UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Universidad Internacional de La Rioja
Máster universitario en Seguridad Informática

Instalación Segura de Servidor Usando Aplicaciones de Código Abierto

Trabajo Fin de Máster
presentado por: Coma Bonias, Alejandro
Director/a: Cobos Guzman, Salvador

Ciudad: Madrid, España
Fecha: 23/9/2017

Resumen

El propósito de este trabajo es ofrecer a cualquier organización la posibilidad de proteger sus sistemas informáticos de una forma gratuita, haciendo uso de herramientas de código abierto tales como el Sistema Operativo, Servidor Web ó Servidor de Base de Datos y asegurando un alto nivel de seguridad.

La metodología que se ha seguido es la de investigar el estado del arte en cuanto a las diferentes herramientas de código abierto disponibles, comparar entre ellas y elegir la más conveniente, para luego en base a la documentación abierta disponible definir unos objetivos de seguridad y conseguir una configuración segura y robusta y a su vez documentar y automatizar el proceso.

Como resultado se obtendrá un método mediante el cual, con unos conocimientos básicos de informática, pueda replicarse esta configuración segura en un servidor.

Palabras Clave: servidor seguro, software de código abierto, automatización.

Abstract

The aim of this thesis is to offer any organization the possibility of protecting their systems in a free manner using open source tools, and at the same time ensuring a high security level.

The followed method has been to investigate the state of the art related to the different available open source tools, comparing them and then choosing the most convenient for this work. Later, based on the open available documentation, defining the security goals for achieving a safe and robust configuration and at the same time documenting and automating the process.

The outcome will be a method for replicating this safe server configuration with a basic technological computer knowledge.

Keywords: secure server, open source software, automation.

Lista de Abreviaturas

SO	Sistema Operativo
RHEL	R*ed Hat Enterprise Linux*
RPM	*Red Hat Package Manager*
HTTP	*HyperText Transfer Protocol*
CVE	*Common Vulnerabilities and Exposures*
IMAP	*Internet Message Access Protocol*
POP3	*Post Office Protocol 3*
PHP	*PHP Hypertext Preprocessor*
LOPD	Ley Orgánica de Protección de Datos
SQL	*Structured Query Language*
SOE	*Standard Operating Environment*
YAML	*YAML Ain't Another Markup Language*
SSH	*Secure Shell*
NIST	*National Institute of Standards and Technology*
BIOS	*Basic Input Output System*
DNS	*Domain Name System*
TLS	*Transport Layer Security*
SW	Servidor Web
BD	Base de Datos
DoS	*Denial of Service*
DDoS	*Distributed Denial of Service*
MPM	*Multi-Processing Module*

Índice

Índice de ilustraciones

1. Introducción

1.1. Motivación del trabajo.

La sociedad actual vive en la denominada “Era de la Información” dónde cada individuo dispone de un número elevado y creciente de dispositivos electrónicos, que de una manera u otra están conectados entre sí: ordenadores, teléfonos móviles, electrodomésticos, vehículos, etcétera. A medida que esta tendencia crece, hace a los individuos más y más dependientes de la tecnología, por lo que esta debe ser confiable y segura, ya que en algunos casos incluso vidas humanas dependen de ello, como ocurre con los sistemas de defensa, las centrales nucleares o las presas hidrológicas.

La concienciación de los usuarios de dichas tecnologías sobre la importancia de la ciberseguridad es primordial. Según publica la ENISA (del inglés *European Union Agency for Network and Information Security*) en [1] y el SANS Institute (del inglés *SysAdmin, Networking and Security Institute*) [2] suele ser el usuario el eslabón más débil: la mayoría de ciberataques comienzan con la ingeniería social, que no es más que la recaudación de información sensible acerca de la seguridad sea o no consciente la víctima de esta fuga.

Hoy en día los ciberdelincuentes forman parte de bandas organizadas bien preparadas operando a nivel mundial, que son capaces de estudiar sus objetivos durante meses, esperando pacientemente el descuido de algún usuario para conseguir penetrar los sistemas y lograr sus objetivos, que no solamente se reducen al robo de dinero o tráfico de información. Los intereses políticos, el daño sin motivo, el reconocimiento público o el control remoto de sistemas para otros fines están entre los motivos de un ciberataque a una organización [3].

La ciberseguridad, por lo tanto, debe tratarse como un elemento transversal, que abarca desde el usuario que recibe un correo electrónico hasta el “núcleo” de un sistema informático que ejecuta programas informáticos, pasando por los elementos de red, los ordenadores personales o los dispositivos móviles [4]. Pero sin duda alguna los incidentes de seguridad más graves se dan cuando el atacante consigue penetrar la red, y realiza una escalación de privilegios para tomar el control de los ordenadores centrales.

Dicho lo anterior, ninguna organización, sea pequeña o grande, está a salvo de estos ciberdelincuentes, que utilizan herramientas avanzadas para escanear la red en busca de posibles víctimas e intentar penetrar en cualquier sistema y hacerse con su control.

No todas las organizaciones cuentan con los mismos recursos para evitar ser atacados por un ciberdelincuente u organización criminal. Es cierto que los atacantes suelen invertir mayores esfuerzos en doblegar la seguridad de una gran empresa con diversos fines, pero estas tienen la capacidad de invertir más en seguridad informática que una empresa u organización de menor capital, ya que por otra parte corren el riesgo de sufrir mayores pérdidas. Dado que las organizaciones más pequeñas no están exentas del riesgo de sufrir un incidente de seguridad informática al estar la mayoría de ellas expuestas a Internet, es interesante la propuesta de realizar este trabajo usando herramientas gratuitas.

Por tanto, la motivación de este trabajo es la de elaborar una propuesta de instalación y configuración segura de un servidor de aplicaciones utilizando exclusivamente software de código abierto. Para ello, primero se hará un estudio acerca de qué elementos se deben asegurar, y las herramientas de las que disponemos para ello. Luego se investigará la mejor manera de hacerlo, posteriormente se desarrollará una metodología, y se automatizará el proceso, para que sea replicable. En una fase final, analizaremos la seguridad del servidor, sirviéndonos de herramientas de código abierto.

1.2. Planteamiento del trabajo.

Se propone, tal y como se ha citado anteriormente, el uso de herramientas de código abierto desarrolladas por la comunidad y accesibles por cualquiera de manera gratuita. En cuanto a la ciberseguridad, la ventaja acerca de usar herramientas de código abierto radica en que tanto los desarrolladores del software como otros usuarios pueden analizar el código, detectar vulnerabilidades, y posteriormente contribuir al desarrollo, mejora y mantenimiento del mismo [5], [6].

Se hará primero un estudio del “estado del arte” cuya finalidad será evaluar qué herramientas hay disponibles para este trabajo y decidir cuáles encajan mejor para cumplir el propósito deseado, analizando sus ventajas y desventajas.

En dicho análisis se van a evaluar diferentes herramientas tales como sistemas operativos de código abierto, servidores web y bases de datos. Finalmente se seleccionará la configuración a implementar.

Para el desarrollo del trabajo se hará uso de la documentación oficial de las diferentes herramientas que se deseen instalar y configurar de manera segura, así como también se consultarán artículos escritos por la comunidad, al tratarse de herramientas de código abierto.

Tras el desarrollo de la solución, se deberá probar que esta es replicable y funcional, y se hará un análisis de vulnerabilidades de la misma para determinar que es segura.

El objetivo final es desarrollar una metodología que permita a cualquier administrador de sistemas sin excesiva experiencia replicar la instalación y configuración segura del servidor sin la necesidad de pagar por ninguna licencia de software, asegurando una alta calidad en cuanto a la seguridad.

1.3. Estructura del trabajo.

La estructura de este trabajo se divide en cinco capítulos. En el Capítulo 1 se hace una introducción, dónde se muestra la importancia de la seguridad informática en cualquier organización y la motivación del mismo, para posteriormente abordar el planteamiento de las diferentes fases del trabajo: análisis del estado del arte, definición de objetivos, desarrollo de la metodología y evaluación y posibles trabajos futuros.

En el Capítulo 2 se lleva a cabo el análisis del estado del arte en lo referente a la temática del trabajo, concretamente en cuanto a las herramientas de software libre que componen el servidor: el sistema operativo, el servidor web, la base de datos, las herramientas disponibles para la automatización del proceso de instalación y configuración, y las herramientas de análisis de vulnerabilidades. Como resultado de este análisis se decidirá qué herramientas se van a utilizar para realizar el trabajo. Finalmente se explicará la aportación de este trabajo al estado del arte.

En el Capítulo 3 se enumeran los objetivos concretos que se desean lograr en base al análisis de la documentación disponible. En el caso de este trabajo, al tratarse de software de código abierto hay disponible la suficiente documentación de los proyectos como para asegurar una buena puesta en seguridad de las diferentes aplicaciones utilizadas. Posteriormente se detalla la manera de cumplir dichos objetivos y cómo se puede automatizar la solución.

En el Capítulo 4 se desarrolla detalladamente la metodología necesaria para cumplir los objetivos de seguridad definidos en el capítulo anterior. Posteriormente se hará la evaluación de la metodología para ver si se han cumplido los objetivos que se habían definido mediante el análisis de vulnerabilidades. Se hará uso de las herramientas de análisis de vulnerabilidades escogidas en el Capítulo 2.

Finalmente, en el Capítulo 5 se verán las conclusiones y se plantearán posibles trabajos futuros.

2. Estado del arte.

Es fundamental realizar un análisis del estado del arte de la tecnología que ayude a tomar decisiones en cuanto a qué herramientas vamos a utilizar para realizar este trabajo. Se trata de comparar entre varias alternativas que sean referentes tecnológicos en la actualidad, teniendo en cuenta el aspecto de que trata el trabajo: la seguridad. Por tanto, no se va a considerar el uso de herramientas que no cumplan un mínimo de calidad en cuanto a la seguridad, que no sean configurables hasta un grado en el que puedan ser consideradas seguras, o simplemente que provengan de fuentes desconocidas o poco probadas. Un ejemplo de esto sería si se utilizase para este trabajo software proveniente de fuentes no oficiales, poco contrastadas o poco fiables.

Como el objetivo de este trabajo es realizar la instalación segura tanto del sistema operativo como de algunas aplicaciones de servidor se buscarán artículos y bibliografía acerca de qué tecnologías son las más usadas. Se tratarán de soportar las decisiones tomadas teniendo en cuenta la seguridad de los productos. Una vez estudiada la configuración segura que debe aplicarse, se hará uso de alguna herramienta que nos permita automatizar el proceso de instalación segura o *hardening*, para finalmente acabar el trabajo con un análisis de vulnerabilidades de la instalación segura.

2.1. Sistema Operativo.

Llegados a este punto del trabajo, por el momento el único requisito para definir qué sistema operativo vamos a utilizar es que sea un sistema operativo de código abierto y sea considerado seguro.

A día de hoy, uno de los proyectos de código abierto más importantes y activos a nivel mundial es el *kernel* (núcleo) de Linux. Se trata de un proyecto que ha cumplido recientemente 25 años de vida, desde que en 1991 su creador Linus Torvalds lo inició. A su estado actual, y desde que su código está introducido en el sistema distribuido de control de versiones “Git” [7] en el año 2005, han contribuido más de 13500 desarrolladores y hace un total actual de 22 millones de líneas de código [8].

Al desarrollo del kernel de Linux contribuyen empresas de la talla de Intel, Red Hat, Linaro, Samsung, SUSE, IBM, Renesas, Google y AMD. Cabe destacar del informe que presentó la *Linux Foundation* en 2016: "*25 Years of Linux Kernel Development*" que solamente el 7,7% de los desarrolladores que han contribuido recientemente no percibe un salario por ello, es decir, el 92,3% de los desarrolladores que contribuyen al *kernel* de Linux perciben un salario. Tal y como reza este informe*, "el kernel de Linux, pues, se ha convertido en un recurso común desarrollado a gran escala por empresas que son feroz competencia en otras áreas."*

Es, por tanto, un claro ejemplo de proyecto de código abierto de éxito, ya que alrededor del *kernel* de Linux nacen multitud de distribuciones: Arch Linux, Debian, CentOS, Fedora, RHEL, Elementary OS, Gentoo, Ubuntu, Mandriva, Mint, etcétera [8].

Sin embargo, mientras que todas (o casi todas) estas distribuciones Linux pueden usarse como sistemas operativos de servidor, las distribuciones más usadas para ello son tres: Ubuntu, Debian y CentOS (ver Figura 1), y siendo por tanto Unix el sistema operativo más usado en los servidores web por delante de a Windows y muy por encima de OS X (ver Figura 2)

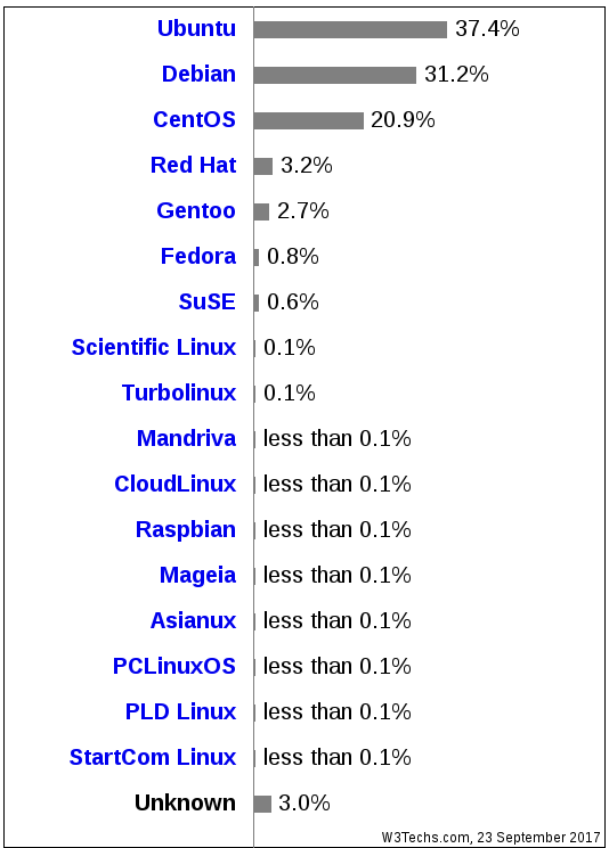


Figura 1: Servidores web usando diversas subcategorías de Linux. Fuente: w3techs.com

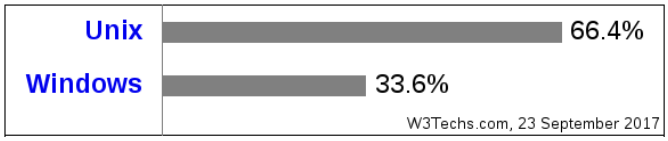


Figura 2: Porcentajes de sitios web usando diversos sistemas operativos. Fuente: w3techs.com

Sería muy complicado elegir entre estas tres distribuciones Linux más usadas como sistema operativo de servidor web basándose en cuál de ellas es más segura, o cuál de ellas es mejor para este propósito. Se pueden encontrar multitud de artículos en Internet donde unos autores defenderán una de ellas, y otros la otra. Lo cierto es que probablemente se acertaría con cualquiera de ellas si posteriormente se hace una buena puesta en seguridad o *hardening* del sistema operativo. Dado que no es el objetivo de este trabajo extenderse demasiado en este punto, se elegirá el sistema operativo CentOS por las siguientes razones:

1. El ciclo de vida total de la *major release* de CentOS es de 10 años [9], frente a los aproximadamente 5 años en los demás (en el caso de las versiones denominadas *Long Term Support)* [10]. Esto alarga el ciclo de vida de un servidor entre actualizaciones mayores, reduciendo los costes de migración de una versión a la siguiente.
2. CentOS es considerado un sistema operativo muy estable y robusto, dado que hereda los repositorios de software de su versión *Enterprise (Red Hat Enterprise Linux - RHEL)* [11].
3. Es compatible con software (paquetes *Red Hat Package Manager* ó RPMS) desarrollados para Fedora o RHEL.
4. Las versiones del software que incluye suelen ser estables, ya que se actualizan en consonancia con las actualizaciones de RHEL (entre 24 y 72 horas después [12]).

2.2. Servidor Web.

El siguiente paso tras haber seleccionado el sistema operativo para el servidor será definir los servicios que va a proporcionar para poder hacer un análisis del estado del arte de las aplicaciones disponibles.

Al instalar un servidor, una de las principales funciones para las que se utiliza es la de un servidor de aplicaciones o páginas web. Como muestra la Figura 3, los servidores web más utilizados en el mundo son de código abierto (Apache y Nginx) y entre las dos suponen el 83,9% según *Web Technology Surveys.*

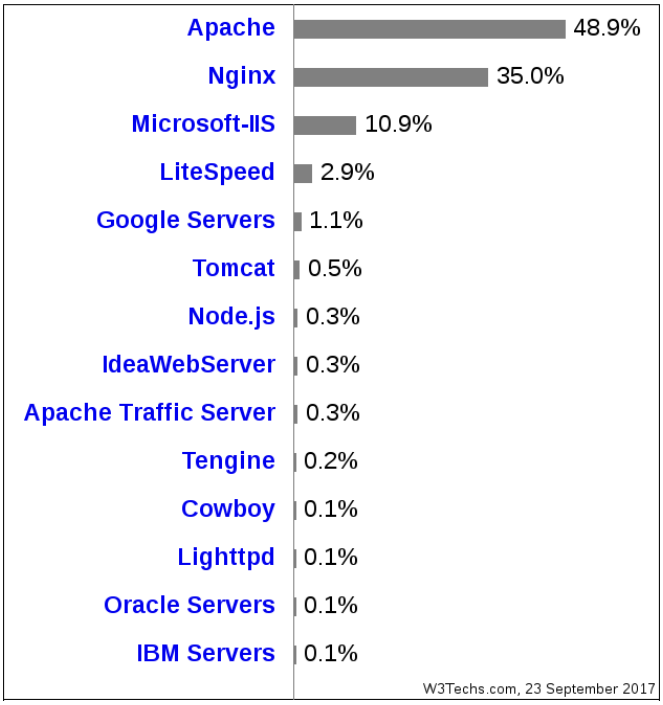


Figura 3: Porcentajes de Servidores Web. Fuente: w3techs.com

La decisión acerca de qué aplicación se usará como servidor web por tanto estará entre dos candidatos: Apache y Nginx.

- Apache HTTP Server [13], liberado en el año 1995, es el servidor web más utilizado desde 1996. Según se indica en su sitio web, *"La meta de este proyecto es proveer un servidor seguro, eficiente y extensible que provea servicios HTTP en sincronismo con los estándares HTTP actuales"*. En la página principal se encuentra un enlace a los "informes de seguridad", donde se indican las vulnerabilidades solucionadas en

las diferentes actualizaciones de código, un enlace para suscribirse a un boletín de notificaciones sobre nuevos parches, o para reportar posibles nuevos problemas de seguridad. También se puede encontrar un enlace a los “consejos de seguridad” bastante útiles para una instalación segura básica del software. Las vulnerabilidades del servidor HTTP Apache son etiquetadas según uno de los estándares de caracterización de vulnerabilidades de seguridad más conocidos: el estándar CVE (Common Vulnerabilities and Exposures) [14].

