ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Maestría en Seguridad Informática Aplicada

"HARDENING EN SERVIDOR WEB LINUX APACHE, PHP Y CONFIGURAR EL FIREWALL DE APLICACIONES MODSECURITY PARA MITIGAR ATAQUES AL SERVIDOR"

EXAMEN DE GRADO (COMPLEXIVO)

Previo a la obtención del grado de:

MAGISTER EN SEGURIDAD INFORMÁTICA APLICADA

BYRON ALBERTO DELGADO QUISHPE

GUAYAQUIL - ECUADOR

AÑO 2015

AGRADECIMIENTO

Doy gracias a Dios, por haberme dado la sabiduría y darme las fuerzas para culminar con éxito mis estudios. A mi esposa Lorena Castillo por haberme apoyado moralmente en esta etapa de mi carrera profesional, alentándome en cada momento a seguir adelante, a mis padres por su ayuda incondicional de estar siempre pendientes de mí, a cada uno de los profesores que impartieron sus conocimientos que fueron de gran ayuda.

DEDICATORIA

El presente trabajo dedico en especial a mi madre por el esfuerzo brindado cada día de darme todo el apoyo y consejos de seguir siempre adelante con mis estudios para poder ser quien soy un profesional.

TRIBUNAL DE SUSTENTACIÓN

por: MSig. Cruz María Falcones

MGS. Gonzalo Luzardo

PROFESOR DELEGADO

POR LA MAESTRÍA EN SEGURIDAD

INFORMÁTICA APLICADA

MGS. Roky Barbosa

PROFESOR DELEGADO

POR LA FACULTAD EN INGENIERÍA

EN ELECTRICIDAD Y COMPUTACIÓN

RESUMEN

En el presente documento el lector podrá conocer el detalle realizado en la recolección de información al objetivo, realizando un análisis de vulnerabilidades encontrado en el servidor.

Se realizara un hardening al servidor web, se procederá a revisar las directivas en los archivos de configuración del servicio apache, php, y se procederá a realizar instalación y configuración de un firewall de aplicaciones llamado mod_security la cual nos permitirá mitigar ataques a nuestro servidor web.

Una vez realizada los cambios en los archivos de configuración se puede determinar los resultados obtenidos.

ÍNDICE GENERAL

AGRADECIMIENTO ii

DEDICATORIA iii

TRIBUNAL DE SUSTENTACIÓN iv

RESUMEN v

ÍNDICE GENERAL vi

ABREVIATURAS Y SIMBOLOGÍAS viii

ÍNDICE DE FIGURAS ix

ÍNDICE DE TABLAS xi

INTRODUCCIÓN xiii

CAPÍTULO 1 1

GENERALIDADES 1

1.1 TÍTULO 1

1.2 DESCRIPCIÓN DEL PROBLEMA 1

1.3 SOLUCIÓN PROPUESTA 4

CAPÍTULO 2 6

METODOLOGÍA Y DESARROLLO DE LA SOLUCIÓN 6

2.1 GENERALIDADES Y ARQUITECTURA DE UN SITIO WEB 6

2.2 PLATAFORMA ACTUAL Y ANALISIS DE VULNERABILIDADES. 11

2.3 HARDENING INICAL DEL SERVIDOR WEB. 24

2.4 CONFIGURACIÓN DE ARCHIVO HTTPD.CONF 26

2.5 CONFIGURACIÓN DE ARCHIVO PHP.INI 32

2.6 INSTALACIÓN Y CONFIGURACIÓN DE PORTSENTRY. 34

2.7 INSTALACIÓN Y CONFIGURACIÓN FIREWALL DE APLICACIÓN MODSECURITY. 34

CAPÍTULO 3 38

RESULTADOS DE LA REVISIÓN. 38

3.1 PRUEBAS INTERNAS 38

3.2 INFORME DE PRUEBAS 42

CONCLUSIONES Y RECOMENDACIONES 46

BIBLIOGRAFÍA 48

ABREVIATURAS Y SIMBOLOGÍAS

DoS	Denegación de servicio.
HTTP	Hypertext Transfer Protocol
OWASP	The Open Web Application Security Project
SQL Injection:	Es una técnica de inyección de código que explota una vulnerabilidad.
SSH	Secure Shell -intérprete de órdenes seguro
WAF	Web Application Firewall
XSS	Cross Site Scripting

ÍNDICE DE FIGURAS

Figura 2.1 Comando de Nmap muestra versión de S.O 12

Figura 2.2 Comando de Nmap usando directiva -A 13

Figura 2.3 Comando de Nikto muestra HTTP TRACE 14

Figura 2.4 Análisis de Vulnerabilidad con Nessus 15

Figura 2.5 PHP.info 17

Figura 2.6 Configuración de Fail2ban 25

Figura 2.7 Comando http -V 27

Figura 2.8 Desactivar el acceso a los listados de directorios 28

Figura 2.9 HttOnly y Secure flag 30

Figura 2.10 Archivo de configuración de mod_security.conf 35

Figura 2.11 Reglas de modsecurity-crs 35

Figura 2.12 Archivo Logs de Http 36

Figura 3.1 Prueba de OS Command Injection 39

Figura 3.2 Prueba de XSS – Reflected 39

Figura 3.3 Prueba Code Injection 40

Figura 3.4 Listado de Ficheros 40

Figura 3.5 Información de cabeceras 41

Figura 3.6 Barra de navegación con HTTP 41

Figura 3.7 Bloqueo de accesos por fuerza bruta 42

Figura 3.8 Ingreso de ssh al puerto 2020 usuario administrator 43

Figura 3.9 Verificación de protocolo https 43

Figura 3.10 Navegación con HTTPS 44

Figura 3.11 Tag: Web Attack/File Injection 45

Figura 3.12 Tag: Web Attack/Ssi_Injection.....45

Figura 3.13 Tag: Web Attack/Sql_Injection 45

ÍNDICE DE TABLAS

Tabla 1 Características del Servidor Web 11

Tabla 2 Detalle de puertos abiertos.....12

INTRODUCCIÓN

El siguiente trabajo describe la revisión de la tecnología que utilizan y demostrar los ataques que se pueden efectuar al no contar con los mecanismos de defensa como es el hardening del servidor de aplicaciones web para que sean seguros, en nuestro caso se usa el servidor apache y php como lenguaje de programación.

En el capítulo 1 se da a conocer los antecedentes que motivaron la realización de estas pruebas, así como también se contextualiza el problema y la solución propuesta.

En el capítulo 2 se revisara la plataforma actual, y se realizara un análisis de vulnerabilidades, también se realizara un hardening inicial y se procederá a configurar los archivos de configuración de los servicios apache y php, etc.

En el capítulo 3 se realiza el análisis de resultados y se documenta la ejecución de varias pruebas de concepto de las vulnerabilidades identificadas.

CAPÍTULO 1

GENERALIDADES

1.1 TÍTULO

Hardening en Servidor Web Linux Apache, Php y configurar el firewall de aplicación ModSecurity para mitigar ataques al Servidor.

1.2 DESCRIPCIÓN DEL PROBLEMA

La tecnología ha sido de mucha ayuda a la humanidad, pero así mismo ha permitido la aparición de nuevas amenazas concernientes con la seguridad de la información, en aplicativos web.

Apache es el servidor HTTP con mayor presencia en Internet. La configuración de Apache se realiza editando un archivo donde tiene varias directivas, el cual tiene instrucciones que debe seguir Apache para su funcionamiento.

La empresa actualizo los equipos de cómputo, el cual cumple la función de alojar los aplicativos web de la empresa. En este servidor se encuentra instalado el sistema operativo Linux (Centos 7), y Apache como servidor web, y lenguaje de programación PHP.

El problema presentado es que dicho servidor se encuentra con las configuraciones básicas, por la cual está expuesto a diferentes vulnerabilidades, que se describen a continuación:

- **Inyección de SQL.** Esta vulnerabilidad permite que un atacante envíe una entrada con un contenido que provoque una interferencia en la interacción de la aplicación con las bases de datos del back-end, y así ser capaz de recuperar datos arbitrarios por medio de la aplicación, interferir en la lógica de la misma, o ejecutar comandos sobre el servidor de base de datos.
- **Quiebre de la autenticación.** Esta categoría de vulnerabilidad comprende diferentes defectos dentro del mecanismo de inicio de sesión de la aplicación, lo cual puede permitir que un atacante adivine contraseñas débiles, lance un ataque de fuerza bruta, o evite por completo el proceso de inicio de sesión.
- **Quiebre de los controles de acceso.** Esto comprende los casos en que la aplicación fracasa en proteger adecuadamente el acceso a sus datos y

sus funcionalidades, potencialmente permitiendo que un atacante visualice datos sensibles de otro usuario hospedados en el servidor, o lleve a cabo acciones privilegiadas.

- **Fuga de información.** Comprende los casos en que una aplicación divulga información sensible, la que es utilizada por un atacante para desencadenar un ataque contra la aplicación, mediante el manejo de un error derivado de un defecto.

