

UNIVERSIDAD OBERTA DE CATALUÑA
UNIVERSIDAD AUTÓNOMA DE BARCELONA
UNIVERSIDAD ROVIRA I VIRGILI
UNIVERSIDAD DE LAS ISLAS BALEARES

IMPLEMENTACIÓN DE WAZUH EN UNA ORGANIZACIÓN PÚBLICA

TRABAJO FIN DE MÁSTER
MÁSTER UNIVERSITARIO EN SEGURIDAD
DE LAS TICs

Autor: **Javier Polo Cózar**

Director: **Pau del Canto Rodrigo**

Responsable Área: **Víctor García Font**

Junio 2020



Esta obra está sujeta a una licencia de [Reconocimiento-NoComercial-SinObraDerivada 3.0 España \(CC BY-NC-ND 3.0 ES\)](https://creativecommons.org/licenses/by-nc-nd/3.0/es/) de Creative Commons

AGRADECIMIENTOS

"Porque la gratitud en silencio no sirve a nadie."

*A **Rosa**, por el apoyo, la paciencia y el tiempo dedicado a la familia mientras realizaba este trabajo en particular y el Máster en general.*

*A **Violeta** y **Mireia**, los dos luceritos de mi vida.*

*A **Pau del Canto Rodrigo**, director del TFM, por sus consejos y guía durante su desarrollo.*

*A **Juan Luis Garrido Castro**, por los consejos aportados y por permitirme realizar este trabajo en la organización.*

A los que ya no están, siempre estarán en nuestra mente y en nuestros corazones.

Muchas gracias a todos.

FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>Implementación de Wazuh en una organización pública</i>
Nombre del autor:	<i>Javier Polo Cózar</i>
Nombre del consultor:	<i>Pau del Canto Rodrigo</i>
Nombre del PRA:	<i>Víctor García Font</i>
Fecha de entrega (mm/aaaa):	<i>06/2020</i>
Titulación:	<i>Máster en Seguridad de las TIC</i>
Área del Trabajo Final:	<i>Seguridad empresarial</i>
Idioma del trabajo:	<i>Español</i>
Palabras clave	<i>SIEM, Monitorización, Wazuh, HIDS, IDS, amenazas, detección, vulnerabilidades, bastionado, intrusos</i>
Resumen del trabajo (máximo 250 palabras): <i>Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo</i>	
<p>La información se ha convertido en el recurso más valioso en el mundo y debemos esforzarnos en protegerla mejorando nuestras capacidades de detección de ciberataques. Los SIEM pueden ayudarnos a ello y pueden ser piezas muy importantes para asegurar y proteger los activos de las organizaciones y de todo el tráfico de red.</p> <p>En este trabajo fin de Máster hemos implementado la arquitectura de Wazuh y ELK Stack en nuestra organización, permitiéndonos protegerla de una forma multidisciplinar: correctiva (mediante la detección de vulnerabilidades), preventiva (mediante el bastionado de servidores), proactiva (mediante la recolección de logs, integridad de ficheros o análisis de malware en tiempo real), informativa (mediante diferentes tipos de notificaciones), reactiva (mediante mecanismos de respuesta activa (active response) ante las diferentes alertas generadas) y personalizada (permiéndonos monitorizar dispositivos sin agente y creando nuestras propias reglas y decodificadores).</p> <p>Hemos descubierto una solución de software libre muy completa. Además, al tratarse nuestra organización de una administración pública, nos ayudará a cumplir con el Esquema Nacional de Seguridad (ENS), de obligado cumplimiento desde el año 2010.</p>	
Abstract (in English, 250 words or less):	
<p>Data have turned into the most valuable resource in the world and we must make an effort to protect them, improving our cyber attack detection capabilities. SIEMs can help us to achieve it so they can become very important tools to secure and protect enterprise assets and network traffic.</p> <p>In this Master's thesis we have deployed the Wazuh and ELK Stack architecture in our organization, allowing us to protect it in a multidisciplinary way: corrective (through vulnerability detection), preventive (through server hardening), reactive (through active response mechanisms which are triggered when alerts are generated) and customized (being able of monitoring agentless devices and creating our own rules and decoders).</p> <p>We have discovered a very complete open source solution. Due to the fact that our organization is a public administration, it will help us to accomplish with the National Security Framework (ENS), which is mandatory since the year 2010.</p>	

