Securización y monitorización de servidores Ubuntu mediante Ansible y Zabbix

Automatizando y monitorizando la seguridad de sistemas Ubuntu Server con Ansible y Zabbix

Santiago Sierra González

Máster Universitario en Ciberseguridad y Privacidad
Seguridad empresarial

Nombre Tutor/a de TF
Amadeu Albós Raya
Profesor/a responsable de la asignatura
Victor Garcia Font

junio de 2023

Universitat Oberta de Catalunya

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-CompartirIgual 3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	***Securización y monitorización de servidores Ubuntu mediante Ansible y Zabbix***
Nombre del autor:	*Santiago Sierra González*
Nombre del consultor/a:	*Amadeu Albós Raya*
Nombre del PRA:	*Victor Garcia Font*
Fecha de entrega (mm/aaaa):	*06/2023*
Titulación o programa:	Máster universitario en Ciberseguridad y Privacidad
Área del Trabajo Final:	*Seguridad empresarial*
Idioma del trabajo:	*Castellano*
Palabras clave	*Seguridad, automatización, monitorización.*

Resumen del Trabajo

La magnitud de las infraestructuras basadas en virtualización ha crecido en gran medida, tanto en tamaño, alcance y complejidad. Esto supone que la gestión de la seguridad de estas infraestructuras también incrementa. Este trabajo pretende explorar el uso de herramientas de automatización y monitorización, como Ansible y Zabbix, para agilizar la configuración de seguridad de servidores virtuales Ubuntu para posteriormente monitorizar su estado con Zabbix.

Este proyecto pretende realizar un análisis inicial sobre el estado de seguridad de las máquinas Ubuntu Server 20.04 para establecer los controles que se deberán automatizar y monitorizar. El objetivo es que Ansible sea capaz de asegurar el correcto estado de los controles y Zabbix sea capaz de monitorizar ciertos aspectos relacionados con la seguridad. De esta forma se puede tener un estándar automatizado y monitorizado para aplicar en serie configuraciones sobre múltiples máquinas de un entorno sin necesidad de realizar las configuraciones manualmente.

Abstract

The scale of virtualization-based infrastructures has grown greatly, both in size, scope and complexity. This means that the security management of these infrastructures is also increasing. This work aims to explore the use of automation and monitoring tools, such as Ansible and Zabbix, to streamline the security configuration of Ubuntu virtual servers and then monitor their status with Zabbix.

This project aims to perform an initial analysis of the security status of Ubuntu Server 20.04 machines to establish the controls to be automated and monitored. The goal is that Ansible should be able to ensure the correct state of the controls and Zabbix should be able to monitor certain aspects related to security. This way it is possible to have an automated and monitored standard to serially apply configurations on multiple machines in an environment without the need to perform the configurations manually.

Índice

Lista de Imágenes

1. Introducción

La digitalización y el ritmo de crecimiento digital a nivel mundial ha convertido en prácticamente un estándar el uso de sistemas informatizados para las actividades de negocio de la mayoría de las empresas.

Este mismo incremento de digitalización conlleva una mayor superficie y complejidad en la administración de los sistemas. Este escenario ha propiciado el incremento de ataques informáticos y la evolución de estos mismos.

1.1. Contexto y justificación del Trabajo

Dado este escenario de crecimiento en superficie, complejidad y evolución constantes de la seguridad de sistemas lleva a que cada vez sea más difícil para las organizaciones mantener sus sistemas seguros.

En este trabajo se pretende explorar una solución de seguridad de la información en entornos productivos basados en la automatización de configuraciones de seguridad y la monitorización de sistemas.

La automatización y la monitorización son elementos clave, ya que la primera reduce el tiempo y recursos de realizar una acción, como bastionar un servidor, y la segunda permite mantener vigilados los sistemas para detectar o prevenir problemas.

En este trabajo se pretende trabajar sobre el entorno interno de la empresa Brain2Store[1], dedicada a la implementación de soluciones de almacenamiento y virtualización para empresas.

La empresa dispone de un entorno de Zabbix[2] interno y otro dedicado a ofrecer un servicio de MaaS (Monitoring as a Service). En este trabajo, se pretende trabajar únicamente sobre el entorno interno para extender la monitorización incluyendo la seguridad de sistemas en un entorno reducido.

Por otra parte, se pretende crear un sistema que permita la automatización de la configuración de seguridad de las máquinas virtuales y sus servicios basado en Ansible[3].

Ambos sistemas se pueden considerar independientes, pero se pueden complementar mutuamente, ya que la automatización puede preparar los sistemas para ser monitorizados y la monitorización puede generar acciones automatizadas a ciertas alertas.

1.2. Objetivos del Trabajo

A continuación, se listan los objetivos de este proyecto:

- Elaboración de mecanismos de automatización y monitorización de la seguridad de sistemas para agilizar la administración de la seguridad de la información.
- Creación de baselines acorde a un estándar oficial, gubernamental o internacional.
- Implementar automatización de la seguridad de sistemas acorde a unas baselines determinadas.
- Implementar la monitorización de la seguridad de sistemas acorde a unas baselines determinadas.
- Investigar la interacción reciproca entre automatización y monitorización para generar acciones automatizadas a ciertas alertas.

1.3. Impacto en sostenibilidad, ético-social y de diversidad

Dentro de los 17 que conforman los Objetivos de Desarrollo Sostenible (ODS)[4] se pueden hacer distinciones para clasificarlos entre aquellos referentes a la sostenibilidad, los relacionados con la ética y responsabilidad social, y aquellos referentes a la diversidad y los derechos humanos

Dentro de la agrupación de sostenibilidad se encontrarían los objetivos de energía asequible y segura (ODS 7), industria innovadora (ODS 9), ciudades sostenibles (ODS 11), consumo y producción responsables (ODS 12), acción climática (ODS 13) y preservación de la vida marina y terrestre (ODS 14 y 15).

Dentro de este grupo se pretende observar un impacto positivo en los puntos de industria y ciudades sostenibles, puesto que el objetivo de este proyecto es optimizar o agilizar la administración de la seguridad de sistemas, reduciendo el tiempo y los recursos necesarios para desplegar, mantener y monitorizar una infraestructura. En el resto de los puntos no se observaría ningún impacto apreciable.

En este segundo grupo podríamos incluir los puntos de erradicar la pobreza (ODS 1), acabar con el hambre en el mundo (ODS 2), acceso a agua potable (ODS 6), crecimiento económico sostenible (ODS 8) y justicia e instituciones justas (ODS 16).

En cuanto a esta dimensión, se puede intuir un impacto positivo, aunque más indirecto, ya que se otorgan una serie de herramientas y mecanismos que faciliten la gestión de la ciberseguridad a los administradores de sistemas a la hora de producir entornos más seguros para las empresas. Lo que permitiría un crecimiento económico más seguro acorde al punto 8 de los ODS mencionado anteriormente.

Finalmente, en el ámbito de diversidad y derechos humanos tendríamos los objetivos de igualdad de género (ODS 5) y reducción de la desigualdad (ODS 10). En esta dimensión se espera que el impacto sea nulo en ambos ODS nulo dada la naturaleza tan estrictamente técnica del proyecto.

1.4. Enfoque y método seguido

Este proyecto se puede ver de diferentes puntos de vista. Si nos centramos en la visión global podemos seguir de forma secuencial, ya que primero se debe preparar un entorno, realizar unas líneas base, una implementación, unas pruebas y finalmente una toma de resultados.

Pero, por otra parte, la automatización y la monitorización pueden interactuar recíprocamente, por lo que se puede optar por un desarrollo en paralelo o intercalado que vaya complementando las otras partes. Se pueden automatizar configuraciones y se pueden crear acciones automatizadas a cierta monitorización.

Por lo que para este proyecto me he decantado por una metodología híbrida entre desarrollo en cascada para desarrollo global del proyecto y un desarrollo en paralelo más iterativo para la parte de implementación de las soluciones.

La primera etapa, **Investigación y preparación**, se debe investigar el estado del arte y posibilidades para este proyecto, decidir el alcance del proyecto, preparar el laboratorio y documentarlo.

Posteriormente, en la segunda etapa de **Elaboración de las baselines**, se realizaría un análisis de riesgos y/o técnico para encontrar los problemas de seguridad y realizar unos objetivos a cumplir.

En la tercera etapa de **Implementación de soluciones**, se trasladarían estas baselines a Ansible y Zabbix para conseguir los resultados de dichos objetivos.

Una vez se implementen todas las soluciones, entrando en el **Análisis de resultados**, se realizarán pruebas de despliegue y monitorización para revisar el estado final de seguridad de los sistemas y evaluar la mejora y funcionalidades añadidas.

Para acabar, quedaría una última etapa **Conclusiones y revisión**, para elaborar las conclusiones, redactar la documentación y revisar de forma global el proyecto.

1.5. Planificación del Trabajo

Este proyecto contiene, como se ha comentado anteriormente, una primera etapa de investigación sobre el estado del arte, estándares de seguridad y posibles enfoques del proyecto. Seguido de una toma de decisiones sobre el alcance del proyecto y objetivos principales.

Para posteriormente, proceder con el diseño del laboratorio sobre el que trabajar. Una vez diseñado se procederá a crear las máquinas necesarias para dicho laboratorio.

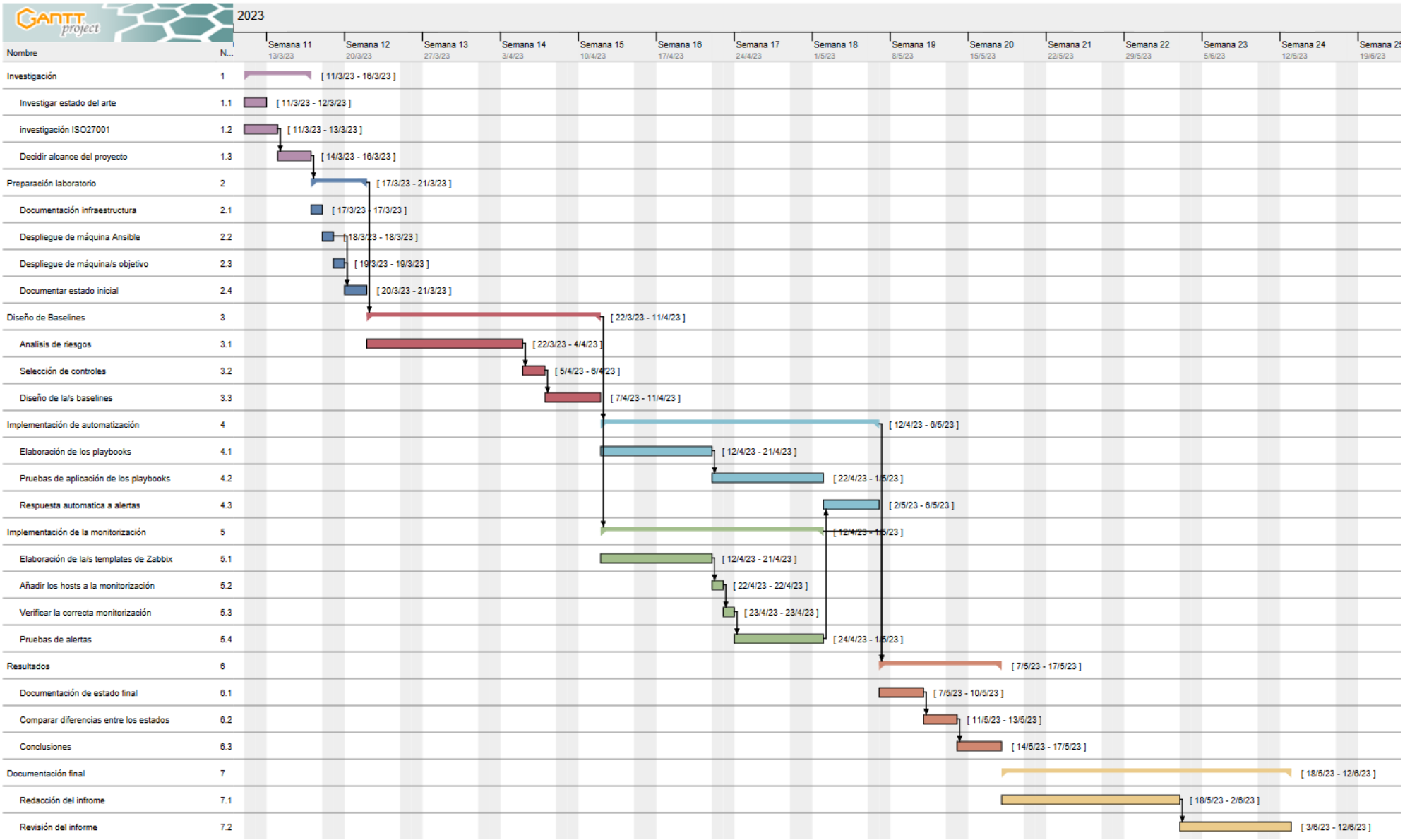
Entrando en la etapa de Elaboración de baselines, se realizará un análisis de riesgos para valorar el estado inicial y las posibles amenazas a las que se exponen los sistemas del laboratorio.

Cuando el análisis esté completado, se podrá hacer una selección de qué medidas se deberían implementar y elaborar unas baselines que marquen los objetivos de seguridad a conseguir.
La tercera etapa consiste en trasladar esas baselines a los playbooks de Ansible[5] y templates de Zabbix[6] para ir añadiendo las configuraciones, monitorizaciones y acciones automáticas correspondientes.

Una vez finalizada la implementación quedaría abordar la penúltima etapa, donde se deberán realizar pruebas para verificar que ambos sistemas son capaces preparar y monitorizar los sistemas acorde a dichas directrices. Con estos resultados se podrán comparar los estados iniciales y los finales y valorar la mejora y funcionalidades obtenidas.

Finalmente, en la última etapa, quedaría la elaboración de las conclusiones y del resto del documento para la su posterior, y final, revisión.

Diagrama de Gantt:



1.6. Breve sumario de productos obtenidos

El principal producto de este trabajo es el conjunto del sistema de automatización y monitorización realizados partir de las herramientas Ansible y Zabbix. Un sistema pensado para funcionar con dichas herramientas para conseguir unos despliegues bastionados y una posterior monitorización para mantener al día la seguridad de dichas máquinas.

1.7. Breve descripción de los otros capítulos de la memoria

En el siguiente capitulo se presentará un **estudio preliminar**, con una pequeña descripción contextual del estado del arte y la valoración de los objetivos y alcance del proyecto.