Las vulnerabilidades mencionadas fácilmente podrían comprometer los elementos más importantes de la seguridad informática que son: confidencialidad, integridad, disponibilidad.

En base a los problemas mencionados, el crecimiento de la tecnología es cada vez más en aumento y las plataformas desarrolladas con tecnología web en las empresas hay la necesidad de ser competitivas en dar soluciones online utilizando el internet para realizar todo tipo de transacciones, por estos motivos es importante implementar una infraestructura lo suficientemente segura, para evitar ataques informáticos.

1.3 SOLUCIÓN PROPUESTA

La presente propuesta tiene como objetivo describir el estado actual del servidor de aplicaciones sin realizar el hardening y el estado final posterior a la aplicación de estas técnicas de seguridad.

Con la herramienta Nmap se procederá a verificar los servicios y puertos relacionados que se encuentren abiertos en nuestro servidor, para en lo posterior verificar si es correcto o no que se encuentre en este estado.
Para realizar el hardening del servidor se procede a implementar las siguientes técnicas de seguridad:

- Actualización de parámetros de configuración del servidor apache, esta actualización permitirá evitar pérdidas de información, en los encabezados de los mensajes de respuestas http, que pueda ser utilizada por un atacante.

- Realizar bloqueos de acceso a los usuarios que no tengan permisos necesarios para ver documentos que son ofrecidos por el servidor.

- También se implementara un acceso seguro (HTTPS) hacia nuestro servidor implementando SSL.

- Se realizara actualizaciones en los parámetros de configuración de php.

- Se implementara Portsentry que es un sistema de detección de intrusos para detectar ataques.

Adicional, es conveniente instalar y configurar el firewall de aplicación ModSecurity, que es una herramienta que nos ayuda a detectar y prevenir ataques contra el servidor.

Los beneficios a obtener serán:

- Mejora la seguridad del servidor frente a amenazas internas y externas.
- Reducir los riesgos asociados con fraude y el error humano.
- Maximización de la disponibilidad de los servidores.
- Permite al cliente contar con las mejores prácticas de seguridad que pueden ser establecidas para sus diferentes plataformas operativas.

CAPÍTULO 2

METODOLOGÍA Y DESARROLLO DE LA SOLUCIÓN

2.1 GENERALIDADES Y ARQUITECTURA DE UN SITIO WEB.

El protocolo HTTP- (Hypertext Transfer Protocol): Es un protocolo perteneciente a la familia TCP que trabaja en la capa 7 (Aplicación) del modelo de referencia OSI, y es utilizado por todos los navegadores Web, y a nuestros días por un número considerable de aplicaciones empresariales, y de uso común. Por lo tanto todos los sitios web comunican a sus usuarios finales (Cliente/Servidor) mediante el uso del protocolo HTTP [1] (wikipedia, 2015).

El protocolo HTTP

Es un protocolo donde el mensaje es intercambiado entre el cliente y el servidor.

Componentes importantes a tener presente dentro del protocolo HTTP, tales como:

- Headers
- Message Body
- HTTP Request
- HTTP response

El protocolo HTTP – **"HTTP Request":** Es importante tener presente la forma en que trabaja en protocolo HTTP, cuando solicita la conexión a un servidor Web, usando un cliente como Firefox por ejemplo. A modo de pruebas cuando conectamos con el sitio web www.portal.com, sucede lo siguiente:

- Se hace en intercambio en tres vías de forma cómo trabaja TCP
- Luego del intercambio de 3 vías, se hace una solicitud (Request Method) para obtener el contenido de la página, la cual puede ser del tipo GET por ejemplo. Hay otros métodos de solicitud, tales como: POST, PUT, DELETE, OPTIONS, TRACE.

  En este tipo de solicitudes, se identifican variables y/datos importantes tales como la versión del protocolo HTTP, la variable Host, Accept, User Agent, Keep-Alive entre otros.

El protocolo HTTP – **"HTTP Response":** Una vez que el servidor web ha recibido la solicitud del tipo (Request), el servidor web, responde a dicha solicitud realizada por el cliente (Browser-Navegador web) con un (Response). Dentro de las variables que podemos encontrar dentro este tipo de respuesta,

se encuentran: Date, Cache-Control, Content-Type, Content-Encoding, Server, entre otros.

Teniendo presente que el protocolo HTTP es un protocolo sin estado, es importante entender conceptos como las cookies y las sesiones.

Métodos HTTP

Cuando se está atacando aplicaciones web, se usan casi exclusivamente los métodos más comúnmente utilizados: GET y POST. Pero hay que ser consciente de algunas diferencias importantes entre estos métodos, ya que pueden afectar la seguridad de la aplicación si se pasan por alto. [1] (wikipedia, 2015).

El método GET está diseñado para recuperar recursos. Se puede utilizar para enviar parámetros para el recurso solicitado en la cadena de consulta URL. Esto permite a los usuarios marcar una dirección URL de un recurso dinámico que puedan volver a utilizar u otros usuarios pueden recuperar el recurso equivalente en una ocasión posterior (como en una consulta de búsqueda guardada). Las URLs se muestran en la pantalla y quedan almacenadas en varios lugares, tales como el historial del navegador y los logs de acceso del servidor web.

El método POST se ha diseñado para realizar acciones. Con este método, los parámetros de la solicitud pueden enviarse tanto en la cadena de consulta URL como en el cuerpo del mensaje. Aunque la URL aún se puede marcar, los parámetros enviados en el cuerpo del mensaje serán excluidos del marcador. Estos parámetros serán también excluidos de los distintos lugares en que los registros de las direcciones URL son almacenados incluyendo la cabecera Referer.

Debido a que el método POST se ha diseñado para la realización de acciones, si un usuario hace clic en el botón Atrás del navegador para volver a la página que se accede mediante este método, el navegador vuelve a emitir la petición de forma automática en su lugar, se advierte al usuario de lo que está por hacer, se le pregunta si quiere volver a enviar la petición.

Esto evita que los usuarios realicen sin querer una acción más de una vez. Por esta razón, las peticiones POST siempre se deben utilizar cuando una acción se está realizando.

Además de los métodos GET y POST, el protocolo HTTP soporta otros métodos que se han creado con fines específicos. Aquí están los otros que es conveniente conocer:

HEAD funciona de la misma manera como una petición GET, salvo que el servidor no debe devolver un cuerpo del mensaje en su respuesta. El servidor debe devolver los mismos encabezados que hubiera regresado al hacer la misma petición GET correspondiente.
Por lo tanto, este método se puede utilizar para comprobar si un recurso existe antes de realizar una solicitud GET para ello.

TRACE está diseñado para fines de diagnóstico. El servidor debe devolver en el cuerpo de la respuesta el contenido exacto del mensaje de petición que recibió. Este se puede utilizar para detectar el efecto de los servidores proxy entre el cliente y el servidor que pueden manipular la petición.

OPTIONS pide al servidor informar los métodos HTTP que están disponibles para un recurso en particular. El servidor normalmente devuelve una respuesta que contiene una cabecera “Allow” que lista los métodos disponibles.

PUT intenta subir el recurso especificado al servidor, utilizando el contenido del cuerpo de la solicitud. Si este método está habilitado, puede ser usado para atacar a la aplicación, como por ejemplo subir una secuencia de comandos arbitraria y ejecutarla en el servidor.

Existen muchos otros métodos HTTP que no son directamente relevantes para atacar aplicaciones web. Sin embargo, un servidor web podría estar expuesto a un ataque si ciertos métodos peligrosos están disponibles.

2.2 PLATAFORMA ACTUAL Y ANALISIS DE VULNERABILIDADES.

La plataforma donde se realizara el hardening de servidor web cuenta con las siguientes características:

Tabla 1 Características del Servidor Web

HOSTNAME	SRVWEB
MARCA	IBM
MODELO	X3550 M3
SERIE	KQ36N75
S.O.	Centos 7
CARACTERISTICAS:	Intel Xeon E5620 /2, 2.4 Ghz, 12M, 23 GB,446 GB HD
IP:	192.168.0.8

Se realiza reconocimiento activo del Servidor Web con la herramienta Nmap. Como se muestra en a continuación podemos ver información respecto a los puertos abiertos y versiones de los servicios levantados.

