las personas se refieran al servidor web como Apache.

Apache es un servidor web con características que lo hacen muy moldeable:

- Multiplataforma: Corre no solamente en Linux, sino en cualquier otra plataforma que hay en el mercado
- Multiarquitectura: Corre en cualquier arquitectura de hardware existente
- Modular: Es una característica importante de apache y es que se le pueden agregar módulos, no solamente los creados por el autor, sino por terceras partes con la finalidad de incrementar la funcionalidad de éste.

Apache usa tecnologías orientadas a procesos mediante la cual cada conexión requiere de un hijo para atenderla, recargándose mucho el servicio cuando hay muchas conexiones concurrentes.

40 Apache: servidor web de la fundación Apache http://httpd.apache.org

4.2.2 nginx

El servidor nginx (leído como “engine-x”) es un servidor web orientado a servir peticiones en ambientes con gran cantidad de accesos. También funciona como servidor de proxy reverso y como servidor de proxy de mail (POP3, IMAP4, SMTP).

Al igual que la mayoría de servidores web modernos, el funcionamiento del intérprete de páginas web dinámicas recae sobre un programa externo a través de fastcgi.

Tiene un consumo de memoria muy bajo pues los autores se han concentrado en usar técnicas que hacen un uso eficiente de funciones del kernel de Linux así como que han reducido la cantidad de funcionalidades que se ofrecen a básicamente las que son necesarias por los usuarios, al contrario de apache que en la actualidad está visto como sobrecargado por la inmensa cantidad de funciones conque viene empaquetado.

Nginx usa una tecnología orientada a eventos, al contrario de la orientación a procesos de apache. A través de las técnicas orientadas a eventos, un sólo hilo puede atender las peticiones que van llegando al servidor.

4.3 Servidores web orientados a la seguridad

Durante nuestra investigación, pudimos notar que existe al menos un servidor web que ofrece características de seguridad en su configuración[68]. Este servidor es conocido como el servidor web de hiawatha-webserver [41] al que nos referiremos como hiawatha. Es un servidor relativamente nuevo pues comenzó en el 2002, sin embargo con la implementación de fastcgi en a finales del 2006 [69] y características avanzadas como manejo de XSLT[42], IPv6[43], reescritura de URL a inicios del 2007 es que despega en su popularidad.

El autor mantiene en continuo desarrollo el sistema, implementando nuevas ideas y características en él. En Enero del 2012 cambia la base de la encriptación por SSL de OpenSSL a PolarSSL[44], una decisión que demostró su validez en este año al demostrarse una grave falla de seguridad en OpenSSL conocida como Heartbleed[32] que no afecta a PolarSSL.

En Mayo del 2012 se incorpora la capacidad de realizar proxy reverso tal y como los servidores web Apache y nginx que dominan el mercado.

Su autor afirma en la sección *Project Goals*:

41 hiawatha-webserver: servidor web de Hugo Leisink http://www.hiawatha-webserver.org

42 XSLT: Transformaciones XSL

43 IPv6: Protocolo de Internet, versión 6. Versión de IP sucesora de IPv4

44 PolarSSL: Paquete de SSL desarrollado por http://polarssl.org

“Because of my great interest in IT security, I paid extra attention to security while I was working on Hiawatha. Beside all the default security measures you can expect from a modern webserver, there are a lot of security features in Hiawatha you won't find in any other webserver. Many of them started as an experiment, but in the meantime, most of them have proven to be very usefull.

A second thing I wanted my webserver to be is easy-to-use. This resulted in a readable configuration syntax and not having to be a HTTP or CGI expert in order to get Hiawatha running.

I strongly believe that many features you find in several other webservers, shouldn't be placed inside a webserver but in a web application. They make them big and bloated. Hiawatha only has the features necessary to do what a webserver has to do; serving web applications. Hiawatha's small size makes it therefor perfect for embedded systems and older hardware.”

Podemos de esta afirmación obtener la siguiente información:

- hiawatha implementa las medidas de seguridad que otros servidores web tienen por defecto
- hiawatha además implementa otras medidas de seguridad agregadas por el autor
- hiawatha es un servidor web fácil de configurar
- El autor piensa que muchas características que se implementan en otros servidores web, deberían manejarse desde las aplicaciones. Esto es: el objetivo de un servidor web es servir páginas web, y estas características le cargan demasiado al servidor.
- El código fuente de hiawatha es pequeño.

4.4 Servidores a analizar en este documento

En esta tesis realizaremos un análisis de los servidores Apache, nginx y hiawatha en vista de que Apache y nginx son los dos servidores más populares al momento en internet y de las características de seguridad ofrecidas por hiawatha.

4.5 Características de seguridad disponibles en los servidores a analizar

Además de las características deseadas mostradas en la Tabla 6, se mostrarán características comunes y propias que vienen implementadas en cada servidor web por defecto en CentOS.

4.5.1 Líneas de código y comunidad

Una parte importante del desarrollo de un sistema es la comunidad que rodea a esta aplicación y las líneas de código.

- Una comunidad pequeña puede ser indicador de riesgo ya que los pocos autores de ella pueden desaparecer por diversas causas.
- La cantidad de líneas de código indican la complejidad de la aplicación desarrollada y puede ser un indicador de posibles errores ocultos o descubiertos en ella debido a que, lógicamente, mientras más líneas de código, más difícil es realizar una auditoría a la aplicación.

Estas mediciones se han obtenido de Ohloh [70], una herramienta web para realizar diversas comparativas en aplicaciones en Software Libre.

Servidor web	Tamaño de la comunidad	Líneas de código	Comentarios en el código
Apache	117	2 275 269	Bien comentado
Nginx	30	139 169	Muy poco comentado
Hiawatha	3	55 970	Bien comentado

Tabla 8: Servidores web según tamaño de comunidad, cantidad de líneas y calidad del código[70]

Fortalezas:

- Apache: La cantidad de colaboradores es suficientemente alta como para mantener la estabilidad del proyecto. Esta cantidad de colaboradores tiene que ver con la antiguedad y popularidad de este sistema. Tiene un código fuente bien comentado.
- Nginx: La cantidad de colaboradores es bastante alta como para mantener la estabilidad del proyecto. Esta cantidad de colaboradores tiene que ver también con la antiguedad y popularidad de este sistema.
- Hiawatha: Un código fuente pequeño y bien comentado.

Debilidades:

- Apache: una enorme cantidad de código fuente, superior a los 2.2 millones. La gran cantidad de colaboradores podría hacer que se demoren las labores de coordinación entre ellos.
- nginx: Un código muy poco comentado que le hace muy difícil de leer a nuevos desarrolladores.

- hiawatha: Una muy baja cantidad de colaboradores le hace propenso a que pueda ser abandonado el proyecto.

4.5.2 Vulnerabilidades publicadas por servidor

También se realizó un análisis de las vulnerabilidades que han sido reportadas para cada uno de los servidores web antes analizados por el sitio secunia [71]. Secunia clasifica estas vulnerabilidads según el estado de la resolución de las vulnerabilidades, cuán crítica es la vulnerabilidad, el origen del ataque y el impacto que causa esta vulnerabilidad en el servidor atacado. Un resumen tabulado para cada uno de los servidores web puede ser obtenido en el *Anexo IV*.

Como resultado de un análisis de esta información de secunia.com podemos colegir que:

Apache

- Ha sido reportadas 86 vulnerabilidades
- De ellas permanecen sin corregir el 9.57% de ellas y el 8.48% han tenido soluciones parciales.
- El 31.37% han tenido un nivel de criticidad de Extremo, Alto o Medio.
- El 83.71% han podido ejecutarse de forma remota.
- Sobre el impacto que pueden causar estas vulnerabilidades: El 41.19% consistían en ataques de negación de servicio, el 9.72% permitía acceso a información del sistema; 6.77% escalada de privilegios; 12.81% exposición de información sensitiva y 14.17% ataques de XSS y 9.72% acceso a información del sistema; 13.34% Evasión a las seguridades; 1.09 spoofing y 1.09 manipulación de datos.

nginx

- Ha sido reportadas 17 vulnerabilidades
- De ellas permanecen sin corregir el 17.71%
- El 59% han tenido un nivel de criticidad de extremo a medio
- El 76.65% de estos ataques pueden ser ejecutados remotamente 76.65%
- Sobre el impacto que pueden causar estas vulnerabilidades: El 27.88% consistían en ataques con acceso al sistema, 16.12% ataques de spoofing, 17.71% ataques de XSS y 22% ataques de evasión de las seguridades

Hiawatha

- Ha sido reportada 1 vulnerabilidad
- Por el tipo de vulnerabilidad consideramos que esta ha tenido un nivel de criticidad medio

- Esta vulnerabilidad pudo ser explotada remotamente
- El impacto que puede causar era un ataque de negación de servicio

Como podemos observar, **hiawatha** presenta menos vulnerabilidades y de menor impacto y nivel de criticidad, sin embargo esto podría argumentarse que es por su poca popularidad o tiempo en el mercado; aunque es cierto que también puede verse desde el punto de vista de utilizar un código compacto que permite mejores auditorías.

Apache y nginx han sido objeto de ataques de negación de servicio, de XSS entre otras.

4.5.3 Desempeño ante ataques de negación de servicio

En Marzo del 2014, Chris Wadge publicó sus hallazgos sobre el desempeño de ciertos servidores web ante un ataque de negación de servicio estilo slowloris[45] utilizando la herramienta SlowHTTPTest [72], los resultados de estos experimentos los mostró en el weblog de hiawatha de los que podemos sacar las siguientes conclusiones.

4.5.3.1 Análisis de desempeño de servidor Apache

A menos de 19 segundos desde que comenzó el ataque con la herramienta SlowHTTPTest el servidor dejó de servir contenidos. Durante casi 95 segundos mantuvo las conexiones en pendiente y luego las comenzó a cerrar . Durante todo este tiempo no volvió a servir contenidos.

4.5.3.2 Análisis de desempeño de servidor Nginx

El servidor dejó de servidor contenidos durante 8 segundos entre el segundo 16 y el 24 y durante menos de 8 segundos entre el segundo 40 y el 48. No dejó conexiones en espera.

4.5.3.3 Análisis de desempeño de servidor Hiawatha

Durante todo el periodo del experimento no dejó de servir conexiones ni dejó conexiones pendientes.

4.5.4 Análisis de resultados

De acuerdo a los resultados obtenidos en este experimento podemos deducir que:

- Sí existen y han existido vulnerabilidades en los servidores web analizados.
- Apache tuvo un desempeño muy lejanos a lo deseado ante un ataque de negación de servicio.

45 Slowloris: ataque DoS que abre conexiones parciales a servidores web para agotar sus recursos

- Nginx tuvo un comportamiento bastante adecuado aunque evidenció ciertas pérdidas de conectividad.
- Hiawatha fue el servidor que mejor comportamiento tuvo ya que no dejó de servir peticiones durante todo el tiempo de desarrollo del ataque.

4.6 Características de seguridad ofrecidas por los servidores a estudiar

En base a lo estudiado en este capítulo, realizamos una investigación sobre las características de seguridad que nos ofrecen los servidores Apache, nginx y hiawatha para protegernos de las fallas más explotadas por los atacantes contra aplicaciones dinámicas, estas fueron definidas en el capítulo de Resumen del estado del arte de la presente tesis y confirmadas en los capítulos posteriores y son:

- Fallas de inyección
- XSS
- Fallas en la configuración
- Exposición de datos sensitivos
- CSRF

Las fallas en la configuración las dividiremos en varios subtipos para estudiarlos más claramente que serán:

- Facilidad de configuración del servidor web
- Posibilidad de aplicar bloqueos por países
- Posibilidad de aplicar bloqueos por IP
- Posibilidad de aplicar bloqueos mediante autenticación
- Corrección de caracteres especiales en URL
- Modificación de información a mostrar por el servidor web en los encabezados

Las características que aquí se analizarán se refieren únicamente a las que son ofrecidas por el fabricante del servidor web analizado.

4.6.1 Características de seguridad ofrecidas por apache

Inyección de SQL	XSS	CSRF	Exposición de datos sensitivos	Bloqueo por IP	Bloqueo por autenticación	Chequeo de caracteres especiales en url	Modificación de información en headers
mod_rewrite	mod_rewrite	mod_rewrite	mod_ssl	Sí	Sí	mod_rewrite	Sí

Tabla 9: Características de seguridad ofrecidas por apache [A]

4.6.2 Características de seguridad ofrecidas por nginx

Inyección de SQL	XSS	CSRF	Exposición de datos sensitivos	Bloqueo por IP	Bloqueo por autenticación	Corrección de caracteres en URL	Modificación de información en headers
query_string	query_string	query_string	mod_ssl	Sí	Sí	Sí	Sí

Tabla 10: Características de seguridad ofrecidas por nginx [A]

4.6.3 Características de seguridad ofrecidas por hiawatha

Inyección de SQL	XSS	CSRF	Exposición de datos sensitivos	Bloqueo por IP	Bloqueo por autenticación	Corrección de caracteres en URL	Modificación de información en headers
Sí	Sí	Sí	PolarSSL	Sí	Sí	Sí	Sí

Tabla 11: Características de seguridad ofrecidas por hiawatha [A]

4.6.4 Comparativa de características de seguridad

Como podemos observar existen propuestas de solución en los casos de nginx y de apache las cuales se manejan a través de configuraciones realizadas mediante módulos que permiten analizar los headers de las conexiones y aceptar o rechazar las conexiones. Son soluciones complejas pues tanto el administrador de apache como el de nginx deben tener en cuenta las posibles variantes que puedan ir surgiendo y modificando sus archivos de configuración o .htaccess para incluir nuevas modalidades que puedan surgir de estos ataques.

Hiawatha permite algo similar a través del parámetro URLToolKit, sin embargo también tiene incluido dentro del código embebidas estas detecciones. Pero en el caso de hiawatha, este trabajo se maneja de forma más cómoda puesto que el autor se ocupa de liberar, conjuntamente con cada versión, cualquier modificación o adición que tenga que hacerse para prevenirse de estos ataques, como es el ejemplo de hiawatha-9.5 en el cual le solicitamos y se aceptó[74] la inclusión del análisis del header “Origin:” para mitigar ataques de CSRF. Además hiawatha no solamente bloquea los intentos de intrusión, sino que además bloquea durante un intervalo de tiempo que podemos definir al atacante de forma tal que si intentara ulteriores ataques de cualquier otro tipo, el sistema le estaría bloqueando.

Aunque las soluciones de SSL son externas a los servidores web, las hemos mencionado porque son provistas por el fabricante. En el caso de PolarSSL, utilizado por hiawatha, este contiene un código más compacto y con una pisada más pequeña que su contraparte más popular OpenSSL, usado por nginx y apache, la cual incluso está siendo reescrita por otro grupo llamado LibreSSL[46] con la finalidad de que tenga una menor pisada y un código más claro y simple sin perder su funcionalidad.

Hiawatha además brinda otras protecciones que no vienen incluídas en los servidores apache y nginx como son el caso de validaciones de: datos basura, envío de encabezado aleatorio y chqueo de md5 para directorios.

En efecto existen herramientas de terceros como WAF que permiten detectar estos tipos de ataques, pero como podemos observar es hiawatha el único servidor que viene con configuraciones específicas incorporadas en el servidor web para protegernos de este tipo de ataques.

46 LibreSSL: paquete de SSL desarrollado por http://libressl.org

4.7 Análisis de las estadísticas de ataques a sitios web del país

Para realizar el análisis de ataques a sitios web del país utilizaremos la información del sitio zone-h.org.

Para realizar esta labor verificaremos la información que hemos recibido de zone-h.org desde el día 6 de Enero del 2014 al 1ro de Mayo del 2014. Este reporte nos llega solamente de redes de Ecuador. Analizaremos solamente los reportes confirmados (*accepted*) que haya emitido zone-h.org

Para obtener las estadísticas, se generó un script con una salida en CSV (**Anexo V).**

4.7.1 Estadísticas por tipo de servidor

De los 176 sitios Ecuatorianos reportados por zone-h.org como desfigurados durante el periodo muestreado que nos arroja los siguientes resultados:

Desglose por Sistema Operativo

OS	Desfiguraciones	Apache	Nginx	IIS
Linux	159	146	13	0
BSD	2	2	0	0
Windows	14	2	0	12
Desconocido	1	1	0	0

Tabla 12: Desfiguraciones según Sistema Operativo utilizado [A]

Desglose por dominio de segundo nivel (SLD):

SLD	Total	Apache	Nginx	IIS
.gob.ec	41	41	0	0
.com.ec	66	59	7	0
.edu.ec	27	14	2	11
.ec	31	26	4	1
.net.ec	2	2	0	0
.k12	1	1	0	0
.org.ec	7	7	0	0
.mil.ec	1	1	0	0
Total	176	151	13	12

Tabla 13: Desfiguraciones según SLD [A]

Desfiguraciones recurrentes y masivas por tipo de servidores web

	Apache	Nginx	IIS
Desfiguraciones	151	13	12
Recurrentes	13	1	8
Masivas	71	8	12

Tabla 14: Desfiguraciones según tipo de servidor web [A]

Desglose de servidores por tipo de ataque

Tipo de Ataque	Cantidad	Apache	Nginx	IIS
File Inclusion	39	38	0	1
SQL Injection	48	32	6	10
Configuración	12	10	2	0
Administrador	1	1	0	0
Falla servidor (OS)	35	33	2	0
Falla app web	21	21	0	0
SSH	7	7	0	0
No responde	3	2	1	0
FTP	1	1	0	0
Web Server Intrusion	3	2	1	0
Ingeniería social	1	1	0	0
Vulnerabilidad conocida	3	2	0	1
Envenenamiento de URL	2	1	1	0

Tabla 15: Desfiguraciones según tipo de ataque utilizado [A]

4.7.2 Resultados del análisis

Observando los datos tabulados anteriormente podemos colegir que:

Según la tabla "*Desglose por Sistema Operativo*" el 91.4% de las desfiguraciones ocurrieron a servidores tipo Unix.