Posteriormente se introducirá el **entorno de laboratorio** sobre el que se trabajará y como se pretende modificar o adaptar para llevar a cabo el proyecto.

El siguiente paso es la **auditoría y la selección de controles** para la realización de los objetivos sobre que configuraciones se pretenden aplicar y que aspectos monitorizar.

Antes de empezar con la implementación es necesario **desarrollar de forma teórica las tecnologías** con las que se va a trabajar y que otras herramientas o tecnologías existen.

El siguiente paso es mostrar la **implementación** para mostrar en el siguiente capitulo los resultados.

Finalmente se **analizarán los resultados** para extraer unas **conclusiones** en base a los resultados obtenidos.

2. Estudio preliminar

2.1. Estado del Arte

Desde hace varios años se observa una tendencia en la migración de los procesos de negocio a servicios Cloud[7]. Las infraestructuras de los proveedores son principalmente compuestas por sistemas basados en Linux, dada su eficiencia y versatilidad.

Pese a ello, el modelo de solución On-premise sigue siendo el modelo dominante más extendido[8]. Este modelo consiste en el alojamiento y mantenimiento de la infraestructura por parte del cliente.

Dentro de estas soluciones On-premise, existe una gran heterogeneidad en la visión global debido a las necesidades únicas de cada organización y la diversidad de soluciones disponibles.

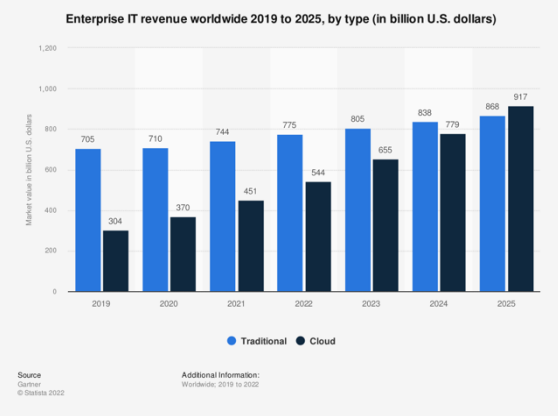


Imagen 1: Beneficios en infraestructura On-Premise/Cloud [9]
14/6/23 23:04:00

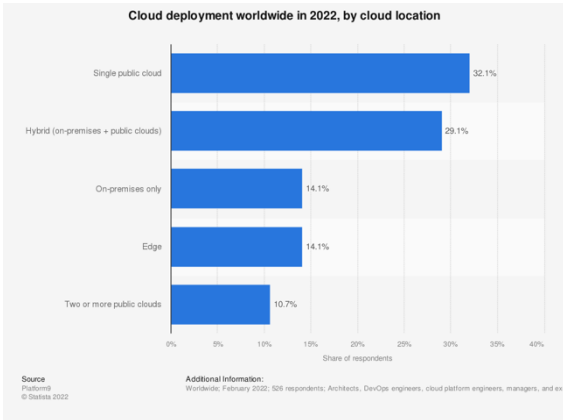


Imagen 2: Despliegue de infraestructura por tipo de Cloud/On-Premise [10]

Pero se puede observar que la virtualización es prácticamente una constante hoy en día. Lo habitual es disponer de servidores virtualizando todas las máquinas necesarias para los procesos de negocios.

Entrando en los sistemas operativos vemos que Windows Server es el sistema predominante, con alrededor del 70% de uso[11]. Windows Server es más extendido por la popularidad de su versión de escritorio, la facilidad de gestión básica y la utilización de Active Directory. Pero requiere de licencias y un mayor consumo de recursos.

Linux, al ser un sistema más eficiente, estable, seguro y polivalente, está más presente en la infraestructura Cloud, servicios segmentados o servicios basados en open source. Según encuestas de W3Techs, Unix es usado en el 80% de los sitios web conocidos (teniendo en cuenta que un sitio web puede tener ambos sistemas)[12].

Este crecimiento en tamaño y complejidad obliga a que el paradigma de la administración se adapte. De la misma forma que el Cloud es una opción adoptada debido a que simplifica algunos aspectos de la infraestructura, las herramientas de gestión avanzan con un objetivo similar, el de simplificar la administración de sistemas complejos.

En el ámbito de la virtualización, herramientas para la administración de múltiples máquinas de forma centralizada suponen una mejora en la calidad de la infraestructura. Por eso se ha extendido el uso de Windows Active Directory, porque dispone de mecanismos de gestión de dominios corporativos enteros si están basados en Windows.

También es útil disponer de herramientas para el desarrollo, despliegue y mantenimiento de infraestructuras que permitan una gestión más simple y centralizada que la de la configuración manual.

Ahí es donde podríamos ver herramientas como Ansible, Chef, Puppet o SaltStack para este tipo de tareas. O podríamos mantener un control del estado de la organización con herramientas de monitorización como Zabbix, Nagios o Tenable Security Center.

2.2. Selección del alcance del proyecto

Dado el amplio alcance de la seguridad de un sistema he preferido centrar el desarrollo del proyecto en torno a un sistema, o familia de sistemas, concreto. He decidido centrarme en la seguridad de sistemas Linux por dos principales razones.

La primera es porque, a pesar de ser utilizado solamente en menos del 25% de los servidores[13], son principalmente utilizados para alojar servicios web, bases de datos u otros servicios por su mejor rendimiento y consumo de recursos. Windows se utiliza principalmente para dominio corporativo mediante Active Directory o para entornos muy simples.

La segunda razón es que suele ser más difícil, complejo y/o tedioso realizar configuraciones en sistemas Linux, especialmente si no disponen de interfaz gráfica. Por lo que es más probable que se pase por alto o no se disponga del personal capacitado para configurar adecuadamente dichos servidores. Exponiendo de esta forma los sistemas a más vulnerabilidades.

3. Entorno de laboratorio

3.1. Resumen de la infraestructura

El entorno sobre el que trabajar está basado sobre VMWare vSphere 7[14] sobre un hardware de 2 servidores ESXi y una cabina de almacenamiento NetApp[15]. Es un entorno simple para alojar las máquinas virtuales para la infraestructura interna y las máquinas para los sistemas Zabbix, uno interno y uno para el servicio MaaS, B2Alert.

Toda la infraestructura se encuentra en el mismo rack. La cabina de datos NetApp contiene los discos y es la que aloja los volúmenes de los datastores para vSphere. La virtualización utiliza estos datastores para alojar físicamente las máquinas virtuales del entorno productivo de la empresa.

Los datos de la infraestructura tienen 3 niveles de respaldo de datos. Se realizan snapshot a nivel de volumen en la cabina. Los snapshots se replican contra otra cabina NetApp de respaldo. Y finalmente se realizan copias a cintas mediante una librería de cintas.

Para este proyecto no es necesario disponer de un hardware o configuración de infraestructura especifica, ya que todo el proyecto se desarrolla a nivel de sistema operativos sobre un entorno de virtualización.

3.2. Entorno de Zabbix

La monitorización mediante Zabbix está implementada de forma no convencional, ya que el entorno está separado en 3 máquinas, como se puede apreciar en la Imagen 3. Para cubrir las funciones de base de datos (B2S-ZABBIX-BD01-PRE), aplicación (B2S-ZABBIX-APP01-PRE) y frontend (B2S-ZABBIX-FE01-PRE) de forma independiente, pese a que Zabbix se puede instalar en una sola maquina[16] o desplegando un appliance virtual[17].

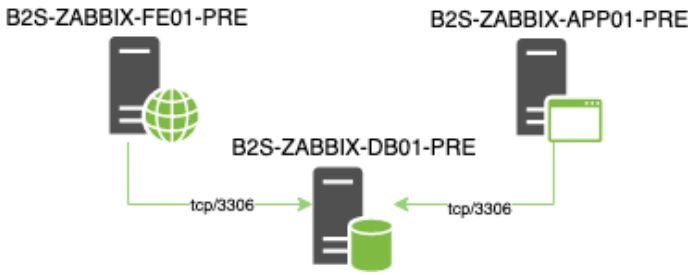


Imagen 3: Diagrama lógico de las VMs de Zabbix

Todas las máquinas son CentOS 7 con la versión 5.0 LTS de Zabbix, instalando en las máquinas de frontend y aplicación los respectivos paquetes para dichos roles.

La máquina con el rol de base de datos solo contiene MariaDB como DBMS (DataBase Maangement System). La base de datos se encuentra en una unidad montada mediante una LUN (Logical Unit Number) que a nivel práctico se comporta como si estuviera conectando un disco físico a esa máquina.

La base de datos en la máquina B2S-ZABBIX-BD01-PRE está almacenada en una unidad de disco independiente al disco del sistema proporcionado mediante iSCSI, de forma que se muestra como si fuera un disco físico conectado.

Diariamente se realiza una copia de la base de datos de Zabbix hacia un volumen NFS alojado en la cabina de respaldo. Por lo que además de disponer de copias a nivel de volumen y máquina virtual, también hay copias de la base de datos por separado.

3.3. Automatización

Para llevar a cabo la automatización se ha creado un servidor con Ubuntu Server 20.04 en el que se ha instalado la aplicación de Ansible posteriormente añadiendo el repositorio e instalando el paquete siguiendo la documentación oficial[18].

```
$ sudo add-apt-repository --yes --update ppa:ansible/ansible
$ sudo apt install ansible
```

Una vez instalado la herramienta de Ansible ya se podrían desarrollar los playbooks y roles para ejecutarlos desde esta máquina y que se realicen las acciones. Ansible no requiere de configuraciones en una primera instancia, ya que las configuraciones se pueden aplicar mediante archivos de configuración en el directorio desde donde se van a ejecutar los playbooks.

La máquina servidor de Ansible debe tener acceso mediante SSH a todas las máquinas objetivo. Como todas las máquinas se encuentran en la misma subred no habrá problema, pero se debe tener en cuenta si es necesario revisar el enrutamiento o las reglas de algún firewall si alguno de estos estuviera entre esta máquina y las máquinas objetivo.

3.4. Máquina/s objetivo

Ubuntu es un sistema operativo conocido, versátil y relativamente simple comparado con otros sistemas Linux/Unix. Ubuntu es útil principalmente para utilidades o servidores de uso interno para la organización. Se pueden desplegar herramientas de monitorización, servidores Apache, bases de datos internas o servidores de syslog entre otros.

Las acciones de configuración y monitorización se realizarán sobre una máquina Linux Ubuntu Server 20.04 siguiendo una instalación por defecto. Se ha optado por esta distribución porque pertenece a la gama de entrada de servidores Linux y es una distribución ampliamente conocida y utilizada.

Se pretende utilizar máquinas virtuales con las especificaciones mínimas para Ubuntu: 2 CPUs, 4GB de memoria y 16GB de disco duro.

Utilizando la imagen de Ubuntu se configurarán las opciones básicas como una IP estática, DNS de Google (8.8.8.8 y 8.8.4.4), usuario del sistema e instalación opcional de OpenSSH.

A partir de ahí, se puede hacer el estudio para valorar el estado inicial en base a unos controles, seleccionar cuales son más relevantes y crear los mecanismos de corrección y monitorización de los puntos seleccionados.

4. Auditoría y selección de controles

Antes de poder realizar cualquier acción de automatización y monitorización se deben tener claros los estados iniciales y finales de los sistemas, por lo que es necesario realizar una auditoria previa para determinar ese estado inicial y elegir qué puntos se quieren corregir o mejorar.

Pera evaluar estos puntos se deben elegir unos criterios, que pueden ser propios o basados en algún estándar. Para facilitar esta labor existen herramientas para evaluar y asegurar el cumplimiento de estándares y especificaciones.

4.1. Estándares de seguridad

A la hora de configurar sistemas, aplicaciones o servicios existen infinidad de posibles formas de llevarlo a cabo y el alcance de las configuraciones puede no quedar claro. Por eso existen estándares que marcan unas directrices de configuración que forman un marco homogeneizador para la configuración segura de los sistemas.

Estos estándares pueden ser del propio fabricante, gubernamentales o internacionales. El standard del DISA (Defense Information Systems Agency) estadounidense y el del CIS (Center of Internet Security) internacional son 2 de los más conocidos y utilizados. Existen algunos específicos como el PCI-DSS (Payment Card Industry Data Security Standard) que es un estándar para la configuración de sistemas de proceso, almacenamiento o intercambio de datos de tarjetas de pago.

Por lo que estos estándares recopilan una serie de controles para comprobar si el sistema cumple con los requisitos de seguridad de dicho estándar. En el caso de un sistema operativo, puede haber controles para revisar la política de contraseñas, permisos de usuario o deshabilitar servicios inseguros, por ejemplo.

Estos estándares no son inmutables y pueden ser adaptables mediante el uso de perfiles de cumplimiento en función de factores como el alcance o el uso. Cada estándar puede estructurar de forma diferente los controles para formar los perfiles y ajustar al alcance y profundidad de cada perfil. En el caso del CIS se tienen 2 niveles, un primer nivel para cubrir los puntos más críticos y un segundo nivel para cubrir elementos más específicos y con más detalle.

4.2. Introducción a SCAP

El Security Content Automation Protocol[19] es un protocolo desarrollado e impulsado por el NIST para crear un formato común y unificado para la administración de configuraciones y aspectos de seguridad de sistemas.

SCAP incluye protocolos como el CVE (Common Vulnerabilities and Exposures), CPE (Common Platform Enumeration) o CCE (Common Configuration Enumeration), por ejemplo. Así como otros protocolos o estándares de información de seguridad, para el formato del contenido o de los resultados. Esto permite crear unas checklists, que serían equivalentes a listas de la compra, pero en aspectos de seguridad.

Al tener este marco de referencia común se facilita la creación, o adaptación, de herramientas para hacer uso de este para el cumplimiento de requisitos de seguridad. Estos estándares, o cualquier estándar, se puede codificar con este protocolo para que se pueda automatizar la verificación de los controles.

4.2.1. OpenSCAP

OpenSCAP[20] es una herramienta de código abierto para la aplicación del protocolo SCAP y poder utilizar checklists para evaluar el estado de los sistemas y servicios. Esta herramienta será útil para evaluar el estado de las máquinas objetivo de este proyecto antes y después del desarrollo del proyecto.

OpenSCAP ofrece una serie de checklists preparadas para cumplir ciertos estándares en ciertos sistemas. Para este caso, se ha utilizado el estándar del CIS (Center of Internet Security) disponible en el repositorio oficial de OpenSCAP[21].