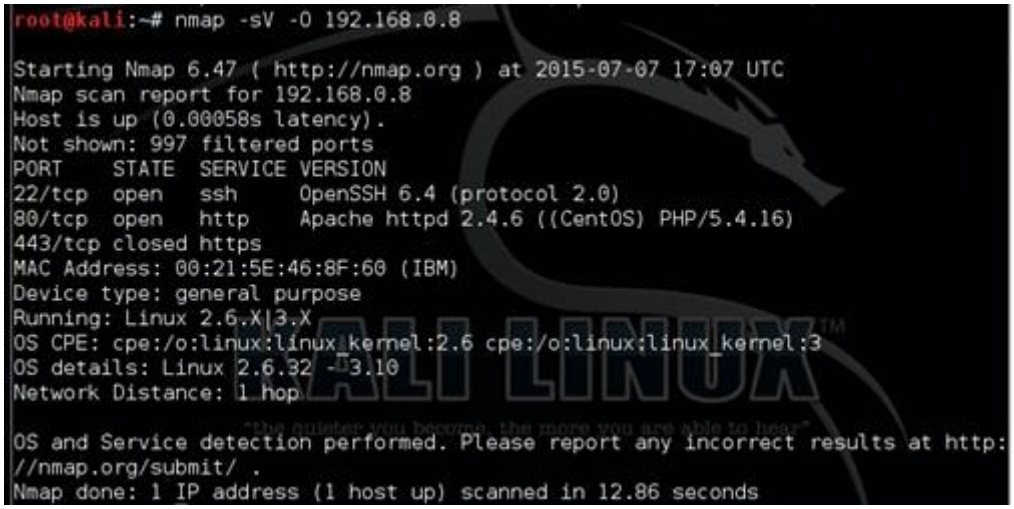

Figura 2.1 Comando de Nmap muestra versión de S.O

Detalle de puertos abiertos:

Tabla 2 Detalle de puertos abiertos

Dirección IP:	192.168.0.8			
Sistema Operativo:	Linux Centos			
Puerto	**Protocolo**	**Status**	**Servicio**	**Versión**
20	TCP	ABIERTO	SSH	OPENSSH 6.4
80	TCP	ABIERTO	HTTP	APACHE HTTPD.2.4.6
443	TCP	CERRADO	HTTPS	(CENTOS / PHP 5.4.16)

Comando: nmap –A 192.168.0.8

Figura 2.2 Comando de Nmap usando directiva -A

Aquí nos muestra que el método HTTP TRACE está activo.

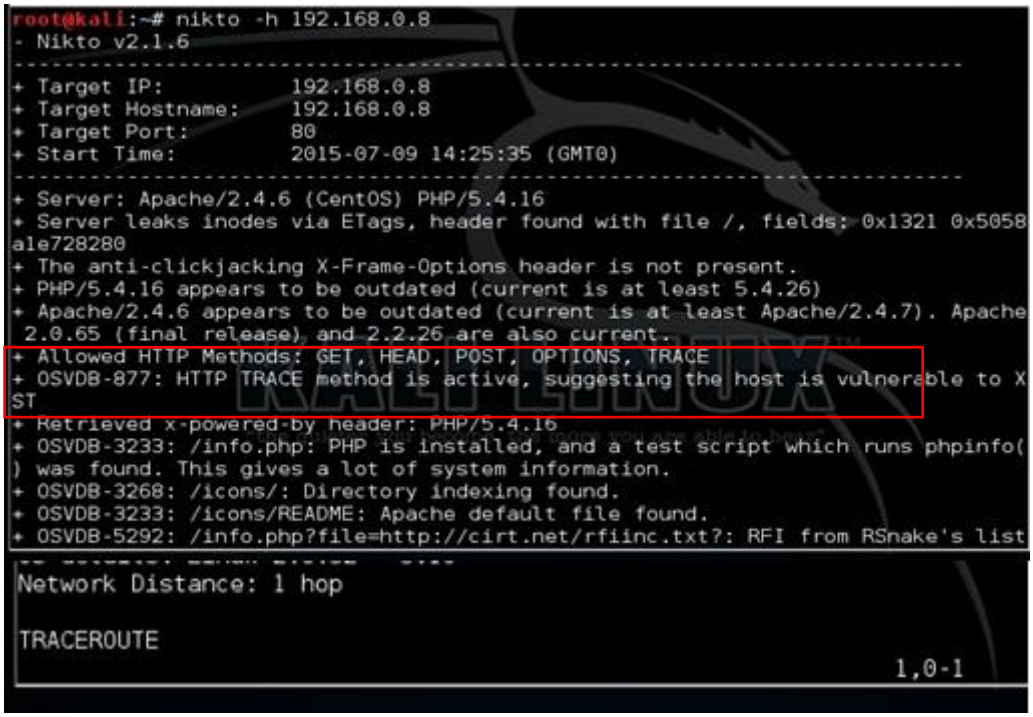

Figura 2.3 Comando de Nikto muestra HTTP TRACE

- **ANALISIS DE VULNERABILIDADES.**

Con ayuda de la herramienta Nessus realizamos un análisis de vulnerabilidad al servidor web, aplicando políticas de **Web Application Tests.**

192.168.0.8

Summary

Critical	High	Medium	Low	Info	Total
0	0	2	0	7	9

Details

Severity	Plugin Id	Name
Medium (5.0)	11229	Web Server info.php / phpinfo.php Detection
Medium (4.3)	11213	HTTP TRACE / TRACK Methods Allowed
Info	10107	HTTP Server Type and Version
Info	11219	Nessus SYN scanner
Info	18261	Apache Banner Linux Distribution Disclosure
Info	24260	HyperText Transfer Protocol (HTTP) Information
Info	43111	HTTP Methods Allowed (per directory)
Info	48243	PHP Version
Info	84574	Backported Security Patch Detection (PHP)

Figura 2.4 Análisis de Vulnerabilidad con Nessus

Detalle del Análisis de Vulnerabilidades al Servidor Web.

Vulnerabilidad Medio (6.0): Info.php Servidor Web / Detección phpinfo.php

Referencias:

Nessus descubrió la siguiente URL que llama phpinfo ():

- Http://192.168.0.8/info.php

Numero de Vulnerabilidades Identificadas: 1

Servicios- Ítems Afectados:

Servidor WWW

Puerto TCP 80

Descripción de la Vulnerabilidad:

Muchos tutoriales de instalación PHP instruyen al usuario crear un archivo PHP que llama a la función de PHP phpinfo () para propósitos de depuración.

Detalles de la Vulnerabilidad:

Varias aplicaciones PHP también pueden incluir un archivo de este tipo. Al acceder a un archivo de este tipo, un atacante remoto puede descubrir una gran cantidad de información sobre el servidor web remoto, incluyendo: - El nombre de usuario del usuario que instala PHP y si es un usuario SUDO. - La dirección IP de la máquina. - La versión del sistema operativo. - La versión del servidor web. - El directorio raíz del servidor web. - La información de configuración de la instalación de PHP remoto.

Riesgo:

Al acceder a un archivo de este tipo, un atacante remoto puede descubrir una gran cantidad de información sobre el servidor web remoto.

Evidencias:

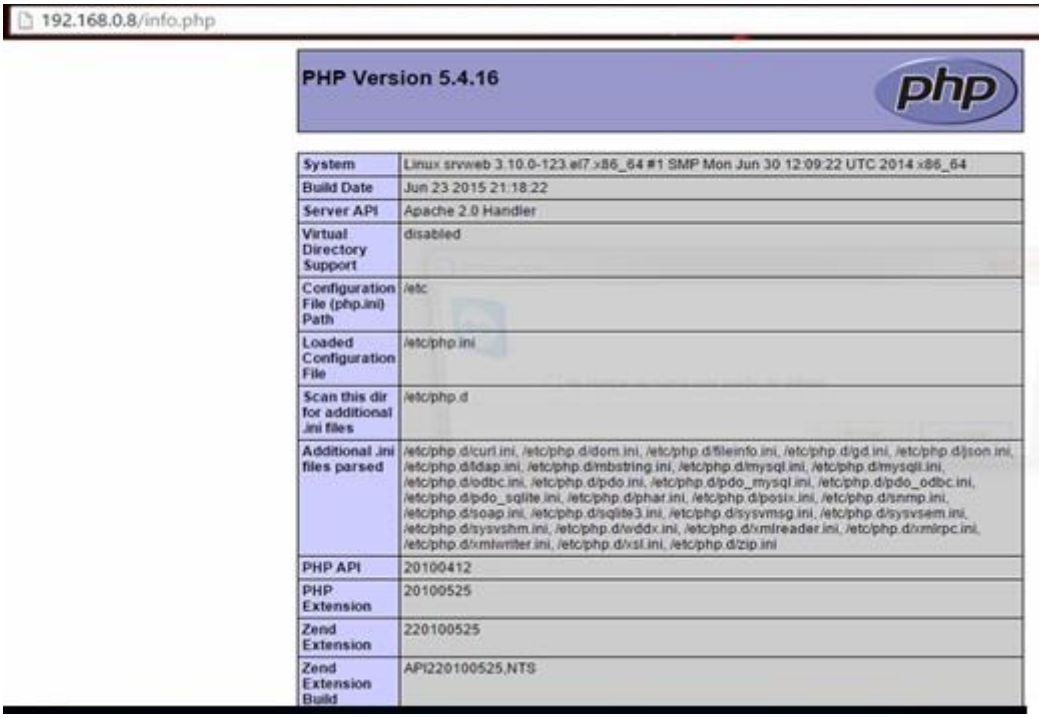

Figura 2.5 PHP.info

Recomendación para solucionar esta Vulnerabilidad:

Elimine el archivo (s) afectado.