ÍNDICE

Lista de Acrónimos	XIII
Glosario	XV
1 Introducción	1
1.1 Contexto y justificación	1
1.2 Objetivos	2
1.2.1 Objetivos generales	2
1.2.2 Objetivos específicos	2
1.3 Metodología	3
1.4 Planificación: fases y tareas	3
1.5 Recursos software y hardware	7
1.5.1 Recursos software	7
1.5.2 Recursos hardware	7
1.6 Estado del arte	7
1.7 Estructura del documento	8
2 Investigación y Análisis	9
2.1 Estudio y análisis general de los SIEM	9
2.1.1 Definición de un SIEM	9
2.1.2 Definición de un IDS	10
2.1.3 Arquitectura general de un SIEM	10
2.2 Necesidad de un SIEM en la empresa	11
2.3 Requisitos del SIEM a implantar	12
2.4 Elección del SIEM	13
2.5 Análisis del SIEM elegido: Wazuh + Elk Stack	16
2.5.1 Capacidades de Wazuh	16
2.5.2 Componentes de Wazuh	17
2.5.3 Arquitectura de Wazuh	18
3 Diseño y toma de decisiones	21
3.1 Arquitectura de red de la organización	21
3.2 Elección de implantación: virtual, física o contenedor	24
3.2.1 Implantación en una máquina física	24

3.2.2	Implantación en un contenedor Docker	24
3.2.3	Implantación de una imagen virtual	25
3.2.4	Método de implantación seleccionado	26
3.3	Arquitectura de red: centralizada o distribuida	26
3.4	Diseño final de implantación y arquitectura	27
4	Instalación	29
4.1	Instalación en pruebas del SIEM	29
4.1.1	Instalación de Wazuh Server	29
4.1.2	Instalación del servidor ELK Stack	30
4.1.3	Actualización de la versión de Wazuh y de ELK	30
4.1.4	Problemas de seguridad en una instalación por defecto	30
4.2	Instalación en producción del SIEM securizado	31
4.2.1	Securizar la API de Wazuh	31
4.2.2	Securizar Elastic Stack	33
4.2.3	Autenticación para Elastic Stack	36
4.3	Monitorización de los servidores Wazuh en Nagios	39
5	Despliegue	41
5.1	Despliegue automático de agentes	41
5.2	Despliegue automático en sistemas Linux	41
5.3	Despliegue automático en sistemas Windows	45
5.3.1	Despliegue del agente Wazuh por GPO	45
5.4	Creación de grupos en Wazuh	45
6	Casos de uso	47
6.1	File Integrity Monitoring: FIM	47
6.2	Auditoría mediante who-data	50
6.2.1	Auditoría who-data en Linux	50
6.2.2	Auditoría who-data en Windows	52
6.3	Detección de vulnerabilidades	54
6.3.1	Detección de vulnerabilidades en servidores Ubuntu	54
6.3.2	Detección de vulnerabilidades en servidores Windows	56
6.4	Configuración centralizada	58
6.5	Integración con VirusTotal	59
6.5.1	Usar FIM para monitorizar un directorio	60
6.6	Monitorización sin agente (agentless)	62
6.6.1	Creación del decoder	63
6.6.2	Creación de reglas: atomic y composite rules	64
6.7	Active Response	68
6.7.1	Detección del ataque	68
6.7.2	Definición del comando	68
6.7.3	Definir la respuesta activa	69
6.7.4	Generar una alerta cuando se dispara una respuesta activa	69
6.8	Bastionado de servidores	71
6.8.1	Monitorización de políticas de seguridad con SCA	71
6.9	Notificaciones de alertas y generación de informes	75
6.9.1	Notificación de alertas	75
6.9.2	Envío de informes diarios	76

6.10 Integración con APIs externas: Slack	79
7 Conclusiones y mejoras	81
7.1 Conclusiones	81
7.2 Mejoras	82
Bibliografía	85
Libros	85
Artículos	86
Recursos web	86