De esta forma se puede agilizar la comprobación del cumplimiento en lugar de realizar las comprobaciones a mano todas las veces. En el caso de una organización, el tiempo necesario para realizar las comprobaciones en todas las máquinas se reduce drásticamente ya que la comprobación de una checklist puede tardar cuestión de minutos en lugar de horas.

Aunque OpenSCAP es de código abierto, existen otro tipo de herramientas se software propietario como Nessus[22]; que también permiten la ejecución de checklists de SCAP. Nessus es una herramienta más completa e intuitiva, pero para el propósito de este proyecto OpenSCAP proporciona las funciones necesarias.

4.2.2. Uso de OpenSCAP

OpenSCAP tiene cliente en línea de comandos y con interfaz gráfica. En nuestro caso se ha utilizado el cliente de terminal ya que los servidores objetivo no tienen interfaz gráfica.

En el caso de Ubuntu el cliente lo podemos instalar con el siguiente comando:

```
$ sudo apt install libopenscap8
```

OpenSCAP permite la ejecución remota de comprobaciones, de forma que desde el cliente se realiza una conexión SSH contra el sistema en cuestión, ejecuta todas las pruebas y extrae los resultados. Este sistema permite una gestión más centralizada, aunque sigue siendo necesario tener instalado OpenSCAP en el destino.

OpenSCAP permite exportar los resultados en formato XML o como informe HTML. Este último es especialmente útil ya que nos muestra una vista con estadísticas, resúmenes y vistas detalladas de todos las pruebas y sus resultados. Como SCAP permite la inclusión de comentarios en los checks de una checklist, es habitual ver en estándares reconocidos que los controles suelen tener instrucciones, o incluso scripts, sobre cómo solucionar los controles no superados.

Una checklist puede contener todos los perfiles de uno o más estándares. La checklist contiene todos los controles y cada perfil tiene una selección de controles asociado. De esta forma se permite una mayor personalización sin necesidad de utilizar un archivo para cada perfil y tener duplicidad de gran parte del contenido, como los controles.

Como se ha comentado anteriormente, el estándar del CIS dispone de 2 niveles, por lo que en el caso de la checklist tenemos que cada nivel está configurado como un perfil. Con el comando `oscap info` podemos extraer información sobre una checklist, o acceder directamente al listado de perfiles con el parámetro `--profiles`.

```
ssierrago@tfm-ubuntu01:~$ oscap info --profiles ssg-ubuntu2004-ds.xml
xccdf_org.ssgproject.content_profile_cis_level1_server:CIS Ubuntu 20.04 Level 1 Server Benchmark
xccdf_org.ssgproject.content_profile_cis_level1_workstation:CIS Ubuntu 20.04 Level 1 Workstation Benchmark
xccdf_org.ssgproject.content_profile_cis_level2_server:CIS Ubuntu 20.04 Level 2 Server Benchmark
xccdf_org.ssgproject.content_profile_cis_level2_workstation:CIS Ubuntu 20.04 Level 2 Workstation Benchmark
xccdf_org.ssgproject.content_profile_standard:Standard System Security Profile for Ubuntu 20.04
xccdf_org.ssgproject.content_profile_stig:Canonical Ubuntu 20.04 LTS Security Technical Implementation Guide (STIG) V1R1
```

Imagen 4: Visualización de los perfiles dentro de una checklist

Una vez tengamos OpenSCAP instalado y la checklist descargada podemos proceder a ejecutar la herramienta con el siguiente comando:

```
$ oscap xccdf eval --report <pathHTML> --profile <perfil> <checklist>
```

De esta forma se ejecutarán las comprobaciones del perfil seleccionado y obtendremos un informe en formato HTML en el lugar que le hayamos especificado. Este informe nos servirá en el siguiente punto para realizar la valoración inicial del sistema.

4.3. Estudio inicial sobre Ubuntu

4.3.1. Estado inicial

Después de instalar la herramienta de openSCAP en la máquina y ejecutar la prueba se puede obtener un resultado rápido sobre el estado de cumplimiento con el estándar del CIS nivel 1.

Una vez realizado el análisis se muestra como el sistema cumple con algunos de los requisitos, pero hay ciertos aspectos de seguridad que no se encuentran correctamente configurados.

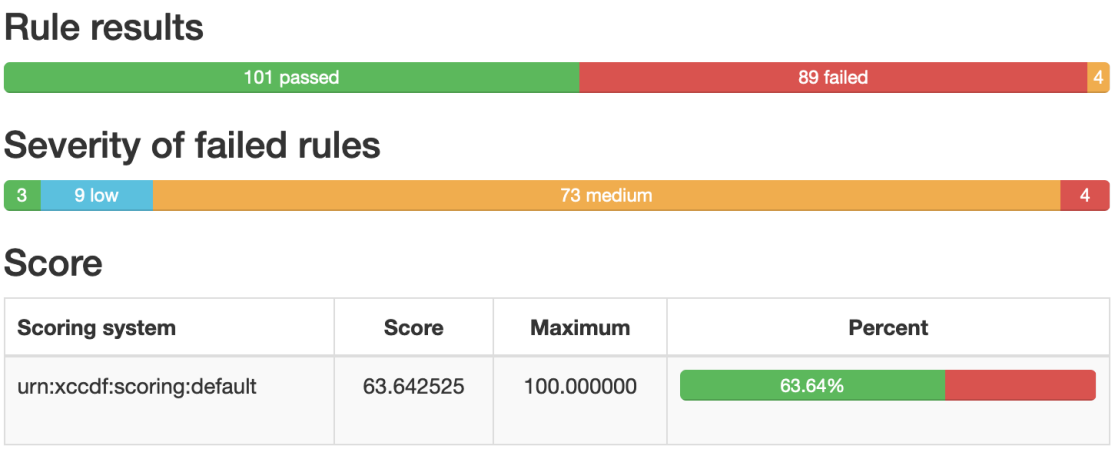


Imagen 5: Resultados OpenSCAP perfil CIS nivel 1 en Ubuntu 20.04

Como se puede ver en la Imagen 5, se puede observar que se han superado el 63,64% de las comprobaciones.

Entre estos puntos deficientes vemos que los 4 considerados como graves se corresponden con proteger con contraseña el gestor de arranque (grub2), instalar el NTP (Network Time Protocol), habilitarlo y deshabilitar los accesos sin contraseña por SSH.

Entre el resto de los puntos se pueden destacar los relacionados con la política de contraseñas, permisos de archivos y usuarios, configuraciones de red, configuraciones de logs y de algunos servicios.

4.3.2. Selección de controles

Para realizar la baseline es necesario hacer una selección de los controles que se tendrán en cuenta. Estos controles son necesarios tanto para la parte de la automatización con Ansible como para la monitorización con Zabbix.

Por lo que, habiendo evaluado los puntos fallidos y superados, se puede hacer una selección de controles a tener en cuenta a la hora de elaborar los objetivos de la baseline.

- **AIDE (Advanced Intrusion Detection Environment):** Sistema de verificación de integridad de archivos. El objetivo es instalarlo y configurarlo para verificar periódicamente la integridad de los archivos.
- **Administración de permisos:** Hay varios aspectos que giran en torno a los permisos sobre archivos o acciones sobre el sistema.
    - **Permisos de sudo:** Limitar los comandos de sudo mediante pseudo-terminal y habilitar los logs de comandos de sudo.
    - **Permisos de Umask:** Esta opción afecta a los permisos por defecto de un usuario o servicios a la hora de crear archivos y directorios.
    - **Permisos de usuarios:** Limitar los permisos sobre ciertos componentes del sistema por parte de los usuarios sin permisos de administrador.
    - **Permisos sobre archivos críticos:** Los archivos críticos del sistema y ciertos servicios (claves SSH por ejemplo) deben ser solo accesibles para los usuarios necesarios para el correcto funcionamiento del sistema o servicio.
- **Política de passwords:** Las pruebas fallidas entorno a este punto son las que especifican que los requisitos de longitud, complejidad y rotación de las contraseñas. Por lo que un punto importante es aplicar las configuraciones para asegurar que las contraseñas sean robustas y se actualicen periódicamente.
- **Configuración del GRUB:** Asegurarse de proteger la modificación del GRUB mediante contraseña y limitar los permisos de lectura y escritura para evitar modificaciones por parte de usuarios con pocos privilegios.
- **Configuración de red:** Los principales puntos a tratar son el uso de firewall, deshabilitar IPv6 y deshabilitar opciones a nivel de kernel para limitar acciones que no son necesarias para un rol de servidor.
- **Servicios del sistema:** El sistema viene con unos servicios que pueden ser configurados o eliminados para ofrecer un funcionamiento más seguro.
    - **NTP:** El Network Time Protocol permite mantener la misma hora entra varios dispositivos, lo que es crucial para tener una buena trazabilidad de los eventos. Por lo que es importante configurarlo y monitorizar su correcto funcionamiento.
    - **Telnet:** Es un servicio para conexiones remotas inseguro y no debería utilizarse, por lo que el objetivo es deshabilitarlo y monitorizar que no se habilite.

    - **SSH:** Es un protocolo mucho más seguro que telnet, pero se puede hacer más seguro limitando los intentos, los timeouts, las sesiones y monitorizando los logs.
    - **Rsyslog:** Los logs de eventos deberían registrarse en local, y en un servidor remoto a ser posible. El servicio Rsyslog permite la gestión centralizada de los diferentes logs del sistema y la posibilidad del envío de esos logs a otros servidores para almacenamiento de logs.
- **Paquetes instalados:** Se debe asegurar una instalación mínima para ajustarse a que la máquina disponga de los servicios y paquetes estrictamente necesarios para su funcionamiento. Cuanto menor sea la cantidad de software instalado, menor es la superficie de ataque.

4.3.3. Elaboración de la baseline

Partiendo de los controles seleccionados anteriormente y seleccionando algunos elementos adicionales como la configuración del firewall del sistema o la configuración de las dependencias necesarias para Zabbix podemos elaborar nuestra baseline con los objetivos.

Por lo que el resumen de la baseline sería el siguiente:

- Ubuntu
    - Monitorizar
        - Acciones con sudo
        - Servicios críticos
        - Checksums de archivos
        - Errores de acceso por SSH
    - Automatizar
        - Configuración de AIDE
        - Permisos de usuario
        - Política de contraseñas
        - Configurar firewall
        - Configuración de sudo
        - Configuración de Rsyslog
        - Configurar SNMP (Zabbix)
        - Configurar opciones de red
        - Configurar agente de Zabbix
        - Configuración de permisos y protección del GRUB
        - Configurar NTP
        - Eliminar Telnet
        - Eliminar paquetes y servicios innecesarios
        - Configuración segura de SSH

5. Tecnologías para la implementación

Como ya se ha comentado anteriormente, la automatización y monitorización mantienen una simbiosis donde se complementen e influyen mutuamente. La primera puede configurar o mantener la monitorización, y esta segunda puede avisar para tomar acciones automatizadas.

La automatización en el caso de este trabajo responde a automatizar las tareas de configuración del sistema, para evitar tener que realizar todas esas configuraciones una a una y tener cierta personalización que plantillas de máquinas virtuales (formato OVA) no ofrecen. Ansible permite realizar este tipo de tareas de forma sencilla y rápida.

Y en este caso, la monitorización pretende monitorizar aspectos básicos de una máquina, como el consumo de recursos, y aspectos orientados a la seguridad como los accesos fallidos o la ejecución de código con permisos de administrador.

Zabbix es una herramienta para la monitorización de recursos en general. Presenta una buena integración con gran variedad de dispositivos, no solo servidores y máquinas virtuales. Pero el objetivo es intentar añadir esa funcionalidad adicional para la monitorización de estas características de seguridad.

El uso de ambas herramientas permitirá la creación de los mecanismos (playbook de Ansible y template de Zabbix) para que se puedan cumplir los puntos de la baseline establecida. A continuación, se detallará más a fondo ambas herramientas y su funcionamiento.

5.1. Ansible

Ansible es una aplicación open-source destinada a la automatización de despliegues, configuraciones o actualizaciones de sistemas y aplicaciones.

Esta herramienta permite crear una serie de recetas, llamados playbooks[5], donde podemos especificar las tareas, o requisitos, que se quieren completar. Dentro de un playbook podemos especificar que tareas realizar, sobre que hosts o que variables son necesarias, por ejemplo.

Ansible dispones de módulos para llevar a cabo dichas tareas. Estos módulos permiten instalar aplicaciones, administrar servicios, buscar parámetros en archivos de configuración, ejecutar comandos o copiar archivos entre otras muchas acciones.

Estas funciones permiten la ejecución de grupos de instrucciones a múltiples hosts para la automatización de despliegues o de configuración de sistemas. En nuestro caso queremos que Ansible sea capaz de aplicar los puntos de la baseline.

Los archivos de Ansible se estructuran en formato YAML, lenguaje de serialización de datos, similar al formato JSON. Este formato permite estructurar la información de las tareas, inventarios, roles, variables, etc. Todas las tareas, variables, handlers y playbooks se estructuran como listas y diccionarios anidados para especificar cada elemento y sus parámetros.

5.1.1. Módulos y tareas

Los módulos[23] son las herramientas que Ansible utiliza para realizar tareas específicas. Estos módulos se especifican en una tarea para indicar que se debe realizar y que parámetros.

En el siguiente ejemplo se muestra un ejemplo de cómo especificar una tarea para instalar NTP en formato YAML. Todos los elementos se deben procesar como una lista de ítems. El "-" indica una lista y las entradas indentadas se interpretan como diccionarios.

```yaml
# Install NTP package
- name: Install NTP
  package:
    name: ntp
    state: present
  tags:
    - hardening
    - services
```

Imagen 6: Tarea en Ansible para asegurar que NTP se encuentra instalado.

En formato json suele ser más fácil de interpretar, sobre todo si se está familiarizado con la programación, y se vería de la siguiente manera.

```json
[
  {
    "name": "Install NTP",
    "package": {
      "name": "ntp",
      "state": "present"
    },
    "tags": [
      "hardening",
      "services"
    ]
  }
]
```

Por lo que esta tarea con nombre "Install NTP" está indicando usar el módulo package para asegurarse de que el paquete ntp esté instalado. Si ya se encuentra instalado Ansible lo dará por correcto y si no, intentará instalarlo utilizando las herramientas que forman parte del motor de Ansible.