Vulnerabilidad Medio (4.3): HTTP TRACE /TRACK Methods Allowed

Referencias:

Información de fugas / Divulgación	CWE-200	Una exposición de información es la divulgación intencional o no intencional de información a un actor que no está expresamente autorizado a tener acceso a esa información.

Numero de Vulnerabilidades Identificadas: 1

Servicios- Ítems Afectados:

Servidor WWW

Puerto TCP 80

Descripción de la Vulnerabilidad:

El servidor web remoto es compatible con los métodos TRACE y / o TRACK. TRACE y TRACK son métodos HTTP que se utilizan para las conexiones del servidor de depuración web.

Detalles de la Vulnerabilidad:

El método HTTP TRACE le pregunta a un servidor web para repetir el contenido de la petición de vuelta al cliente para propósitos de depuración. El método HTTP TRACE se describe en la norma HTTP 1.1 (RFC 2616 , sección 9.8).

TRACE El método TRACE se utiliza para invocar un mando a distancia, la capa de aplicación loop-back vuelta del mensaje de solicitud. El destinatario final de la solicitud debe reflejar el mensaje recibido de vuelta al cliente como entidad cuerpo de un (OK) Respuesta 200.

Riesgo:

Los atacantes pueden abusar funcionalidad HTTP TRACK para obtener acceso a la información en las cabeceras HTTP como las cookies y los datos de autenticación.

Impacto:

Los atacantes pueden abusar funcionalidad HTTP TRACK para obtener acceso a la información en las cabeceras HTTP como las cookies y los datos de autenticación. En presencia de otras vulnerabilidades entre dominios en los navegadores web, la información de cabecera sensible podría ser leída desde cualquier dominio que apoyan el método HTTP TRACK.

Evidencias:

Nessus envía la siguiente petición TRACE:

```
TRACE /Nessus19502751.html HTTP/1.1
Connection: Close
Host: 192.168.0.8
HTTP/1.1 200 OK
Date: Thu, 16 Jul 2015 22:42:02 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
Keep-Alive: timeout=5, max=100
Host: 192.168.0.8
```

Recomendación para solucionar esta Vulnerabilidad:

Deshabilitar estos métodos. Consulte la salida de plug-in para más información.

Vulnerabilidad Info:

Nessus SYN Scanner

Apache Banner Linux Distribution Disclosure

Backported Security Patch Detection (PHP)

HTTP Methods Allowed

HTTP Server Type and version

Hypertext Transfer Protocol (HTTP) Information

PHP version

Web Server Directory Enumeration

Numero de Vulnerabilidades Identificadas: 8

Servicios- Ítems Afectados:

Servidor WWW

Puerto TCP 22, 80

Descripción de la Vulnerabilidad:

- **Nessus SYN Scanner**

  Este plugin es 'entreabierta' escáner de puertos un SYN. Será bastante rápido incluso en contra de un blanco con cortafuegos. Tenga en cuenta que las exploraciones SYN son menos intrusivo que TCP (full connect) exploraciones contra los servicios rotos, pero podría causar problemas para los servidores de seguridad menos robustos y también dejar sin cerrar las conexiones en el destino remoto, si el red es cargado.

- **Apache Banner Linux Distribution Disclosure**

  Este plugin extrae el banner del servidor web Apache y los intentos de determinar qué distribución de Linux el host remoto están en ejecución.

- **Backported Security Patch Detection (PHP)**

  Las revisiones de seguridad pueden haber sido "backported" al PHP remoto instalar sin cambiar su número de versión. Controles basados en Banner han sido deshabilitado para evitar falsos positivos. Tenga en cuenta que esta prueba es sólo informativo y no denota ningún problema de seguridad.

- **HTTP Methods Allowed**

  Al llamar al método OPTIONS, es posible determinar qué métodos HTTP se permiten en cada directorio. Como esta lista puede ser incompleta, el plugin también prueba - si están habilitados 'pruebas exhaustivas "o" Habilitar pruebas de aplicaciones web "está ajustado a" sí 'en la política de exploración - diversos métodos conocidos HTTP en cada directorio y los considera como no soportado si recibe un código de respuesta de 400, 403, 405, o 501. Nota que la salida plug-in sólo es informativo y no indica necesariamente la presencia de cualquier vulnerabilidad de seguridad.

- **HTTP Server Type and versión**

  Este plugin intenta determinar el tipo y la versión del servidor web remoto.

- **Hypertext Transfer Protocol (HTTP) Information**

  Esta prueba da un poco de información sobre el protocolo HTTP remoto - la versión utilizada, ya sea HTTP Keep-Alive y HTTP pipelining están

habilitados, etc. Esta prueba es sólo informativo y no denota ningún problema de seguridad.

- **PHP versión**

  Este plugin intenta determinar la versión de PHP disponible en el servidor web remoto.

- **Web Server Directory Enumeration**

  Este plugin intenta determinar la presencia de varios directorios comunes en el servidor web remoto. Mediante el envío de una solicitud de un directorio, el código de respuesta del servidor web indica si es un directorio válido o no.

Evidencias:

- **Nessus SYN Scanner**

  El puerto 22 / tcp se encontró que era abierto

- **Apache Banner Linux Distribution Disclosure**

  La distribución de Linux detecta fue:

  - CentOS 7

- **Backported Security Patch Detection (PHP)**

  Dar las credenciales de Nessus para realizar comprobaciones locales.

- **HTTP Methods Allowed**

  Sobre la base de la respuesta a una solicitud de OPTIONS:

  Método HTTP GET HEAD OPTIONS POST TRACE se permite en:
  /
  /icons

- **HTTP Server Type and version**

  El tipo de servidor web remoto es:

  Apache / 2.4.6 (CentOS) PHP / 5.4.16

- **Hypertext Transfer Protocol (HTTP) Information**

Versión Protocolo: HTTP / 1.1
SSL: no
Keep-Alive: sí
Opciones permitidas: (No implementado)
Encabezados:
Fecha: Jue, 16 de julio 2015 22:42:05 GMT
Servidor: Apache / 2.4.6 (CentOS) PHP / 5.4.16
Última modificación: Jue, 16 de octubre 2014 13:20:58 GMT

- **PHP versión**

  Nessus fue capaz de identificar la siguiente información de la versión de PHP.

Versión: 5.4.16
Fuente: Servidor: Apache / 2.4.6 (CentOS) PHP / 5.4.16
Fuente: http://192.168.0.8/info.php

- **Web Server Directory Enumeration**

  Se descubrieron los siguientes directorios:

  /cgi-bin, /icons

Recomendación para solucionar esta Vulnerabilidad:

- **Nessus SYN Scanner**

  Proteja su objetivo con un filtro IP.

- **Apache Banner Linux Distribution Disclosure**

  Si no desea mostrar esta información, editar 'httpd.conf' y establecer la directiva 'ServerTokens Prod' y reiniciar Apache.

- **HTTP Server Type and version**

  Puede configurar la directiva 'ServerTokens Prod' para limitar la información que emana desde el servidor en sus cabeceras de respuesta.

2.3 HARDENING INICAL DEL SERVIDOR WEB.

- **Actualizar el sistema**

  Se debe de tener el servidor con las últimas actualizaciones del sistema operativo, así nos ayuda a que los paquetes del sistema cuenten con las debidos parches actualizados, para evitar fallas de seguridad.

```
yum -y update
```

- **Crear un nuevo usuario con privilegios**

  Crear un nuevo usuario con privilegios de administrador, para evitar usar el usuario root en conexiones remotas.

```
useradd $username
passwd  $username
usermod -a -G wheel $username
```

- **Instalar y Configurar fail2ban**

  Instalamos y configuramos Fail2ban sirve para controlar ataques de fuerza bruta sobre servicios primordiales dentro del ambiente de servidores web como son: ssh, ftp entre otros.

  Se procede a la instalación del servicio fail2ban:

```
rpm -Uvh http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-5.noarch.rpm
yum install fail2ban –y
```

  Luego procedemos a realizar la configuración, mediantes sus principales archivos con la finalidad de establecer los principales parámetros que nos ayudaran a detener ataques de fuerza bruta.