Apéndices

Apéndice A Hyper-V	A-1
A.1 Creación de switch virtual en Hyper-V	A-1
A.2 Creación de máquinas virtuales	A-2
A.2.1 Servidor Wazuh en pruebas	A-2
A.2.2 Servidor Wazuh en producción	A-3
A.2.3 Servidor ELK Stack en pruebas	A-3
A.2.4 Servidor ELK Stack en producción	A-3
A.3 Clúster de conmutación por error: alta disponibilidad	A-4
Apéndice B Ubuntu Server 18.04.4 LTS	B-1
B.1 Instalación Ubuntu Server 18.04.4 LTS en Hyper-V	B-1
B.2 Configurar Ubuntu 18.04.4 LTS	B-2
B.2.1 Deshabilitar cloud-init	B-2
B.2.2 Configuración de la red	B-2
B.2.3 Configuración del FQDN - DNS	B-3
B.2.4 Cambiar zona horaria	B-5
Apéndice C Instalación de Wazuh Server	C-1
C.1 Instalación desde repositorios	C-1
C.1.1 Añadir el repositorio de Wazuh	C-1
C.1.2 Instalación del Manager de Wazuh	C-2
C.1.3 Instalación de la API de Wazuh	C-2
C.1.4 Desactivar actualizaciones automáticas de Wazuh	C-3
C.1.5 Instalar Filebeat	C-3
C.2 Actualización Wazuh Manager, Wazuh API y Filebeat	C-4
Apéndice D Instalación del servidor ELK	D-1
D.1 Pre-requisitos ELK Stack	D-1
D.2 Elasticsearch	D-2
D.3 Kibana	D-3
D.4 Problemas con Kibana	D-4
D.4.1 Kibana no arranca: Kibana server is not ready yet	D-4
D.4.2 Another Kibana instance appears to be migrating the index	D-4
D.5 Actualización Kibana y Elasticsearch	D-5

D.6	Actualización del plugin de Kibana	D-5
Apéndice E Certificados digitales		E-1
E.1	Generación de certificados para wazuh-server	E-1
E.2	Generación de certificados para wazuh-elk	E-2
Apéndice F Monitorización con Nagios		F-1
F.1	Instalación plugins de Nagios	F-1
F.2	Instalación cliente NRPE v3.2.1	F-1
Apéndice G Ansible		G-1
G.1	Instalación Ansible en el servidor Wazuh	G-1
G.2	Configuración del servidor Ansible y sus clientes	G-1
G.2.1	Intercambio de claves entre servidores por ssh	G-1
G.3	Instalación de Audit	G-3
Apéndice H Despliegue por GPO		H-1
H.1	Problema implementación de software GPO	H-1
H.2	Fichero de transformación .mst con InstEd	H-1
H.3	Creación de la directiva EQU_InstalarAgenteWazuh	H-2

ÍNDICE DE FIGURAS

1.1	Diagrama de Gantt: Planificación temporal del trabajo	6
2.1	Arquitectura de un SIEM	11
2.2	Cuadrante mágico de Gartner para SIEM 2020	13
2.3	Arquitectura de Wazuh centralizada	18
2.4	Arquitectura de Wazuh distribuida	19
2.5	Cluster Wazuh	20
3.1	Esquema de red sede principal	22
3.2	Esquema de red sede formación	23
3.3	Diseño final de implantación y arquitectura Wazuh+ELK Stack	28
4.1	Securización de Wazuh API	33
4.2	Conexión segura portal Kibana	38
4.3	Portal autenticación Kibana con X-Pack	38
4.4	Servidor Wazuh monitorizado en Nagios	39
4.5	Servidor ELK monitorizado en Nagios	39
5.1	Ejecución exitosa de Ansible	43
5.2	Agente Wazuh en ejecución	44
5.3	Agentes Wazuh de servidores Linux en Kibana	44
5.4	Agentes Wazuh de servidores Windows en Kibana	45
6.1	Integridad de fichero de contraseñas Keepass	48
6.2	Alerta en el fichero de contraseñas Keepass	49
6.3	Auditoría de ficheros Linux	51
6.4	Auditoría de ficheros en Windows - Integrity Monitoring	52
6.5	Auditoría de ficheros en Windows	53
6.6	Vulnerabilidades en servidores Linux/Ubuntu	57
6.7	Vulnerabilidades en servidores Windows	59
6.8	Alerta virus con VirusTotal	61
6.9	Top alertas hacolx08: virus detectado	61
6.10	Intento de autenticación erróneo en switch HP	66
6.11	Alerta regla compuesta: 5 intentos de acceso ssh en 10 minutos switch HP	67
6.12	Bloqueo de IP con active response	70