- NGINX (engine-x) [15] nació en el año 2002, y su uso en los sitios web más activos ha tenido un crecimiento continuo desde entonces (ver Figura 4). En su sitio web se indica: “NGINX es un servidor HTTP y proxy inverso gratuito, de código abierto y de alto rendimiento, así como un servidor proxy IMAP/POP3. Es conocido por su alto rendimiento, estabilidad, riqueza funcional, configuración simple y bajo consumo de recursos”. También se encuentra un enlace al apartado “Seguridad”, donde se muestran las últimas vulnerabilidades encontradas (CVE), los parches disponibles, y una dirección de correo electrónico donde reportar posibles problemas de seguridad encontrados.

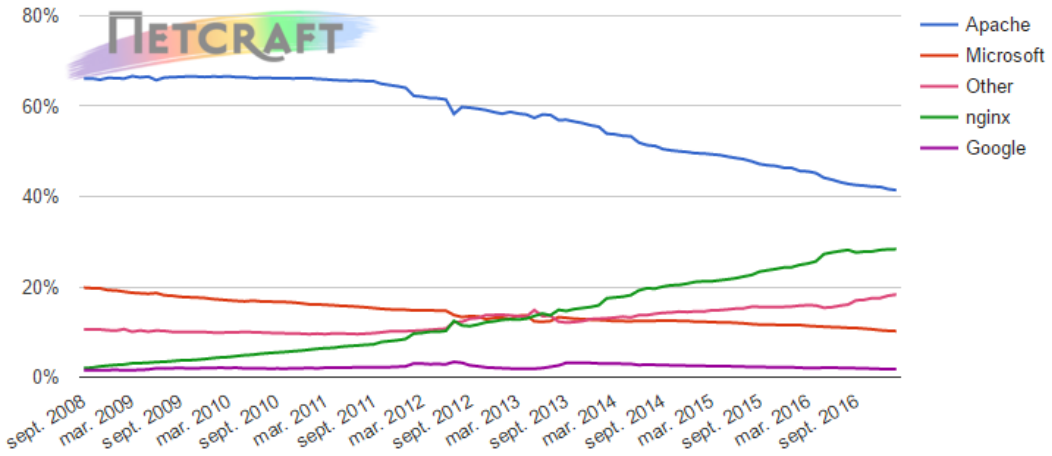


Figura 4: Aplicaciones de servidor web utilizadas por los sitios web con más tráfico. Fuente: Netcraft

Ambos son proyectos maduros, que ofrecen a los usuarios estabilidad, seguridad, documentación y gran cantidad de información.

En cuanto a las ventajas e inconvenientes entre uno y otro destacan:

- NGINX promete a los usuarios un mejor comportamiento en entornos de alta carga de conexiones simultáneas, menor uso de memoria y una mejor escalabilidad.
- Apache es más maduro, con un soporte de la comunidad más extendido, tiene módulos para soportar una gran variedad de lenguajes de programación de sitios web (PHP, Python, Perl, Ruby on Rails) y requiere de menos esfuerzo de aprendizaje y configuración.

Como comentario acerca de esta comparación entre ambas aplicaciones puede decirse que, dadas las bondades de uno y otro, hay autores que recomiendan el uso combinado de ambos: NGINX tiene un rendimiento muy superior sirviendo contenido web estático (páginas web que no utilizan aplicaciones o bases de datos) y Apache tiene mejor compatibilidad y variedad de módulos oficiales sirviendo páginas con contenido dinámico. Por esto se pueden encontrar artículos que recomiendan el uso combinado de ambos, haciendo NGINX de servidor proxy inverso ofreciendo el contenido estático del sitio web, y redirigiendo las peticiones de contenido dinámico a Apache para que sea este quien las procese [16], [17].

En cuanto a la seguridad que ofrecen ambos, dado que se trata de dos proyectos maduros y con un amplio soporte de la comunidad, no se puede decir que haya un claro dominador entre ellos: ambos se pueden considerar productos seguros, y la seguridad final que se consiga con uno u otro va a depender en gran medida de la configuración que se haga de los mismos.

Finalmente, y considerando que uno de los objetivos de este trabajo es que esté dirigido a un público más amplio y que no requiera de un conocimiento muy avanzado para su despliegue, se va a escoger Apache como servidor web. Pese a que el rendimiento de NGINX es mejor en entornos de alta carga de contenido estático a igualdad de recursos hardware, el mayor soporte de la comunidad, la excelente documentación oficial y la menor curva de aprendizaje, hace que se escoja Apache frente a NGINX.

2.3. Base de Datos.

Otra de las aplicaciones más comúnmente utilizadas en un servidor es un motor de base de datos. Las bases de datos se utilizan para almacenar información de una forma persistente utilizando esquemas estructurados. Hoy en día prácticamente todas las aplicaciones informáticas hacen uso de una base de datos para el almacenamiento de la información.

Para cualquier empresa u organización, desde el punto de vista de la seguridad informática, la base de datos es quizá el elemento más importante a proteger. Si un atacante lograse comprometer la integridad o privacidad de la base de datos ganando acceso a los datos contenidos o consiguiendo modificar los mismos, se vería gravemente amenazada la continuidad de la actividad de dicha entidad.

Los datos que contiene una base de datos pueden ser de diferentes tipos, tanto empresariales, datos contables, de configuración de los sistemas o datos de carácter personal. Tanta es la importancia de los datos de carácter personal que en España están amparados por la LOPD (Ley Orgánica de Protección de Datos) [18]. La LOPD define los requisitos legales que deben cumplirse y qué medidas de protección se deben aplicar a las bases de datos que contienen datos de diferentes niveles de categorización.

El solo incumplimiento de la LOPD en un fichero que contenga datos de carácter personal, incluso cuando no se haya producido un incidente de seguridad, conlleva multas que, dependiendo de la gravedad, oscilan entre los 900 a los 600.000 euros [18].

Como podemos ver en la Figura 5, los cinco motores de bases de datos más populares son por este orden: Oracle, MySQL, Microsoft SQL Server, PostgreSQL y MongoDB.

Rank			DBMS	Database Model	Score		
Jun 2017	May 2017	Jun 2016			Jun 2017	May 2017	Jun 2016
1.	1.	1.	Oracle	Relational DBMS	1351.76	-2.55	-97.49
2.	2.	2.	MySQL	Relational DBMS	1345.31	+5.28	-24.83
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1198.97	-14.84	+33.16
4.	4.	↑ 5.	PostgreSQL	Relational DBMS	368.54	+2.63	+61.94
5.	5.	↓ 4.	MongoDB	Document store	335.00	+3.42	+20.38
6.	6.	6.	DB2	Relational DBMS	187.50	-1.34	-1.07
7.	7.	↑ 8.	Microsoft Access	Relational DBMS	126.55	-3.33	+0.32
8.	8.	↓ 7.	Cassandra	Wide column store	124.12	+1.01	-7.00
9.	9.	↑ 10.	Redis	Key-value store	118.89	+1.44	+14.39
10.	10.	↓ 9.	SQLite	Relational DBMS	116.71	+0.64	+9.92

Figura 5: Top 10 de Motores de Base de Datos. Fuente: db-engines.com

Cabe hacer un breve inciso en cuanto al significado de la columna "*Database Model*". Distinguimos cuatro modelos de base de datos en este Top 10:

1. *Relational DBMS*: Son también llamadas bases de datos "SQL". Organizan la información en distintas tablas, que por lo general guardan una relación unas con otras. Son el modelo más extendido.
2. *Document Store*: Su característica principal es que están optimizadas para almacenar y consultar documentos, usando los metadatos de estos.
3. *Key-Value Store*: Almacenan la información en pares con claves únicas, a diferencia de las tablas en las bases de datos relacionales. Su simplicidad hace que sean más eficientes en algunos escenarios concretos.
4. *Wide Column Store*: Almacenan la información en registros que contienen columnas de nombre y tamaño dinámicos. Una entrada puede contener billones de columnas.

Como podemos observar en la Figura 5, los modelos de base de datos más comunes son las bases de datos relacionales. Los otros modelos de base de datos surgen de necesidades específicas de escalabilidad por grandes volúmenes de almacenamiento (*Document Store*) o por necesidad de almacenar datos con una estructura específica (*Key-Value y Wide Column*). Por tanto vamos se tomarán en cuenta solamente las bases de datos relacionales.

Entre las bases de datos relacionales que encontramos en la Figura 5, las más populares entre las que son consideradas de código abierto son MySQL y PostgreSQL.

- MySQL [19] muestra en su página principal algunos de sus principales clientes: Youtube, Paypal, Google, Facebook, Twitter, eBay ó Cisco. MySQL fue adquirido en 2008 por la empresa Sun Microsystems, que posteriormente fue adquirida por Oracle en el año 2010. Cuando Oracle se hizo con la propiedad de MySQL, varios de sus desarrolladores crearon un *"fork"* (o bifurcación) llamado MariaDB. Al parecer Oracle es reacio a liberar todo el código de MySQL. Algunos de sus módulos son de código cerrado, no se liberan de manera continua parches de seguridad y la comunidad es más activa en el proyecto de G*it* soportando y desarrollando MariaDB que el código liberado de MySQL [20], [21] y [22].

- PostgreSQL [23], según se indica en su página web *"cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una fuerte reputación en cuanto a fiabilidad, integridad de los datos y exactitud"*. Cuenta con soporte para la integración (conectores) de multitud de lenguajes de programación y presume de que su código es completamente abierto.

Dado que MySQL ya no se puede considerar completamente de código abierto, se traslada la decisión a tomar entre MariaDB y PostgreSQL.

De la misma forma que en el punto anterior, las medidas de seguridad que proporcionan tanto MariaDB como PostgreSQL son más que suficientes si la configuración de las herramientas es la correcta, por tanto, no hay una clara decisión acerca de por cuál decantarse en este sentido.

Para tomar la decisión acerca de qué motor de base de datos utilizar, se va a recurrir a la conocida arquitectura LAMP [24], que es un acrónimo usado para describir un sistema de infraestructura típico, que consiste en 4 elementos:

1. Linux (Sistema Operativo)
2. Apache (Servidor Web)
3. MySQL / MariaDB (Gestor de Base de Datos)
4. Perl / PHP / Python (Lenguajes de Programación)

Al ser una configuración ampliamente utilizada y popular, y la elección por defecto tanto en sistemas Red Hat como Ubuntu se decide hacer uso de MariaDB.

2.4. Herramientas de Automatización de la Configuración.

Llegados a este punto del análisis del estado del arte, donde ya se ha decidido qué herramientas van a componer el servidor, se hace necesario el análisis de las diferentes aplicaciones existentes cuya utilidad es la de automatizar la instalación y aplicar la configuración deseada.

Los beneficios que aporta la automatización de la instalación son varios:

1. Hace que el proceso sea fácilmente repetible.
2. Hace que el proceso sea mejorable e iterativo.
3. Hace que el proceso esté libre de errores humanos.
4. Hace que el proceso sea más eficiente en términos temporales.

2.4.1 Automatización de la Instalación del SO.

Tal y como se ha concluido en el apartado 2.1, el Sistema Operativo que se va a utilizar es CentOS 7. Este Sistema Operativo cuenta con una herramienta que permite a los administradores de sistemas la personalización y automatización de la instalación, para que no sea necesaria la intervención manual durante la instalación. Esta herramienta es conocida como *kickstart* [25], y puede encontrarse una amplia variedad de documentación acerca de cómo usarla. En este trabajo nos basaremos en la guía oficial de instalación de RHEL 7 (*Red Hat Enterprise Linux 7*) [26] que tal y como se indicaba anteriormente, es la versión empresarial del mismo producto. A modo introductorio, ya que no es el objetivo de este apartado entrar en detalles, cabe mencionar que mediante la instalación vía *kickstart* se pueden personalizar aspectos como el particionamiento de los discos, los paquetes de software a instalar o habilitar el uso de la herramienta de seguridad propia de Linux SELinux (*Security-Enhanced Linux*).

La personalización y automatización de la instalación dará como resultado lo que puede denominarse como SOE (del inglés *Standard Operating Environment*). Los beneficios que proporcionan los SOE a las organizaciones son [27]:

- Incrementar la eficiencia del aprovisionamiento de sistemas.
- Estandarización de los sistemas.
- Repetibilidad de los procesos de instalación.

2.4.2 Automatización de la Configuración.

Una vez se ha instalado el SOE, es común hacer uso de herramientas que ayuden a configurar automáticamente y de una forma más precisa y detallada tanto el sistema operativo como las diferentes aplicaciones. De esta manera se consigue el mayor grado de precisión y seguridad en la configuración.

Estas herramientas son conocidas como herramientas de administración de servidores, y sirven generalmente para automatizar los procesos que antiguamente se ejecutaban siguiendo un manual de configuración. Mediante estas herramientas de automatización de la configuración, se obtienen los siguientes beneficios [28]:

- Convertir complejos procedimientos en procesos automáticos fácilmente repetibles, evitando así posibles errores humanos.
- Estos procesos pueden ser llevados a cabo por personas con menor experiencia o conocimientos técnicos.
- Las empresas también se benefician de las herramientas de administración de servidores ya que se aumenta la eficiencia y se reducen costes operacionales en el despliegue y mantenimiento de los sistemas.

Tal y como puede verse en la referencia bibliográfica [29], existen numerosas herramientas de administración de servidores de código abierto. De entre las numerosas herramientas que se mencionan, destacan por número de contribuyentes a sus proyectos en *Git* [30], [31] y [32] en orden de mayor a menor: Ansible, Saltstack, y Chef.

Tal y como ocurre en los anteriores apartados, donde se ha escogido entre una u otra opción, cualquiera de las tres herramientas podría ser usada para el propósito que aquí se busca. A modo de resumen, las características principales de las tres aplicaciones son:

1. Ansible [33]: Destaca tanto por su arquitectura “sin agente”, es decir, no es necesario que haya una aplicación ejecutándose como servicio en la máquina a configurar, tanto por su rápida curva de aprendizaje debido al uso del lenguaje YAML (del inglés *YAML Ain’t Another Markup Language*) [34] como a los numerosos módulos tanto oficiales y no oficiales disponibles en su página de contenidos [35]. También destaca su facilidad de instalación y por proporcionar toda la seguridad del conocido protocolo SSH (del inglés *Secure Shell*) [36].
2. Saltstack [37]: Puede ser utilizado “sin agente” apoyándose en las mismas herramientas que Ansible (Python y SSH) aunque no sea su configuración por defecto. También hace uso del lenguaje YAML para la definición de las instrucciones a ejecutar. Sin embargo la opinión general es que no es tan sencillo el aprendizaje inicial, no llega a tener la misma variedad de módulos y la documentación disponible no está al mismo nivel [38], [39].
3. Chef [40]: Utiliza una arquitectura cliente-servidor y por tanto no es “sin agente”. El lenguaje utilizado para la creación de libros de instrucciones es JSON (del inglés *JavaScript Object Notation*) [41], similar al lenguaje YAML pero algo más complejo. Pese a que es un producto más maduro, su curva de aprendizaje es algo más lenta que en el caso de Ansible y Saltstack [42], [43].

Expuesto lo anterior, se decide optar por Ansible como herramienta de administración de servidores, principalmente debido a que su curva de aprendizaje es menor, hay multitud de documentación y módulos oficiales que extienden su funcionalidad y pueden ser aprovechados para nuestro propósito. Además, es una aplicación cuya instalación por defecto no requiere la instalación de ningún software en el servidor a configurar, y en cuanto a la seguridad que proporciona, aprovecha la fortaleza de las conexiones SSH con lo que además no es necesaria la apertura de puertos adicionales en el cortafuegos mientras el sistema sea accesible en el momento de la configuración vía SSH [44].

2.5. Herramientas de Análisis de la Seguridad.

Una vez se haya completado la instalación y configuración segura de todos los elementos (sistema operativo CentOS 7, servidor web Apache HTTP Server y base de datos MariaDB) mediante el uso de las herramientas que se han descrito en el apartado anterior, se procederá a evaluar el grado de seguridad que se ha conseguido.

De la misma manera en que hay disponibles sistemas operativos, servidores web y bases de datos de código abierto, también se desarrollan herramientas de análisis de seguridad o vulnerabilidades que se pueden encontrar de manera gratuita.

- OpenVAS [45]: es la utilidad de escaneo de vulnerabilidades de código abierto más conocida. Es una solución que combina numerosas utilidades para ofrecer una completa habilidad de escaneo y administración (solución) de vulnerabilidades. Tiene una base de datos de más de 50.000 NVTs (del inglés *Network Vulnerability Tests*).
- OpenSCAP [46]: similar a OpenVAS, es otra herramienta de auditoría, escaneo de vulnerabilidades y administración de las soluciones. Su fortaleza se basa en que está soportado por el NIST (del Inglés *National Institute of Standards and Technology*), que es una agencia que depende del gobierno de los Estados Unidos y de la empresa Red Hat.

Cabe destacar que estas dos herramientas son específicas para analizar vulnerabilidades en los sistemas operativos y los servicios que proporcionan, a diferencia de otras herramientas cuyo uso principal está más enfocado al análisis de vulnerabilidades de código o aplicación.

Ambas herramientas son recomendadas en la guía de seguridad de RHEL 7 [47]. Sin embargo OpenSCAP está enfocado a un usuario con altos conocimientos de seguridad, mientras que OpenVAS es más accesible y por tanto se utilizará OpenVAS para el análisis.

2.6. Aportación del Trabajo al Estado del Arte.

Tras el análisis del estado del arte en lo relativo al trabajo, puede afirmarse que hay multitud de herramientas de código abierto cuyos proyectos están en etapa de madurez y con un alto nivel de contribución, ya que cuentan con multitud de contribuyentes y son proyectos activos, con actualizaciones de código diarias.

Esto es positivo en cuanto a la seguridad que proporcionan, ya que la amplia comunidad de desarrolladores podrá proveer a los usuarios de parches de seguridad con regularidad mediante la liberación de actualizaciones.

Se han seleccionado diversos elementos de software con gran potencial sobre los que hay disponible una extensa documentación y ejemplos de configuración segura. Sin embargo, pese a disponer de toda esa información, es difícil que uno o varios administradores de sistemas sin conocimientos de seguridad puedan completar el objetivo de este trabajo, dada la gran dispersión de información y la, generalmente, escasa formación en seguridad de los equipos de sistemas y desarrollo en las empresas [48], [49] y [50].

Por lo tanto la aportación de este trabajo al estado del arte es la de sintetizar en un sólo documento toda esta información. Utilizándolo como referencia, un administrador de sistemas con unos mínimos conocimientos de seguridad podrá utilizarlo adaptando el desarrollo a sus necesidades, garantizando un nivel alto de seguridad.