Cada módulo tiene sus funcionalidades y sus parámetros. Pero la principal ventaja es que muchos módulos permiten funcionar dando información de muy alto nivel. No es necesario encontrar el comando para reiniciar un servicio o saber que herramienta utilizar, simplemente se especifica que se quiere cierto servicio reiniciado.

5.1.2. Roles y estructura

Ansible tiene una estructura bastante definida, aunque no obligatoria, para estructurar los playbooks y los archivos. Si bien podemos crear un playbook con toda la información, la práctica más habitual es el uso de roles. Al crear una carpeta "roles" dentro de la carpeta de los playbooks, podemos poner en nuestro playbook que cierto grupo de hosts (o todos) tengan ese rol. Esto hará que al ejecutar el playbook se apliquen todas las tareas dentro de ese rol a ese grupo de hosts.

Los roles son una forma más visual e intuitiva de ver y agrupar estos conjuntos de tareas, ya que es más fácil verlo como asignar a ciertas máquinas un rol que cumplir, y asegurar todos los requisitos que ello conlleva.
Si bien se pueden crear todos los archivos de un rol manualmente, se puede utilizar el comando `ansible-galaxy init <rol>` dentro de la carpeta de roles para crear toda la estructura de directorios y archivos para dicho rol.[24]

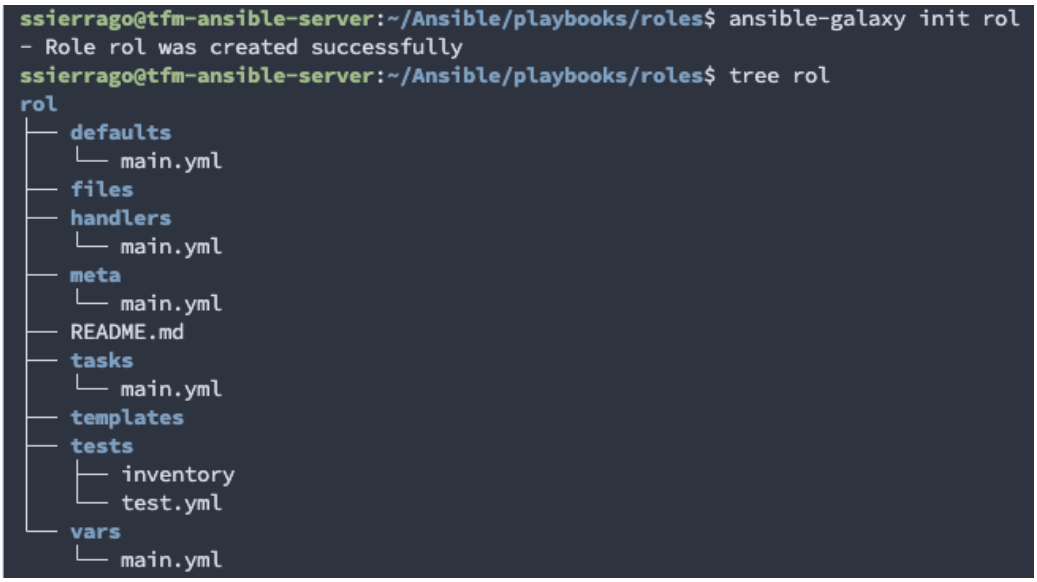


Imagen 7: Estructura de los archivos de un rol

Esta estructura es más ordenada e intuitiva a la hora de configurar los diferentes elementos que componen el rol.

Ansible Galaxy[25] es un repositorio público para la publicación y acceso a playbooks y roles para la comunidad. Para este proyecto se ha creado un rol desde cero pero dada la modularidad de Ansible siempre se pueden utilizar roles obtenidos del portal y adaptarlos a las necesidades específicas que se quieran cumplir.

5.1.3. Inventarios

A la hora de ejecutar un playbook necesitamos elegir sobre que hosts o grupo de hosts ejecutarlo. Para ello podemos especificarlo directamente, al ejecutar o en el playbook, o se pueden crear inventarios.

Un inventario en Ansible no deja de ser un listado jerárquico de las máquinas que se tienen en una infraestructura. Este archivo puede estar en formato YAML o INI, como se puede apreciar en la Imagen 8 e Imagen 9 respectivamente.

```yaml
ubuntu:
  hosts:
    10.1.11.6:
    10.1.11.20:
    # ...

rhel:
 hosts:
    10.1.11.7:
    # ...
...
```

Imagen 8: Ejemplo de inventario en YAML

```ini
[all]
[ubuntu]
10.1.11.6
10.1.11.20
# ...

[rhel]
10.1.11.7
# ...
```

Imagen 9: Ejemplo de inventario en INI

Al utilizar un archivo de inventario podemos especificar en la configuración de ansible (o en un archivo ansible.cfg en el raíz del proyecto) cual es el fichero de inventario y de esta forma poder referenciar directamente a las máquinas o grupos definidos en el inventario dentro de un playbook o rol.

5.1.4. Flujo de Ejecución

Para ejecutar un playbook con uso de los roles, necesitamos crear un archivo donde especifiquemos los datos necesarios para la ejecución del playbook.

```yaml
---
- name: Hardening and monitoring of Ubuntu Systems
  hosts: ubuntu
  gather_facts: true
  roles:
    - ubuntu20.04
...
```

Imagen 10: Playbook para el rol Ubuntu20.04

En el caso de la Imagen 10 tenemos especificado el nombre del playbook, el grupo de host sobre el que aplicar y la asignación del rol de Ubuntu.

La opción `gather_facts` permite a Ansible recolectar información sobre las máquinas objetivo. Esto permite utilizar cierta información para alterar el comportamiento o para personalizar algunas tareas. Por ejemplo, utilizando la información de la distribución extraída de esta forma podemos hacer que cierta tarea no se ejecute en ciertas distribuciones.

Cuando se ejecuta un playbook como el mostrado anteriormente, Ansible intentará ejecutar todas las tareas del rol para el grupo de hosts especificado.

5.1.5. Control de errores y handlers

Durante la ejecución puede ser que haya acciones solo sean necesarias si se han realizado cambios. Los handlers son tareas que permiten ser llamadas por otras tareas cuando estas últimas han realizado algún cambio.

Por ejemplo, un servicio solo es necesario reiniciarlo si se ha hecho algún cambio, por lo que la tarea de configuración puede llamar al handler de reinicio del servicio si ha realizado un cambio. Si nada ha cambiado no se realiza esta llamada y el servicio no se reinicia.

Pero también pueden fallar tareas o que surjan circunstancias no deseables, para lo que también existen mecanismos similares al "try…catch" de otros lenguajes de programación.

Un "try…catch" es una instrucción que permite ejecutar un código comprendido dentro del try y actuar ante ciertos errores y/o excepciones en el catch para actuar ante ciertos errores en tiempo de ejecución.

Ansible permite agrupar tareas con `block` y poder realizar tareas en caso de fallo con `rescue`. De esta forma podemos especificar que si un grupo de tareas falla a cierto punto se ejecuten acciones como intentar otra solución o revertir los cambios.

5.2. Alternativas a Ansible

Las principales alternativas a Ansible son Puppet[26] y SaltStack[27]. Pese a que todas son herramientas para la automatización de tareas de configuración y administración de sistemas, tienen sus diferencias en su enfoque y mecanismos de funcionamiento.

Ansible se puede considerar las más intuitiva y fácil de usar porque se ha concebido con la idea de ser intuitiva y accesible. Por ejemplo, el uso de YAML por parte de Ansible y SaltStack vuelve más legible para el ser humano la codificación de las tareas y facilita la administración de estas. Puppet utiliza una codificación más similar a la programación, Puppet DSL (Domain-Specific Languaje)[28].

Ansible es la única de las mencionadas que no requiere de agente. Las otras opciones requieren que la maquina objetivo disponga del respectivo agente instalado para comunicarse con el nodo de control, mientras que Ansible solo necesita que la máquina se accesible mediante SSH para establecer la comunicación entre nodo de control y máquina cliente.

Debido su enfoque distinto, todas las opciones tienen unos usos o escenarios donde se podrían considerar más adecuadas. Gracias a su versatilidad Puppet es una mejor opción para entornos estables y gestión a largo plazo, más orientado a la labor de administración de sistemas.

Ansible, por el contrario, es mucho más idóneo para despliegues rápidos, periódicos y de tamaño moderado gracias a su simplicidad en la programación y control del flujo de trabajo.

SaltStack es una opción muy similar a Ansible, pero presenta una mejor integración en entornos orientados a sistemas Windows o integración con el Cloud.

Ansible es más idóneo para entornos virtualizados y/o principalmente basados en Linux, aunque puede funcionar con Windows también. Sin embargo, Ansible presenta un catálogo de módulos muy amplio para simplificar la ejecución de tareas y puede llegar a ser mejor opción para empresas con poca experiencia en automatización y/o programación.

5.3. Zabbix

Zabbix es una herramienta de monitorización para la monitorización de sistemas. Zabbix permite recoger métricas, almacenarlas y generar alarmas para estados anómalos o críticos de los valores recogidos.

Los sistemas que se pueden monitorizar no se limitan a máquinas virtuales, sino que se pueden monitorizar switches, firewalls, servidores o cabinas de almacenamiento.

Para recopilar los datos el método más común es mediante el protocolo SNMP (Simple Network Management Protocol). La información, parámetros u opciones de los sistemas tienen un identificador que se pueden utilizar para preguntar o editar. En el caso de Zabbix se realizan consultas de lectura para extraer los datos de las métricas que se deseen.

Para los sistemas Linux y Windows se dispone de un agente utilizado para extraer una cantidad mayor de métricas, como estado de los servicios o lectura de archivos, por ejemplo.

5.3.1. Items

Los ítems son las métricas de datos que se recogen. Un host necesita tener ítems para todos los aspectos que se quieran monitorizar. Estos valores se pueden obtener mediante snmp, agente instalado, comprobaciones externas u otros protocolos.

Por ejemplo, se puede utilizar snmp para comprobar el estado de las interfaces de un dispositivo para revisar que se encuentra levantada. Zabbix cada X tiempo realizará consultas para recoger el estado y tener valores actualizados del estado de las interfaces.

5.3.2. Triggers

Los Triggers son expresiones que sirven para generar alertas ante ciertas condiciones. Utilizando los ítems se genera una expresión lógica para comparar el valor recogido con un umbral.

En el ejemplo anterior, se puede crear un Trigger para generar una alerta cuando el estado de la interfaz sea Down. De forma que cuando la expresión se cumpla se genere una alerta que avise de que puede haber un problema con esa interfaz.

Como no todos los problemas son igual de graves, los Triggers tienen asociada una gravedad que va desde Informational hasta Disaster. Esto permite clasificar las alertas en función de su gravedad e impacto en el sistema o actividad de la organización.

Los Triggers también disponen de otras opciones como la posibilidad de utilizar expresión para la recuperación. De la misma forma que una interfaz caída genera una alerta, se puede configurar que cuando la interfaz vuelva a estar operativa se solucione la alerta y no aparezca en el dashboard.

5.3.3. Proxy

Si Zabbix tiene que acceder a multiples infraestructuras en diferentes entornos o localizaciones no es practico que la aplicación acceda directamente a todos los entornos. Por lo que para eso existen los proxies.

Un proxy es una máquina que se despliega en una infraestructura y se encarga de recoger los datos de los dispositivos configurados a su alcance. Cuando el proxy recoge los datos los envía al servidor principal.

La monitorización es mayoritariamente pasiva desde el dispositivo, debido a que no es el dispositivo monitorizado quien inicia la comunicación para transmitir los datos. Es el proxy de Zabbix quien solicita los datos y el dispositivo responde.

Al configurar un host en Zabbix se especifica que proxy es el encargado de recolectar los datos y el servidor principal se comunica con el proxy para informarle del nuevo host y que datos debe recoger.

5.3.4. Templates

Los templates permiten crear, como su nombre indica, plantillas para definir una estructura de métricas y alertas que aplicar a un host.

En nuestro caso el objetivo es agrupar todas las métricas necesarias para cumplir con la baseline en una template.

Las templates se pueden anidar e incluir otras templates en su interior. Zabbix dispone de un conjunto bastante extenso de templates.

Para nuestro caso se ha creado una template nueva y anidado varias templates sencillas para comprobar el estado superficial del estado de la máquina; como el acceso por ICMP ping, estado y accesibilidad de SNMP, o estado del servicio del agente de Zabbix.

5.4. Alternativas a Zabbix

El ámbito de la monitorización es extenso y se puede extender a la mayoría de los ámbitos de los sistemas. Zabbix tiene un enfoque más orientado hacia la monitorización los recursos y estado de los sistemas. En ese ámbito la principal alternativa es Nagios[29].

En el ámbito de la monitorización de la seguridad, Tenable Security Center[30] y Nexpose[31] son las más conocidas. Ambas son similares y orientadas a la monitorización de seguridad.

Tenable Security Center (Tenable SC) permite el descubrimiento de sistemas dentro de la infraestructura y realizar escaneos de se seguridad periódicos para mantener un control de las vulnerabilidades a la que se exponen los sistemas.
Zabbix en este aspecto se queda en un nivel superficial donde las comprobaciones son manuales o sobre aspectos limitados. Tenable SC permite la verificación del estado de las máquinas y compararlo con bases de datos de vulnerabilidades para identificar que vulnerabilidades afectan a esa máquina.

6. Implementación

6.1. Automatización

La parte de la automatización es la parte que agrupa más peso debido a que es la parte que presenta un mayor margen de mejora. Zabbix tiene unas funciones algo limitadas a la hora de monitorizar campos específicos como la seguridad.

Ansible es un sistema muy versátil y que nos permite actuar de forma más directa sobre los controles de la auditoría de SCAP, ya que esta se centra en la configuración, no en la administración.

El playbook desarrollado permite verificar y corregir los puntos que seleccionamos anteriormente para la baseline. A continuación, se desglosan los diferentes puntos y las tareas creadas para asegurar el correcto estado.

6.1.1. Configurar AIDE

AIDE (Advanced Intrusion Detection Environment) es una herramienta de verificación de integridad de directorios y archivos. AIDE crea una base de datos con los checksums de los principales archivos de configuración del sistema para detectar ediciones, adiciones o eliminaciones de archivos del sistema.

Un checksum se obtiene de la aplicación de algoritmos criptográficos de hash para obtener una cadena de carácter fijo en base al contenido de un archivo. Como el hash se obtiene de la aplicación de los algoritmos sobre el contenido, un cambio en el contenido puede generar un hash, o checksum, completamente diferente.