Copiamos el archivo de configuración por defecto de fail2ban como jail.local que será ejecutado y nos sobrescribirá las configuraciones que tengamos en jail.conf.

```
[DEFAULT]

#Las IPs que no quieres bloquear, tu red interna por ejemplo
ignoreip = 127.0.0.1
#El tiempo de bloqueo en segundos.
bantime  = 600
#El maximo numero de intentos
maxretry = 2
#Especifica el backend que se utiliza para detectar cambios en el logpath
backend  = systemd
banaction = firewallcmd-ipset
findtime = 600
chain = INPUT
action_ = %(banaction)s[name=%(__name__)s, port="%(port)s", protocol="%(protocol)s", chain="%(chain)s"]
action_mw = %(banaction)s[name=%(__name__)s, port="%(port)s", protocol="%(protocol)s", chain="%(chain)s"]
              %(mta)s-whois[name=%(__name__)s, dest="%(destemail)s", protocol="%(protocol)s", chain="%(chain)s"]
action_mwl = %(banaction)s[name=%(__name__)s, port="%(port)s", protocol="%(protocol)s", chain="%(chain)s"]
               %(mta)s-whois-lines[name=%(__name__)s, dest="%(destemail)s", logpath=%(logpath)s, chain="%(chain)s"]
action = %(action_mw)s

#El servicio a vigilar ssh
[sshd]
enabled = true
port = ssh
filter = sshd
action = iptables[names=SSH, port=ssh, protocol=tcp]
logpath = /var/log/auth.log
maxretry = 3

[dropbear]
enabled  = false
port = ssh
filter = sshd
logpath = /var/log/dropbear
maxretry = 6

[pam-generic]
enabled = false
filter = pam-generic
port = all
banaction = iptables-allports
```

Figura 2.6 Configuración de Fail2ban

- **Asegurando SSH.**

  Se ha convertido en el estándar para el acceso remoto, reemplazando al protocolo telnet. SSH ha hecho que los protocolos como telnet sean redundantes, en su mayoría, por el hecho de que la conexión es encriptada y las contraseñas no son enviadas en texto plano para que todos la puedan ver.

Cambiamos el número de puerto 22 para evitar posibles ataques de script que atacan directamente al puerto 22. Se cambiara el la opción de Protocol 2 ya que la 1 está en desuso. Deshabilitar el usuario root.

```
vi /etc/ssh/sshd_config
Port 2020
# The default requires explicit activation of protocol 1
Protocol 2
PermitRootLogin no
AuthorizedKeysFile      .ssh/authorized_keys
```

2.4 CONFIGURACIÓN DE ARCHIVO HTTPD.CONF

El archivo httpd.conf tiene varias directivas o reglas que nos ayuda a que nuestro servidor web este seguro a los frecuentes ataques informáticos. [4] (Inteco, 2012)

Con el comando httpd -V, nos muestra la versión del Apache que se está ejecutando y la ruta donde se encuentra alojado el archivo de configuración.

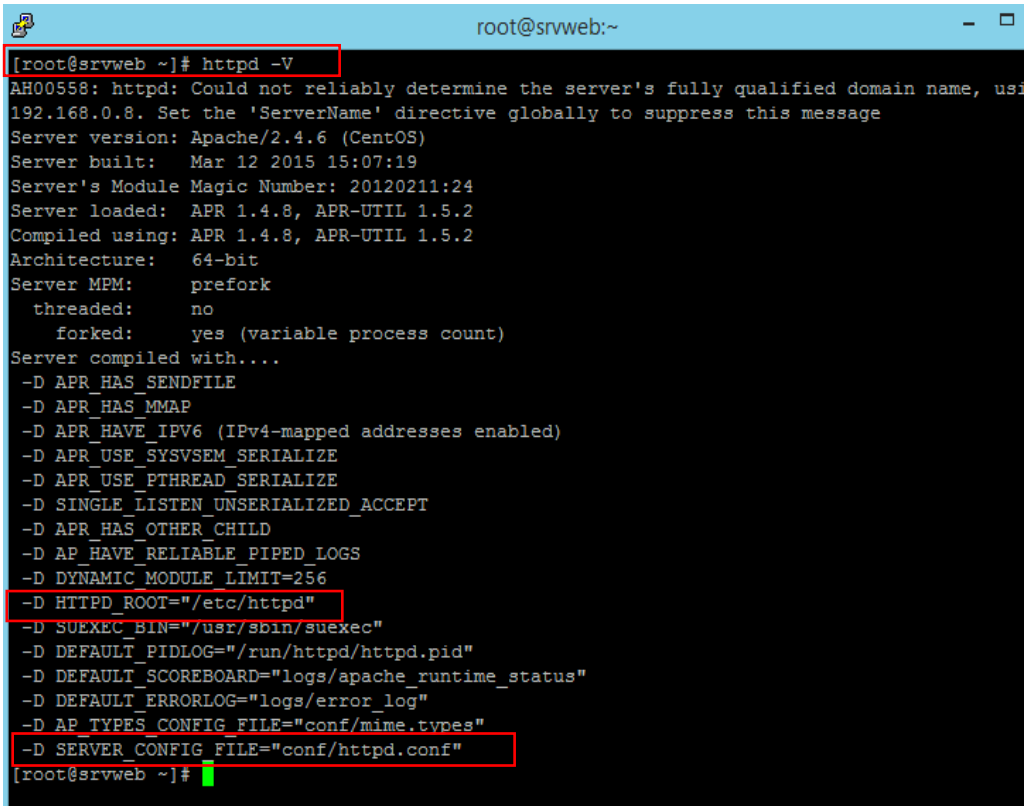

Figura 2.7 Comando http -V

- **Control de los archivos que se publican.**

  El problema más frecuente en los servidores web es de no aplicar las directivas adecuadas para el control de documentos que se alojan en directorios donde son alojadas las páginas web.

  DocumentRoot /var/www/html

- **Denegar el acceso a los archivos por defecto.**

  Se realiza la denegación de acceso a todos los directorios por defecto y luego se permite el acceso sólo a los directorios específicos en el DocumentRoot explícitamente:

- **Desactivar el listado de directorios**

  Desactivar la lista de directorios en el navegador esta directiva puede tener valores como **All** y **None** para activar o desactivar todas las opciones disponibles. Para la configuración del directorio raíz se aconseja desactivar todas por seguridad, lo que incluiría la opción None o Indexes.[2] (Kuma, 2015)

Figura 2.8 Desactivar el acceso a los listados de directorios

- **Retirar información de versión del Server.**

  Esta es una de las primeras cosas a tener en cuenta, es la de exponer la versión del servidor web que se está utilizando. La exposición de la versión significa que está ayudando a los atacantes a dar información muy útil. La configuración predeterminada expondrá Apache Versión y tipo de sistema operativo. [2] (Kuma, 2015)

```
SerververTokens Prod
ServerSignature Off
```

La directiva ***ServerSignature*** eliminará la información de versión de la página generada como 403, 404, 502, etc. ***Servertokens*** cambiara la cabecera de producción única, es decir Apache.

- **Disminuye el valor máximo de tiempo de espera.**

  Por el defecto el tiempo de espera es de 300 segundos. Puedes disminuirlo por seguridad para prevenir ataques de esta manera:

```
Timeout 45
```

- **Restricciones sobre las peticiones.**

  Con LimitRequestBody especificamos el tamaño máximo en bytes que se pueden especificar en una petición.

  LimitRequestBody nos puede ayudar de varias maneras:

  - Defendiéndonos ante posibles denegaciones de servicio.
  - Limitando el tamaño de los ficheros que un usuario puede subir a nuestro servidor.

  Acceptfilter: Configura optimizaciones para sockets de escucha de un protocolo

  LimitRequestBody: Restringe el tamaño total del cuerpo de las peticiones HTTP enviadas desde el cliente

  LimitXMLRequestBody: Limita el tamaño del cuerpo de una petición XML

LimitRequestFields: Limita el número de campos de la cabecera de las peticiones HTTP del cliente que serán aceptadas

LimitRequestFieldSize: Limita el tamaño permitido de las cabeceras de las peticiones HTTP de los clientes

LimitRequestLine: Limita el tamaño la línea de petición HTTP que será aceptada.

```
AcceptFilter http data
AcceptFilter https data
LimitRequestBody 1048576
LimitXMLRequestBody 10485760
LimitRequestFields 32
LimitRequestFieldSize 8000
LimitRequestLine 4000
```

- **HttpOnly and Secure flag**

  Sin tener HttpOnly y Secure flag en el encabezado de respuesta HTTP, es posible robar o manipular sesión de aplicación web y cookies.

```
Header edit Set-Cookie ^(.*)$ $1;HttpOnly;Secure
```

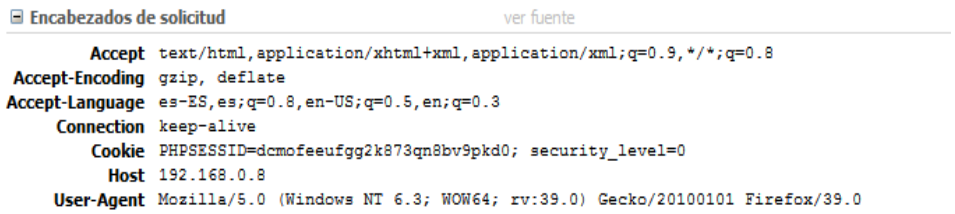

Figura 2.9 HttOnly y Secure flag

- **Server Side Include**

  Ataque SSI permite la explotación de una aplicación web mediante la inyección de secuencias de comandos en las páginas HTML o ejecutar código de forma remota.

- **Configuración de Apache con soporte SSL/TLS.**

  Los servidores web utilizan HTTP por defecto, que es un protocolo de texto claro. Como su nombre indica, un protocolo de texto no cifrado que no aplica ningún tipo de encriptación de los datos. Puede exponerse a cualquier ataque "man-in-the-middle" capaz de ver el contenido de los paquetes en tránsito con analizadores de paquetes cuidadosamente colocados.