Según "*Desglose por dominio de segundo nivel (SLD)*":

- Más de la mitad de las desfiguraciones de organizaciones .edu.ec y .k12.ec fueron realizadas a apache o nginx.
- Las organizaciones .com.ec y .ec desfiguradas utilizan mayormente apache aunque también hay algunas que utilizan nginx

- Las desfiguraciones a .gob.ec, .net.ec, .org.ec y a .mil.ec fueron exclusivamente realizadas a apache

Excepto en el caso de las .edu.ec donde hubieron 10 ataques a IIS y a .ec donde hubo 1 representando el 6.81%; el resto de dominios utilizaban apache o nginx cuando fueron desfigurados. Acumulando un 85.79% de los ataques que usaban servidores apache y el 7.38% usaban nginx.

De acuerdo a "*Desfiguraciones masivas y recurrentes por tipo de servidores web*" entre el 47% en el caso de apache y el 100% en el caso de IIS, las desfiguraciones fueron masivas, esto es: afectaron a más de un sistema web que corría en el mismo servidor.

Sin embargo el 8.6% de las desfiguraciones ocurridas a apache fueron recurrentes, esto es: el sitio ha sido desfigurado más de una vez en el tiempo. La tasa de recurrencia para nginx era de 7.69%; y la de IIS era del 66.67%.

De acuerdo a "*Deglose de servidores por tipo de ataque*":

Los tipos de ataque más aprovechados según el tipo de servidor son:

- Apache:
  - Inclusión de archivos con 25.15%,
  - SQL Injection con 21.19%,
  - Fallas en el Sistema Operativo con 21.85% y
  - Fallas en la aplicación web con un 13.9%
  - Todas estas suman un : 82.09% de las fallas explotadas.
- Nginx:
  - SQL Injection con un 45.15% de las fallas
  - Configuración del sistema: 15.38% y
  - Fallas en el Sistema Operativo: 15.28%
  - Todas ellas suman: 75.91%
- IIS:
  - SQL Injection con el 83.33%

Las observaciones aquí indicadas confirman lo expuesto en el documento que indica que la mayoría de las fallas ocurren por problemas de programación o de administración. Pudiéndose corregir la mayoría de estas aplicando correctas políticas de protección del servidor web.

Capítulo 5 : Metodología

5.1 Aplicación vulnerable a proteger

La aplicación que utilizaremos para realizar las pruebas conocida como “Damn Vulnerable Web Application”(DVWA)[47] la cual tiene como característica que es vulnerable a las pruebas que realizaremos a los servidores web. De esta forma podremos validar las diferencias en las respuestas que se ofrecen al acceder al mismo tipo de falla, desde diferente servidor web analizado.

Esta es una aplicación hecha en PHP y con acceso a una Base de Datos MySQL.

5.2 Creación de los escenarios de prueba

En esta sección se describirá cómo se crea el escenario de prueba para validar la reacción ante ataques hacia la aplicación de prueba antes descrita. Este escenario consistirá en un servidor con Linux CentOS-6 el cual estará instalado y asegurado.

Una vez se instale el servidor y los repositorios necesarios, este será asegurado y fortalecido. Posteriormente se instalarán los paquetes necesarios para correr los 3 servidores web así como una instalación vulnerable que nos servirá de pruebas para cada escenario.

5.2.1 Configuración de Sistema Operativo base

5.2.1.1 Instalación inicial y particionamiento

El sistema base que usaremos será CentOS en la última versión pública y estable que es CentOS-6. Como nuestra arquitectura lo permite, usaremos CentOS-6 de 64bits, aunque estas pruebas pueden ser hechas igualmente en 32bits.

La máquina virtual en todos los casos tendrá los valores mínimos recomendados por CentOS-6 para la RAM 1GB. Además tendrá un disco de 30GB particionado en el formato sugerido por CentOS-6 que es ext4, este espacio en disco nos dará suficiente espacio para poder realizar nuestras pruebas. El procesador será el mismo provisto por el sistema hospedero de virtualización.

47 DVWA: Aplicación de pruebas de vulnerabilidades de RandomStorm http://www.dvwa.co.uk/

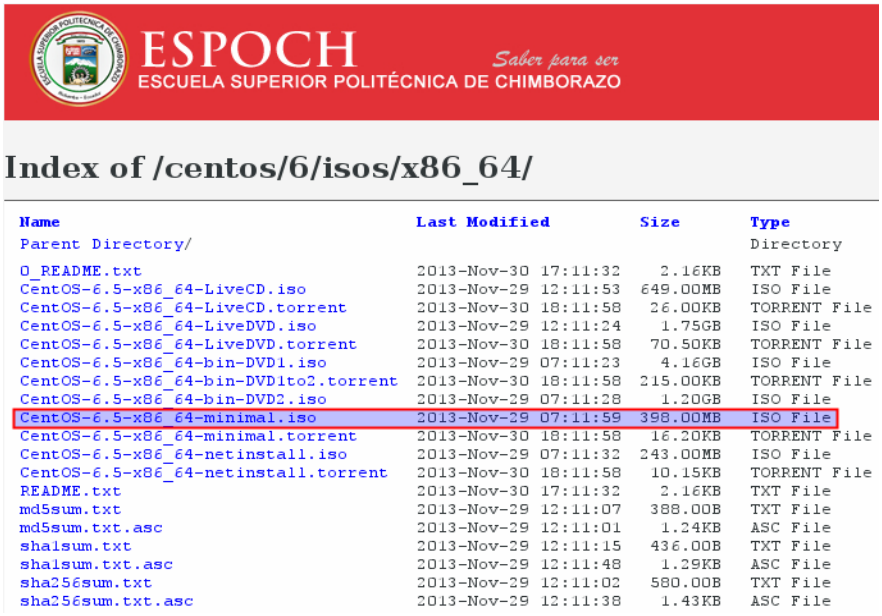


Figura 17: Descarga de iso mínimo de CentOS-6 [A]

Para instalar el Sistema Operativo base se optó por hacer lo que CentOS conoce como una instalación mínima, que consiste en bajar el ISO de instalación de CentOS conocido como mínimal como mostramos en la figura:

Se verifica la suma md5 de la figura con el objetivo de asegurarnos que es la original que CentOS nos brinda. Para ello obtenemos el archivo sha1sum.txt y comprobamos:

```
[eperez@laptop CentOS-6.5-x86_64-minimal]$ egrep minimal sha1sum.txt
f21a71e8e31df73297bdd1ccd4a64a36831284bd  CentOS-6.5-x86_64-minimal.iso
[eperez@laptop CentOS-6.5-x86_64-minimal]$
[eperez@laptop CentOS-6.5-x86_64-minimal]$ sha1sum CentOS-6.5-x86_64-minimal.iso

f21a71e8e31df73297bdd1ccd4a64a36831284bd  CentOS-6.5-x86_64-minimal.iso
```

Figura 18: Verificación de suma SHA de iso descargado [A]

Como podemos ver, la suma sha1 que reporta el archivo sha1sum.txt es la misma suma que obtenemos al hacer sha1sum al ISO del CentOS minimal lo que demuestra que el ISO bajado es idéntico al ISO que CentOS publica en su sitio web.

La instalación se realiza normalmente con la excepción del particionamiento. Para ello, en la ventana que nos solicita el tipo de particionamiento, escojemos la opción: Crear diseño personalizado como podemos observar al fondo de la siguiente Figura:

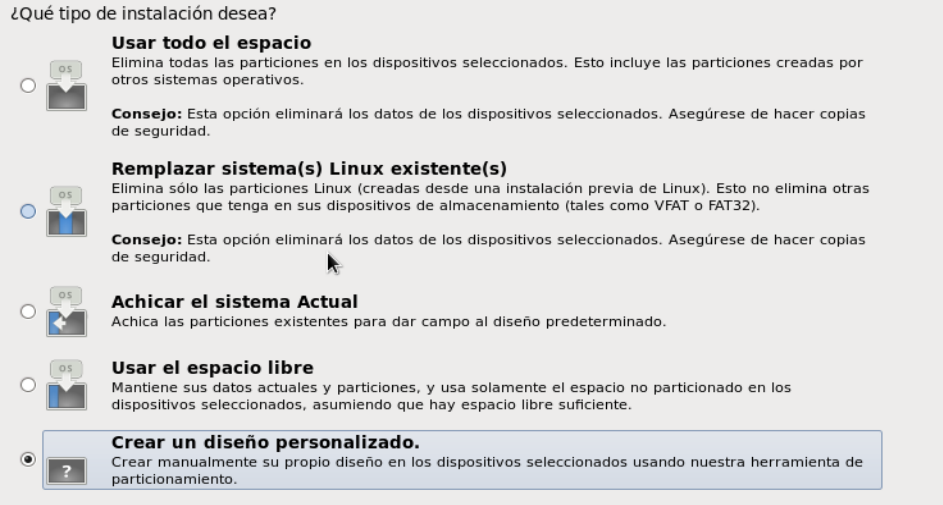


Figura 19: Selección de particionamiento personalizado [A]

Al continuar, veremos que tenemos un disco vacío llamado sda

Figura 20: Disco sda que se usará para la instalación [A]

En esta ventana procedemos a crear las particiones que serán:

Partición	Espacio
/boot	200M
/	5000M
/tmp	1024M
SWAP	1024M
/var	El resto del disco

Tabla 16: Propuesta de particionamiento en Sistema Operativo de pruebas [A]

El objetivo de particionar de esta forma, será poderle implementar seguridades al Sistema Operativo a nivel del sistema de archivos, tales como las descritas anteriormente de noexec.

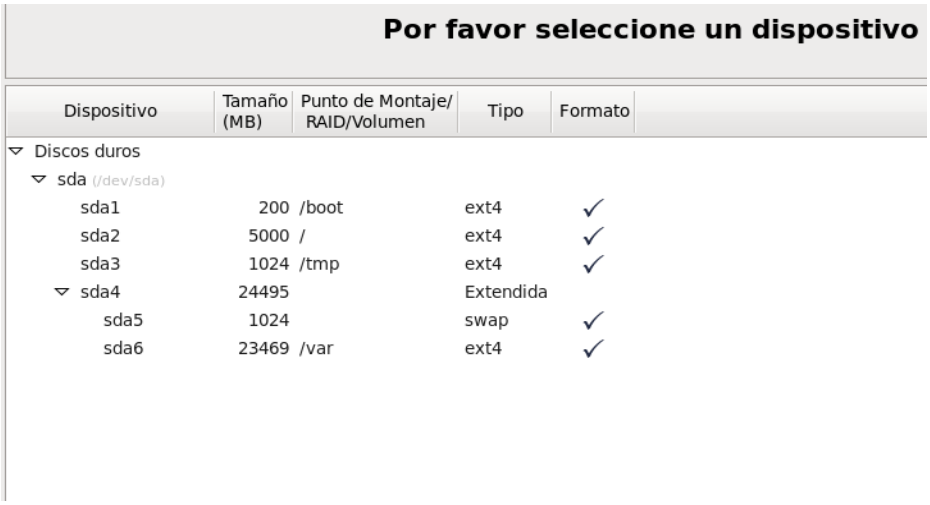


Figura 21: Disco sda particionado [A]

El orden y el tamaño de las particiones dependen de cómo se fueron agregando y del tamaño del disco en que se están haciendo las pruebas.

5.2.1.2 Paquetes instalados por defecto

El resto del proceso se sigue normalmente, como es el disco de instalación mínima, solamente instalarán los paquetes del sistema necesarios para este tipo de instalación. De esta forma garantizamos que un atacante que logre acceder al sistema, no pueda hacer uso de paquetes innecesariamente instalados.

5.2.1.3 Activación de la interfaz de red

En las instalaciones mínimas, CentOS-6 no activa la interfaz de red por lo que, tan pronto entramos, procedemos a activarle editando el archivo /etc/sysconfig/network-scripts/ifcfg-eth0, poniendo la siguiente configuración:

```
DEVICE=eth0
TYPE=Ethernet
ONBOOT=yes
NM_CONTROLLED=no
BOOTPROTO=dhcp
```

Con la finalidad de luego clonar las máquinas, hemos eliminado los parámetros de UUID y de HWADDR ya que HWADDR impediría que las máquinas clonadas arranquen su red pues al clonarse, se cambia la MAC address de las tarjetas de red. El UUID al momento no es necesario y por claridad le eliminamos.

Como podemos observar, el parámetro ONBOOT ha sido puesto a yes con la finalidad de que arranque al iniciar el sistema y al poner la opción NM_CONTROLLED a no se ha desactivado la posibilidad de que el servicio NetworkManager le controle a la interfaz, servicio que por demás no está disponible en las instalaciones mínimas.

Luego de esto procedemos a reiniciar el servicio de red y ya estará activa nuestra interfaz:

```
service network restart
chkconfig network on
```

5.2.1.4 Activación de servicio sshd

Este paso no es estrictamente necesario pero deseamos acceder remotamente a la máquina por ssh por lo que procedemos a activar el servicio de sshd:

```
service sshd restart
chkconfig sshd on
```

5.2.1.5 Activación de puerto 80 en el firewall

El firewall por defecto de CentOS viene activado en las instalaciones mínimas, este firewall solamente permite conexiones entrantes dirigidas al puerto 22/TCP que está siendo atendido por el servicio de sshd.

Para realizar nuestras pruebas necesitaremos abrir al menos el puerto 80/TCP en nuestro firewall, para ello editamos el archivo /etc/sysconfig/iptables para que quede de la siguiente forma:

```
[root@base ~]# cat -n /etc/sysconfig/iptables
     1  # Firewall configuration written by system-config-firewall
     2  # Manual customization of this file is not recommended.
     3  *filter
     4  :INPUT ACCEPT [0:0]
     5  :FORWARD ACCEPT [0:0]
     6  :OUTPUT ACCEPT [0:0]
     7  -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
     8  -A INPUT -p icmp -j ACCEPT
     9  -A INPUT -i lo -j ACCEPT
    10  -A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
    11  -A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
    12  -A INPUT -j REJECT --reject-with icmp-host-prohibited
    13  -A FORWARD -j REJECT --reject-with icmp-host-prohibited
    14  COMMIT
[root@base ~]# _
```

Figura 22: Puerto 80 abierto en firewall de CentOS [A]

Como podemos observar, en la línea 11 hemos agregado una regla para permitir conexiones nuevas hacia el puerto 80/TCP.

Reiniciamos el firewall :

```
service iptables restart
```

y podríamos verificar con el comando iptables -nvL que ya está activada esta opción:

Figura 23: Verificación de puerto abierto en firewall de CentOS [A]

5.2.1.6 Opciones al filesystem

A las particiones /tmp, /boot y /var les agregaremos la opción noexec.

Para ello editamos el archivo /etc/fstab y agregamos la opción noexec luego de la opción default de cada una de estas particiones:

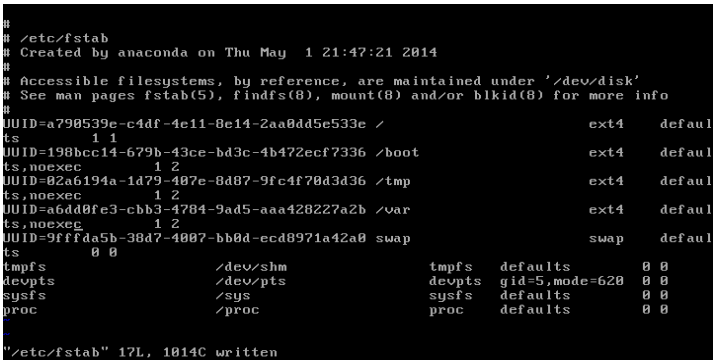


Figura 24: Aplicación de noexec a particiones creadas [A]

Posteriormente remontamos cada una de estas particiones, verificando que no nos diera ningún tipo de mensaje en ninguna de estas operaciones, lo que permite validar que lo antes escrito está bien:

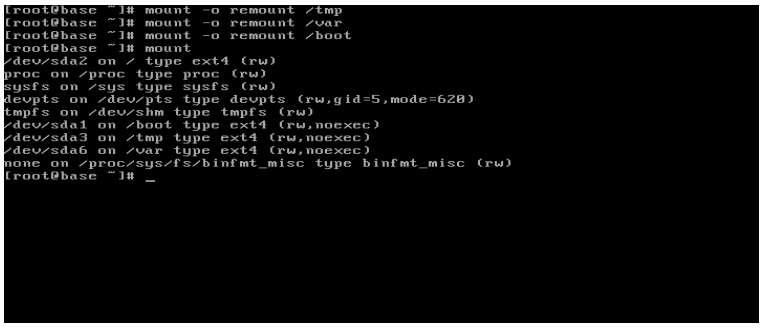


Figura 25: Verificación de implementación de noexec [A]

Como podemos verificar, tanto para /boot, /tmp y para /var, aparece la opción noexec al final de cada línea.

5.2.1.7 Configuración de repositorios

Con la finalidad de poder instalar cada uno de los servidores web, procederemos a instalar los siguientes repositorios:

- Repositorio de *epel* pues este contiene el paquete *nginx*
- Repositorio de *centosec* pues este contiene el paquete *hiawatha*

Estos paquetes no se instalarán ahora, solamente dejaremos listos los repositorios con la finalidad de usarles posteriormente.

epel: bajamos e instalamos la última versión de epel-release con el comando:

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

centosec: bajamos e instalamos la última versión de centosec con el comando:

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

5.2.1.8 Actualización inicial

Con la finalidad de tener las últimas actualizaciones de CentOS, procedemos a actualizar el sistema:

```
yum -y update
```

5.2.2 Instalación de paquetes de servidores web

Podemos ya instalar los 3 servidores web que probaremos, para ello ejecutamos:

```
yum install httpd nginx hiawatha php-fpm php-mysql php mysql-server
unzip
```

Con esto ya tenemos listo para configurar y activar los servidores web según se les vaya necesitando.