6.13	Políticas SCA servidor miembro Windows 2012 R2	73
6.14	SCA - CIS benchmark W2012R2 Member Server L1	74
6.15	SCA - CIS benchmark Debian 9 L1	74
6.16	SCA - CIS benchmark Debian 8 L1	75
6.17	Alerta de ataque por fuerza bruta ssh por correo	77
6.18	Informe diario alertas nivel 10 o más Wazuh	78
6.19	Notificación en Slack de alertas de Wazuh	79
A.1	Creación de switch virtual en Hyper-V	A-2
A.2	Seleccionar rol para configurar alta disponibilidad	A-4
A.3	Máquinas con alta disponibilidad	A-5
A.4	Roles creados para alta disponibilidad	A-5
B.1	Agregar entrada en DNS servidor Wazuh	B-4
B.2	Agregar entrada en DNS servidor ELK	B-4
B.3	Registro de zona de búsqueda inversa Wazuh	B-4
C.1	systemctl status wazuh-manager	C-2
C.2	systemctl status wazuh-api	C-3
D.1	Elasticsearch	D-3
D.2	Kibana desplegado en pruebas	D-5
F.1	Servidor Wazuh monitorizado en Nagios	F-4
F.2	Servidor ELK monitorizado en Nagios	F-4
G.1	ssh-copy-id a servidor hacolx09	G-2
G.2	Despliegue de auditd en servidores Ubuntu mediante Ansible	G-4
H.1	Creación Transformación en InstEd	H-2

ÍNDICE DE TABLAS

1.1	Planificación del TFM	5
2.1	Ventajas y desventajas de AlientVault OSSIM	14
2.2	Ventajas y desventajas de Apache Metron	14
2.3	Ventajas y desventajas de SIEMonster (CE)	14
2.4	Ventajas y desventajas de Prelude OSS	15
2.5	Ventajas y desventajas de MozDef	15
2.6	Ventajas y desventajas de Wazuh	15
6.1	Expresiones regulares en decoder HP	63
6.2	Matching del decoder	64
F.1	Servicios Wazuh+ELK a monitorizar en Nagios	F-3

LISTA DE ACRÓNIMOS

BOJA Boletín Oficial de la Junta de Andalucía 12

CA Autoridad Certificadora 30–32, 34

CIS Center for Internet Security 71

CPE Common Platform Enumeration 59

CVE Common Vulnerabilities and Exposure 16

ENS Esquema Nacional de Seguridad 1, 2, 11, 12

FIM File Integrity Monitoring 47

FQDN Fully Qualified Domain Name 30, E-2

GDPR General Data Protection Regulation 17

GPG13 Good Practice Guide 13 17

GPO Group Policy Object 45

HA High Availability 19

HIDS Host Intrusion Detection System 10, 17

IDMEF Intrusion Detection Message Exchange Format 15

IDS Intrusion Detection System 1

IOCs Indicators of Compromise 17

IT Information Technology 10

JSON JavaScript Object Notation 79

LTS Long Term Support 27

- NAT** Network Address Translation A-1
- NIDS** Network Intrusion Detection System 10
- NIPS** Network Intrusion Prevention System 10
- NRPE** Nagios Remote Plugin Executor 39, F-1
- NVD** National Vulnerability Database 54, 56
- OVAL** Open Vulnerability Assesment Language 18
- PCI DSS** Payment Card Industry Data Security Standard 17
- RDP** Remote Desktop Protocol 21
- SCA** Security Configuration Assessment 71
- SEM** Security Event Management 8
- SIEM** Security Information and Event Management 1
- SIM** Security Information Management 7
- SOAR** Security Orchestration And Automation Response 8
- TFM** Trabajo Fin de Máster 3
- UEBA** User Event Behavioral Analysis 8
- VPN** Virtual Private Network 21
- XCCDF** eXtensible Configuration Checklist Description Format 18