  HTTPS permite que una página proporcione los siguientes servicios:
  Asegura que todos los paquetes de tránsito hacia y desde los servidores están cifrados.

  Establece la confianza con un certificado digital oficial, de manera que los servidores impostores no pueden pretender ser el servidor real.
  La primera cosa necesaria para la creación de HTTPS es un certificado digital. Los certificados digitales se pueden obtener de cualquiera de los métodos siguientes.

  Ahora agregamos el SSL.

Volvemos a teclear en la terminal lo siguiente:

```
yum install mod_ssl –y
mkdir /etc/httpd/ssl
openssl req -x509 -nodes -days 365 -newkey rsa:2048 –keyout
/etc/httpd/ssl/apache.key -out /etc/httpd/ssl/apache.crt
```

Procedemos a configurar el archivo ssl.conf:

```
vi /etc/httpd/conf.d/ssl.conf
DocumentRoot "/var/www/html"
ServerName www.srvweb.com:443
SSLCertificateFile /etc/httpd/ssl/apache.crt
SSLCertificateKeyFile /etc/httpd/ssl/apache.key
```

Reiniciamos el servicio httpd:

```
sudo service httpd restart
```

Habilitamos en el firewall el puerto 443

```
sudo firewall-cmd --permanent --add-port=443/tcp
sudo firewall-cmd --reload
sudo service httpd restart
```

2.5 CONFIGURACIÓN DE ARCHIVO PHP.INI

PHP es un lenguaje de programación orientado a la creación de páginas webs dinámicas o aplicaciones web. Es uno de los lenguajes más extendidos en este ámbito y por ello debemos proteger y configurar nuestros servidores lo mejor posible para evitar un acceso no autorizado.

La configuración de PHP se encuentra en un archivo ini, llamado php.ini.

```
# Restringir la exposición de información de PHP
expose_php = Off

# Restringir los mensajes de error de PHP
display_errors = Off

# Loguear todos los errores PHP
log_errors = On
error_log = /var/log/php_scripts_error.log

# Restringir el tamaño de archivos a subir
file_uploads = on
upload_max_filesize = 1M

# Deshabilitar la ejecución remota de código
allow_url_fopen = Off
allow_url_include = Off

# Configuración de control de recursos (mitigación de DoS)
# Configuración en segundos
max_execution_time = 30
max_input_time = 30
memory_limit = 128M

# Deshabilitar funciones PHP inseguras
```

```
disable_functions = exec, passthru, shell_exec, system,proc_open, popen,curl_exec, curl_multi_exec, parse_ini_file, show_source, eval, base64_decode
```

Limitar el acceso de PHP al sistema de archivos

```
open_basedir = "/var/www/"
```

2.6 INSTALACIÓN Y CONFIGURACIÓN DE PORTSENTRY.

Portsentry es una aplicación que provee protección contra ataques de barrido de puerto y ataques de DDoS, la aplicación utiliza las bondades de tcpwrappers e iptables para bloquear hosts con comportamientos anormales.

2.7 INSTALACIÓN Y CONFIGURACIÓN FIREWALL DE APLICACIÓN MODSECURITY.

Mod_security (una detección de intrusiones de código abierto y el motor de la prevención para las aplicaciones web que se integra a la perfección con el servidor web) esta herramienta es muy importante ya que se pueden utilizar para proteger un servidor web en contra de la fuerza bruta o ataques DoS.

Instalar y configurar ModSecurity".

```
yum install mod_security mod_security_crs
systemctl restart httpd.service
```

Cuando la instalación se haya completado, usted encontrará los archivos de configuración en /etc/httpd/conf.d.

```
[root@srvweb etc]# ls -l /etc/httpd/conf.d/
total 20
-rw-r--r--. 1 root root 2893 mar 12 10:07 autoindex.conf
-rw-r--r--. 1 root root 2139 jun  9  2014 mod_security.conf
-rw-r--r--. 1 root root  366 mar 12 10:08 README
-rw-r--r--. 1 root root 1252 mar 12 09:57 userdir.conf
-rw-r--r--. 1 root root  824 mar 12 09:57 welcome.conf
[root@srvweb etc]#
```

Figura 2.10 Archivo de configuración de mod_security.conf

```
# mkdir /etc/httpd/crs
# cd /etc/httpd/crs
# wget https://github.com/SpiderLabs/owasp-modsecurity-crs/tarball/master
# tar xzf master
# mv SpiderLabs-owasp-modsecurity-crs-ebe8790 owasp-modsecurity-crs
```

Instalación del conjunto de reglas Core (también conocido como CRS) ofrece el servidor web con instrucciones sobre cómo comportarse en determinadas condiciones.

```
root@srvweb crs]# ll
otal 280
rw-r--r--. 1 root root 280014 jul 21 12:14 master
rwxrwxr-x. 9 root root   4096 jul 21 12:18 owasp-modsecurity-crs
root@srvweb crs]#
```

Figura 2.11 Reglas de modsecurity-crs

Ahora es el momento de configurar mod_security. Ingresamos a la carpeta owasp-modsecurity-crs, y copiamos el archivo de ejemplo modsecurity_crs_10_setup.conf.example en otro archivo sin la extensión .example:

```
cp modsecurity_crs_10_setup.conf.example modsecurity_crs_10_setup.conf
```

Cargamos el modulo y parámetros en nuestro web server dentro de nuestro archivo de configuración /**etc/httpd/conf/httpd.conf**.

```
LoadModule security2_module modules/mod_security2.so
<IfModule security2_module>
   Include crs/owasp-modsecurity-crs/modsecurity_crs_10_setup.conf
   Include crs/owasp-modsecurity-crs/base_rules/*.conf
</IfModule>
```

Luego procedemos a reiniciar el apache y revisamos el comportamiento de nuestro servidor web.

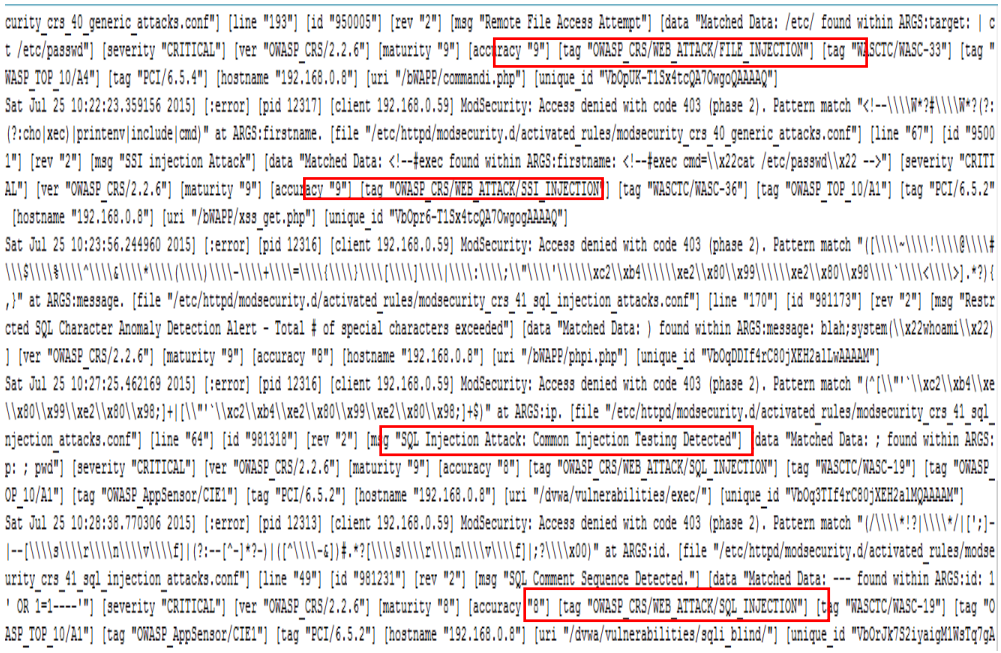

Figura 2.12 Archivo Logs de Http

La línea de ModSecurity en el registro de errores de Apache se divide en nueve elementos. Cada elemento proporciona información acerca de por qué se desencadenó el evento.

- La primera parte dice qué archivo de reglas desencadenó este evento.

- La segunda parte dice lo que la línea en el archivo de reglas la regla comienza el.
- El tercer elemento que le dice qué regla se dispara.
- El cuarto elemento que le dice a la revisión de la regla.
- El quinto elemento contiene datos especiales para fines de depuración.
- El sexto elemento define la gravedad de registro de esta gravedad del suceso.
- La séptima sección describe lo que ocurrió la acción y en qué fase se ha producido.

Mod_evasive es un módulo de apache que provee al servidor acciones de evasión en caso de un ataque Dos, DDos o Fuerza Bruta. Está diseñado además como herramienta de detección y administración y que puede ser configurado para interactuar con Firewalls, Iptables, entre otros.