3. Requisitos de Seguridad. Objetivos.

En este capítulo se tratarán de concretar los objetivos de seguridad que se desean conseguir tanto de una manera amplia como específica, para luego en el siguiente capítulo llevar a cabo la fase de investigación. Tras esta fase de investigación, se deberá obtener el detalle a nivel de configuración de las diferentes aplicaciones que hará que se cumpla dicho objetivo, y posteriormente se automatizará la solución.

3.1. Seguridad en el SO (Sistema Operativo).

Para la definición de objetivos de seguridad en lo referente al sistema operativo, se hará uso de la Guía de Seguridad de Red Hat Enterprise Linux 7 [47] (RHEL 7), ya que tal y como se indica en el apartado 2.1, RHEL 7 es la versión empresarial de CentOS 7 y por tanto la guía de seguridad de RHEL 7 es más completa que la de CentOS 7 [51], a la vez que compatible casi en su totalidad.

Objetivo 1. SO: Seguridad en la instalación del Sistema Operativo.

La seguridad en la instalación del Sistema Operativo deberá considerar lo siguiente:

1. **Seguridad relacionada con la BIOS (del inglés *Basic Input Output System*):**

    1.1. **Seguridad mediante contraseña de acceso**. Para evitar el acceso a la configuración de BIOS es recomendable proteger el mismo mediante una contraseña. De esta manera se evitará que pueda hacerse una configuración insegura de la BIOS (por ejemplo mediante la desactivación de los disipadores de calor o un borrado lógico de los discos por acceso a la controladora).

    1.2. **Seguridad UEFI** [52] (del inglés *Unified Extensible Firmware Interface*). La BIOS debe configurarse de manera que solamente pueda arrancar un Sistema Operativo en modo UEFI, y además intentar primero arrancar en modo seguro (del inglés *Secure Boot*). Mediante el modo UEFI seguro se comprueba que el arranque del sistema operativo no haya sido comprometido por algún *malware.*

Automatización.

Las anteriores configuraciones no pueden ser automatizadas fácilmente ya que son configuradas de manera presencial en el servidor físico. Debido a ello, pueden ser consideradas como seguridad física.

Existen herramientas proporcionadas por los fabricantes de hardware que permiten la automatización de la configuración de BIOS para la configuración masiva de servidores, pero están fuera del alcance de este trabajo.

2. Seguridad en el particionado de los discos y cifrado.

2.1. **Separación de directorios en particiones de disco.** Se recomienda crear particiones separadas para los directorios:

1) **`"/boot":`** Es la partición de arranque. No debe estar cifrada ni resultar corrupta o de lo contrario el sistema no arrancará.

2) **`"/tmp"`** y **`"/var/tmp":`** Ambos son utilizados para contener datos temporales o que pueden ser eliminados en cualquier momento. Se separan para no llenar otras particiones importantes que puedan provocar el fallo del sistema.

3) **`"/home":`** Se utiliza para contener datos de usuario en una partición separada de **`"/"`** para evitar su llenado, para evitar borrados accidentales por parte de los usuarios y por facilidad a la hora de actualizar el sistema operativo (el directorio `"/home"` queda intacto en las actualizaciones de SO).

Automatización.

Las anteriores particiones son creadas durante la instalación del sistema operativo, definidas mediante el *kickstart* ya mencionado anteriormente. Un ejemplo de particionado de un disco de 10GB sería:

```
part /boot --fstype=xfs --size=1024 --asprimary
part / --fstype=xfs --size=4096 --asprimary --encrypted  --cipher=aes-cbc-essiv:sha256 --passphrase=CifradoHDDs
part /tmp --fstype=xfs --size=512 --encrypted  --cipher=aes-cbc-essiv:sha256 --passphrase=CifradoHDDs
part /var/tmp --fstype=xfs --size=512 --encrypted  --cipher=aes-cbc-essiv:sha256 --passphrase=CifradoHDDs
part /home --fstype=xfs --size=1024 --encrypted  --cipher=aes-cbc-essiv:sha256 --passphrase=CifradoHDDs
part swap --recommended
```

3. **Instalación mínima.**

   Se recomienda la instalación del mínimo de elementos necesarios para el funcionamiento del sistema para evitar así posibles vulnerabilidades.

   Automatización

   La automatización de la configuración para una instalación mínima se hará mediante el fichero de *kickstart*.

4. **Restricción del acceso de red durante el proceso de instalación.**

   En el momento de la instalación es posible que se utilicen programas no actualizados o parcheados que puedan contener vulnerabilidades. Se recomienda por lo tanto no exponer estos sistemas a la red.

   Automatización

   La instalación se hará con el sistema desconectado de la red externa. Es tarea del administrador de sistemas asegurarse de que la red en la que está conectado el servidor a configurar no tenga acceso al exterior.

5. Procedimientos de post-instalación.

5.1. **Actualización del sistema operativo y todos sus paquetes a la última versión disponible**. Mediante el comando `"yum -y update"` se pueden actualizar todos los paquetes del sistema a la última versión disponible en los repositorios configurados. Deberá hacerse tras la puesta en seguridad, conectando el servidor a Internet.

5.2. **Asegurarse de que el servicio de cortafuegos está habilitado y arrancado.** Mediante el comando `"estemple start firewalld && systemctl enable firewalld"` se puede arrancar y habilitar el servicio de cortafuegos.

5.3. **Deshabilitar servicios que no vayan a ser utilizados.** Mediante el comando `"systemctl list-units | grep service"` se pueden ver los servicios arrancados o habilitados. Deshabilitaremos el servicio *postfix,* que por defecto viene activado, excluyendo el paquete en la instalación.

Automatización.

La automatización de los tres puntos anteriores la haremos tras la instalación del sistema utilizando Ansible:

```
- name: Actualizar SO y todos sus paquetes
  yum:
    name: '*'
    state: latest

- name: Habilitar y arrancar cortafuegos
  systemd:
    name: firewalld
    enabled: yes

- name: Deshabilitar postfix (mail server)
  systemd:
    name: postfix
    enabled: no
    state: stopped
```

Objetivo 2. SO: Seguridad en las Actualizaciones.

1. **El sistema y todos sus componentes deberán actualizarse regularmente para evitar riesgos de seguridad.** Esto se consigue ejecutando regularmente una actualización completa del sistema y sus aplicaciones, o bien aplicando los parches liberados por la comunidad ó las páginas oficiales.

   Automatización.

   En el caso de la actualización periódica del Sistema Operativo, y al tratarse de una operación que puede provocar la incompatibilidad de algunas configuraciones por cambios en las nuevas versiones, deberá hacerse regularmente (por ejemplo de forma semanal) de una forma manual, preferiblemente haciendo previamente una copia de seguridad del sistema. El comando a ejecutar es el que se ha mencionado anteriormente: `"yum update"`.

Objetivo 3. SO: Herramientas y Servicios para la Puesta en Seguridad.

Deben considerarse los siguientes elementos a configurar:

1. **Control de accesos de superusuario (*root*).** Se escoge no desactivar por completo el uso de *root* en el servidor para evitar problemas en su gestión por parte de usuarios poco experimentados. Se dará permiso al usuario "**admin1**" para escalar sus privilegios a *root*. De esta manera quedará constancia de todos los intentos de escalación de privilegios, reflejando fecha y hora. Se escoge no compartir la clave de *root* con el usuario "**admin1**" para mayor seguridad. Se establecerán políticas de caducidad de la contraseña y se eliminará el tiempo de gracia del comando sudo.

   Automatización.

   Se creará y configurará el usuario "admin1" y se harán las demás configuraciones mediante Ansible:

```
- user:
    name: admin1
    group: wheel
    password:
$6$ZTmiXaN27kI1.E9S$4sf98ep6PnFY9vTwowXC7Vep4zQLAT8.rMR/4LMiSPvZaH
0p1L1fFuMeK3PjIf2tXKrLwP4s8IUnqpn/aApls1

- shell: chage -W 20 -M 30 admin1

- lineinfile:
    path: /etc/sudoers
    state: present
    backup: yes
    line: 'Defaults    timestamp_timeout=0'
```

   NOTA: El valor introducido en "Password:" es un *hash* SHA-512 de la contraseña real del usuario "admin1" que será "UniR2018"**.**

2. **Seguridad en el arranque del Sistema Operativo.** Se configurará el gestor de arranque del sistema operativo "*grub*" para que requiera la introducción de una contraseña para editar las opciones de arranque. De esta forma se evita que un atacante pueda tener acceso y arrancar en modo "*single*", obteniendo acceso inmediato como *root* a la máquina.

   Automatización

   Será necesario calcular el *hash* de la contraseña primero. Se utilizará la contraseña "UniRgrubPWD" Desde la línea de comandos de otro sistema Centos 7 se ejecuta:

```
[root@centos7master ~]# grub2-mkpasswd-pbkdf2
Introduzca la contraseña:
Reintroduzca la contraseña:
El hash PBKDF2 de su contraseña es
grub.pbkdf2.sha512.10000.8093B341D210BEE1CE06866E98BAA4884A8FDA469
4B6A0C5C3D0EBE32412039D6ED7E36CEAEEFFDB10D90EE87A535D15BF433
93ADE71F875124BD37EE922CFC5.3DF35D9DBE7E39FAECE5DE8CC24028624
069CFF186DFB0ADBD0F4DAFD1DC2FB163DF6AB0E3DE23460BA2905A90480
97492B746AE2A29A040D677B02E834B682D
```

3. Y el *hash* anterior será utilizado para automatizar la configuración incluyendo la siguiente línea en el *kickstart*:

```
bootloader --iscrypted
--password=grub.pbkdf2.sha512.10000.8093B341D210BEE1CE06866E98BAA488
4A8FDA4694B6A0C5C3D0EBE32412039D6ED7E36CEAEEFFDB10D90EE87A53
5D15BF43393ADE71F875124BD37EE922CFC5.3DF35D9DBE7E39FAECE5DE8C
C24028624069CFF186DFB0ADBD0F4DAFD1DC2FB163DF6AB0E3DE23460BA2
905A9048097492B746AE2A29A040D677B02E834B682D
```

4. **Seguridad en los servicios instalados y uso de protocolos seguros.** No deberán haber servicios habilitados y en desuso. Además se utilizarán protocolos seguros como SSL (del inglés *Secure Socket Layer*) y TLS (del inglés *Transport Layer Security*) cuando sea posible.

   Automatización

   Se hará una instalación en la que por defecto se prohíban todas las comunicaciones, solamente permitiendo aquellas que realicen conexiones seguras. Esto se configura habilitando el servicio de *firewall* en el *kickstart*.

5. **Seguridad en el acceso por red y cortafuegos**. Guarda relación con el punto anterior en cuanto a que un servicio es normalmente utilizado para acceder a un recurso por la red. Se configurará el cortafuegos para controlar que los servicios no sean accesibles de forma no controlada. Para ello se usará *firewalld* que es el cortafuegos del sistema.

   Por defecto, el cortafuegos está habilitado y las políticas son restrictivas, por lo que habrá que añadir el servicio de servidor web HTTPS. De lo contrario las conexiones serán bloqueadas por el cortafuegos.

   Automatización

   La automatización se consigue mediante la línea `"firewall --enabled --service=https"` en el *kickstart*.

6. **Seguridad en las consultas DNS** (del inglés *Domain Name System*). Para evitar vulnerabilidades relacionadas con la resolución de nombres, se hará uso de DNSSEC para hacer la resolución de nombres segura. Se instalará la utilidad *unbound* y se configurará para que las consultas DNS se hagan contra el propio servidor (*localhost*).

   Automatización

   Se configurará *localhost (127.0.0.1)* como servidor DNS mediante el *kickstart*:

```
# Configuración del interfaz de red con la resolución de nombres

network --bootproto=static --device=eth0 --ip=192.168.124.20
--netmask=255.255.255.0 --gateway=192.168.124.1 --nameserver=127.0.0.1
--activate
```

Se hará uso de Ansible para instalar, habilitar y arrancar *unbound*:

```
# Instalación y activación de unbound para DNSSEC

- name: install the latest version of unbound
  yum:
    name: unbound
    state: latest

- name: Habilitar y arrancar unbound (DNSSEC)
  systemd:
    name: unbound
    enabled: yes
```

Objetivo 4. SO: Auditoría del Sistema.

La auditoría de eventos en los logs del sistema es un elemento fundamental para la detección y corrección de posibles incidentes de seguridad. Es importante establecer una buena configuración de *logging* de los eventos que se pueden considerar relevantes en lo referente a la seguridad. La auditoría no proporciona mayor seguridad pero sí es una herramienta para la mitigación de incidentes de seguridad.

El sistema de auditoría está instalado por defecto en Centos 7 y registra la mayoría de eventos de seguridad considerados relevantes. Además, tiene disponibles varios perfiles de seguridad preconfigurados que cumplen varios estándares de certificación (STIG, PCI-DSS y NIS entre otros).

Automatización.

Como ya se ha dicho, el sistema de auditoría está instalado y se ejecuta por defecto. Aprovechando que hay disponibles algunas reglas pre-configuradas se va a utilizar la configuración incluida en el fichero `"30-nispom.rules"` que está preparado para sistemas que cumplen la certificación del NIS. Para ello se han de copiar además los ficheros `"10-base-config.rules"` y `"99-finalize.rules"` desde la ruta `"/usr/share/doc/audit-2.6.5/rules/"` a `"/etc/audit/rules.d/"`. Esto se hará mediante Ansible:

```
- debug:
    msg: "Aplicando configuracion de Auditoría del Sistema"
- copy:
    src: /usr/share/doc/audit-2.6.5/rules/10-base-config.rules
    dest: /etc/audit/rules.d/10-base-config.rules
- copy:
    src: /usr/share/doc/audit-2.6.5/rules/99-finalize.rules
    dest: /etc/audit/rules.d/99-finalize.rules
- copy:
    src: /usr/share/doc/audit-2.6.5/rules/30-nispom.rules
    dest: /etc/audit/rules.d/30-nispom.rules
```

3.2. Seguridad en el SW (Servidor Web).

Para la definición de objetivos de seguridad en lo referente al servidor web Apache HTTP Server, se hará uso de la página web oficial [13], donde se encuentran las recomendaciones de configuración segura de la aplicación, más en concreto en la sección “Consejos de Seguridad” [53].

Objetivo 5. SW: Protección contra ataques DoS.

Los ataques DoS (del inglés *Denial of Service*) son muy comunes en el ámbito de los servidores web. Un ataque DoS pretende saturar de peticiones un servidor desde uno o varios orígenes mediante el uso de algún programa de tipo *malware* (software malicioso) o *botnet* (red de robots informáticos). Al enviar tal número de peticiones, el servidor presenta lentitud y puede llegar a fallar, generando un error interno en el servidor.

La protección contra ataques DoS o DDoS (del inglés *Distributed Denial of Service*) es un elemento muy importante para una eficaz configuración segura.

En la guía de seguridad de Apache se recomienda configurar ciertas directivas que pueden ayudar a mitigar problemas, como:

- *RequestReadTimeout*: limita el tiempo de espera hasta que un cliente envía una petición y el mínimo de datos enviados para no cortar la conexión.
- *TimeOut*: establece el tiempo de espera hasta devolver un error.
- *KeepAliveTimeout*: establece el tiempo de espera entre peticiones de mantener la conexión abierta (petición keepalive).
- *LimitRequestBody*, *LimitRequestFields*, *LimitRequestFieldSize*, *LimitRequestLine*, *LimitXMLRequestBody* limitan el consumo de recursos por parte de los clientes. Todos los valores anteriores tienen establecido un valor por defecto determinado menos *LimitRequestBody*, por lo que se ha de establecer uno, o de lo contrario el valor por defecto es "ilimitado".
- *AcceptFilter*: descarga parte de la carga de procesado al sistema operativo.
- Usar "MPM" (del inglés *Multi-Processing Module*) para poder gestionar más conexiones simultáneas.

Algunas de las configuraciones anteriores vienen activadas por defecto, como por ejemplo *AcceptFilter* y el uso de un MPM multihilo. Otras son configurables y dependerán de los recursos físicos del servidor donde se instale Apache. Para un ajuste fino del rendimiento del servidor, se recomienda visitar la documentación oficial, concretamente la sección "*Apache Performance Tuning*" [54]. Para una configuración generalizada se va a parametrizar el fichero "`/etc/httpd/conf/httpd.conf`" por debajo de los valores por defecto, por ejemplo:

```
RequestReadTimeout header=10-20,MinRate=500 body=10,MinRate=500
TimeOut 30
KeepAliveTimeout 3
LimitRequestBody 512000
```

Automatización.

Se incluirán los parámetros necesarios en el fichero de configuración mediante Ansible:

```
- blockinfile:
    path: /etc/httpd/conf/httpd.conf
    block: |
      RequestReadTimeout header=10-20,MinRate=500 body=10,MinRate=500
      TimeOut 30
      KeepAliveTimeout 3
      LimitRequestBody 512000
```

Objetivo 6. SW: Permisos en Archivos y Directorios del Servidor.

Los archivos y directorios del servidor deben tener los permisos apropiados para que en el caso de comprometerse la seguridad de la aplicación de servidor web (httpd), el atacante no pueda acceder a dichos archivos con los privilegios con los que cuenta el usuario con el que se ejecuta el proceso httpd.

Es importante por tanto que ejecutables, archivos de configuración y directorios sean solamente modificables por el usuario root. Esta es la configuración por defecto, pero en caso, por ejemplo, de cambiar la directiva *ServerRoot* habría que asegurarse de que los directorios a los que apunta son solamente modificables por root.

Automatización.

No es necesario realizar ninguna acción ya que se mantendrán las configuraciones por defecto.

Objetivo 7. SW: Permisos de Navegación por el Sistema de Ficheros.

Debe garantizarse que el usuario apache (al que *root* transfiere el proceso httpd tras arrancarlo) no pueda acceder a una ruta del sistema de ficheros que esté por encima del directorio que se ha configurado como *ServerRoot* (directorio raíz del servidor web), que es donde se ubican los ficheros propios del servidor web.

Por ejemplo: el *ServerRoot* suele estar configurado por defecto en la ruta absoluta `"/var/www/html"`. Por tanto, un usuario del servidor web no debería poder acceder a la ruta `"/var/www"`. Para evitar lo anterior, deberá protegerse el servidor contra el uso de enlaces simbólicos.

La configuración a aplicar por tanto será:

- FollowSymLinks: desactivar, evita que se puedan utilizar enlaces simbólicos.
- Indexes: desactivar, evita que los directorios sean navegables.

Automatización.

Se modificará el fichero de configuración `"/etc/httpd/conf/httpd.conf"`

para eliminar estas dos directivas utilizando Ansible:

```
- replace:
    path: /etc/httpd/conf/httpd.conf
    regexp: '(FollowSymLinks|Indexes)'
    replace: ''
    backup: yes
```

Objetivo 8. SW: Ocultar Información de la Versión del Servidor.