5.2.3 Descarga de la aplicación vulnerable

En vista de que la misma aplicación se utilizará en cada servidor web, procederemos a dejarla descargada en la figura base para luego poderla instalar desde este archivo descargado.

```
wget https://github.com/RandomStorm/DVWA/archive/v1.0.8.zip
```

Procedemos a abrir el paquete DVWA-1.0.8.zip hacia el directorio /var/www/html

```
cd /var/www/html
unzip /root/DVWA-1.0.8.zip
mv DVWA-1.0.8 dv
```

Editamos el archivo de configuración y verificamos la clave del usuario root:

```
vi dv/config/config.inc.php
```

Permitimos el uso de inclusión de url en php, editamos /etc/php.ini y cambiamos:

```
allow_url_include = On
```

Para ejecutar cada uno de los servidores web, se utilizará el usuario apache y grupo apache, para ello cambiamos los permisos del directorio /var/www/html al usuario y grupo antes indicado.

```
chown apache.apache /var/www/html -R
```

Damos todos los permisos al directorio uploads:

```
chmod 777 /var/www/html/dv/hackable/uploads/
```

Arrancamos el servidor mysql:

```
service mysqld start
chkconfig mysqld on
```

5.2.4 Configuración inicial de servidores web

Cada uno de los servidores web serán instalados y configurados de forma tal que puedan ejecutar aplicaciones dinámicas en PHP y acceder al directorio /var/www/html donde esta estará almacenada. Esta configuración será lo más cercana a la configuración por defecto que proponga el fabricante del paquete.

Todos los servidores web estarán desactivados por defecto, en vista de que es una aplicación vulnerable y solamente se arrancará uno para realizar pruebas. Luego de finalizadas las pruebas se apagará el servidor web con la finalidad de que pueda ser arrancado otro.

Al servidor Linux se le asignó la IP 192.168.1.5 y se apuntó la dirección hc6pe.no-ip.org a la IP pública de la red, la URL de acceso a la aplicación vulnerable es: http://hc6pe.no-ip.org/dv/

5.2.5 Configuración de php-fpm

Los servidores web más actuales utilizan el servicio php-fpm[48] para ejecutar aplicaciones dinámicas como un proceso separado del servidor web, de esta forma logran un mejor desempeño al ejecutar estas aplicaciones. Tanto nginx como hiawatha utilizan esta técnica para ejecutar aplicaciones php.

Este servicio escuchará en su puerto por defecto, el 9000/TCP.

48 PHP-FPM: FastCGI Process Manager, permite la ejecución de PHP en servidores web

Para ello se edita el archivo /etc/php-fpm.d/www.conf y se verifica que el usuario y grupo bajo el cual corre el proceso php-fpm que sea el mismo del servidor web

```
user = apache
group = apache
```

Posteriormente se arranca el servicio de php-fpm

```
service php-fpm restart
chkconfig php-fpm on
```

5.2.6 Configuración inicial del servidor apache

El servidor web apache funciona por defecto, una vez arrancado se puede validar el acceso a la aplicación vulnerable accediendo a la URL de la aplicación vulnerable: http://hc6pe.no-ip.org/dv/ la aplicación se ejecuta a través de mod_php que es un módulo que viene incluído por defecto en apache.

5.2.7 Configuración inicial del servidor nginx

Las configuraciones particulares realizadas al servidor nginx fueron:

- nginx.conf se cambió el usuario a: user apache
- Se configuró el archivo /etc/nginx/conf.d/default.conf para que pudiera acceder al php-fpm y utilizar de documentroot a /var/www/html

Los archivos de configuración del servidor nginx puede ser vista en el Anexo VII.

5.2.8 Configuración inicial del servidor hiawatha

El servidor web hiawatha:

- Se le configuró el host por defecto como hc6pe.no-ip.org
- Se definió el archivo de inicio a index.php
- Se activó el php a través de un proceso fastcgi llamado dv que se conectará al php-fpm en el puerto 9000
- Se definió el WebsiteRoot a /var/www/html que es donde está la aplicación vulnerable
- Se dejaron activadas las protecciones que vienen por defecto en hiawatha contra CSRF, XSS e Inyección de SQL.

La configuración del servidor hiawatha puede ser obtenida del Anexo VIII.

5.3 Pruebas de vulnerabilidades a servidores web

El proceso siempre comenzará realizando los siguientes pasos:

1. Se accede con un usuario y una clave a la DVWA

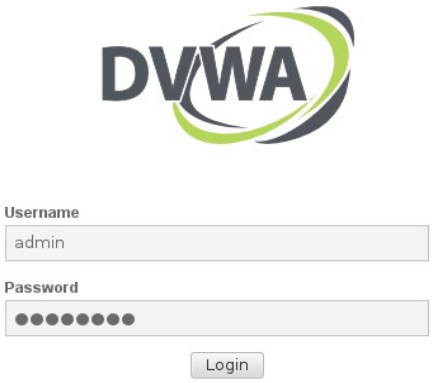


Figura 26: Pantalla de inicio de sesión en DVWA [A]

2. Se verifica que DVWA esté en nivel de seguridad bajo pues es el nivel que presenta todas las vulnerabilidades para ser aprovechadas por un atacante

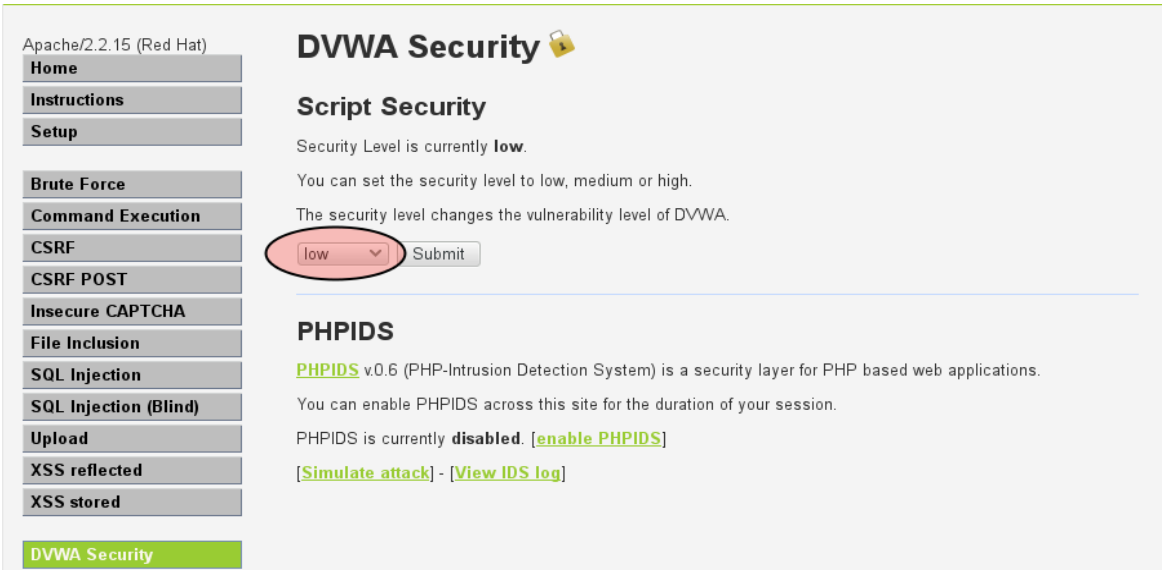


Figura 27: Selección de nivel de seguridad bajo en DVWA [A]

3. Se reinicia la BD de DVWA al cambiar de servidor web.

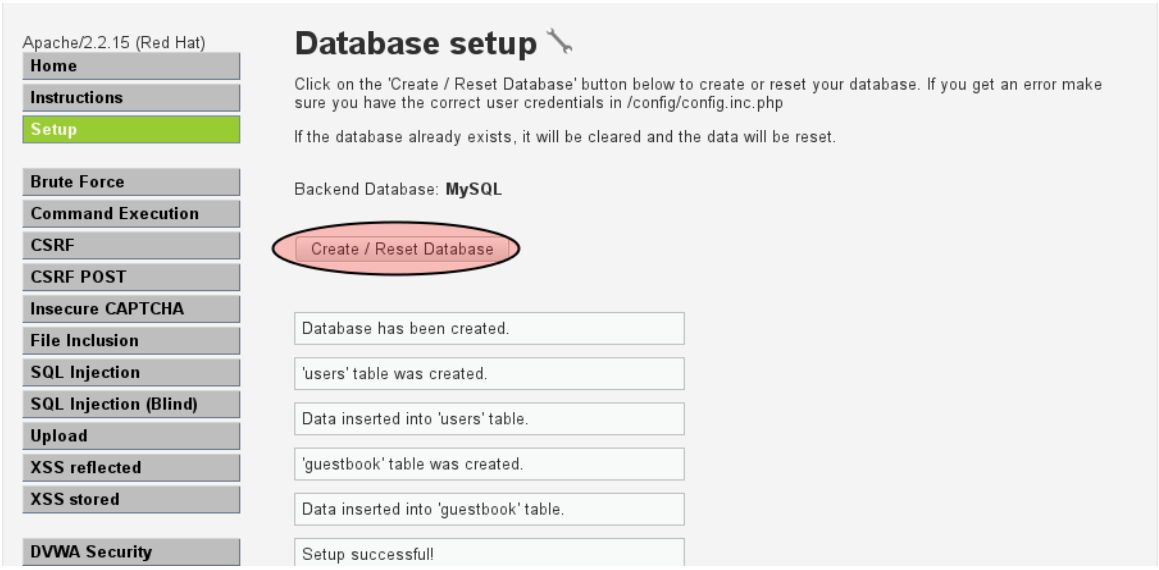


Figura 28: Proceso de inicialización de datos en DVWA [A]

4. Hemos modificado el DVWA para que muestre en la esquina superior izquierda el servidor web que estamos utilizando:

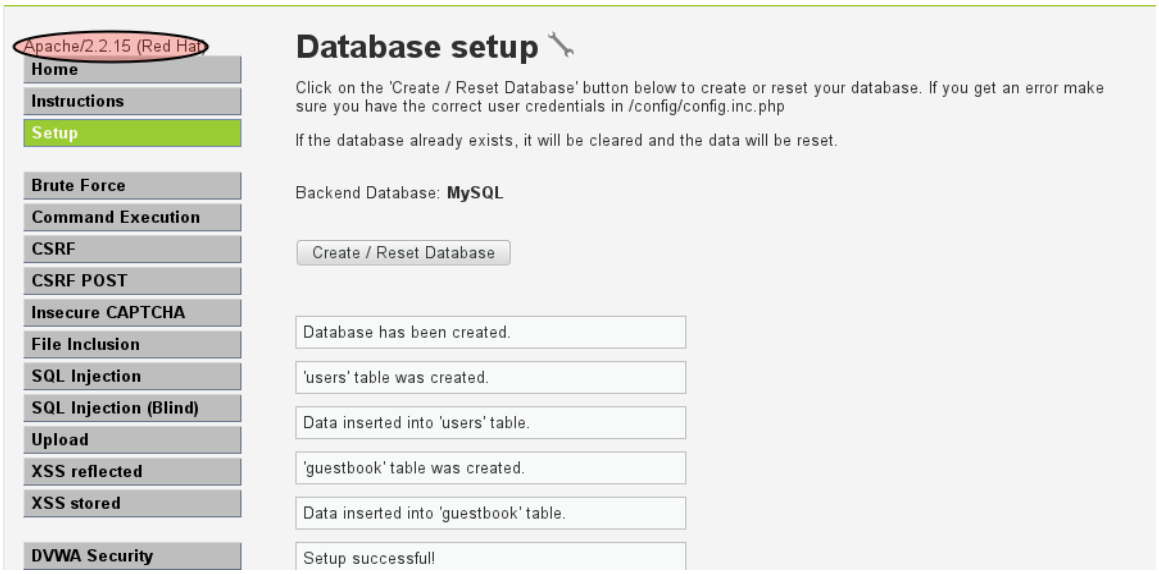


Figura 29: Verificación de tipo de servidor en DVWA [A]

5.3.1 Procedimiento para explotar las vulnerabilidades en DVWA:

Las vulnerabilidades se probarán contra un sitio temporal donde alojamos la aplicación DVWA, este se accede por: http://hc6pe.no-ip.org/dv/

A continuación explicaremos el proceso de explotación de cada una de las vulnerabilidades:

Inyección de SQL (SQL Injection):

El programador de DVWA espera uno introduzca un dato que asignará a la variable "*id*". En la URL podremos notar que pasa por GET el *id* que se le introduce (id=1 en la siguiente figura):

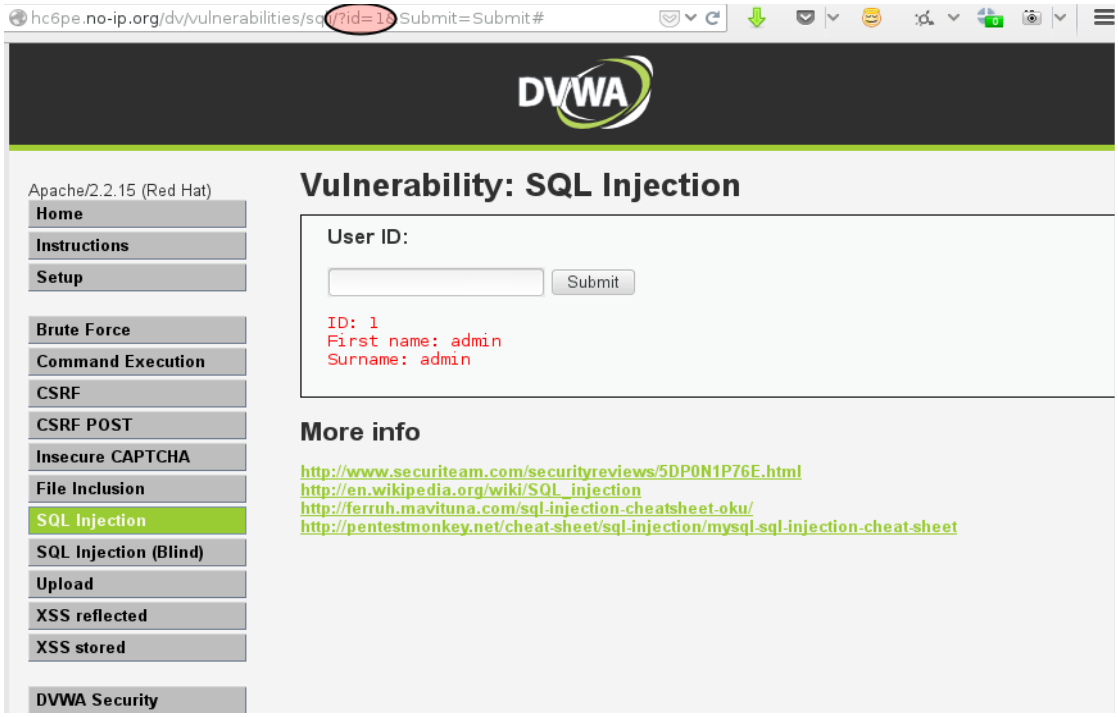


Figura 30: Procedimiento para explotar una Inyección de SQL en DVWA [A]

En vez de un ID como espera el programador, enviaremos sentencias SQL modificando la URL, inyéctandole solicitudes SQL a la variable *id* con la finalidad de obtener información adicional del servidor.

XSS Reflejado (XSS Reflected):

El programador de la apliacción espera que el usuario le escriba un texto simple.

Sin embargo nosotros en el menú *XSS Reflected* le introduciremos la llamada a un script con la finalidad de que lo ejecute.

El script usado para el experimento será solamente una demostración de que se puede insertar código malicioso, consistirá en un mensaje modal de alerta que emitirá un mensaje, por ejemplo: "123" e invocaremos de la siguiente forma:

```
<SCRIPT>alert(123)</SCRIPT>
```

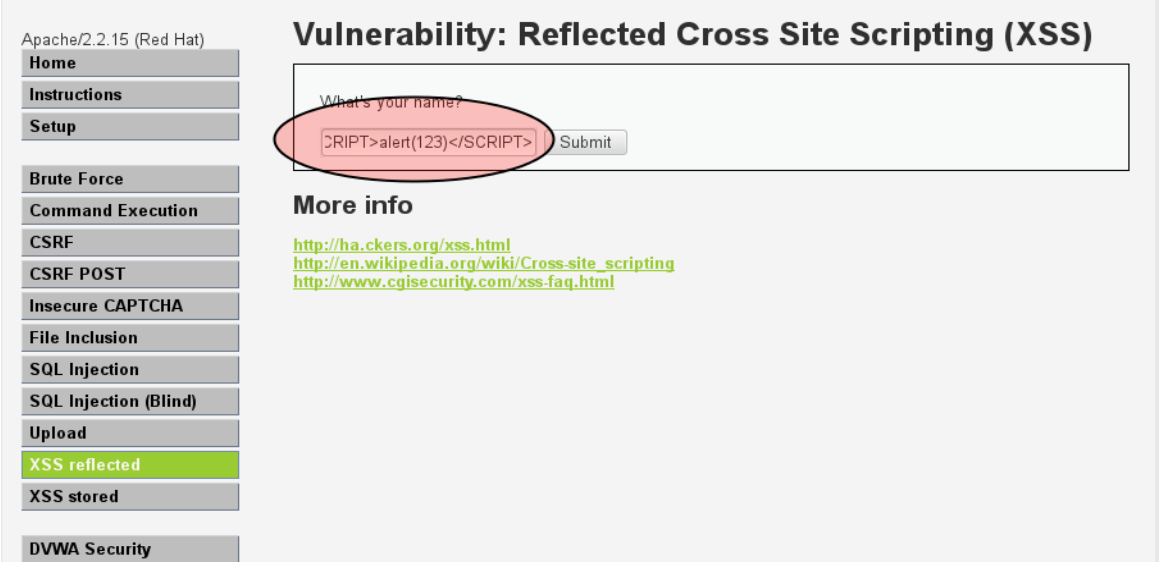


Figura 31: Procedimiento para ejecutar XSS Reflejdo en DVWA [A]

XSS Almacenado (XSS Stored): El programador pretende hacer un bulletin board, también conocido como un foro; en el cual podemos escribir un texto. Esto no es validado y nosotros le enviaremos un script con la finalidad de que a toda persona que entre a este foro, se le ejecute este script.