Instalar y configurar Mod_Evasive.

```
yum install httpd-devel gcc
yum install mod_evasive
systemctl restart httpd.service
```

CAPÍTULO 3

RESULTADOS DE LA REVISIÓN.

3.1 PRUEBAS INTERNAS.

Se realiza pruebas con la aplicación bWPP, se trata de una aplicación web vulnerable, especialmente con propósitos educativos, la cual nos ayuda a descubrir y observar varios tipos de vulnerabilidades, que los atacantes informáticos pueden aprovechar. [5] (Mesellem, 2014)

- ✓ OS Command Injection
- ✓ XSS – Reflected
- ✓ PHP Code Injection

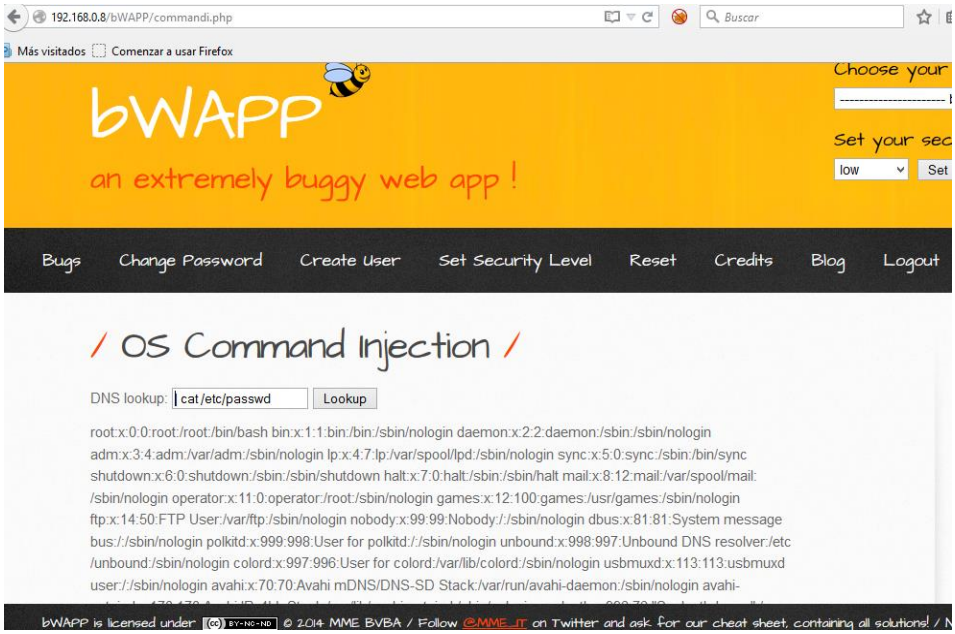

Figura 3.1 Prueba de OS Command Injection

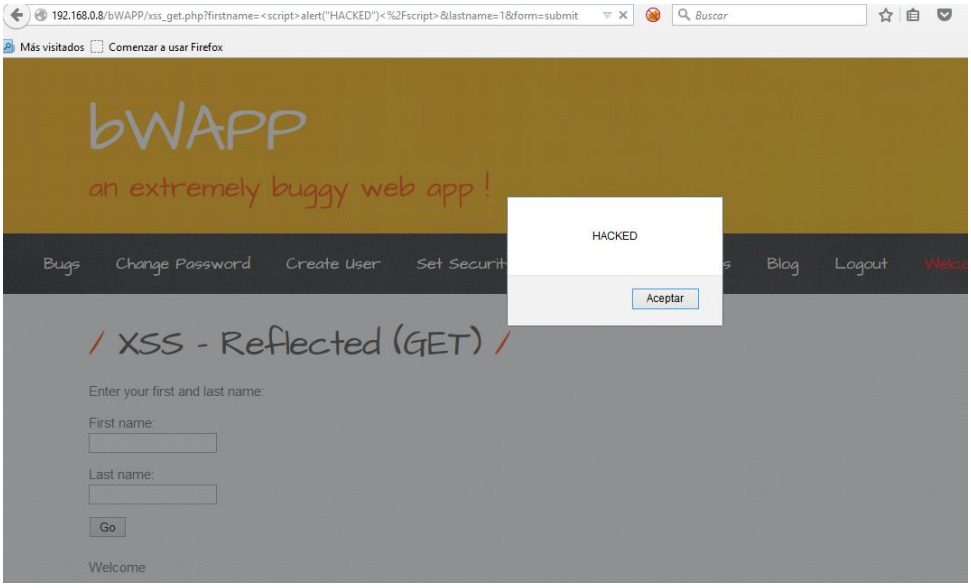

Figura 3.2 Prueba de XSS – Reflected

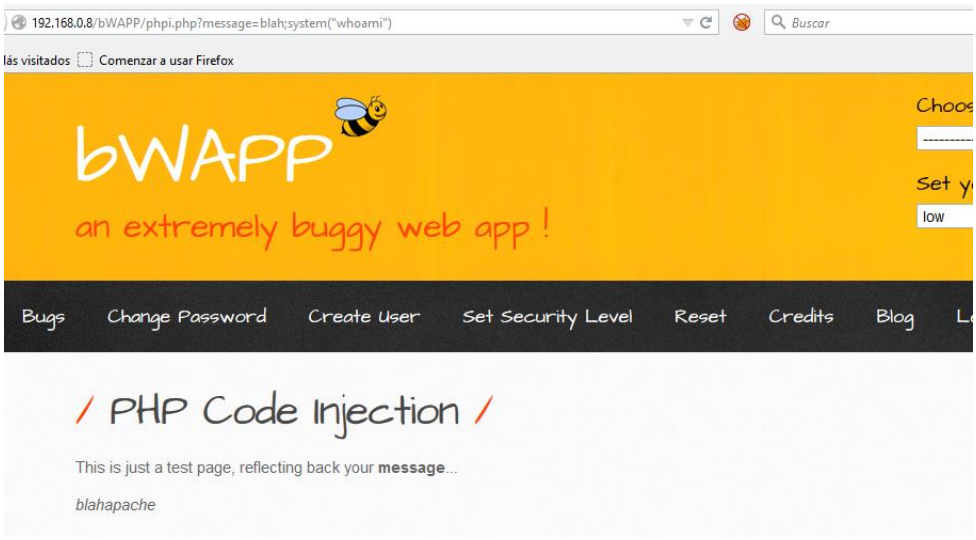

Figura 3.3 Prueba Code Injection

Acceso a listado de archivos.

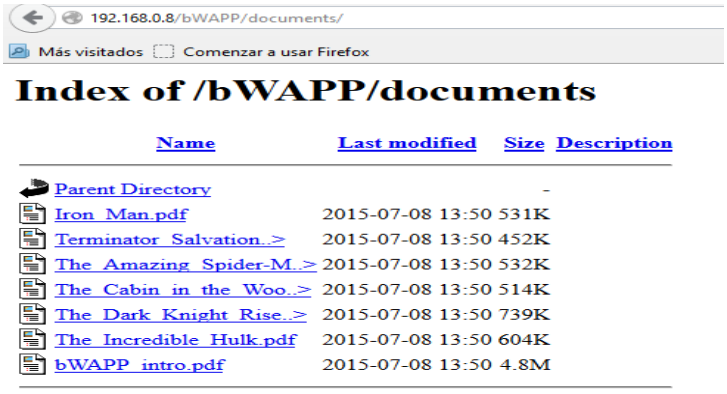

Figura 3.4 Listado de Ficheros

Información de Cabeceras.

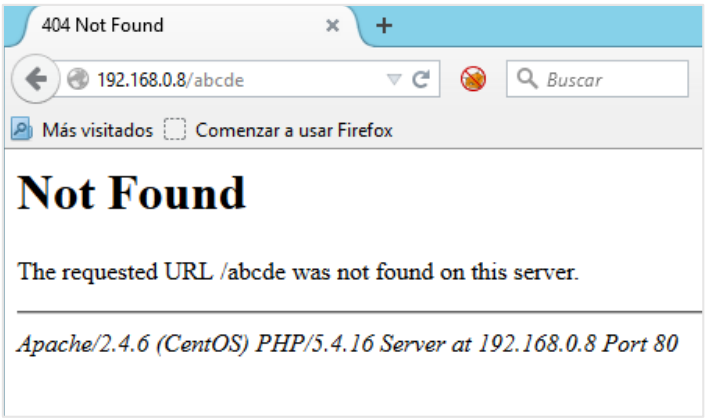

Figura 3.5 Información de cabeceras

Navegación con protocolo HTTP.