Es importante no dar información a los posibles atacantes acerca de la versión del servidor que se está utilizando en las cabeceras de las respuestas HTTP, ya que esta puede ser usada para explotar vulnerabilidades conocidas en dicha versión. Para ello, en la guía de seguridad de RHEL 7 [47] se recomienda configurar la directiva *ServerTokens* con el parámetro "Prod", de manera que solamente se devuelva el producto (Apache)

Automatización.

Se automatizará añadiendo la directiva al archivo de configuración mediante Ansible:

```
- lineinfile:
    path: /etc/httpd/conf/httpd.conf
    state: present
    line: 'ServerTokens Prod'
```

Objetivo 9. SW: Examinar los Logs.

Una de las tareas más importantes en lo que respecta a la seguridad de un servidor web es la revisión de logs, no solamente cuando se produce un incidente de seguridad sino de manera preventiva.

Examinando los logs de acceso y de error se pueden encontrar huellas de intentos de explotación de vulnerabilidades, intentos de acceso a zonas de administración por fuerza bruta, o IPs de origen las cuales añadir a la lista negra en el cortafuegos del sistema.

Automatización.

Pese a que existen herramientas de monitorización automáticas de los logs que analizan patrones (como AWStats ó GoAccess), están fuera del alcance de este documento. Sin embargo se recomienda su estudio y utilización para administradores de sistemas con conocimientos más avanzados.

Objetivo 10. SW: Acceso Mediante Protocolos Seguros y Cifrados Robustos.

Se deberá limitar el acceso web inseguro al servidor (HTTP) para que esté disponible solamente vía HTTPS. Además, se deberá configurar para hacer uso solamente de cifrados seguros y no obsoletos. Una posible fuente de información al respecto es la que pone disponible Mozilla en la URL https://mozilla.github.io/server-side-tls/ssl-config-generator/, donde se permite seleccionar la versión de servidor Apache (obtener mediante el comando `"httpd -v"`) y OpenSSL (obtener mediante el comando `"openssl version"`), y devuelve el bloque de configuración a implementar como se puede observar en la Figura 6.

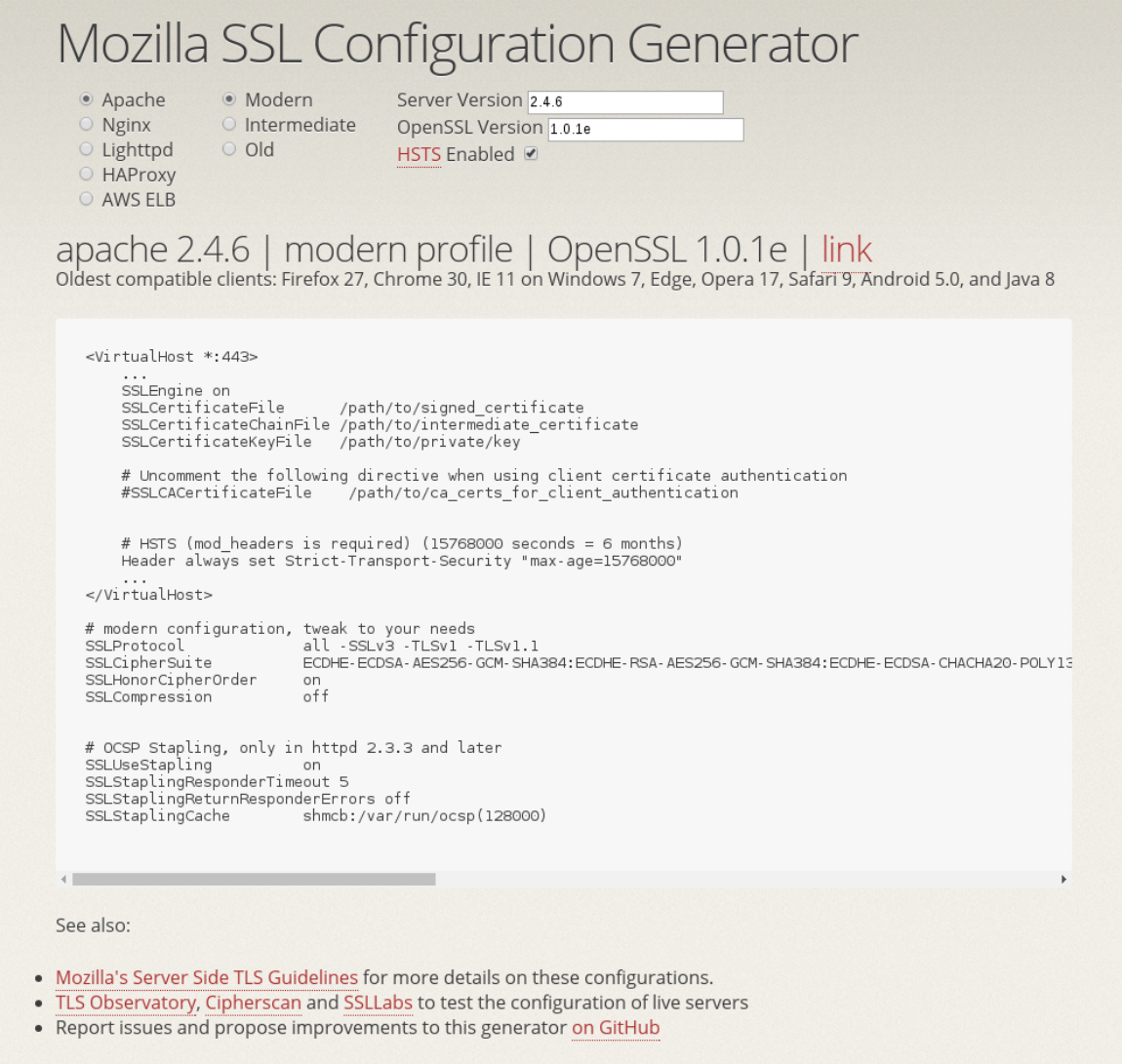


Figura 6: Generador de configuración de seguridad SSL. Fuente: Mozilla.

Automatización.

Antes de nada es necesario instalar el módulo SSL de Apache:

```
- name: Instalar ultima version del modulo ssl de Apache
  yum:
    name: mod_ssl
    state: latest
```

Para generar los certificados del servidor [55] que se han de configurar en el 'VirtualHost':

```
- stat:
    path: "{{ httpd_ssl_folder }}/private/ca.key"
  register: cakey

- name: Crear clave privada de la CA
  shell: openssl genrsa 2048 > {{ httpd_ssl_folder }}/private/ca.key
  when: cakey.stat.exists == False

- stat:
    path: "{{ httpd_ssl_folder }}/private/ca.csr"
  register: cacsr

- name: Crear CSR (peticion de certificado firmado) con la clave de la CA.
  shell: openssl req -new -key {{ httpd_ssl_folder }}/private/ca.key -out
{{ httpd_ssl_folder }}/private/ca.csr -subj '/C=ES/ST=ES/L=Valencia/O=My
Company/OU=IT Department/CN=HTTPD Server'
  when: cacsr.stat.exists == False

- stat:
    path: "{{ httpd_ssl_folder }}/certs/ca.crt"
  register: cacrt

- name: Crear certificado del servidor firmado con la clave de la CA
  shell: openssl x509 -req -days 365000 -in {{ httpd_ssl_folder }}/private/ca.csr -signkey
{{ httpd_ssl_folder }}/private/ca.key -out {{ httpd_ssl_folder }}/certs/ca.crt
  when: cacrt.stat.exists == False
```

Se deberá crear un fichero llamado `"01-vhost.conf"` con la configuración segura del servidor web que se extrae de los pasos anteriores. Se copiará dicho fichero a la ubicación `"/etc/httpd/conf.d/"` mediante Ansible.

```
- blockinfile:
    path: /etc/httpd/conf.d/01-vhost.conf
    create: yes
    block: |
          # modern configuration, tweak to your needs
          SSLProtocol             all -SSLv3 -TLSv1 -TLSv1.1
          SSLCipherSuite          ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-
AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-
CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-
GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256
          SSLHonorCipherOrder     on
          SSLCompression          off
          SSLSessionTickets       off

          # OCSP Stapling, only in httpd 2.3.3 and later
          SSLUseStapling          on
          SSLStaplingResponderTimeout 5
          SSLStaplingReturnResponderErrors off
          SSLStaplingCache        shmcb:/var/run/ocsp(128000)

           <VirtualHost *:443>
                SSLEngine on
                SSLCertificateFile /etc/pki/tls/certs/ca.crt
                SSLCertificateKeyFile /etc/pki/tls/private/ca.key
                        # HSTS (mod_headers is required) (15768000 seconds = 6 months)
                        Header always set Strict-Transport-Security "max-age=15768000"
                        DocumentRoot /var/www/html
                ServerName yoursite.com
           </VirtualHost>
```

Objetivo 11. SW: Actualizaciones.

Deberá mantenerse el servidor actualizado de manera que se solucionen las posibles vulnerabilidades aplicando los parches que proporcionan dichas actualizaciones.

Automatización.

En el caso de la actualización periódica del Servidor Web, al tratarse de una operación que puede provocar la incompatibilidad de algunas configuraciones por cambios en las nuevas versiones deberá hacerse regularmente de una forma manual. El comando a ejecutar es: `"yum update httpd".`

3.3. Seguridad en la BD (Base de Datos).

Para determinar los objetivos de seguridad en lo referente a la BD se hará uso de la guía de seguridad que publica MariaDB [56].

Objetivo 12. BD: Seguridad en la instalación de la Base de Datos.

Uno de los primeros pasos a la hora de instalar MariaDB es el de hacer una puesta en seguridad de la instalación usando el *script* `"mysql_secure_installation"`. Este script se encarga de:

- Establecer una contraseña para el usuario root de MariaDB.
- Permitir sólo login local (No permitir acceso por red).
- Eliminar cuentas de usuario anónimas.
- Eliminar la base de datos de test.

Automatización.

Para automatizar esta configuración se reutilizará código Ansible escrito por la comunidad, como por ejemplo el disponible en [57]. Se harán algunos cambios con respecto al código quedando como sigue:

```
- debug: msg="Instalando y Configurando MariaDB"

- name: Instalando MariaDB
  yum: name=mariadb-server state=present

- name: Instalando python-mysqldb para usar Ansible
  yum: name=MySQL-python state=present

- debug: msg="Creando password de usuario root de MariaDB"
  when: mysql_root_password == 'M@ri@DBrootPWD'

- name: Password de root para acceso desde localhost
  mysql_user: name=root host={{item}} password={{mysql_root_password | mandatory}}
state=present
  with_items:
    - '::1'
    - '127.0.0.1'
    - 'localhost'

- name: Eliminando base de datos de test
  mysql_db: name=test state=absent

- name: Eliminando usuarios anónimos
  mysql_user: name='' state=absent host_all=yes
```

Objetivo 13. BD: Conexiones Seguras con la Base de Datos.

Las comunicaciones con la base de datos se hacen por defecto usando texto plano, por lo que un atacante con acceso a las comunicaciones podría robar información. Para evitarlo, MariaDB puede configurarse para que toda comunicación con el servidor se haga cifrada. Para ello se ha de modificar el archivo de configuración del servicio `mariadb.service`, añadiendo el parámetro `"--ssl"` y para mayor seguridad, se añadirá también `"--ssl_cipher=TLSv1.2"`. Además hay que crear los certificados que habrá que configurar posteriormente para ser usados en las conexiones seguras. Desde la página oficial de MariaDB se recomienda utilizar la guía [58].

Automatización.

Se realizará la configuración mediante Ansible:

```
- replace:
    path: /usr/lib/systemd/system/mariadb.service
    regexp: '--basedir=/usr'
    replace: '--basedir=/usr --ssl --ssl_cipher=TLSv1.2'
    backup: yes
```

Para crear los certificados:

```
vars:
  mariadb_ssl_folder: /var/lib/mysql/ssl

- name: Creando directorio que contiene los certificados de MariaDB
  file:
    path: {{ mariadb_ssl_folder }}
    state: directory
    mode: 0755
    owner: mysql
    group: mysql

- name: Crear clave privada de la CA
  shell: openssl genrsa 2048 > {{ mariadb_ssl_folder }}/ca-key.pem

- name: Crear certificado con la clave de la CA
  shell: openssl req -new -x509 -nodes -days 365000 -key {{ mariadb_ssl_folder }}/ca-
key.pem -out {{ mariadb_ssl_folder }}/ca-cert.pem -subj '/C=ES/ST=ES/L=Valencia/O=My
Company/OU=IT Department/CN=MariaDB CA Admin'

- name: Crear certificado del servidor
  shell: openssl req -newkey rsa:2048 -days 365000 -nodes -keyout
{{ mariadb_ssl_folder }}/server-key.pem -out {{ mariadb_ssl_folder }}/server-req.pem -subj
'/C=ES/ST=ES/L=Valencia/O=My Company/OU=IT Department/CN=MariaDB Server'

- name: Procesar clave RSA
  shell: openssl rsa -in {{ mariadb_ssl_folder }}/server-key.pem -out
{{ mariadb_ssl_folder }}/server-key.pem

- name: Firmar certificado del servidor
  shell: openssl x509 -req -in {{ mariadb_ssl_folder }}/server-req.pem -days 365000 -CA
{{ mariadb_ssl_folder }}/ca-cert.pem -CAkey {{ mariadb_ssl_folder }}/ca-key.pem -set_serial
01 -out {{ mariadb_ssl_folder }}/server-cert.pem

- name: Crear certificado cliente
  shell: openssl req -newkey rsa:2048 -days 365000 -nodes -keyout
{{ mariadb_ssl_folder }}/client-key.pem -out {{ mariadb_ssl_folder }}/client-req.pem -subj
'/C=ES/ST=ES/L=Valencia/O=My Company/OU=IT Department/CN=MariaDB User'

- name: Procesar clave RSA
  shell: openssl rsa -in {{ mariadb_ssl_folder }}/client-key.pem -out
{{ mariadb_ssl_folder }}/client-key.pem
```

```
- name: Firmar certificado del servidor
  shell: openssl x509 -req -in {{ mariadb_ssl_folder }}/client-req.pem -days 365000 -CA
{{ mariadb_ssl_folder }}/ca-cert.pem -CAkey {{ mariadb_ssl_folder }}/ca-key.pem -set_serial
01 -out {{ mariadb_ssl_folder }}/client-cert.pem

- name: Verificar certificados creados. Aborta si no es correcto.
  shell: openssl verify -CAfile {{ mariadb_ssl_folder }}/ca-cert.pem
{{ mariadb_ssl_folder }}/server-cert.pem {{ mariadb_ssl_folder }}/client-cert.pem
  register: ssl_verification_output
  failed_when: ssl_verification_output.rc != 0
```

Al finalizar la ejecución de lo anterior estarán disponibles los certificados de cliente y servidor en la ruta `"/var/lib/mysql/ssl"` y faltará configurar MariaDB para utilizarlos:

```
- blockinfile:
    path: /etc/my.cnf.d/server.cnf
    insertafter: '^\[mysqld\]'
    block: |
      ssl-ca={{ mariadb_ssl_folder }}/ca-cert.pem
      ssl-cert={{ mariadb_ssl_folder }}/server-cert.pem
      ssl-key={{ mariadb_ssl_folder }}/server-key.pem
```

Objetivo 14. BD: Actualizaciones

Deberá mantenerse el servidor actualizado de manera que se solucionen las posibles vulnerabilidades aplicando los parches que proporcionan dichas actualizaciones.

Automatización.

En el caso de la actualización periódica de la Base de Datos, al tratarse de una operación que puede provocar la incompatibilidad de algunas configuraciones por cambios en las nuevas versiones deberá hacerse regularmente de una forma manual. El comando a ejecutar es: `"yum update mariadb-server"`.

4. Desarrollo de la metodología y evaluación.

4.1. Desarrollo de la metodología.

Durante la elaboración del Capítulo 3 se han recopilado distintas configuraciones para realizar la instalación y puesta en seguridad de forma automática. Para ello se hará uso de dos herramientas: el fichero de *kickstart* y *playbooks* de Ansible.

Desde un punto de vista práctico es interesante para el administrador de sistemas poder modificar algunas de las configuraciones que se proponen tanto en el fichero de *kickstart* como en los *playbooks* de Ansible. Por ello, se propone el uso de un servidor auxiliar desconectado de la red, desde el cual servir los ficheros necesarios para automatizar la instalación vía HTTP, como se explica en la Figura 7.

