Instituto Universitario Aeronáutico

Especialización en Seguridad Informática

Trabajo Final Integrador para obtener el título de Especialista en Seguridad Informática

Auditoría y Segurización de un Servidor Web

Autor: Lic. Alicia D. Castro

Director: Mg. Eduardo Casanovas

-2016-

A Dios por ser mi apoyo y fortaleza y a mi madre por su gran amor y su ayuda continúa.

Agradecimientos

A mi familia por su continuo apoyo y estimulo.

A mis compañeros de estudio y los profesores de la carrera por su permanente entusiasmo, compañerismo, buena actitud y ayuda mutua que permitieron realizar los estudios a distancia de manera entretenida y entusiamada.

Al magister Eduardo Casanovas por su contagioso entusiasmo, por su ayuda continúa en el desarrollo de este trabajo y buena predisposición durante toda la carrera.

A todos ellos, Muchas Gracias!!!

Índice General

Índice de Imágenes

1. Introducción

La seguridad no es una meta, es un proceso. Como tal, la prevención y el constante fortalecimiento de las fronteras del sistema corporativo resultan elementos vitales en la defensa de los activos en el ciberespacio. Los desafíos de seguridad se enfocan en la seguridad de los equipos y de la red. Un servidor web puede ser explotado y a partir de ahí ganar acceso a los datos o al resto de los equipos dentro de la organización que estén conectados al mismo.

La web presenta nuevos retos. La cantidad de personas y compañías que acceden a internet aumenta rápidamente y todos lo hacen a través de browsers web. Por esta razón, muchas compañías, empresas, comercios están entusiasmados en mostrarse a través de la web y permitir a los usuarios realizar sus transacciones en plataformas del negocio en forma online, por ejemplo comercio electrónico, pago de servicios, entre otros. Pero la realidad es que Internet y la web son extremadamente vulnerables a ser comprometidos de varias maneras: daño en la reputación de las empresas, pérdida o robo de información, inactividad del servicio, pérdida de dinero, etc., por lo cual estas empresas necesitan infraestructuras y servicios web seguros.

Los servicios que brinda un servidor web generalmente están expuestos al uso público y accesible desde Internet, por lo cual existen muchas posibilidades de ser atacados, para llevar a cabo estos ataques se utilizan los puntos débiles de cada servicio, estos puntos débiles o "vulnerabilidades" se deben reconocer y mitigar para así proteger al equipo y a la información. También influye la configuración deficiente de los servidores web y el desarrollo de las aplicaciones web utilizando algún framework web con potenciales fallas de seguridad.

Por todo esto, las empresas y organizaciones deben tomar conciencia de la importancia de tener servidores web seguros, que garanticen la disponibilidad del servicio web, la confidencialidad e integridad de los datos en tránsito o almacenados en el servidor.

Situación Problemática

En la actualidad, la mayoría de las organizaciones, independientemente del tamaño de la misma, dispone de un sitio web para divulgar por Internet su negocio, su identidad, su imagen, etc.

Un servidor web puede utilizarse como plataforma de acceso a toda la infraestructura informática de la organización, una vez comprometida la seguridad del servidor web, un atacante podría tener acceso al resto de la infraestructura pudiendo acceder a otros recursos sensitivos de la organización como es la información de la organización, configuración de equipos y acceso a los sistemas.

Es prioritario que los servidores web estén protegidos frente a cualquier tipo de amenazas, ya que siguen siendo un blanco fácil para cualquier tipo de atacante. Los ataques a estos servidores suelen conocerse rápidamente ya que en cuestión de segundos una gran mayoría de la población se dará cuenta de que hay algo que se ha modificado en el servidor comprometido ó el servidor esta caído. La mayor parte de estos ataques, en la actualidad, vienen como consecuencia de una mala configuración del servidor o un mal diseño del mismo, así también como un mal desarrollo de la aplicación web.

Los usuarios pueden fácilmente descargar y ejecutar herramientas de hacking diseñadas para buscar, encontrar y explotar vulnerabilidades conocidas de forma automatizadas y con una alta tasa de éxito. Para evitar esto los administradores de los servidores web deben ejecutar pruebas de penetración periódicas para conocer las vulnerabilidades más comunes a los que están expuestos, este es el primer paso para la protección y representa la mejor oportunidad de bloquear los ataques.

Existe una diversidad de herramientas y dispositivos para analizar, proteger, auditar servidores web, que serán de gran utilidad a la hora de proteger el entorno web. Entre las funciones se pueden mencionar: firewall de host, firewall de aplicación web, monitoreo, análisis de vulnerabilidades, pruebas de intrusión, entre otras.

La seguridad ante las amenazas de Internet dependerá de la destreza del administrador a la hora de configurar las herramientas que garanticen la seguridad de servidor. La persona que adquiere un servidor dedicado es responsable de realizar sus propias tareas de mantenimiento para tener siempre el equipo actualizado y poder hacer frente a posibles amenazas.

Un incidente de seguridad puede generar:

- Pérdidas de datos con las posibles implicaciones legales (registros médicos, registros contables de clientes, etc.) .
- Pérdida directa de beneficios (ventas vía web, el servidor de archivos inactivo, etc.) con la correspondiente afectación económica.
- Costos en tiempo de personal.
- Pérdida de la confianza por parte del cliente.
- Mala Publicidad por parte de los medios de comunicación.

Un servidor web que comunique información (datos) importante debe estar protegidos contra ataques que violen la integridad, confidencialidad o autenticidad estableciendo una navegación segura a través del protocolo HTTPS. Hay que tener presente que el tráfico de la

red e Internet viaja sin cifrar y puede ser leído con facilidad por un atacante, por lo cual hay que utilizar los diferentes mecanismos existentes para cifrar el tráfico de datos por la red. Algunos esquemas sólo cifran los datos que se envían (como el correo cifrado con PGP), otros cifran la sesión (SSL), y algunos cifran la carga de datos de los paquetes (IPSec, VPN's).

Un factor importante en la seguridad de aplicaciones Web es la programación de las mismas. Una programación deficiente conduce a ataques del estilo de ejecución remota de código, SQL Inyection y Cross Site Scripting. En (Figura 1) se muestran algunos ataques a aplicaciones e infraestructuras web.

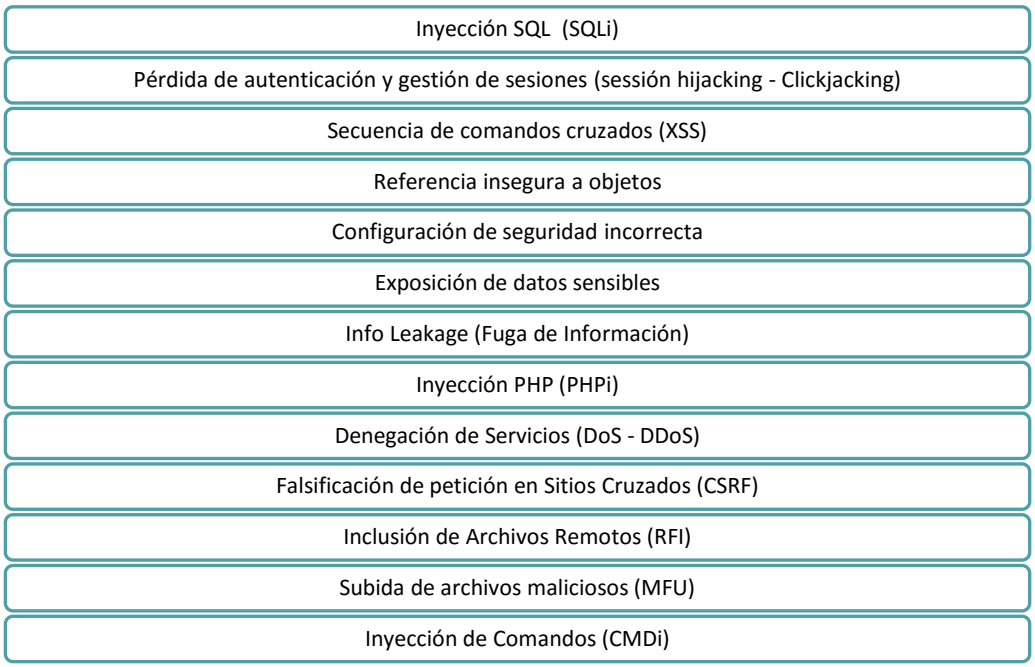


Figura 1. Tipos de Ataques

Análisis estadístico de los ciber ataques

La empresa **Akamai´s** presenta un análisis de los ataques observados en el 2014 y 2015. En Q3 2015 (tercer trimestre del 2015) Akamai ha conseguido mitigar un total de 1.510 ataques de denegación de servicio distribuido, esto supone un incremento del 180% en comparación de Q3 2014, y un 23% más que en Q2 2015. El número de ataques han utilizado menos ancho de banda y menos tiempo en atacar al objetivo, pero han sido más numerosos. Los mega ataques (ataques superiores a 100Gbps) se han reducido a 8 en Q3 2015 comparados con los 12 mega ataques de Q2 2015 y los 17 mega ataques del Q1 2015. El mayor ataque DDoS en Q3 2015 utilizó el botnet XOR DDoS empleando 149Gbps, mientras que en Q2 2015 fue de 250Gbps (1).

El ataque a aplicaciones web representa el 55% de todos los ataques.

Comparando 2014 y Q1 2015 se puede observar que el foco de ataque en más de 100Gbps de ataques DDoS son ataques de reflexión SSDP[1] representando el primer tipo de ataques sobre infraestructura, ya que afectan a dispositivos hogareños o equipos sin segurizar, incluso escapan a la detección y mitigación. Luego continúan los ataques de inundación usando SYN y luego inundación usando UDP. Los ataques DDoS en capa de aplicación continúan siendo un riesgo, ya que los atacantes aprovechan proxies abiertos en internet. Del análisis estadístico se observa que sigue incrementándose en el primer periodo del 2015 los ataques DDoS.

Los análisis de la mala configuración de sitios web e intentos de obtención de dominio en Q1 del 2015 han relevado que mucho de los sitios afectados comparten la misma dirección IP y han sido hosteados en el mismo servidor. Si un atacante puede encontrar un sitio web vulnerable, entonces puede infectar al servidor web entero.

Esta empresa también sugiere usar TLS en vez de SSL dado los problemas de Vulnerabilidades como Poodle, Shellshock y Heartbleed que surgieron en el 2014

Con respecto a las estadísticas relacionadas con ataques a aplicaciones web, Akamai analizo más de 178 millones de ataques a aplicaciones web, enfocándose en los siete más populares, los cuales incluyen SQL inyection (SQLi), inclusión de archivos locales (lfi), inclusión de archivos remotos (rfi), inyección php (PHPi), inyección de comandos (CMDi), inyección ognl en JAVA (JAVAi) y subir archivos maliciosos (mfi).

Otro estudio del 2014 realizado por ***IT Governance,*** sobre los ataques a infraestructura web, indica que una forma efectiva de defenderse frente a los ciber ataques es realizar pruebas de penetración regularmente, donde se puedan conocer las vulnerabilidades, siendo el primer paso en el proceso de protección. El informe indica que 30.000 sitios web son hackeados diariamente a través de la distribución de malware, el 96% de los sitios web tienen vulnerabilidades y en esa misma proporción tienen las aplicaciones en promedio 14 vulnerabilidades, luego de un ataque pasan en promedio 200 días para detectar este hecho. Más de un millón de sitios web que usan wordPress son vulnerables a ataques SQL Inyección y por lo mismo aproximadamente 50.000 sitios son hackeados usando esta aplicación y esta vulnerabilidad. El 80% de los sitios web corresponden a pequeñas empresas.

Los números que muestra este informe permiten observar con claridad la importancia de segurizar los servidores web.

El reporte de vulnerabilidades de ***Ceniz*** en el 2013 (2), indicó que los mayores ataques a aplicaciones web es a través de ataques XSS, fuga de información, clickjacking (administración

---

1 Los ataques de reflexión SSDP son un tipo de ataques que amplifican el ataque de denegación de servicios distribuida (DDoS) abusando del protocolo SSDP (Protocolo de descubrimiento de servicios Simple) que viene en millones de dispositivos hogareños o de oficina (routers, servidores medianos, web cams, smart TVs e impresoras).

de sesión) e inyecciones en PHP, debido a la deficiencias en el código de programación de la aplicación y por las fallas en la sanitización de la infraestructura de la aplicación web. El siguiente gráfico (Figura 2) muestra el porcentual de ataques analizados por Ceniz.

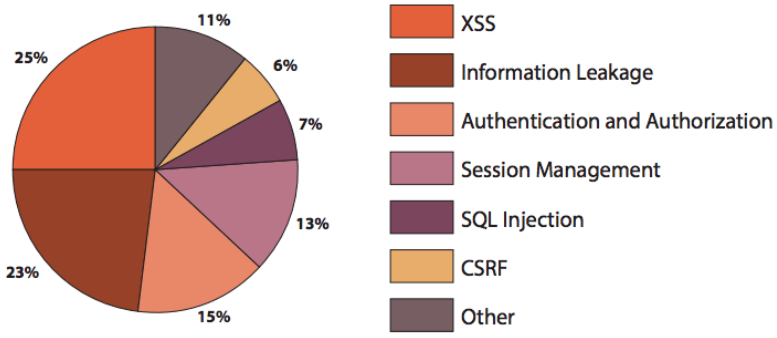


Figura 2. Gráfico porcentual de Ataques en Aplicaciones WEB

1.1. Objetivos y Alcance del Trabajo

1.1.1. Objeto de Estudio

- Auditoría del sistema operativo y los servicios web.
- Revisión y configuración de seguridad del sistema operativo.
- Efectuar una configuración de seguridad a los distintos servicios remotos (web, base de datos, FTP).
- Análisis de vulnerabilidades de los servicios web.
- Brindar continuidad y disponibilidad de los servicios web.
- Evitar la posibilidad de uso ilegal o impropio de los servicios web.
- Evitar borrado o modificación de datos (páginas web, información de bases de datos, deterioro del sitio web, etc.).
- Ofuscar información brindada por el servidor web.

1.1.2. Objetivos Generales

El presente trabajo tiene los siguientes objetivos generales:

- Auditar la seguridad en las configuraciones del servidor Linux dedicado.
- Evaluar la seguridad en las diferentes conexiones remotas.

- Realizar un análisis de riesgos y vulnerabilidades del equipo.
- Segurizar el equipo, desde el Sistema Operativo hasta los diferentes servicios instalados.
- Brindar una autenticación segura a las aplicaciones web existentes
- Resguardar la información y las configuraciones de manera segura
- Monitorizar el funcionamiento del servidor
- Asegurar la disponibilidad del servidor web.
- Controlar el acceso a los diferentes recursos del sistema.
- Garantizar la integridad en los datos almacenados.
- Brindar confidencialidad en las comunicaciones remotas.

1.1.3. Objetivos Específicos

De los objetivos generales se desprenden los siguientes objetivos específicos.

- Realizar un análisis de vulnerabilidades sobre el entorno web.
- Evaluar y configurar la seguridad del Sistema Operativo Linux.
- Evaluar la seguridad del servicio web apache y realizar las configuraciones necesarias para tener un servidor web seguro.
- Evaluar la seguridad y aplicar las configuraciones para lograr seguridad en la transferencia de archivos (ftp).
- Diagnosticar y segurizar las bases de datos: Mysql y postgres y la aplicación web de administración de las mismas.
- Valorar la seguridad en el acceso remoto e implementar las correcciones necesarias para un acceso remoto seguro.
- Crear certificados digitales para la comunicación segura en el servidor web y lograr así autenticación, confidencialidad e integridad en comunicación web y FTP.
- Configurar un firewall de host para evitar ataques DDoS y denegar accesos no autorizados.
- Realizar backup de información y configuraciones encriptados con el correspondiente copiado cifrado de los archivos generados en otro equipo.
- Tener un servidor de respaldo para lograr disponibilidad ante caídas del equipo principal.
- Establecer alertas para la protección del equipo.
- Monitorear eventos que correspondan a potenciales ataques.

1.1.4. Delimitación Del Trabajo

El desarrollo del presente trabajo tiene el siguiente alcance:

- Auditoría del servidor web.
- Análisis de vulnerabilidades de las configuraciones del sistema operativo y los servicios activos.
- Análisis y soluciones a los accesos a los diferentes servicios del sistema.
- Aplicación de seguridad sobre el Sistema Operativo y sobre las distintas configuraciones de los servicios activos.
- Cifrar las comunicaciones remotas.
- Análisis de las configuraciones correspondientes a los módulos de los lenguajes de programación.

No se considerará:

- La infraestructura de red de la organización;
- Los usuarios de los servicios;
- La política de seguridad de la organización;
- El código de las aplicaciones.

El siguiente gráfico (Figura 3) permite observar los servicios de seguridad y los métodos que se utilizarán para lograrlo, que se tendrán en cuenta en el presente trabajo.

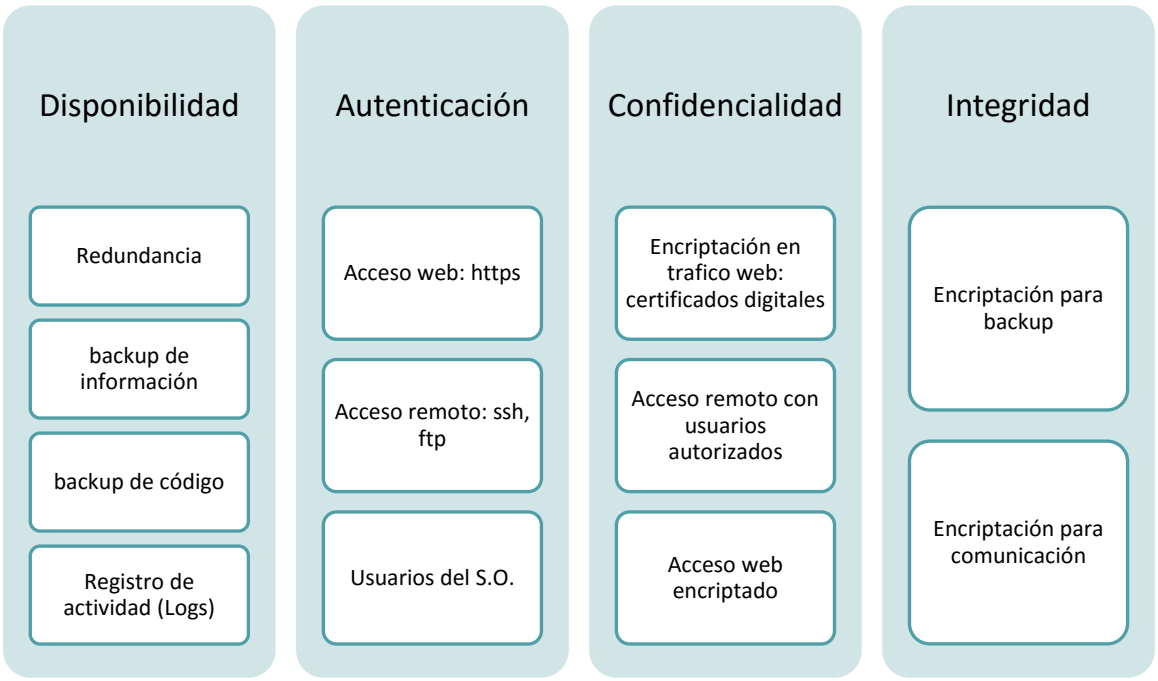


Figura 3. Servicios de Seguridad Informática y sus incumbencias en un servidor web

Se espera que este trabajo sirva de referencia a administradores de infraestructura web para lograr mayor seguridad y privacidad de sus activos, pudiendo aplicar las medidas de seguridad indicadas en este informe, en su infraestructura. Así también a los desarrolladores web ofrecerles un resumen de recomendaciones de seguridad para el desarrollo de sus aplicaciones web.

1.1.5. Metodología del trabajo

El desarrollo de este proyecto seguirá la metodología propuesta por Garzón (3), agrupadas en fases y definidas por etapas.

La metodología está basada en seis fases:

1. Planeación,
2. Políticas de Seguridad,
3. Aseguramiento Físico,
4. Arquitecturas de red,
5. Aseguramiento y configuración de Sistemas Operativos y Servicios,
6. Auditoría de Sistemas.

La fase de políticas de seguridad no será contemplada dado las características de la arquitectura planteada. El resto de las fases serán llevadas a cabo en forma parcial o total según corresponda.

Partiendo de la identificación y mitigación de los riesgos a los que se encuentra expuesta la información publicada en el servidor web, continuando por la segurización (hardening) del equipo, ya sea sistema operativo como los servicios activos.

A continuación se describe el propósito de cada fase:

Fase 1: Planeación

Con el fin de crear un plan de aseguramiento de la compañía, inicialmente se debe estudiar el cometido de la misma para conocer su misión y visión, y de esta manera generar una solución de seguridad de acuerdo a sus necesidades y requerimientos. Posteriormente se ha de realizar un análisis de los datos y la información de la compañía, en donde se establecerán los recursos que se van proteger, priorizándolos de manera tal que se cuantifique el impacto en el caso de suspensión del servicio de las aplicaciones, definiendo medidas y procedimientos de seguridad, asimismo en esta etapa se debe hacer un análisis de los posibles riesgos que puede tener la compañía, en donde se identifiquen las amenazas y las vulnerabilidades tanto físicas como lógicas, proporcionándoles un valor de impacto y la probabilidad de que se materialicen.

En esta fase solo se realizará un análisis de riesgos, evaluando las vulnerabilidades presentes en el sistema. Acotando el ámbito de trabajo al servidor web, realizando un análisis del Sistema

Operativo, los software de los servicios instalados y las aplicaciones web publicadas en el mismo.

Fase 2: Políticas De Seguridad

En esta fase se pretende implantar una serie de métodos, controles y directrices para el correcto uso de los recursos informáticos en general, con el objetivo de respaldar el plan de seguridad. Al realizar esta labor, la organización debe ser consiente del nivel de responsabilidad que debe asumir, puesto que las políticas de seguridad se elaboran a la medida de cada organización y de acuerdo con su infraestructura tecnológica de información, así como de su cometido. Desafortunadamente ésta no es una tarea sencilla ni rápida, ya que la elaboración de las políticas debe ser de tal manera que éstas sean breves y de fácil entendimiento, así como también de una implementación sencilla. Además ha de tenerse en cuenta que las políticas deben apoyarse en estándares internacionales de seguridad, sin contradecir las diferentes regulaciones gubernamentales particulares para cada país. Una vez creadas las políticas, la implementación de las mismas se hace a través de procedimientos detallados, los cuales describirán los pasos a seguir, con el fin de cumplir el objetivo de la política de seguridad.

Sin embargo el proceso de documentación de procedimientos es extenso, ya que se requieren una serie de actividades con el fin de lograr el cumplimiento del estándar. La implantación de estas políticas, hará que el valor de la organización aumente así como su credibilidad y reputación.

Esta fase no se contemplará en el desarrollo de este proyecto, ya que excede el alcance del mismo.

Fase 3: Aseguramiento Físico

La seguridad física es una de las vías fundamentales para minimizar los riesgos al interior de la empresa. La fase de aseguramiento físico se enfoca en:

- Ubicación física y disposición del centro de cómputo: Este es uno de los puntos de mayor cuidado y atención, puesto que aquí se encuentra la infraestructura central de la información y de comunicaciones de las cuales depende la compañía para sus labores diarias;
- Control de acceso físico: Esta etapa refuerza la anterior, debido a que se indican los parámetros para controlar el acceso a las personas autorizadas a la organización;
- Seguridad administrativa: Busca la capacitación de los empleados de la organización en cuanto al manejo y aseguramiento de los recursos en general.
- Realización de Planes de Contingencia: Se ocupa de los procedimientos de recuperación del negocio, en caso de algún incidente de seguridad.

De esta fase solo se considerará los planes de contingencia reducido, proponiendo una alternativa ante un incidente de seguridad para la continuidad de los servicios web.

Fase 4: Arquitecturas De Red

Las arquitecturas de red permiten diferenciar las redes empresariales de las redes públicas, con el fin de mantener la información confidencial fuera del alcance de terceros. Dentro de los diferentes tipos de arquitecturas propuestas, existen las llamadas zonas desmilitarizadas o DMZ (demilitarized zone), las cuales protegen la red privada de la red pública, separando el tráfico de red interna en dos grupos, con el fin de asegurar el control de acceso a los servidores públicos desde la red pública y privada. Las arquitecturas propuestas son:

- Arquitecturas sin DMZ
    - Red básica con un solo Firewall.
    - Red básica con un solo Firewall y un host bastión
- Arquitecturas con DMZ
    - Firewall básico con una DMZ
    - Firewall dual con una DMZ
- Arquitectura de Red con IDS
- Arquitectura de Red Inalámbrica (WLAN)

Una vez establecida la arquitectura, teniendo en cuenta sus ventajas y desventajas, se procede con la implementación física de la misma, configurando y actualizando cada uno de los dispositivos que hacen parte de la arquitectura elegida, de tal manera que en su mayoría desaparezca las vulnerabilidades presentes en éstas.

La arquitectura de red del proyecto no puede ser alterada, solo puede ser segurizada, por lo cual se implementará una arquitectura sin DMZ configurando un solo firewall de host para controlar accesos al equipo.

Fase 5: Aseguramiento Y Configuración de Sistemas Operativos y Servicios

El sistema operativo controla el acceso y uso de los recursos de una máquina, siendo uno de los elementos más apetecibles para intentar explotar cualquier vulnerabilidad. Los criterios a tener en cuenta son:

- Identificación y autentificación de los usuarios.
- Control de acceso a los recursos del sistema.
- Monitoreo de las acciones realizadas por los usuarios.
- Auditoría de los eventos de posible riesgo.
- Garantía de integridad de los datos almacenados.
- Garantía de la disponibilidad de los recursos.

El aseguramiento de los servicios que presta una organización es una tarea que debe ser realizada con sumo cuidado configurándolos de manera correcta y llevando un seguimiento periódico de los archivos de registro que son creados en cada uno de éstos.

Se evaluarán diferentes herramientas que brindarán información sobre la seguridad del Sistema Operativo y los servicios del servidor como los escaneadores de puertos, escáner web y de vulnerabilidades y sniffers de red que brindan soporte para la comprobación de las configuraciones previamente establecidas. Se considerarán las vulnerabilidades encontradas en la primera fase y se realizarán las configuraciones necesarias en el sistema Operativo y en los servicios activos para segurizarlos.

Fase 6: Auditoría De Sistemas

Una vez implementada la metodología y aseguradas todas las áreas que se tuvieron en cuenta en el plan de seguridad, se procede con la auditoría de sistemas, con el fin de verificar del éxito de la implementación y el buen desempeño de los sistemas de información, ya que proporciona los controles necesarios para que los sistemas sean confiables y contengan un buen nivel de seguridad. En la auditoría de sistemas se determina si los sistemas de información salvaguardan los activos, mantiene la integridad de los datos y utiliza eficientemente los recursos.

Dentro de las principales áreas que hacen parte de la auditoría de sistemas, se encuentran:

- Auditoría Física: La auditoría es el medio que va a proporcionar la evidencia de la seguridad física en el ámbito en el que se va a desarrollar la labor profesional, por lo tanto se debe asumir que ésta no se limita a comprobar la existencia de los medios físicos, sino también su funcionalidad, racionalidad y seguridad, así como también el correcto uso de las políticas y procedimientos de seguridad.
- Auditoría de Bases de Datos: La importancia de la auditoría del entorno de bases de datos radica en que es el punto de partida para poder realizar la auditoría de las aplicaciones que utilizan esta tecnología.
- Auditoría de Redes: En este punto ha de auditarse hasta qué punto las instalaciones físicas ofrecen garantías suficientes para la seguridad de los datos. De la misma manera es necesario monitorear la red, revisar los errores o situaciones anómalas que se producen y además tener establecidos los procedimientos para detectar y aislar equipos posiblemente infectados o comprometidos.
- Auditoría de Aplicaciones: El objetivo de este punto consiste en ayudar a planificar, preparar y llevar a cabo auditorías de aplicaciones en funcionamiento, en cuanto al grado de cumplimiento de los objetivos para los que éstas fueron creadas.

- Auditoría de la Seguridad: En este tipo de auditoría se debe evaluar si los modelos de seguridad están en concordancia con las nuevas arquitecturas, las distintas plataformas y las posibilidades de las comunicaciones.

Si bien esta fase no se contempla en el alcance del trabajo, se llevará a cabo de forma parcial, considerando solo la auditoría de bases de datos y auditoría de seguridad, a través de diferentes pruebas para comprobar la correcta segurización del sistema. Sin embargo restaría aplicar test de penetración, tratando de explotar las vulnerabilidades y fallos de seguridad encontrados y así completar está etapa.

Estas fases pueden ser agrupadas en las siguientes etapas.

1. ***Entender la Infraestructura***: consiste en identificar cada uno de los dispositivos de hardware o software residentes en la infraestructura que soportan los procesos del negocio. Esta selección debe iniciarse con los servicios prestados, continuar luego con los procesos asociados a estos servicios y de allí, determinar los activos o dispositivos que soportan estos procesos.
2. ***Análisis de vulnerabilidades***: En esta fase se realizará una clasificación de activos o dispositivos con base en la confidencialidad de la información que guardan y la importancia del activo para la continuidad del proceso en estudio. Por medio del uso de herramientas para la detección de vulnerabilidades
3. ***Segurización***: Una vez determinado los activos y las vulnerabilidades presentes se tomarán las medidas preventivas, con el fin de prevenir efectos adversos sobre la prestación de los servicios; para ello se realizarán pruebas donde se aplicarán las configuraciones necesarias sobre el sistema operativo y los servicios para agregar la capa de seguridad faltante. También se aplicarán las medidas de seguridad física necesarias para evitar fallas de seguridad del equipo. Siendo necesario:
    - Definir un horario adecuado para los cambios de las configuraciones en el equipo, ya sea en horas de bajo tráfico o mejor aún en horarios de no prestación de servicios, si esto fuera posible.
    - Realizar un análisis de riesgo sobre la no disponibilidad de activos críticos, si es posible con una estimación de probabilidad y estimación del impacto.
    - Llevar a cabo medidas de contingencia:
        - Definir estrategías de contingencia para activos críticos.
        - Indicar responsables y roles de las actividades durante la contingencia.
        - Realizar respaldos de la información de los activos involucrados.
        - Guardar configuraciones de equipos involucrados.
    - Realizar monitoreo de los servicios durante las pruebas: ya sea tiempos de respuesta excesivos y eventos o incidentes de seguridad.

- Se debe informar a los responsables del equipo y autoridades pertinentes sobre la realización de las pruebas.

Con la ayuda de las herramientas de Análisis de vulnerabilidades y de explotación, se evalúa la criticidad de cada una de las vulnerabilidades encontradas y sugiere cuales deben ser solucionadas en el corto, mediano o largo plazo, para lo cual suele generarse un plan de remediación de vulnerabilidades indicando tiempo a implantar el control respectivo a la vulnerabilidad así como el costo del control, capacidad, administración y facilidad de implementación.

4. ***Pruebas de Explotación de Vulnerabilidades***: Una vez clasificadas las vulnerabilidades más críticas, se debe realizar una prueba sobre ellas con el fin de realizar su explotación. Este proceso lo llamaremos pentesting o pruebas de penetración. En la medida en que las herramientas de software utilizadas sean más inteligente y/o estructuradas el proceso será más corto y no requerirá un perfil tan sofisticado. El proceso de explotación debe incluir la posibilidad de escalar privilegios (tomar control del dispositivo como administrador) con el fin de tomar control total del sistema y de esta manera seguir de manera estricta la forma en que se llevan a cabo los ataques en la vida real.

2. Marco Teórico

2.1. Definiciones

- ***Amenaza***: posibilidad de violación de la seguridad, que existe cuando se da una circunstancia, capacidad, acción o evento que pudiera romper la seguridad y causar perjuicio (4). Una amenaza es un potencial ataque que, por explotación de la vulnerabilidad puede apropiarse de recursos de valor de la aplicación, como puede ser bases de datos, archivos de sistema, etc.
- ***Vulnerabilidad***: son defectos presentes en programas de software que los ciberdelincuentes utilizan para atacar ordenadores. Una vulnerabilidad es un flujo en el diseño, implementación, operación y administración de un sistema que puede ser explotado para violar las políticas de seguridad del sistema.
- ***Ataque***: cualquier acción que comprometa la seguridad de la información de una organización derivado de una amenaza inteligente; un acto inteligente y deliberado para eludir los servicios de seguridad y violar la política de seguridad de un sistema (4).
- ***Mecanismo de seguridad:*** mecanismo diseñado para detectar un ataque a la seguridad, prevenirlo o restablecerse de él.
- ***Exploit:*** Código que aprovechan vulnerabilidades para acceder a equipos e infectarlos (5).