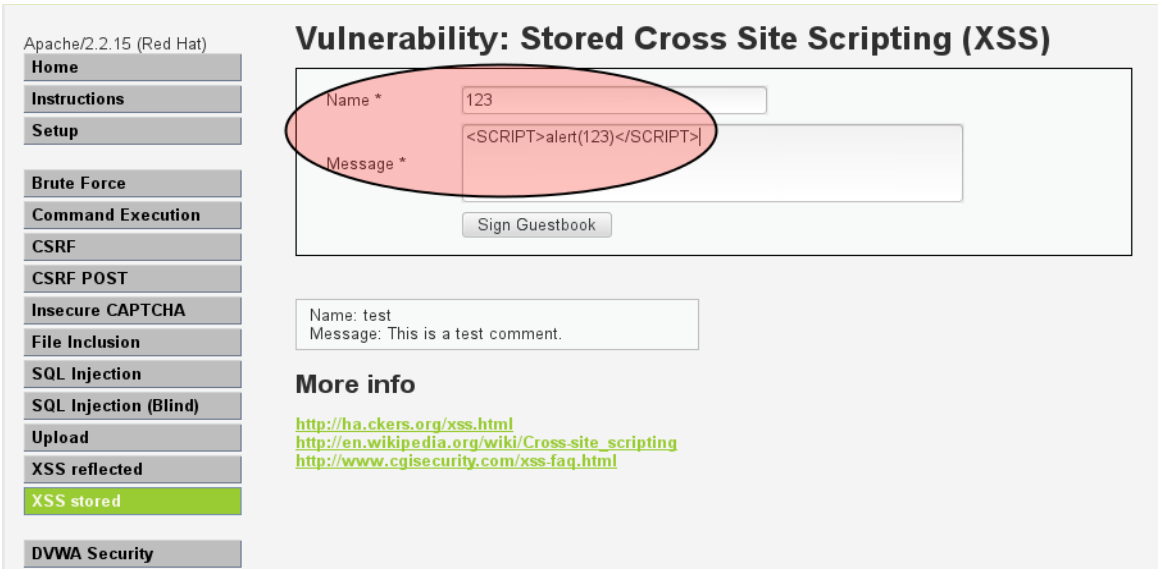


Figura 32: Procedimiento para ejecutar XSS Almacenado en DVWA [A]

CSRF GET:

El programador solicita dos datos que son la nueva clave. Luego pasa por GET los datos de la clave para que se cambie.

Nosotros invocaremos este GET desde otro sitio y, como el usuario está dentro del sistema vulnerable (DVWA), intentaremos cambiar la clave desde el sitio remoto.

Para ello:

- Accederemos a la DVWA y nos mantendremos dentro de ese sistema.
- En el mismo navegador, pero una ventana aparte, procedemos a acceder a una URL de otro sitio la cual está especialmente preparada por un atacante para invocar via GET a la URL para cambiar la clave.
- Para ello creamos en nuestro sitio www.ecualinux.com un html llamado dvget.html que puede ser visto en el Anexo IX. Este archivo dvget.html enviará un GET con una nueva clave desde el sitio remoto (www.ecualinux.com) hacia nuestro servidor de prueba (http://hc6pe.no-ip.org).
- Como resultado final, la DVWA nos indicará si pudo o no cambiar la clave.

CSRF POST:

El programador solicita dos datos que son la nueva clave. Luego pasa por GET los datos de la clave para que se cambie.

Nosotros invocaremos este GET desde otro sitio y, como el usuario está dentro del sistema vulnerable, intentaremos cambiar la clave desde el sitio remoto.

Para ello:

- Accederemos a la DVWA y nos mantendremos dentro de ese sistema.
- En el mismo navegador, pero una ventana aparte, procedemos a acceder a una URL de otro sitio la cual está especialmente preparada por un atacante para invocar via POST a la URL para cambiar la clave.
- Para ello creamos en nuestro sitio www.ecualinux.com un html llamado dvpost.html que puede ser observado en el Anexo X. Este dvpost.html enviará un POST con una nueva clave desde el sitio remoto (www.ecualinux.com) hacia nuestro servidor de prueba (http://hc6pe.no-ip.org).
- Como resultado final, la DVWA nos indicará si pudo o no cambiar la clave.

5.3.2 Pruebas a apache

Se inicia el servicio httpd y se procede a realizar las pruebas de respuesta a vulnerabilidades de Inyección de SQL, XSS y CSRF:

```
service httpd start
```

5.3.2.1 Inyección de SQL

Sustituimos el valor de la variable id por un query que se desea ejecutar, por ejemplo:

`http://hc6pe.no-ip.org/dv/vulnerabilities/sqli/?id=a%27+UNION+SELECT+user%2Cpassword+FROM+users+%23&Submit=Submit#`

Como respuesta obtenemos este listado solicitado

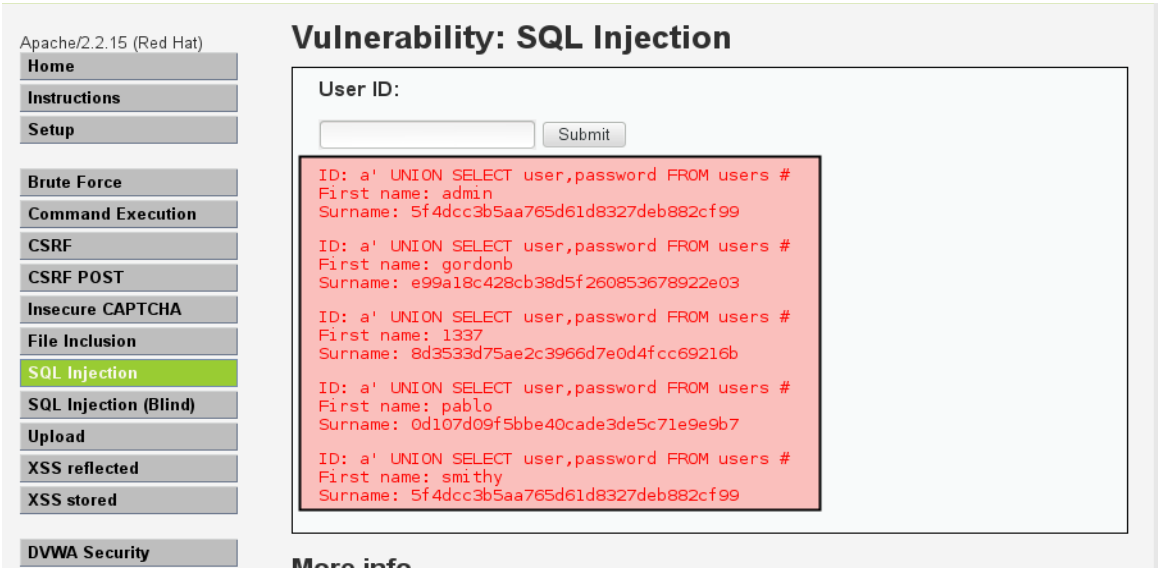


Figura 33: Resultado de ejecución de inyección de SQL en Apache [A]

El ataque de Inyección de SQL fue exitoso.

5.3.2.2 XSS reflejado

Al insertar el SCRIPT, como resultado obtenemos lo siguiente:

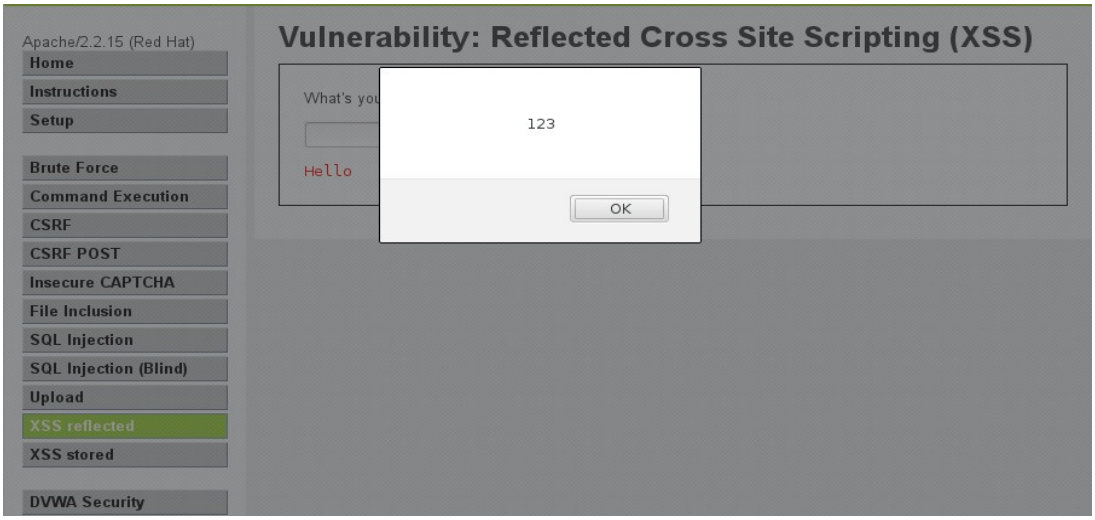


Figura 34: Resultado de ejecución de XSS reflected en apache [A]

El ataque fue exitoso.

5.3.2.3 XSS Almacenado

Insertamos el post con el script, todo usuario que entre a esta URL ejecutará el script:

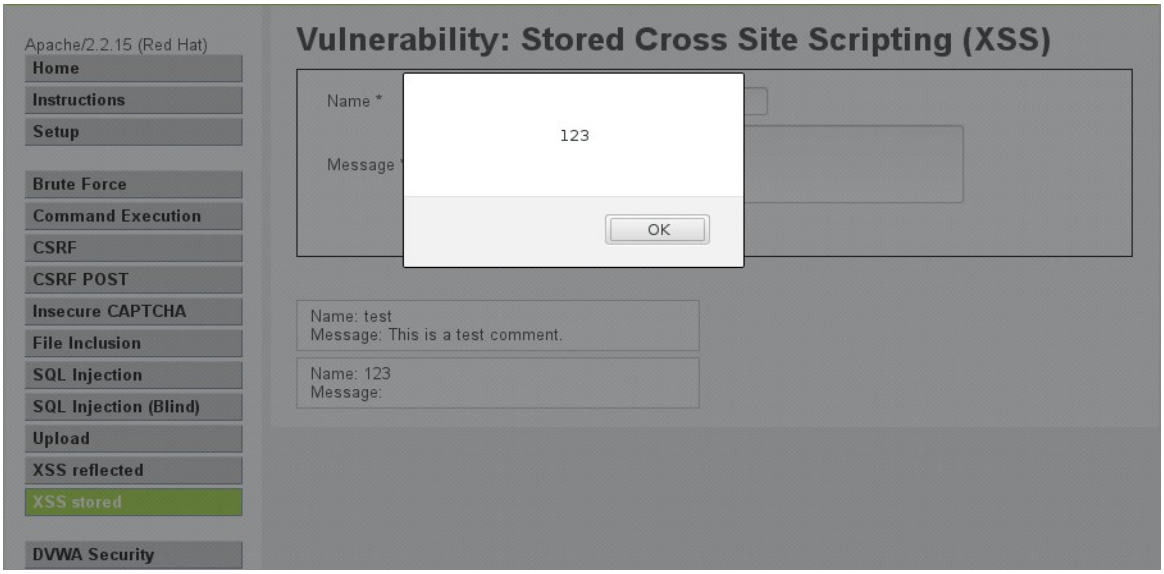


Figura 35: Resultado de ejecución de XSS Stored en apache [A]

El ataque fue exitoso.

5.3.2.4 CSRF GET

Una vez dentro de nuestro servidor de pruebas http://hc6pe.no-ip.org procedemos a acceder al sitio www.ecualinux.com/dvget.html y verificamos que la clave fue cambiada:

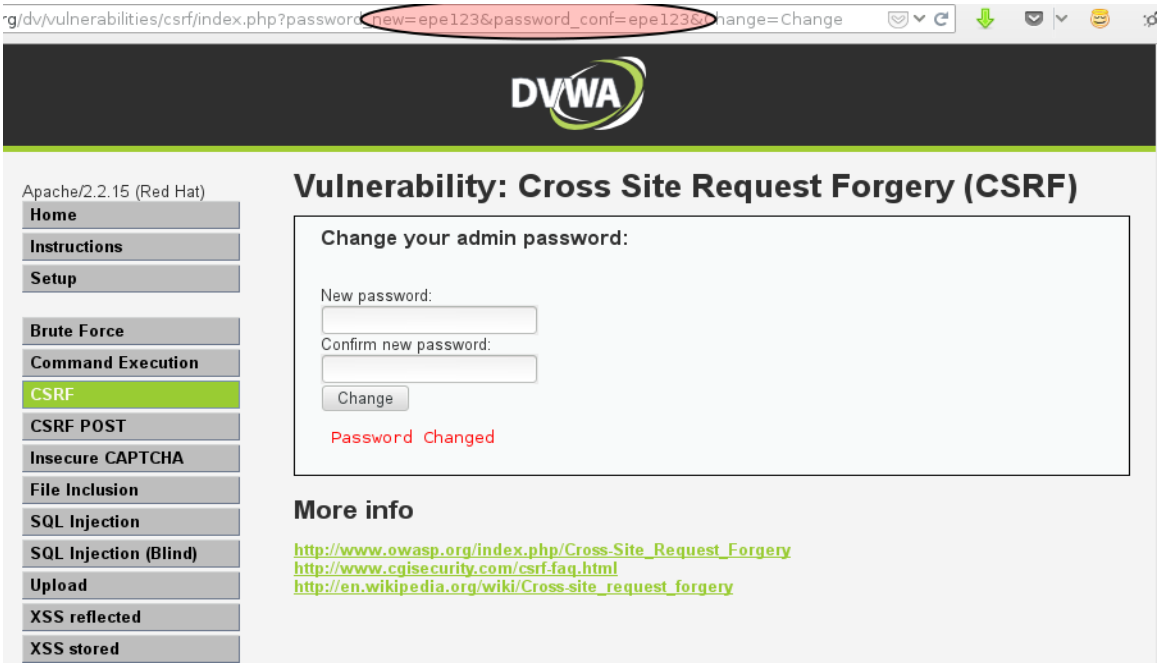


Figura 36: Resultado de ejecución de CSRF GET en Apache [A]

El ataque fue exitoso.

5.3.2.5 CSRF POST

Una vez dentro de nuestro servidor de pruebas http://hc6pe.no-ip.org procedemos a acceder al sitio www.ecualinux.com/dvpost.html y verificamos que la clave fue cambiada. Como estábamos dentro del sistema DVWA se realizará el cambio.

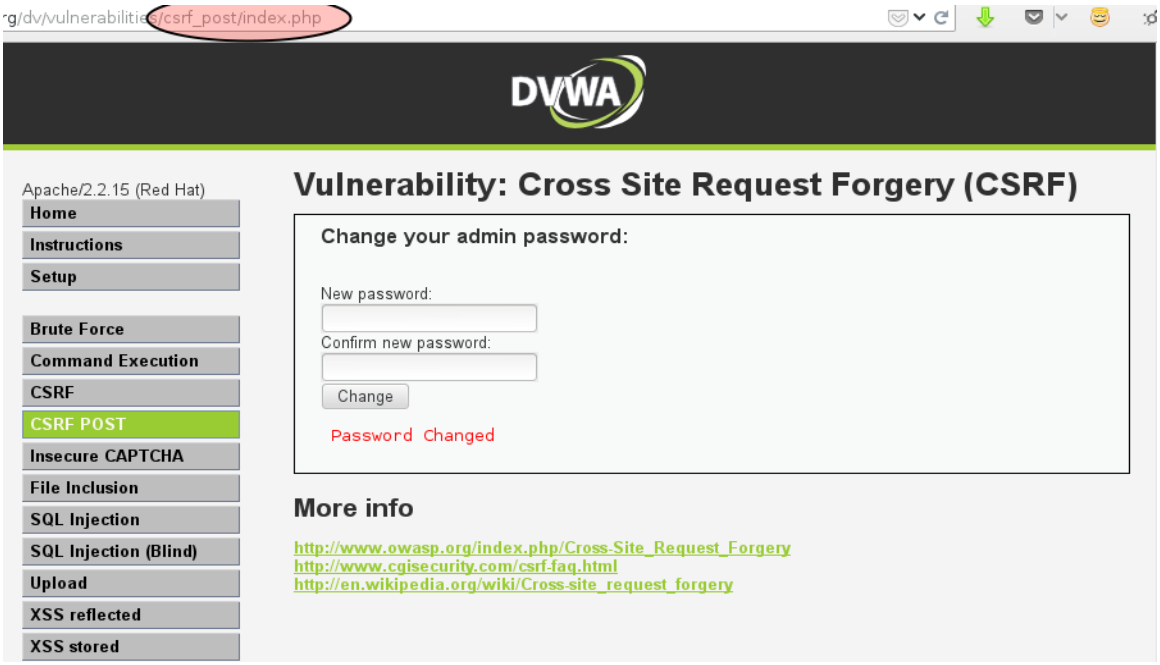


Figura 37: Resultado de ejecución de CSRF POST en Apache [A]

El ataque fue exitoso.