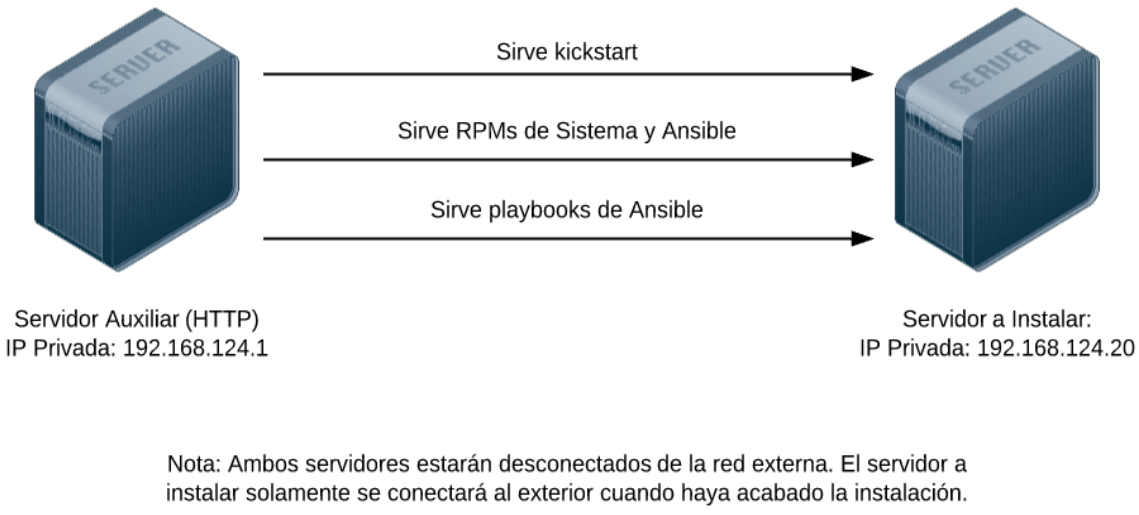
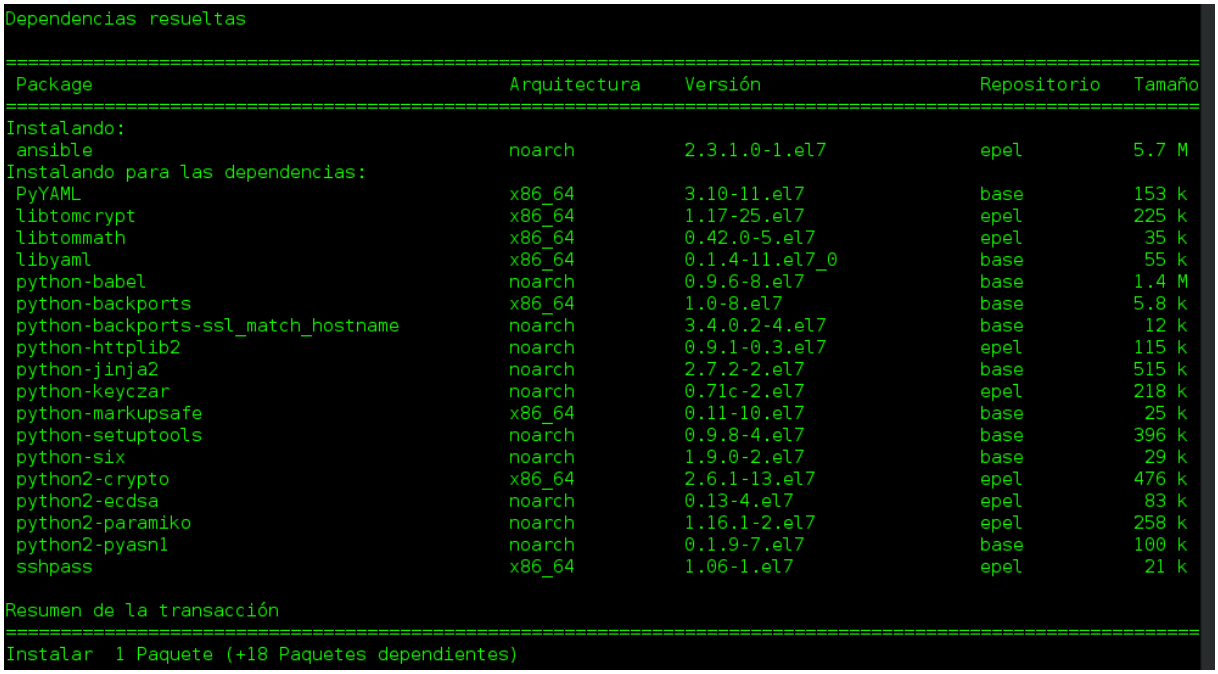


Figura 7: Servidor Auxiliar

La instalación del servidor auxiliar con un servidor web desconectado de Internet queda en manos del administrador de sistemas, ya que es una operación sencilla y común. Se describirán solamente las acciones necesarias para que sirva el *kickstart*, *playbooks* de Ansible y los paquetes de instalación o RPMs. Para esto último, se puede obtener un listado de dependencias necesarias de Ansible ejecutando la instrucción `"yum install ansible"` desde un sistema con el paquete "epel-release" instalado, como se muestra en la Figura 8:



```
Dependencias resueltas

================================================================================
 Package                                Arquitectura    Versión                 Repositorio   Tamaño
================================================================================
Instalando:
 ansible                                noarch          2.3.1.0-1.el7           epel          5.7 M
Instalando para las dependencias:
 PyYAML                                 x86_64          3.10-11.el7             base          153 k
 libtomcrypt                            x86_64          1.17-25.el7             epel          225 k
 libtommath                             x86_64          0.42.0-5.el7            epel           35 k
 libyaml                                x86_64          0.1.4-11.el7_0          base           55 k
 python-babel                           noarch          0.9.6-8.el7             base          1.4 M
 python-backports                       x86_64          1.0-8.el7               base          5.8 k
 python-backports-ssl_match_hostname    noarch          3.4.0.2-4.el7           base           12 k
 python-httplib2                        noarch          0.9.1-0.3.el7           epel          115 k
 python-jinja2                          noarch          2.7.2-2.el7             base          515 k
 python-keyczar                         noarch          0.71c-2.el7             epel          218 k
 python-markupsafe                      x86_64          0.11-10.el7             base           25 k
 python-setuptools                      noarch          0.9.8-4.el7             base          396 k
 python-six                             noarch          1.9.0-2.el7             base           29 k
 python2-crypto                         x86_64          2.6.1-13.el7            epel          476 k
 python2-ecdsa                          noarch          0.13-4.el7              epel           83 k
 python2-paramiko                       noarch          1.16.1-2.el7            epel          258 k
 python2-pyasn1                         noarch          0.1.9-7.el7             base          100 k
 sshpass                                x86_64          1.06-1.el7              epel           21 k

Resumen de la transacción
================================================================================
Instalar  1 Paquete (+18 Paquetes dependientes)
```

Figura 8: Listado de Dependencias de Ansible

Todos los paquetes anteriores deben estar disponibles en el servidor auxiliar para poder hacer la instalación sin conexión.

Dichos RPMs están disponibles en las URLs:

- https://fedoraproject.org/wiki/EPEL
- http://mirror.centos.org/centos/7

Suponiendo el servidor HTTP auxiliar haya sido instalado con la configuración de directorios por defecto, el siguiente paso es crear un repositorio en dicho servidor desde donde poder ofrecer los RPMs anteriormente nombrados al servidor que está siendo instalado.

Para ello habrá que descargar los RPMs listados anteriormente para posteriormente copiarlos en un directorio público del servidor auxiliar, por ejemplo `"/var/www/html/pub/repo_aux_ansible"`.

Una vez copiados allí, solamente habría que ejecutar la instrucción `"createrepo /var/www/html/pub/repo_aux_ansible"` en el servidor auxiliar para crear el repositorio, y la configuración del mismo en el *kickstart*.

Además, para que la instalación se haga de manera que el servidor esté completamente desconectado de la red externa, ha de descargarse la imagen ISO del DVD completo ("*Everything ISO*") de CentOS 7 de [59]. Una vez descargado, se copiará todo el contenido del DVD al servidor auxiliar, por ejemplo en la ruta `"/var/www/html/pub/CentOS-everything"` y se ejecutará la instrucción `"createrepo /var/www/html/pub/CentOS-everything"` para generar los metadatos del repositorio.

El fichero de *kickstart* quedaría de la siguiente manera:

```
# Configuracion de idioma y lenguaje
keyboard --vckeymap=es --xlayouts='es'
lang es_ES.UTF-8

# Configuracion de zona horaria
timezone Europe/Madrid --isUtc

# Configuracion de red
network  --bootproto=static --device=ens3 --gateway=192.168.124.1 --ip=192.168.124.20
--nameserver=127.0.0.1 --netmask=255.255.255.0 --ipv6=auto --activate
network  --hostname=secure-server

# Pass de root cifrado
rootpw $1$ZnxmHeBf$IppMjEcBN.6E5RN0qbvET/ --iscrypted

# Opciones generales
auth --passalgo=sha512 --useshadow
selinux --enforcing
firewall --enabled
skipx
firstboot --disable
reboot

# Repositorio Auxiliar
repo --name=repo_aux_ansible --baseurl=http://192.168.124.1/pub/repo_aux_ansible
repo --name=CentOS-everything --baseurl=http://192.168.124.1/pub/CentOS-everything

# Opciones de bootloader
bootloader --append="rhgb quiet crashkernel=auto" --iscrypted
--password=grub.pbkdf2.sha512.10000.8A4D0E0C6C8AA68D8E8B9EC746EE10EA3A7B7
A11D50ED82C1469EBB2E1C638D27398A67070CBC2D59F22CDF758EFDBC1177BBCA
BAB09F2A98BAA01002C1569B8.D82D406A66F5544BC4BDD2A0A77284DA78BFA6544
9FD61BAA5C5944E2AC1AAD7FE8F083C25FF393F9E30147C97942F5527E7FD35547F
BB9C7F3B89B1FD79141B
zerombr
```

```
# Particionado de discos
clearpart --all --initlabel
part /boot --fstype=xfs --size=1024 --asprimary
part / --fstype=xfs --size=4096 --asprimary --encrypted  --cipher=aes-cbc-essiv:sha256 --passphrase=CifradoHDDs
part /tmp --fstype=xfs --size=512 --encrypted  --cipher=aes-cbc-essiv:sha256 --passphrase=CifradoHDDs
part /var/tmp --fstype=xfs --size=512 --encrypted  --cipher=aes-cbc-essiv:sha256 --passphrase=CifradoHDDs
part /home --fstype=xfs --size=1024 --encrypted  --cipher=aes-cbc-essiv:sha256 --passphrase=CifradoHDDs
part swap --recommended

# Paquetes a instalar: instalacion minima
%packages
@^minimal
@core
chrony
kexec-tools
ansible
-postfix
%end

%post --log=/root/ansible_post.log
# Configuramos repositorio privado
cat > /etc/yum.repos.d/Centos-everything.repo <<EOF
[Centos-everything]
name=Centos-everything
baseurl=http://192.168.124.1/pub/CentOS-everything
gpgcheck=0
enabled=1
EOF

# Establecemos el inventario donde se ejecuta ansible (localhost)
echo "localhost ansible_connection=local" >> /etc/ansible/hosts

# Creamos directorio de Ansible, descargamos playbooks, ejecutamos Ansible y eliminamos los ficheros.
mkdir /root/ansible
cd /root/ansible
curl -O http://192.168.124.1/pub/ansible/ansible.tgz
tar -xvf ansible.tgz
ansible-playbook main.yml
rm -rf /root/ansible
```

```
# Creamos servicio firstboot que se ejecuta tras reinicio para acabar de configurar mariadb (el servicio ha de estar ejecutandose)
cd /root
curl -O http://192.168.124.1/pub/ansible/ansible_firstboot.sh
chmod +x /root/ansible_firstboot.sh

cat > /etc/systemd/system/ansible_firstboot.service <<EOF
[Unit]
Description=Servicio para acabar configuracion tras reinicio
After=mariadb.service multi-user.target

[Service]
ExecStart=/root/ansible_firstboot.sh

[Install]
WantedBy=multi-user.target
EOF

systemctl enable ansible_firstboot.service

%end
```

Este fichero ha de estar disponible en el servidor auxiliar, por ejemplo en la ruta: `"/var/www/html/ks.cfg"` de forma que al iniciar la instalación del servidor seguro esté disponible para su uso.

Por otra parte, los *playbooks* de Ansible se estructuran de la siguiente manera:

```
ansible/
├── main.yml
└── roles
    └── secure-server
        └── tasks
            ├── audit.yml
            ├── create_user.yml
            ├── dnssec.yml
            ├── httpd_certs.yml
            ├── httpd_install.yml
            ├── httpd_vhost.yml
            ├── main.yml
            ├── mariadb_certs.yml
            └── mariadb_install.yml
```

Para aquellos que no estén familiarizados con Ansible, esta es la estructura de proyecto recomendada en la guía oficial [60], según la cual se invocará al *playbook* mediante el comando "`ansible-playbook ansible/main.yml`". Dentro de este se definen algunas variables (como rutas ó passwords), y se invocan los roles donde aplicar los cambios. En el caso de este trabajo solamente se ha definido el rol "*secure-server*". Finalmente la ejecución leerá el fichero "`ansible/roles/secure-server/tasks/main.yml`" que incluye a todos los demás de manera organizada.

Para ser utilizados durante el proceso de instalación, se comprimirán los ficheros y se copiarán en la ruta "`/var/www/html/pub/util/ansible/ansible.tgz`", fichero que descargará el sistema durante la instalación vía *kickstart* para posteriormente ejecutar Ansible y configurar el servidor.

Dada la extensión de estos *playbooks*, se dejan a disposición del lector en el Anexo I.

El proceso de instalación consta finalmente de las siguientes etapas:

1. Inicio de la instalación del SO. El servidor a instalar utiliza el *kickstart* que le proporciona el servidor auxiliar. Ha de ser pasado como parámetro en la pantalla inicial de instalación pulsando la tecla "Tab" sobre la opción "*Install CentOS Linux 7*" dejando un espacio detrás del texto ya presente, escribiendo: `"ks=http://192.168.124.1/ks.cfg"` (ver Figura 9) y posteriormente se pulsa *Intro* para comenzar la instalación.

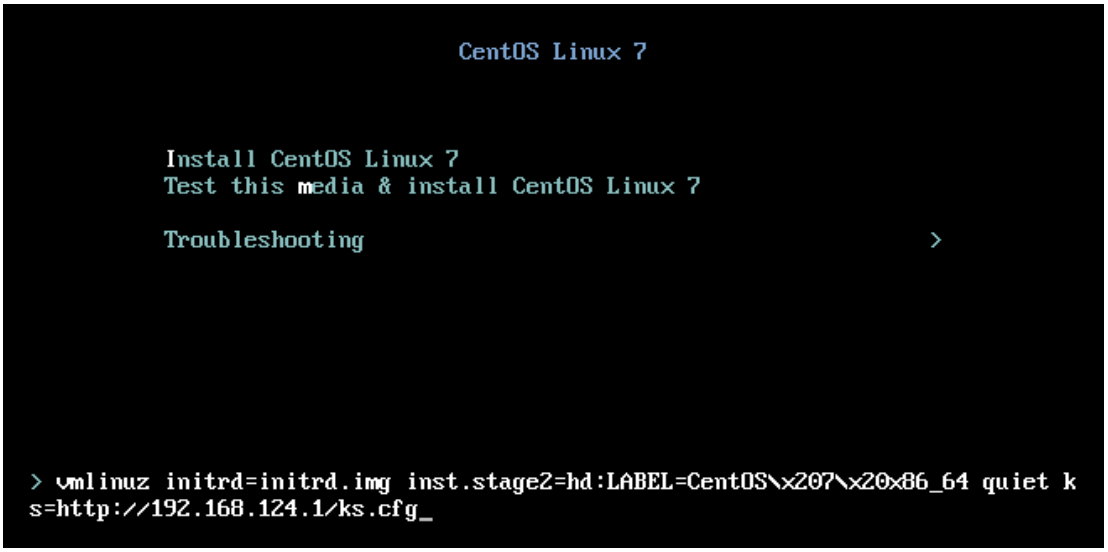


Figura 9: Pantalla de instalación. Indicar ruta de kickstart.

2. Proceso de instalación del SO. Se configuran ciertos parámetros incluidos en el *kickstart* y se instala Ansible en el sistema utilizando el repositorio configurado en el servidor auxiliar. La instalación es completamente desatendida. El aspecto del instalador es el mostrado en la Figura 10:

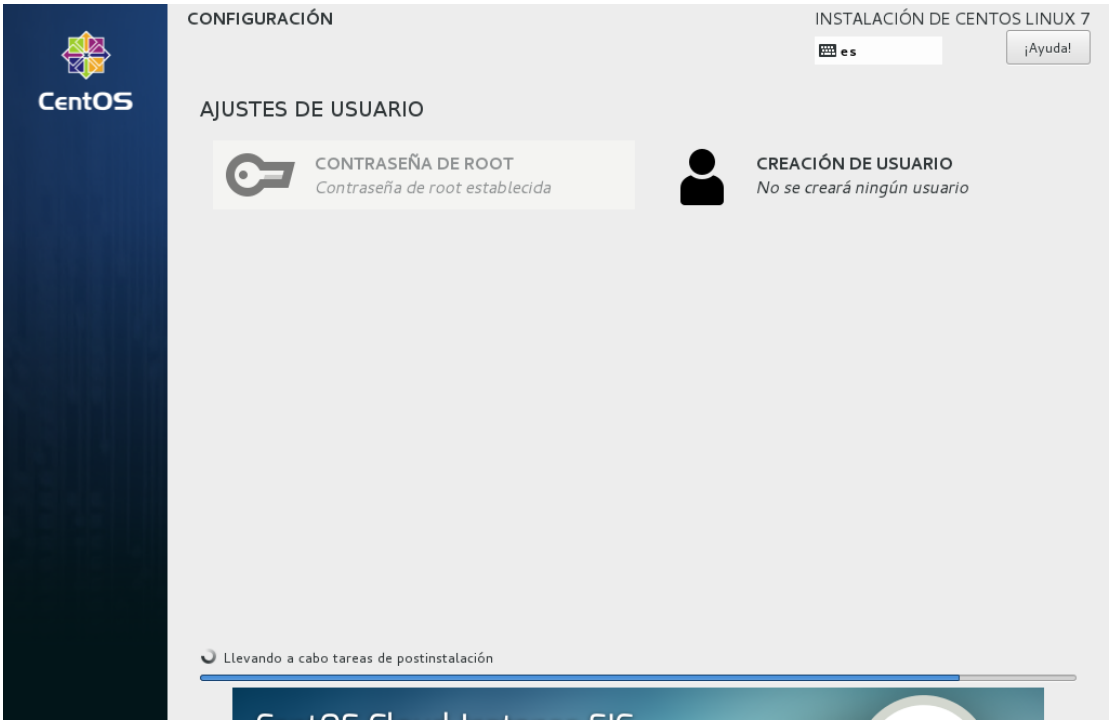


Figura 10: Aspecto de la instalación desatendida.

3. Fin de la instalación del SO. En la última parte del *kickstart* antes de la primera ejecución del sistema se ejecutan Ansible para acabar la configuración. También se programa una ejecución en el primer reinicio para la configuración del servidor de Base de Datos, ya que para hacer modificaciones en él ha de estar en ejecución y no es posible arrancarlo durante la instalación.

4. Primer reinicio. Se ejecuta un servicio que descarga el último *playbook* para configurar la base de datos y se eliminan los ficheros que han sido utilizados. La configuración del sistema ha finalizado.

4.2. Evaluación. Análisis de Vulnerabilidades.

Una vez comprobado que la instalación acaba correctamente, no se encuentra ningún mensaje de error en los logs y probado que se ha aplicado la configuración deseada, es el momento de ejecutar un análisis de vulnerabilidades del sistema para evaluar su seguridad y mitigar los posibles problemas que aparezcan.

Para ello se va a utilizar OpenVAS versión 9, herramienta de análisis de vulnerabilidades de código abierto. Para obtener un resultado más enfocado al usuario, se recomienda utilizar un servidor con el entorno gráfico instalado (*Desktop*).

El administrador de sistemas deberá instalar OpenVAS en un servidor en la misma red que el servidor seguro. En este caso, la red donde se encuentra el servidor seguro es la 192.168.124.0/24. Se va a utilizar otro servidor Centos 7 con el entorno gráfico instalado y la dirección IP 192.168.124.10.