2.2. Factores de Riesgo

Los atacantes pueden potencialmente usar rutas diferentes a través de la aplicación para hacer daño al negocio u organización. Cada una de estas rutas representa un riesgo que puede, o no, ser lo suficientemente grave como para justificar la atención (6). A veces, estas rutas son triviales de encontrar y explotar, otras no tanto.

Para determinar el riesgo en la organización, se puede evaluar la probabilidad asociada a cada agente de amenaza, vector de ataque, y la debilidad en la seguridad, y combinarla con una estimación del impacto técnico y de negocios para la organización. En conjunto, estos factores determinan el riesgo global, como se puede observar en la (Figura 4).

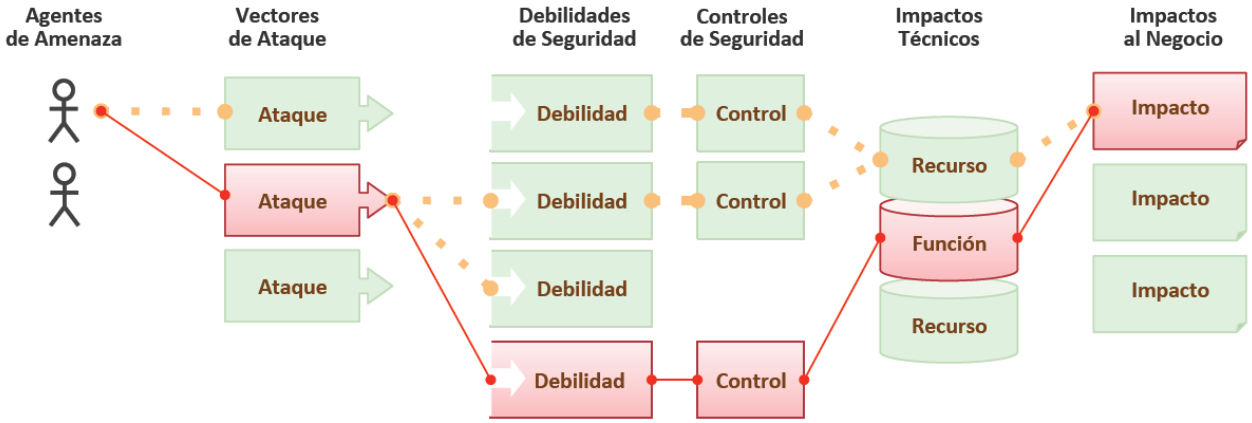


Figura 4. Factores que determinan el riesgo

2.3. Tipos de ataques

Se pueden clasificar los ataques a la seguridad, según la recomendación X.800 y RFC2828, entre (4):

1. **Ataque pasivo:** intenta conocer o hacer uso de información del sistema, pero no afecta a los recursos del mismo. Incluyen las escuchas secretas o sniffing del tráfico de la red entre el navegador web del usuario final y el servidor web, obteniendo acceso a la información del sitio web que se supone está restringida. Entre ellos están la obtención de contenido de mensajes y el análisis de tráfico.
2. **Ataque activos**: intenta alterar los recursos del sistema o afectar su funcionamiento. Incluye hacerse pasar por otra persona (tergiversación), alterar mensajes en tránsito y alterar información del sitio web. Implican alguna modificación del flujo de datos o la creación de un flujo falso y se pueden dividir en cuatro categorías: suplantación de identidad, repetición, modificación de mensajes e interrupción del servicio.

2.4. Seguridad WEB

La WWW (World Wide Web) es una aplicación cliente/servidor ejecutándose en Internet y en intranet TCP/IP. La web es vulnerable a ataques en los servidores web sobre Internet, donde un atacante puede ganar acceso a los datos y al sistema.

2.4.1. Amenazas de la Seguridad Web

Las amenazas pueden llevar a diversas consecuencias, las cuales pueden afectar a la empresa que brinda el servicio web, ya sea afectando su imagen empresarial, creando inoperatividad que lleva a pérdidas económicas, permitiendo el acceso a los equipos e información, generando perjuicios a clientes (usuarios) del servicio web al permitir obtener información personal, acarrear a un mal funcionamiento del sitio web.

En la siguiente tabla se refleja la relación entre las consecuencias y medidas para mitigar estas amenazas de los distintos servicios de seguridad (4).

Amenazas a:	Consecuencias	Medidas
Integridad	• Modificación de los datos del usuario. • Modificación de información almacenada en la memoria del equipo. • Modificación del tráfico de mensajes en tránsito. • Pérdida de información. • Comprometer un equipo dejándolo vulnerable a otras amenazas.	• Configuración adecuada de SO • Mecanismos criptográficos • Suma de comprobación de error (checksums)
Confidencialidad	• Escuchas secretas en la red. • Robo de información del servidor y de configuración de la red. • Robo de datos del cliente. • Pérdida de información. • Pérdida de privacidad del cliente.	• Mecanismos de encriptación en la web • Uso de proxies
	• Denegación de los servicios Web. • Inundación de la web con mensajes fraudulentos. • Llenado del espacio de disco o memoria.	• Configuración de seguridad en el S.O. y en los servicios activos

Disponibilidad	• Aislar el equipo a través de ataques DNS • Evitar el normal funcionamiento del equipo y que los usuarios operen correctamente, generando molestias	• Firewall • Antivirus
Autenticación	• Suplantación ilegitima del usuario. • Falsificación de datos. • Falsificación del usuario. • Creencia que la información falsa es válida.	• Técnicas criptográficas

Existen distintas situaciones que pueden dar lugar a problemas de seguridad, entre ellas:

1. Errores en las configuraciones de los equipos de la red y los servidores. En (Figura 5) se pueden observar una clasificación por área problemática.
2. Errores de programación de las aplicaciones web.

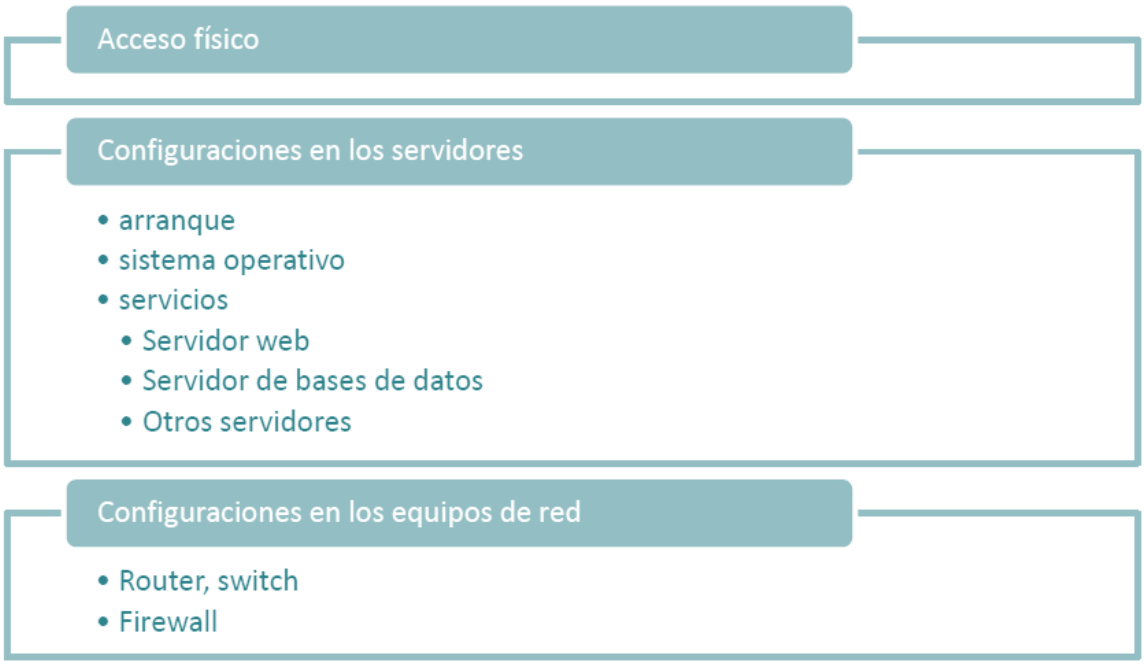


Figura 5. Vulnerabilidades en configuraciones

Cuando nos referimos a fallas en las aplicaciones web, nos centramos en errores u omisiones en la codificación de dichas aplicaciones.

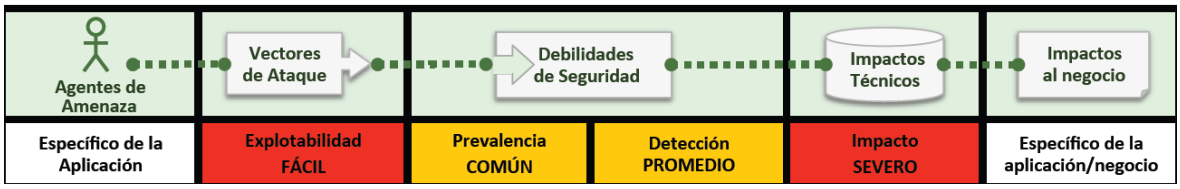
El proyecto abierto de seguridad en Aplicaciones Web "OWASP" arma una lista de los diez riesgos más críticos en Aplicaciones Web. Las vulnerabilidades del Top 10 son seleccionadas y priorizadas de acuerdo a datos de relevancia, en combinación con estimaciones consensuadas de explotabilidad, detectabilidad e impacto. La lista de TOP 10 correspondiente al 2013 es (6):

- A1- Inyección (SQL, PHP, SO, LDAP)

- A2- Pérdida de Autenticación y Gestión de Sesiones (session hijacking)
- A3-Secuencia de Comandos en Sitios Cruzados (Cross Site Scripting (XSS))
- A4- Referencia Directa Insegura a Objetos
- A5- Configuración de Seguridad Incorrecta
- A6- Exposición de Datos Sensibles
- A7- Ausencia de control de acceso a las funciones
- A8- Falsificación de peticiones en Sitios Cruzados (Cross Site Request Forgery (CSRF))
- A9- Uso de componentes con vulnerabilidades Conocidas
- A10- Redirecciones y reenvíos no validados.

A continuación se describirán brevemente:

A1- Inyección: se puede hacer una inyección de código y ejecutar comandos arbitrariamente en el servidor.



Se considera de fácil explotación ya que el atacante envía ataques con cadenas simples de texto, los cuales explotan la sintaxis del intérprete a vulnerar. Casi cualquier fuente de datos puede ser un vector de inyección, incluyendo fuentes internas.

Las fallas de inyección ocurren cuando una aplicación envía información no confiable a un intérprete. Se encuentran frecuentemente en las consultas SQL, LDAP, XPath o NoSQL, comandos del sistema operativo, intérpretes de XML, encabezados de SMTP, etc. Estas fallas son fáciles de identificar al analizar el código, pero difíciles realizando sólo pruebas. Los analizadores y los fuzzer pueden ayudar a los atacantes a encontrarlas.

El impacto puede ser severo ya que puede causar pérdidas o corrupción de datos, denegación de servicios, incluso dejar sin funcionamiento algún equipo.

Evitar un riesgo de inyección requiere mantener los datos no confiables separados de los comandos y de las consultas.

Algunas fallas de inyección:

- **Inyección PHP**: Se pueden ejecutar comandos (system, exec, etc.) del Sistema Operativo desde código php. Suele pasar cuando no existen validaciones correctas.

  Ejemplo: Este programa toma una variable por método get y crea un directorio

```
<?
```

```
if (trim($_GET['directorio'])!=NULL) {
system ("mkdir {$_GET['directorio']}");
echo "directorio creado {$_GET['directorio']}";
}
else
echo "directorio vacio";
?>
```

Si el directorio donde se encuentra el script inicial tiene permisos 777 o pertenece al usuario que ejecuta el servicio de apache, podrá crear los directorios. La url del navegador sería: http://web/script.php?directorio=directorionuevo (donde "directorionuevo" es el nombre del directorio que se creará) se ejecuta un comando de sistema operativo con la función system para crear el directorio.

Como en linux/unix, se puede ejecutar más de un comando en una misma línea, se puede colocar en la url varios comandos, por ejemplo: http://web/script.php?directorio=directorionuevo%20pwd con esto se creará el directorio "directorionuevo", y además, dará la ruta desde donde se está ejecutando el script.

Lo más preocupante es que se puede descargar malware al servidor, para hacerlo parte de una botnet para atacar a otros equipos.

Se pueden implementar distintas soluciones, desde configuraciones especificas en el archivo php.ini, hasta mejoras en la programación de las aplicaciones PHP. En el capítulo 3 sección 3.3.6 se proponen configuración de segurización de PHP y en la sección 3.4 se sugieren recomendación para los desarrolladores de aplicaciones.

- **Inyección de Script y HTML**: cuando el atacante encuentra un sitio que puede ser comprometido modifica el mismo inyectando script dañinos o HTML que redirecciona al visitante (victima) a otro sitio donde se encuentra el malware que permite explotar la vulnerabilidad, esto es conocido como **ataque watering Hole**, este malware permite analizar las aplicaciones instaladas (Java, Adobe Reader, Flash, etc.) de cada víctima que accede el sitio “modificado” y luego explotar las vulnerabilidades de estas aplicaciones en esa víctima. Este tipo de ataque permite realizar espionaje, recoger información, realizar ataques tipo APT (Amenazas Persistentes Avanzadas). Un ejemplo es el ataque Ghost RAT. Otro ejemplo es el **Ataque Server Side Include** (SSI) que permite la explotación de una aplicación web inyectando scripts en las páginas HTML o ejecutando código remotamente. Incrementa la carga en el servidor, si existe un entorno compartido y las aplicaciones web tienen un alto nivel de tráfico se recomienda deshabilitar SSI.
- **Inyección SQL**: consiste en insertar o inyectar código SQL malicioso dentro de código SQL, para alterar el normal funcionamiento y hacer que se ejecute el código “invasor” dentro del sistema, permitiendo acceder a datos de la Base de datos,

modificar el comportamiento de la aplicación, ejecutar sentencias, cambiar permisos, utilizar procedimientos almacenados, etc. El problema radica al filtrar erróneamente las variables utilizadas en parte de la página con código SQL. Mitigar este tipo de ataques, consiste en una buena programación de las consultas SQL y la aplicación, por ej.se puede filtrar las entradas de los usuarios reemplazando la aparición de ‘ por ” (dos comillas simples) e incluso evitando que los usuarios puedan pasar caracteres como \ / “ ‘ o cualquier otro que se nos ocurra que puede causar problemas.

A2 - Pérdida de Autenticación y Gestión de sesiones

El atacante utiliza filtraciones o vulnerabilidades en las funciones de autenticación o gestión de sesiones (por ej. Cuentas expuestas, contraseñas, identificadores de sesión) para suplantar a otros usuarios. Se conocen como ataques de secuestro (Hijacking) donde el atacante toma el control de la comunicación entre dos entidades enmascarándose como uno de los participantes, el atacante toma el control mientras se produce la comunicación. El atacante puede solo ganar acceso a los mensajes o modificar los mensajes antes de retransmitirlos (7)**.**

Los desarrolladores a menudo crean su propio esquema de autenticación o gestión de sesiones, llevándolos a tener vulnerabilidades en el cierre de sesión, gestión de contraseñas, tiempo de desconexión, función de recordar contraseña, pregunta secreta, actualización de cuenta, etc. Si bien encontrar estas vulnerabilidades puede ser difícil ya que cada implementación es única, una vez encontradas el impacto puede ser severo ya que puede afectar a una o todas las cuentas de usuarios, suplantando su identidad y realizando las acciones que estos tengan permitido.

Se puede ser vulnerable si:

- Las credenciales de los usuarios no están protegidas cuando se almacenan utilizando un hash.
- Se pueden adivinar o sobrescribir las credenciales a través de funciones débiles de gestión de la sesión.
- La ID de sesión son expuestos en las URL
- Los ID de sesión no expiran o no son invalidados al cierre de sesión.
- Las contraseñas, ID de sesión, credenciales son transmitidos por canales no cifrados.

Algunos ataques relacionados a este riesgo:

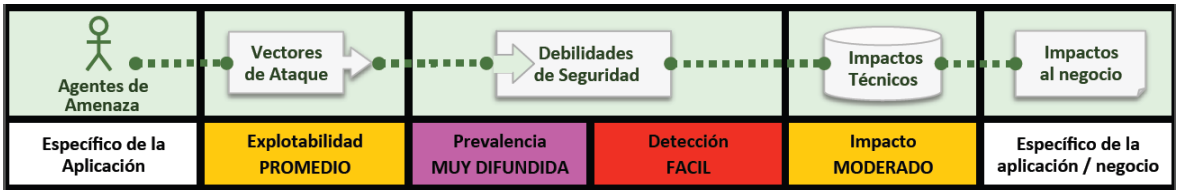
- **Session Hijacking** (secuestro o robo de sesión) que se refiere a que un atacante consigue el identificador de sesión entre una página web y un usuario, de forma que puede hacerse pasar por este y acceder a su cuenta en esa página web, ya que la única forma que tiene la página web de reconocer a un usuario es por medio de su identificador de sesión. Las formas en las que un atacante puede robar este identificador son:
    - A través del ataque por fuerza bruta, probando todos los identificadores aleatoriamente hasta encontrar uno que esté siendo usado. Por lo tanto es importante tener identificadores de sesión de gran longitud y obtenidos aleatoriamente.
    - A través de observar el tráfico de la red del usuario (sniffing) y poder interceptar el identificador de sesión. La forma de prevenirlo es utilizando conexiones cifradas (HTTPS).
    - Si se usa un servidor compartido con otras páginas web, los archivos físicos de las sesiones se guardan en un directorio común para todas las páginas web del servidor, por lo cual si puede acceder en el servidor a su archivo de sesión, podrá acceder al del resto de los usuarios. Esta vulnerabilidad se puede combatir guardando los archivos de sesión de la página web en un directorio al que sólo pueda acceder PHP, o guardando las sesiones en base de datos en lugar de en archivos, esta solución es la más segura ya que la página web será la única que tendrá acceso a la base de datos y, por tanto, a las sesiones.

- **DNS hijacking**: se direcciona al usuario a un sitio distinto al que el solicita. Hay dos tipos de DNS hijacking, uno de ellos el atacante gana acceso a los registros DNS en un servidor y los modifica para redireccionar los requerimientos de un sitio genuino, esto crea una visión de servidor web infectado. Este tipo de ataques es difícil de prevenir porque los administradores solo tienen control a sus propios registros DNS pero no lo tienen a los registros DNS superiores. En el segundo tipo, el atacante falsifica (suplanta) la cuenta de email válido e inunda de emails a los contactos de esa cuenta. Este tipo de ataque puede prevenirse utilizando autenticación en los registros interNIC.
- **IP Spoofing (**falsificación de dirección IP**)** (8): el atacante se enmascara como un host verdadero para ocultar su identidad, falsificando un sitio web o ganando acceso a la red. El atacante obtiene una dirección IP de un host legítimo y altera las cabeceras de los paquetes con esta IP para que aparezca como origen del mensaje. Cuando IP spoofing es usado para atacar a un navegador web, el visitante que tipea una URL de un sitio válido obtiene una página de un sitio web fraudulento creado por el atacante. Si el usuario interactúa ingresando información en el sitio, el atacante puede obtener esta información sensitiva como puede ser password, número de tarjeta de crédito; incluso el atacante podría instalar algún malware para obtener el control de la computadora comprometida o

usar está para ser parte de una botnet. Se puede minimizar el daño implementando técnicas de encriptación, one-time password, y proteger la red implementando firewall para bloquear paquetes entrantes de direcciones orígenes que difieran de la dirección IP del usuario o red interna.

A3 - Secuencia de Comandos en Sitios Cruzados (XSS)

XSS es la falla de seguridad predominante en aplicaciones web. Ocurren cuando una aplicación, en una página enviada a un navegador incluye datos suministrado por un usuario sin ser validados o codificados apropiadamente. Existen tres tipos: almacenadas, reflejadas y basadas en DOM.
Si la página web es vulnerable a XSS el atacante puede insertar un código javascript, aprovechando la explotación de vulnerabilidades del sistema de validación de HTML incrustado, el cual puede estar presente de forma directa (foros, mensajes de error) o indirecta (redirecciones, framesets).
El problema radica en la validación incorrecta del código. Básicamente se trata de engañar al usuario para que pulse una URL especialmente manipulada que ejecutará código Javascript que será interpretado en el navegador del usuario. Si es XSS basada en DOM, el código que inyecta el atacante llamará a diversos objetos DOM para ejecutar acciones en el equipo de la víctima; además, para dar un aspecto más amigable a la URL y para evitar ciertos filtros, ésta suele ser transformada mediante acortadores URL o codificada en hexadecimal.



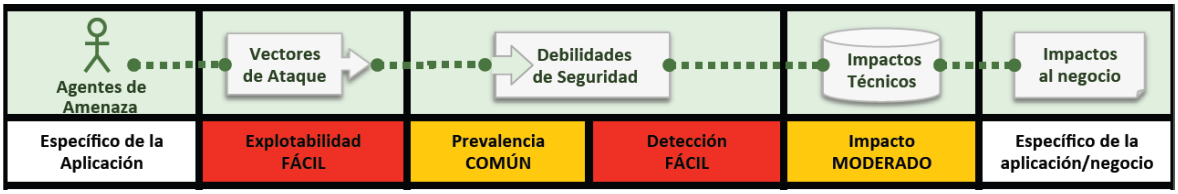
El atacante envía cadenas de texto que son secuencia de comandos de ataque que explotan el intérprete del navegador. Casi cualquier fuente de datos puede ser un vector de ataque, incluyendo fuentes internas tales como datos de la base de Datos (6). La mayoría puede detectarse de forma relativamente fácil a través de pruebas o por medio del análisis de código.

El impacto es moderado ya el atacante puede ejecutar una secuencia de comandos en el navegador de la víctima para secuestrar las sesiones de usuarios, alterar la apariencia del sitio web, insertar código hostil, redirigir usuarios, secuestrar el navegador de la víctima utilizando malware, etc.

Para evitar esta vulnerabilidad, se debe codificar adecuadamente las entradas de datos ingresadas por los usuarios o verificar en el momento de ingreso que los datos sean seguros antes de ser incluidos en la página de salida.

A4 - Referencia directa insegura a Objetos

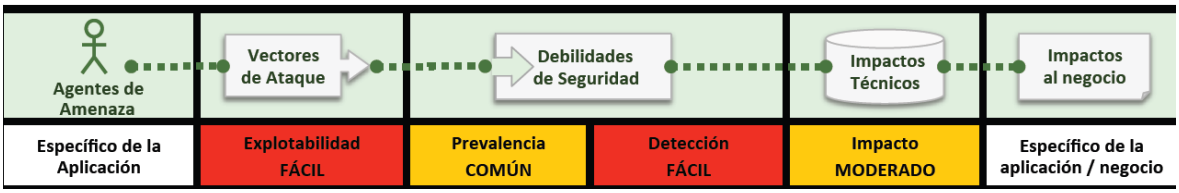
Las aplicaciones no siempre verifican que el usuario tiene autorización sobre el objetivo. Esto resulta en una vulnerabilidad de referencia de objetos inseguros.



La mejor forma de comprobar si una aplicación es vulnerable es verificar que todas las referencias a objetos tienen las protecciones apropiadas, por ejemplo: verificar si el usuario está autorizado a acceder al recurso que solicita, pueden ser referencias directas o indirectas. Normalmente las herramientas automatizadas no detectan este tipo de vulnerabilidades porque no pueden reconocer cuales necesitan protección o cuales son seguras o inseguras, por lo que se requiere un análisis de código de la aplicación.

A5 - Configuración de seguridad incorrecta

Las configuraciones de seguridad incorrecta pueden ocurrir a cualquier nivel de aplicación, incluyendo la plataforma, el servidor web, servidor de aplicación, base de datos, framework de desarrollo y código personalizado.
Un atacante puede acceder a cuentas por defecto, página sin uso, utilizar defectos de software sin corregir, archivos y directorios sin protección, etc. para tener acceso no autorizado o conocimiento del sistema.
Los desarrolladores y los administradores de sistemas deben trabajar juntos para asegurar que las distintas capas estén configuradas apropiadamente.



Las herramientas automatizadas son útiles para detectar parches omitidos, fallas de configuración, uso de cuentas por defecto, servicios innecesarios, etc.
El impacto es moderado ya que permiten al atacante acceso no autorizado a algunas funcionalidades del sistema o datos. Ocasionalmente pueden provocar que el sistema se vea comprometido totalmente.

A6 – Exposición a datos sensibles

Los atacantes generalmente roban datos en tránsito, como contraseñas, datos de configuraciones, etc., a través de ataques como "man in the middle".

La debilidad más común es no cifrar las comunicaciones de datos sensibles. También sucede que se utilizan claves débiles en el cifrado, algoritmos de cifrado débiles o funciones de hash de contraseñas débiles. Las debilidades del navegador son fáciles de detectar pero difíciles de explotar.

El impacto es severo ya que se comprometen los datos que deberían estar protegidos. Por lo cual se deberían saber cuáles son los datos sensibles y aplicar protección extra.

A7 – Control de acceso inexistente a nivel de funcionalidades.

Las aplicaciones no siempre protegen las funcionalidades adecuadamente, a veces a través de configuraciones incorrectas, otras por desarrollo incorrecto de las aplicaciones, generando vulnerabilidades que permiten el acceso no autorizado de los atacantes a funciones del sistema; siendo las funciones administrativas un objetivo clave de este tipo de ataques.
Puede suceder cuando un usuario cambia la URL ó un parámetro a una función con privilegios. Por ejemplo una página proporciona un botón de acción para especificar una función y diferentes acciones requieren diferentes roles, si no se verifica el rol dada una acción esto es una vulnerabilidad.

Ejemplos de esta vulnerabilidad dada por errores de codificación se mencionan a continuación:

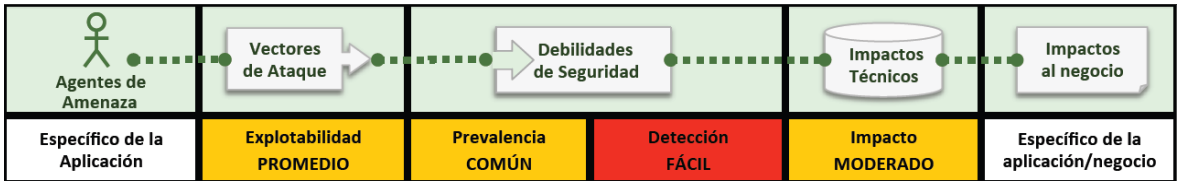
- **Uso de register_globals**: configuración de PHP que controla la disponibilidad de las variables superglobales en un script PHP, por ej. información en post de formularios, datos desde la url (get), o información de las cookies. Cuando register_globals está definida en "On" dentro del php.ini, esto permite a un usuario cualquiera poder inicializar una variable remotamente. Muchas veces no es inicializado el parámetro que se utiliza para incluir archivos indeseados de un atacante, y este podría terminar en una ejecución arbitraria de archivos localizados local y remotamente.

- **Uso del protocolo de llamada a procedimiento remoto que utiliza XML** para codificar las llamadas (XML-RPC), y HTTP como mecanismo de transporte. Un error común en los distintas implementaciones de XML-RPC en PHP es pasar entradas de datos del usuario sin filtrar por la función eval() en el servidor XML-RPC. Esto permite al atacante ejecutar código, en el sistema vulnerable. Cualquier usuario con habilidad de subir XML manualmente al servidor puede insertar código PHP que puede ser ejecutado por la aplicación Web vulnerable.

Puede ser complejo determinar si existen páginas o funcionalidades atacables. Las herramientas automatizadas no suelen encontrar este tipo de vulnerabilidad.

A8- Falsificación de peticiones en Sitios Cruzados (Cross Site Request Forgery (CSRF))

El atacante crea peticiones HTTP falsificadas y engaña a la víctima mediante el envío de etiquetas de imágenes, XSS u otras técnicas. Si el usuario esta autenticado, el ataque tiene éxito. Por ejemplo: el atacante coloca en un iframe de algún sitio controlado por el atacante un enlace con la petición de transferencia de dinero desde la cuenta de la víctima hacia la cuenta del atacante y si la victima visita estos sitios el ataque tiene éxito.
CSRF aprovecha la posibilidad dada por la mayoría de las aplicaciones web que permiten a los atacantes predecir todos los detalles de acción. Los atacantes pueden crear páginas web maliciosas que generen peticiones falsificadas que son indistinguibles de las legítimas.
Los enlaces o formularios que invoquen funciones que permitan cambios de estado son los objetivos más importantes de la vulnerabilidad CSRF.

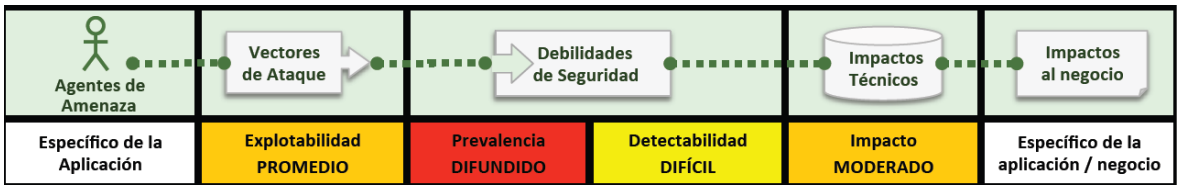


La detección de esta vulnerabilidad es fácil a través de pruebas de penetración y de análisis de código.

El impacto radica en que el atacante puede cambiar cualquier dato que la víctima este autorizado a cambiar o acceder a alguna funcionalidad que esté autorizado.
Para prevenir ataques que afecten a esta vulnerabilidad se precisa la existencia de un token no predecible en cada solicitud HTTP. También incluir herramientas de verificación de usuario válido, como catchas, son útiles.

A9 – Uso de componentes con vulnerabilidades conocidas.

El atacante identifica un componente débil a través de escaneos automáticos o análisis manuales. Se adecua un exploit para explotar la vulnerabilidad y realizar el ataque.
Muchas aplicaciones tienen este tipo de problema ya que la mayoría de los equipos de desarrollo no actualizan los componentes de software o desconocen los mismos.

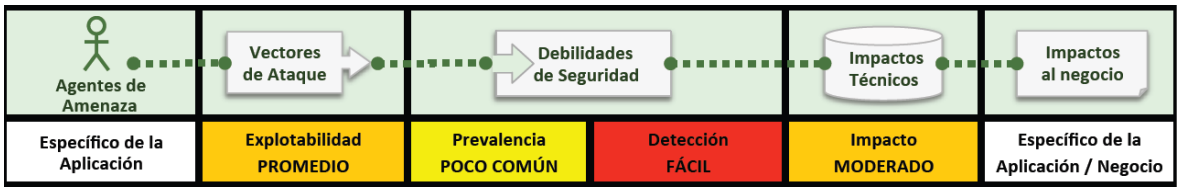


El impacto puede comprometer completamente el equipo y apoderarse de los datos, ya que las debilidades van desde inyección, control de acceso, XSS, entre otras.
Se pueden conocer si las componentes o biblioteca de software que se estén usando son vulnerables. Existen sitios como CVE (Common Vulnerabilities and Exposures) (9) y NVD (National Vulnerability Database) (10) donde pueden observarse las vulnerabilidades reportadas.

A10 – Redirecciones y Reenvíos no válidos

Un atacante crea enlaces a direcciones no validadas y engaña a las víctimas para que hagan clic en dichos enlaces. Las víctimas son propensas a hacer clic sobre ellos ya que el enlace lleva a una aplicación de confianza. El atacante tiene como objetivo los destinos inseguros para evadir los controles de seguridad.

Las aplicaciones redirigen a los usuarios a otras páginas, a veces, a través de un parámetro no validado, permitiendo a los atacantes elegir dicha página. Estas redirecciones pueden intentar instalar código malicioso o engañar a las víctimas para que revelen contraseñas u otra información sensible. El uso de reenvíos inseguros puede permitir evadir el control de acceso. Detectar estas redirecciones sin validar puede resultar fácil.