5.3.2.6 Finalizando pruebas con apache

Se apaga el servicio httpd

```
service httpd stop
```

5.3.3 Pruebas a nginx

Se arranca el servicio nginx y se procede a realizar las pruebas de respuesta a vulnerabilidades de Inyección de SQL, XSS y CSRF:

```
service nginx start
```

5.3.3.1 Inyección de SQL

Sustituimos el valor de la variable id por un query que se desea ejecutar, por ejemplo:

http://hc6pe.no-ip.org/dv/vulnerabilities/sqli/?id=**a%27+UNION+SELECT+user%2Cpassword+FROM+users+%23**&Submit=Submit#

Como respuesta obtenemos este listado solicitado:

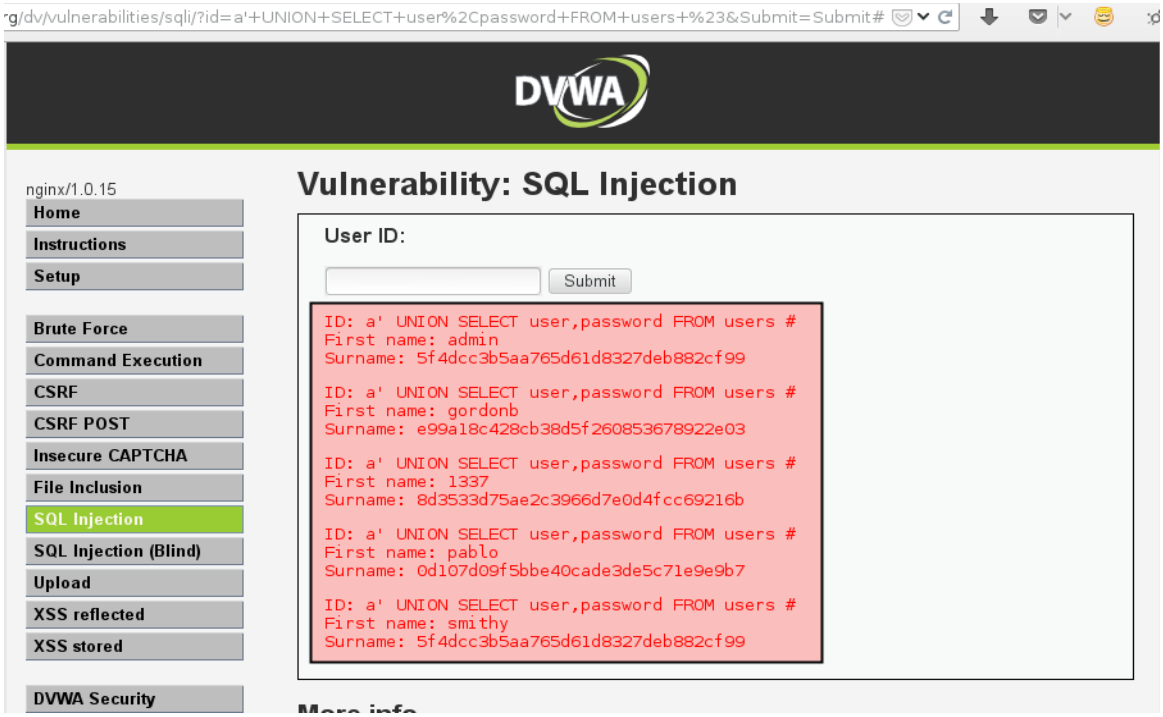


Figura 38: Resultado de ejecución de SQL injection en nginx [A]

El ataque de inyección de SQL fue exitoso.

5.3.3.2 XSS Reflejado

Al insertar el SCRIPT, como resultado obtenemos lo siguiente:

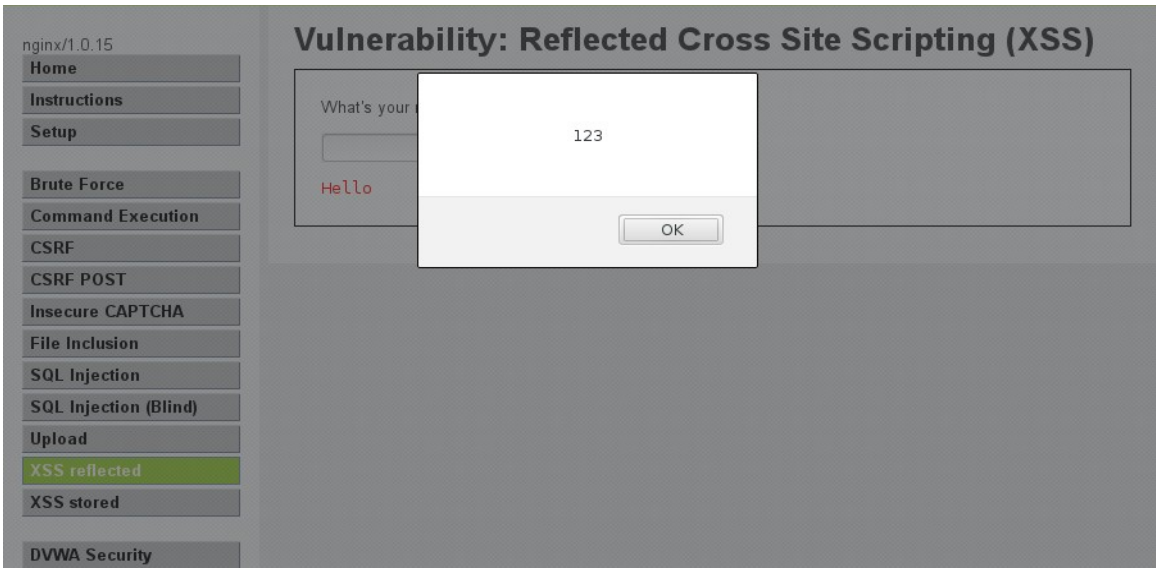


Figura 39: Resultado de ejecución de XSS reflected en nginx [A]

El ataque fue exitoso.

5.3.3.3 XSS Almacenado

Insertamos el post con el script, todo usuario que entre a esta URL ejecutará el script:

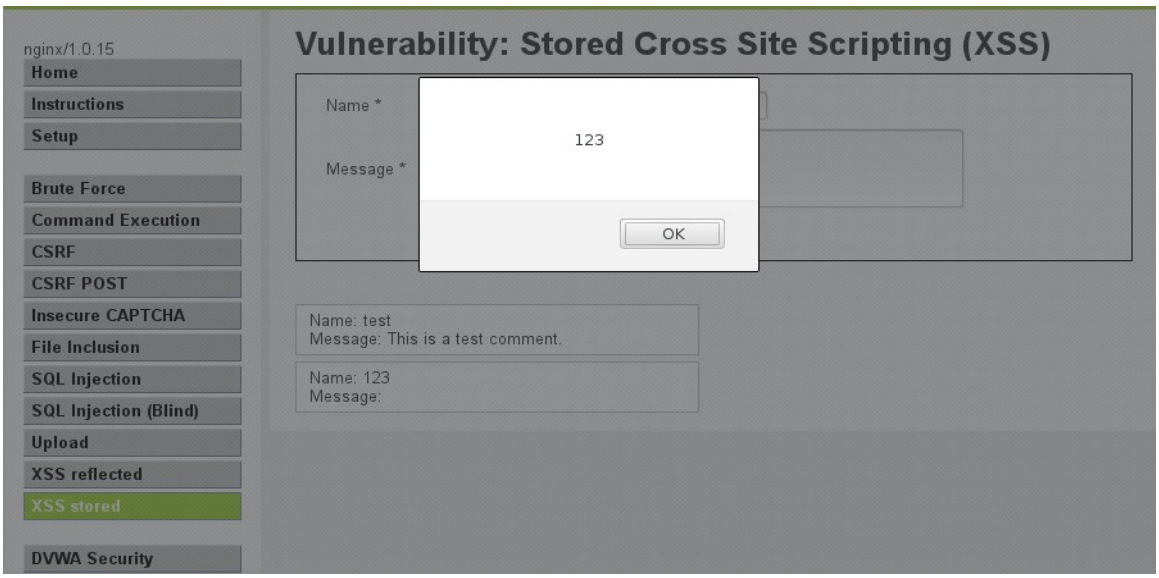


Figura 40: Resultado de ejecución de XSS Stored en nginx [A]

El ataque fue exitoso.

5.3.3.4 CSRF GET

Una vez dentro de nuestro servidor de pruebas http://hc6pe.no-ip.org procedemos a acceder al sitio www.ecualinux.com/dvget.html y verificamos que la clave fue cambiada:

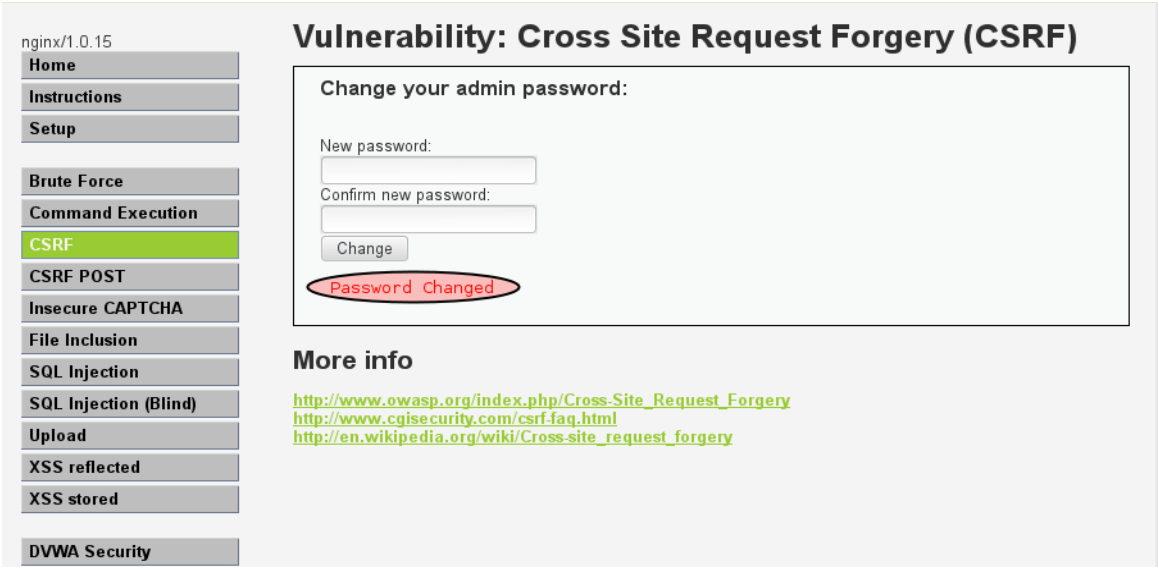


Figura 41: Resultado de ejecución de CSRF GET en nginx [A]

El ataque fue exitoso.

5.3.3.5 CSRF POST

Una vez dentro de nuestro servidor de pruebas http://hc6pe.no-ip.org procedemos a acceder al sitio www.ecualinux.com/dvpost.html y verificamos que la clave fue cambiada. Como estábamos dentro del sistema DVWA se realizará el cambio.

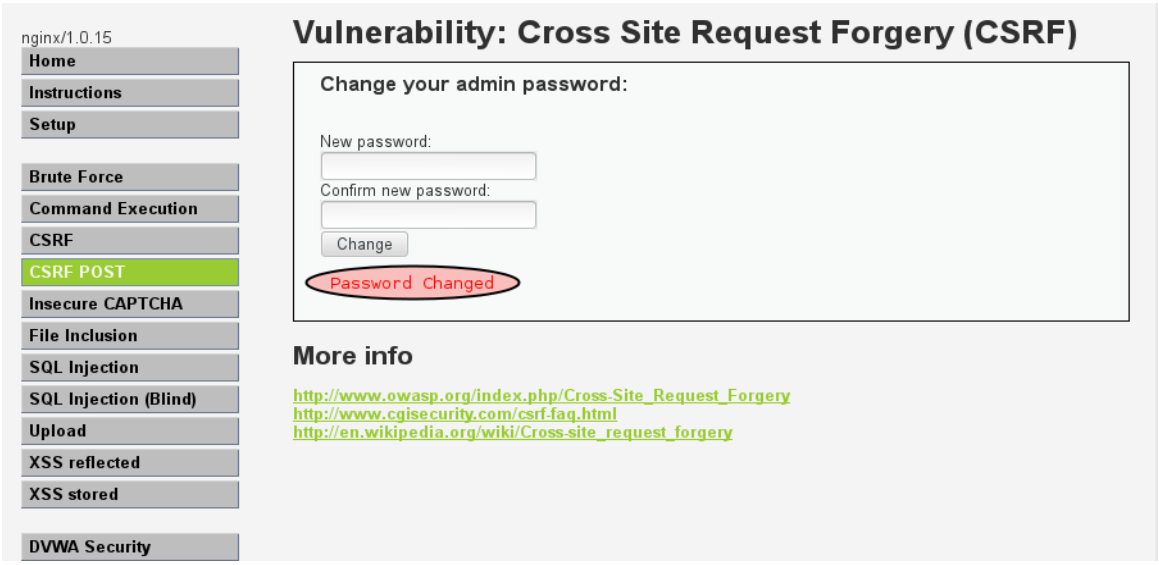


Figura 42: Resultado de ejecución de CSRF POST en nginx [A]

El ataque fue exitoso.

5.3.3.6 Finalizando pruebas con nginx

Se apaga el servicio nginx

```
service nginx stop
```

5.3.4 Pruebas a hiawatha

Se arranca el servicio hiawatha y se procede a realizar las pruebas de respuesta a vulnerabilidades de Inyección de SQL, XSS y CSRF:

```
service hiawatha start
```

5.3.4.1 Inyección de SQL

Sustituimos el valor de la variable id por un query que se desea ejecutar, por ejemplo:

```
http://hc6pe.no-ip.org/dv/vulnerabilities/sqli/?id=a
%27+UNION+SELECT+user%2Cpassword+FROM+users+%23&Submit=Submit#
```

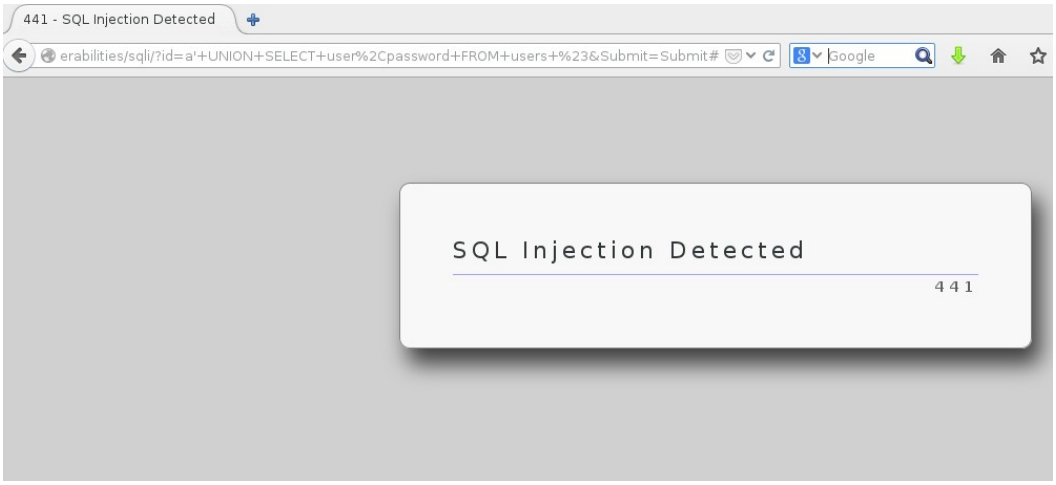


Figura 43: Resultado de ejecución de intento de Inyección de SQL en hiawatha [A]

Como podemos observar, hiawatha no muestra los resultados de este intento de inyección sino que indica que detectó una inyección de sql. En los logs de hiawatha aparece lo siguiente:

```
192.168.1.25|Mon 19 May 2014 20:53:12 -0500|127.0.0.1|/dv/vulnerabilities/sqli/|
SQLi|/dv/vulnerabilities/sqli/?id=a%27+UNION+SELECT+user%2Cpassword+FROM+users+%
23&Submit=Submit
```

Figura 44: Presentación en archivo exploit.log de intento de Inyección de SQL en hiawatha [A]

El ataque no fue exitoso.

5.3.4.2 XSS reflejado

Al insertar el SCRIPT, como resultado obtenemos lo siguiente:

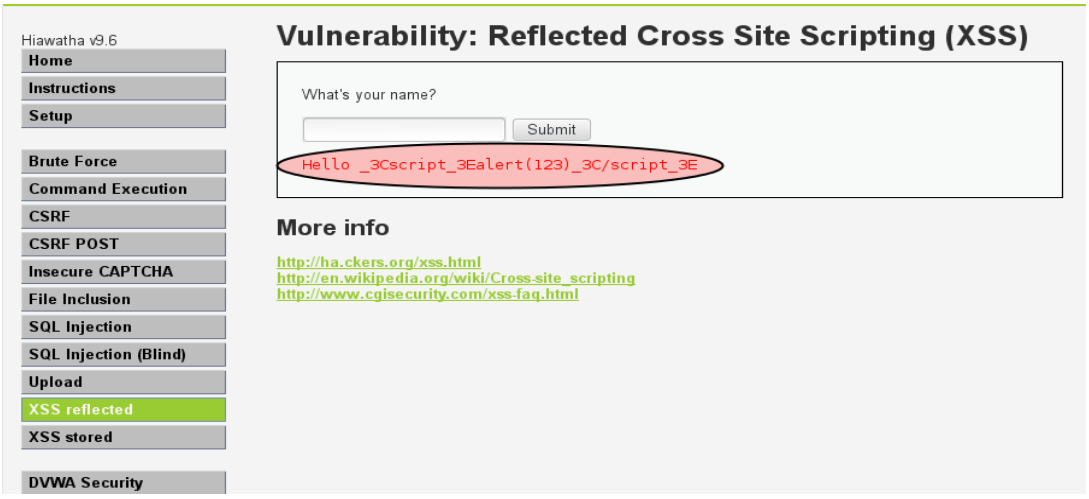


Figura 45: Resultado de ejecución de XSS Reflected en hiawatha [A]

No se obtienen los resultados esperados por un atacante que sería una ventana de Alerta como en los casos anteriores. Lo que se puede observar es que el mensaje fue sanitizado y se convirtieron los caracteres peligrosos en equivalentes no dañinos para un servidor web.

En los logs de hiawatha queda almacenada una alerta de intento de XSS:

```
186.47.253.32|Tue 20 May 2014 21:05:58 -0500|hc6pe.no-ip.org|/dv/vulnerabilities
/xss_r/|XSS|name=%3Cscript%3Ealert%28123%29%3C%2Fscript%3E
```

Figura 46: Presentación en archivo exploit.log de intento de XSS en hiawatha [A]

5.3.4.3 XSS Almacenado

Intentamos almacenar un mensaje que contiene un script que abrirá una ventana de diálogo:

No obteniendo el resultado esperado:

Figura 47: Resultado de ejecución de XSS stored en hiawatha [A]

Como podemos observar, se almacena un mensaje vacío. Y no se emite el mensaje de alerta que se intentó insertar.

5.3.4.4 CSRF GET

En este caso el exploit contra el servidor hiawatha sí funciona.