El esquema es el que se muestra en la Figura 11:

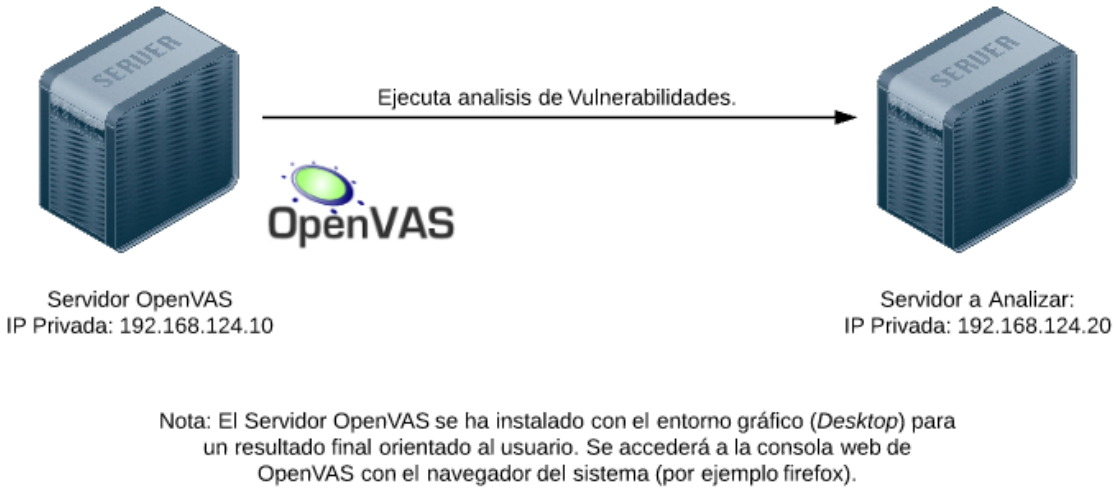


Figura 11: Servidor OpenVAS

Las instrucciones para la instalación de OpenVAS son las siguientes:

- Instalar repositorio EPEL.
  - `yum -y install epel-release`
- Instalar redis (Base de Datos tipo clave-valor).
  - `yum -y install redis`
- Instalar las utilidades necesarias para el funcionamiento de OpenVAS.
  - `yum -y install wget bzip2 texlive net-tools alien gnutls-utils`
- Instalar repositorio de Atomicorp que proporciona un método de instalación mejorado:
  - `wget -q -O - https://www.atomicorp.com/installers/atomic | sh`
    - `Do you agree to these terms? (yes/no) [Default: yes] Intro`
    - `Enable repo by default? (yes/no) [Default: yes]: Intro`
- Ejecutar el instalador de OpenVAS:
  - `yum -y install openvas`
- Configurar socket de redis correctamente: editar el archivo `/etc/redis.conf` añadiendo las líneas:
  - `unixsocket /run/redis/redis.sock` (asegurarse de que no hay otro definido la instalación de OpenVAS añade una línea al final del archivo definiendo otra ruta).
  - `unixsocketperm 700`
- Reiniciar y habilitar redis:
  - `systemctl enable redis && systemctl restart redis`
- Ejecutar la instalación de OpenVAS:
  - `openvas-setup`

    Usar los siguientes valores en la instalación:
    - `Downloader [Default: rsync] Intro`
    - `Allow connections from any IP? [Default: yes] Intro`
    - `Enter administrator username [Default: admin] : Intro`

        - `Enter Administrator password: OpenVAS`

- Editar el archivo `/etc/openvas/openvassd.conf` añadiendo:
    - `kb_location = /run/redis/redis.sock`
- Editar de nuevo el archivo `/etc/redis.conf` eliminando la línea:
    - `unixsocket /tmp/redis.sock`
- Reiniciar servicios de OpenVAS
    - `systemctl restart gsad`
    - `systemctl restart openvas-manager`
    - `systemctl restart openvas-scanner`
- Abrir el puerto de OpenVAS en el firewall:
    - `firewall-cmd --permanent --add-port=9392/tcp`
    - `firewall-cmd --reload`

Al finalizar todas las instrucciones anteriores se deberá abrir el navegador web del sistema (por ejemplo firefox) y acceder a la URL: https://192.168.124.10:9392 insertando las credenciales *Username*: admin y *Password*: OpenVAS (ver Figura 12).

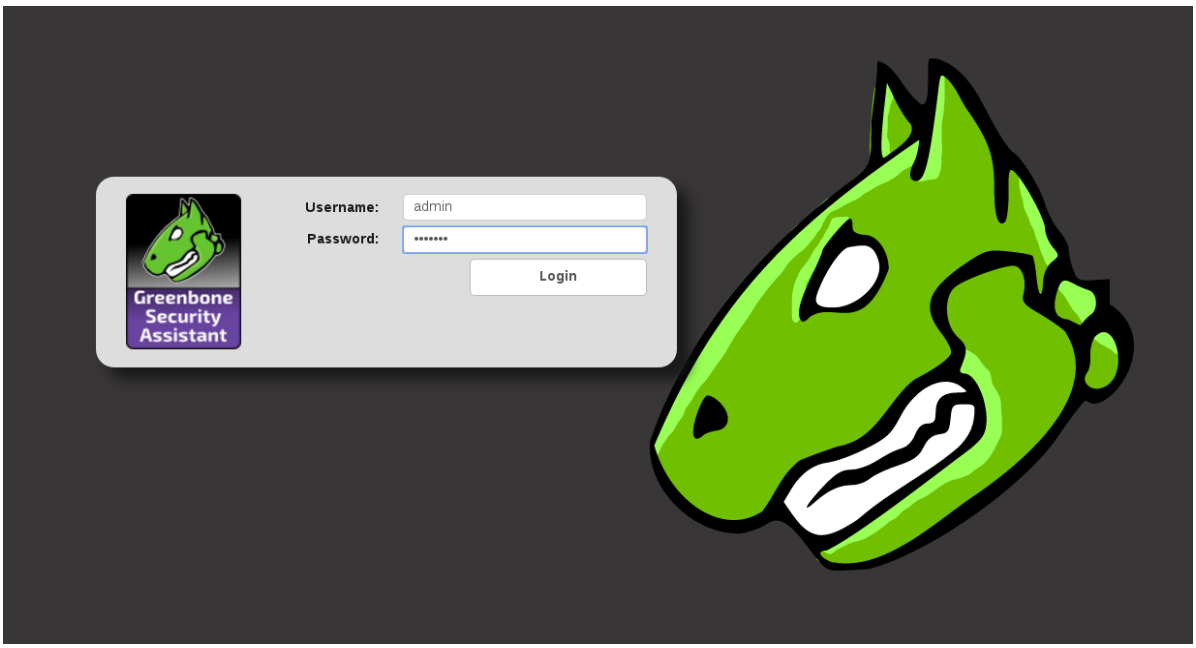


Figura 12: Pantalla de login de OpenVAS

Al hacer click en el botón *Login* se accede al panel de bienvenida (*Dashboard)* que en un primer momento estará vacío. Para ejecutar el primer análisis, se debe seleccionar *Scans* → *Tasks:*

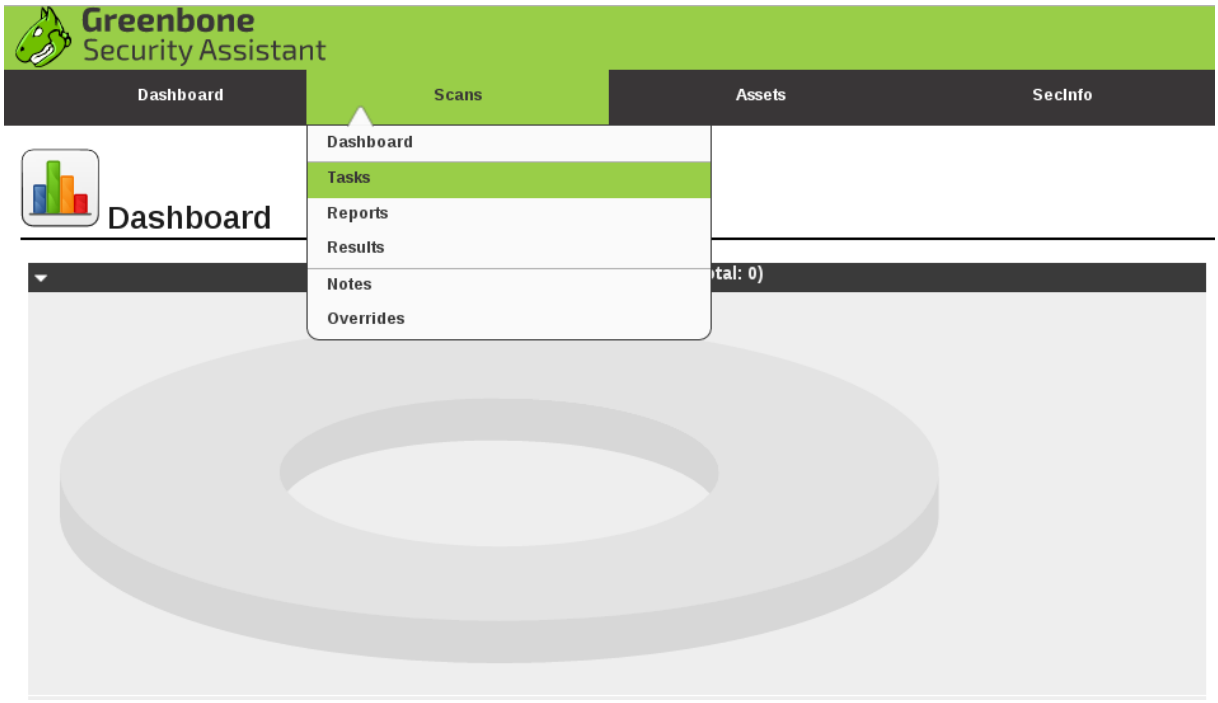


Figura 13: Pantalla de bienvenida de OpenVAS

Seguidamente aparece un mensaje que recomienda hacer click en el botón  (*Wizard)* en la parte superior izquierda de la pantalla. Tras hacer click en él aparece una pantalla como la que se muestra en la Figura 14, donde ha de introducirse la dirección IP de servidor a escanear (en este caso 192.168.124.20) y hacer click en el botón *Start Scan*.

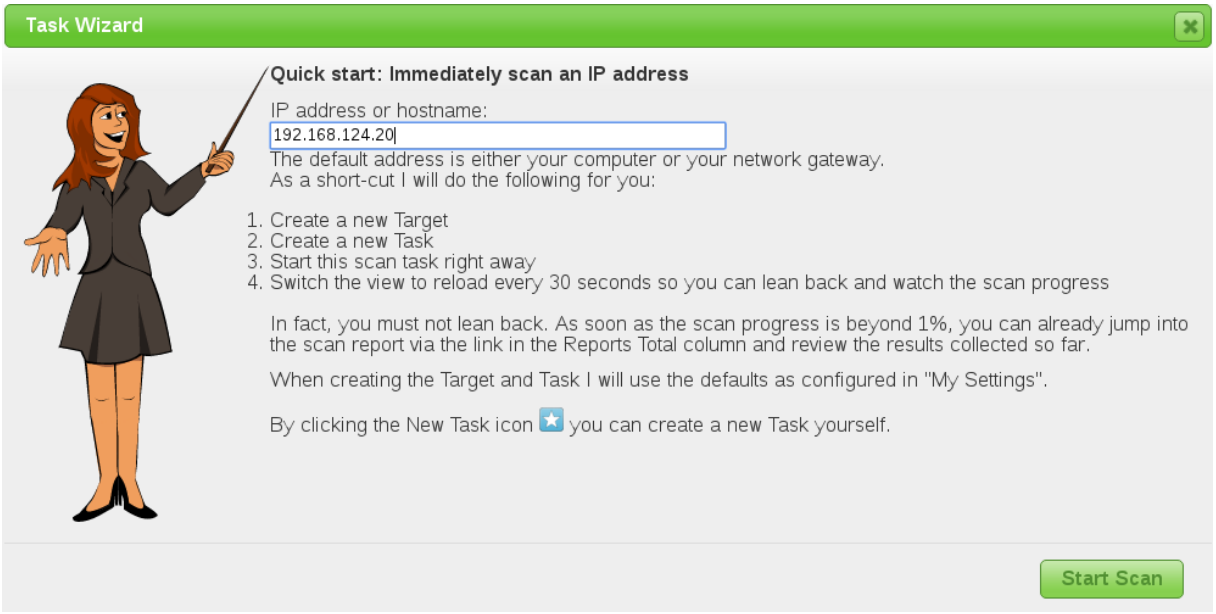


Figura 14: Asistente (Wizard) de inicio de escaneo de vulnerabilidades de OpenVAS

A partir de este momento queda esperar a que el análisis de vulnerabilidades se complete, momento en el cual se indicará la leyenda *Done* (Hecho) en la barra de progreso de la columna *Status*. Se podrá acceder a las vulnerabilidades encontradas durante el escaneo haciendo click en la misma barra de progreso que indica *Done* (Hecho). Aparecerá una pantalla similar a la que se encuentra en la Figura 15.

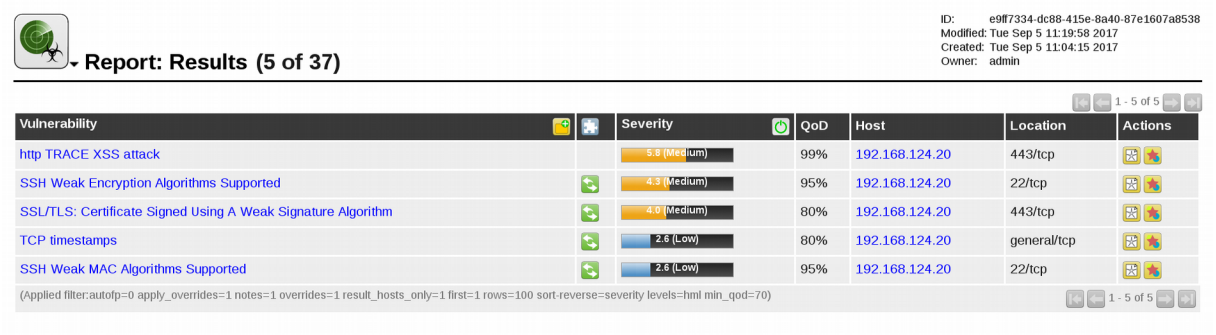
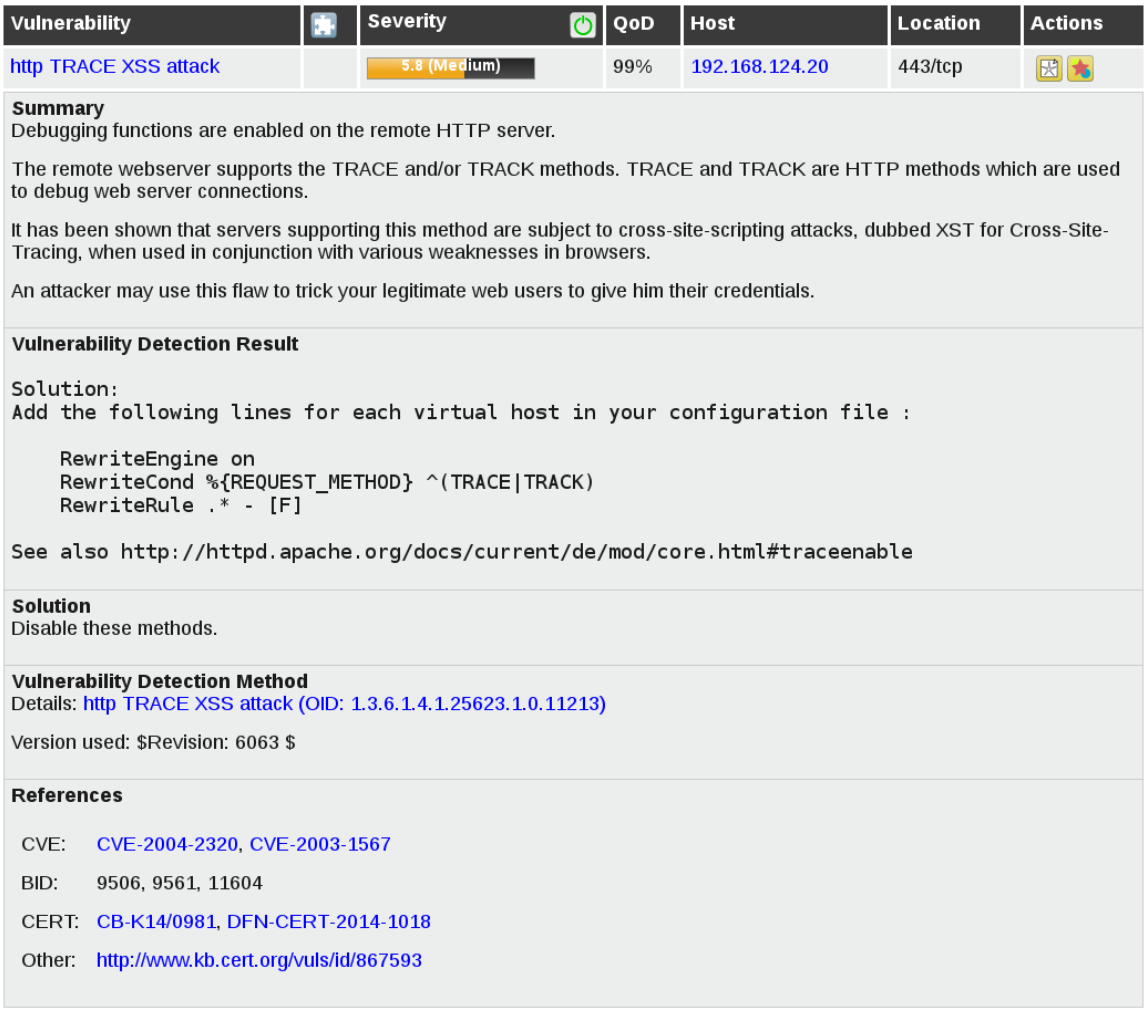


Figura 15: Vulnerabilidades Encontradas por OpenVAS

Como se puede observar, OpenVAS ha encontrado 5 posibles vulnerabilidades en el sistema. Es el momento de hacer click sobre cada una de ellas para ver en qué consiste cada una y cómo se puede actuar para mitigarlas.

La primera de ellas es "*http TRACE XSS attack*". Al hacer click sobre ella se muestra lo siguiente:



Vulnerability	Severity	QoD	Host	Location	Actions
http TRACE XSS attack	5.8 (Medium)	99%	192.168.124.20	443/tcp	

Summary
Debugging functions are enabled on the remote HTTP server.

The remote webserver supports the TRACE and/or TRACK methods. TRACE and TRACK are HTTP methods which are used to debug web server connections.

It has been shown that servers supporting this method are subject to cross-site-scripting attacks, dubbed XST for Cross-Site-Tracing, when used in conjunction with various weaknesses in browsers.

An attacker may use this flaw to trick your legitimate web users to give him their credentials.

Vulnerability Detection Result

```
Solution:
Add the following lines for each virtual host in your configuration file :

    RewriteEngine on
    RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK)
    RewriteRule .* - [F]

See also http://httpd.apache.org/docs/current/de/mod/core.html#traceenable
```

Solution
Disable these methods.

Vulnerability Detection Method
Details: http TRACE XSS attack (OID: 1.3.6.1.4.1.25623.1.0.11213)

Version used: $Revision: 6063 $

References

CVE:	CVE-2004-2320, CVE-2003-1567
BID:	9506, 9561, 11604
CERT:	CB-K14/0981, DFN-CERT-2014-1018
Other:	http://www.kb.cert.org/vuls/id/867593

Figura 16: http TRACE XSS attack

Como se observa en la Figura 16, OpenVAS provee información detallada acerca de la vulnerabilidad, solución, método de detección empleado y referencias a diferentes bases de datos de estándares de definición de vulnerabilidades donde puede encontrarse más información (CVE, BID, CERT y Otras). En este caso, hay que modificar el vhost que se definió en el Objetivo 10 (SW: Acceso Mediante Protocolos Seguros y Cifrados Robustos) para que incluya las tres reglas que se indican como solución.