Un ataque relacionado a esta vulnerabilidad es el ataque de modificación de la interfaz del usuario (UI) **o Secuestro de clic ó Clickjacking**, se produce cuando un atacante usa múltiples capas opacas o transparentes para engañar a un usuario que hace clic en algún link o botón redirigiéndolo a otro sitio. Es una técnica maliciosa para engañar a usuarios de Internet con el fin de que revelen información confidencial o tomar control de su computadora cuando hacen clic en páginas web aparentemente inocentes. Puede tomar la forma de código embebido o script que se ejecuta sin el conocimiento del usuario; por ejemplo, aparentando ser un botón para realizar otra función. Las maneras de prevenir este tipo de ataques:

1. Activando la propiedad X-Frame-Options en las respuestas de las cabeceras HTTP indicando al browser que no permita redirección a otros dominios.
2. Empleando código defensivo dentro de la IU para asegurarse que los frames actuales corresponde a la ventana de máximo nivel.

2.4.2. Ataques a la seguridad de servidores web

Un tipo de ataque muy común en ámbitos de servidores web es la ***denegación de servicio (DoS)***, donde un atacante se encarga de realizar una serie muy numerosa de peticiones a un servidor con el objetivo de que se sobrecargue y se colapse, denegando el servicio a otros posibles usuarios. Este tipo de ataque provoca consumo de recursos computacionales, tales como ancho de banda, espacio de disco, tiempo de procesador, alteración de información de configuración como las rutas de encaminamiento, alteración de información de estado como interrupción de sesiones TCP, interrupción de componentes físicos de red; afectando la disponibilidad del servicio, la imagen empresarial y los servicios web brindados.

Existe una variante de este tipo ataque denominada ***denegación de servicio distribuida (DDoS)***, donde existe una red distribuida de atacantes conformada por un conjunto de usuarios sincronizados, que a su vez son víctimas (usuarios o servidores infectados con troyanos o virus, sin saberlo) que silenciosamente forman parte de botnets. Se considera más peligroso, ya que no se trata de una sola entidad la que intenta atacar, sino de una «flota» de atacantes muy numerosa, en la que un sólo servidor normalmente se desborda al recibir tantas peticiones (11).

El ataque se puede dar de muchas formas. Pero todas tienen en común que utilizan la familia de protocolos TCP/IP para conseguir su propósito. Ejemplos de este tipo de ataque son:

- **Slowloris:** escrito por Robert “RSnake" Hansen. Crea muchas conexiones hasta el servidor web y las mantiene abiertas el mayor tiempo posible, enviando periódicamente cabeceras HTTP, esto provoca que el servidor no pueda aceptar nuevas conexiones, dejando el servidor inactivo para operar con otros usuarios. Se puede mitigar este tipo de ataques utilizando los módulos de Apache: mod_limitipconn, mod_qos, mod_evasive, mod_security, mod_noloris, y mod_antiloris y configurando apropiadamente el firewall, proxies y demás dispositivos de red.
- **Inundación SYN (SYN floods)**: La inundación SYN envía un flujo de paquetes TCP/SYN (varias peticiones con Flags SYN en la cabecera), muchas veces con la dirección de origen falsificada. Cada uno de los paquetes recibidos es tratado por el destino como una petición de conexión, causando que el servidor intente establecer una conexión al responder con un paquete TCP/SYN-ACK y esperando el paquete de respuesta TCP/ACK (Parte del proceso de establecimiento de conexión TCP de 3 vías). Sin embargo, debido a que la dirección de origen es falsa o la dirección IP real no ha solicitado la conexión, nunca llega la respuesta. Estos intentos de conexión consumen recursos en el servidor y acaparan el número de conexiones que se pueden establecer, reduciendo la disponibilidad del servidor para responder peticiones legítimas de conexión. SYN cookies provee un mecanismo de protección contra Inundación SYN, eliminando la reserva de recursos en el host destino, para una conexión en momento de su gestión inicial.
- **Inundación ICMP:** pretende agotar el ancho de banda de la víctima. Consiste en enviar de forma continuada un número elevado de paquetes ICMP Echo request de tamaño considerable a la víctima, de forma que esta ha de responder con paquetes ICMP Echo reply lo que supone una sobrecarga tanto en la red como en el sistema de la víctima.
- **Inundación UDP:** consiste en generar grandes cantidades de paquetes UDP contra la víctima elegida. Debido a la naturaleza sin conexión del protocolo UDP, este tipo de ataques suele venir acompañado de IP spoofing.

2.4.3. Mecanismos de Mitigación de ataques

Garzón (3) considera que se debe abarcar varios frentes de seguridad para reducir al mínimo la efectividad de los ataques que pueden aprovechar las vulnerabilidades. Estos frentes son:

- Seguridad Lógica: aplicando barreras y elaborando procedimientos que ayuden a proteger la información;
- Seguridad Física: usando defensas físicas y procedimientos de control;
- Políticas y Estándares: a través de documentos de la organización para administrar y proteger la información;
- Auditoría de Sistemas: con la revisión y evaluación de los controles de sistemas y procedimientos de informática.

Para poder definir un mecanismo de mitigación de ataques, es necesario conocer los modelos de seguridad y así definir un mecanismo guiado por el modelo de seguridad elegido. Existen tres modelos de seguridad (12):

1. ***Modelo de Seguridad Negativa***. Un modelo de seguridad negativo monitorea los requerimientos de anomalías, comportamiento inusual y ataques a aplicaciones web comunes. Mantiene un puntaje de las anomalías por cada requerimiento, por dirección IP, por sesión de aplicación, por cuenta de usuario, de tal forma que los requerimientos con un alto puntaje anómalo son logueados o rechazados.
2. ***Modelo de Seguridad Positivo***. En este modelo sólo se aceptan los requerimientos que son conocidos como válidos, el resto son rechazados, por lo que se requiere conocer sobre la aplicación web. Es útil en aplicaciones ampliamente usadas pero poco modificadas.
3. ***Vulnerabilidades o debilidades conocidas***. En este modelo la premisa consiste en actualizar el sistema corrigiendo las vulnerabilidades. Es útil tener alguna herramienta de patching conocida como "virtual patching" para reducir las oportunidades.

Existen distintos mecanismos para afrontar los ataques y brindar mayor seguridad a los servidores web, entre ellos:

- **Utilizar el módulo mod_evasive:** es un módulo para Apache que se encarga de gestionar una tabla hash con las IPs con accesos más frecuentes al servidor. Así, si detecta un número desmesuradamente alto de peticiones desde una misma IP las anulará por un tiempo. Intenta frenar ataques de DDoS, que lleven a denegar el acceso al servidor web, sin embargo no servirá de mucho si se intenta provocar una denegación a la red local, router u otro elemento relacionado con el flujo de red, ya que el módulo actúa a nivel de aplicación (11).
- **Aplicar un firewall de aplicación web (WAF – Web Application Firewall):** El WAF es un dispositivo de hardware o software que analiza el tráfico web (entre el servidor web y los usuarios) y protege las aplicaciones web de diversos ataques como SQL Injection, Cross Site Scripting, etc. WAF ha sido desarrollado para establecer una capa de

seguridad externa incremental para detectar y/o prevenir ataques. Existen distintos mecanismos de aplicar un firewall en una aplicación web, se indicarán algunos de ellos.

- El ***módulo mod_security*** de Apache permite detectar prevenir de instrucciones a aplicaciones web, permite monitorear el tráfico HTTP y analizar en tiempo real cambios en la infraestructura existente. Se puede aplicar en el modelo de seguridad de vulnerabilidades conocidas ya que se puede corregir la seguridad de la aplicación de manera externa sin tener que modificar el código de la aplicación. A través de un conjunto de reglas se puede establecer una serie de acciones (bloquear, informar en un registro, ignorar, etc...) para requerimientos entrantes basados de diferentes criterios como variables CGI, cabeceras HTTP, variables de entorno, scripts, etc. (13) (12). Es especialmente potente, por tres razones:
    - Posee un nivel de personalización muy amplio.
    - Funciona en una capa previa al servidor web, bloqueando los accesos antes de llegar a ser procesados por el servidor web.
    - Permite el uso de expresiones regulares en el conjunto de reglas.
    - Permite crear un archivo log de auditoría, almacenado en detalle los requerimientos.
- Usar ***TCP_WRAPPERS*** para asegurar el servidor contra intrusiones externas. Es un sistema de Listas de Control de Acceso (ACL) que se usa para filtrar el acceso de red a servicios de protocolos de Internet que corren en sistemas operativos, como Linux o BSD. Permite que las direcciones IP, los nombres de terminales y/o respuestas de consultas sean usadas como tokens sobre los cuales filtrar. Se controla desde los archivos /etc/hosts.allow y /etc/hosts.deny. Primero se comprueba hosts.allow, evaluando las reglas desde la primera a la última. Si encuentra una regla que permita específicamente entrar deja conectarte al servicio, caso contrario comprueba el archivo hosts.deny, en búsqueda de una regla que deniegue el acceso. En caso de no encontrar una regla permite la entrada. Por defecto tiene una política permisiva, por lo cual se podría aplicar una política restrictiva para brindar mayor seguridad, colocando en el archivo hosts.deny como última línea: ALL:0.0.0.0/0.0.0.0 lo que significa que deniega el acceso a todos los servicios desde cualquier host, de modo que cualquier servicio que no esté especificado su acceso, quedará bloqueado, conocido como lista negra.

```
En host.allow:
# permitir acceso desde la red interna de 10.0.0.*
in.telnetd: 10.0.0.0/255.255.255.0
en hosts.deny:
# denegar acceso a telnetd desde cualquier parte
in.telnetd: 0.0.0.0/0.0.0.0
```

El problema con TCP_WRAPPERS es que un atacante permite conocer que hay un puerto abierto que brinda algún servicio, aunque no consigue ingresar.

- **Iptables**: es un sistema de Firewall vinculado al Kernel de Linux. Se puede definir políticas de filtrado del tráfico que circula por la red. Permite evitar acceso mediante ssh, ftp, no poder cargar la página web, bloquear los ping, entre otras acciones.

  Tiene incorporado tres *tablas* o *listas de reglas* (14):

  - Filter: La tabla por defecto para el manejo de paquetes de red.
  - Nat: Usada para alterar paquetes que crean una nueva conexión y utilizada para la *Traducción de direcciones de red* (*NAT*).
  - Mangle: Usada por tipos específicos de alteración de paquetes.

  Cada una de estas tablas tiene un grupo de *cadenas* incorporadas que corresponden a las acciones llevadas a cabo por el filtro de la red.

  Las cadenas internas para la tabla filter son las siguientes:

  - *INPUT* — Aplica a los paquetes recibidos a través de una interfaz de red.
  - *OUTPUT* — Esta cadena sirve para paquetes enviados por medio de la misma interfaz de red que recibió los paquetes.
  - *FORWARD* — Esta cadena sirve para paquetes recibidos en una interfaz de red y enviados en otra.

  Las cadenas internas para la tabla nat son las siguientes:

  - *PREROUTING* — Altera los paquetes de red cuando estos llegan.
  - *POSTROUTING* — Esta cadena altera paquetes antes de que sean enviados por medio de una interfaz de red.
  - *POSTROUTING* — Altera los paquetes de red cuando estos son enviados.

  Cada paquete de red recibido o enviado desde un sistema Linux está sujeto al menos a una tabla. Sin embargo, un paquete puede estar sometido a múltiples reglas dentro de cada tabla antes de emerger al final de la cadena. La estructura y propósito de estas reglas puede variar, pero normalmente buscan identificar un paquete que viene de o se dirige hacia una direccción IP en particular, o un conjunto de direcciones, cuando utiliza un determinado protocolo y servicio de red.

  Si bien se tiene que definir una política por defecto para el firewall, cada cadena se le puede aplicar una política específica.

  Hay dos maneras de implementar un firewall:

1. Política por defecto ACEPTAR: en principio todo lo que entra y sale por el firewall se acepta y solo se denegará lo que se diga explícitamente.
2. Política por defecto DENEGAR: todo está denegado, y solo se permitirá pasar por el firewall aquellos que se permita explícitamente.

Cada cadena tiene una política por defecto:

1. ACCEPT: Si la regla especifica un objetivo ACCEPT para un paquete que coincida, el paquete se salta el resto de las verificaciones de la regla y se permite que continúe hacia su destino.
2. DROP: Si una regla especifica un objetivo DROP, a ese paquete se le niega el acceso al sistema y no se envía nada de vuelta al servidor que envió el paquete.
3. REJECT: Si una regla especifica el objetivo opcional REJECT, el paquete es descartado, pero se envía un paquete de error al que envió el paquete.
4. QUEUE: Si una regla especifica un objetivo QUEUE, el paquete se pasa al espacio del usuario.

El orden en el que se ponen las reglas de firewall es determinante. Independientemente de su destino, cuando un paquete cumple una regla en particular en una de las tablas, se les aplica un objetivo (target) o acción a ellos. Si ninguna de estas reglas en la cadena se aplica al paquete, entonces el paquete es tratado de acuerdo a la política del firewall por defecto. Hay que tener cuidado, porque si existen reglas muy permisivas al principio, puede que las siguientes no se apliquen y no cumplan el objetivo buscado.

- ***Port Knocking.*** El Port-knocking se basa en realizar varios intentos de conexión consecutivos a ciertos puertos para establecer una conexión (15). Por ejemplo, tener configurado el knockd (demonio encargado del port-knocking) para que al recibir los intentos de conexión en el puerto 2199 y 9123, se abra el firewall para nuestra IP en el puerto 22, es decir indicar una secuencia de accesos denegados por puertos erróneos para poder autorizar el acceso al puerto correcto.

- ***Sistema de Detección de Intrusos De Host (IDS):*** Un IDS basado en host monitorea y analiza diferentes aspectos y el estado general del sistema, entre ellos: monitoreo y análisis de logs, actividad de programas en ejecución, estado de la red, detección de rootkits, detección de procesos ocultos, monitoreo de los registros de Windows, etc. Ejemplos: OSSEC, Fail2ban, SAMHAIN, Personals firewalls, etc.

2.4.4. Técnicas para lograr una comunicación segura

En el modelo TCP/IP, utilizado en las comunicaciones entre equipos, pueden existir distintas vulnerabilidades en los protocolos asociados a cada comunicación y un atacante puede explotarlas. Cada día se descubren nuevas deficiencias, la mayoría de las cuales se hacen públicas por los organismos internacionales, tratando de documentar y si es posible solucionar y contrarestar los problemas. Considerando las capas de la pila de protocolos TCP/IP, se mencionarán como afectan las vulnerabilidades a cada capa:

- Capa de Acceso a la Red: Se presentan problemas de control de acceso y confidencialidad.
- Capa de Internet: cualquier ataque que afecte al datagrama IP. Se usan técnicas como sniffing. Suplantación de mensajes, modificación de datos, denegación de mensajes y retrasos en los mismos.
- Capa de Transporte: se pueden encontrar problemas de autenticación, integridad y confidencialidad. Intercepción de sesiones TCP, entre otras.
- Capa de Aplicación: se presentan varias deficiencias asociadas a los diferentes protocolos o aplicaciones de esta capa. Entre ellas podemos encontrar vulnerabilidades relacionadas con:
    - DNS: Se conocen también como DNS spoofing, un atacante puede modificar la información que suministra ésta base de datos, ó acceder a información que muestre la topología de la red de una organización.
    - telnet: El mayor problema es que esta aplicación transmite información del usuario y contraseña en texto claro, por lo que permite la captura de esta información a través de técnicas de sniffing.
    - FTP: envía información en texto claro.
    - HTTP: brindar información del servidor desde el cliente HTTP, esto es posible mediante la ejecución remota de código en el servidor.

Existen varios recursos para proveer seguridad, según los servicios que prestan o los mecanismos que se usen (16).

Considerando el stack de protocolos TCP/IP, en la siguiente imagen (Figura 6) pueden observarse por cada capa los distintos protocolos que adicionan la capa de seguridad a los protocolos originales.

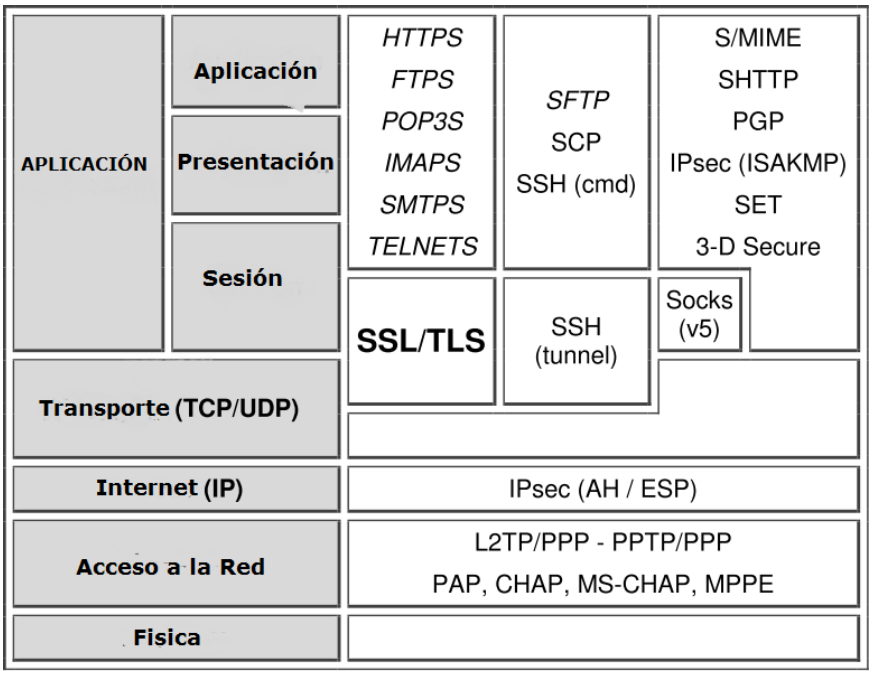


Figura 6. Ubicación relativa de las herramientas de seguridad en la pila de protocolos TCP/IP

La World Wide Web es una aplicación cliente/servidor ejecutando sobre Internet y sobre intranet TCP/IP que presenta nuevos desafíos asociados en el contexto del servidor web y de la seguridad de la red.

Existen distintos recursos para proveer de seguridad web:

- Usar seguridad IP (IPsec), en capa 3 de la pila de protocolos TCP/IP, cuya ventaja es la transparencia a usuarios finales y a las aplicaciones de propósito general,
- Implementar una solución sobre TCP, donde existen distintas maneras de lograrlo:
    - Utilizando SSL (Secure Sockets Layer) o el estándar de Internet TLS (Transport Layer Security) ya sea a través de una generalidad completa de SSL (TLS) donde puede ser provisto como parte de un protocolo y así ser transparente a las aplicaciones o a través de SSL embebido en algún paquete específico, por ej. El browser IExplore viene con SSL (16).
    - Creando túneles entre dos entidades utilizando SSH.
- El servicio de seguridad en capa de aplicación puede ser embebido en una aplicación en particular, por ejemplo Kerberos o S/MIME

En esta sección se explicará cómo implementar seguridad sobre TCP, utilizando SSL o TLS y se introducirá brevemente en SSH.

SSL está diseñado para usarse con TCP y así proveer servicios de seguridad confiable extremo a extremo. Existen dos conceptos importantes en SSL: conexión y sesión.

Conexión se refiere al transporte de un tipo de servicio, en SSL las conexiones son relaciones peer-to-peer y son transitorias, cada conexión se asocia a una sesión. Una sesión SSL es una asociación entre un cliente y un servidor creadas por el protocolo Handshake y definida por un conjunto de parámetros de seguridad criptográficos. Pueden existen múltiples conexiones seguras entre las partes. En la práctica esto no es muy común y se aplica distintos estados en cada conexión.

TLS es un estándar IETF de capa de Transporte, definido en la RFC 5246, cuyo objetivo es tener una versión estándar de Internet de SSL, la primera versión de TLS fue conocida como SSLv3.1. Provee una conexión segura con autenticación mutua, confidencialidad e integridad de datos, generación y distribución de claves, negociación de los parámetros de seguridad. Sin embargo en este protocolo no cuenta con la posibilidad de multiplexar distintas aplicaciones sobre una única sesión TLS.

Para poder utilizar una única sesión a través de varias aplicaciones surgió el protocolo **MTLS** (TLS Multiplexing), es un protocolo a nivel aplicación que corre sobre el protocolo de registro de TLS. El diseño de MTLS provee multiplexación de aplicación sobre una única sesión TLS. Por lo tanto en vez de asociar una sesión TLS con cada aplicación, MTLS permite varias aplicaciones y protege de los intercambios sobre la única sesión TLS. MTLS está actualmente en estado borrador con fecha abril del 2011 (17).

Arquitectura SSL/TLS

A SSL lo conforman dos capas de protocolos: Protocolo de registro SSL (SSL Record Protocol) y los protocolos de capa superior: Handshake Protocol, Change Cipher Spec Protocol, y Alert Protocol, los cuales administran los intercambios SSL (Figura 7). El protocolo de registro SSL provee los servicios básicos de seguridad a varios protocolos de capas superiores. Entre ellos, se puede mencionar HTTP (hypertext transfer protocol), proveyendo de un servicio de transferencia web entre el cliente y el servidor que opere con SSL (**HTTPS**), él cual se explica más adelante.

Con SSL se puede lograr:

- Confidencialidad, a través del protocolo Handshake quien define una clave secreta compartida que es usada para encriptar los payloads SSL
- Integridad del mensaje, que se obtiene cuando el protocolo Handshake forma un código de autenticación de mensaje (MAC) con la clave secreta compartida.
- Autenticación, cuando el protocolo Handshake permite que el cliente y el servidor se autentiquen uno con otro y negociar el algoritmo de encriptación,

la MAC y las claves criptográficas utilizadas para proteger los datos que envía el registro SSL.

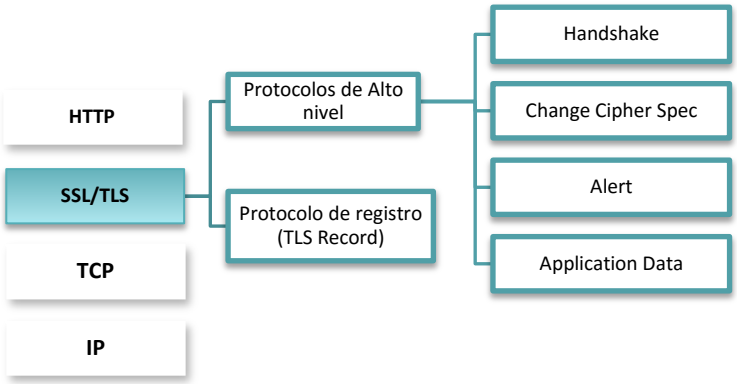


Figura 7. Protocolos que conforman SSL

TLS mantiene el formato del registro, así como los campos de cabecera, y hace uso del algoritmo HMAC definido en la RFC 2104 y de la función pseudo aleatoria para expandir el secreto dentro de los datos ya sea para la generación de la clave o para la validación.

La clave secreta compartida es un valor de 48 bytes de única vez generado en cada sesión para asegurar el intercambio de claves, se realiza en dos etapas: la primera el secreto pre-master es intercambiado, segundo el secreto master es calculado en ambas partes. El intercambio se realiza a través de RSA[2] o por Diffie-Hellman (DH). Si se usa RSA el secreto pre-master es generado por el cliente y encriptado con la clave RSA pública del servidor y enviado al servidor, el cual desencripta usando su clave privada y obtiene el secreto pre-master, mientras que en Diffie-Hellman el cliente como el servidor generan una clave pública con DH, luego intercambien estas claves y cliente y servidor realizan el cálculo DH para crear el secreto pre-master compartido (Figura 8).

La clave pública del servidor será expuesta a través de un certificado digital (explicado en la sección posterior), el cual deberá estar disponible para el usuario del servicio.

Luego el certificado de clave pública debe ser firmado por la CA. Una firma se crea mediante la adopción de la función de hash de un mensaje y el cifrado con la clave

[2] RSA (Rivest, Shamir y Adleman) es un sistema criptográfico de clave pública desarrollado en 1977. Es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente.

pública del remitente, si la firma es DSS[3] el algoritmo hash usado es SHA-1 o SHA-256 y si la firma es RSA el hash se calcula con MD5 y SHA-1 y luego se concatenan y se encriptan con la clave pública del servidor.

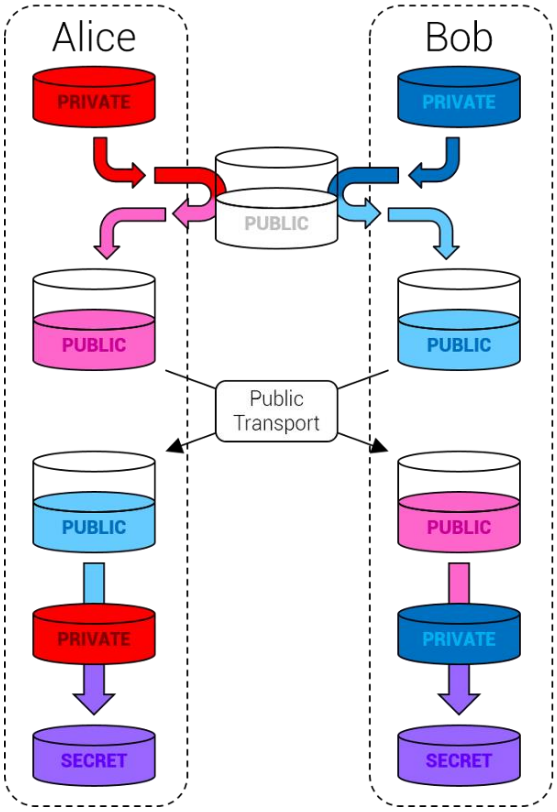


Figura 8. Intercambio de cables usando Diffie-Hellman

Funcionamiento SSL

La parte más compleja de SSL es el protocolo Handshake, el cual es usado antes que los datos de cualquier aplicación sean transmitidos.

El protocolo handshake cuando recibe los datos enviados por las aplicaciones de capa Aplicación, comienza una serie de intercambios de mensajes entre el cliente y el servidor para establecer los parámetros de seguridad de la comunicación (Figura 9).

[3] DSA (Digital Signature Algorithm) es un estándar del Gobierno Federal de los Estados Unidos de América o FIPS para firmas digitales. Sirve para firmar y no para cifrar información. Una desventaja de este algoritmo es que requiere mucho más tiempo de cómputo que RSA.

- Fase 1: Establecer los parámetros de seguridad. Es la fase inicial de una conexión lógica y se establecen los parámetros de seguridad asociadas a la conexión. El intercambio lo inicia el cliente enviando un mensaje client_hello al servidor y espera por un mensaje server_hello que contenga los mismos parámetros que él le envió.
- Fase 2: autenticación del servidor e intercambio de claves. El servidor comienza esta fase enviando su certificado para la autenticación. Según el método que se haya utilizado para el intercambio de claves, es necesario enviar el mensaje server_key_exchange.
- Fase 3. Autenticación del cliente e intercambio de claves. Una vez recibido el mensaje server_done, el cliente debe verificar que el servidor provee un certificado válido y debe chequear la validez de los parámetros. Si todo es satisfactorio, el cliente envía los mensajes al servidor. Si el servidor requiere un certificado el cliente comienza con la fase de envió de mensaje de certificado. Si el certificado no es adecuado el cliente envía una alerta de no_certificate.
- Fase 4. Finalización. Esta fase complete la configuración para una conexión segura.

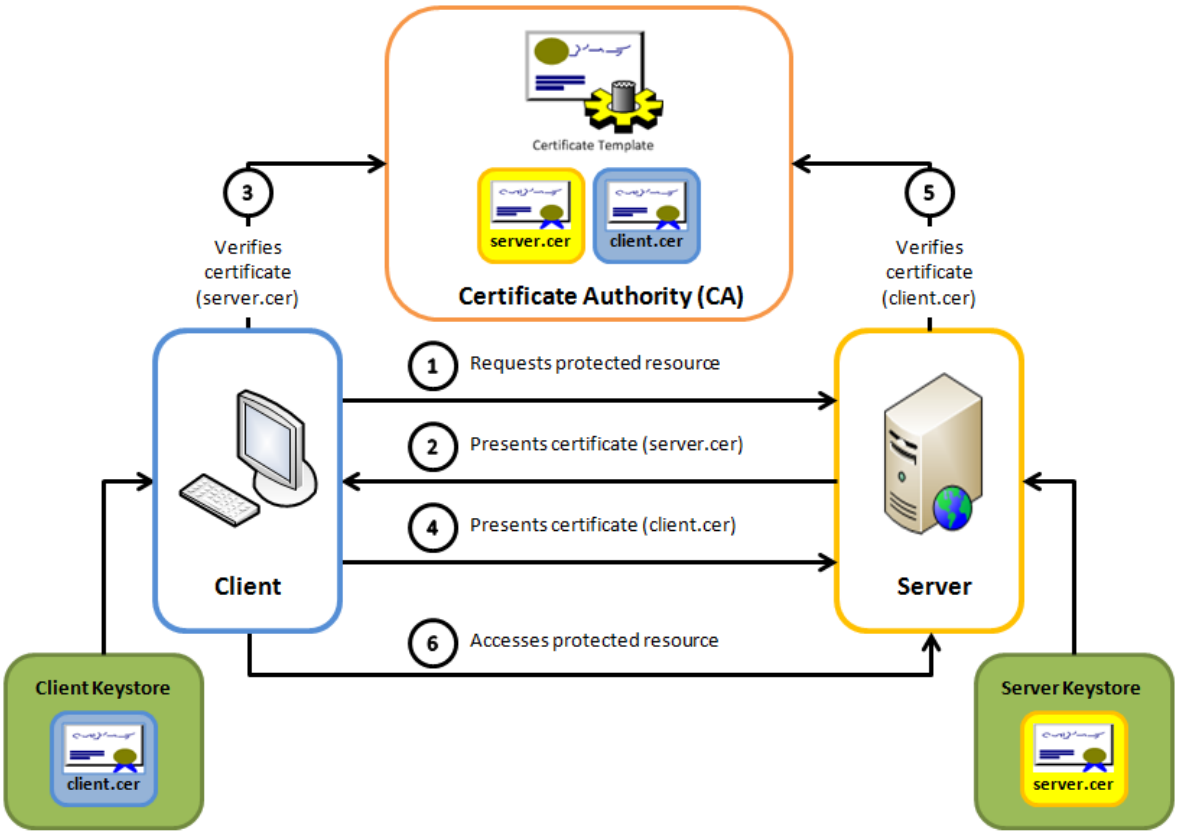


Figura 9. Autenticación basada en certificados digitales entre un cliente y un servidor.

El protocolo de registro SSL recibe el mensaje a ser transmitido, y con los parámetros de seguridad establecidos por el protocolo Handshake, comienza a fragmentar el mensaje en bloques, opcionalmente los comprime, se le aplica el código de autenticación de mensaje (MAC), se lo encripta y se le agrega una cabecera, todo esto conforma un segmento TCP. Para la encriptación de un flujo se considera el mensaje comprimido y la MAC usando un algoritmo de encriptación simétrico (Figura 10).

El objetivo es hacer uso del pequeño valor correspondiente al secreto compartido pero generar grandes bloques de datos de manera de asegurar de distintos tipos de ataques realizados sobre las funciones hash y los códigos MAC. Para brindar mayor seguridad se usan ambos algoritmo hash MD5 y SHA, cada uno se aplica a una parte del mensaje.