AIDE utiliza estas funciones para generar checksums de todos los archivos y los almacena en una base de datos. Cuando se realiza una verificación, AIDE vuelve a calcular el checksum del archivo y lo compara con el que tiene almacenado para ver si es el mismo. Si coinciden significa que ese archivo está intacto.

AIDE no evita la intrusión ni permite remediarla de ninguna forma, pero permite tener un control sobre la modificación de archivos para detectar patrones maliciosos y llegar a detectar ataques.

La tarea en Ansible sigue los pasos necesarios para instalar AIDE y configurar revisiones periódicas.

Primero es necesario instalar el paquete de AIDE. El siguiente paso es inicializar la base de datos con la herramienta aideinit incluida en el paquete instalado anteriormente para generar una base de datos (aide.db.new.gz).

AIDE utiliza por defecto el path */var/lib/aide/aide.db* como archivo de base de datos, por lo que el siguiente paso es copiar y renombrar el archivo generado en el paso anterior hasta el directorio mencionado al principio de este paso.

Con estos pasos ya se tendría una instalación funcional de AIDE, pero lo recomendable es configurar la ejecución periódica de verificaciones. Por lo que utilizando cron se pude automatizar la ejecución del comando `aide.wrapper --check` para ejecutarse como mínimo una vez al día.

Este proceso no es recomendable realizarlo con mucha frecuencia porque en función de la cantidad de datos y especificaciones del sistema puede tardar cierto tiempo y afectar al rendimiento. Pero una o dos veces al día en horarios de baja actividad es suficiente para mantener un registro actualizado de la integridad de los archivos.

El resultado de estas verificaciones podría monitorizarse en Zabbix, pero AIDE es un proceso muy extenso y relativamente lento, limitando la frecuencia de las verificaciones. En su lugar, como veremos posteriormente, se pueden calcular checksums desde el propio Zabbix y solo para los archivos que sea necesario.

6.1.2. Política de passwords

Uno de los puntos más importantes a la hora de mantener la seguridad de un sistema es el correcto uso de las contraseñas, sus requisitos y su duración. Por eso es importante que las contraseñas se renueven y cumplan unos requisitos de complejidad para asegurar que las contraseñas no son fáciles de adivinar.

Para ello se puede utilizar el PAM (Pluggable Authentication Modules). Estos módulos permiten editar el comportamiento de la autenticación de una forma relativamente sencilla.

El módulo `pwquality` permite establecer los requisitos de complejidad de las contraseñas. Con el módulo `pam_pwhistory` se puede limitar la reutilización de contraseñas y asegurar que no se utilicen las mismas contraseñas. El módulo `pam_tally2` nos permitirá configurar el bloqueo temporal de los usuarios en caso de fallo de autenticación.

Para la renovación de la contraseña se puede editar el archivo */etc/login.defs* para modificar los parámetros de edad mínima, máxima e inactividad de una contraseña. En nuestro caso se ha decidido poner un máximo de 3 meses y un periodo máximo de inactividad de 30 días.

6.1.3. Configuración de sudo

A la hora de securizar el uso de comandos con sudo es importante que la actividad se registre y no permita más acciones de las que se deben realizar.

Para lo primero se debe configurar la opción en la configuración de sudo para guardar los logs. Estos se pueden guardar en local directamente o enviarse a un servidor syslog.

En este caso Ubuntu ya incluye el servicio de rsyslog y las tareas se encargan de añadir el parámetro de syslog como local en la configuración de sudo (*/etc/sudoers*) y añadir la entrada correspondiente en la configuración de rsyslog.

```yaml
# Verify if log destination is defined in the file and set the correct value
- name: Check if logfile is configured properly in /etc/sudoers
  lineinfile:
    path: /etc/sudoers
    regexp: ^[\s]*Defaults\s(.*)\blogfile=[-]?.+\b(.*)$
    line: Defaults \1syslog=local1\2
    backrefs: true
  become: true
  register: sudoers_logfile_option

# If "Defaults syslog=..." isn't defined it is added in this step
- name: Configure logfile in /etc/sudoers
  lineinfile:
    path: /etc/sudoers
    line: Defaults syslog=local1
  become: true
  when: sudoers_logfile_option is defined and not sudoers_logfile_option.changed

# Ensure rsyslog puts all sudo logs in a dedicate logfile
- name: Configure rsyslog config files
  block:
    - name: Configure rsyslog logfile for sudo
      lineinfile:
        path: /etc/rsyslog.d/50-default.conf
        regexp: ^local1.\*\s*.+$
        line: local1.*  {{ sudo_logfile }}

    - name: Change rsyslog log format
      lineinfile:
        path: /etc/rsyslog.d/50-default.conf
        regexp: '^$ActionFileDefaultTemplate'
        line: '$ActionFileDefaultTemplate RSYSLOG_FileFormat'
  notify: "Restart rsyslog service"
```

Imagen 11: Configurar logs de sudo a través de rsyslog

Para la segunda parte, la opción de forzar el uso de pseudo-terminal limita el acceso al terminal real del sistema para impedir o dificultar que un atacante mantenga el control una vez la ejecución del proceso haya terminado.

Esta opción para el sudo se habilita utilizando el parámetro `Defaults use_pty` en el archivo de */etc/sudoers*.

```yaml
- name: Configure pseudo-terminal for sudo commands
  lineinfile:
    path: /etc/sudoers
    regexp: '^Defaults\s+use_pty'
    line: 'Defaults use_pty'
  become: true
```

Imagen 12: Configurar PTY para comandos con sudo

6.1.4. Configuración de sshd

El servicio de servidor SSH viene con una configuración por defecto que busca un balance entre seguridad y facilidad de uso, pero esto último implica realizar ciertas concesiones a la seguridad para un uso más cómodo en el día a día.

Sin embargo, podemos cambiar la configuración para limitar intentos, impedir acciones inseguras como prohibir el acceso mediante usuario root o reducir el abanico de cifrados solo aquellos que sean más robustos.

Para realizar las configuraciones de todos los parámetros de buenas prácticas del servicio SSH se puede abordar de diferentes formas.

La más manual sería la de ir comprobando uno a uno los parámetros con el módulo ansible.builtin.lineinfile. Pero es un enfoque demasiado lento y poco elegante.

Otra opción viable es crear un archivo .conf con todos los parámetros y dejarlo en la carpeta /etc/*sshd_conf.d*. Este archivo se puede generar con una plantilla de jinja2,

Pero si se quiere que los parámetros sean valores por defecto es importante que estos parámetros sean los últimos en incluirse, ya que en caso de que haya varias asignaciones distribuidas entre los archivos, el servicio se quedará con el primer valor que encuentre.

Como esto último es dependiente de los diferentes archivos, es más viable combinar ambos enfoques para editar directamente el archivo de configuración principal. Para ello se ha combinado el uso de templates jinja2 con el módulo ansible.builtin.blockinfile. Este módulo funciona de manera muy similar al módulo lineinfile pero permitiendo la adición y modificación de bloques enteros de texto.

La template utilizada (*templates/sshd-config.j2*) incluye todos los parámetros a configurar y las respectivas asignaciones en función de las variables definidas para el rol (defaults/main.yml). Al hacer un lookup podemos decir a Ansible que “compile” el archivo y lo utilice como contenido para el módulo blockinfile.

El módulo blockinfile se asegura que todos estos parámetros sean insertados después de la inclusión de los archivos personalizados *(/etc/sshd_config.d/*.conf*).

```
- name: Edit config file
  blockinfile:
    block: "{{ lookup('template','sshd-config.j2') }}"
    dest: '/etc/ssh/sshd_config'
    insertafter: '^Include \/etc\/ssh\/sshd_config\.d\/\*\.conf'
    marker: "# {mark} ANSIBLE SECURE CONFIGURATION BLOCK"
    validate: /usr/sbin/sshd -T -f %s
```

Imagen 13: Configuración de parametros para SSH

6.1.5. Instalar y configurar el servicio NTP

El protocolo NTP (Network Time Protocol) permite que los sistemas utilicen un servidor común como fuente de la hora del sistema. Los relojes software o hardware de cada sistema no son perfectos y tener desviaciones que provoquen una desincronización.

Por lo que, por ejemplo, los logs registrados en varios sistemas al mismo tiempo pueden registrarse con marcas de tiempo (timestamps) dieferentes o en el orden incorrecto. Si alguien cruza una puerta y el reloj de esa habitación marca las 10:00 y lo registra; y cuando cruza a la siguiente el reloj de esta otra sala marca las 9:55 y lo registra. Por lo que a la hora de ver ambos registros puede malinterpretar el orden en el que la persona ha cruzado las habitaciones.

Por eso se establece un servidor NTP, que se encarga de establecer la hora que todos los sitemas deberán adoptar. Los sistemas periódicamente preguntan la hora al servidor y en base a la respuesta ajustan su hora local para mantenerla sincronizada.

Para instalarlo solo es necesario verificar que el paquete “ntp” se encuentre instalado, eliminar de la configuración los servidores por defecto y substituirlos por los utilizados en la organización. Estos servidores pueden ser internos o externos, pero es importante que sean los mismos que los configurados en el resto de sistemas por el motivo comentado anteriormente.

```yaml
- name: Install NTP
  apt:
    name: ntp
    state: present

- block:
  - name: Remove defualt NTP servers
    lineinfile:
      path: /etc/ntp.conf
      regexp: '^pool.*'
      state: absent
  - name: Configure NTP servers
    lineinfile:
      path: /etc/ntp.conf
      regexp: '^server {{item}}'
      line: server {{item}}
    loop: "{{ ntp_servers }}"
  notify: "Restart NTP service"
```

Imagen 14: Tareas para instalar y configurar el cliente NTP

6.1.6. Configuración de permisos

A la hora de mantener un sistema seguro, dentro de lo razonable, es importante asegurarse de que los archivos más críticos tengan los permisos mínimos para funcionar.

Por eso es importante que archivos como */etc/shadow*, que contiene los hashes de las contraseñas de usuario, solo sean visibles por el usuario root.

Hay otros archivos como */etc/passwd*, que contiene el listado de usuarios, donde el usuario root debe ser el único en poder editar el archivo, pero el resto de los usuarios podrían tener acceso de solo lectura.

El libre acceso a alguno de los archivos anteriores permitiría a un usuario malintencionado editar esos archivos para cambiar propiedades de los usuarios sin pasar por otro tipo de controles y poder escalar privilegios a un usuario, por ejemplo.

Entrando en los permisos a nivel de usuario es necesario asegurar la correcta mascara. La umask (user file-creation mode mask) determina los permisos predeterminados para los nuevos archivos creados por un usuario. Para evitar que los usuarios o ciertas shells (como C o Bash) creen archivos con permisos demasiado amplios se deben modificar los respectivos archivos de configuración para especificar una umask adecuada.

```yaml
- name: Set Umask permissions
  block:
  - lineinfile:
      path: /etc/{{ item }}
      regexp: '(?i)^UMASK'
      line: 'UMASK 027'
      create: true
    loop:
      - 'csh.cshrc'
      - 'login.defs'

  - lineinfile:
      path: /etc/{{ item }}
      regexp: '(?i)^umask'
      line: 'umask 027'
      create: true
    loop:
      - 'bashrc'
      - 'bash.bashrc'
      - 'profile'
```

Imagen 15: Tareas para la asignación de permisos de umask

De esta forma no solo se abordan los permisos actuales, sino los permisos de los archivos creados en el futuro.

Otro punto relacionado con los usuarios es asegurar que el directorio home de cada usuario tiene unos permisos adecuados, como 750 (rwxrw----). Permitiendo control total solo al usuario en cuestión y solo lectura-escritura a los pertenecientes al mismo grupo.

```yaml
- name: Set Home directory permissions
  block:
    - name: find home directories
      find:
        path: /home/
        file_type: directory
        get_checksum: true
      register: home_directories
    - name: Set permissions for every home directory
      file:
        path: '{{ item }}'
        mode: '0750'
        recurse: true
      loop: "{{ home_directories.files | groupby('checksum') | map(attribute='1.0.path') | list }}"
```

Imagen 16: Tarea para configurar los permisos de las carpetas de usuario

6.1.7. Configuración segura del GRUB

El grub es un gestor de arranque de Linux y es el encargado de arrancar el sistema o elegir que sistema ejecutar en el momento del arranque de una máquina. El GRUB es muy importante ya que permite editar opciones del arranque del sistema.

Por lo que un usuario malintencionado puede llegar a editar la secuencia de arranque y alterar el comportamiento del sistema para escalar privilegios o realizar otro tipo de acciones de forma ilegitima.

Por eso es importante limitar los permisos sobre los archivos de configuración del GRUB y sobre que usuarios pueden realizar estas ediciones.

Para el primer punto se puede utilizar el mismo modulo utilizado en la Imagen 16 para asegurar que los archivos de configuración no se puedan modificar tan fácilmente.

Para el segundo punto la solución es crear un usuario “boot” con permisos limitados dentro del sistema y que será el único autorizado a realizar cambios en el arranque. Lo primero es crear el usuario en el sistemas con una contraseña que generaremos en la propia ejecución y que luego será mostrada por consola al ejecutar el playbook. El usuario boot no tiene una terminal asignada y su contraseña no expira.

```yaml
# Generate a password for grub user
- name: Generate grub password
  set_fact:
    grub_password: "{{ lookup('password','/dev/null') }}"
  no_log: true

# Create boot user in the system
- name: Create boot user
  block:
    - name: Create user
      user:
        name: boot
        group: root
        password_expire_max: 99999
        create_home: false
        shell: '/sbin/nologin'

    - name: Change user password
      shell: 'echo boot:{{ grub_password }} | chpasswd'
      no_log: true
```

Imagen 17: Creación del usuario boot

El siguiente paso es configurar los archivos del GRUB con el usuario autorizado, la contraseña en formato pbkdf2 y restringir el acceso por defecto. Estas tareas se pueden observar en la Imagen 18. Con esto se podría recargar al configuración del GRUB y en el arranque se debería poder arrancar sin problemas, pero a la hora de de editar el arranque sería necesario autenticarse con el usuario boot para poder editar el arranque.

```yaml
- name: Configure grub files
  block:
    # Set superuser boot and password
    - name: Add password to config in /etc/grub.d/40_custom
      block:
        - name: Configure grub user
          lineinfile:
            path: /etc/grub.d/40_custom
            regexp: '^set superusers='
            line: 'set superusers="boot"'
          become: true
        - name: Configure grub password
          lineinfile:
            path: /etc/grub.d/40_custom
            regexp: '^password_pbkdf2 boot '
            line: "password_pbkdf2 boot {{ grub_pbkdf2_password.stdout | regex_search('(grub\\.pbkdf2\\..*)','\\1') | first }}"
          become: true
      when: custom_config_file.stat is defined and custom_config_file.stat.exists
    # Set --unrestricted on boot options to only protect the edit option
    - name: Configure /etc/grub.d/10_Linux
      replace:
        path: /etc/grub.d/10_linux
        regexp: (echo "menuentry '\$\(.*\)'\s+)(?:--unrestricted├──users\s*.*)?\s*(\${CLASS}.*)
        replace: \1 --unrestricted \2
  notify: "Reload grub options"
when:
    - '"/boot/efi" not in ansible_mounts | map(attribute="mount") | list'
    - '"grub2-common" in ansible_facts.packages'
```

Imagen 18: Tareas para configurar los archivos del GRUB

6.1.8. Configuración del firewall

Un firewall a nivel de host es útil para aportar una última línea de defensa perimetral. Lo recomendable es limitar los accesos a los servidores a solo aquellos servicios necesarios.