La segunda de ellas es "*SSH Weak Encryption Algorithms Supported*". Al hacer click sobre ella se muestra lo siguiente:

Vulnerability	Severity	QoD	Host	Location	Actions
SSH Weak Encryption Algorithms Supported	4.3 (Medium)	95%	192.168.124.20	22/tcp	

Summary
The remote SSH server is configured to allow weak encryption algorithms.

Vulnerability Detection Result

```
The following weak client-to-server encryption algorithms are supported by the remote
service:

3des-cbc
aes128-cbc
aes192-cbc
aes256-cbc
arcfour
arcfour128
arcfour256
blowfish-cbc
cast128-cbc
rijndael-cbc@lysator.liu.se


The following weak server-to-client encryption algorithms are supported by the remote
service:

3des-cbc
aes128-cbc
aes192-cbc
aes256-cbc
arcfour
arcfour128
arcfour256
blowfish-cbc
cast128-cbc
rijndael-cbc@lysator.liu.se
```

Solution

Solution type: Mitigation

Disable the weak encryption algorithms.

Vulnerability Insight
The `arcfour` cipher is the Arcfour stream cipher with 128-bit keys. The Arcfour cipher is believed to be compatible with the RC4 cipher [SCHNEIER]. Arcfour (and RC4) has problems with weak keys, and should not be used anymore.

The `none` algorithm specifies that no encryption is to be done. Note that this method provides no confidentiality protection, and it is NOT RECOMMENDED to use it.

A vulnerability exists in SSH messages that employ CBC mode that may allow an attacker to recover plaintext from a block of ciphertext.

Vulnerability Detection Method
Check if remote ssh service supports Arcfour, none or CBC ciphers.

Details: SSH Weak Encryption Algorithms Supported (OID: 1.3.6.1.4.1.25623.1.0.105611)

Version used: $Revision: 4490 $

References

Other: https://tools.ietf.org/html/rfc4253#section-6.3

https://www.kb.cert.org/vuls/id/958563

Figura 17: SSH Weak Encryption Algorithms Supported

Esta vulnerabilidad hace referencia al uso de algoritmos de cifrado considerados débiles en el protocolo SSH. Para mitigar esta vulnerabilidad, es necesario configurar el servicio SSH para que no acepte dichos certificados.

En la Figura 18 se encuentra una sección del manual de `sshd_config`:

Figura 18: Sección Ciphers del manual de sshd_config

Como se puede observar, por defecto están habilitados algunos considerados débiles y por tanto se debe añadir en el fichero "`/etc/ssh/sshd_config`" la línea:

- `Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com,chacha20-poly1305@openssh.com`

Además, hay que añadir también dicha línea en el fichero "`/etc/ssh/ssh_config`" ya que la vulnerabilidad afecta tanto al ejercer de servidor como de cliente. Finalmente es necesario reiniciar el servicio para que aplique los cambios.

Nota: al configurar de manera restrictiva los algoritmos de cifrado puede que ya no sea posible conectar al servidor vía ssh desde algún cliente que no soporte estos algoritmos.

La tercera vulnerabilidad encontrada por OpenVAS es "*SSL/TLS: Certificate Signed Using A Weak Signature Algorithm*". Al hacer click sobre ella se muestra lo siguiente:

Vulnerability	Severity	QoD	Host	Location	Actions
SSL/TLS: Certificate Signed Using A Weak Signature Algorithm	4.0 (Medium)	80%	192.168.124.20	443/tcp	

Summary
The remote service is using a SSL/TLS certificate chain that has been signed using a cryptographically weak hashing algorithm.

Vulnerability Detection Result

```
The following certificates are part of the certificate chain but using insecure signature
algorithms:

Subject:              CN=HTTPD Server,OU=IT Department,O=My Company,L=Valencia,ST=ES,C=ES
Signature Algorithm:  sha1WithRSAEncryption
```

Solution
Solution type: Mitigation

Servers that use SSL/TLS certificates signed using an SHA-1 signature will need to obtain new SHA-2 signed SSL/TLS certificates to avoid these web browser SSL/TLS certificate warnings.

Vulnerability Insight
Secure Hash Algorithm 1 (SHA-1) is considered cryptographically weak and not secure enough for ongoing use. Beginning as late as January 2017 and as early as June 2016, browser developers such as Microsoft and Google will begin warning users when users visit web sites that use SHA-1 signed Secure Socket Layer (SSL) certificates.

Vulnerability Detection Method
Check which algorithm was used to sign the remote SSL/TLS Certificate.

Details: SSL/TLS: Certificate Signed Using A Weak Signature Algorithm (OID: 1.3.6.1.4.1.25623.1.0.105880)

Version used: $Revision: 4781 $

References

Other: https://blog.mozilla.org/security/2014/09/23/phasing-out-certificates-with-sha-1-based-signature-algorithms/

Figura 19: SSL/TLS: Certificate Signed Using A Weak Signature Algorithm

Aquí se indica que el algoritmo con el que se ha firmado el certificado que utiliza el servidor web es considerado débil. Al crear el certificado para el servidor web, uno de los pasos ejecutaba dos instrucciones similares a:

- `openssl req -new -key ca.key -out ca.csr` [parámetros omitidos]
- `openssl x509 -req -days 365000` [parámetros omitidos]

Para solucionarlo bastará con añadir el parámetro "`-sha256`" al final de estos comandos.

Nota: los certificados utilizados en este trabajo son autogenerados y en cualquier caso serán considerados inseguros por los navegadores web. Para un entorno productivo se deberá contar con certificados firmados por una entidad certificadora reconocida por los navegadores web más comunes (Firefox, Google Chrome, Safari ó Microsoft Edge).

La cuarta vulnerabilidad es "*TCP timestamps*". Al hacer click sobre ella se muestra lo siguiente:

Vulnerability	Severity	QoD	Host	Location	Actions
TCP timestamps	2.6 (Low)	80%	192.168.124.20	general/tcp	

Summary
The remote host implements TCP timestamps and therefore allows to compute the uptime.

Vulnerability Detection Result

```
It was detected that the host implements RFC1323.

The following timestamps were retrieved with a delay of 1 seconds in-between:
Packet 1: 133620626
Packet 2: 133621663
```

Impact
A side effect of this feature is that the uptime of the remote host can sometimes be computed.

Solution
Solution type: Mitigation

To disable TCP timestamps on linux add the line 'net.ipv4.tcp_timestamps = 0' to /etc/sysctl.conf. Execute 'sysctl -p' to apply the settings at runtime.

To disable TCP timestamps on Windows execute 'netsh int tcp set global timestamps=disabled'

Starting with Windows Server 2008 and Vista, the timestamp can not be completely disabled.

The default behavior of the TCP/IP stack on this Systems is to not use the Timestamp options when initiating TCP connections, but use them if the TCP peer that is initiating communication includes them in their synchronize (SYN) segment.

See also: http://www.microsoft.com/en-us/download/details.aspx?id=9152

Affected Software/OS
TCP/IPv4 implementations that implement RFC1323.

Vulnerability Insight
The remote host implements TCP timestamps, as defined by RFC1323.

Vulnerability Detection Method
Special IP packets are forged and sent with a little delay in between to the target IP. The responses are searched for a timestamps. If found, the timestamps are reported.

Details: TCP timestamps (OID: 1.3.6.1.4.1.25623.1.0.80091)

Version used: $Revision: 5740 $

References

Other: http://www.ietf.org/rfc/rfc1323.txt

Figura 20: TCP timestamps

Esta se refiere a la posibilidad de que un atacante sea capaz de saber el tiempo que lleva un servidor encendido. Para solucionarlo hay que añadir la línea:

- `net.ipv4.tcp_timestamps = 0`

Al archivo "`/etc/sysctl.conf`".

Por último, la quinta vulnerabilidad encontrada es "*SSH Weak MAC Algorithms Supported*". Al hacer click sobre ella se muestra:

Figura 21: SSH Weak MAC Algorithms Supported

Y está relacionada con el uso de algoritmos MAC (del inglés *Message Authentication Codes*). Involucra los mismos ficheros que la vulnerabilidad "*SSH Weak Encryption Algorithms Supported*" dado que se trata otra vez del protocolo SSH. La solución se encuentra mirando de nuevo el manual de sshd_config en la sección MACs. Para solucionarlo hay que añadir la siguiente línea en los ficheros `"/etc/ssh/ssh_config"` y `"/etc/ssh/sshd_config"`:

- `MACs hmac-sha1-etm@openssh.com,umac-64-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha2-512-etm@openssh.com,hmac-ripemd160-etm@openssh.com,hmac-sha1,umac-64@openssh.com,umac-128@openssh.com,hmac-sha2-256,hmac-sha2-512,hmac-ripemd160`

Tras analizar todas las vulnerabilidades encontradas por OpenVAS y enumerar sus soluciones se ha repetido el análisis para comprobar que realmente se solucionan dichas vulnerabilidades.

Además se han añadido estas soluciones o bien a los *playbooks* de Ansible utilizados durante la instalación segura del servidor o bien a un nuevo *playbook* llamado "`openvas_mitig.yml`" para mejorar la calidad de la instalación automatizada.

Los *playbooks* de Ansible utilizados están disponibles en el Anexo I.

5. Conclusiones y trabajo futuro.

Dada la importancia de que toda organización disponga de unas mínimas garantías en cuanto a la seguridad de sus sistemas y datos, se propuso al principio de este trabajo el desarrollo de un método que proporcionase dicha seguridad utilizando herramientas de código abierto, es decir, sin que el software utilizado suponga ningún coste.

Durante la fase de análisis del estado del arte en el Capítulo 2 se ha podido comprobar cómo la gran mayoría de aplicaciones que son utilizadas en los servidores en Internet son de código abierto. Durante esta fase de análisis del estado del arte se ha podido encontrar gran cantidad de información de diversas fuentes que puede ser utilizada para hacer una configuración segura del servidor. La contribución de este trabajo al estado del arte se basa en la recopilación y criba de toda esta información y la generación de una metodología de configuración segura automatizada y fácilmente personalizable con unos conocimientos mínimos en materia de seguridad informática garantizando un alto nivel de seguridad.

En el Capítulo 3, basándose en la documentación oficial de los proyectos o en la documentación disponible gracias a la comunidad de desarrolladores, se establecieron unos objetivos de seguridad que debían cumplirse en tres áreas principalmente: Sistema Operativo, Servidor Web y Base de Datos. Se investigó la manera de cumplir los objetivos y se definió la manera de automatizarlo por medio de dos herramientas: el *kickstart* de la instalación del SO o mediante la herramienta de administración de sistemas Ansible. De esta manera se consigue la automatización del método evitando posibles errores humanos durante el proceso de instalación, haciendo el proceso repetible y reduciendo también el tiempo y recursos necesarios para realizar la configuración segura.

Finalmente, en el Capítulo 4 se explicó de manera detallada la arquitectura utilizada y el desarrollo de la metodología, reuniendo los elementos de automatización que se habían ido construyendo en el capítulo anterior, consiguiendo cumplir todos los objetivos establecidos de una manera automatizada. Además se realizó un análisis de vulnerabilidades mediante la herramienta OpenVAS que proporciona instrucciones para mitigar dichas vulnerabilidades. Dichas instrucciones se incorporaron en los *playbooks* de Ansible para mejorar la calidad final de la instalación segura.

La conclusión en cuanto al desarrollo del trabajo es que se puede conseguir un alto nivel de seguridad utilizando para ello solamente herramientas de código abierto. Sin embargo debe mencionarse que a medida que el tamaño de una organización aumenta, aumentan también las necesidades de soporte, personalización de los productos y urgencia de aplicación de parches de seguridad, por lo que todas las herramientas utilizadas en este trabajo disponen de versiones comerciales de manera que las grandes empresas pueden disponer de un servicio acorde a sus necesidades.

En cuanto al trabajo futuro y posibles mejoras a este trabajo se proponen los siguientes temas:

- Sería interesante un trabajo centrado en la comparación de seguridad obtenida utilizando herramientas de código abierto frente a herramientas de pago.
- En cuanto al análisis de vulnerabilidades realizado con OpenVAS, podría profundizarse más en el uso y personalización de la herramienta.
- Otra línea de trabajo propuesta es la de utilizar para el análisis de vulnerabilidades OpenSCAP en lugar de OpenVAS y hacer una comparativa de los resultados obtenidos. OpenSCAP es una herramienta muy potente que cuenta con un alto nivel de granularidad en la personalización, pudiendo conseguir una puesta en seguridad muy exhaustiva.

Como conclusión final acerca trabajo es necesario recordar al lector que la seguridad es un elemento transversal a toda organización, que abarca desde el usuario que accede al correo electrónico desde un terminal hasta el servidor central, y por lo tanto es labor de todo profesional de la ciberseguridad y de la alta dirección concienciar a todo miembro de la organización acerca de las buenas prácticas en lo referente a ella. Además, todo administrador de sistemas que afronte la puesta en seguridad de cualquier sistema debe comprender que la seguridad no es cosa de un día, es decir: no basta con hacer una buena puesta en seguridad en la instalación de un sistema, ya que cada día aparecen nuevas vulnerabilidades que pueden afectar en mayor o menor medida a su organización. Por ello durante el desarrollo del trabajo se ha hecho hincapié en que todo elemento software que compone un sistema debe ser actualizado regularmente para evitar ser víctima de un ciberataque.

6. Referencias y Enlaces.

Bibliografía

[1] ENISA. (2008). Social Engineering: Exploiting the Weakest Links. [Online]. Disponible: "https://www.enisa.europa.eu/publications/iitl/at_download/fullReport"
[2] SANS Institute. (2015). Methods for Understanding and Reducing Social Engineering Attacks. [Online]. Disponible: https://www.sans.org/reading-room/whitepapers/critical/methods-understanding-reducing-social-engineering-attacks-36972
[3] McAfee. (2011). Prospective Analysis on Trends in Cybercrime from 2011 to 2020. [Online]. Disponible: https://www.mcafee.com/es/resources/white-papers/wp-trends-in-cybercrime-2011-2020.pdf
[4] ICF. (2016). Addressing the Wholeof-EnterpriseThreat. [Online]. Disponible: https://www.icf.com/-/media/files/icf/white-papers/2015/whole_of_enterprise_threat.pdf
[5] Hewlett Packard Enterprise. (2016). Reduce Security Risks from Open Source Software. [Online]. Disponible: https://www.hpe.com/h20195/v2/GetPDF.aspx/4AA0-8061ENW.pdf
[6] Optimus Information, Inc.. (2015). Open-Source vs. Proprietary Software Pros and Cons. [Online]. Disponible: http://www.optimusinfo.com/downloads/white-paper/open-source-vs-proprietary-software-pros-and-cons.pdf
[7] GitHub. (2017). GitHub Official Web Page. [Online]. Disponible: https://github.com/
[8] The Linux Foundation. (2016). 25 Years of Linux Kernel Development. [Online]. Disponible: http://go.linuxfoundation.org/linux-kernel-development-report-2016
[9] Red Hat. (2017). Red Hat Enterprise Linux Life Cycle. [Online]. Disponible: https://access.redhat.com/support/policy/updates/errata/
[10] Debian. (2016). Debian Releases. [Online]. Disponible: https://wiki.debian.org/DebianReleases
[11] The CentOS Project. (2017). About CentOS. [Online]. Disponible: https://www.centos.org/about/
[12] The CentOS Project. (2017). CentOS FAQ: 2. How long after Red Hat publishes a fix does it take for CentOS to publish a fix?. [Online]. Disponible: https://wiki.centos.org/FAQ/General#head-cea9337e6513cc1567c4d05afbd693f1f7038ccb
[13] The Apache Software Foundation. (2017). Apache HTTP Server Project. [Online]. Disponible: https://httpd.apache.org/
[14] The SANS Institute. (2003). Vulnerability naming schemes and description languages:CVE, Bugtraq, AVDL and VulnXML. [Online]. Disponible: https://www.sans.org/reading-room/whitepapers/threats/vulnerability-naming-schemes-description-languages-cve-bugtraq-avdl-vulnxml-1058
[15] Nginx.org. (2017). Nginx Official Web Site. [Online]. Disponible: https://nginx.org
[16] Barkan Saeed. (2016). Apache vs Nginx. [Online]. Disponible: http://vizteck.com/blog/apache-vs-nginx/
[17] Walker Rowe @ Anturis. (2014). Nginx vs Apache. [Online]. Disponible: https://anturis.com/blog/nginx-vs-apache/

[18] Jefatura del Estado. (1999). Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal. [Online]. Disponible: https://www.boe.es/buscar/pdf/1999/BOE-A-1999-23750-consolidado.pdf
[19] MySQL. (2017). Sitio Web Oficial de MySQL. [Online]. Disponible: www.mysql.com
[20] Rikki Endsley. (2013). 5 Reasons It's Time to Ditch MySQL. [Online]. Disponible: https://blog.smartbear.com/open-source/5-reasons-its-time-to-ditch-mysql/
[21] Seravo Oy. (2015). 10 reasons to migrate to MariaDB (if still using MySQL). [Online]. Disponible: https://seravo.fi/2015/10-reasons-to-migrate-to-mariadb-if-still-using-mysql
[22] MariaDB Foundation. (2012). Disappearing test cases or did another part of MySQL just become closed source?. [Online]. Disponible: https://mariadb.org/disappearing-test-cases/
[23] PostgreSQL. (2017). Sitio Web Oficial de PostgreSQL. [Online]. Disponible: www.postgresql.org
[24] A. Anderson, R. Levy, T. Peterson, T. Robinson, A. Rosales, B. Semple, M. Sheard, I. Werpoler. (2015). Turbo LAMP White Paper: The LAMP Stack for Today's Demanding Application Workload Requirements. [Online]. Disponible: http://www.zend.com/assets/turbolamp-whitepaper.pdf
[25] Red Hat. (2017). Chapter 24. Kickstart Installations. [Online]. Disponible: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Installation_Guide/chap-kickstart-installations.html
[26] Red Hat. (2017). 24.2. How Do You Perform A Kickstart Installation?. [Online]. Disponible: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Installation_Guide/sect-kickstart-howto.html
[27] Red Hat. (2011). The path to operational efficiency: Implementing a Standard Operating Environment. [Online]. Disponible: http://www.europe.redhat.com/pdf/RH_RHEL_GPS.pdf
[28] ServiceNow. (2013). Orchestration For Modern IT: Helping IT Accelerate Business . [Online]. Disponible: https://www.servicenow.com/content/dam/servicenow/other-documents/ebook/downloads/Runbook_Automation_Helping_IT_Accelerate_Its_Business.pdf
[29] Wikipedia. (2017). Comparison of Open-Source Configuration Management Software. [Online]. Disponible: https://en.wikipedia.org/wiki/Comparison_of_open-source_configuration_management_software
[30] GitHub. (2017). GitHub: Ansible. [Online]. Disponible: https://github.com/ansible/ansible
[31] GitHub. (2017). GitHub: Saltstack / Salt. [Online]. Disponible: https://github.com/saltstack/salt
[32] GitHub. (2017). GitHub: Chef. [Online]. Disponible: https://github.com/chef/chef
[33] Red Hat. (2017). Continous Integration and Continous Deployment With Ansible. [Online]. Disponible: https://www.ansible.com/continuous-deployment-whitepaper
[34] Red Hat. (2017). YAML Syntax. [Online]. Disponible: http://docs.ansible.com/ansible/YAMLSyntax.html
[35] Red Hat. (2017). Ansible Galaxy Official Web Site. [Online]. Disponible: https://galaxy.ansible.com/