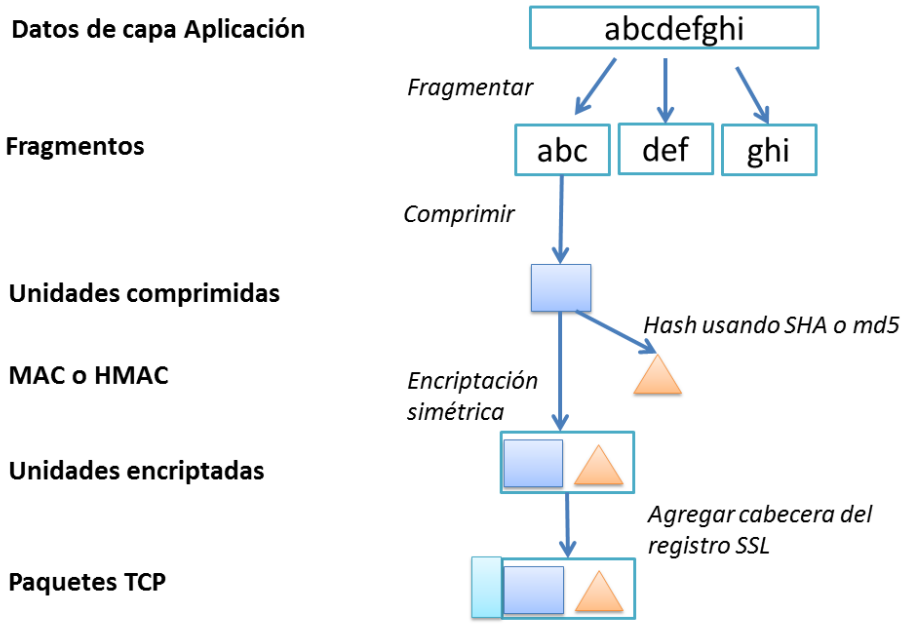


Figura 10. Operación del Protocolo de Registro SSL

Certificados digitales

El estándar X.509 define la estructura de los certificados y un protocolo alternativo de autenticación basado en el uso de certificados de clave pública. Se basa en el uso de criptografía de clave pública y firma digital. No exige el uso de algún algoritmo específico pero recomienda RSA y para la firma digital usar alguna función hash.

El corazón del esquema X.509 es el certificado de clave pública asociado con cada usuario. Así el certificado del usuario será creado por alguna autoridad certificadora (CA). Cada certificado contiene la clave pública del usuario y es firmada por la clave privada de la CA.

Para disponer de certificados, es necesario disponer de una autoridad certificadora (Certificate Authority - CA) que valide el certificado. Las CA verifican la autenticidad de la empresa, firman el certificado de seguridad con alguno de los certificados raíz de la CA bajo una fuente de encriptación, quedando el certificado listo para ser utilizado en el servidor.

La CA firma el certificado con la clave privada. Si la clave pública correspondiente es conocida por el usuario, luego el usuario puede verificar que el certificado fue firmado por la CA es válido.

Los campos que conforman un certificado son:

- Versión: Por defecto es 1. Si el identificador único está presente debe ser 2. Si existen más de una extensión la versión debe ser 3.
- Número de serie: es un valor entero único asociado a la CA.
- Identificador del algoritmo de firma: indica que algoritmo se usó para firmar el certificado junto con los parámetros asociados.
- Nombre del emisor: nombre de la CA que creó y firmó el certificado.
- Período de validez: consiste en dos fechas, la primera y la última fecha de validez del certificado.
- Nombre del sujeto: nombre del usuario a quien se refiere el certificado. Ya que el certificado tiene la clave pública del sujeto, quien almacena la clave privada correspondiente.
- Información de la clave pública del sujeto: la clave pública del sujeto además del identificador de algoritmo que se usó con la clave junto con los parámetros asociados.
- Identificador único del emisor: un campo opcional usado para identificar unívocamente a la CA emisora
- Identificador único del sujeto: campo adicional usado para identificar al sujeto.
- Extensiones: un conjunto de campos de extensión. Una extensión fue incorporada en la versión 3.
- Firma: contiene un código hash de los otros campos encriptados con la clave privada de la AC. Este campo incluye el identificador del algoritmo de firma.

El estándar utiliza la siguiente notación para definir al certificado:

CA <<A>>= CA {V, SN, AI, CA, UCA, A, UA, Ap, TA}

CA <<A>>: El certificado de un usuario A emitido por la autoridad certificadora CA

CA[I]: será firmado con I por CA. El resultado de I será un código hash encriptado agregando:
V: la versión del certificado,
SN: número de serie del certificado,
AI: identificador del algoritmo usado para firmar el certificado,
CA: Nombre de la autoridad certificante,
UCA: (opcional) identificador único de la CA,
A: Nombre del usuario
UA: (opcional) identificador único del usuario A,
Ap: clave pública del usuario A,
TA: Periodo de validez del certificado.

Una autoridad Certificadora puede ser:

- ***Autoridad Certificadora privada:*** entidad privada que venden los certificados, , por ej. Verisign, Thawte, beTRUSTed, etc., validando la autenticidad de las empresas que adquieren los certificados. En este caso los navegadores web tienen los certificados de estas CA y por lo tanto reconocen los certificados validados por ellas, siendo el proceso de encriptación de la comunicación transparente para el usuario web;
- ***Autoridad Certificadora local***: entidades locales instalados en la propia empresa, la cual genera y firma los certificados a utilizar en el servidor web como en el resto de los servicios que precisen encriptación, la desventaja en este tipo de CA es que son locales y por lo tanto los navegadores web de los usuarios no disponen del certificado raíz de la CA, mostrando un mensaje de "certificados no válido", es un mensaje de alerta de seguridad, si el usuario autoriza la comunicación con este sitio o instala en forma manual este certificado en futuras comunicaciones no se observará ese mensaje de alerta.

Un certificado generado por una CA tiene las siguientes características:

- Cualquier usuario con acceso a la clave pública de una CA puede verificar que la clave pública del usuario fue certificada.
- Ninguna parte puede modificar el certificado sin ser detectado.
- Todos los certificados de los usuarios pueden ser ubicados en el directorio común con acceso de todos los usuarios. De igual modo un usuario puede transmitir su certificado a otro usuario. Si el usuario B tiene el certificado del usuario A, B puede tener confidencialidad en el mensaje ya que lo encripta con la clave pública de A evitando que sea observado el mensaje en tránsito. Si el mensaje es firmado con la clave privada de A es infalsificable.

Revocación de Certificados: Los certificados puede revocarse por expiración del periodo de validez. Pero también existen ocasiones donde es deseable revocar algún certificado antes que expire:

- La clave privada del usuario ha sido comprometida.
- El usuario ya no está certificado por esa CA
- El certificado de la CA ha sido comprometido

Cada CA mantiene una lista de todos los certificados revocados que no han expirado emitidos por la CA.

Cada lista de certificados revocados (CRL) es guardada en el directorio y firmada por el emisor. Incluye el nombre del emisor, fecha de creación de la lista, fecha de emisión de la siguiente lista y entrada de cada certificado revocado.

Cuando un usuario recibe un certificado, debe determinar si no ha sido revocado. El usuario podría revisar el directorio cada vez que recibe un certificado, pero para evitar la demora de búsqueda en la AC, es preferible que el usuario almacene en su cache local la lista de certificados revocados.

Extensiones de los Certificados digitales

Al referirse a Certificados Digitales surgen distintas extensiones de los archivos. Se comentarán algunas de ellas (16) (17).

- **.pem**: definido en la RFC's 1421 como un formato contenedor que puede incluir un certificado público solamente o puede incluir un certificado donde se incluya la clave pública, la clave privada y el certificado raíz. El nombre es Privacy Enhanced Mail (PEM), basado en una traducción de base64 de las claves x509 ASN.1. Usado generalmente en software libre
- **.pkcs12 .pfx .p12:** originalmente definidas por RSA como un estándar criptográfico de clave pública, la variante "12" fue introducido por Microsoft, convirtiéndose en un estándar privado para mejorar la seguridad de los archivos pem, pfx se utiliza en los navegadores IE y p12 en Firefox. A diferencia de los archivos .pem este contenedor está totalmente encriptado. Usado preferentemente por sistemas Windows y ampliado su uso por sistemas libres a través de Openssl que permite introducir el par de claves en un archivo .pem.
- **.cert y crt**: formatos de exportación de la clave pública del certificado, corresponde al requerimiento de firma del certificado. El formato actual es PKCS10 definido en RFC 2986, incluye algunos detalles de la clave del certificado requerido como sujeto, organización, estado, etc, como la clave pública del

certificado que debe ser firmada por la CA. El certificado es un certificado público que incluye la clave pública pero no la clave privada.

- **.der**: una manera de codificar la sintaxis ASN.1 en binario, un archivo .pem es un archivo .der codificado en Base64, es decir es el formato padre de pem. Por defecto Windows exporta los archivos de certificados con extensión .der
- **.key:** Es un archivo PEM que contiene solo la clave privada de un certificado específico. Es una extensión usada convencionalmente pero no estandarizada. En Apache frecuentemente se almacena en /etc/ssl/private.
- **.p7b**: definido en RFC 2315 es usado para el intercambio de certificados en Windows.
- **.pkcs7:** estándar abierto usado por java y soportado por Windows. No contiene la clave privada.
- **.crl**: corresponde a la lista de certificados revocados. La Autoridad Certificadora genera una lista de certificados revocados que puede ser descargada de los sitios web de las CA.

HTTPS

HTTPS (HTTP sobre SSL) se refiere a la combinación de HTTP y SSL (o TLS) para implementar una comunicación segura entre el navegador web y el servidor web. El protocolo HTTPS utiliza el puerto 443 y está documentado en la RFC 2818 (16).

Los elementos que son encriptados son:

- URL del documento requerido
- El contenido del documento
- El contenido del formulario completado por el usuario en el browser.
- Las cookies enviadas desde el browser al servidor y desde el servidor al browser
- Contenido de la cabecera HTTP

Cuando se inicia una conexión, el agente actúa como un cliente HTTP y actúa como un cliente TLS, cuando se ha finalizado el intercambio TLS el cliente inicia el primer requerimiento HTTP. Todos los datos HTTP serán enviados como datos de aplicación TLS. Por lo tanto ante una solicitud TLS comienza estableciendo una conexión TCP entre la entidad TCP del lado cliente y la entidad TCP del servidor. Para comprobar la autenticidad del cliente y el servidor, se realizan los intercambios de certificado entre ambos, quienes los validan con la CA. Una vez completada la validación comienza la transmisión de datos a través de un canal encriptado.

Existen niveles de conexión en HTTPS:

- El nivel HTTP donde un cliente HTTP requiere una conexión al servidor HTTP enviando un requerimiento de conexión a la capa de nivel inferior, típicamente TCP, pero puede ser TLS/SSL.
- El nivel de TLS, donde una sesión es establecida entre un cliente TLS y un servidor TLS. La sesión puede soportar una o más conexiones al mismo tiempo.

MTLS

Los protocolos TLS (Tranport layer security) y MTLS (Mutual Transport Layer Security) proveen de encriptación a las comunicaciones y autenticación en los puntos finales en Internet.

Una conexión entre servidor-servidor confiable en MTLS para una autenticación mutua. En una conexión MTLS el servidor origina un mensaje y el otro servidor recepciona el mensaje intercambiando certificados realizan una mutua autenticación con la autoridad certificadora. El certificado provee la identidad de cada servidor al otro.

Si un cliente ejecuta MTLS sobre TLS, debe conectarse al servidor que pasivamente escucha por solicitudes de conexiones TLS en algún puerto válido. El cliente envía un mensaje TLS de conexión para comenzar el intercambio TLS. Una vez completado, el cliente y el servidor pueden establecer y administrar múltiples canales de aplicación utilizando requerimientos y respuestas MTLS.

En la Figura 11 pueden observarse distintos escenarios de comunicación TLS y MTLS. Un ejemplo sería: una oficina de servidores de comunicaciones podría usar estos dos protocolos para crear una red de servidores segura y asegurar que toda la comunicación por la red este encriptado (20). Por ejemplo: Todas las comunicaciones SIP entre un servidores ocurre sobre MTLS y las comunicaciones SIP de un cliente a un servidor ocurren sobre TLS.

TLS y MTLS ayudan a prevenir ataques “eavesdropping” y “man-in-the middle”. En un ataque man in the middle el atacante redirige comunicaciones entre dos nodos red a través de su computadora siendo este accionar desconocido por los administradores de red.

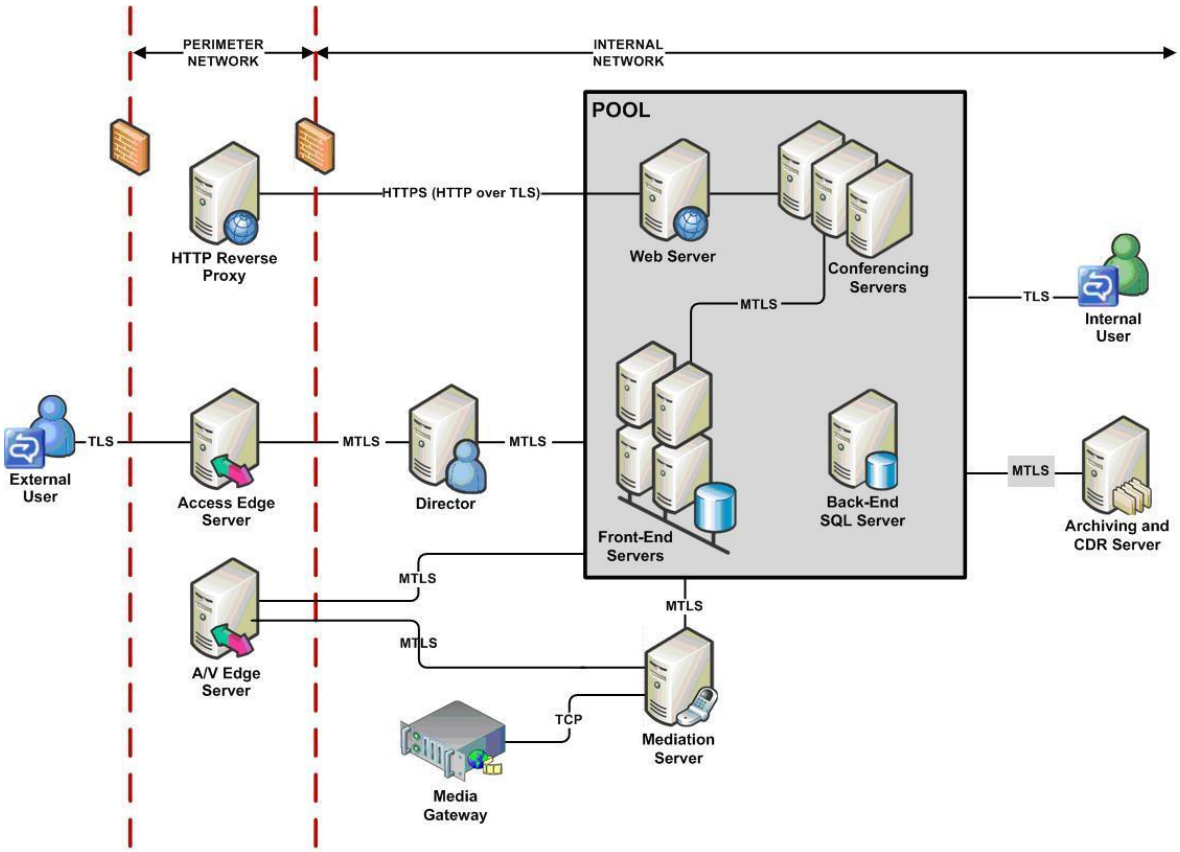


Figura 11. Escenarios de conexión TLS y MTLS

Secure Shell (SSH)

SSH es un protocolo para las comunicaciones en red seguras diseñado para ser relativamente simple y asequible para ser implementado. La versión inicial (SSH1) estuvó enfocada en proveer una conexión remota segura para reemplazar TELNET y otros esquemas de conexión remota que no proveen seguridad.

SSH provee capacidad cliente/servidor más general y puede ser usado para otras funciones de red, como son la transferencia de archivos y correo electrónico.

Una nueva versión de SSH (SSH2) corrige un conjunto de vulnerabilidades presentes en el esquema original. Ha sido propuesta en el estándar IETF RFC 4250 a través de 4256.

Las aplicaciones SSH cliente y SSH servidor están disponible en la mayoría de los Sistemas Operativos, siendo un método elegido para conexión remotas.

SSH está organizado en tres protocolos sobre TCP (Figura 12):

1. Protocolos de capa de transporte: Provee la autenticación del servidor, integridad y confidencialidad de datos. Puede incluir, opcionalmente, compresión.
2. Protocolos de Autenticación del usuario: Autentica al usuario del servidor.
3. Protocolo de conexión: mutiplexa múltiples canales de conexiones lógicas sobre una única conexión.

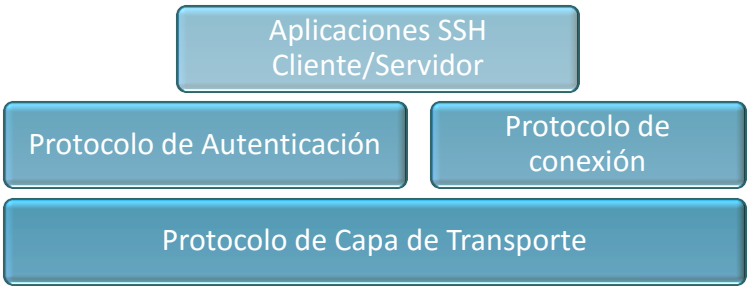


Figura 12. Pila de Protocolos de SSH

2.5. Test de Penetración

Un test de penetración es un método para evaluar la seguridad de una computadora o red simulando un ataque. Un test es una acción que tiende a mostrar una vulnerabilidad en una aplicación.

El método de pruebas de penetración de una aplicación Web indicado por OWASP está dividido en dos modos:

- Modo pasivo: donde se intenta entender la lógica de la aplicación. Las herramientas consisten en obtener información, por ejemplo HTTP proxy para observar las solicitudes y respuestas HTTP, obteniendo información de HTTP: headers, parámetros y cookies.
- Modo activo: incluye una metodología que implica 9 categorías que reúnen 66 controles. Las categorías son:
    - Pruebas de la Configuración de Administración
    - Pruebas de la lógica de Negocio
    - Pruebas de la Autenticación
    - Pruebas de Autorización
    - Pruebas de Administración de Sesión
    - Pruebas de Validación de Datos
    - Pruebas de Denegación de Servicios
    - Pruebas de Servicios Web

3. Desarrollo

En esta sección se llevará a cabo el desarrollo práctico del proyecto, cumplimentando la metodología propuesta por Garzon, siguiendo las etapas por él mencionadas, indicado en la sección 1.1.5 Comenzando con la fase de planeación la cual involucra obtener información sobre la infraestructura web sobre la cual se trabaja, realizando distintas actividades como entrevistas, escaneo de red y web, análisis de vulnerabilidades. Una vez cumplimentada está fase se proseguirá con las fases de arquitectura de red configurando el firewall de host, aseguramiento físico implementado planes de contingencia y la fase de aseguramiento y configuración de Sistema Operativo y los servicios. Luego de aplicar las segurizaciones correspondientes a las fases mencionadas se aplica la fase de auditoría revisando que las configuraciones realizadas cumplan con el objetivo planteado.

Al concluir este capítulo se obtendrá un conjunto de recomendaciones para aplicar la seguridad a cada uno de los elementos de una infraestructura web.

Los elementos dentro de la arquitectura web sobre los cuales se evaluará y aplicará la seguridad pueden observarse en la (Figura 13)

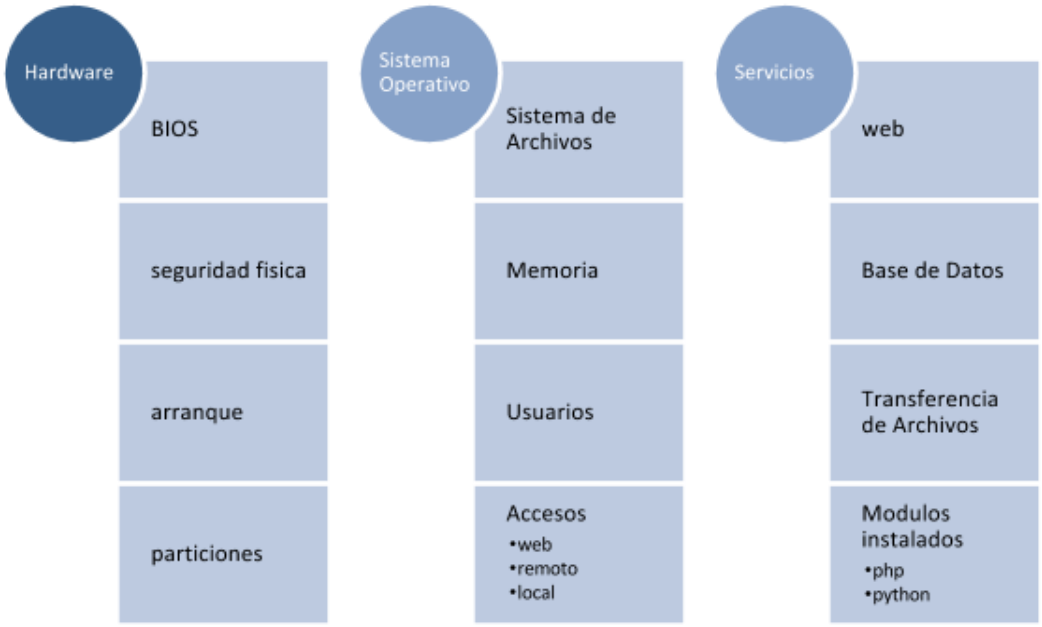


Figura 13. Seguridad en una Arquitectura Web

Para llevar a cabo cada una de las actividades indicadas se han seleccionado distintos software open source, con licencia de software libre ejecutadas sobre un Sistema Operativo Linux o una aplicación web con acceso gratuito. El listado de las herramientas informáticas utilizadas en las distintas etapas del proyecto se han indicado en la Figura 14.

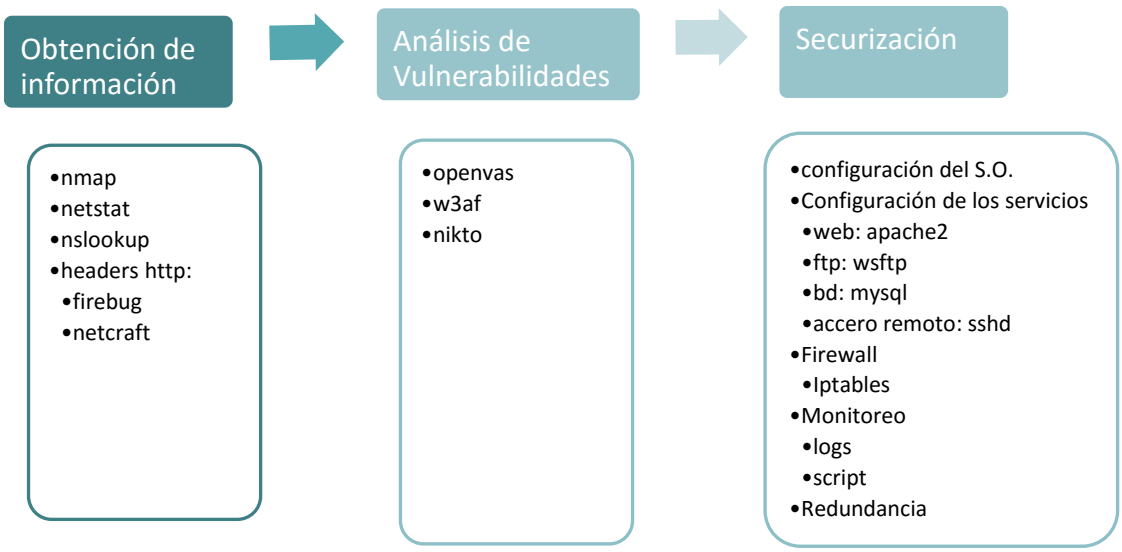


Figura 14. Herramientas utilizadas durante el desarrollo

Se recomienda realizar distintos pruebas sobre la infraestructura y la topología de la arquitectura. A través de estas herramientas se puede obtener mucha información sobre la aplicación web o sobre las configuraciones de la plataforma web sobre la cual se ejecuta la aplicación.

Oswap (6) recomienda la ejecución de los siguientes test, según el análisis que se quiera llevar a cabo:

- Test para evaluar la seguridad de las conexiones remotas al equipo, utilizando los protocolos SSL/TLS (OWASP-CM-001)
- Test de evaluación de configuración del servidor de DB (OWASP-CM-002)
- Test de configuración de Infraestructura y administración (OWASP-CM-003)
- Test de Administración de configuración de la Aplicación Web (OWASP-CM-004)
- Test sobre el manejo de extensiones de archivos (OWASP-CM-005): Las extensiones de archivo pueden exponer sistemas adicionales conectados a la aplicación.

- Test de evaluación de archivos viejos, sin referencias y de backup (OWASP-CM-006)
- Test de evaluación sobre configuración de interfaces de aplicación de administración e infraestructura (OWASP-CM-007)
- Test para evaluar los métodos HTTP y XST (OWASP-CM-008) : en este test se evalúa si el servidor web fue configurado permitiendo daños potenciales de comandos HTTP y si es posible Cross Site Tracing (XST)

3.1. Obtención de información

Esta etapa tiene como objetivo obtener información del sistema (gathering information) para entender y conocer la infraestructura web. Para lo cual se realizó un relevamiento de la Información a través de una entrevista al administrador del equipo, y se utilizaron diferentes herramientas informáticas para obtener información como escaneadores de puertos, escaneadores web, etc. También se llevó a cabo un análisis de vulnerabilidades de las aplicaciones instaladas.

Una vez concluido los análisis utilizando estas herramientas, se evaluó la información resultante, se contrastó con la información brindada por el administrador y se concluyeron cuáles son los puntos claves a considerar en la siguiente etapa correspondiente a la segurización del equipo servidor y las aplicaciones instaladas.

3.1.1. Entrevista

La entrevista al administrador del servidor tiene como objetivo conocer información relacionada al hardware y software instalado en el equipo, como así también información sobre las configuraciones de los diferentes servicios activos.

Se obtuvo la siguiente información:

1. Hardware: El equipo servidor es una computadora con hardware básico, no es un equipo servidor dedicado. Las características son:
   Procesador: AMD FX 4100 3.6 GHZ QuadCore
   Memoria RAM: 4 GB (DDR3) 133 MHZ
   Disco Rígido: 512 GB
2. Sistema Operativo

Linux Ubuntu Server 12.04 (AMD 64 bits) con actualizaciones de seguridad automáticas, la autenticidad de los usuarios son administrados en los archivos /etc/passwd y /etc/shadow

3. Servicios instalados
    - Servidor web: apache2.2.22, utilizando host virtuales para los distintos sitios
    - Servidor Bases de datos: mysql Ver 14.14 Distrib 5.5.22
    - Módulos de lenguajes de programación instalados:
        - PHP 5.5.9
        - python 2.7
    - Acceso web a Base de datos: phpmyadmin 3.4.10.1deb1
    - Transferencia de Archivos FTP: vsftpd server, versión 2.3.5
    - Acceso remoto: openssh. OpenSSL 1.0.1 14 Mar 2012
4. Usuarios

    Existen usuarios que pertenecen al grupo administradores y usuarios restringidos a cada aplicación web limitando el acceso solo al espacio asignado para almacenar los archivos correspondientes al sitio web.
5. Aplicación:

    Se brinda servicio web, transferencia de archivos, administración web de base de datos y acceso ssh restringido a algunos usuarios
6. Configuraciones actuales
    - El servidor web Apache tiene configurado hosts virtuales
    - Existen varios esquemas de bases de datos, la bd test por defecto y el resto para las aplicaciones web, los accesos y privilegios de los usuarios a las mismas están restringidos según la aplicación correspondiente.
    - El servidor de transferencia de archivos FTP tiene una lista de usuarios que solo tienen acceso al servicio FTP, la comunicación no se encuentra cifrada. No tiene habilitado SSL.
    - La comunicación web no se encuentra encriptada
    - No se disponen de certificados digitales.
    - El acceso remoto se realiza únicamente a través de consola a través de SSH, disponiendo de una lista de usuario permitidos.

- Los desarrolladores utilizan un cliente FTP para acceder a los repositorios de código, teniendo solo acceso al espacio de disco asignado para subir el código de las aplicaciones, no pudiendo acceder al resto del disco.
- Para el acceso a la base de datos por parte de los desarrolladores, se dispone de una aplicación web: phpmyadmin, donde a cada desarrollador se le asigna un usuario y password para el acceso al esquema de la base de datos asignada, restringiendo el acceso a otras bases de datos presentes en el servidor de bases de datos.
- Se realizan backup de la información en otro equipo, él envió se realiza utilizando el protocolo samba. Los archivos enviados solo están comprimidos, no encriptados. La comunicación tampoco se encuentra encriptada.
- Con respecto a la encriptación de la información en el equipo, sólo se encuentra encriptado parte del disco, correspondiente al home del usuario administrador.
- No se han aplicado reglas de firewall ni al equipo ni a los servicios.
- Con respecto a documentación que respalda las normativas o políticas de seguridad, se desconocen la existencia de las mismas y por lo tanto las decisiones relativas a seguridad de la información no se rigen por ninguna normativa.

3.1.2. Búsqueda de información: Escaneo

Para obtener información del sistema se utilizaron varias técnicas y varias herramientas:

El escaneo es la determinación de las características de una red o sistemas remotos, con el objetivo de identificar los equipos disponibles y alcanzables desde Internet, así como los servicios que ofrece cada uno. Permite saber los sistemas existentes, los servicios ofrecidos por ellos, organización de los equipos, los sistemas que ejecutan, etc.

Entre los métodos de escaneo se incluyen técnicas como:

- Ping sweep
- Escaneo de puertos
- Firewalking
- Trace routing
- Identificación de sistema operativo

La técnica Ping sweep usa ICMP Echo para identificar los equipos existentes en las redes objetivos de un ataque, típicamente accesibles desde Internet. Constituye uno de los pasos principales en la obtención de información. Empleando los paquetes ICMP de tipo echo o echo reply se puede saber si una determinada IP está activa o no, los mensajes de error ICMP, como red inalcanzables, TTL excedido, o ausencia de respuesta indican un sistema no está disponible que no es alcanzable. Básicamente se ejecuta un ping a la IP o DNS del equipo objetivo, si este responde, el equipo está "vivo".

Escaneo de puertos

Al escanear los puertos de los sistemas se descubren puntos de entrada a los mismos, que abren las puertas a nuevas vulnerabilidades potenciales en base a la implementación del servidor que escucha tras cada puerto.

Además esta técnica permite identificar el tipo de sistema existente, así como el sistema operativo y las aplicaciones que ofrecen un servicio en la red y su versión asociada.

Para conocer los servicios activos en el sistema se realizó un análisis intrusivo ejecutando los comandos **nmap** y **telnet**, que permiten conocer puertos abiertos y servicios activos.

Para el descubrimiento de sistemas se suele utilizar nmap con los parámetros -PA ó -PE que suelen ser suficiente cuando se analizan redes locales, pero para auditorías de seguridad se recomienda utilizar un conjunto más completo de sondas de descubrimiento.

Según el objetivo del análisis se utilizan distintos parámetros (18).

Nmap -PS [lista de puertos] (Ping TCP SYN)

Esta opción envía un paquete TCP vacío con la bandera SYN puesta. El puerto destino por omisión es el 80, pero se puede añadir un puerto o lista de puertos como parámetro (p.ej. -PS22, 23, 25,80). Por ejemplo: Para comprobar si el puerto telnet está abierto, Nmap –p 23 192.168.1.34 y para conocer todos los puertos abiertos se utiliza el carácter * como comodín, Nmap –p * 192.168.1.34 (Figura 15).

nmap -PU [lista de puertos] (Ping UDP)

El ping UDP es otra opción para descubrir sistemas. Esta opción envía un paquete UDP vacío a los puertos indicados. Si no se especifica ningún puerto se utiliza el puerto 31338 por omisión. La principal ventaja de este tipo de sondeos es que atraviesan cortafuegos y filtros que sólo analizan TCP.

nmap -sP <IP>

Esta exploración es útil cuando se desea hacer una búsqueda rápida de la red destino para ver qué hosts están en línea sin tener que detener el escaneo con los objetivos de puertos abiertos.

nmap -F <IP>

La opción -F indica a nmap realizar un análisis de sólo los 100 puertos más utilizados.

nmap -O <IP>

El parámetro –O permite activar el sistema de detección de sistema operativo del equipo analizado (Figura 17).

nmap -sV <IP>

El parámetro -sV se utiliza para detectar la versión de los servicios. Este tipo de prueba es muy poderosa ya que brinda información importante al posible atacante, él cual conociendo la versión de los servicios activos puede realizar ataques aprovechando vulnerabilidades de estos software en esas versiones (Figura 18).