Para realizar las pruebas accederemos a la URL http://www.ecualinux.com/dvget.html como resultado se puede comprobar que la clave es cambiada:

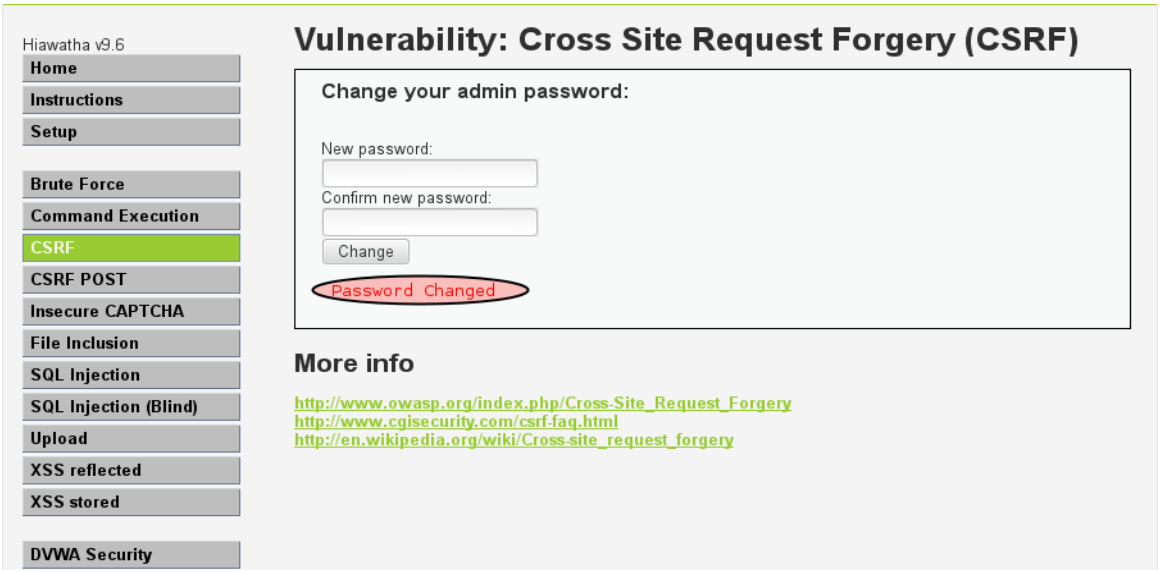


Figura 48: Resultado de ejecución de CSRF GET en hiawatha [A]

5.3.4.5 CSRF POST

Procedemos a acceder a http://www.ecualinux.com/dvpost.html y el cambio no se produce. Esto se evidencia pues no aparece el mensaje en rojo confirmando el cambio de clave.

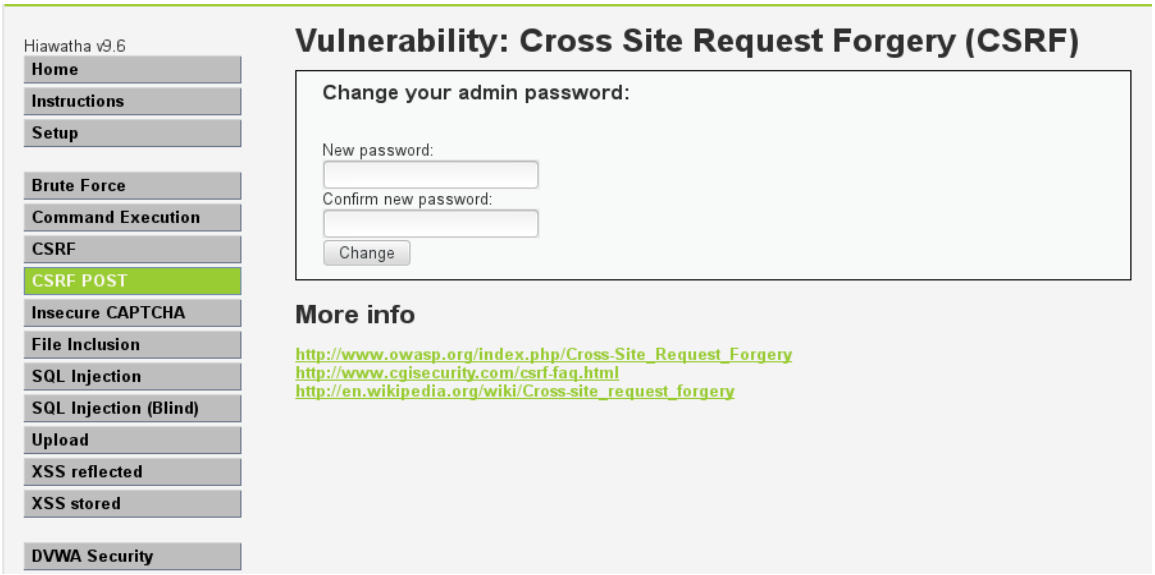


Figura 49: Resultado de ejecución de CSRF POST en hiawatha [A]

Además podemos ver en el archivo /var/log/hiawatha/exploit.log que se guarda un intento de CSRF:

```
186.47.253.32|Tue 20 May 2014 21:14:01 -0500|hc6pe.no-ip.org|/dv/vulnerabilities
/csrf_post/index.php|CSRF|password_new=epe123&password_conf=epe123&Change=Change
```

Figura 50: Presentación en archivo exploit.log de intento de CSRF en hiawatha [A]

5.3.4.6 Finalizando pruebas con hiawatha

Se apaga el servicio hiawatha

```
service hiawatha stop
```

5.4 Análisis de resultados

Como podemos analizar de la información anterior, el servidor hiawatha fue capaz de detectar y evitar intentos de inyección de SQL, XSS reflejado y XSS almacenado, así como CSRF enviados a través de POST desde referers no autorizados. Ni apache ni nginx fueron capaces de detectar estos ataques y ninguno de los dos evidenciaron problemas en sus archivos de log.

Procederemos en las siguientes secciones a utilizar al hiawatha como proxy reverso para proteger a otros servidores web y crearemos un paquete de instalación de hiawatha con configuraciones apropiadas para proteger y optimizar el servicio web.

Servidor	Inyección SQL	XSS Reflejado	XSS Almacenado	CSRF GET	CSRF POST
Apache	No	No	No	No	No
Nginx	No	No	No	No	No
Hiawatha	Sí	Sí	Sí	No	Sí

Tabla 17: Capacidad de detección de ataques en servidores web analizados [A]

5.5 Uso de servidor fortalecido como proxy reverso para proteger otros servidores web

En esta sección se describirá cómo configurar uno de los servidores fortalecidos para actuar como proxy reverso delante de otros servidores web, de forma tal que pueda lograrse mejorar la seguridad de estos.

El proceso se realiza hacia un servidor web instalado en el mismo servidor, se probará tanto apache como nginx como servidor web a proteger. Este proceso se puede realizar igualmente hacia un servidor localizado en otra IP y el servidor web podría ser cualquier servidor web que se necesite proteger.

Para comenzar, se apaga cualquier servidor web que pueda estar encendido en el equipo.

```
service httpd stop
service nginx stop
service hiawatha stop
```

5.5.1 Configuración de proxy reverso en hiawatha

En la sección “binding” del archivo /etc/hiawatha/hiawatha.conf adiciona:

```
Interface = 192.168.1.5
```

Donde 192.168.1.5 es la IP de la tarjeta de red del servidor.

La sección general del archivo /etc/hiawatha/hiawatha.conf se cambia a:

```
Hostname = hc6pe.no-ip.org
WebsiteRoot = /var/www/html
ReverseProxy .* http://127.0.0.1:80/
PreventCSRF = yes
PreventSQLi = yes
PreventXSS = yes
AccessLogfile = /var/log/hiawatha/access.log
ErrorLogfile = /var/log/hiawatha/error.log
```

El cambio más importante es el parámetro: ReverseProxy el cual pasará todas las solicitudes hacia el servidor web a proteger que estará escuchando en nuestros ejemplos en la IP 127.0.0.1

5.5.2 Configuración de servidores http para aceptar conexiones desde proxy reverso

5.5.2.1 Apache

Puesto que en nuestras pruebas ambos servidores estarán corriendo en el mismo equipo, procedemos a configurar al servidor apache para escuchar solamente en el puerto 80/TCP de la IP 127.0.0.1. Puesto que en la IP de la tarjeta de red del servidor estará escuchando hiawatha. Para ello editamos el archivo: /etc/httpd/conf/httpd.conf y se cambia la opción:

```
Listen 80
```

Por:

```
Listen 127.0.0.1:80
```

Se reinicia el servidor apache.

```
service httpd restart
```

Se verifica que el apache esté corriendo solamente en 127.0.0.1 con el comando:

```
netstat -ntupa|egrep http
```

5.5.3 nginx

Puesto que en nuestras pruebas ambos servidores estarán corriendo en el mismo equipo, procedemos a configurar al servidor apache para escuchar solamente en el puerto 80/TCP de la IP 127.0.0.1. Puesto que en la IP de la tarjeta de red del servidor estará escuchando hiawatha. Para ello editamos el archivo: /etc/nginx/conf.d/default.conf y se cambia la opción:

```
listen       80 default_server;
```

a:

```
listen       127.0.0.1:80 default_server;
```

Se reinicia el servidor nginx

```
service nginx restart
```

Se verifica que nginx esté escuchando solamente en localhost:

```
netstat -ntupa|egrep nginx
```

5.5.4 Pruebas con hiawatha como proxy reverso a Apache

Luego de configurar hiawatha como proxy reverso, se realizaron las mismas pruebas con apache de servidor web detrás del proxy reverso. Debemos destacar que hiawatha impidió los ataques de Inyección de SQL, XSS y CSRF POST. Podemos notar en cada una de las imágenes que el nombre del servidor web es Apache pues hiawatha no modifica los headers para cambiar el nombre del servidor.

Para comenzar las pruebas procedemos a apagar nginx y encender hiawatha y apache:

```
service nginx stop
service apache start
service hiawatha start
```

5.5.4.1 Inyección de SQL

Como podemos observar, el hiawatha utilizado como proxy reverso de un servidor apache impide la ejecución de la inyección de SQL.

Este es el mensaje típico de hiawatha y no de apache. Hiawatha ni siquiera muestra la pantalla de DVWA puesto que ante un ataque de inyección de SQL, como pudimos observar anteriormente, emite un mensaje de error 441 para bloquear el intento de intrusión.

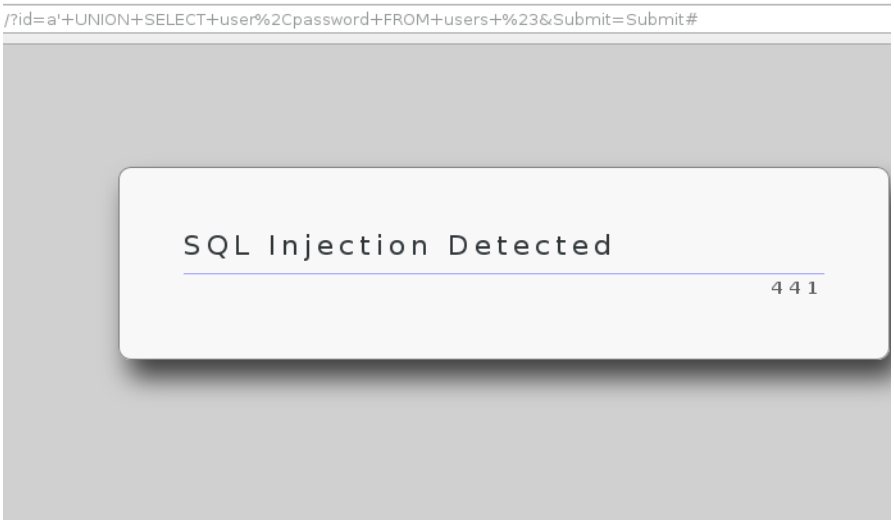


Figura 51: Resultado de ejecución de Inyección de SQL con hiawatha como proxy reverso a apache [A]

5.5.4.2 XSS reflejado

Como podemos observar, hiawatha sanitiza la entrada evitando la inserción de XSS

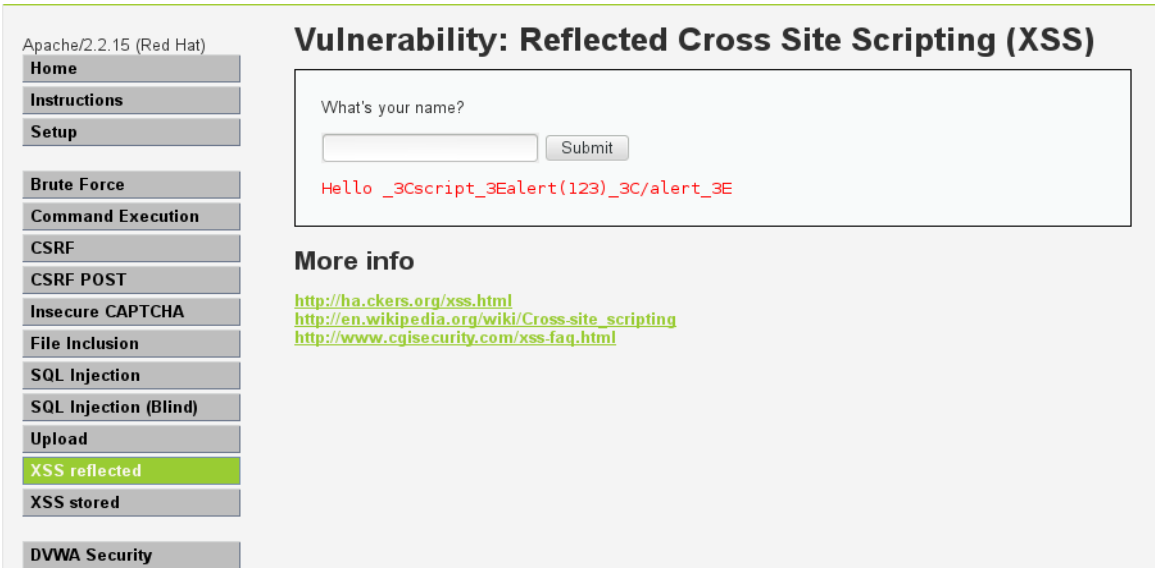


Figura 52: Resultado de ejecución de XSS reflejado con hiawatha como proxy reverso a apache [A]

5.5.4.3 XSS almacenado

El intento de insertar XSS almacenado es sanitizado por hiawatha:

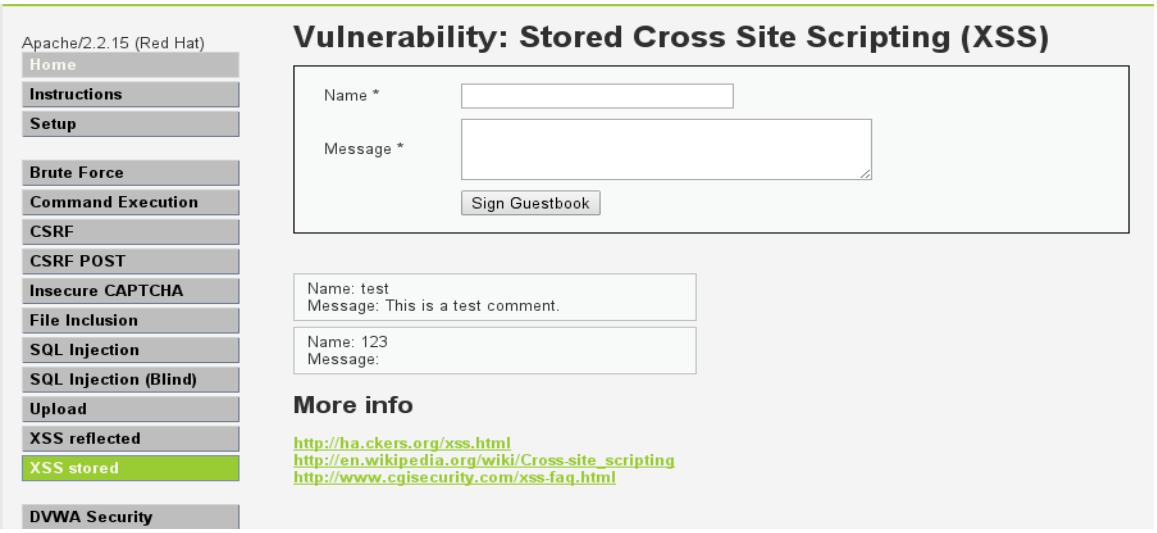


Figura 53: Resultado de ejecución de XSS almacenado usando hiawatha como proxy reverso a apache [A]

5.5.4.4 CSRF GET

En este caso, hiawatha como proxy tiene el mismo comportamiento que cuando se intentó ejecutar CSRF GET contra hiawatha directamente. Es decir, sí lo permitió:

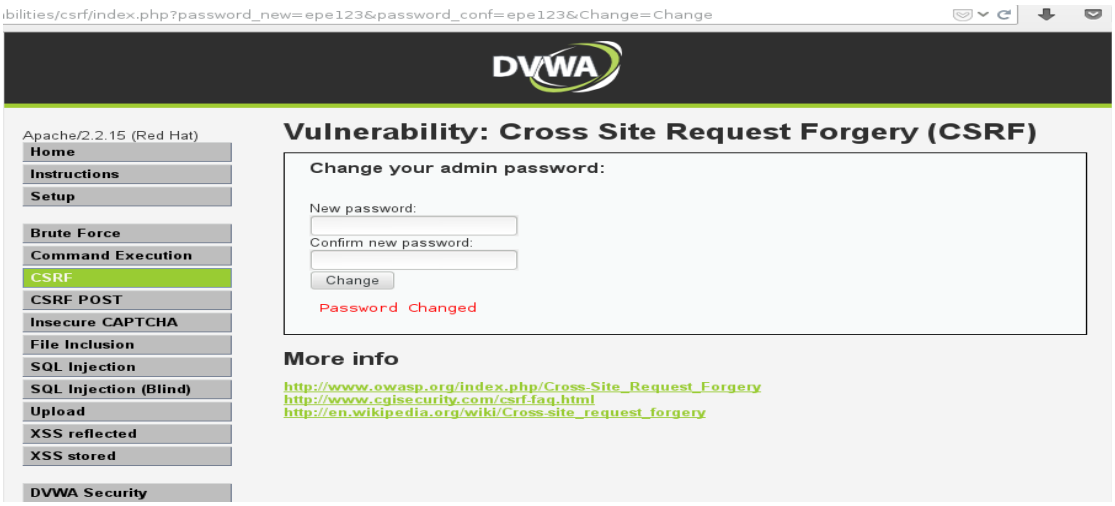


Figura 54: Intento de ejecución de CSRF GET con hiawatha como proxy reverso a apache [A]

5.5.4.5 CSRF POST

Hiawatha efectivamente bloquea el intento de CSRF, incluso como podemos observar, impide la ejecución del script:

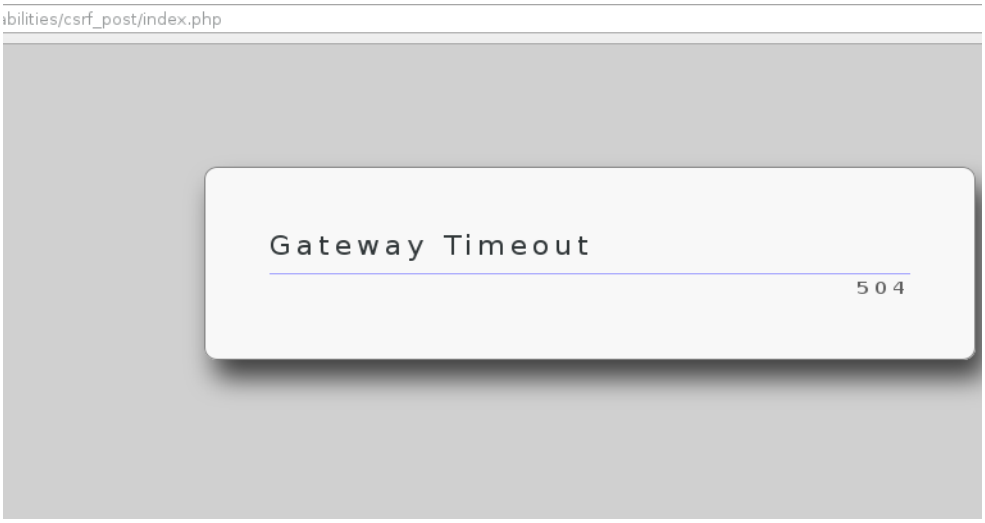


Figura 55: Intento de ejecución de CSRF POST con hiawatha como proxy reverso a apache [A]

5.5.5 Pruebas con hiawatha como proxy reverso a nginx

Luego de configurar hiawatha como proxy reverso, se realizaron las mismas pruebas con nginx de servidor web detrás del proxy reverso. Debemos destacar que hiawatha impidió los ataques de Inyección de SQL, XSS y CSRF POST. Podemos notar en cada una de las imágenes que el nombre del servidor web es nginx pues hiawatha no modifica los headers para cambiar el nombre del servidor.