Por lo que para el caso básico de un servidor lo ideal sería solo permitir las conexiones por el puerto 22 para SSH. En nuestro caso necesitaremos los puertos 161 y 10050 para la monitorización por SNMP y el Agente de Zabbix respectivamente.

Ansible tiene un módulo para realizar las configuraciones para el firewall ufw, lo que permite una configuración más sencilla que utilizar los comandos para añadir las reglas.

```
- name: Configure ufw rules
  block:
    - name: Allow ssh
      ufw:
        rule: allow
        port: ssh
        proto: tcp
        src: 0.0.0.0/0 # Everyone
        log: true
    - name: Allow snmp
      ufw:
        rule: allow
        port: snmp
        proto: udp
        src: 0.0.0.0/0
    - name: Allow 10050/tcp (Zabbix-Agent)
      ufw:
        rule: allow
        port: 10050
        proto: tcp
        src: "{{ proxy_ip }}"
  notify: "Reload ufw"
```

Imagen 19: Creación de reglas para el firewall de Ubuntu

6.1.9. Eliminar Telnet y otros servicios innecesarios

Telnet es considerado un protocolo de comunicación considerado inseguro por su falta de encriptación en la transmisión de datos e incluso las credenciales de acceso. Protocolos como SSH ofrecen un nivel de seguridad mucho mayor y debería optarse por este último.

Por lo que para evitar el uso de telnet por parte de software o usuarios malintencionados se debería eliminar la herramienta para establecer este tipo de conexiones.

Por otra parte, existen recomendaciones pare prescindir de ciertos servicios y aplicaciones que suelen incluirse en configuraciones típicas y no siempre son necesarias. Siempre es una buena práctica reducir la superficie de ataque y sin un software no es necesario, aunque no sea inseguro, es mejor quitarlo.

Estos pueden ser el caso de herramientas para ejercer de servidor DNS, servidor DHCP, servidor HTTP o herramientas como clientes LDAPS, SNMP o CIFS. Si cualquiera de estas funciones fuera necesaria, siempre se puede obviar esta recomendación.

6.1.10. Configuración de parámetros de red

La versatilidad del kernel de Linux permite realizar una gran variedad de acciones, no todas tienen porque ser necesarias. En el caso de un servidor sin rol de router no se necesitan todas las opciones que el kernel permite hacer para dicha labor, por lo que se deberían deshabilitar.

A través de sysctl, y sus archivos de configuración, podemos especificar que funciones se van a deshabilitar o habilitar. En nuestro caso se quiere deshabilitar todas las acciones con reenvío de paquetes o comunicación para el enrutamiento.

Por otra parte, hay opciones que independientemente del rol del servidor es conveniente habilitar; como registrar logs de paquetes erróneos, aceptar por defecto los paquetes de ICMP seguro, o usar TCP syncookies para evitar ataques de TCP SYN Flood.

Todo esto se puede realizar creando un archivo .conf con todos estos parámetros y colocarlo en */etc/sysctl.d/*. A diferencia de lo especificado previamente con SSH, en este caso la máxima prioridad la tiene el último valor recogido.

Colocando el archivo de forma que alfabéticamente sea el último nos asegura que, a la hora de recargar los valores, nuestro archivo se lea el último y sobrescriba cualquier otro valor.

```
# Disable packet Redirect
net.ipv6.conf.all.accept_redirects = 0
net.ipv6.conf.default.accept_redirects = 0
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.default.secure_redirects = 0
# Disable IPv6 Forwarding
net.ipv6.conf.all.forwarding = 0
# Disable Source-Routed packets
net.ipv6.conf.all.accept_source_route = 0
net.ipv6.conf.default.accept_source_route = 0
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0
# Disable IPv6 Router Advertisments
net.ipv6.conf.all.accept_ra = 0
net.ipv6.conf.default.accept_ra = 0
# Log IPv4 martian packets (impossible addresses or spoofed packets)
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.default.log_martians = 1
# Enable IPv4 Reverse Path Filtering
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1
# Ignore IPv4 ICMP Broadcast Echo Requests
net.ipv4.icmp_echo_ignore_broadcasts = 1
# Ignore IPv4 Bogus ICMP Error messages
net.ipv4.icmp_ignore_bogus_error_responses = 1
# Enable IPv4 Syncookies
net.ipv4.tcp_syncookies = 1
# Disable IPv4 send ICMP Redirects
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0
# Disable IPv4 IP Forwarding
net.ipv4.ip_forward = 0
```

Imagen 20: Archivo con todos los parámetros a modificar en sysctl

6.1.11. Monitorización con Zabbix

Como Ansible nos permite una infinidad de posibles acciones, es posible que la propia preparación de ese host para ser monitorizado se realice automáticamente.

Para la monitorización serán necesarias dos tareas principales: Configurar SNMP e instalar el agente de Zabbix.

Para conseguir la monitorización por SNMP el paquete snmpd debe estar instalado y se debe editar el archivo de configuración. SNMP tiene varias versiones, las versiones 1 y 2 funcionan mediante community strings, mientras que la versión 3 funciona mediante usuarios.

En este entorno la monitorización se realiza mediante SNMPv2 y ya se dispone de una community string de solo lectura. Por lo que para que ese host pueda ser accedido es necesario configurar la community string en el archivo de configuración. En ese mismo parámetro se puede limitar que hosts o subredes tienen acceso a esa community, por lo que en nuestro caso se limitará el acceso a solo la IP del proxy, que es quien realizará las consultas.

Después de reiniciar el servicio ya tendremos los cambios aplicados y podremos monitorizar por SNMP el host.

Entrando en el agente, lo primero es instalarlo. Para ello tendremos que añadir el repositorio correcto para la versión de Zabbix deseada y distribución de Linux presente. Con ello podremos descargar el agente y tendremos que configurar el archivo de configuración, en */etc/zabbix/zabbix_agent2.conf*.

En dicho archivo se debe configurar la IP del proxy en los parámetros de Server y ServerActive, y el parámetro de hostname para utilizar el obtenido por el propio Zabbix. Los 2 últimos parámetros son necesarios para la monitorización activa, utilizada para monitorizar los logs.

Adicionalmente, si se pretende extraer información de ciertos archivos de log, es necesario que el usuario del servicio de Zabbix tenga permisos de lectura sobre dichos archivos. Con ACL (Access Control List) se puede regular el acceso de forma más precisa que con herramientas de chmod y chown.

```yaml
- name: Set access permissions for zabbix user
  block:
    - name: Set access to sudo.log
      acl:
        path: "{{sudo_logfile}}"
        entity: zabbix
        etype: user
        permissions: rx
        state: present
    - name: Set access to auth.log
      acl:
        path: "/var/log/auth.log"
        entity: zabbix
        etype: user
        permissions: rx
        state: present
  notify: "Restart Zabbix-Agent2 service"
```

Imagen 21: Otorgar permisos un usuario mediante ACLs

Como posteriormente el agente de Zabbix va a necesitar de un script para descubrir los archivos de configuración, hay que asegurar que el script se copie a la máquina y se realice la configuración pertinente en el archivo de configuración del agente.

```
- name: Copy script for file lld
  block:
    - name: Ensure path exists
      file:
        path: /usr/lib/{{ item }}
        owner: zabbix
        group: zabbix
        state: directory
      loop:
        - 'zabbix'
        - 'zabbix/externalscripts'

    - name: Copy script
      copy:
        src: conf_file_lld.py
        dest: /usr/lib/zabbix/externalscripts/conf_file_lld.py
        owner: zabbix
        group: zabbix
        mode: 0750
```

Imagen 22: Tarea para la copia del script para la discovery rule

```
- name: Set User Parameter for file discovery rule
  lineinfile:
    path: /etc/zabbix/zabbix_agent2.conf
    regexp: '^UserParameter=file.conf.discovery'
    line: 'UserParameter=file.conf.discovery,/usr/lib/zabbix/externalscripts/conf_file_lld.py'
    insertafter: '^# UserParameter.*$'
```

Imagen 23: configuración de la discovery en el archivo de configuración de Zabbix

6.2. Monitorización

La monitorización es la segunda parte de este trabajo, y es la que permitirá mantener un control de ciertos aspectos de seguridad una vez realizada la securización.

6.2.1. Eventos de sudo

Los permisos de super-usuario se deben vigilar para evitar un abuso de estos debido a su elevado potencial de disrupción y/o alteración de un sistema. Por lo que, partiendo de los logs que ya se están generando gracias a la automatización previa, se pueden recoger con Zabbix y generar alertas al respecto.

El agente de Zabbix nos da acceso a poder leer logs con su respectivo item key. Esto permite que el item recoja cada línea del archivo de log como un valor recogido por el item.

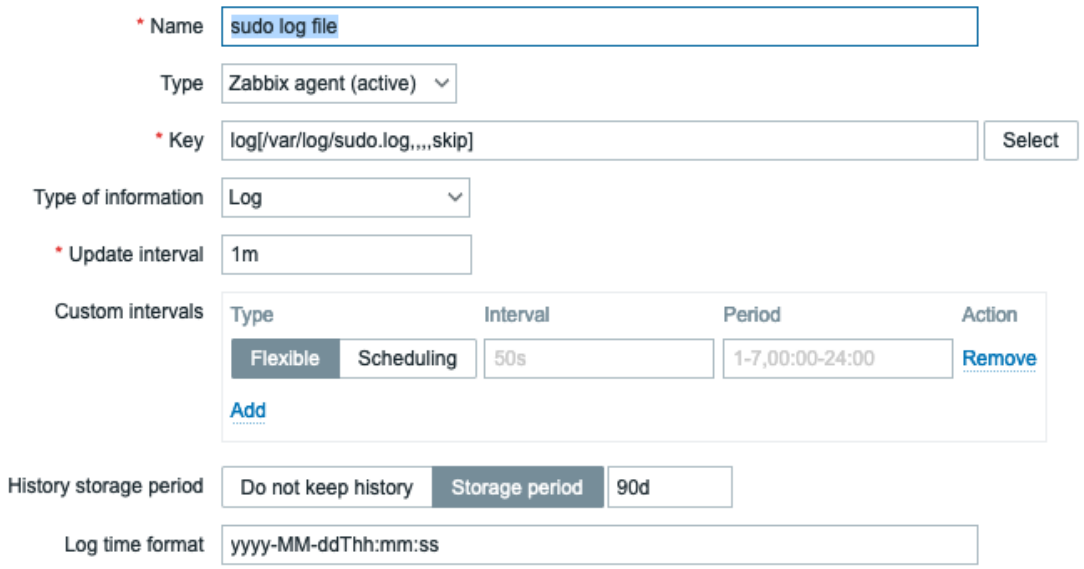


Imagen 24: Configuración del item para recoger logs de */var/log/sudo.log*

Con este item podemos generar ítems dependientes para filtrar solo un tipo de log en concreto, o crear directamente un trigger para generar una alerta.

Zabbix otorga la opción de realizar acciones de preprocesado para los valores recogidos. Estas acciones permiten editar valores, validar patrones o conversiones de tipo. Para el caso de los logs se puede hacer uso de expresiones regulares para recoger solo un tipo de eventos deseados o filtrar aquellos que no se deban tener en cuenta.

Para el caso de sudo, se ha querido monitorizar cuando un comando sea ejecutado con estos privilegios, cuando un usuario falle en la autenticación al ejecutar el comando o cuando un usuario sin permisos (no registrado en el grupo de sudoers) intente ejecutar un comando, aunque la autenticación es correcta.

La expresión del trigger utilizar una expresión regular para ir comprobando si los logs que se van recogiendo coinciden con el mensaje que se esperaría de este tipo de problemas.

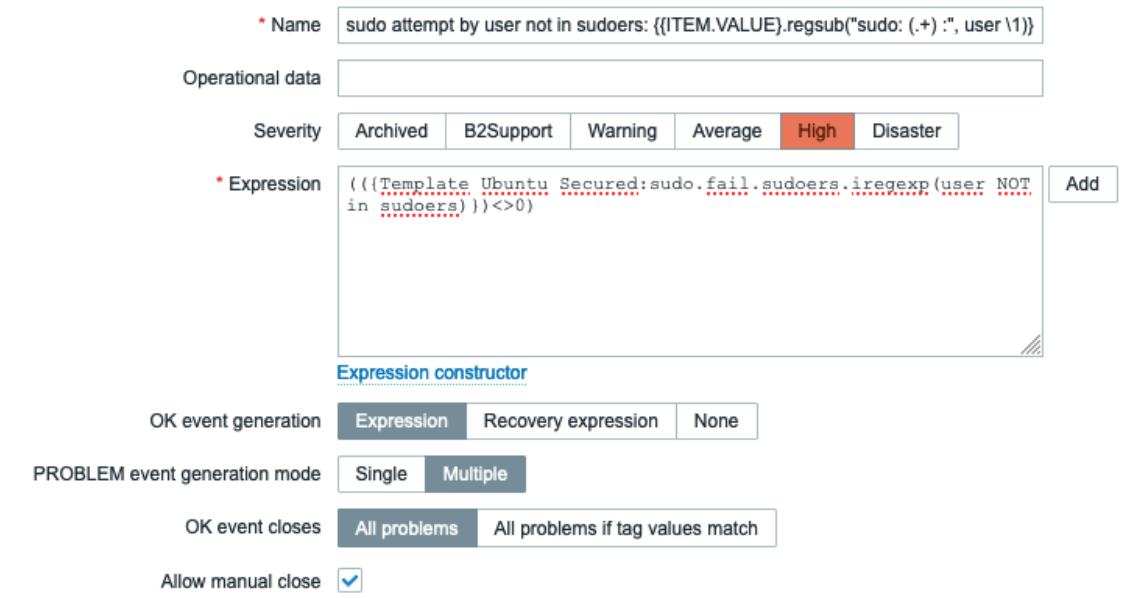


Imagen 25: Trigger para una alerta de intento de ejecución sin permisos

Zabbix tiene una serie de macros para extraer directamente información interna, como nombres, configuraciones o valores. Estas macros pueden ser útiles para personalizar los mensajes de alerta, etiquetas o como parámetros para scripts o algunos item keys.