Nmap –sL <IP> (Sondeo de lista)

El sondeo de lista es un tipo de descubrimiento de sistemas que tan solo lista cada equipo de la/s red/es especificada/s, sin enviar paquetes de ningún tipo a los objetivos. Por omisión, Nmap va a realizar una resolución inversa DNS en los equipos, para obtener sus nombres. Es sorprendente cuanta información útil se puede obtener del nombre de un sistema. Por ejemplo fw.chi.playboy.com es el firewall de la oficina en Chicago de Playboy Enterprises. Adicionalmente, al final, Nmap reporta el número total de direcciones IP. El sondeo de lista es una buena forma de asegurarse de que son las direcciones IP correctas del objetivo. En la (Figura 16) se observa información brindada por este comando.

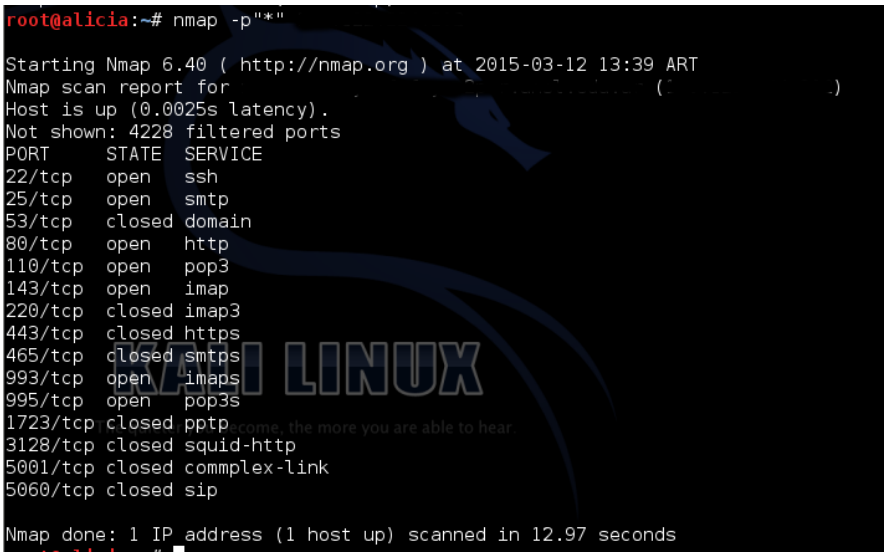


Figura 15. Nmap –p * IP para conocer puertos habilitados

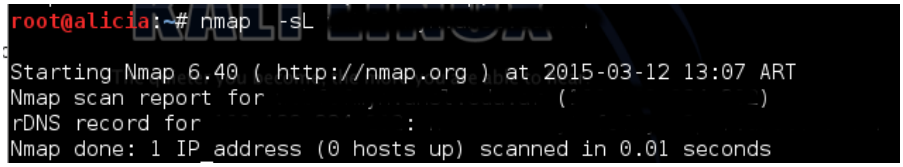


Figura 16. Descubrir información del sistema sin escanear puertos

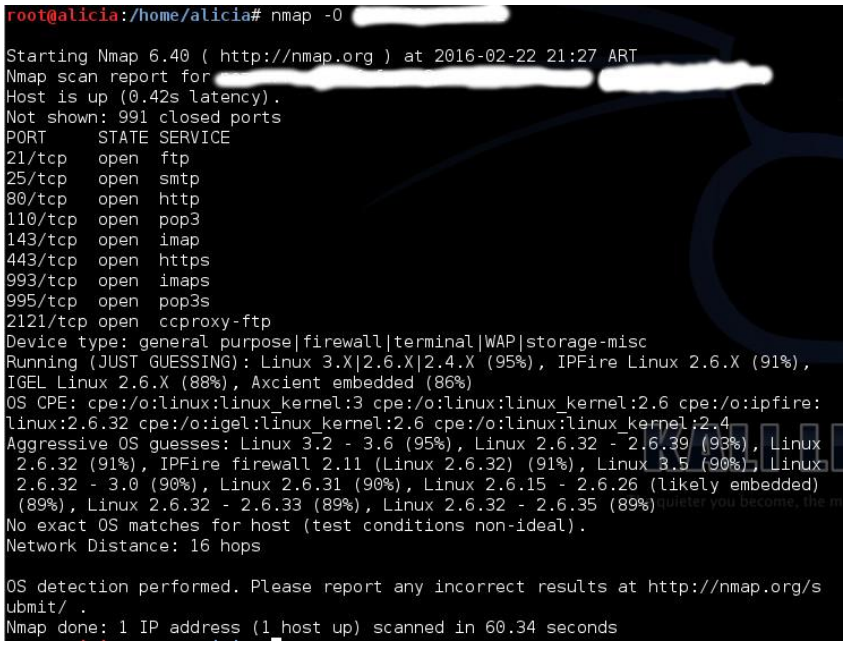


Figura 17. Conocer Sistema Operativo usando NMAP

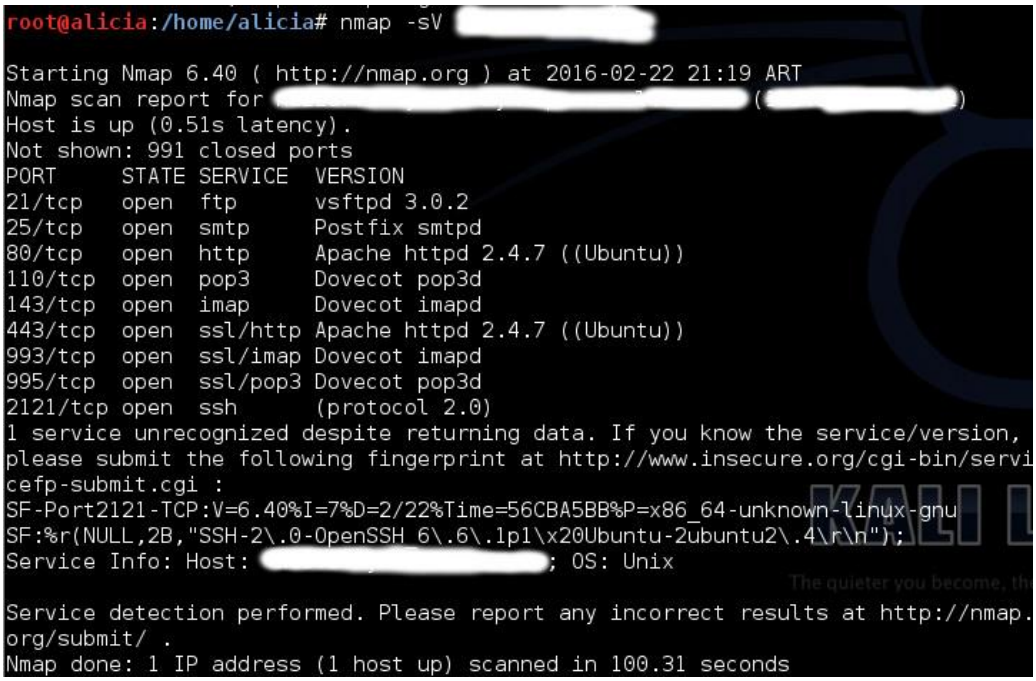


Figura 18. Descubrir versión de los servicios activos

Del análisis realizado con el comando nmap, se observa que los puertos abiertos y las versiones del Sistema Operativo como los servicios son los que se obtuvieron en la entrevista.

Telnet

Otra prueba que se realizó, fue determinar si el puerto de la base de datos se encontraba abierto y se podía acceder desde un acceso remoto usando telnet, el resultado fue negativo, como se ve en la siguiente Figura (Figura 19).

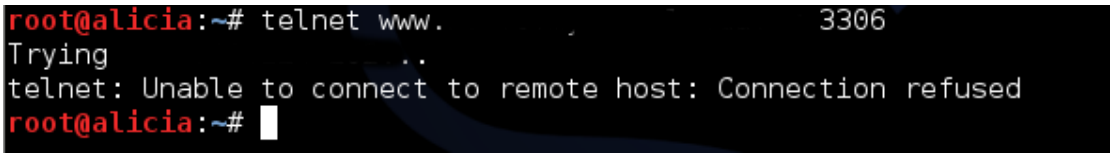


Figura 19. Telnet a Base de datos

Obtener información de las páginas web

Los headers (cabeceras) de una conexión HTTP, muestran información que los atacantes pueden utilizar para conocer las vulnerabilidades de un sistema y así poder ser un punto fácil de ataque. Entre la información que muestran incluye (Figura 20):

- Software y versión correspondiente al servidor web
- Tipo y versión de sistema operativo
- Propietario del domino, etc.
- Software y versión del motor de base de datos
- Lenguaje scripting de desarrollo con su versión
- Dirección IP, DNS, etc.

Con esta información un atacante podría observar las vulnerabilidades de los distintos software instalados y así propiciar un ataque a este servidor web.

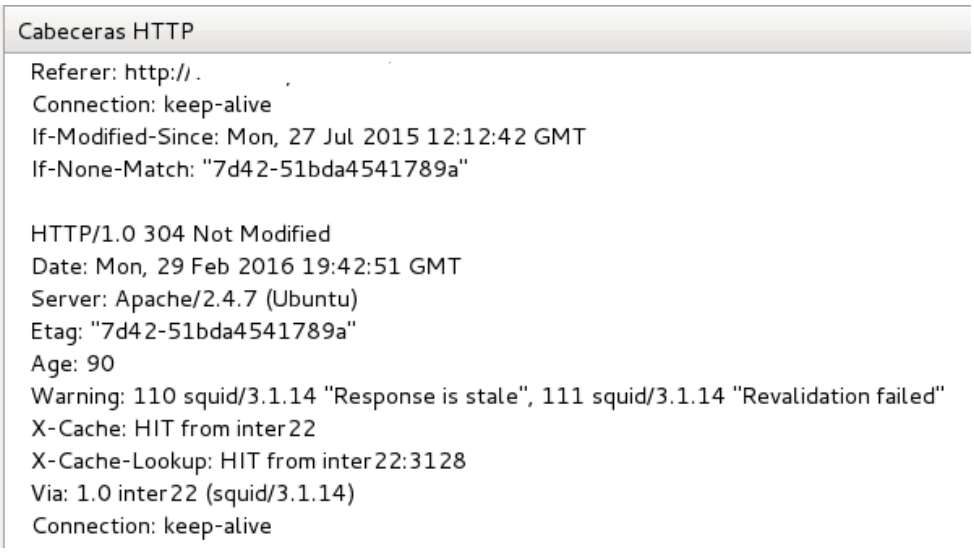


Figura 20. Información en un headers HTTP

Existen distintas herramientas para examinar las cabeceras HTTP (headers), entre ellas:

- Plugins que se instalan en el navegador web, por ej. el plugin Firebug en Mozilla Firefox,
- Comandos de consola, por ejemplo navegadores de consola como w3m, lynx, entre otros. La *(*Figura 21*)* muestra información obtenida usando el browser por consola w3m con el parámetro *dump_head*
- Programas gráficos,
- Sitios online que brindan información de los headers de los sitios web, Por ejemplo Netcraft (19), que también analiza sistema operativo y aspectos de internet como

proveedores de servicios de internet y autoridades de certificados digitales SSL (Figura 22).

Figura 21. Obtener información de headers http usando w3m

Figura 22. Información obtenida de los headers http usando netcraft

En los headers también puede observarse si se han aplicado algún tipo específico de configuración y a partir de ahí realizar determinado tipo de ataques, como Cross Site Scripting (XSS), clickjacking, entre otros. Estos ataques podrían evitarse configurando correctamente el servidor apache, seteando algún tipo específico de directivas. En las siguientes imágenes (Figura 23, Figura 24, Figura 25) puede observarse los headers mostrados por el plugin firebug y las directivas que informan si el sistema es vulnerable o no por cada tipo de ataque.

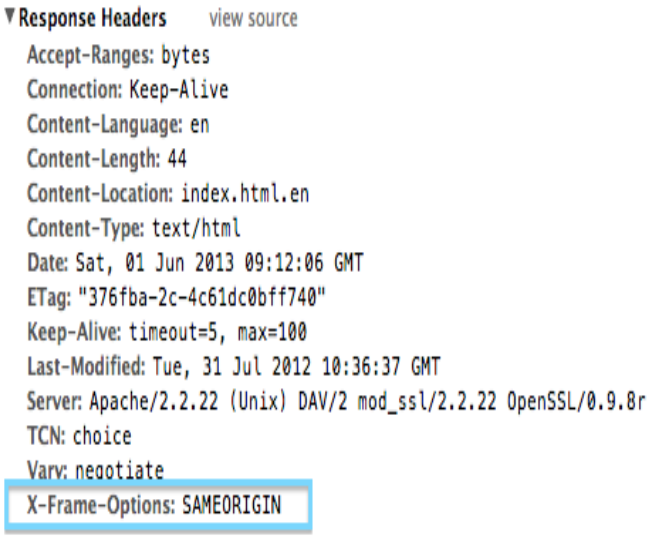


Figura 23. Información útil para ataques Clickjacking: opción X-Frame-Options

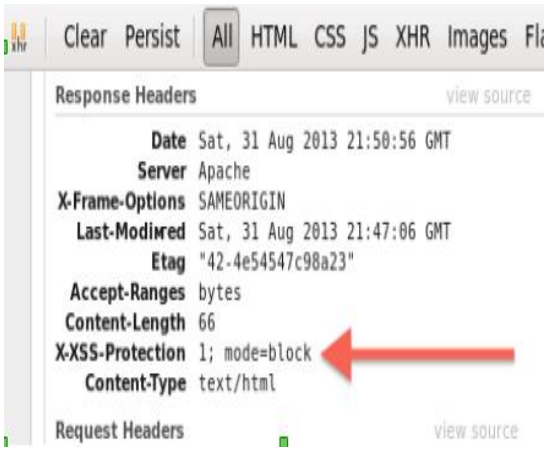


Figura 24. Información para ataques X-XSS

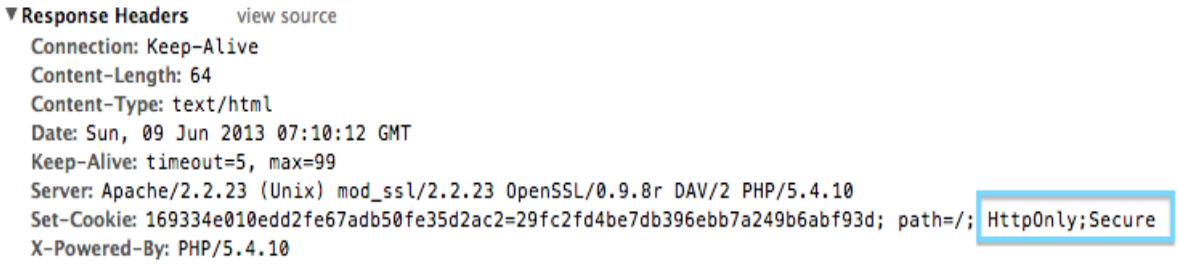


Figura 25. Información útil para ataques Cross Site Scripting (CSS)

Obtener información sobre el nombre de dominio

Existen numerosas técnicas que son usadas para identificar nombres de dominio (DNS) asociado a una dirección IP. Por ejemplo los comandos: host, nslookup, dig permiten obtener una IP dado un dominio o viceversa.

3.2. Análisis de Vulnerabilidades

Para evaluar la seguridad ya sea en la configuración del servidor como en los diferentes servicios y módulos instalados, se llevarán a cabo el análisis de vulnerabilidades utilizando distintas herramientas open source online o instaladas. El análisis afectará a:

- Sistema Operativo
- Servicios
  - web: Apache
  - Base de Datos: Mysql y visualizador web: phpmyadmin
  - FTP: wsftp
  - ssh
- Lenguajes de Programación
  - PHP
  - python

El Sistema Operativo es el software obligatorio que cual controla el acceso y uso de los recursos de una máquina, siendo uno de los elementos más buscados para intentar explotar cualquier vulnerabilidad. Algunas de las vulnerabilidades típicas del SO son:

- Ausencia de administración de las actualizaciones del SO
- Aplicaciones obsoletas de terceras partes
- Carencia de ejecución de las contraseñas
- Falta generalizada de rigor en el sistema: Por ej. FTP anónimo que permite que cualquiera acceda a archivos sensibles, recursos compartidos sin protección que permiten la enumeración de cuentas de usuario, cualquier servicio que se le ocurra, personas que no necesitan tener información sobre la configuración del sistema pero la tienen, lo cual las coloca en un lugar privilegiado para allanar el sistema.
- Ausencia de respaldo.

Los servicios que brinda un servidor web generalmente están expuestos al uso público y accesible desde Internet, por lo cual existen muchas posibilidades de ser atacados, para llevar a cabo estos ataques se utilizan los puntos débiles de cada servicio, estos puntos débiles o “vulnerabilidades” se deben reconocer y mitigar para así proteger al equipo e información.

Scanner web

Los scanners de aplicaciones web o de servidores web son herramientas que permiten ayudar a prevenir ataques detectando vulnerabilidades en la configuración del servidor o en la programación de las aplicaciones, por ejemplo: SQL injection, buffer overflows, cross site scripting, ejecución de archivos maliciosos y sesiones hijacking, entre otros.

En este trabajo se utilizaron: Openvas, Nikto, w3af.

Openvas (20): framework que brinda distintos servicios y herramientas, permitiendo ser un poderoso y comprensible escáner de vulnerabilidades y una solución de administración de vulnerabilidades. El escáner es muy efectivo y dispone de una base de datos de test de vulnerabilidades de red actualizados (NVTs). El informe brindado por Openvas muestra por cada vulnerabilidad encontrada, el nivel de amenaza y cuál es el servicio afectado, así también URL para mayor información.

W3AF (Web Application Attack and Audit Framework) (21): permite encontrar y explotar vulnerabilidades de las aplicaciones web, por ejemplo SQL injection, ataques XSS, buffer overflow, ejecución de archivos maliciosos, administración de sesión, etc. Formado por distintos plugins que permiten realizar descubrimiento, auditoría o incluso explotación a través de un conjunto de exploits ya disponibles.

Nikto (22): scanner de servidor web Open Source que permite realizar pruebas comprensibles en múltiples ítems, ya que puede verificar más de 6000 programas ó archivos potencialmente dañinos, y revisar versiones obsoletas. También permite revisar las configuraciones del servidor como la presencia de múltiples archivos índex, opciones del servidor HTTP, etc.

Resultados Obtenidos

Es importante hacer notar que los software mencionados son herramientas automatizadas que pueden devolver falsos positivos, por lo cual es necesario la intervención humana para detectar los vulnerabilidades realmente positivas

OpenVas

El resultado del análisis de este software es muy completo, en formato pdf. La información la clasifica por severidad en alta, media y baja indicando la cantidad de ocurrencias de cada una. También informa los logs encontrado en el análisis. En la (Figura 26) se pueden observar los ítems correspondientes a los servicios según el nivel de amenaza encontrados.

Service (Port)	Threat Level
imap (143/tcp)	High
imaps (993/tcp)	High
pop3 (110/tcp)	High
pop3s (995/tcp)	High
smtp (25/tcp)	High
imaps (993/tcp)	Medium
pop3s (995/tcp)	Medium
smtp (25/tcp)	Medium
general/tcp	Medium
imap (143/tcp)	Log
imaps (993/tcp)	Log
pop3 (110/tcp)	Log
pop3s (995/tcp)	Log
smtp (25/tcp)	Log
general/tcp	Log
general/CPE-T	Log
general/HOST-T	Log
general/icmp	Log
http (80/tcp)	Log
ssh (22/tcp)	Log

Figura 26. Listados de servicios según nivel de amenaza reportado por OpenVas

En el informe por cada vulnerabilidad reportada, se menciona el grado de severidad, nombre del test de vulnerabilidad ejecutado, CVE donde se ha reportado la vulnerabilidad, y URLs de información. La Figura 27 permite ver una salida de OpenVas.

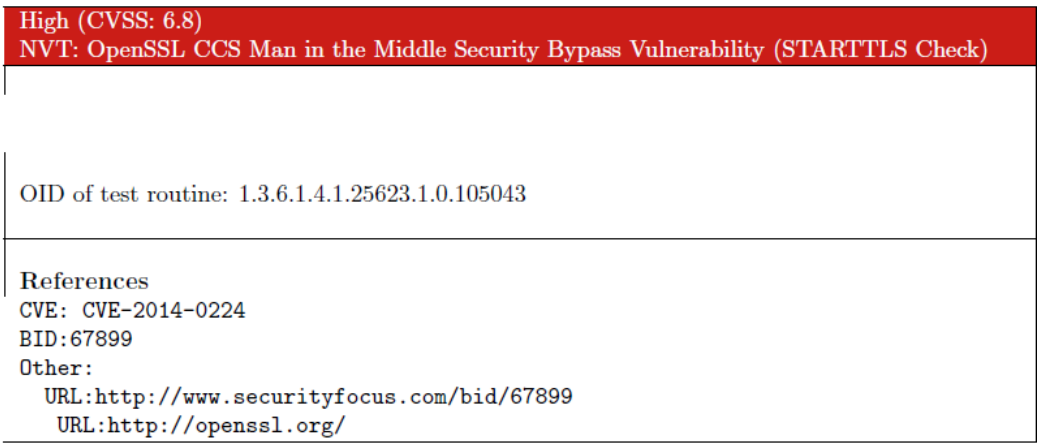


Figura 27. Vulnerabilidad encontrada por OpenVas

En el reporte se han encontrado vulnerabilidades que responden a escaneo de directorio, revisión de los cifradores SSL, información sobre tiempos de TCP, información sobre servicios en ejecución (tipo, nombre y versión), puertos abiertos, banners, expiración de certificados SSL, OS Fingerprinting, traceroute, CGI encontrados.

W3AF

La salida del análisis realizado por w3af retorno los siguientes resultados, donde se puede observar fecha del análisis, plugin que se ejecutó para comprobar ciertos puntos, información que podría ser utilizada por un atacante para obtener acceso o explotar alguna vulnerabilidad.
A modo de ejemplo se mencionarán algunas líneas obtenidas de la salida del análisis

En esta línea se observa que se utilizó el plugin de fuerza bruta y detecto un directorio con respuesta OK (Cod 200), así mismo el nombre del directorio "wp-content" da información del framework de desarrollo que se está utilizando "WordPress". Si el directorio no se encuentra bien configurado podría ser una puerta de entrada al servidor. La información del framework es útil para que el atacante pueda observar las vulnerabilidades de este software y tratar de explotarlas.

> [jue 21 may 2015 20:59:28 ART] Directory bruteforcer plugin found directory "http://web /wp-content/" with HTTP response code 200 and Content-Length: 0.

En la línea siguiente se observa que se encontró la palabra "fix" que puede ser útil, permite conocer el código de la aplicación.

> [jue 21 may 2015 19:40:17 ART] A comment with the string "fix" was found in: "http://web". This could be interesting. This information was found in the request with id 1.

Las siguientes líneas muestran la información del servidor y del módulo PHP, información muy útil para un atacante.

> [jue 21 may 2015 19:40:19 ART] The server header for the remote web server is: "Apache/2.2.9 (Debian) PHP/5.2.6-1+lenny16 with Suhosin-Patch". This information was found in the request with id 6.
>
> [jue 21 may 2015 19:40:21 ART] "X-Powered-By" header for this HTTP server is: "PHP/5.2.6-1+lenny16". This information was found in the request with id 7.
>
> [jue 21 may 2015 19:40:25 ART] The remote Web server has a custom configuration, in which any non existent methods that are invoked are defaulted to GET instead of returning a "Not Implemented" response. This information was found in the requests with ids 4 to 5.

En esta línea se observa que el sitio es vulnerable a ataques clickjacking

> [jue 21 may 2015 19:40:41 ART] The whole target has no protection (X-Frame-Options header) against ClickJacking attack

Nikto

Muestra resultados similares a los anteriores y algunos más, como: Información del sistema y versión de PHP, alerta sobre falta protección de ataque clickjacking y cross site scripting, directorios indexados, vulnerabilidad header etag, entre otras.

```
+ GET /: Retrieved x-powered-by header: PHP/5.3.10-1ubuntu3.4
+ GET /: The anti-clickjacking X-Frame-Options header is not present.
+ -12184: GET /index.php?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000:
/index.php?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive
information via certain HTTP requests that contain specific QUERY strings.
+ -3268: GET /ayuda/: /ayuda/: Directory indexing found.
+ -3092: GET /ayuda/: /ayuda/: This might be interesting...
+ GET /phpmyadmin/changelog.php: Cookie phpMyAdmin created without the httponly flag
+ GET /prueba/: Server leaks inodes via ETags, header found with file /prueba/, inode:
10358555, size: 34, mtime: 0x5108ccfaa1212
```

De la información obtenida, sobre la versión del software instalado y realizando control sobre las vulnerabilidades reportadas por cada software, se observa que la versión del software de acceso web a la Base de datos: phpmyadmin 3.4.10.1deb1 es vulnerable sugiriendo actualizarlo a una versión que haya subsanado esta vulnerabilidad.

3.3. Configuración de Seguridad

En esta sección se aplicarán las medidas necesarias para mitigar o minimizar los posibles ataques, a partir del listado de vulnerabilidades encontradas en la sección previa. Se seguirán distintas recomendaciones de seguridad, las cuales se aplicaran en las siguientes áreas:

- Inicio del equipo
- Sistema Operativo
- Gestión de Red
- Servicios del Sistema Operativo
- Módulos de lenguajes de programación

Además se brindarán consejos de seguridad para los desarrolladores de aplicaciones web y para los administradores de sistemas.

3.3.1. Seguridad Física y de arranque

Una vez que los atacantes ganan acceso al equipo, es importante que pueden encontrarse con distintas barreras evitando tengan acceso al sistema o a modificar los parámetros de hardware del equipo: memoria, procesador, unidades de almacenamiento, entre otras. Se mencionarán tips de seguridad para mitigar el acceso a la BIOS y al arranque del sistema.

- **BIOS:** A continuación se lista un conjunto de acciones para proveer un nivel mínimo de seguridad a la BIOS (23):
    - Configurar password de acceso a la BIOS. Medida de seguridad para prevenir bootear el sistema o realizar cambios malintencionados.
    - Bloquear la secuencia de arranque de un equipo, limitándola solo al disco duro
    - Eliminar cualquier dispositivo (disquetera, cd rom, etc) que no se utilice
    - Eliminar los puertos serie, paralelos o usb limitando la posibilidad de instalar una grabadora de CDROM o disco duro externo y así arrancar el sistema desde ahí y/o extracción de datos.
    - Tener distintas particiones para los directorios que los usuarios tienen acceso: /tmp y /home. Esto brinda mayor seguridad y disponibilidad, un /home lleno podría dar como resultado la incapacidad de que los usuarios pudieran hacer acceder al sistema. Tener particiones separadas para /etc, /var, y /usr también es una buena práctica.
    - Deshabilitar el reinicio del sistema utilizando las combinación de teclas Control-Alt-Delete, editando el archivo /etc/inittab y comentando la línea:

      ```
      #ca::ctrlaltdel:/sbin/shutdown -t3 -r now
      ```

      Para observar el cambio # /sbin/init q.

- **Gestor de Arranque**

  Se debe evitar que un usuario modifique la secuencia de arranque de forma que permita a un usuario acceder a un prompt de root en un shell de comandos sin preguntar contraseñas u otro tipo de mecanismos de seguridad (23). Sugiriendo incorporar password de acceso al arranque.
    - LILO: Se le puede indicar argumentos a ser considerando en el arranque, siendo el argumento más dañino "imagename single", lo cual arranca Linux en modo de único usuario. Hay que editar el archivo /etc/lilo.conf, donde se indica el tiempo que LILO espera a que el usuario introduzca datos antes de arrancar, se aconseja poner en 0 e Incorporar la línea

restricted para pedir contraseña, solo puede usarse acompañado de la opción password.

```
timeout=00
restricted
password=<password>
```

Luego se debe cambiar los permisos del archivo para que solo lo lea el root, ya que se han escrito password sin encriptar e inmutarlo para evitar futuras modificaciones:

```
chmod 600 /etc/lilo.conf
chattr -i /etc/lilo.conf
```

- GRUB (Grand Unified Bootloader): Crear una password para GRUB, usando el algoritmo de hash MD5 (24).

```
#grub-md5-crypt
```

Editar el archivo /boot/grub/menu.lst o /boot/grub/grub.cfg (ambos son enlaces simbólicos entre ellos) y agregar una nueva línea con el valor retornado del paso anterior. Para probar, rebootear el sistema e intentar ingresar, presionando “p”.

3.3.2. Sistema Operativo

La seguridad del Sistema Operativo es importante porque controla los dispositivos de hardware, sistema de archivos y protocolos de red. Un bug en el kernel puede causar problemas de confidencialidad, integridad y disponibilidad (25). Un ataque al sistema de memoria modifica el sistema de archivos o los paquetes en tránsito por la red, por lo que los bugs de corrupción de memoria afectan a la integridad, permitiendo corromper los datos y metadatos del sistema de archivo o incluso modificar el código del kernel.

Por lo tanto es uno de los elementos más buscados para intentar explotar una vulnerabilidad, por lo cual es necesario:

- Identificar y autenticar a los usuarios.
- Tener un control de acceso a los recursos del sistema.
- Monitorear las acciones realizadas por los usuarios.
- Auditar los eventos para evaluar posibles riesgos.
- Garantizar la integridad de los datos almacenados.
- Garantizar la disponibilidad de los recursos

Se puede segurizar las diferentes funcionalidades de un Sistema Operativo, agrupándolos por:

- *Gestión del Procesador y Memoria*: el SO tiene que asegurar que cada proceso obtiene una parte del tiempo del procesador, y que el procesador es usado eficientemente.
- *Gestión de los Sistemas de Almacenamiento*: el SO define como son almacenados los datos de una manera fiable.
- *Gestión de Entrada/Salida*: el SO debe ser capaz de gestionar la interacción de los componentes de hardware, e interactuar con las aplicaciones y el usuario.
- *Gestión de la Red*: Aunque puede considerarse parte de la entrada y salida, su importancia le permite tomar entidad propia. El Sistema operativo debe aplicar las medidas necesarias para permitir la comunicación en red segura de las distintas aplicaciones.

Gestión del Procesador y de Memoria:

En un sistema multiproceso podemos distinguir dos tipos de protecciones de memoria: La protección de los procesos o tareas entre ellos y la protección del propio sistema operativo (kernel) de los procesos. El kernel no se puede considerar un proceso aislado con su propio espacio de memoria, ya que se debe permitir la ejecución de determinadas partes del kernel por parte de cada proceso, manteniendo siempre el espacio de memoria del proceso por lo cual tiene acceso a instrucciones privilegiadas, y por consiguiente debe proteger su espacio de memoria y establecer puntos de entrada determinados. Para realizar esta protección se debe recurrir a los niveles de privilegio de ejecución, habitualmente existen dos niveles de ejecución: Nivel de kernel y nivel de usuario. El nivel de kernel tiene ciertos privilegios, como la ejecución de instrucciones privilegiadas, acceso a los registros, acceso a la tabla de traducciones, etc. (26).

Cuando un bug del kernel permite al atacante acceder a una pieza de información a la cual no debería tener acceso, es un problema potencial de seguridad. La información puede ser almacenada desde los registros del kernel al stack del kernel, del heap del kernel a las caches relacionadas con el sistema de archivos, o en los buffers de red. Si ocurre un ataque a través del acceso de escritura a la memoria del kernel se puede violar la confidencialidad comprometiendo la integridad.

Debido a la memoria del kernel se expone a problemas de bloqueo como los deadlocks o livelocks. Esto lleva a ataques de denegación de servicios, llevando a la no disponibilidad del servicio.

El primer paso para agregar un mecanismo de defensa al kernel es conocer los actores del sistema.