[36] NIST. (2015). What You Need to Know About the NIST Guidelines for Secure Shell. [Online]. Disponible: http://info.ssh.com/nist_7966_secure_shell_whitepaper
[37] SaltStack. (2017). Saltstack Official Web Site. [Online]. Disponible: https://saltstack.com/
[38] UpGuard. (2017). SaltStack vs Ansible Revisited. [Online]. Disponible: https://www.upguard.com/articles/saltstack-vs-ansible-revisited
[39] Alexandre Joseph. (2016). Ansible vs Salt Meta Review. [Online]. Disponible: http://www.alexandrejoseph.com/blog/2016-03-23-ansible-vs-salt-meta-review.html
[40] Chef Software, Inc. (2017). Chef Official Web Site. [Online]. Disponible: https://www.chef.io/chef/
[41] Json.org. (2017). Introducing JSON. [Online]. Disponible: http://www.json.org/
[42] UpGuard. (2017). [Infographic] Ansible vs Chef. [Online]. Disponible: https://www.upguard.com/articles/ansible-vs-chef
[43] Ali Raza. (2016). Puppet vs. Chef vs. Ansible vs. SaltStack. [Online]. Disponible: http://www.intigua.com/blog/puppet-vs.-chef-vs.-ansible-vs.-saltstack
[44] Red Hat. (2017). Use Case: Security & Compliance. [Online]. Disponible: https://www.ansible.com/security-and-compliance
[45] OpenVAS.org. (2017). OpenVAS Official Web Site. [Online]. Disponible: http://www.openvas.org/
[46] Red Hat. (2016). OpenSCAP Official Web Page. [Online]. Disponible: https://www.open-scap.org/
[47] Red Hat. (2017). Red Hat Enterprise Linux 7 Security Guide. [Online]. Disponible: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/pdf/Security_Guide/Red_Hat_Enterprise_Linux-7-Security_Guide-en-US.pdf
[48] Dharmesh Mehta. (2007). Effective Software Security Management. [Online]. Disponible: https://www.owasp.org/images/2/28/Effective_Software_Security_Management.pdf
[49] Kaspersky Lab. (2016). Lack of Security Talent: An Unexpected Threat To Corporate Cybersafety. [Online]. Disponible: https://kas.pr/58E2
[50] Mike Ambrosone @ Vircom. (2017). Cybersecurity Trends: Lack of Cybersecurity Experts. [Online]. Disponible: https://www.vircom.com/blog/lack-of-cybersecurity-experts/
[51] CentOS Project. (2017). CemtOS - HowTos - OS Protection. [Online]. Disponible: https://wiki.centos.org/HowTos/OS_Protection
[52] Bitfeed.co. (2017). UEFI: what it is and what you need to know. [Online]. Disponible: http://www.bitfeed.co/page/uefi-what-it-is-and-what-you-need-to-know
[53] The Apache Software Foundation. (2017). Security Tips. [Online]. Disponible: https://httpd.apache.org/docs/2.4/misc/security_tips.html
[54] The Apache Software Foundation. (2017). Apache Performance Tuning. [Online]. Disponible: https://httpd.apache.org/docs/2.4/misc/perf-tuning.html
[55] CentOS. (2014). Setting up an SSL secured Webserver with CentOS. [Online]. Disponible: https://wiki.centos.org/HowTos/Https
[56] MariaDB. (2017). Securing MariaDB. [Online]. Disponible: https://mariadb.com/kb/en/the-mariadb-library/securing-mariadb/

[57] Dev-Sec. (2017). dev-sec/ansible-mysql-hardening/tasks/mysql_secure_installation.yml. [Online]. Disponible: https://github.com/dev-sec/ansible-mysql-hardening/blob/master/tasks/mysql_secure_installation.yml
[58] Vivek Gite. (2017). How to setup MariaDB SSL and secure connections from clients. [Online]. Disponible: https://www.cyberciti.biz/faq/how-to-setup-mariadb-ssl-and-secure-connections-from-clients/
[59] The CentOS Project. (2017). Download CentOS. [Online]. Disponible: https://www.centos.org/download/
[60] Red Hat, Inc.. (2017). Playbook Roles and Include Statements. [Online]. Disponible: http://docs.ansible.com/ansible/latest/playbooks_roles.html

ANEXO I: Playbooks

El contenido de los *playbooks* de Ansible está estructurado como sigue:

```
ansible/
```

```
├── main.yml
```

```
---
- name: Playbook Principal
  hosts: all
  vars:
    httpd_ssl_folder: /etc/pki/tls
    mariadb_ssl_folder: /var/lib/mysql/ssl
    mariadb_root_password: M@ri@DBrootPWD
  roles:
    - secure-server
```

```
└── roles
    └── secure-server
        └── tasks
            ├── main.yml
```

```
---
- debug:
    msg: "Aplicando Configuración Segura en el Servidor"

- include: dnssec.yml
- include: create_user.yml
- include: audit.yml
- include: httpd_certs.yml
- include: httpd_install.yml
- include: httpd_vhost.yml
- include: mariadb_install.yml
- include: mariadb_certs.yml
- include: openvas_mitig.yml
```

├── dnssec.yml

```
---
- debug:
    msg: "Instalando DNSSEC"

- name: Instalar ultima version de unbound
  yum:
    name: unbound
    state: latest
    disablerepo: base,extras,updates

- name: Habilitar y arrancar unbound (DNSSEC)
  systemd:
    name: unbound
    enabled: yes
```

├── create_user.yml

```
- debug:
    msg: "Creando usuario Admin1"

- user:
    name: admin1
    group: wheel
    password:
$6$ZTmiXaN27kI1.E9S$4sf98ep6PnFY9vTwowXC7Vep4zQLAT8.rMR/4LMiSPvZaH0p1L1f
FuMeK3PjIf2tXKrLwP4s8IUnqpn/aApls1

- debug:
    msg: "Politica de expiracion de password de admin1. Aviso a los 20 dias y caducidad a
los 30."

- shell: chage -W 20 -M 30 admin1

- debug:
    msg: "Eliminar tiempo de gracia del comando sudo."

- lineinfile:
    path: /etc/sudoers
    state: present
    backup: yes
    line: 'Defaults    timestamp_timeout=0'
```

```
        ├── audit.yml
```

```
---
- debug:
    msg: "Aplicando configuracion de Auditoría del Sistema"
- copy:
    src: /usr/share/doc/audit-2.6.5/rules/10-base-config.rules
    dest: /etc/audit/rules.d/10-base-config.rules
- copy:
    src: /usr/share/doc/audit-2.6.5/rules/99-finalize.rules
    dest: /etc/audit/rules.d/99-finalize.rules
- copy:
    src: /usr/share/doc/audit-2.6.5/rules/30-nispom.rules
    dest: /etc/audit/rules.d/30-nispom.rules
```

```
        ├── httpd_certs.yml
```

```
---
- debug:
    msg: "Generando Certificados de Apache HTTPD Web Server"

- stat:
    path: "{{ httpd_ssl_folder }}/private/ca.key"
  register: cakey

- name: Crear clave privada de la CA
  shell: openssl genrsa 2048 > {{ httpd_ssl_folder }}/private/ca.key
  when: cakey.stat.exists == False

- stat:
    path: "{{ httpd_ssl_folder }}/private/ca.csr"
  register: cacsr

- name: Crear CSR (peticion de certificado firmado) con la clave de la CA.
  shell: openssl req -new -key {{ httpd_ssl_folder }}/private/ca.key -out
{{ httpd_ssl_folder }}/private/ca.csr -subj '/C=ES/ST=ES/L=Valencia/O=My Company/OU=IT
Department/CN=HTTPD Server' -sha256
  when: cacsr.stat.exists == False

- stat:
    path: "{{ httpd_ssl_folder }}/certs/ca.crt"
  register: cacrt

- name: Crear certificado del servidor firmado con la clave de la CA
  shell: openssl x509 -req -days 365000 -in {{ httpd_ssl_folder }}/private/ca.csr -signkey
{{ httpd_ssl_folder }}/private/ca.key -out {{ httpd_ssl_folder }}/certs/ca.crt -sha256
  when: cacrt.stat.exists == False
```

```
        ├── httpd_install.yml
```

```
---
- debug:
    msg: "Instalando y Configurando Apache HTTPD Web Server"

- name: Instalar ultima version del Servidor Web Apache HTTP Web Server (httpd)
  yum:
    name: httpd
    state: latest
    disablerepo: base,extras,updates

- name: Instalar ultima version del modulo ssl de Apache HTTP Web Server (httpd)
  yum:
    name: mod_ssl
    state: latest
    disablerepo: base,extras,updates

- blockinfile:
    path: /etc/httpd/conf/httpd.conf
    backup: yes
    block: |
      RequestReadTimeout header=10-20,MinRate=500 body=10,MinRate=500
      TimeOut 30
      KeepAliveTimeout 3
      LimitRequestBody 512000

- replace:
    path: /etc/httpd/conf/httpd.conf
    regexp: '(FollowSymLinks|Indexes)'
    replace: ''
    backup: yes

- replace:
    path: /etc/httpd/conf/httpd.conf
    regexp: '^Listen 80'
    replace: ''
    backup: yes

- lineinfile:
    path: /etc/httpd/conf/httpd.conf
    state: present
    line: 'ServerTokens Prod'
    backup: yes
```

```
        ├── httpd_vhost.yml
```

```
---
- debug:
    msg: "Creando Vhost del Servidor Web"

- blockinfile:
    path: /etc/httpd/conf.d/01-vhost.conf
    create: yes
    backup: yes
    block: |
        # modern configuration, tweak to your needs
        SSLProtocol             all -SSLv3 -TLSv1 -TLSv1.1
        SSLCipherSuite          ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-
AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-
CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-
GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256
        SSLHonorCipherOrder     on
        SSLCompression          off
        #OCSP Stapling, only in httpd 2.3.3 and later
        SSLUseStapling          on
        SSLStaplingResponderTimeout 5
        SSLStaplingReturnResponderErrors off
        SSLStaplingCache        shmcb:/var/run/ocsp(128000)

          <VirtualHost *:443>
              SSLEngine on
              SSLCertificateFile /etc/pki/tls/certs/ca.crt
              SSLCertificateKeyFile /etc/pki/tls/private/ca.key
              # http TRACE XSS attack mitigation (CVE-2004-2320, CVE-2003-1567)
              RewriteEngine on
              RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK)
              RewriteRule .* - [F]
                    # HSTS (mod_headers is required) (15768000 seconds = 6 months)
                    Header always set Strict-Transport-Security "max-age=15768000"
                    DocumentRoot /var/www/html
                    ServerName yoursite.com
           </VirtualHost>

  register: createvhost
```

```
# NOTA:
# Se habilita el servidor web y se abre el servicio en el firewall solamente cuando está configurado y es seguro.
- name: Habilitar el servicio de Apache (httpd)
  systemd:
    name: httpd
    enabled: yes
  when: createvhost.changed
```

```
├── mariadb_install.yml
```

```
---
- debug: msg="Instalando y Configurando MariaDB"

- name: Instalando MariaDB
  yum: name=mariadb-server state=present disablerepo=base,extras,updates

- name: Instalando python-mysqldb para usar Ansible
  yum: name=MySQL-python state=present disablerepo=base,extras,updates

- replace:
    path: /usr/lib/systemd/system/mariadb.service
    regexp: '--basedir=/usr'
    replace: '--basedir=/usr --ssl --ssl_cipher=TLSv1.2'
    backup: yes
```

```
        ├── mariadb_certs.yml
```

```
---
- debug:
    msg: "Creando certificados de MariaDB"

- name: Creando directorio que contiene los certificados de MariaDB
  file:
    path: "{{ mariadb_ssl_folder }}"
    state: directory
    mode: 0755
    owner: mysql
    group: mysql

- name: Crear clave privada de la CA
  shell: openssl genrsa 2048 > {{ mariadb_ssl_folder }}/ca-key.pem

- name: Crear certificado con la clave de la CA
  shell: openssl req -new -x509 -nodes -days 365000 -key {{ mariadb_ssl_folder }}/ca-
key.pem -out {{ mariadb_ssl_folder }}/ca-cert.pem -subj '/C=ES/ST=ES/L=Valencia/O=My
Company/OU=IT Department/CN=MariaDB CA Admin'

- name: Crear certificado del servidor
  shell: openssl req -newkey rsa:2048 -days 365000 -nodes -keyout
{{ mariadb_ssl_folder }}/server-key.pem -out {{ mariadb_ssl_folder }}/server-req.pem -subj
'/C=ES/ST=ES/L=Valencia/O=My Company/OU=IT Department/CN=MariaDB Server'

- name: Procesar clave RSA
  shell: openssl rsa -in {{ mariadb_ssl_folder }}/server-key.pem -out
{{ mariadb_ssl_folder }}/server-key.pem

- name: Firmar certificado del servidor
  shell: openssl x509 -req -in {{ mariadb_ssl_folder }}/server-req.pem -days 365000 -CA
{{ mariadb_ssl_folder }}/ca-cert.pem -CAkey {{ mariadb_ssl_folder }}/ca-key.pem -set_serial
01 -out {{ mariadb_ssl_folder }}/server-cert.pem

- name: Crear certificado cliente
  shell: openssl req -newkey rsa:2048 -days 365000 -nodes -keyout
{{ mariadb_ssl_folder }}/client-key.pem -out {{ mariadb_ssl_folder }}/client-req.pem -subj
'/C=ES/ST=ES/L=Valencia/O=My Company/OU=IT Department/CN=MariaDB User'
```

```
- name: Procesar clave RSA
  shell: openssl rsa -in {{ mariadb_ssl_folder }}/client-key.pem -out
{{ mariadb_ssl_folder }}/client-key.pem

- name: Firmar certificado del servidor
  shell: openssl x509 -req -in {{ mariadb_ssl_folder }}/client-req.pem -days 365000 -CA
{{ mariadb_ssl_folder }}/ca-cert.pem -CAkey {{ mariadb_ssl_folder }}/ca-key.pem -set_serial
01 -out {{ mariadb_ssl_folder }}/client-cert.pem

- name: Verificar certificados creados. Aborta si no es correcto.
  shell: openssl verify -CAfile {{ mariadb_ssl_folder }}/ca-cert.pem
{{ mariadb_ssl_folder }}/server-cert.pem {{ mariadb_ssl_folder }}/client-cert.pem
  register: ssl_verification_output
  failed_when: ssl_verification_output.rc != 0
```

```
└── openvas_mitig.yml
```

```
---
- debug: msg="Mitigando vulnerabilidades restantes encontradas con OpenVAS"

- blockinfile:
    path: /etc/ssh/sshd_config
    create: no
    backup: yes
    block:  |
        # SSH Weak Encryption Algorithms Supported mitigation (OID:
1.3.6.1.4.1.25623.1.0.105611)
        Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-
gcm@openssh.com,chacha20-poly1305@openssh.com
        # SSH Weak MAC Algorithms Supported (OID: 1.3.6.1.4.1.25623.1.0.105610)
        MACs hmac-sha1-etm@openssh.com,umac-64-etm@openssh.com,umac-128-
etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha2-512-
etm@openssh.com,hmac-ripemd160-etm@openssh.com,hmac-sha1,umac-
64@openssh.com,umac-128@openssh.com,hmac-sha2-256,hmac-sha2-512,hmac-
ripemd160

- blockinfile:
    path: /etc/ssh/ssh_config
    create: no
    backup: yes
    block:  |
        # SSH Weak Encryption Algorithms Supported mitigation (OID:
1.3.6.1.4.1.25623.1.0.105611)
        Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-
gcm@openssh.com,chacha20-poly1305@openssh.com
        # SSH Weak MAC Algorithms Supported (OID: 1.3.6.1.4.1.25623.1.0.105610)
        MACs hmac-sha1-etm@openssh.com,umac-64-etm@openssh.com,umac-128-
etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha2-512-
etm@openssh.com,hmac-ripemd160-etm@openssh.com,hmac-sha1,umac-
64@openssh.com,umac-128@openssh.com,hmac-sha2-256,hmac-sha2-512,hmac-
ripemd160

- blockinfile:
    path: /etc/sysctl.conf
    create: no
    backup: yes
    block:  |
        # TCP timestamps mitigation (OID: 1.3.6.1.4.1.25623.1.0.80091)
        net.ipv4.tcp_timestamps = 0
```

Además, durante el kickstart de instalación se crea un servicio que ejecuta el script "ansible_firstboot.sh" en el primer arranque. Su código es el siguiente:

```
#!/bin/bash
#
# Script que finaliza la configuracion en el primer reinicio tras la
# instalacion. Es necesario hacerlo asi porque el servicio mariadb ha
# de estar levantado. Para aplicar ciertas configuraciones sobre la BD
#
################################################################################

# Directorio /root
cd /root/

# Descargar playbook y ejecutar
curl -O http://192.168.124.1/pub/ansible/play_after_install.yml
logger "ANSIBLE: Ejecutando playbook tras reinicio"
ansible-playbook play_after_install.yml

# Deshabilitar servicio
systemctl disable ansible_firstboot.service

logger "ANSIBLE: Ejecucion finalizada"
```

El contenido del playbook que ejecuta ("`play_after_install.yml`") es:

```
---
# Este playbook es necesario para configurar la BD ya que el servicio ha de estar
arrancado.
- name: Playbook Tras Instalación
  hosts: all
  vars:
    mariadb_root_password: M@ri@DBrootPWD

  tasks:
    - name: Habilitar y arrancar MariaDB
      systemd:
        name: mariadb
        enabled: yes
        state: started

    - name: Eliminando base de datos de test
      mysql_db: name=test state=absent

    - name: Eliminando usuarios anónimos
      mysql_user: name='' state=absent host_all=yes

    - name: Password de root para acceso desde localhost
      mysql_user: name=root host={{item}} password={{mariadb_root_password |
mandatory}} state=present
      with_items:
        - '::1'
        - '127.0.0.1'
        - 'localhost'

# Regla de firewall que permite acceso desde localhost
    - name: Acceso a mariadb desde local
      firewalld:
        immediate: true
        permanent: true
        rich_rule: 'rule family="ipv4" service name="mysql" source address="127.0.0.1"
accept'
        state: enabled
# Una vez configurada, se reinicia el servicio.
    - name: Habilitar y arrancar MariaDB
      systemd:
        name: mariadb
        state: restarted
```