En el trigger creado en la Imagen 25 se observa cómo se referencia al item para extraer el valor del item y poder introducir información del evento en el propio título de la alerta. En la Imagen 26 se puede ver como el título de la alerta ha introducido el nombre del usuario y el comando que ha intentado ejecutar.

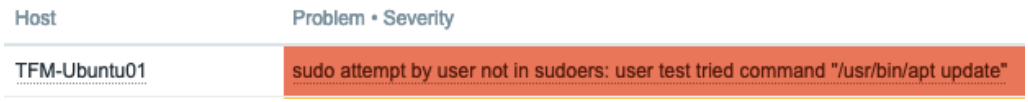


Imagen 26: Alerta generada por intento de ejecución de comando sin permisos

Cuando un usuario intenta ejecutar un comando sin estar en el grupo de sudoers, como en la Imagen 27, se generará una entrada de log en el archivo */var/log/sudo.log*, que será recogido por el agente y generará una alerta como la que se puede apreciar en la Imagen 26.

```
root@tfm-ubuntu01:/home/ssierrago# su test
$ sudo apt update
[sudo] password for test:
test is not in the sudoers file.  This incident will be reported.
$
```

Imagen 27: Intento de ejecución de comando por parte de usuario sin permisos

6.2.2. Eventos de SSH

Utilizando el mismo método que en el punto anterior se pueden extraer eventos relacionados con los accesos a la máquina mediante SSH. De esta forma se pueden crear alertas para los eventos que se considere por parte de la organización.

Para este caso se ha creado una alerta para avisar cuando un usuario cometa un fallo de autenticación (al introducir incorrectamente la contraseña). Como se puede ver en la Imagen 29, se ha generado una alerta en base al intento de acceso fallido que se puede ver en la Imagen 28. En la alerta se ha registrado el intento, el usuario utilizado y la IP de origen.

```
ssierrago@tfm-ansible-server:~$ ssh test@10.1.11.6
test@10.1.11.6's password:
Permission denied, please try again.
test@10.1.11.6's password:
Permission denied, please try again.
test@10.1.11.6's password:
Received disconnect from 10.1.11.6 port 22:2: Too many authentication failures
Disconnected from 10.1.11.6 port 22
```

Imagen 28: Intento fallido de acceso SSH

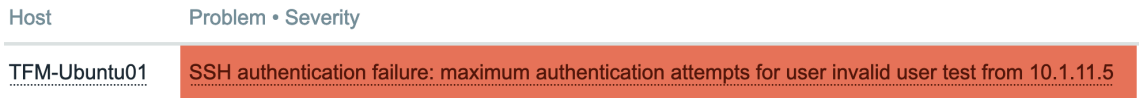


Imagen 29: Alerta generada en el dashboard por intento fallido de acceso SSH

6.2.3. Estado de los servicios

Una de las opciones que permite el agente de Zabbix es la monitorización del estado de los servicios. Esto es una opción relativamente simple que nos permita crear una regla de descubrimiento para encontrar los servicios y crear alertas para dichos servicios.

Esta Discovery rule puede ser algo confusa, ya que por defecto descubre todos los servicios. Esto puede llevar a que se generen una gran cantidad de ítems, de los que no todos tienen porque ser de utilidad.

Por lo que se ha realizado un filtrado mediante la opción de filtros para la discovery rule y solo crear métricas para los servicios de interés. El filtro utilizado incluye aquellos servicios críticos del sistema (como systemd o cron) y algunos de interés para la configuración realizada previamente (como ntp o sshd).

De esta forma se puede tener control de los servicios para comprobar un correcto funcionamiento de estos. Esto incluye también a aquellos que intervienen en la seguridad, como por ejemplo ufw, cron o Rsyslog.

6.2.4. Integridad de archivos relevantes

Anteriormente se había mencionado la importancia de mantener la integridad de ciertos archivos críticos para el sistema o con información sensible. Por eso se puede verificar la integridad de dichos archivos periódicamente y generar alertas en caso de modificación.

Mediante el agente de Zabbix, se puede crear un ítem que calcule el checksum del archivo. En caso de una modificación del archivo, el checksum generado en la siguiente comprobación será diferente y se generará una alerta.

Esto puede ocasionar falsos positivos debido a que, en caso de cambios legítimos, como una actualización o la creación de un usuario, el checksum cambiará y generará una alerta.

Otro problema es que esta alerta no otorga información sobre que se ha modificado en el archivo, solo el hecho de haber sido modificado. Pero de todas formas sirve como un aviso de modificación de un archivo importante que convendría revisar si ha sido legitimo o malicioso.

Para llevar a cabo esta monitorización, los ítems se pueden crear manualmente para cada archivo que se quiera monitorizar o hacer uso de una LLD (Low-Level Discovery). Zabbix por defecto dispone de una serie de reglas de descubrimiento predeterminadas, pero se pueden crear scripts personalizados.

El script debe devolver los valores en un formato determinado (lista de diccionarios) y se debe especificar en la configuración del agente la key y el script. Al reiniciar el agente se podrá crear una regla como la utilizada en el apartado anterior.

De la misma forma que no es necesario monitorizar todos los servicios, no es necesario monitorizar la integridad de todos los archivos, por lo que con el uso de filtros se puede filtrar que archivos se van a descubrir para la creación de los ítems y sus respectivos triggers.

Como se puede apreciar en la Imagen 30, cuando se añade una línea a un archivo, provoca un cambio en el hash md5 y provoca una alerta, como se puede apreciar en la Imagen 31

```
root@tfm-ubuntu01:/home/ssierrago# md5sum /etc/passwd
979b3cb30cbe77af0c88330ac5e9e907  /etc/passwd
root@tfm-ubuntu01:/home/ssierrago# echo "Not_a_real_user::0:0:suspicious user:/root:/bin/bash" >> /etc/passwd
root@tfm-ubuntu01:/home/ssierrago# md5sum /etc/passwd
aaf23aee5b38249b4255f08d587a8df8  /etc/passwd
root@tfm-ubuntu01:/home/ssierrago#
```

Imagen 30: Añadir una línea al archivo /etc/passwd

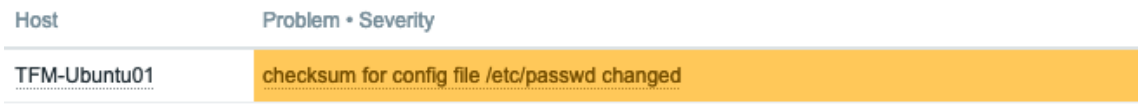


Imagen 31: Alerta de cambio de checksum en el archivo *`/etc/passwd`*

Para obtener más información se pueden crear ítems para extraer el contenido del archivo y comprobar las líneas que nos interese monitorizar. Esta solución sería el otro extremo y solo sería útil para parámetros muy específicos.

Al igual que en el apartado anterior, estas métricas se obtienen mediante una discovery rule para encontrar los archivos que se encuentren en */etc/*. Pero se han aplicado filtros para reducir el abanico de archivos descubiertos a aquellos que se necesite monitorizar.

7. Resultados

Los resultados obtenidos los podemos valorar de varias formas, ya que algunos puntos son cambios estáticos, como el bastionado de la configuración, y otros más dinámicos orientados de cara a futuro, como la monitorización.

7.1. Resultados de SCAP

Como para la valoración inicial de estos aspectos más estáticos su partió del protocolo SCAP, podemos volver a aplicarlo para valorar la mejora. Utilizando el mismo perfil podemos ver una mejora en el resultado, se ha incrementado de 101 a 168 reglas superadas. Lo que dejaría una puntuación final de 79,94% sobre el estándar del CIS.

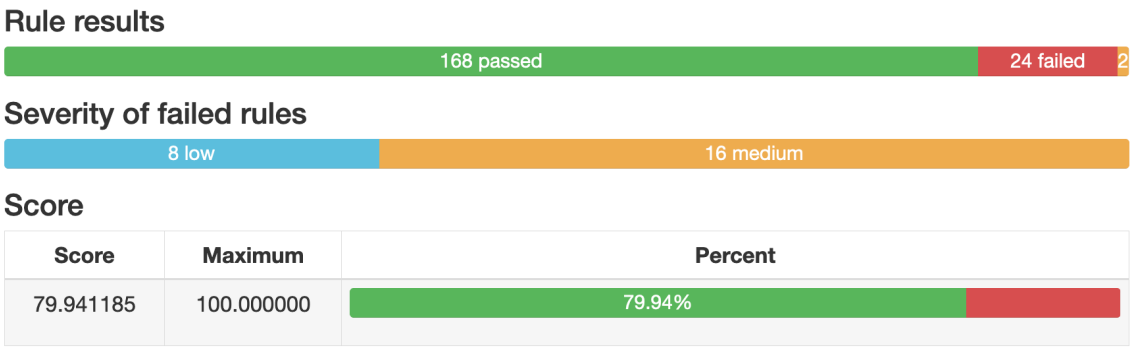


Imagen 32: Resultados de OpenSCAP con perfil CIS nivel 1 despues de aplicar el playbook

Para que el análisis sea más cercano a la baseline diseñada se ha creado un perfil personalizado quitando los puntos que se habían descartado en el análisis inicial. Con el uso de la GUI de OpenSCAP, OpenSCAP Workbench se puede editar una checklist y crear perfiles personalizados. Estos perfiles personalizados se pueden exportar como archivo de personalización (tailoring file).

Para utilizar un archivo de personzalización solo es necesario referenciar al nobre del perfil personalizado y utilizar el parámetro `--tailoring-file` para indicar la localización del archivo mencionado anteriormente.

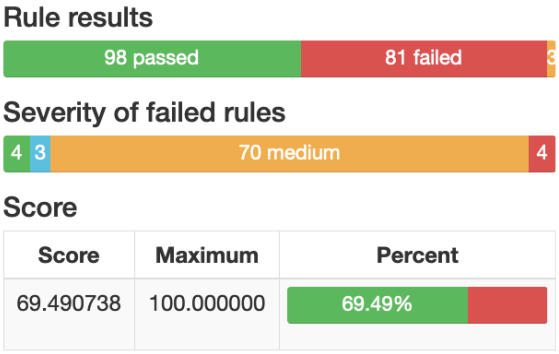


Imagen 33: Resultados OpenSCAP perfil personalizado antes de aplicar el playbook

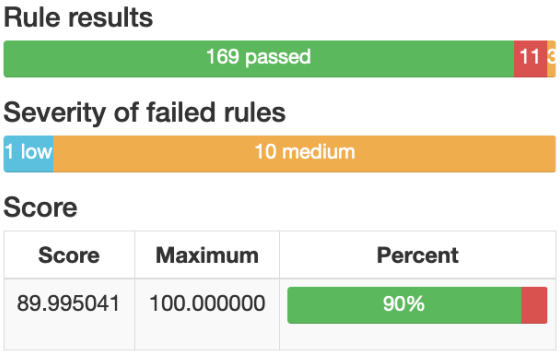


Imagen 34: Resultados OpenSCAP perfil personalizado después de aplicar el playbook

Como podemos ver en la Imagen 33, el informe antes de aplicar la automatización muestra una puntuación de 69,49%, habiendo fallado 81 reglas.

Al aplicar el playbook se han configurado los puntos mencionados en la baseline y podemos ver como el resultado ha crecido hasta un 90%, como se puede observar en la Imagen 34. Se han podido solucionar 4 reglas de prioridad alta, 60 de prioridad media y 7 de prioridad baja o indefinida.

Siguen quedando algunos puntos pendientes que pueden ser normales, como el límite de tiempo de los usuarios existentes, ya que el usuario boot utilizado para proteger el arranque no tiene límite de tiempo para la contraseña.

Pero se puede apreciar una mejora considerable en la seguridad global del sistema, habiendo configurado de forma más segura los elementos o servicios elegidos en la elaboración de la baseline.

Se podría extender el rol de Ansible para ir asegurando el estado de más puntos del estándar, e incluso extenderlo a otro nivel del estándar para terminar automatizar por completo el cumplimiento del estándar.

7.2. Resultados de la monitorización

En cuanto a lo monitorización, los resultados no son tan fácilmente cuantificables, pero se ha conseguido poder monitorizar algunos aspectos de la seguridad de los sistemas.

Pese a que Zabbix no es una herramienta diseñada específicamente para la monitorización de seguridad de sistemas, como Nessus o Nexpose, se ha podido implementar la monitorización de aspectos como la integridad de archivos, estado de servicios críticos, monitorización de eventos del sistemas o accesos al sistema.

Uno de los aspectos donde más se ha profundizado es en la monitorización de los diferentes archivos de log para generar métricas y alertas. Gracias a esta funcionalidad, se han podido generar alertas para fallos de autenticación o uso de comandos con elevación de privilegios.

Esta funcionalidad se muestra limitada en este proyecto, pero partiendo de la misma configuración, se pueden leer otros archivos de log y utilizar otras expresiones regulares para seleccionar eventos.

Se puede concluir que Zabbix no es la mejor opción para la monitorización, pero si ya se dispone de un sistema de monitorización o se pretende utilizar para uso general dentro de la organización se puede ampliar la monitorización a este ámbito de la seguridad.

8. Conclusiones y trabajos futuros

8.1. Conclusiones

Como se ha visto en los inicios de este proyecto, la seguridad por defecto de los sistemas Linux no es deficiente, pero presenta un claro margen de mejora.

La configuración y revisión de esta configuración se puede volver una tarea compleja y lenta. Por lo que es útil la existencia de herramientas para la automatización y administración de dicha configuración.