http://192.168.0.8

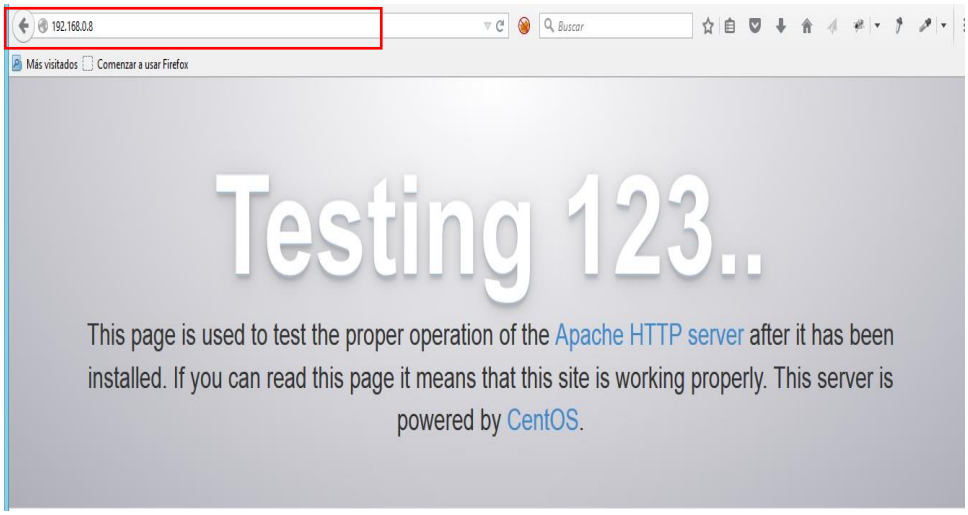

Figura 3.6 Barra de navegación con HTTP

3.2 INFORME DE PRUEBAS.

Aquí se detalla el informe de pruebas luego del hardening realizado al servidor web, la configuración de los servicios apache, php, instalando herramientas como fail2ban, mod_security, mod_evasive, podemos demostrar que el servidor está evitando ataques que pueden ocasionar denegación de servicios.

Fail2ban. Podemos ver en los logs que está bloqueando las conexiones remotas que intentan accesos por fuerza bruta.

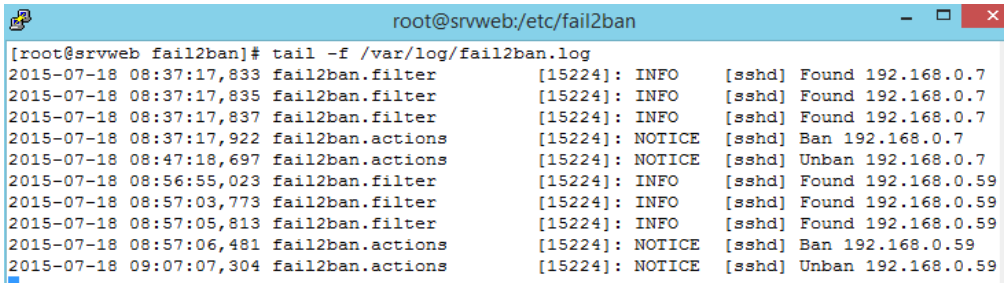

Figura 3.7 Bloqueo de accesos por fuerza bruta

Ssh. Podemos observar cuando queremos ingresar con usuario root no permite ingresar ya que se encuentra deshabilitado, y el puerto 22 ya no se encuentra disponible.

```
[root@srvhdr ~]# ssh root@192.168.0.8
ssh: connect to host 192.168.0.8 port 22: No route to host
[root@srvhdr ~]# ssh -p 2020 root@192.168.0.8
root@192.168.0.8's password:
Permission denied, please try again.
```

Aquí se realizó la prueba con el usuario administrator y el puerto 2020 y tenemos acceso exitoso.

```
[root@srvhdr ~]# ssh -p 2020 administrator@192.168.0.8
administrator@192.168.0.8's password:
Last login: Sat Jul 18 12:25:31 2015 from 192.168.0.59
[administrator@srvweb ~]$
```

Figura 3.8 Ingreso de ssh al puerto 2020 usuario administrator

Escribimos en la barra de navegación lo siguiente: **https://192.168.0.8**

Figura 3.9 Verificación de protocolo https

Aquí observamos que ya aparece con el protocolo https.

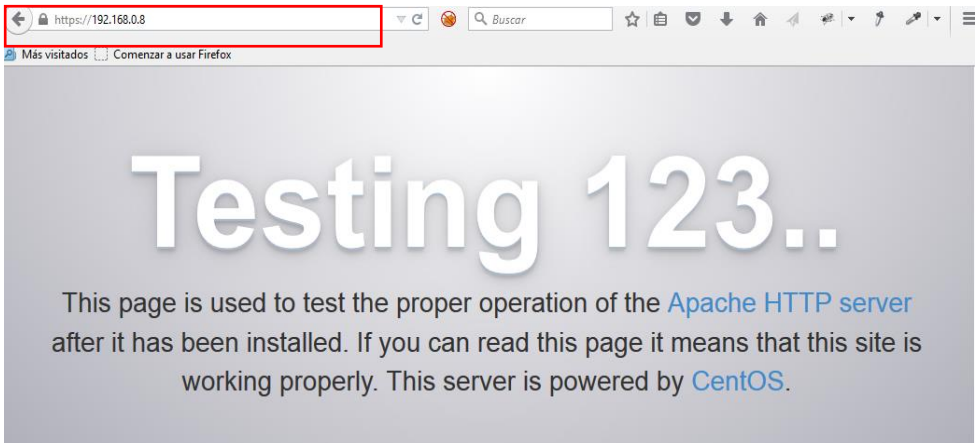

Figura 3.10 Navegación con HTTPS

La OWASP ModSecurity Core Rule Set (CRS) proporciona protecciones contra la siguiente categoría de ataques. [3] (OWASP, 2015)

- Protección HTTP – detecta violaciones del protocolo HTTP y una política de uso definido localmente.
- En tiempo real búsquedas lista negra - utiliza Reputación tercera Partido IP.
- HTTP Denegación de Servicio Protecciones - defensa contra inundaciones HTTP y HTTP lentos ataques de denegación de servicio.
- Ataques Web Común de Protección - detecta ataques a la seguridad de aplicaciones web común.
- Detección Automatización - detecta bots, crawlers, escáneres y actividad maliciosa otra superficie.
- Integración con Escaneo AV para la subida de archivos - detecta archivos maliciosos subidos a través de la aplicación web.
- Protección de Troya - detecta el acceso a los troyanos.

Logs de bloqueo usando mod_security con owasp_crs.

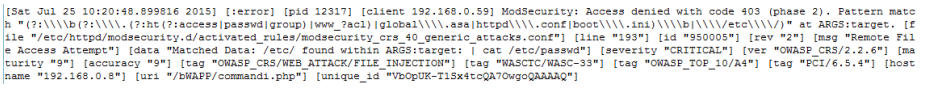

Figura 3.11 Tag: Web Attack/File Injection

.

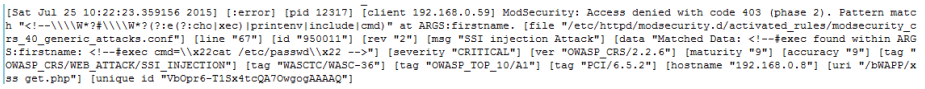

Figura 3.12 Tag: Web Attack/Ssi_Injection

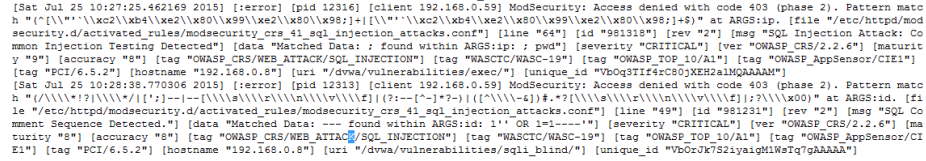

Figura 3.13 Tag: Web Attack/Sql_Injection

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES.

1. Asegurar nuestro servidor web es importante para proteger nuestros datos y propiedad intelectual de la mano de los Hackers.

2. Cuando se procede a la instalación de un Servidor Web Apache, sabemos que viene una configuración por defecto, la cual son insuficientes para ser utilizadas como un servidor web seguro, ya que estaría vulnerable a ataques informáticos.

3. Se recomienda la implantación de políticas y procedimientos de actualización y aplicación de parches de seguridad sobre el servidor web.

RECOMENDACIONES.

1. A nivel general remover de los directorios de publicación los archivos info.php, los cuales brindan información de configuración y demás sobre el servidor web.

2. Es indispensable verificar los servicios que se usan desde los equipos clientes y establecer las reglas del firewall, esto con el fin de que solo estén públicos los servicios que se requieren de esta forma.

BIBLIOGRAFÍA

[1] Wikipedia, Hypertext Transfer Protocol+, https://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol, 14 Octubre 2012

[2] Chandan Kuma, Apache Web Server Hardening & Security Guide, http://geekflare.com/apache-web-server-hardening-security/, 14 Febrero 2015

[3] OWASP, OWASP ModSecurity Core Rule Set (CRS), https://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project, 22 Abril 2015

[4] Inteco, Guía básica para la securización del servidor web Apache, https://www.incibe.es/file/5j9r8LaoJvysvd-q5JFcbQ, Julio, 2012

[5] Malik Mesellem, bWAPP a buggy web application!, http://itsecgames.blogspot.com/, 28 Junio, 2014