A continuación se mencionarán distintas configuraciones en el sistema operativo para mitigar posibles ataques:

Memoria Compartida.: El dispositivo de memoria compartida /run/shm por defecto se carga con permisos de lectura, escritura y ejecución de programas. Muchos exploits se ejecutan a través de aplicaciones web inseguras aprovechando esta configuración y atacan los servicios en ejecución. Una solución es modificar la forma de montar la memoria compartida restringiendo los permisos a lectura/escritura, usando la siguiente directiva en el archivo /etc/fstab:

```
none        /run/shm tmpfs      rw, noexec , nosuid, nodev    0           0
```

Información del sistema: Por defecto Linux en el momento de realizar el logueo de un usuario muestra información: nombre de la distribución de Linux, versión del kernel y nombre del servidor, para evitar esto hay que editar el archivo /etc/rc.d/rc.local y comentar las siguientes líneas:

```
#echo "" > /etc/issue
#echo "$R" >> /etc/issue
#echo "Kernel $(uname -r) on $a $(uname -m)" >> /etc/issue
#cp -f /etc/issue /etc/issue.net
#echo >> /etc/issue
```

Luego se eliminan los archivos “/etc/issue.net” y “/etc/issue”. Estos archivos tienen el banner del login del usuario en una conexión remota, al borrarlo se recrea ante una nueva conexión.

Gestión de los Sistemas de Almacenamiento

Hay que tener muy presente la seguridad relacionada con los archivos del sistema operativo. Si estos archivos son accedidos o modificados pueden causar daños al correcto funcionamiento del sistema, incluso dejarlo inoperativo. A continuación se listarán los principales archivos a tener en cuenta en relación a la seguridad:

- El archivo de contraseñas **/etc/passwd** es el archivo más crítico en Linux. Contiene el mapa de nombres de usuarios, identificaciones de usuarios y la ID del grupo primario al que pertenece cada persona. Este archivo es legible y puede ser accedido por distintas aplicaciones.

- Las contraseñas deberían estar cifradas y por lo tanto almacenadas en el archivo /etc/shadow. Las contraseñas se guardan utilizando un hash de un sólo sentido. Las contraseñas no pueden obtenerse a partir de la forma cifrada, sin embargo, se puede tratar de encontrar una contraseña utilizando fuerza bruta para pasar por el hash cadenas de texto y compararlas, una vez que encuentres una que coincide, sabes que has conseguido la contraseña. Este archivo debería estar protegido, y sólo el root debería tener acceso de lectura al mismo.

- El archivo */etc/group* contiene toda la información de pertenencia a grupos, y opcionalmente elementos como la contraseña del grupo (generalmente almacenado en gshadow en los sistemas actuales), es de acceso público y su formato es:

```
nombregrupo:contraseña_cifrada:GID:miembro1,miembro2,miembro3
```

- El archivo */etc/gshadow* contiene los grupos, contraseñas y miembros. También debería estar protegido y sólo el usuario root debería tener permiso de lectura al mismo.

- El archivo */etc/login.defs* permite definir valores por defecto para diferentes programas como useradd y expiración de contraseñas.

- El archivo de shells (*/etc/shells*) contiene una lista de shells válidos. Para que un usuario realice login exitosamente, su shell debe estar en este archivo, por lo cual se debe editar esta lista para solo permitir login a los usuarios validos.

- El archivo */etc/securetty* contiene una lista de tty's desde donde el usuario root puede hacer un login. Los tty's de la consola suelen ir de /dev/tty1 a /dev/tty6. Los puertos serie son /dev/ttyS0 y superiores. Generalmente, sólo se debería permitir conectar al root desde /dev/tty1, y es aconsejable deshabilitar la cuenta de root.

- El archivo */etc/host.conf* es utilizado para obtener la dirección IP a partir del nombre de host.

- El archivo */etc/services* mantiene una lista de los servicios más comunes asociados a un puerto y habilita a los programas clientes y servidores a convertir el nombre del servicio al puerto. Solo el usuario root debería cambiar este archivo.

- El archivo de cada usuario *~/.bash_history* almacena los 500 últimos comandos tipeados. Si se reduce la cantidad de comandos se puede mantener protegido a los usuarios en el servidor, por ej. ingresar la password en la Shell y este valor se almacene por mucho tiempo.

Algunos consejos para brindar mayor seguridad al sistema de archivos:

- Configurar el sistema de archivos correctamente.
- Inmunizar a los archivos de sistemas, es decir solo el usuario root puede modificarlos.

```
chattr +i  archivo
```

- Cifrar toda la partición o una parte del sistema de archivos. La principal desventaja de estos sistemas es que no cifran los metadatos (estructura de directorio, nombres de archivos, tamaños o fechas de modificación). El sistema de archivos encriptado EFS (Encrypting File System) es un ejemplo de este tipo de encriptación.
- Cifrar los datos almacenados en directorio de trabajo. Se puede usar las herramientas: encfs (27), ecrypfs (28).

Gestión de Entrada/Salida

- ***Accesos***
  - Deshabilitar la opción de login a las tty que no se necesite, comentando las tty en el archivo **/**etc/securetty
  - Lograr una conexión segura. Para acceder a la consola de forma remota y mantener seguridad en la conexión se recomienda utilizar un túnel encriptado. Esto se puede lograr utilizando SSH, y denegar el acceso a través del servicio telnet. En la configuración SSH cambiar el puerto estándar, usar versión 2, no permitir usuario root en el login inicial, aumentar el cifrado a 2048 bits, usar intercambio de claves, denegar el acceso por password vacías, determinar tiempo limitado para iniciar sesión y limitar acceso a usuarios:

```
sudo vi /etc/ssh/sshd_config
Port 4242  //puerto habilitado para conexión
Protocol 2
PermitRootLogin no
PermitEmptyPassword no
AllowUsers Adminstrador // solo el usuario Administrador puede loguearse
```

- ***Usuarios***

Para que la autentificación de un usuario funcione correctamente se necesita algún archivo con UID de nombres de usuarios, GID de nombres de grupos,

contraseñas para todos los usuarios. Los programas necesitan acceso al archivo de contraseñas, para lograr almacenar las contraseñas de modo seguro y accesibles se realiza un hash de las contraseñas y se guarda el hash, cuando un usuario necesite autentificar, se valida el hash de la contraseña que introduce con el almacenado. Con la potencia computacional actual y realizando un ataque de fuerza bruta se puede obtener la información buscada en una cantidad de tiempo razonable. Por lo cual se aconseja

- Utilizar un algoritmo de hashing más fuerte como SHA3 o SHA512
- Implementar un esquema de autentificación, para lo cual es necesario añadir un módulo PAM y editar el archivo de configuración de los programas permitiéndole que use ese módulo para hacer la autentificación. El módulo PAM (pluggable Authentication Modules) es un conjunto de librerías para Linux que permiten al administrador del sistema escoger como autenticar a los usuarios de las aplicaciones. PAM introduce una capa de middleware entre la aplicación y el mecanismo real de autentificación. Además se pueden manejar cuentas y datos de sesiones.

Otras sugerencias de configuración para aumentar la seguridad relacionada a los usuarios:

- Seguridad de Contraseñas: Elegir una password correcta. Se debería tener una política para definir password (longitud, caracteres, tiempos de revocación, etc). Se sugiere ejecutar periódicamente un password cracker para evaluar la password. Para definir una longitud mínima de password, editar el archivo /etc/login.defs y modificar la cantidad de caracteres en la línea (23)

```
PASS_MIN_LEN 8
```

- Usar /etc/shadow para almacenar las contraseñas. Inmutar este archivo para prevenir la eliminación, cambios o creación de link simbólico. Deberían inmutarse los archivos /etc/passwd, /etc/shadow, /etc/group y /etc/gshadow.
- Deshabilitar todas las cuentas inactivas.
- Habilitar el comando su solo al grupo admin.

```
groupadd admin
usermod −a −G admin $USER
dpkg−statoverride −−update −−add root admin 4750 / bin/su
```

- Evitar terminales desatendidas. Primero hay que definir el tiempo de inactividad para desloguear automáticamente al usuario del bash. Editar el archivo /etc/profile y agregar las líneas.

```
“HISTFILESIZE=”.
```

```
TMOUT=7200 (tiempo en segundo)
```

Esto corresponde a todos los usuarios, si se desea configurarlo a un usuario en particular editar el .bashrc del usuario (23).

- Reducir el historial del usuario almacenado en el archivo ~/.bash_history . Cambiar las directivas HISTFILESIZE y HISTSIZE en el archivo /etc/profile, que determinan la cantidad de comandos antiguos para todos los usuarios. También se puede indicar que luego de que el usuario se desloguee se borre el archivo .bash_history evitando así que el atacante use información de este archivo, para lograr esto hay que modificar el archivo /etc/skel/.bash_logout agregando la directiva

```
rm -f $HOME/.bash_history.
```

- Se aconseja deshabilitar tantos servicios de inetd.conf como sea posible. Los servicios como systat/netstat y finger proporcionan demasiada información. El acceso a programas iniciados por inetd se puede controlar con facilidad mediante el uso de TCP_WRAPPERS. Otra medida de seguridad es inmutar el archivo para que solo el usuario root puede modificarlo.

```
chattr +i /etc/inetd.conf
```

- **Mitigar ataques de Denegación de Servicios (DoS)**. Asignando recursos limitados a cada usuario del sistema, configurando el archivo /etc/security/limits.conf, este archivo puede ser usado para controlar y limitar el uso de recursos por los usuarios del sistema.

```
* hard core 0
* hard rss 5000
* hard nproc 20
```

Se restringe la creación de archivos raíz, la cantidad máxima de procesos a 20 y el uso de memoria a 5MB (rss 5000) para cada usuario excepto el usuario root, el uso de * indica todos los usuarios que se loguen en el servidor.

3.3.3. Gestión de Red

Los ataques que afectan a la red, están enfocados en el protocolo TCP/IP, por lo tanto, se mencionarán consejos aplicables en el Sistema Operativo relacionado con la red.

- **Deshabilitar el routeo del sistema**. Los protocolos de routeo pueden causar problemas, ya que si un atacante pudo enviar un paquete enrutado de origen en la red, entonces él sería capaz de interceptar las respuestas y engañar a su anfitrión para que piense que se está comunicando con un host de confianza, por lo tanto se recomienda desactivar el enrutamiento de origen IP, editando el archivo /etc/sysctl.conf e incorporando las siguientes líneas (23):

```
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0
```

- **Mitigar ataques de DoS.** Existen distintas formas de llevar a cabo este ataque. Un ataque de denegación de servicios (DoS) consume todos los recursos de su equipo incapacitándolo de responder a conexiones legítimas debido al gran volumen de tráfico haciendo necesario reiniciar el servidor para que vuelva a funcionar con normalidad (23) (29) (30). Alguno de los ataques DoS conocidos son:
    - Un ataque TCP SYN. Para habilitar la protección TCP SYN editar el archivo /etc/sysctl.conf e incorporar las siguientes líneas:

```
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_syn_backlog = 2048
net.ipv4.tcp_synack_retries = 2
net.ipv4.tcp_syn_retries = 5
```

    - Congestión de red a través de requerimientos de broadcast: Cuando se produce un mensaje desde el servidor hacia una dirección broadcast, este mensaje será distribuido a todos los equipos de la red, quienes responderán generando una congestión de red (un ataque de DoS). Para evitar este tipo de ataques, se edita el archivo /etc/sysctl.conf y se incorpora la siguiente línea:

```
net.ipv4.icmp_echo_ignore_broadcasts = 1
```

- **Deshabilitar Aceptaciones de redirecciones ICMP:** El router informa al host una ruta correcta usando un mensaje redirect ICMP. Un atacante puede forzar paquetes de redirección ICMP para alterar las tablas de routeo de un host, alterando la seguridad del mismo, y dando la posibilidad de desviar el flujo de datos, por lo cual se aconseja desactivar la opción de aceptar redicciones ICMP. Editar el archivo /etc/sysctl.conf e incorporar las siguientes líneas (23):

```
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0
```

- **Ignorar las peticiones de repuesta a ping:** Esta configuración permite ignorar las peticiones de respuestas ping, evitando que el atacante conozca si el sistema está operativo (29) (23). Editar el archivo /etc/sysctl.conf e incorporar la siguiente línea:

```
net.ipv4.icmp_echo_ignore_all=1
```

- **Habilitar protección de desfragmentación**. En caso de usar el servidor como Gateway para enmascarar el tráfico a internet (IP masquerading) se recomienda habilitar esta protección e incorporar la siguiente línea al archivo /etc/sysctl.conf (23):

```
net.ipv4.ip_always_defrag = 1
```

- **Habilitar protección de mensajes de error en la red.** Ante errores en la red, para recibir alertas, se tiene que incorporar la siguiente en el archivo /etc/sysctl.conf:

```
net.ipv4.icmp_ignore_bogus_error_responses = 1
```

- **Prevenir ataques IP Spoofing**: IP Spoofing es un exploit que engaña la relación de confianza, un equipo se setea como observador legitimo indicando que todas las actividades de la red son legítimas. Editar el archivo /etc/host.conf

```
order bind , hosts
multi on
nospoof on
```

La opción **order** indica el orden de los servicios, se recomienda que primero revise el servidor (bind) y luego el archivo hosts para mejor performance y seguridad en todos los servidores. La opción **multi** determina que el archivo puede tener múltiples direcciones IP (múltiples interfaces). La opción **nospoof** indica que el equipo no permite spoofing.

Se puede habilitar la protección Spoofing para prevenir la suplantación de la red o envíos de paquetes en un ataques DoS, evitando que el servidor no sea utilizado para él envió de paquetes de direcciones IP no válidas (29). Esta configuración no debe aplicarse en equipos que funcionen como routers o firewall. Editar el archivo /etc/sysctl.conf y añadir las líneas.

```
net.ipv4.conf.all.rp_filter = 2
net.ipv4.conf.default.rp_filter = 2
```

El valor de los parámetros puede ser 0 (valor por omisión, no realizar ninguna comprobación), 1 (rechazar únicamente las suplantaciones evidentes) y 2 (realizar una comprobación exhaustiva). Se aconseja seleccionar la opción de comprobación exhaustiva.

- **Registro de actividades sospechosas:** permite registrar en los archivos de actividad del sistema aquellas situaciones potencialmente sospechosas: intento de envío de paquetes con dirección no válida (spoofed packets), paquetes con cambio de rutas (source routed packets) y otras situaciones similares (redirect packets) (23) (29). Para activar el registro de esta actividad, incorporar las líneas al archivo /etc/sysctl.conf:

```
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.default.log_martians = 1
```

Comunicación Segura

La encriptación de la comunicación es la técnica utilizada para tener procesos de comunicación seguros de cada servicio (web, FTP, acceso remoto).

Como se mencionó en la sección 2.4.4, para llevar a cabo este proceso de encriptación se utilizará encriptación asimétrica e intercambio de claves usando Diffie-Hellman. Para lo cual es necesaria la creación de Certificados Digitales para cada servicio. Se puede adquirir los certificados a través de un proveedor autorizado denominado Autoridad Certificante, por ejemplo: Verisign, Thawte, Comodo, etc., y luego se instalan en el servidor correspondiente. Ó disponer de una Autoridad Certificadora local a partir de la cual se generará un certificado raíz y un certificado por cada servicio requerido el cual será firmado por la CA local utilizando su certificado raíz. Estos certificados deben tener el mismo nivel de encriptación utilizado por los proveedores autorizados. Esta última opción será la implementada en este proyecto.

Los cifrados que se realizarán son:

1. Cifrado de datos para acceso remoto: utilizando los comandos SSH y SCP.
2. Cifrado en la comunicación web: Con HTTPS, utilizando certificados digitales.
3. Cifrado de datos en transferencia FTP: con FTPS, utilizando certificados digitales.

Configuración de SSH

Al querer comunicarse con el servidor de forma remota, existen distintas aplicaciones que utilizan SSH para realizar una Shell remoto, donde se crea un túnel cifrado entre el cliente y el servidor, permitiendo que la comunicación o transmisión de archivos se realice de forma segura.

SSH esta disponible en la mayoría de los Sistemas Operativos, pero en caso de querer usar los comandos SSH y SCP y no tener SSH instalado, habrá que instalar en el servidor la aplicación sshserver y en el cliente sshclient, luego por consola se ejecuta el comando correspondiente:

```
# ssh user@ip
# scp archivo_a_copiar user@ip:/home/user
```

Se puede utilizar en el cliente aplicaciones gráficas que son clientes SSH, ya sea para conectarse al servidor como para copiar archivos en él. Por ej. Putty, winscp, mobaxterm, entre otras.

Configurar la Autoridad Certificante y crear el certificado raíz

En esta sección se explicará cómo configurar una Autoridad Certificante local y crear el certificado raíz.
Una vez creado el certificado raíz, se podrán crear los certificados para cada servicio o requerimiento en específico y podrán ser firmados con el certificado raíz dado por la autoridad certificante local (31).
Se utilizará el software "openssl" para la creación de los diferentes certificados.

1. El comando para instalar este software es:

```
apt-get install openssl
```

2. Luego se sugiere crear el directorio *CA* que va a contener los certificados de nuestra entidad certificadora local "privada":

```
cd /etc/ssl
mkdir CA
chmod 0700 /etc/ssl/CA
```

3. Crear dos subdirectorios: certs y private

```
cd /etc/ssl/CA
mkdir certs private
```

4. Crear la base de datos para mantener un seguimiento de la firma de cada certificado.

```
echo '100001' >serial
touch certindex.txt
```

5. Customizar el archivo de configuración openssl.cnf para indicar directorio donde se almacenarán los certificados

6. Crear el certificado raíz (cacert.pem) y la clave privada (cakey.pem) de la *Autoridad Certificadora* "*privada*". Se utilizará el algoritmo de encriptación RSA con un tamaño de clave de 2048 bits.

```
cd /etc/ssl/CA
openssl req –new -x509 –extensions v3_ca -newkey rsa:2048 -days 3650 -keyout cakey.pem -out cacert.pem
```

Encriptar la comunicación web (HTTPS)

Para una comunicación segura entre el sitio web y el usuario final, es necesario que la comunicación se encuentre encriptada. Para lograr esto, se creará un certificado digital para el servidor web, el cual será firmado por el certificado raíz otorgado por la Autoridad Certificante local privada creado en el paso anterior (32).
También es necesario instalar el modulo SSL en Apache y configurar los VirtualHost de Apache para que escuche en el puerto 443 correspondiente al protocolo HTTPS (33).

1. Crear el certificado para el servidor web.

Durante la creación se solicitará información sobre el dominio, la parte critica corresponde a "Common Name", que debe ingresarse el nombre del servidor, por ej. web.your_domain o la dirección IP. Si se cubre distintos subdominios pueden ingresarse *.your.domain. en "Organizational Unit" se puede indicar el uso del certificado indicando "web server"

```
openssl req -new -nodes -out serverweb-req.pem -keyout private/serverweb-key.pem
```

2. Firmar el certificado con la CA "privada".

Con el siguiente comando se firmará el certificado del servidor web (serverweb-req.pem). Es necesario conocer la contraseña de la clave privada de la CA y tener el certificado del servidor web creado previamente. Se crearán dos archivos, el certificado para el servidor web que será exportado y una copia del certificado almacenado en la carpeta newscerts

```
openssl ca –out serverweb-cert.pem -infiles serverweb-req.pem
```

3. Instalar el certificado y la clave privada en el servidor web.
Crear la carpeta */etc/apache2/ssl* y copiar la clave pública (serverweb-key.pem) y el certificado firmado (serverweb-cert.pem). Tener en cuenta la seguridad de la carpeta donde se almacenaran los certificados.

```
sudo mkdir /etc/apache2/ssl
sudo cp serverweb-key.pem serverweb-cert.pem /etc/apache2/ssl
```

4. Instalar el modulo SSL en Apache:

```
sudo a2enmod ssl
```

5. Revisar el archivo /etc/apache2/ports.conf y permitir trabajar con el puerto 443, descomentando las siguientes líneas:

```
<IfModule mod_ssl.c>
        Listen 443
</IfModule>
<IfModule mod_gnutls.c>
        Listen 443
</IfModule>
```

6. Editar el archivo correspondiente al virtualhost *"/etc/apache2/sites-available/default-ssl", agregando el apartado "configuración ssl"*. A modo de ejemplo.

```
<VirtualHost 192.168.1.1:443>
    DocumentRoot /var/www/html
    ServerName 192.168.1.98
    ServerAdmin someone@your.domain
    ErrorLog /etc/httpd/logs/ssl_error_log
    TransferLog /etc/httpd/logs/ssl_access_log
    SSLEngine On
    SSLCertificateFile /etc/httpd/conf/ssl.crt/name-cert.pem
    SSLCertificateKeyFile /etc/httpd/conf/ssl.key/name-key.pem
   <Files ~ "\.(cgi|shtml|php)$">
     SSLOptions +StdEnvVars
   </Files>
   <Directory "/var/www/cgi-bin">
     SSLOptions +StdEnvVars
   </Directory>
   SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown
   CustomLog /etc/httpd/logs/ssl_request_log \
     "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
</VirtualHost>
```

7. Activar la configuración del sitio con SSL y reiniciar el servidor:

```
a2ensite default-ssl
```

Recordar: como la Autoridad Certificadora es local, los navegadores no reconocen el certificado (Figura 28). Para solucionar el aviso de navegación insegura, se puede importar el certificado correspondiente a la AC en el navegador. Para esto se obtiene el certificado serverweb-cert.pem y se importa en el navegador web.

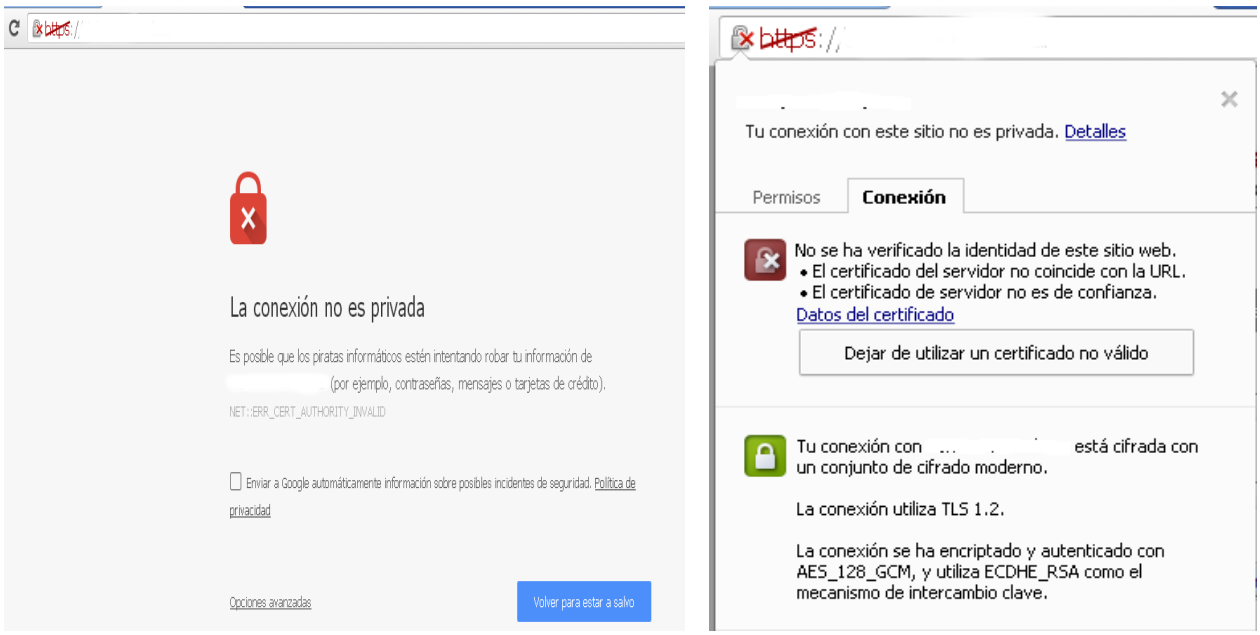


Figura 28. Conexión HTTPS con certificado creado por una AC local privada

3.3.4. Servicios

El aseguramiento de los servicios que presta una organización es una tarea que debe ser realizada con sumo cuidado configurándolos de manera correcta y llevando un seguimiento periódico de los archivos de registro (logs) que cada servicio genera.

Es importante evaluar quien accede al servicio, la aplicación que se utiliza para conectarse al equipo, que usuario tiene permitido operar con cada servicio, que permisos tiene cada usuario, cuando se ejecuta cada servicio, entre otros. Hay muchos servicios que utilizan el protocolo SSL para la comunicación segura, por ejemplo: HTTPS, SMTPS, IMAPS, SSH, POP3S, etc.

Se debe fijar los permisos para ejecutar o detener los procesos en el momento de arranque del sistema (23). Solo el usuario root debería tener permitido leer, escribir y ejecutar archivos dentro del directorio.

```
# chmod -R 700 /etc/rc.d/init.d/*
```

Servidor web

En esta sección se brindarán sugerencias y se explicará la configuración apropiada para brindar seguridad y privacidad en el servidor web Apache.

La configuración por defecto del servidor web apache contiene mucha información sensitiva, la cual puede ser muy útil a un atacante siendo susceptible a condiciones que permiten a atacantes ganar control inapropiado del servidor, por ejemplo ganar acceso para modificar archivos del sistema con acceso restringido o ejecutar comandos dentro del servidor. Por lo cual es necesario conocer qué medidas necesarias para mitigar estos ataques brindando una capa extra de seguridad al servidor web.

La seguridad consiste en permitir a las personas ver lo que ellas pueden ver y prevenir que vean lo que no pueden ver. Se deben tener precauciones para proteger el servidor de accesos maliciosos y modificaciones al sitio web (33), una técnica utilizada es el ocultamiento de la información, evitando que los atacantes conozcan información de la configuración y/o software utilizado en el servidor web.

En la seguridad de un servidor existen dos pasos a tenerse en cuenta: la autenticación y la autorización, aunque son actividades separadas, se confunden un poco en el contexto de los módulos del servidor web apache. Aunque la principal diferencia entre la mayoría de los módulos de seguridad es como almacenan las credenciales, todos proveen funcionalidad para validar si un usuario esta autenticado y autorizado para acceder a un recurso. Algunos recursos del servidor web requieren acceso restringido, permitiendo su acceso a través de la validación por contraseñas, pero no siempre es así, siendo esta una característica considerada en los ataques.

Se mencionan algunas soluciones para mitigar las vulnerabilidades web más típicas:

- **Evitar banner**. Por defecto el servidor web al generar una respuesta en los header muestra información de la versión de php, apache y el sistema operativo (35). Para no exponer la versión del software correspondiente al servidor web que se esté usando, se deben remover los banner de versión. Modificar el archivo de configuración de apache /etc/apache/apache2.conf (httpd.conf), incorporando las siguientes directivas y luego reiniciar apache2 con force-reload para que actualice la nueva configuración (2):

```
ServerTokens ProductOnly

ServerSignature Off
```

  La directiva ServerTokens (36) (37) controla como el servidor responde en el campo header, si incluye o no la descripción del sistema operativo así como información de

los módulos compilados. Se aplica a la configuración de mod_security de apache. Los posibles valores son: Major|Minor|Min[imal] | Prod[uctOnly] |OS|Full

Major → Server: Apache/2
Minor → Server: Apache/2.0
Minimo → Server: Apache/2.0.55
OS → Server: Apache/2 (Debian)
Prod → Server: Apache
Full → valor por defecto que muestra todo: Server: Apache/2.0.55 (Debian) PHP/5.1.2-1+b1 mod_ssl/2.0.55

Se puede cambiar "Apache" por otro nombre indicado en la directiva "SecServerSignature", previo se debería setear ServerTokens en Full, en la configuración de Mod_Security del servidor Apache.

```
SecServerSignature YourServerName
```

La directiva ServerSignature permite agregar un pie de página a los documentos generados por el servidor (mensajes de error, etc.). Por defecto es off, pero puede ser sobrescrita a nivel de VHOST

```
ServerSignature On|Off|Email
```

- **Deshabilitar listado de directorios.** Un usuario puede listar los directorios utilizando el browser web, permitiendo conocer y/o descargar archivos privados (2). Es importante restringir a los usuarios solo a las paginas establecidas. En el archivo /etc/apache2/apache2.conf en el directorio correspondiente cambiar la directiva Options a **None** o **–Indexes** , por ejemplo:

```
<Directory /opt/apache/htdocs>
Options None
Order allow, deny
Allow from all
</Directory>
```

- **Header Etag**. Es un mecanismo de validación HTTP que usa la cache web para almacenar un identificador único de cada recurso, con esto el servidor puede comparar si se ha realizado algún cambio en el recurso comparando los etags. Esta vulnerabilidad permite que atacantes remotos obtengan información sensitiva, como número de inodo, límites de multipart MIME y procesos hijos (2), ha sido registrada en CVE-2003-1418 (38). Para prevenir esta vulnerabilidad se agrega la siguiente directiva en el archivo de configuración de apache para deshabilitar la generación de nodos etag.

```
FileETag None
```

- Ejecutar Apache utilizando un usuario y grupo específico (35).

- Deshabilitar módulos innecesarios. Se deberían eliminar todos los módulos que no se necesiten para lograr reducir la exposición potencial de fallas de seguridad (33) (35).

- **Proteger la configuración del sistema**: Restringir el acceso a archivos fuera del directorio raíz. Con la directiva ***AllowOverride*** se previene el uso de .htaccess en todos los directorios. Este archivo está en cada subdirectorio y un usuario puede sobrescribir las directivas de apache. Evaluar las opciones necesarias para el directorio raíz y habilitar o deshabilitar las mismas. Nunca debería utilizarse la directiva ***Options All*****.** Se deniega acceso a todo y luego se permite acceso al directorio especifico con la directiva <directory path>. Modificar el archivo apache2.conf

```
<Directory />
  Order deny, allow
   Deny from all
  AllowOverride None
  Options none
</Directory>
```

- **Configurar correctamente los permisos de los archivos.** El directorio bin del servidor debería ser solo propietario el usuario root, grupo root y tener permisos de 755 (rwxr-xr-x). Los directorios de documentos como htdocs, cgi-bin e icons deberían tener permisos de ejecución por parte del servidor pero en ningún caso pueden ser modificables por el usuario del sitio. El directorio conf debe tener permiso de lectura y escritura por el root, los directorios include y libexec debería ser leído por cualquiera pero no puede ser escrito. El propietario del directorio logs debe ser el usuario root y debe tener permiso de escritura para ese usuario (33). Es buena práctica crear un grupo e incorporar los usuarios autorizados permitiendo ver o modificar los mismos (35).

```
mkdir /usr/local/apache2
cd /usr/local/apache2
mkdir bin logs conf
groupadd apacheadmin
chown -R root:apacheadmin bin logs conf
chmod -R 770 bin logs conf
nano /etc/group
apacheadmin:x:1121:admin
```

- Restringir el acceso a una red o IP específica a un determinado sitio. Con la directiva ***Allow o deny from*** se puede permitir o denegar el acceso a un sitio desde una IP (por ej. Allow from 10.10.1.21) ó desde una red (por ej. Allow from 10.10.0.0/24) (35)

- El protocolo HTTP 1.1 soporta muchos métodos de requerimientos que pueden llevar a un riesgo potencial (2). Típicamente se necesitan los métodos request: GET, HEAD, POST en una aplicación web. Se pueden limitar los requerimientos editando el archivo de configuración de apache, incorporando las siguientes directivas:

```
<LimitExcept GET POST HEAD>
                deny from all
</LimitExcept>
```

- Requerir autenticación fuerte y débil, para algún recurso en particular (33). Se puede asegurar acceso a algún recurso a través de autenticación. Usando la directiva **satisfy** para requerir ambos tipos de autenticación. Por ejemplo solo acceso a usuarios del grupo ventas conectados desde algún equipo de la red 192.168.1.

```
# Hacer cumplir todas las restricciones
Satisfy All
# Requerir password
AuthType Basic
AuthName Sensitive
AuthUserFile /www/passwords/users
AuthGroupFile /www/passwords/groups
Require group ventas
# Requerir acceso desde cierta red
Order deny,allow
Deny from all
Allow from 192.168.1
```

- Restringir el acceso a ciertas URLs a traves de un servidor proxy (33). Se puede bloquear alguna URL dada por la concordancia de una expresión regular.

```
<Directory proxy:*>
RewriteEngine On
# Deshabilitar el acceso a archivos de películas rm o ra
RewriteRule "\.(rm|ra)$" "-" [F,NC]
```

- **Mitigar Ataques Server Side Include (SSI)** deshabilitando el módulo include el cual permite la explotación de aplicaciones web a través de inyección de scripts dentro de las páginas HTML ó ejecutando código remoto, llevando así a incrementar el riesgo de sobrecargar al servidor, esto se agrava si existe un entorno compartido con mucho tráfico de las aplicaciones web. Se puede deshabilitar agregando la directiva “includes” en Options del archivo de configuración de apache (2).

```
<Directory /opt/apache/htdocs>
Options –Indexes -Includes
Order allow,deny
Allow from all
</Directory>
```

También se aconseja remover los módulos objetos dinámicos compartidos (DSO) sin utilizar. Estos módulos están presentes en la directiva LoadModule del archivo de configuración de apache.