Como habíamos visto en el capítulo anterior, los resultados indican que se ha podido incrementar la seguridad del sistema de forma relativamente sencilla gracias a Ansible.

Con respecto a Zabbix, se ha visto que, pese a no ser una herramienta dedicada a la seguridad, se puede realizar una monitorización básica de la seguridad de los sistemas.

El resultado en la parte de Ansible lo consideraría satisfactorio, ya que gracias al playbook desarrollado se pueden desplegar máquinas Ubuntu con un mayor nivel de seguridad.

En el apartado de Zabbix creo que se podrían haber creado más métricas y alertas para contemplar algunos aspectos de los implementados en la automatización. Sin embargo, la creación de estas métricas y alertas se podría realizar de forma similar a los ítems y triggers creados para lo que se ha podido implementar.

8.2. Trabajos futuros

Durante este proyecto se habían barajado varias opciones para extender el alcance del proyecto, pero por las limitaciones de profundidad y tiempo hay ideas que se podrían explorar a partir de esta base.

Este concepto de securización y monitorización se podría extender vertical y horizontalmente. Verticalmente se podría extender a niveles de más alto nivel al propio sistema operativo, como servicios y aplicaciones. Se podrían crear roles para realizar configuraciones seguras de servicios como Apache, MySQL o Docker.

Horizontalmente se podría extender para monitorizar una mayor variedad de versiones, distribuciones y sistemas Operativos. Red Hat Enterprise Linux (RHEL) es la principal contraparte a Ubuntu en cuanto a servidores Linux en entornos productivos, y era una buena opción para un alcance más extensivo.

9. Glosario

Cloud – Modelo de infraestructura ofrecida a modo de servicio, prescindiendo de la necesidad de mantenimiento de una infraestructura física en un centro de datos propio de la oranización. Los recursos se alojan en las instalaciones del proveedor y este cede estos recursos para ser usados por la organización que contrata el servicio.

On-premise – Modelo de infraestructura tradicional donde el hardware y los servicios se alojan y se mantienen en las instalaciones de la organización.

Virtualización – Tecnología para la representación virtual de sistemas físicos. Sobre un mismo hardware se pueden alojar varias instancias virtuales. Las instancias virtuales imitan el comportamiento y funciones del hardware físico, pero todas funcionan sobre un único hardware físico compartido.

Máquina virtual – Instancia software que emula el funcionamiento de una máquina de hardware físico.

Hipervisor – Software para crear y ejecutar máquinas virtuales. Aunque las máquinas virtuales sean software, necesitan de unos recursos físicos por debajo que la hagan funcionar. El hipervisor se encarga de gestionar el acceso de las máquinas al hardware para que estas puedan funcionar.

Servidor (hardware) – Hardware sobre el que se se puede instalar un sistema operativo concreto o un hypervisor para alojar máquinas virtuales.

Servidor (rol) – Una máquina tiene un rol de servidor cuando su función es la de proporcionar un servicio. Puede ser un servidor web para mostrar una web, un servidor de archivos o un alojar una base de datos.

Servidor (servicio) – Un servicio de servidor es aquel que permite la creación de un acceso a otros dispositivos para ofrecer un determinado servicio o archivos. Por ejemplo, Apache es un servicio de servidor web.

Baseline – Una línea base es una línea imaginaria para hablar de un estándar o estado mínimo necesario.

Syslog – Protocolo para la estandarización de los mensajes de log de los sistemas. Permite dar un formato común a los logs y establecer los mecanismos de comunicación para facilitar el almacenamiento remoto y centralizado de los logs de una organización.

DNS – Systema de Resolución de Nombre (Domain Name System). Permite traducir nombres de dominio a sus respectivas direcciones IP. Funciona de forma similar a una guía telefónica, donde se encuentra una relación entre los nombres y los teléfonos para buscar el teléfono de cierta persona. Los DNS públicos permiten a los ordenadores consultar las IP de los sitios webs para poder acceder. Un DNS se puede aplicar a nivel interno para realizar esta labor, pero solo con los dispositivos de dentro de la organización.

Vulnerabilidad – Fallo, deficiencia o debilidad que puede permitir a un usuario no legitimo y/o no autorizado realizar acciones para atentar contra la disponibilidad, privacidad o integridad de los datos de un sistema.

Autenticación – Acción de acreditar que el usuario, servicio o sistema es realmente quien dice dice ser y verificar así su identidad al acceder a un sistema o información.

Encriptación – Proceso de codificar la información mediante algoritmos criptográficos para que el contenido de la información solo se puede descifrar y, por lo tanto, leer por sus legitimos destinatarios.

XML – El eXtensible Markup Languaje es un lenguaje para la estructuración y representación de la información. Su formato es versátil y estructurado, permitiendo almacenar información de forma relativamente legible y para gran variedad de informaciones. Esto permite que que si varios programas se tengan que intercambiar información o importar/exportar configuraciones lo puedan hacer mediante este formato.

HTML – El HyperText Markup Language es un lenguaje de marcado que sirve principalmente para la estructuración de páginas web. Es un estándar basado en el uso de las etiquetas para delimitar la estructura y los elementos del contenido. Su funcionamiento superficial es similar al formato XML.

SSH – Protocolo de Shell Seguro (Secure SHell). Permite establecer conexiones remotas seguras con otros dispositivos. Funciona con arquitectura cliente-servidor, donde el destino de la conexión es el servidor y el origen de la conexión el cliente. La comunicación se encuentra autenticada y cifrada.

Hash – producto de aplicar una función criptográfica para obtener una cadena de caracteres de longitud fija a partir de un conjunto de datos de entrada de longitud arbitraria. El funcionamiento de las funciones de hash provoca que un pequeño cambio en los datos de entrada pueda producir un resultado completamente diferente.

Checksum – Método de verificación de la integridad de una información. El objetivo es detectar modificaciones o errores en la transmisión de una información. Si se el hash de los datos en origen y destino son idénticos significa que el mensaje se conserva integro. Si los hashes difieren implica que se ha modificado la información, ya sea maliciosa o accidentalmente.

10. Bibliografía

[1] «Consultora TI especializada en soluciones de almacenamiento, seguridad, virtualización y cloud computing», *Brain2store*. https://www.brain2store.com/ (accedido 26 de marzo de 2023).

[2] «Zabbix :: The Enterprise-Class Open Source Network Monitoring Solution». https://www.zabbix.com/index (accedido 26 de marzo de 2023).

[3] A. Hat Red, «Ansible is Simple IT Automation». https://www.ansible.com (accedido 26 de marzo de 2023).

[4] M. J. Gamez, «Objetivos y metas de desarrollo sostenible», *Desarrollo Sostenible*. https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/ (accedido 26 de marzo de 2023).

[5] «Ansible playbooks — Ansible Documentation». https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_intro.html (accedido 26 de marzo de 2023).

[6] «Zabbix Integrations and Templates». https://www.zabbix.com/integrations (accedido 26 de marzo de 2023).

[7] «Cloud IT infrastructure spending worldwide 2026», *Statista*. https://www-statista-com.eu1.proxy.openathens.net/statistics/503686/worldwide-cloud-it-infrastructure-market-spending/ (accedido 9 de mayo de 2023).

[8] «Enterprise infrastructure buyer and cloud deployment share 2026», *Statista*. https://www-statista-com.eu1.proxy.openathens.net/statistics/1313867/enterprise-infrastructure-buyer-cloud-deployment-share/ (accedido 9 de mayo de 2023).

[9] «Global cloud shift revenue 2025», *Statista*. https://www-statista-com.eu1.proxy.openathens.net/statistics/1298890/worldwide-enterprise-it-revenue-by-type/ (accedido 9 de mayo de 2023).

[10] «Cloud deployment worldwide 2022», *Statista*. https://www-statista-com.eu1.proxy.openathens.net/statistics/1323967/cloud-deployment-use-worldwide-location/ (accedido 9 de mayo de 2023).

[11] «Global server share by OS 2018-2019», *Statista*. https://www.statista.com/statistics/915085/global-server-share-by-os/ (accedido 9 de mayo de 2023).

[12] «Usage Statistics and Market Share of Operating Systems for Websites, April 2023». https://w3techs.com/technologies/overview/operating_system (accedido 11 de abril de 2023).

[13] «Server Operating System Market Share, Size | Analysis by 2029». https://www.fortunebusinessinsights.com/server-operating-system-market-106601 (accedido 26 de marzo de 2023).

[14] «VMware vSphere | Enterprise Workload Platform», *VMware*. https://www.vmware.com/products/vsphere.html (accedido 5 de abril de 2023).

[15] «Data Storage: Data Storage in the Cloud & Data Center Storage | NetApp». https://www.netapp.com/data-storage/ (accedido 5 de abril de 2023).

[16] «Download and install Zabbix». https://www.zabbix.com/download (accedido 26 de marzo de 2023).

[17] «Download Zabbix appliance». https://www.zabbix.com/download_appliance (accedido 5 de abril de 2023).

[18] «Installing Ansible on specific operating systems — Ansible Documentation». https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html (accedido 11 de abril de 2023).

[19] I. T. L. Computer Security Division, «Security Content Automation Protocol | CSRC | CSRC», *CSRC | NIST*, 7 de diciembre de 2016. https://csrc.nist.gov/Projects/Security-Content-Automation-Protocol (accedido 6 de abril de 2023).

[20] «Home | OpenSCAP portal». https://www.open-scap.org/ (accedido 26 de marzo de 2023).

[21] «Releases · ComplianceAsCode/content», *GitHub*. https://github.com/ComplianceAsCode/content/releases (accedido 26 de marzo de 2023).
[22] «Download Tenable Nessus Vulnerability Assessment», *Tenable®*. https://www.tenable.com/products/nessus (accedido 10 de junio de 2023).
[23] «Working With Modules — Ansible Documentation». https://docs.ansible.com/ansible/2.9/user_guide/modules.html (accedido 7 de mayo de 2023).
[24] «Creating Roles — Ansible Documentation». https://galaxy.ansible.com/docs/contributing/creating_role.html (accedido 3 de mayo de 2023).
[25] «Ansible Galaxy». https://galaxy.ansible.com/ (accedido 26 de marzo de 2023).
[26] «Puppet Infrastructure & IT Automation at Scale | Puppet by Perforce». https://www.puppet.com/ (accedido 15 de mayo de 2023).
[27] «saltstack/salt». SaltStack, 10 de junio de 2023. Accedido: 10 de junio de 2023. [En línea]. Disponible en: https://github.com/saltstack/salt
[28] P. Leach y J. Bond <jean@puppet.com>, «The Puppet language». https://puppet.com/docs/puppet/7/puppet_language.html (accedido 10 de junio de 2023).
[29] «Nagios - The Industry Standard In IT Infrastructure Monitoring», *Nagios*. https://www.nagios.org/ (accedido 10 de junio de 2023).
[30] «Tenable Security Center (Formerly Tenable.sc)», *Tenable®*. https://www.tenable.com/products/tenable-sc (accedido 10 de junio de 2023).
[31] «Nexpose On-Premise Vulnerability Scanner», *Rapid7*. https://www.rapid7.com/products/nexpose/ (accedido 10 de junio de 2023).

11. Anexos

11.1. Ejemplo de recogida de datos de eventos de /var/log/sudo.log

Timestamp	Local time	Value
2023-06-13 16:22:24	2023-06-13 14:22:06	2023-06-13T14:22:06.223836+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/0 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/vi /etc/zabbix/zabbix_agent2.conf
2023-06-13 15:26:24	2023-06-13 13:26:12	2023-06-13T13:26:12.070242+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/su
2023-06-13 15:22:24	2023-06-06 12:05:33	2023-06-06T12:05:33.436391+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/ls -la
2023-06-13 15:22:24	2023-06-06 12:00:14	2023-06-06T12:00:14.886382+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/ls -la
2023-06-13 15:22:24	2023-06-06 11:56:12	2023-06-06T11:56:12.155694+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/ls -la
2023-06-13 15:22:24	2023-06-06 11:55:44	2023-06-06T11:55:44.568212+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/ls
2023-06-13 15:22:24	2023-06-06 11:54:18	2023-06-06T11:54:18.962352+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/ls
2023-06-13 15:22:24	2023-06-06 11:49:51	2023-06-06T11:49:51.857963+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/apt update
2023-06-13 15:22:24	2023-06-06 11:49:49	2023-06-06T11:49:49.830323+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/apt updatew
2023-06-13 15:22:24	2023-06-06 11:46:01	2023-06-06T11:46:01.460691+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/ls
2023-06-13 15:22:24	2023-06-06 11:39:27	2023-06-06T11:39:27.124098+00:00 tfm-ubuntu01 sudo: test : user NOT in sudoers ; TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/ls
2023-06-13 15:22:24	2023-06-06 11:38:00	2023-06-06T11:38:00.245807+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/passwd test
2023-06-13 15:22:24	2023-06-06 11:37:17	2023-06-06T11:37:17.239246+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/passwd test
2023-06-13 15:22:24	2023-06-06 11:36:18	2023-06-06T11:36:18.742717+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/passwd test
2023-06-13 15:22:24	2023-06-06 11:36:01	2023-06-06T11:36:01.916517+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/sbin/useradd test
2023-06-13 15:22:24	2023-06-06 11:34:06	2023-06-06T11:34:06.478083+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/ls
2023-06-13 15:22:24	2023-06-06 11:34:02	2023-06-06T11:34:02.123553+00:00 tfm-ubuntu01 sudo: ssierrago : 3 incorrect password attempts ; TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/ls
2023-06-13 15:21:24	2023-06-06 11:32:55	2023-06-06T11:32:55.149103+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/0 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/su
2023-06-13 15:21:24	2023-06-06 11:32:35	2023-06-06T11:32:35.855393+00:00 tfm-ubuntu01 sudo: ssierrago : 3 incorrect password attempts ; TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/ls
2023-06-13 15:21:24	2023-06-06 11:32:12	2023-06-06T11:32:12.549820+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/0 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/ls
2023-06-13 15:21:24	2023-06-06 11:30:47	2023-06-06T11:30:47.564293+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/0 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/vi /var/log/sudo.log
2023-06-13 15:21:24	2023-06-04 20:10:07	2023-06-04T20:10:07.015967+00:00 tfm-ubuntu01 sudo: ssierrago : TTY=pts/1 ; PWD=/home/ssierrago ; USER=root ; COMMAND=/usr/bin/vi /etc/snmp/