Para comenzar las pruebas procedemos a apagar httpd y encender hiawatha y nginx:

```
service httpd stop
service nginx start
service hiawatha start
```

5.5.5.1 Inyección de SQL

En esta caso el intento no funciona.

Como podemos observar, hiawatha bloqueó el intento de inyección de SQL protegiendo de esta forma a la aplicación vulnerable que está siendo ejecutada desde nginx:

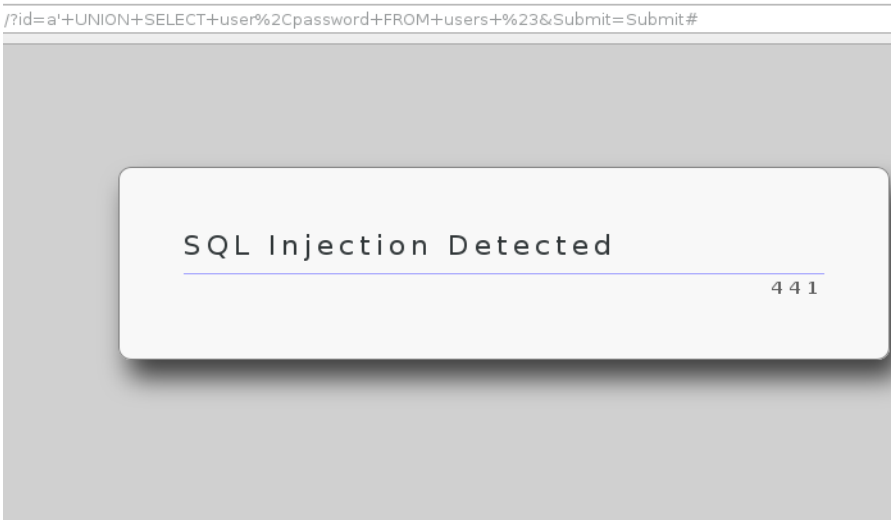


Figura 56: Intento de ejecución de Inyección de SQL con hiawatha como proxy reverso a nginx [A]

5.5.5.2 XSS reflejado

Hiawatha sanitiza la entrada de forma tal que se impide el XSS:

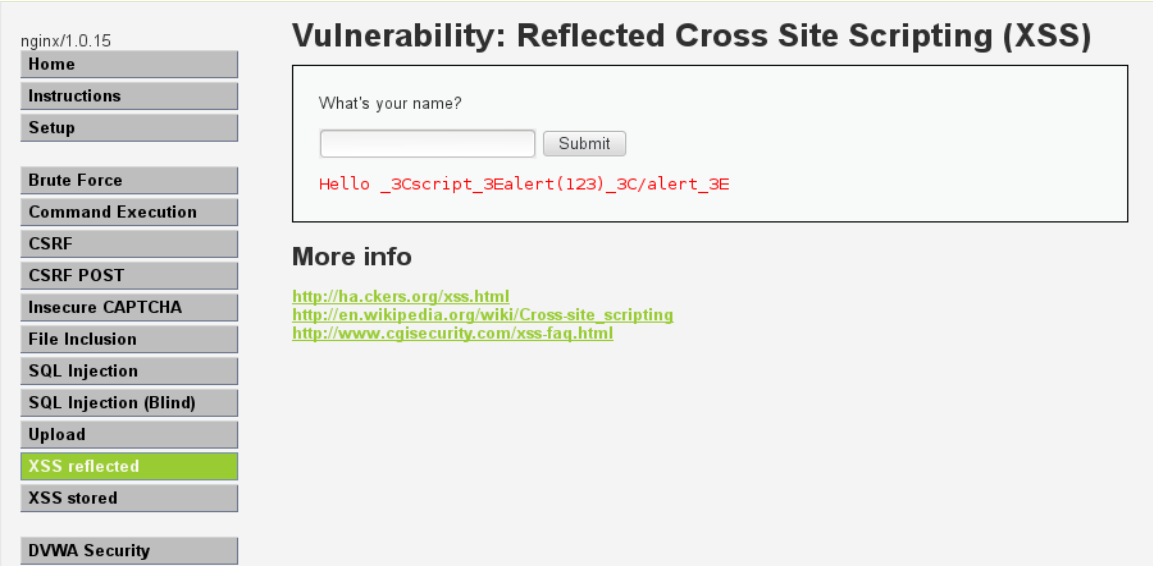


Figura 57: Intento de ejecución de XSS Reflejado con hiawatha como proxy reverso a nginx [A]

5.5.5.3 XSS almacenado

Hiawatha sanitiza la entrada de forma tal que se impide el XSS:

Figura 58: Intento de ejecución de XSS Almacenado con hiawatha como proxy reverso a nginx [A]

5.5.5.4 CSRF GET

En este caso, hiawatha como proxy tiene el mismo comportamiento que cuando se intentó ejecutar CSRF GET contra hiawatha directamente. Es decir, sí lo permitió:

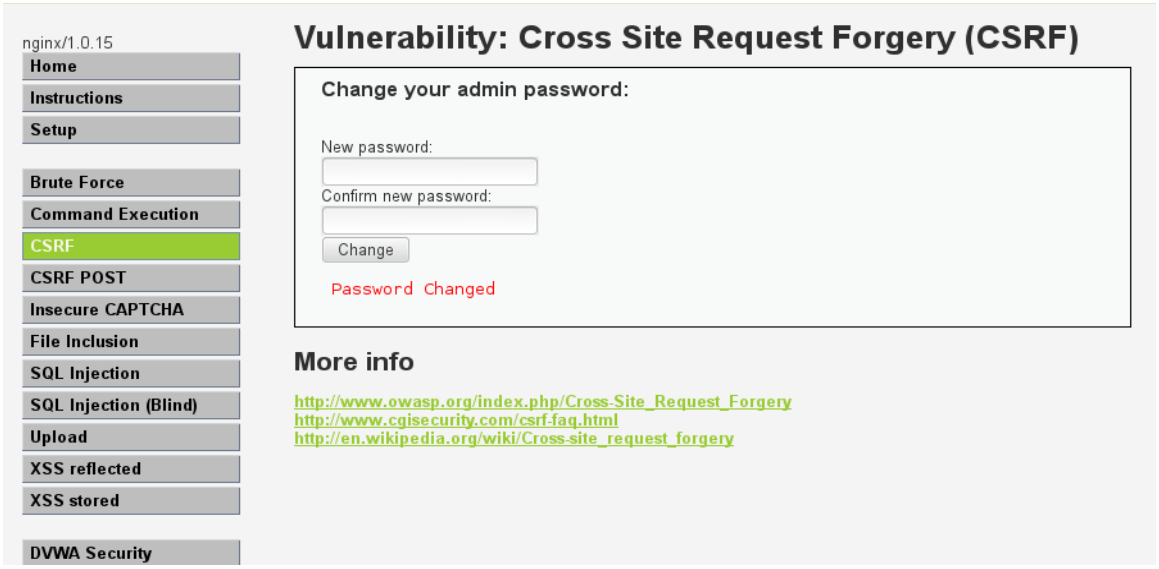


Figura 59: Intento de ejecución de CSRF GET con hiawatha como proxy reverso a nginx [A]

5.5.5.5 CSRF POST

Hiawatha efectivamente bloquea el intento de CSRF, incluso como podemos observar, impide la ejecución del script:

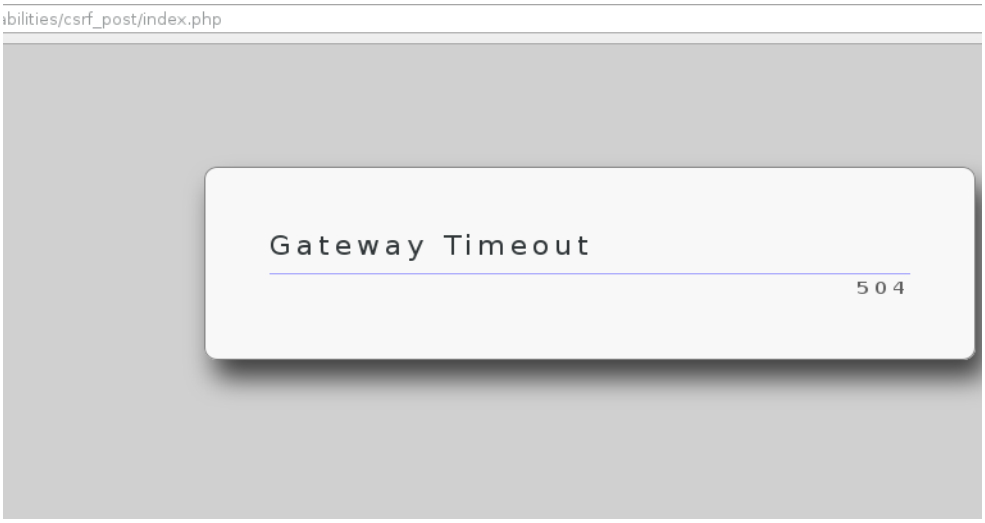


Figura 60: Intento de ejecución de CSRF POST con hiawatha como proxy reverso a nginx [A]

5.6 Creación de instalador configurado por defecto para proteger y optimizar servicio web.

En vista de las prestaciones que ofrece hiawatha se ha creado un paquete de instalación del servidor hiawatha en su última versión actual. Este paquete viene con las seguridades configuradas por defecto en su archivo /etc/hiawatha/hiawatha.conf

El paquete podrá ser obtenido en los anexos de la tesis y del sitio: http://centos6.ecualinux.com/ se han creado versiones para 32 y para 64bits.

Capítulo 6 : Conclusiones y recomendaciones

Al finalizar nuestra investigación podemos llegar a las siguientes conclusiones:

- Se determinó que existen riesgos que son activamente aprovechados con la finalidad de obtener información sobre sitios web, sobre usuarios de un sitio web, o modificar los contenidos o datos de un sitio.
- Del análisis de las desfiguraciones realizadas a sitios ecuatorianos, determinamos que la mayoría de estos corren en apache o nginx, dándonos una clara indicación de que estos servidores no solamente permiten ataques que conllevan a la desfiguración sino que además pueden y deben ser asegurados.
- Es posible detectar ataques que aprovechan vulnerabilidades tipo XSS, CSRF e Inyección de SQL
- El servidor web hiawatha webserver detecta y bloquea intentos de aprovechar vulnerabilidades XSS, CSRF e Inyección de SQL realizadas contra aplicaciones web dinámicas vulnerables.
- Aún cuando existen medidas paliativas como el uso de reescrituras o WAF, los servidores Apache y nginx no permiten la detección y bloqueo de estos intentos.
- Se analizaron, probaron y describieron las características de seguridad disponibles en servidores web en entornos de Software Libre que les permite detectar, informar e incluso bloquear intentos de intrusión.
- Mediante el uso de una aplicación vulnerable de pruebas, se documentó la respuesta que los servidores estudiados daban a los intentos de aprovechamiento de intentos de intrusión utilizando estas vulnerabilidades
- Se estudió el estado del arte en la reacción ante ataques contra servidores web en Software Libre encontrándose que existe una amplia gama de posibilidades para detectar, mitigar, bloquear o eliminar estos intentos de intrusión que van desde una programación con énfasis en la seguridad, fortalecimiento del Sistema Operativo, selección de un servidor web que brinde estas protecciones, a aplicaciones de usuario que detecten y bloqueen estos intentos.
- Se determinó un servidor web que puede detectar, mitigar o bloquear intentos de intrusión aprovechando vulnerabilidades en las aplicaciones dinámicas; describiéndose el proceso para instalarle.
- Se creó y publicó un paquete de instalación del servidor web orientado a la seguridad de forma tal que las personas interesadas en mejorar la seguridad de sus aplicaciones dinámicas puedan instalarle y aprovechar sus características de seguridad.

- Este trabajo aporta un procedimiento que permite mejorar la seguridad de las aplicaciones web dinámicas que puede ser utilizado por el personal informático interesado en mejorar la seguridad de sus redes de comunicaciones.
- Con este trabajo además se consiguió probar las seguridades de los servidores web analizados.
- El autor del servidor web hiawatha pudo incorporar una nueva característica detectar ataques de CSRF conocida como el header Origin: la cual implementó gracias a nuestros estudios.
- Además se hicieron ajustes al servidor web hiawatha que salieron de forma oficial en la versión 9.6 en base a hallazgos realizados durante nuestro estudio.
- Se verificó que el servidor hiawatha webserver puede ser utilizado como proxy reverso con la finalidad de proteger a otros servidores web de instituciones que no pueden dejar de utilizar sus servidores web actuales.
- Para futuros trabajos se sugiere que se realicen estudios basados en las tasas de desfiguraciones de sitios web en el país y su relación con la versión y el tipo del servidor web utilizado.
- Este estudio se realizó sobre sitios que corren en redes IPv4, sugerimos además validar estos estudios cuando se usen redes IPv6 y confirmar o no que estos datos son consistentes con los estudiados en redes IPv4.
- La maestría nos permitió refrescar y ampliar nuestros conocimientos sobre temas relacionados con las redes de comunicación en campos necesarios para nuestro desarrollo profesional futuro.
- La Pontificia Universidad Católica del Ecuador ofrece un ambiente apropiado para el aprendizaje y desarrollo de habilidades.
- Sugerimos que en vista de que Apache y nginx son servidores web utilizados ampliamente en las redes de nuestro país y de que ocurren casos de ataques a sitios web provocados por inadecuada programación de los sitios web, debe tratar de implementarse servicios que protegan a los servidores web ya instalados siendo hiawatha una opción a tomar en cuenta para protegernos de estos ataques.
- Debe insistirse en auditorías de código y de configuración de servidores con la finalidad de fortalecer estos dos elementos que son claves al exponer un sitio web a internet.
- El personal administrativo y programadores deben estar adecuadamente preparados en técnicas de seguridad para mitigar o eliminar el impacto que pueden causar vulnerabilidades como las antes estudiadas.

- El personal de administración de la red puede tomar las sugerencias indicadas en esta tesis para fortalecer los servidores de sus redes.
- Con la finalidad de mantener actualizada la información recomendamos realizar posteriores análisis de respuesta de servidores web ante ataques a aplicaciones dinámicas.
- En vista de la importancia de la seguridad en las redes, los docentes de la Maestría en Redes de Comunicación de la PUCE deben hacer énfasis en aspectos de seguridad.

Bibliografía

Libros

[A] Estudio de las características de seguridad de servidores web en entornos de Software Libre aplicables a la protección de sitios dinámicos, Ernesto Pérez Estévez, Pontificia Universidad Católica del Ecuador, Quito, 2014

Internet

[1] Ecuador, Decreto ejecutivo 1014, 10 de Abril de 2008, Registro Oficial, 23 de Abril de 2008, num 322, p1

[2] International Telecommunications Union. ITU ACTIVITIES RELATED TO CYBERSECURITY [en línea]. Disponible en Web: http://www.itu.int/cybersecurity/ [consulta: 7 enero 2014].