- **Prevenir ataques Cross Site Tracing (CST).** Deshabilitar el seguimiento de requerimientos HTTP. Por defecto está habilitado en la configuración de apache los métodos de seguimiento, lo que permite un potencial ataque Cross Site Tracing y así el atacante obtiene información de las cookies de navegación. Para deshabilitar incorporar la siguiente directiva en el archivo de configuración de apache (2).

```
TraceEnable off
```

- **Mitigar el ataque Cross Site Scripting (CSS)**. A través de la utilización de los flag HttpOnly y Secure en las cookies, caso contrario podría manipularse las sesiones u cookies de la aplicación web siendo esto muy peligroso. Asegurar que el modulo mod_headers.so está habilitado en el archivo de configuración de apache (a2enmod headers) e incorporar la siguiente directiva.

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

Se puede forzar esta protección incorporando una directiva y evitando que sea bypaseada por el usuario en el browser (2).

```
Header set X-XSS-Protection “1; mode=block”
```

- **Para evitar el ataque Clickjacking**. La opción X-Frame-Options en los headers de respuesta HTTP se usan para indicar si un browser permite o no abrir una página en un frame o un iframe. Para prevenir que el contenido de un sitio esté embebido en otro hay que tener habilitado el modulo mod_headers.so e incorporar la siguiente directiva al archivo de configuración de apache (39):

```
Header always append X-Frame-Options SAMEORIGIN
```

Sameorigin: permite que una página sea desplegada en un frame del mismo origen que la página en sí.
Deny: previene que la página despliegue un frame o iframe
Allow-from uri: permite que la pagina despliegue solo en el origen específico.

- **Prevenir ataques de sesión hijacking**. Deshabilitar el protocolo HTTP1.0 ya que tiene debilidades de seguridad relacionadas con los ataques de sesión hijacking, por lo cual se sugiere deshabilitarlos usando el módulo mod_rewrite, habilitando la directiva ***RewriteEngine*** y agregando en la condición Rewrite que permita solo HTTP 1.1, en el archivo de configuración de apache.

```
RewriteEngine On
RewriteCond %{THE_REQUEST} !HTTP/1.1$
RewriteRule .* - [F]
```

Restringir imágenes a ser usadas fuera del sitio: otros sitios pueden estar enlazados a imágenes de nuestro sitio robando así el ancho de banda. Se puede asegurar el acceso a las imágenes agregando las siguientes líneas al archivo .htaccess en el directorio donde están las imágenes o en el archivo apache2.conf. Incluso se puede cambiar la imagen cuando se quiera acceder fuera del sitio (33). Esta solución causa que todos los requerimientos a imágenes sean rechazados devolviendo el error 403.

```
<FilesMatch "\.(jpg|jpeg|gif|png)$">
        SetEnvIfNoCase Referer "^http://([^/]*\.)?myserver.com/" local_referrer=1
        Order Allow,Deny
        Allow from env=local_referrer
</FilesMatch>
```

- **Para mitigar ataques SlowLoris**, se suelen usan distintos módulos de apache: mod_limitipconn, mod_qos, mod_evasive, mod_security, mod_noloris, y mod_antiloris. Otra configuración de seguridad consiste en disminuir el tiempo que espera el servidor por cierto tipo de eventos antes de responder un requerimiento.

```
Timeout 60
```

- **Prevenir ataques de fuerza Bruta**. Se podría deshabilitar a los usuarios que realicen repetidos intentos de autenticación fallidos o si intenten crackear la password. Los módulos de autenticación de apache no evitan esto por lo cual el recurso usual es realizar monitoreo sobre los archivos de logs o se puede usar alguna herramienta como mod_security para indicar cuando se esta produciendo un ataque. Ingresar las siguientes líneas en el archivo apache2.conf.

- **Proteger los archivos del servidor de la ejecución de Script Maliciosos** en el servidor web. Los script podrían acceder a modificar y destruir archivos localizados en el servidor web sino han sido adecuadamente protegidos. Para que esto no suceda se aconseja que los archivos no sean escribibles por usuarios nobody o del grupo nobody y que los archivos confidenciales no sean leídos por estos usuarios.

Base de Datos

Las bases de datos son componentes esenciales de cualquier aplicación web, permitiendo a los sitios web proveer variedad de contenido dinámico. Es prioritario considerar la protección de las bases de datos para proteger la información sensible almacenada (40).

Muchos desarrolladores web no son conscientes de cómo las consultas SQL pueden ser manipuladas, y asumen que una consulta SQL es una orden fiable. Esto significa que las consultas SQL son capaces de eludir controles de acceso, evitando así las comprobaciones de autenticación y autorización estándar, e incluso algunas veces, las consultas SQL podrían permitir el acceso a comandos a nivel del sistema operativo del servidor.
La inyección directa de comandos SQL es una técnica donde un atacante crea o altera comandos SQL existentes para exponer datos ocultos, sobrescribir los valiosos, o peor aún, ejecutar comandos peligrosos a nivel de sistema en el equipo que hospeda la base de datos. Esto se logra a través de la práctica de tomar la entrada del usuario y combinarla con parámetros estáticos para elaborar una consulta SQL. Debido a la falta de validación en la entrada de datos y a la conexión a la base de datos con privilegios de superusuario. Por ejemplo un atacante podría crear un superusuario en la base de datos a través de una consulta SQL.
Una medida importante de seguridad es verificar la configuración de los motores de bases de datos, para asegurarse que no tienen fallas de seguridad. Esto incluye verificar la forma en que se instaló la base de datos (cambiar configuraciones por defecto), los privilegios de los usuarios de base de datos en el sistema operativo y las bitácoras de transacciones. Además tener actualizados los parches de seguridad del motor que se esté usando y evitar utilizar las base de datos con versiones que incluyen vulnerabilidades conocidas.

Una vez que un atacante obtiene acceso directo a una base de datos (eludiendo el servidor web), los datos sensibles almacenados podrían ser divulgados, mal utilizados o modificados, perdiendo confiabilidad e integridad de los datos. Encriptar los datos es una buena forma de mitigar esta amenaza. Se podría codificar algún método de encriptación, o utilizar los métodos de los lenguajes de programación, por ejemplo: PHP tiene las funciones Mcrypt y Mhash (40).

Además, es necesario evaluar desde que aplicaciones, los usuarios realizan las consultas SQL y aplicar las restricciones necesarias, como:

- Delimitar el acceso, concediendo privilegios mínimos a los usuarios, procedimientos y/o datos.
- Eliminar cuentas del servidor de bases de datos anónimas, con passwords en blanco, que no se utilicen o cuyo tiempo de activación haya expirado.

Aplicar los siguientes consejos de seguridad, independiente del motor de base de datos elegido:

- Cambiar la contraseña del usuario roto por defecto y aplicar una política de cambios de contraseñas regularmente. No elegir claves que puedan aparecer en un diccionario.
- Eliminar la Base de Datos test

- No almacenar ninguna clave sin cifrar en la base de datos. Si alguien tuviera acceso podría obtener la lista completa de claves y utilizarlas. Utilizar funciones hash para encrestar la información.
- Realizar monitoreo continuo sobre las operaciones en la base de datos, a través de los logs del sistema.
- Realizar auditorías regularmente sobre las bases de datos, usuarios y operaciones.
- Para evitar ataques SQL Injection se puede usar mod_Rewrite. Si se detecta expresión regular "drop table" se redirecciona la URL, esto sirve usando el método GET, pero si se usa POST esto no se puede evaluar, por lo cual debería usarse el módulo mod_security que permite detectar y prevenir ataques realizados usando GET o POST (41) e incorporar en la configuración de apache la siguiente directiva:

```
SecFilter drop[[:space:]]table
```

Seguridad en MySQL

MySQL utiliza seguridad basada en Listas de Control de Acceso (ACLs). También tiene soporte para conexiones cifradas mediante SSL entre clientes y servidores.

De ser posible siga estas recomendaciones (42):

- Instalar el servidor MySQL usando el comando **mysql_secure_installation** para realizar una instalación segura.
- Ejecutar MySQL con un usuario sin privilegios. Nunca ejecutar el servidor MySQL con el usuario root de Linux. Esto se logra configurando mysqld para que rechace ejecutarse como root a menos que se utilice explícitamente la opción –user=root. Para ejecutar mysqld mediante un usuario de linux diferente, añada la opción user que especifica el nombre del usuario al grupo [mysqld] del archivo /etc/my.cnf:

```
[mysqld]
user=mysql
```

- Asegúrese de que el único usuario linux con permisos de lectura o escritura en los directorios de la base de datos es el usuario que ejecuta mysqld.
- Solo la cuenta root de MySQL tiene que tener acceso a la tabla user en la base de datos Mysql. Si un atacante obtiene acceso a esta tabla tendrá acceso a las password encriptadas de los usuarios.

- Conceder privilegios mínimos a los usuarios. Revisar el sistema de privilegios de acceso de MySQL. Las sentencias GRANT y REVOKE se utilizan para controlar el acceso a MySQL. Nunca otorgar privilegios a un mismo usuario sin tener en cuenta el equipo desde el que se conecta.
- Utilizar claves para todos los usuarios MySQL.
- Cambiar o filtrar el puerto del motor de base de datos para que no sea visto desde lugares no confiables. En MYSQL el puerto por defecto es 3306.
- Usar conexiones encriptadas, MySQL soporta conexiones SSL internas desde la versión 4.0.0.
- Colocar MySQL tras el firewall o en una zona desmilitarizada (DMZ). Cerrar el puerto a conexiones remotas.
- No permitir el uso de enlaces simbólicos a tablas. Puede desactivarse con la opción --skip-symbolic-links.
- No otorgar los privilegios PROCESS o SUPER a usuarios no-administrativos.
- No otorgar el privilegio FILE a usuarios no-administrativos. Cualquier usuario que posea este privilegio puede escribir un archivo en cualquier de lugar del sistema de archivos con los privilegios del demonio mysqld. Para hacer esto un poco más seguro, los archivos generados con SELECT ... INTO OUTFILE no sobreescriben archivos existentes, y pueden ser escritos por cualquiera. El privilegio FILE puede ser utilizado para leer archivos que sean legible por cualquiera o accesible por el usuario linux que ejecuta el servidor. Esto podría utilizarse, por ejemplo, utilizando LOAD DATA para cargar /etc/passwd en una tabla, que podría ser mostrada después con un SELECT.
- Se podría utilizar números IP's en vez de nombres en las tablas de permisos (tablas grant). En cualquier caso, debería ser muy cuidadoso en crear registros en las tablas de permiso utilizando nombres que contengan caracteres comodines.
- Para restringir el número de conexiones permitidas para una misma cuenta, se puede establecer la variable max_user_connections de mysqld. La sentencia GRANT también soporta opciones de control de recursos para limitar la extensión de uso de servidor permitido a una cuenta.
- El comando LOAD DATA puede cargar un archivo que esté localizado en el equipo servidor, o puede cargar un archivo localizado en el equipo cliente cuando se especifica la palabra clave LOCAL. Hay dos aspectos de seguridad potenciales al soportar la versión LOCAL de los comandos LOAD DATA:

- La transferencia del archivo desde el equipo cliente al equipo servidor se inicia mediante el servidor MySQL. En teoría, puede construirse un servidor modificado de forma que el programa cliente transfiera un archivo elegido por el servidor en lugar del archivo especificado por el cliente en el comando LOAD DATA. Tal servidor podría acceder a cualquier archivo en el equipo cliente al que el usuario cliente tuviese acceso de lectura.
- En un entorno Web los clientes se conectan mediante un servidor Web, un usuario podría usar LOAD DATA LOCAL para leer cualquier archivo al que el servidor Web tuviese acceso de lectura (asumiendo que el usuario puediese ejecutar cualquier comando contra el servidor de base de datos). En este entorno, el cliente MySQL es el servidor Web, no el programa ejecutado. Para desactivar todos los comandos LOAD DATA LOCAL desde el lado del servidor, se inicia mysqld con la opción --local-infile=0.
- Para el cliente de línea de comando mysql, LOAD DATA LOCAL puede activarse especificando la opción --local-infile[=1] , o deshabilitarse con la opción --local-infile=0 . De forma similar, para mysqlimport, las opciones --local o -L permite la carga de datos locales. En cualquier caso, el uso exitoso de una operación de carga local requiere que el servidor lo permita.
- Si LOAD DATA LOCAL infile está desactivado en el servidor, un cliente que trate de ejecutar dicho comando recibe el mensaje de error: “ERROR 1148: The used command is not allowed with this MySQL versión”.

Seguridad en PostgresSQL

La seguridad de la base de datos esta implementada en varios niveles (43):

- Protección de los archivos de la base de datos. Todos los archivos almacenados en la base de datos están protegidos contra escritura por cualquier cuenta que no sea la del superusuario de Postgres.
- Las conexiones de los clientes al servidor de la base de datos están permitidas, por defecto, únicamente mediante sockets Unix locales y no mediante sockets TCP/IP. Se debe ejecutar el demonio con la opción -i para permitir la conexión de clientes no locales.
- Las conexiones de los clientes se pueden restringir por dirección IP y/o por nombre de usuario mediante el archivo pg_hba.conf situado en PG_DATA.

- Las conexiones de los clientes pueden ser autentificadas mediante otros paquetes externos.
- A cada usuario de Postgres se le asigna un nombre de usuario y (opcionalmente) una contraseña. Por defecto, los usuarios no tienen permiso de escritura a bases de datos que no hayan creado.
- Los usuarios pueden ser incluidos en grupos, y el acceso a las tablas puede restringirse en base a esos grupos.

La ***Autentificación de Usuarios*** es el proceso mediante el cual el servidor de la base de datos asegura de que el usuario que está solicitando acceso a la base de datos es en realidad quien dice ser. Todos los usuarios que quieren utilizar Postgres se comprueban en la tabla pg_user para asegurarse que están autorizados a hacerlo. Actualmente, la verificación de la identidad del usuario se realiza de distintas formas:

- Desde la shell del usuario: Un demonio que se lanza desde la shell del usuario anota el id original del usuario antes de realizar un setuid al id del usuario postgres. El id original del usuario se emplea como base para todo tipo de comprobaciones.
- Desde la red: Si Postgres se instala como distribuido, el acceso al puerto TCP del postmaster está disponible para todo el mundo. El Administrador de Bases de Datos debe configurar el archivo pg_hba.conf situado en el directorio PG_DATA especificando el sistema de autentificación a utilizar en base al equipo que realiza la conexión y la base de datos a la que se conecta.

Para **controlar el acceso** Postgres proporciona mecanismos para permitir a los usuarios limitar los accesos.

- Los SuperUsuarios de la base de datos (aquellos que tienen el campo pg_user.usesuper activado) ignoran todos los controles de acceso.
- Privilegios de acceso: El uso de los privilegios de acceso para limitar la lectura, escritura y la puesta de reglas a las clases se trata en grant/revoke.
- Borrado de clases y modificación de estructuras: Los comandos que borran o modifican la estructura de una clase, como alter, drop table, y drop index, solo funcionan con el propietario de la clase.
- Evitar que usuarios ejecuten funciones no autorizadas

```
REVOKE ALL ON FUNCTION f2() FROM user1, GROUP PUBLIC;
```

Una medida importante de seguridad es ***asegurar la cuenta de administrador***. Por defecto después de instalar postgreSQL, la cuenta *"postgres"* no tiene definida ninguna clave de acceso y cualquier usuario que tenga acceso a la máquina que este ejecutando PostgreSQL, podrá acceder a todas las bases de datos como usuario*"postgres"* y tener privilegios de administrador, vía sockets. Esta configuración por defecto puede ser un

gran problema de seguridad en una máquina en la que muchos usuarios tengan acceso y existan bases de datos en producción o con datos confidenciales. Existen diferentes maneras de asegurar la cuenta de administrador en PostgreSQL:

1. Tener una cuenta de sistema (postgres) en el sistema operativo sin clave definida. Habrá que ser root y utilizar su - postgres para convertirse en el usuario *"postgres"*. No será posible acceder directamente al sistema como *"postgres"* via TCP/IP.

```
passwd -d postgres
```

2. El acceso mediante sockets será usado solamente por la cuenta de administrador *"postgres"*. El resto de usuarios utilizarán el protocolo TCP/IP.

3. Cambiar los permisos del socket utilizado por PostgreSQL. Modificar en el archivo postgresql.conf las siguientes lineas:

```
listen_addresses = 'localhost'

unix_socket_permissions = 0700
```

4. Utilizar el método de autentificación *'ident'* para la cuenta *"postgres"del* gestor de base de datos. De esta manera solamente el usuario postgres del sistema operativo podrá acceder a PostgreSQL como el usuario *"postgres"* de la base de datos. Actualizar el archivo pg_ident.conf con la siguiente linea:

```
administrador      postgres      postgres
```

   Y comprobar que la única línea en pg_hba.conf con información sobre el usuario "postgres" es la siguiente:

```
local   all      postgres      ident   administrador
```

Mantenerse informado sobre las vulnerabilidades reportadas y aplicar las actualizaciones correspondientes es una buena práctica de seguridad (44).

Administrador web para Bases de Datos: PHPmyadmin

Una recomendación importante es mantenerse informado sobre las diferentes vulnerabilidades reportadas y en caso necesario aplicar las actualizaciones sugeridas: En phpmyadmin en las versiones 3.3.0 a la 3.4.3.2 se realizó un ataque XSS (CVE-2011-3181) (43)

que afectó la funcionalidad “structure snapshot” al no realizar correctamente la validación de los datos pasados a través de los parámetros: tabla, columna e índice. Un atacante remoto podría ejecutar código JavaScript a través de una URL manipulada, por ej. Cookie de sesión del administrador. La víctima usó como navegador web Internet Explorer (versiones menores o iguales a la 8), para ejecutar código JavaScript a través de la funcionalidad de exportación de seguimiento.

También se han reportado problemas de divulgación de información no autorizada y por lo tanto potencialmente comprometer un equipo afectado, en todos los casos se sugiere actualizar a una versión que tenga corregido las vulnerabilidades que permitieron estas fallas de seguridad.

Para brindar mayor seguridad cuando se utiliza un administrador web de acceso a la Base de Datos, se deben tener en cuenta los siguientes puntos (44):

- Mantener actualizada la versión de phpmyadmin y así evitar la explotación de vulnerabilidades encontradas.
- Autenticación de acceso: configurar el archivo /etc/phmyadmin/config.inc.php descomentando las siguientes líneas:

```
$cfg['Servers'][$i]['auth_type'] = 'http';
```

  Esta línea indica el tipo de autentificación (cookies, http o entrar sin pedir contraseña). Con http o cookie, la información de inicio de sesión ingresada (usuario y password) son usuarios de la base de datos, no del sistema operativo. Si se usa “cookie” el password se almacena encriptado con el algoritmo blowfish, en un archivo temporal.

- Tener en cuenta que los datos ingresados están en texto plano, debería aplicarse HTTPS para encriptar la comunicación en este sitio web.
- Por defecto el usuario root de MySQL se puede conectar a la base de datos, se recomienda agregar otro usuario con algunos privilegios de root y no usar root.
- Cambiar URL de acceso: por default se ingresa con phpmyadmin (www.sitio/phpmyadmin), se recomienda cambiarla para brindar mayor seguridad. Editar el archivo /etc/phpmyadmin/apache2.conf y cambiar donde dice:

```
ALIAS /phpmyadmin /usr/share/phpmyadmin, por

ALIAS /nuevaurl /usr/share/phpmyadmin
```

- Prevenir ataques de fuerza bruta, configurando Apache para que loguee información sobre la autenticación de phpMyAdmin y luego analizando los logs para detectar posibles ataques. Un ejemplo de directiva de logs en Apache podría ser:

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %{userID}n %{userStatus}n"
      pma_combined
```

El análisis de los logs permitirá identificar a través del parámetro userStatus que muestra información del estado del usuario si surgió algún evento de denegación. El parámetro userID muestra el nombre del usuario activo. Los posibles valores de userStatus son:

- mysql-denied: denegación de inicio de sesión por parte de Mysql,
- allow-denied: denegación de acceso por medio de reglas “allow/deny”,
- root-denied: denegación de acceso al usuario root en la configuración,
- empty-denied: denegación de acceso por contraseña vacia.

- Se aconseja proteger la instalación pública de phpMyAdmin contra el acceso de los robots, a través del archivo robots.txt en el directorio raíz de su servidor web o bien limitando el acceso mediante la configuración del mismo. Se pueden agregar algunas reglas al archivo .htaccess para filtrar el acceso según el campo user agent.

Transferencia de archivos (FTP)

Hay distintos puntos a considerar en relación al servicio de Transferencia de Archivos. Por un lado la comunicación segura de los datos y por otro la configuración del servidor para lograr confidencialidad e integridad de los datos almacenados. A continuación se listarán algunas sugerencias de configuración para lograr esto (45):

Impedir que los usuarios naveguen por la estructura de directorio

Uno de los puntos peligrosos en un servidor de transferencia de Archivos es la posibilidad que un usuario pueda salirse de su directorio y navegar por el resto de los directorios. Para que los usuarios queden confinados (enjaulados) en un directorio particular se aplicará el concepto de “Jailing”, configurando los parámetros (45):

- chroot_local_user : YES (habilitará la función de chroot())
- chroot_list_enable: YES
- chroot_list_file: /etc/vsftpd/chroot_list (se establece en un archivo la lista de usuarios que quedarán excluidos de la función chroot())

En la configuración predeterminada los usuarios del sistema pueden acceder fuera de su directorio personal, para evitar o limitar esto, se debe configurar la opción chroot_local_user que habilitara la función chroot() y chroot_list_enable y chroot_list_file para establecer la lista de usuarios que quedaran excluidos de la función chroot(). De esta manera un usuario del sistema que utilice FTP solo podrá acceder a su directorio personal.
Otro punto importante es limitar las acciones de los usuarios dentro de su directorio, el usuario no debería tener posibilidad de acceder a una consola de comandos y tener permisos

de escritura en el directorio raíz de su directorio personal. A partir de la versión 3.0 de vsftpd estas características vienen establecidas.

```
chmod 755 /home/mengano
chown root:root /home/mengano
mkdir /home/mengano/public_html
chown mengano:mengano /home/mengano/public_html
usermod -s /sbin/nologin mengano
```

Limitar el ancho de banda de transferencia

Se puede controlar la tasa de transferencia, en bytes por segundos para usuarios anónimos, cuando el servidor FTP es de acceso público. Para ello se tiene que incorporar la directiva anon_max_rate en el archivo de configuración /etc/vsftpd/vsftpd.conf. En el siguiente ejemplo se limita la tasa de transferencia a 500 Kb por segundo para los usuarios anónimos:

```
anon_max_rate=524288
```

Para limitar el ancho de banda de los usuarios locales al servidor, hay que utilizar la directiva "local_max_rate"

```
local_max_rate=1048576
```

Limitar la cantidad de usuarios en simultáneo.

Para establecer el número máximo de clientes que podrán acceder simultáneamente al servidor FTP, se tiene que incorporar la directiva "max_clients" al achivo de configuración.

```
max_clients=20
```

Limitar el número de conexión por IP en simultáneo

Para establecer el número máximo de conexiones que se pueden realizar desde una misma dirección IP en simultáneo hay que incorporar la directiva "max_per_ip" en el archivo de configuración. Tener en cuenta que algunas redes acceden a través de un servidor intermediario (Proxy) o puerta de enlace y debido a esto podrían quedar bloqueados innecesariamente algunos accesos.

```
max_per_ip=10
```

Transmisión encriptada

Para realizar transferencias de archivos de manera segura se puede utilizar ***SSL en el servicio FTP (FTPS)***. Se aplica una capa SSL/TLS al protocolo estándar FTP para cifrar los canales de control y/o datos.

No debe confundirse con el protocolo seguro de transferencia de archivos SFTP el cual suele ser usado con SSH.

Existen distintos usos de FTP y SSL:

- *AUTH TLS* o FTPS Explícito. El cliente se conecta al puerto 21 del servidor y comienza una sesión FTP sin cifrar de manera tradicional, pero pide que la seguridad TLS sea usada y realiza la negociación apropiada antes de enviar cualquier dato sensible.
- *AUTH* definido en RFC 2228.
- *FTPS Implícito,* es un estilo antiguo, pero todavía ampliamente implementado en el cual el cliente se conecta a un puerto distinto (como por ejemplo el 990), y se realiza una negociación SSL antes de que se envíe cualquier comando FTP.

Para obtener FTPS, se debe crear un certificado digital con OpenSSL para el servidor FTP instalado, en este caso vsftpd.

El certificado será almacenado en el servidor FTP, por ejemplo: /etc/vsftpd.

1. Crear el certificado para ftp "vsftpd.pem" y la llave privada "vsftpd-key.pem" (34).

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/vsftpd/vsftpd.key -out
  /etc/vsftpd/vsftpd.pem
```

2. Configurar vsftpd. Editar el archivo /etc/vsftpd/vsftpd.conf e incorporar las siguientes líneas:

```
ssl_enable=YES
# no permitir a usuarios anonimos
allow_anon_ssl=NO
# forzar a usuarios locales a usar ssl
force_local_data_ssl=YES
force_local_logins_ssl=YES
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO
require_ssl_reuse=NO
rsa_cert_file=/etc/vsftpd/vsftpd.pem
rsa_private_key_file=/etc/vsftpd/vsftpd.key
```

La directiva "force_local_logins_ssl=YES" obliga a que los clientes usen un software cliente FTP que soporte AUTH TLS/SSL, caso contrario dejar el valor "NO" y así el cliente puede elegir entre una conexión segura o no.

El cliente debe elegir FTP implícito sobre TLS o FTP explícito sobre TLS (Figura 29) y luego aparecerá la pantalla de información del certificado asociado a esta comunicación segura (Figura 30)

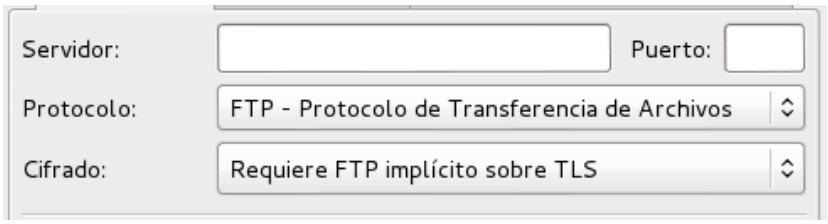


Figura 29. Cliente FTP Filezilla con opción de FTP con TLS Explicito

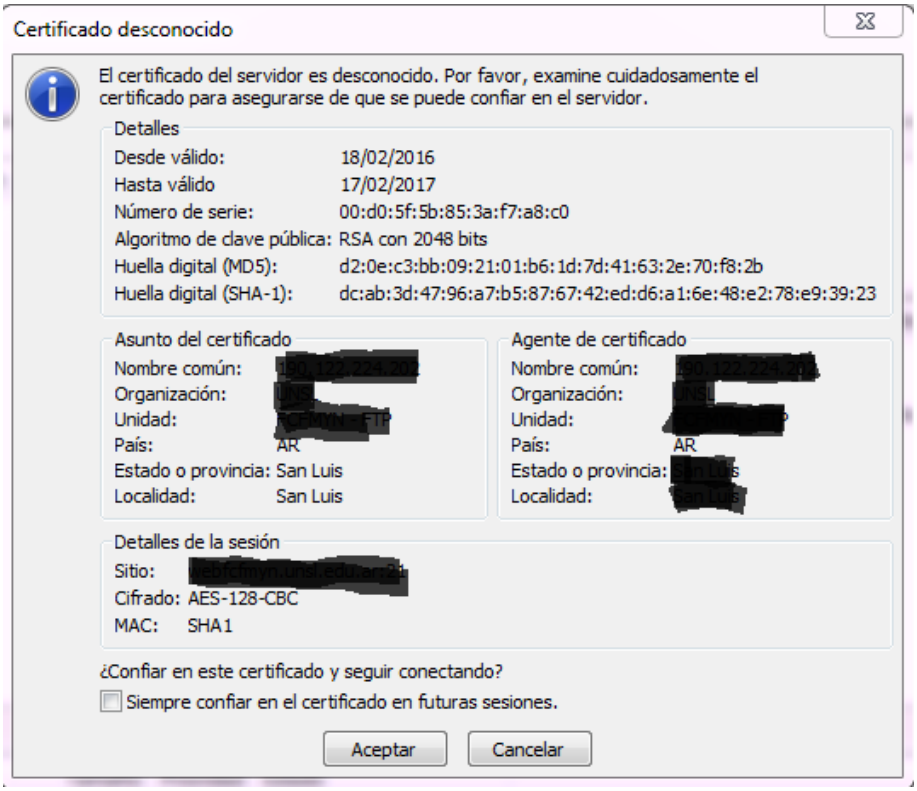


Figura 30. Certificado para la conexión FTP

Firewall

El firewall es el mecanismo básico para permitir o denegar tráfico en la red o en el host. Se implementará firewall de host, aplicando distintas reglas que permitan o denieguen los paquetes que requieren servicios del equipo. Está formado por un conjunto de reglas en las que se examina el origen y destino de los paquetes del protocolo TCP/IP. La implementación se realiza con **iptables** de Linux, definiendo reglas de acceso al host.

Se programó un script de Shell, incluyendo las reglas del firewall. Se lo incluyo en /etc/init.d para que se ejecute siempre que el sistema reinicie.

```
#!/bin/sh
```

```
## Limpiar/borrar las reglas
iptables −F
iptables −X
iptables −Z

#Política determinada
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

#se permite conexión ssh al nuevo puerto
iptables -A INPUT -p tcp --dport 2020 -j ACCEPT

#se permite acceso web
iptables -A INPUT -p tcp --dport 80 -j ACCEPT

#se permite conexión ftp
iptables -A INPUT -p tcp --dport 20:21 -j ACCEPT

# permitir acceso SMTP
iptables -A INPUT -p tcp --dport 25 -j ACCEPT

#bloquear ping menos los mios
iptables -A INPUT -p icmp -s 192.168.10.230 -j ACCEPT
iptables -A INPUT -p icmp -j DROP

# restringir acceso a una ip atacante, logueando la conexion
iptables -i eth2 -A INPUT -s 193.189.116.172 -j LOG --log-prefix "IP DROP SPOOF A:"
iptables -i eth2 -A INPUT -s 193.189.116.172 -j DROP

# resto se cierra
iptables -A INPUT -p tcp --dport 22:23 -j DROP
iptables −A INPUT −p tcp −−dport 3306 −j DROP
iptables -A INPUT -p tcp --dport 6001 -j DROP
```

Con la herramienta iptraf se puede ir depurando y monitoreando el funcionamiento de iptables, ya que permite comprobar si las conexiones TCP/IP se han logrado establecer o no.

Monitoreo

Monitorear un servidor, no solo corresponde a evaluar las actividades que correspondan a actividad inusual del servidor, sino también diagnosticar el correcto funcionamiento del equipo, considerando los servicios de seguridad de disponibilidad e integridad.

Para la disponibilidad del servicio, es preciso considerar el buen funcionamiento del equipo que presta el servicio, por lo tanto es necesario mantener un monitoreo continuo del funcionamiento del mismo: procesador, memoria, disco, red, entre otras.

Una de las consideraciones de seguridad más importante es mantener la integridad de los diferentes archivos de registro de actividad (logs) del sistema o de los servicios que se almacenan en el directorio /var/log. En caso que un atacante logre pasar las diferentes barreras de seguridad, la última línea de defensa es el sistema de logs, por lo tanto es importante considerar un método para lograr la integridad de estos archivos (23). Una buena práctica es habilitar el registro de actividad de cada nuevo servicio, almacenándolos en archivos de logs independientes y monitorearlo continuamente hasta que el servicio funcione con normalidad. Luego puede configurarse para reducir la cantidad de información contemplada, resguardando solo aquellas actividades que representen información valorable del funcionamiento o de la seguridad de los accesos al servicio.

Es útil implementar un monitoreo regular de los archivos de registro de actividad del sistema y de la red (logs), evaluando: tiempo de mensajes, puertos de acceso de ubicaciones externas, eventos inusuales, así también un control sobre los archivos para contemplar posibles cambios generados por accesos malintencionados a los archivos de logs.