[3] Zone-H. "Unrestricted Information" [en línea]. Disponible en Web: http://zone-h.org [consulta: 7 enero 2014]

[4] ecuadorecuatoriano. “Anonymous hackeó la página web de la Presidencia de la República de Ecuador” [en línea]. Disponible en web: http://ecuadorecuatoriano.blogspot.com/2011/08/anonymous-hackeo-la-pagina-web-de-la.html [consulta: 7 enero 2014]

[5] PÉREZ Ernesto. “Imágenes de ayer, nuevamente defacement y ataques a sitios del gobierno, estado y personas públicas de Ecuador” [en línea]. Disponible en Web: http://ernestoperez.com/?p=1240 [consulta: 7 enero 2014]

[6] PÉREZ Ernesto. “10 de Agosto y la inseguridad” [en línea]. Disponible en Web: http://ernestoperez.com/?p=1300 [consulta: 7 enero 2014]

[7] PUENTE Guerrero David Esteban, Implementación de Seguridad en el Servidor de Hosting XpertSites.com, UNIVERSIDAD SAN FRANCISCO DE QUITO, 21 Enero 2013

[8] Villamagua Armijos, Gabriel Ignacio, DESARROLLO DE UN SISTEMA DE SEGURIDAD DE ACCESO PARA APLICACIONES WEB. CASO DE ESTUDIO: SISTEMAS BANCARIOS, Escuela Politécnica Nacional, Facultad de Ingeniería de Sistemas, Enero 2013

[9] NetCraft, “Web server developers: Market share of active sites” [en línea]. Disponible en Web: http://news.netcraft.com/archives/2014/01/03/january-2014-web-server-survey.html [consulta: 7 enero 2014]

[10] Organización de Estados Americanos, “Incident Response Methodology” [en línea]. http://www.oas.org/cyber/documents/IRM-6-Website-Defacement.pdf [consulta: 8 de enero del 2014]

[11] Tim Berners-Lee, "Information Management: A Proposal" [en línea]. http://www.w3.org/History/1989/proposal.html [consulta: 16 de Mayo del 2014]

[12] CERT/CC, "Two Input Validation Problems In FTPD" [en línea]. http://www.cert.org/historical/advisories/CA-2000-13.cfm [consulta: 16 de Mayo del 2014]

[13] Michael Sparks, "WU-FTP your way to root shell access through WU-FTP vulnerabilities", SANS Institute, 2000-2002

[14] CVE, "CVE-2004-0148" [en línea]. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2004-0148 [consulta: 16 de Mayo del 2014]

[15] OWASP, "Top 10 2013/ProjectMethodology" [en línea]. www.owasp.org/index.php/Top_10_2013/ProjectMethodology [consulta: 16 de Mayo del 2014]

[16] OWASP, "Top 10 2013" [en línea]. http://www.owasp.org/index.php/Top_10_2013 [consulta: 16 de Mayo del 2014]

[17] Coordinación de Seguridad de la Informacion, "Aspectos Básicos de la Seguridad en Aplicaciones Web" [en línea]. http://www.seguridad.unam.mx/documento/?id=17 [consulta: 16 de Mayo del 2014]

[18] OWASP, "OWASP: Top Ten Project" [en línea]. http://www.owasp.org/index.php/Top10 [consulta: 16 de Mayo del 2014]

[19] OWASP, "OWASP: Top Ten 2007" [en línea]. http://www.owasp.org/index.php/Top_10_2007 [consulta: 16 de Mayo del 2014]

[20] Dave Wichers, "OWASP_Top-10_2013 – Changes-from-2010.pptx", 2013, página 7 [en línea]. http://owasptop10.googlecode.com/files/OWASP_Top-10_2013%20-%20Changes-from-2010.pptx [consulta: 16 de Mayo del 2014]

[21] Dave Wichers, "OWASP Top 10 - 2010 Presentation", 2010, página 5 [en línea]. http://owasptop10.googlecode.com/files/OWASP_Top_10_-_2010%20Presentation.pptx [consulta: 16 de Mayo del 2014]

[22] OWASP, "Injection Flaws" [en línea]. http://www.owasp.org/index.php/Injection_Flaws [consulta: 16 de Mayo del 2014]

[23] OWASP, "Top 10 2013-A2-Broken authentication and Session Management" [en línea]. http://www.owasp.org/index.php/Top_10_2013-A2-Broken_Authentication_and_Session_Management [consulta: 16 de Mayo del 2014]

[24] OWASP, "Top 10 2013-A3-Cross-Site Scripting (XSS)" [en línea]. https://www.owasp.org/index.php/Top_10_2013-A3-Cross-Site_Scripting_%28XSS%29

[25] OWASP, "DOM Based XSS" [en línea]. [consulta: 16 de Mayo del 2014] http://www.owasp.org/index.php/DOM_Based_XSS

[26] Ivan Ristic, "Protecting Web Applications from Universal PDF XSS". 6Th OWASP AppSec Conference, May 2007

[27] Amit Klein, "DOM Based Cross Site Scripting or XSS of the Third Kind A look at an overlooked flavor of XSS" [en línea]. http://www.webappsec.org/projects/articles/071105.shtml [consulta: 16 de Mayo del 2014]

[28] OWASP, "XSS Filter Evasion Cheat Sheet" [en línea]. https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet [consulta: 16 de Mayo del 2014]

[29] OWASP, "Top 10 2013-A4-Insecure Direct Object References" [en línea]. http://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References [consulta: 16 de Mayo del 2014]

[30] OWASP, "Top 10 2013-A5-Security Misconfigurations" [en línea]. http://www.owasp.org/index.php/Top_10_2013-A5-Security_Misconfiguration [consulta: 16 de Mayo del 2014]

[31] OWASP, "Top 10 2013-A6-Sensitive Data Exposure" [en línea]. http://www.owasp.org/index.php/Top_10_2013-A6-Sensitive_Data_Exposure

[32] CVE, "CVE-2014-160" [en línea]. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160 [consulta: 16 de Mayo del 2014]

[33] OWASP, "Top 10 2013-A7-Missing Function Level Access Control" [en línea]. http://www.owasp.org/index.php/Top_10_2013-A7-Missing_Function_Level_Access_Control [consulta: 16 de Mayo del 2014]

[34] OWASP, "Top 10 2013-A8-Cross-Site Request Forgery (CSRF)" [en línea]. http://www.owasp.org/index.php/Top_10_2013-A8-Cross-Site_Request_Forgery_%28CSRF%29 [consulta: 16 de Mayo del 2014]

[35] Barth, A. et al, "Robust Defenses for Cross-Site Request Forgery", CCS'08, 2008

[36] OWASP, "Top 10 2013-A9-Using Components with Known Vulnerabilities" [en línea]. http://www.owasp.org/index.php/Top_10_2013-A9-Using_Components_with_Known_Vulnerabilities [consulta: 16 de Mayo del 2014]

[37] OWASP, "Top 10 2013-A10-Unvalidated Redirects and Forwards" [en línea]. http://www.owasp.org/index.php/Top_10_2013-A10-Unvalidated_Redirects_and_Forwards [consulta: 16 de Mayo del 2014]

[38] OWASP, "Top 10 2007-Malicious File Execution" [en línea]. http://www.owasp.org/index.php/Top_10_2007-A3 [consulta: 16 de Mayo del 2014]

[39] OWASP, "Top 10 2007-Information Leakage and Improper Error Handling" [en línea]. http://www.owasp.org/index.php/Top_10_2007-A6 [consulta: 16 de Mayo del 2014]

[40] Kajo-Mece, E. et al, "Protection of Web Applications Using Aspect Oriented Programming and Performance Evaluation", BCI'12, 2012, Novi Sad, Serbia

[41] C. Jackson, et al., "The HTTP Origin Header" [en línea]. http://tools.ietf.org/id/draft-abarth-origin-03.html [consulta: 16 de Mayo del 2014]

[42] HOWARD, John D. Thesis: An Analysis of security on the Internet 1989-1995. Carnegie Institute of Technology. Carnegie Mellon University. 1995. EE.UU. http://www.cert.org. Capítulo 12–Página 165

[43] National Vulnerability Database, "NVD Results" [en línea]. https://web.nvd.nist.gov/view/vuln/search-results?adv_search=true&cves=on&query=webserver&cwe_id=&cpe_vendor=&cpe_product=&cpe_version=&pub_date_start_month=-1&pub_date_start_year=-1&pub_date_end_month=-1&pub_date_end_year=-1&mod_date_start_month=-1&mod_date_start_year=-1&mod_date_end_month=-1&mod_date_end_year=-1&cvss_sev_base=&cvss_av=&cvss_ac=&cvss_au=&cvss_c=&cvss_i=&cvss_a=&uscert_ta=&uscert_vn=&oval_query=&cve_id= [consulta: 16 de Mayo del 2014]

[44] RedHat, "How to disable weak SSL ciphers in Apache - mod_ssl" [en línea]. http://access.redhat.com/site/solutions/359523 [consulta: 16 de Mayo del 2014]

[45] Héctor Jara y Federico Pacheco, Ethickal Hacking 2.0, Manuales Users, 2012, Página: 24

[46] Sachdeva M. et al, "DDoS Incidents and their Impact: A Review", The International Arab Journal of Information Technology, Vol. 7, No. 1, January 2010

[47] zone-h, "Unrestricted information" [en línea]. http://zone-h.org [consulta: 16 de Mayo del 2014]

[48] PaulCoyote, "www.DatoSeguro.gob.ec NO es seguro" [en línea]. [consulta: 16 de Mayo del 2014] http://www.ecualug.org/?q=20121126/blog/paulcoyote/wwwdatosegurogobec_no_es_seguro

[49] Pérez E., "Liberen a PaulCoyote" [en línea]. http://ernestoperez.com/?p=1425

[50] Pérez E., "Mi opinión sobre el DDOS de aanonymous en Ecuador" [en línea]. http://ernestoperez.com/?p=781 [consulta: 16 de Mayo del 2014]

[51] Jim, T., Swamy, N., & Hicks, M. (2007, May). Defeating script injection attacks with browser-enforced embedded policies. In Proceedings of the 16th international conference on World Wide Web (pp. 601-610). ACM.

[52] Reis, Charles, Adam Barth, and Carlos Pizano. "Browser security: lessons from google chrome." Queue 7.5 (2009): 3.

[53] Dormann W. et al., "Securing Your Web Browser" [en línea]. http://www.cert.org/historical/tech_tips/securing-web-browser-index.cfm, CERT [consulta: 16 de Mayo del 2014]

[54] Wheeler, D., "Secure programming for Linux and Unix HOWTO" [en línea]. http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO.pdf [consulta: 16 de Mayo del 2014]

[55] Seacord R. "CERT® C Coding Standard, Second Edition, The: 98 Rules for Developing Safe, Reliable, and Secure Systems", Addison-Wesley, 2014, 2nd Edition [consulta: 16 de Mayo del 2014]

[56] RedHat, "Red Hat Enterprise Linux Life Cycle" [en línea]. https://access.redhat.com/site/support/policy/updates/errata/ [consulta: 16 de Mayo del 2014]

[57] Chimmanee, S., Veeraprasit, T., & Srisa-An, C. (2014). A Performance Evaluation of Vulnerability Detection: NetClarity Audito, Nessus, and Retina. International Journal of Computer Science & Network Security, 14(3).

[58] Randals K. Nichols et al., "Wireless Security: Models, Threats and Solutions", McGraw-Hill Telecom Professional Books, 2002 [ISBN: 0-07-138038-8], Capítulo 5, Página 184

[59] Ristic, I, "Apache Security", O'Reilly, March 2005, 0-596-00724-8, Sección 5.4.3. Programming Model Attacks

[60] Wadge C., "Performance testing while under attack" [en línea]. http://www.hiawatha-webserver.org/weblog/64 [consulta: 16 de Mayo del 2014]

[61] Q-success, "Usage of web servers for websites" [en línea]. http://w3techs.com/technologies/overview/web_server/all [consulta: 27 de Abril del 2014]

[62] Q-success, "Historical trends in the usage of web servers for websites" [en línea]. http://w3techs.com/technologies/history_overview/web_server [consulta: 27 de Abril del 2014]

[63] Q-success, "Usage of web servers broken down by ranking" [en línea]. http://w3techs.com/technologies/cross/web_server/ranking [consulta: 27 de Abril del 2014]

[64] Q-success, "Usage of server-side programming languages for websites" [en línea]. http://w3techs.com/technologies/overview/programming_language/all [consulta: 27 de Abril del 2014]

[65] Q-success, "Historical trends in the usage of server-side programming languages for websites" [en línea]. http://w3techs.com/technologies/history_overview/programming_language [consulta: 27 de Abril del 2014]

[66] Netcraft, "April 2014 Web Server Survey" [en línea]. http://news.netcraft.com/archives/2014/04/02/april-2014-web-server-survey.html [consulta: 16 de Mayo del 2014]

[67] W3Techs, "Most popular web servers by country" [en línea]. http://w3techs.com/blog/entry/most_popular_web_servers_by_country [consulta: 16 de Mayo del 2014]

[68] Schürmann, T, "Rápido y seguro con el servidor web Hiawatha" [en línea]. http://www.linux-magazine.es/issue/55/018-025HiawathaLM55.pdf [consulta: 16 de Mayo del 2014]

[69] Leisink, H., "About Hiawatha webserver" [en línea]. https://www.hiawatha-webserver.org/about [consulta: 16 de Mayo del 2014]

[70] Black Duck Software Inc., "Ohloh, the open source network" [en línea]. http://www.ohloh.net/ [consulta: 16 de Mayo del 2014]

[71] Secunia, "Advisories" [en línea]. http://secunia.com/community/advisories/ [consulta: 16 de Mayo del 2014]

[72] Shekyan, S., "Application Layer DoS attack simulator",[en línea]. https://code.google.com/p/slowhttptest/ [consulta: 16 de Mayo del 2014]

[73] Leisink, H., "Re: CSRF checking Origin header" [correo electrónico], 07/Abril/2014, Referencia <5340C471.5010606@ecualinux.com> [consulta: 16 de Mayo del 2014]

Glosario de Términos

ADN	Ácido Desoxirribonucleico
AIDE	Advanced Intrusion Detection Environment
Anonymous	En Español: Anónimo. Grupo de personas que realizan Hacktivismo
Apache	Servidor web de la fundación Apache http://httpd.apache.org
ASP	Active Server Pages
BeeWare	Empresa desarrolladora de producto WAF http://www.bee-ware.net/en/products/web-application-firewall
BD	Base de Datos
BSD	Berkeley Software Distribution
CD	Compact Disk
CERT	Computer Emergency Response Team
CIA	Principios básicos de la seguridad: Confidencialidad, Integridad y Disponibilidad, del Inglés Confidentiality, Integrity, Availability
CGI	Common Gateway Interface
CheckPoint	Sistema integrado de manejo de firewalls de Check Point Software Technologies http://checkpoint.com
CSIRT	Computer System Incident Response Team
CSRF	CrossSite Request Forgery
CSV	Comma Separated Values
DOM	Document Object Model
DoS	Negación de Servicio, del Inglés Denial of Service
DVWA	Damn Vulnerable Web Application. Aplicación de pruebas de vulnerabilidades de RandomStorm http://www.dvwa.co.uk/
EPEL	Extra Packages for Enterprise Linux
FTP	File Transfer Protocol uno de los primeros protocolos de transferencia de archivos entre cliente y servidor. No usa encriptación sino que toda la conexión ocurre en texto claro. A pesar de su evidente inseguridad, se sigue siendo muy utilizado.
FortiGate	Sistema integrado de manejo de firewall de PaloAlto Networks http://fortinet.com
Hacktivismo	Utilización de herramientas de Internet para realizar activismo político
hiawatha-webserver	Servidor web de Hugo Leisink
HTML	HyperText Markup Language
HTTP Referer	Permite determinar desde qué lugar se refirió el acceso que se está realizando. Por tanto permite saber de dónde venía esta petición.

ID	Identificador
IDS	Intrusion Detection System
IIS	Internet Information Server
IMAP 4	Internet Message Access Protocol versión 4
IP	Protocolo de Internet, del Inglés Internet Protocol
IPv4	Protocolo de Internet, versión 4. Primera versión pública de IP
IPv6	Protocolo de Internet, versión 6. Versión de IP sucesora de IPv4
IPS	Intrusion Prevention System
ISO	International Organization for Standardization
ITU	Organización Internacional de Telecomunicaciones,d el Inglés: International Telecommunications Union
JSP	JavaServer Pages
LDAP	Protocolo Ligero de Acceso a Directorios, del Inglés Lightweight Directory Access Protocol
LibreSSL	Paquete de SSL desarrollado por http://libressl.org
Lighttpd	Servidor web escrito por Jan Kneschke http://lighttpd.net
MAC	Media Access Control
mod_security	WAF desarrollado por Trustwave http://www.modsecurity.org/
naxsi	WAF en Software Libre https://github.com/nbs-system/naxsi
Nginx	Servidor web escrito por Igor Sysoev http://nginx.org
NoSQL	No sólo SQL, otros sistemas de base de datos que no siguen el modelo SQL
OpenSSL	Herramienta en Software Libre que provee implementaciones de los protocolos SSL y TLS
OpenVPN	Servidor de VPN desarrollado por OpenVPN Technologies Inc. http://openvpn.net
OS	Operating System
OWASP	Open Web Application Security Project
PDF	Portable Document Format
PHP	PHP Hypertext Pre-processor, lenguaje de programación
PHP-FPM	PHP Hypertext Pre-processor FastCGI Process Manager
POP3	Post Office Protocol version 3
QualsysGuard	WAF desarrollado por Qualsys https://qualys.com/enterprises/qualysguard/web-application-firewall/
RAM	Random Access Memory
RHEL	Distribución de Linux de la empresa RedHat Inc.

rWeb	WAF de DenyAll: http://www.denyall.com/decision-maker/products/rweb_en
SCP	Secure Copy
SFTP	Secure FTP
SHA	Secure Hash Algorithm
SLD	Second Level Domain
Slowloris	Ataque DoS que abre conexiones parciales a servidores web para agotar sus recursos
SMTP	Simple Mail Transfer Protocol
SMS	Short Message Service
Snort	IDS de Snort desarrollado por Sourcefire http://snort.org
SQL	Lenguaje de Consulta Estructurado del Inglés Structured Query Language
SSH	Secure Shell protocolo encriptado de comunicaciones entre el cliente y el servidor. Utilizado no solamente para manejar remotamente un servidor sino que además se puede usar para transferir archivos entre el cliente y el servidor a través de Secure Copy (SCP) o Secure FTP (SFTP)
SSL	Capa de Socket Segura, del Inglés Secure Socket Layer
Suricata	IDS en Software Libre mantenido por http://suricata-ids.org
TCP	Transmission Control Protocol
TLS	Seguridad en capa de transporte, del Inglés Transport Layer Security
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
VPN	Red Privada Virtual, del Inglés Virtual Private Network
WAF	Cortafuegos de aplicaciones web, del Inglés Web Application Firewall
WWW	WorldWide Web
XSLT	Extensible Stylesheet Language Transformations
XSS	Cross-Site Scripting