Algunos de los archivos que registran actividad del sistema o los servicios:

- /var/log/secure: Registra los accesos al servidor que tienen relación con procesos de autentificación.
- /var/log/auth.log: Permite ver los usuarios conectados de modo remoto.
- /var/log/messages: Registra los mensajes de información o nivel superior. Similar al anterior, pero incluyendo otro género de mensajes.
- /var/log/apache2/error.log. Registra los errores que se han producido al acceder a las aplicaciones web alojadas en el servidor apache.
- /var/log/apache2/access.log. Registra los accesos: quien accede (IP), cuando accede (timestamp), pagina visitada, etc.

Las especificaciones sobre el almacenamiento de estos registros puede indicarse en el archivo /etc/syslog.conf y suele venir acompañado del uso del programa logrotate (/etc/logrotate.conf), que se encarga de comprimir los registros cuando ocupan demasiado espacio o ha pasado bastante tiempo de su creación.

Algunos consejos sobre una metodología de monitoreo.

- Realizar una revisión periódica de los registros de seguridad del servidor dedicado.
- Es recomendable tener copia de los archivos de logs más importantes en otro equipo.
- Tener monitoreo en tiempo real de la actividad de base de datos es clave para limitar su exposición, visualizar cambios o actividad irregular, evaluar uso eficiente de la base de datos, para lo cual se podría aplicar agentes inteligentes de monitoreo para detectar de intrusiones y uso indebido. Por ejemplo, alertas sobre patrones inusuales de acceso, que podrían indicar la presencia de un ataque de inyección SQL, cambios

no autorizados a los datos, cambios en privilegios de las cuentas, y los cambios de configuración que se ejecutan a mediante de comandos SQL.

- Tener un monitoreo sobre la actividad de usuarios privilegiados, ayuda a detectar intrusiones, ya que muchos de los ataques más comunes se hacen con privilegios de usuario de alto nivel.
- Implementar y revisar periódicamente los informes sobre los privilegios de los usuarios, como parte de un proceso formal de auditoría.
- Es útil tener herramientas de monitoreo instaladas como nagios o munin, entre otras que permiten evaluar el funcionamiento de los equipos.
    - Munin es una herramienta de monitorio que permiten analizar los recursos ayudando a evaluar la performance de cada uno de ellos, mostrando resultados en gráficos (46).
    - Nagios tiene distintas herramientas, entre las cuales permite monitorear los componentes de la infraestructura como aplicaciones, servicios, sistema operativos, etc. Se puede configurar un servidor de logs para simplificar el proceso de búsqueda de datos en los logs, incorporando alertas para detectar posibles amenanzas (47).
- En caso de no disponer de una aplicación de monitoreo, se deberían crear distintos script, a modo de ejemplo:
    - Script para controlar la performance del equipo que alerte al administrador cuando los recursos tengan limites inferiores a los indicados.
    - Script para verificar los logs y evaluar si existe algún intento de ataque. Por ejemplo listar los intentos de conexión fallidos desglosado por número de intentos, IP's y nombre de usuario, evaluando el patrón "Failed password for"

```
grep -hi "Failed password for " /var/log/secure* | sed "s/invalid user //" | tr -s " " | awk '{print $11" "$9}' | sort | uniq -c | sort -n
```

3.3.5. Recomendaciones de carácter general

Si bien se han indicado en las secciones previas distintas sugerencias para segurizar el sistema operativo y cada elemento que lo conforma, así como también los servicios del Sistema Operativo, aplicación de firewall y monitoreo, a continuación se mencionan reglas de carácter general cuya importancia requiere valoración.

- Mantener las versiones de software instalados actualizadas.
- Estar atento sobre las vulnerabilidades reportadas y evaluar si afectan a las versiones instaladas en los equipos.

- Proteger a los servidores del acceso de usuarios con privilegios restringidos. No se debería proporcionar shells de login interactivos, a menos que sea un requerimiento absoluto.
- Escanear regularmente la red en busca de puertos abiertos, software instalado que no debería estar ahí.
- Confeccionar un buen catálogo de tablas o datos sensibles de la base de datos.
- No elegir contraseñas que puedan aparecer en los diccionarios, existen programas para romperlas y páginas donde verificar la dureza de una contraseña. Tener una política de contraseñas y realizar una revisión periódica de los mimas.

3.3.6. Módulos de lenguajes

Para que las aplicaciones puedan ser compiladas e interpretadas se requiere tener instalado en el equipo servidor aplicativos para tal fin. En caso se ser un servidor web, se deben instalar los módulos de lenguajes de programación utilizados en las aplicaciones web que el servidor aloja y enlazarlos con Apache.

En esta sección se mencionarán las consideraciones necesarias de configuración de seguridad en los módulos de lenguajes de programación instalados en el servidor web bajo estudio.

Módulo PHP

Cuando PHP es usado como un módulo de Apache, hereda los permisos del usuario de Apache (generalmente los del usuario "nobody"). Este hecho representa varios impactos sobre la seguridad y las autorizaciones. Por ejemplo, si se está usando PHP para acceder a una base de datos, a menos que la base de datos disponga de un control de acceso propio, se tendrá que hacer que la base de datos sea asequible por el usuario "nobody". Esto quiere decir que un script malicioso podría tener acceso y modificar la base de datos, incluso sin un nombre de usuario y contraseña. Es completamente posible que una botnet pudiese localizar la página web de administración de una base de datos, y eliminar la misma. Una protección ante este tipo de situaciones es mediante el uso del mecanismo de autorización de Apache, o con modelos de acceso de diseño propio usando LDAP, archivos .htaccess, etc. e incluir ese código como parte de los scripts PHP (48).

Se puede incluir la táctica de cambio de extensiones de los archivos php, indicándole al servidor web que interprete diferentes tipos de ficheros por medio de PHP, así se pueden utilizar extensiones de archivos engañosas. Esto se logra cambiando la extensión de los archivos php y configurando por medio de una directiva de .htaccess o en el archivo de configuración de Apache la interpretación de la nueva extensión.

```
# Hacer que el código de PHP parezca otro tipo de código
AddType application/x-httpd-php .asp .py .pl
# Hacer que el código de PHP parezca de tipo desconocido
AddType application/x-httpd-php .bop .foo .133t
# Hacer que el código de PHP parezca HTML
AddType application/x-httpd-php .htm .html
```

Una técnica de seguridad utilizada es el ocultamiento de información (49), aunque es una solución un poco débil, evita que el atacante pueda conocer el sistema objetivo y así utilizar otras técnicas de ataque. Se mostrarán diferentes configuraciones para ocultar información mostrada por Apache relacionada con PHP: versión, logos, etc. Editar el archivo /etc/php5/apache2/php.ini, modificando los siguientes parámetros:

```
disable_functions = exec , system , shell_exec , passthru
register_globals = Off
expose_php = Off
display_errors = Off
track_errors = Off
html_errors = Off
magic_quotes_gpc = Off
allow_url_fopen = Off
sesión.use_trans_sid = 0
```

La directiva "disable_functions" permite deshabilitar un conjunto de funciones, para evitar ataques de inyección PHP. Entre las funciones PHP que es conveniente restringir están: system, passthru, escapeshellarg, escapeshellcmd, proc_close, proc_open, ini_alter, popen, show_source, pcntl_exec y phpinfo.

La variable “register_globals” cuando está activada permite a un usuario cualquiera poder inicializar una variable remotamente, es decir permite inyectar los scripts con todo tipo de variables, como las de peticiones provenientes de formularios HTML. Esto, unido al hecho de que PHP no requiere la inicialización de variables, es fácil escribir código inseguro. En versiones previas a 4.2.0 esta variable estaba inicializada en ON.

La variable “expose_php” en OFF evita mostrar la versión de PHP en los headers.

La directiva “display_errors” se deshabilita para no permitir mostrar los errores de programación al navegar, información que puede ser utilizada por el atacante.

La directiva “magic_quotes_gpc” en OFF deshabilita las comillas mágicas. Las comillas mágicas son un procedimiento que permite limpiar los datos de entrada de un script PHP.

La directiva “allow_url_fopen” se deshabilita para evitar que ciertas funciones realicen llamadas a archivos que no se encuentran locales, solo permitirá acceder a archivos del servidor que tengan los suficientes permisos.

También se recomienda deshabilitar la variable “session.use_trans_sid”, colocándola en 0, para evitar mostrar el ID de sesión en la dirección. Es el valor por default.

Para evitar la inyección de comandos: wget, GET, curl, y demás comandos utilizados para descargar software, se puede agregar un script en el php.ini en la directiva, auto_prepend_file para que cuando se detecten en la url de cualquier script php, se omita la ejecución del comando.

```
<?
foreach ($_GET as $variable => $valor)
if (eregi("wget |curl |GET ", $valor)) {
exit
}
?>
```

Módulo mod_python

mod_python es una librería Open Source para el servidor web Apache, que permite integrar código python, tanto para la ejecución de lógica de aplicación, como para la interceptación de los propios "handlers" o puntos de control internos de Apache.

Existen versiones de "mod_python" que contienen vulnerabilidades, algunas de ellas son:

- Una vulnerabilidad reportada en mod_python anteriores a 3.0.4 y anteriores a 2.7.9 (CVE-2003-0973), permite que un usuario remoto pueda acceder al servidor Apache y causar una denegación de servicios (httpd crash) a través de una consulta de string.
- La versión de mod_python 3.1.4 (libapache2-mod-python) y anteriores no apropiadamente configurados puede retornar porciones de memoria previamente liberadas cuando un proceso tiene una salida mayor a 16384 bytes. Vulnerabilidad reportada en CVE-2004-2680.
- La versión 3.2.7 de mod_python incluye el módulo "FileSession", que contiene una vulnerabilidad (CVE-2006-1095) que permite a cualquier usuario local la ejecución de código arbitrario con los privilegios del proceso Apache a través de una cookie de sesión manipulada, esta característica no se encuentra activada por defecto.

Se recomienda actualizar la versión de mod_python a una versión posterior a 3.2.7.

3.4. Recomendaciones para el desarrollador de aplicaciones.

Como se mencionó en el alcance del trabajo, no se ha realizado análisis sobre el código de las aplicaciones web alojadas en el servidor, sin embargo se ha recopilado un conjunto de recomendaciones a aplicar en el desarrollo de las aplicaciones web que permitirán fortificarlas. Invitando a los desarrolladores de aplicaciones web a conocerlas y efectuar los cambios para segurizar sus desarrollos.

1. Configurar adecuadamente la variable register_globals de PHP. Las versiones recientes de PHP, por default register_globals está en Off, sin embargo algunos usuarios cambian esta configuración para aplicaciones que lo requieren. Esta configuración puede ser definida en "On" o en "Off" en el archivo php.ini o un .htaccess. La variable debería ser inicializada en "On".
2. Usar funciones de los lenguajes de programación en vez de usar de comandos de Shell en el código. Por ejemplo usar la función mkdir de php, en vez del comando mkdir del system del sistema operativo.
3. En PHP no usar las funciones: shell_exec, exec, system, readfile, passthru, escapeshellcmd, proc_open, posix_uname, posix_getuid, posix_geteuid, posix_getgid, getcwd para comandos en el sistema operativo.
4. Filtrar todas las entradas de datos del usuario antes de procesarlas. Asegurarse que solo datos filtrados sean usados para construir la cadena a ser ejecutada. Existen varias formas de lograrlo:
    - Filtrar las entradas de los usuarios reemplazando la aparición de ' por '' (dos comillas simples) e incluso evitando que los usuarios puedan pasar caracteres como \ / " ' o cualquier otro carácter que pueda causar Inyección SQL.
    - Utilizar funciones especiales como mysql_real_escape_string y addslashes, para ayudar a prevenir ataques Inyección SQL.
5. En PHP 5 se puede utilizar la extensión **mysqli**, que soporta los protocolos de autentificación y clave de acceso mejorados de MySQL, así como las sentencias preparadas con placeholders.
6. Se puede utilizar la función htmlentities(), quien convierte los caracteres especiales en su entidad html por ejemplo el carácter < en <, > en >, etc. Y así ayudar en la mitigación de ataques XSS.

7. Limitar al máximo los permisos del usuario que ejecutan sentencias SQL para evitar Inyección SQL. Por ejemplo utilizando un usuario distinto para las sentencias SELECT, DELETE, UPDATE y asegurándonos que la ejecución de una sentencia sea del tipo permitido. Se debería descartar el uso del usuario 'sa' o alguno que pertenezca al rol 'db_owner' para ejecutar las sentencias de uso habitual de la base de datos. Una solución sería trabajar con procedimientos almacenados ya que el modo en que se pasan los parámetros a los procedimientos almacenados evita la inyección SQL.
8. Validar los datos que introduce el usuario teniendo en cuenta la longitud de los campos y el tipo de datos aceptados.
9. Configurar las sesiones usando HTTPONLY. En PHP esto se consigue con la instrucción: ini_set('session.cookie_httponly', 1), para mitigar ataques XSS.
10. Para mitigar ataques de sesión hijacking se puede cambiar el lugar de almacenamiento de los archivos de sesión de la página web, se mencionan dos opciones:
    - Usar la función session_save_path para guardar los archivos de sesión en un directorio propio que sólo pueda acceder PHP (ya sea por estar fuera del directorio web o con un .htaccess con la instrucción deny from all). Este método no es demasiado fiable, ya que el resto de usuarios podrá leer en ese directorio, para evitar esto configurar privilegios de usuarios.
    - Usar la función session_set_save_handler para guardar las sesiones en la base de datos en lugar de archivos. Esta solución es la más segura ya que la página web será la única que tendrá acceso a la base de datos y, por tanto a las sesiones.
11. Para mitigar ataques clicjacking se puede emplear código defensivo dentro de la Interfaz del Usuario para asegurarse que los frames actuales corresponden a la ventana de máximo nivel.
    - Utilizando la técnica de protección de frames (frame busting) (50), para evitar que el documento se muestre en el iframe. Tener en cuenta que depende del navegador web del usuario. Se añade a la página el siguiente código:

```
<script>
If (top != window){
top.location = window.location
}
</script>
```

    - Bloqueando la navegación causada por top.location en el evento onbeforeunload. El controlador de este evento retorna una consulta al usuario, preguntando si quiere salir de la página o no (50). En el siguiente ejemplo, se protege el iframe con el siguiente código, la página hostil cancela la redirección cuando el usuario pulsa el botón:

```
<script>
  top.location = 'http://google.com'
</script>
<input type="button" value="test" onclick="alert('button works')">
```

12. Antes de elegir un framework de desarrollo leer sobre las vulnerabilidades del mismo y si existen parche o no.

13. Almacenar las contraseñas cifradas. Evitar el uso del método password() ya que es vulnerable y se recomienda el uso de AES y mantener el código fuente resguardado. Se pueden usar funciones hash, evaluar cuales no han sido rotas, no usar MD5 o SHA1. El algoritmo AES es el más completo y complejo. Sería vulnerable en el caso de que alguien consiguiese violar nuestro código fuente y observara la llave que se pasa en la cadena. Para almacenar estas contraseñas necesitamos que nuestro campo sea de tipo BLOB, ya que el resultado de la operación será un dato binario muy aleatorio, por ejemplo:

```
mysql> INSERT INTO usuarios VALUES ('usuario', AES_ENCRYPT ('contraseña','llave'));
```

Una herramienta para hacer análisis de código de aplicaciones web es Graudit (12) que permite encontrar fallos de seguridad potenciales en el código fuente usando la utilidad grep de GNU.

4. Conclusión

Considerando que la integridad y confidencialidad de los datos es una cuestión prioritaria para cualquier empresa y que la autenticación y la disponibilidad del servicio determinan la imagen empresarial y posible perdidas económicas por parte de la empresa u organización, es preciso aplicar las medidas de seguridad necesarias para contemplar estos principios de seguridad en la infraestructura web que brinda servicios e información a los usuarios.

Para responder a estos requerimientos, en el presente trabajo se han llevado a cabo tres etapas: por un lado la etapa de reconocimiento para obtener información y determinar estructura de la plataforma web, luego la etapa de análisis de vulnerabilidades de un servidor web para obtener información sobre las vulnerabilidades presentes y así poder llevar a cabo la etapa de la segurización del servidor abarcando desde el arranque del equipo, el sistema operativo, la red y los servicios que se ejecutan en el mismo.

Luego de realizar cada etapa, la información obtenida fue útil como punto de partida para la etapa siguiente.

En la etapa de reconocimiento se obtuvo información sobre la arquitectura física del equipo, los usuarios y los software instalados en el mismo, en esta etapa se vio reflejada la información que puede ser obtenida a través de distintas herramientas, ya sean instaladas opensource o a través de la web cuyo acceso a la misma no precisa autorización por parte del administrador del equipo o de la organización en sí, información de mucha utilidad para los atacantes de un servidor web. Con todos estos datos se realizó un análisis de vulnerabilidades más específico.

Luego de concluir la segunda etapa, disponiendo de la lista de vulnerabilidades presente en la infraestructura web y considerando un conjunto de recomendaciones sobre la segurización de servidores web se procedió a aplicar las medidas necesarias para mitigar las amenazas producto de las vulnerabilidades, y segurizar al equipo. A través de la aplicación de configuraciones de seguridad en los sectores de arranque, sistema operativo, red y servicios de acceso remoto como son el web, transferencia de archivos, base de datos, accesos remotos, se pudo lograr una infraestructura web más robusta utilizando comunicación encriptada utilizando certificados digitales y con acceso restringido.

Se implementó una política para lograr la disponibilidad del equipo aplicando monitoreo y funciones de detección y reanudación del servicio. El monitoreo consiste en evaluar el normal funcionamiento del equipo, los servicios que brinda y el registro de accesos remotos que

puede llevar a generar ataques de denegación de servicios. Para detectar y reanudar los servicios se desarrolló un script de backup de información sobre los datos y configuraciones del equipo almacenado en otro equipo por resguardo ante una falla física del dispositivo de almacenamiento. La transferencia de datos se realiza a través de una comunicación cifrada. Se implementó un servidor de respaldo el cual se mantiene inactivo hasta que el servidor principal cese su funcionamiento por alguna falla técnica o de configuración, utilizando la información resguardada, se logra la reanudación del servicio con las configuraciones y los datos de los usuarios y aplicaciones actualizados.

Valorando el hecho que los desarrolladores y los administradores de sistemas deben trabajar juntos para asegurar que las distintas capas estén configuradas apropiadamente, se propusieron recomendaciones para los desarrolladores web que deberían contemplarse durante el desarrollo de las aplicaciones web.

A modo de resumen, en la siguiente tabla se exponen algunas medidas de mitigación ante distintos ataques o vulnerabilidades actuales de entornos web.

Ataque o vulnerabilidades	Técnicas de Mitigación o protección
Denegación de servicios (DoS): **SlowLoris**	• Limitar tiempos de respuesta del servidor, configurando apache. Timeout 60 • Limitar recursos en el SO configurando el archivo /etc/security/limits.conf Habilitar protección TCP SYN: en /etc/sysctl incorporar: net.ipv4.tcp_syncookies net.ipv4.tcp_syncookies = 1 net.ipv4.tcp_max_syn_backlog = 2048 net.ipv4.tcp_synack_retries = 2 net.ipv4.tcp_syn_retries = 5 Prevenir respuestas a requerimiento de broadcast: net.ipv4.icmp_echo_ignore_broadcasts = 1 • En FTP: Limitar cantidad de usuarios → max_clients=20 Limitar el número de conexiones IP en simultaneo → max_per_ip=10
Cambios en la configuración	• Inmutar los archivos, evitando modificaciones de los archivos: chattr –i <archivo> • En apache al directorio correspondiente, usar política restrictiva: Order deny, allow, Deny from all
IP Spoofing	• En el SO configurar /etc/sysctl agregando: net.ipv4.conf.all.rp_filter = 2 net.ipv4.conf.default.rp_filter = 1

Modificar tablas de routeo	• Deshabilitar routeo del sistema: en /etc/sysctl net.ipv4.conf.all.accept_source_route = 0 net.ipv4.conf.default.accept_source_route = 0 • Deshabilitar redirecciones ICMP: en /etc/sysctl.conf net.ipv4.conf.all.accept_redirects = 0 net.ipv4.conf.default.accept_redirects = 0 net.ipv4.conf.all.send_redirects = 0 net.ipv4.conf.default.send_redirects = 0
Obtención de información	• Ignorar respuestas ICMP (ping): en /etc/sysctl.conf net.ipv4.icmp_echo_ignore_all=1 En archivo /etc/rc.d/rc.local comentar las siguientes líneas: #echo "" > /etc/issue #echo "$R" >> /etc/issue #echo "Kernel $(uname -r) on $a $(uname -m)" >> /etc/issue #cp -f /etc/issue /etc/issue.net #echo >> /etc/issue • En Apache: /etc/apache/apache2.conf. Evitar banner ServerTokensProductOnly ServerSignature Off Evitar header etag FileETag None • En PHP modificar php.ini. No mostrar versión en header. expose_php en OFF No mostrar errores de programación display_errors en OFF.
Clickjacking	• En Apache: agregar en apache2.conf Header always append X-Frame-Options SAMEORIGIN • En HTML empleando código defensivo dentro de la IU para asegurarse que los frames actuales corresponde a la ventana de máximo nivel. Utilizando la técnica de protección de frames, o bloqueando la navegación causada por top.location en el evento onbeforeunload.
Cross Site Scripting (X-XSS)	• En apache: Evitar acceso a directorios /libraries o /setup/lib y Configurar sesiones, en apache2.conf HttpOnly Secure Header set X-XSS-Protection “1; mode=block” • En PHP: evitar sesiones usando la función: ini_set('session.cookie_httponly', 1) Convertir los caracteres especiales en la correspondiente unidad HTML con la función: htmlentities().

Cross Site Tracing (CST)	• En apache agregar en apache2.conf traceEnable off
Sesión hijacking	• Deshabilitar el protocolo HTTP1.0, habilitando la directiva RewriteEngine en apache RewriteEngine On RewriteCond %{THE_REQUEST} !HTTP/1.1$ RewriteRule .* - [F] • En PHP, cambiar la localización de archivos de sesión: Usar la función session_save_path para guardar los archivos de sesión en un directorio dentro de la cuenta al que sólo pueda acceder PHP ó usar la función session_set_save_handler para guardar las sesiones en la base de datos en lugar de archivos.
Inyección PHP	• En PHP configurar el archivo php.ini register_globals en OFF Deshabilitar funciones: disable_functions = system, passthru, escapeshellarg, escapeshellcmd, proc_close, proc_open, ini_alter, popen, show_source, pcntl_exec, phpinfo, Filtrar entradas de datos del usuario. • Para evitar la inyección de comandos: wget, GET, curl, y demás comandos utilizados para descargar software, se puede agregar un script en el php.ini en la directiva, auto_prepend_file para que cuando se detecten en la url de cualquier script php, se omita la ejecución del comando.
Inyección SQL	• Si los request HTML usan el método get, incluir en mod_rewrite el patrón “drop table” si usan el método post, usar el módulo mod_security y agregar en apache SecFilter drop[[:space:]]table • En PHP filtrar datos externos que serán introducidos en la consulta, utilizando funciones especiales como addslashes y mysql_real_escape_string. • Limitar al máximo los permisos del usuario que ejecutan sentencias SQL • Trabajar con Store Procedure
Inyección de Script y HTML: Server Side Include (SSI)	• Deshabilitar módulo include en apache: <Directory /opt/apache/htdocs> Options -Includes </Directory>
Ataques de Fuerza Bruta	• Monitorear accesos en apache y analizar logs

Falsificación de peticiones en Sitios Cruzados (Cross Site Request Forgery (CSRF))	• En la aplicación debería existir un token no predecible en cada solicitud HTTP. También incluir herramientas de verificación de usuario válido, como catchas, son útiles.
Restringir acceso	• A un sitio web, aplicando restricciones en apache: <Directory /site> Options None AllowOverride None Order deny,allow Deny from all </Directory> • Evitar acceso desde el servidor de Base de Datos a los archivos: Desactivar comandos LOAD DATA LOCAL e iniciar el demonio mysqld con la opción –local-infile=0 • Evitar que desde PHP se puedan accederse a archivos del servidor, aplicar la directiva “allow_url_fopen” in OFF
Deshabilitar listado de directorios	• A través de navegación web, en Apache en <Directory> agregar la directiva Options a None o –Indexes • A través de cliente FTP→ limitar acciones de los usuarios y aislarlos en sus directorios chroot_local_user : YES chroot_list_enable: YES chroot_list_file: /etc/vsftpd/chroot_list
Proteger archivos de la ejecución de script	• Configurar propietario y permisos en los archivos: no tengan permisos de escrituras por usuarios nobody o del grupo nobody y que los archivos confidenciales no tengan permisos de lectura por estos usuarios. • Evitar ejecución de comandos desde consultas de Base de Datos: Desactivar comandos LOAD DATA LOCAL iniciando mysqld con la opción –local-infile=0
Referencia directa insegura a Objetos	• En código HTML verificar autorización sobre los objetos • En SO solo dar permisos al usuario correcto
Exposición a datos sensibles	• Cifrado de comunicaciones

Se espera que este trabajo pueda ayudar a administradores de infraestructura web para lograr mayor seguridad y privacidad de sus activos, pudiendo aplicar estas recomendaciones en su infraestructura.

5. Trabajos Futuros

En el presente trabajo se ha auditado y segurizado una infraestructura web específica de una organización de tamaño reducido, en la cual se han realizado la etapa de reconocimiento y análisis de vulnerabilidades para conocer la infraestructura, sus configuraciones y detectar las vulnerabilidades de los software instalados, concluyendo con la aplicación de medidas de seguridad en las configuraciones para aumentar la seguridad de los software y servicios instalados. Como posibles trabajos futuros se podrían aplicar mejoras a la infraestructura y a la configuración actual:

- Aplicar virtualización, manteniendo en distintas Máquinas Virtuales los diferentes servicios, independizando al servidor web del servidor FTP y de las bases de datos. Analizando en forma apropiada los recursos físicos a asignar a cada MV, aplicando la configuración adecuada a cada nuevo servidor (MV independiente), manteniendo la seguridad y monitoreo correspondiente.
- Se debería analizar con detenimiento si todas las aplicaciones web instaladas en el servidor web deben tener acceso desde el exterior, en caso que existan aplicaciones web con acceso interno, deberían aplicarse configuraciones de seguridad en Apache para restringir acceso externo o si el equipo lo permite tener dos servidores web virtualizados, uno con acceso desde Internet y otro interno.
- Se sugiere automatizar las herramientas de test y generar herramientas de configuración automatizadas de los archivos del Sistema Operativo.
- Evaluar e incorporar reglas más específicas al firewall si así lo requiere. Configurar un firewall waf en el servidor Apache.
- Analizar e incorporar un software IDS Y/o IPS para detección y prevención de ataques.
- Realizar la etapa de explotación de las vulnerabilidades para determinar si la segurización aplicada fue exitosa.

6. Bibliografía

1. https://www.akamai.com/us/en/our-thinking/state-of-the-internet-report/global-state-of-the-internet-security-ddos-attack-reports.jsp. [En línea]

2. https://geekflare.com/apache-web-server-hardening-security/. [En línea]

3. *Metodología de Análisis de Vulnerabilidades para Empresas de Media y Pequeña Escala.* **Daniel Santiago Garzón, Juan Carlos Ratkovich Gomes ,Alejandro Vergara Torres.** Bogotá, Colombia : Pontificia Universidad Javeriana.

4. **Stallings, William.** *Fundamentos de Seguridad en Redes: Aplicaciones y Estandares.* s.l. : Pearson Educacion, 2004.

5. *Diccionario de Amenazas.* s.l. : Sophos, 2013.

6. *OWASP TOP 10. Los 10 riesgos más críticos en Aplicaciones Web.* s.l. : The Owasp Foundation, 2010.

7. http://searchsecurity.techtarget.com/definition/hijacking. [En línea]

8. *http://searchsecurity.techtarget.com/definition/IP-spoofing.* [En línea]

9. https://cve.mitre.org/about/index.html. [En línea]

10. https://nvd.nist.gov/. [En línea]

11. http://www.emezeta.com/articulos/mod-evasive-evitando-denegacion-servicio-distribuida. [En línea]

12. https://www.modsecurity.org. [En línea]

13. http://www.emezeta.com/articulos/mod-security-mas-seguridad-en-tu-web. [En línea]

14. http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-iptables.html. [En línea]

15. http://www.emezeta.com/articulos/port-knocking-simple-pero-seguro. [En línea]

16. **Stallings, William.** *Cryptography and Network Security. Principles and Practice.* s.l. : Prentice Hall. Quinta Edición, 2011.

17. http://tools.ietf.org/html/draft-badra-hajjeh-mtls-06 . [En línea] 2011.

18. http://serverfault.com/questions/9708/what-is-a-pem-file-and-how-does-it-differ-from-other-openssl-generated-key-file. [En línea]

19. https://support.ssl.com/Knowledgebase/Article/View/19/0/der-vs-crt-vs-cer-vs-pem-certificates-and-how-to-convert-them. [En línea]

20. **Bradley, Tony y Shah, Satish.** *Unified Communications For Dummies.* s.l. : Wiley publishing, 2010.

21. https://nmap.org/man/es/. [En línea]

22. http://www.netcraft.com/. [En línea]

23. http://www.openvas.org/. [En línea]

24. http://w3af.org/. [En línea]

25. https://cirt.net/Nikto2. [En línea]

26. **Mourani, Gerhard.** *Securing and Optimizing Linux: RedHat Edition.* [ed.] Red Hat Edition. s.l. s.l. : Open Network Architecture and OpenDocs Publishing. , 2000. ISBN: 0-9700330-0-1.

27. **Dery, Alexander.** *Hardening Debian 4.0.* s.l. : SANS Institute, 2009.

28. **Oldani, Enrico Perla and Massimiliano.** *A guide to kernel exploitation.* s.l. : Elseiver, 2011.

29. **Lopez, Gracia Fernandez.** *Seguridad en Sistemas Operativos.* s.l. : Universidade Da Coruña, 2007.

30. https://help.ubuntu.com/community/FolderEncryption. [En línea]

31. http://ecryptfs.org/. [En línea]

32. http://unaaldia.hispasec.com/2002/11/opciones-de-seguridad-en-linux-traves.html. [En línea]

33. https://www.ndchost.com/wiki/server-administration/hardening-tcpip-syn-flood. [En línea] 2011.

34. http://www.flatmtn.com/article/setting-openssl-create-certificates. [En línea]

35. http://www.flatmtn.com/article/setting-ssl-certificates-apache. [En línea]

36. **Rich Bowen, Ken Coar.** Capitulo 6. Security. *Apache Cookbook.* s.l. : O'Reilly, 2003.

37. http://www.thegeekstuff.com/2011/03/apache-hardening/. [En línea]

38. http://www.ducea.com/2006/06/15/apache-tips-tricks-hide-apache-software-version/. [En línea]

39. https://httpd.apache.org/docs/2.4/mod/core.html. [En línea]

40. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418. [En línea]

41. http://geekflare.com/secure-apache-from-clickjacking-with-x-frame-options/. [En línea]

42. http://php.net/manual/es/security.database.sql-injection.php. [En línea]

43. http://www.supportpro.com/blog/2009/08/mod_security-intro/. [En línea]

44. http://mysql.localhost.net.ar/doc/refman/5.0/es/security.html. [En línea]

45. http://es.tldp.org/Postgresql-es/web/navegable/todopostgresql/security.html. [En línea]

46. https://www.postgresql.org/support/security/. [En línea]

47. http://blog.segu-info.com.ar/2011/08/cross-site-scripting-en-phpmyadmin.html. [En línea] 2011.

48. http://docs.phpmyadmin.net/es/latest/setup.html. [En línea]

49. http://www.alcancelibre.org/staticpages/index.php/09-como-vsftpd. [En línea]

50. http://wiki.vpslink.com/Configuring_vsftpd_for_secure_connections_(TLS/SSL/SFTP). [En línea]

51. http://munin-monitoring.org/. [En línea]

52. https://www.nagios.org/. [En línea]

53. http://php.net/manual/es/security.apache.php. [En línea]

54. http://php.net/manual/es/security.hiding.php. [En línea]

55. http://javascript.info/tutorial/clickjacking. [En línea]

56. http://unaaldia.hispasec.com/2006/03/actualizacion-de-seguridad-de.html. [En línea] 2006.