GLOSARIO

agentless se refiere a las operaciones donde no se necesita ningún servicio, demonio o proceso (un agente) ejecutándose en un segundo plano en la máquina que se quiere monitorizar. [27](#)

bastionado conjunto de actividades llevadas a cabo para reforzar al máximo posible la seguridad de un equipo. [2](#)

Docker Compose Compose es una herramienta para definir y ejecutar aplicaciones Docker de múltiples contenedores. [24](#)

Docker Desktop es una aplicación para MacOS y Windows para construir y compartir aplicaciones en contenedores y microservicios. [24](#)

Docker Toolbox es una solución obsoleta para utilizar Docker en sistemas Mac y Windows antiguos que no cumplen los requisitos de Docker Desktop. [24](#)

endpoint es un dispositivo informático remoto que se comunica con una red a la que está conectado. Por ejemplo: ordenadores de escritorio, portátiles, teléfonos móviles, tablets, servidores, estaciones de trabajo, etc... [26](#), [41](#)

Hyper-V es un programa de virtualización de Microsoft basado en un hipervisor para los sistemas de 64 bits con los procesadores basados en AMD-V o Tecnología de virtualización de Intel. [24](#), [25](#), [A-1](#)

Incoming Webhook es una manera sencilla de enviar mensajes desde las aplicaciones a Slack. Al crear un Incoming Webhook se obtiene una URL única a la que enviar un payload JSON con el mensaje de texto y algunas opciones. [79](#)

indicador de compromiso Los Indicadores de Compromiso o *Indicators of Compromise* (IOCs) hacen referencia a una tecnología estandarizada que consiste en definir las características técnicas de una amenaza por medio de las evidencias existentes en un equipo comprometido, de manera que puedan servir para identificar otros ordenadores afectados por la misma amenaza o prevenirlos de la misma. [17](#)

machine learning aprendizaje automático. 8

Nginx es un servidor web/proxy inverso ligero de alto rendimiento y un proxy para protocolos de correo electrónico. 34

on-premise el término on-premise o en local se refiere al tipo de instalación de una solución de software. Esta instalación se lleva a cabo dentro del servidor y la infraestructura TIC de la empresa. Es el modelo tradicional de aplicaciones empresariales y el opuesto a la nube (cloud). 16, 17

Powershell PowerShell (originalmente llamada Windows PowerShell) es una interfaz de consola (CLI) con posibilidad de escritura y unión de comandos mediante scripts. Esta interfaz de consola está diseñada para su uso por parte de administradores de sistemas, con el propósito de automatizar tareas o realizarlas de forma más controlada. 25, 45

snapshot una copia instantánea de volumen o snapshot es una instantánea o foto del estado de un sistema en un momento determinado 29

VirtualBox es un software de virtualización para arquitecturas x86/amd64 desarrollado por Oracle Corporation. 25

VMware es un software de virtualización multiplataforma (Windows, Linux y MacOS) para procesadores Intel desarrollado por VMware Inc. 25

workaround solución alternativa, rodeo. 26, 57

X-Pack es una extensión de Elastic Stack que proporciona seguridad, alerta, monitorización, informes, aprendizaje automático y muchas otras capacidades. 34



1

INTRODUCCIÓN

1.1. Contexto y justificación

En los últimos años, los datos se han convertido en el recurso más valioso en el mundo. El abanico de las amenazas de seguridad sobre ellos se ha abierto considerablemente y los sistemas de ataque se han vuelto cada vez más sofisticados. Según el Informe del Análisis de las Brechas de Seguridad de Datos de Verizon del 2019, [Verizon Data Breach Investigations Report \(2019 DBIR\)](#) (Verizon, 2019), el 69 % de los ataques fueron realizados por personas ajenas a la organización. De forma más específica al sector al que va dirigido nuestro trabajo, en la Administración Pública, se incrementa el porcentaje hasta el 75 %. Además, los datos comprometidos fueron en un 68 % datos internos de la organización y el 68 % de los ataques identificados se descubrieron, como poco, un mes más tarde de la brecha inicial. Eso sin tener en cuenta que hay un alto número de ataques que no son detectados, con lo que el porcentaje sería mucho mayor. Teniendo en cuenta esta información, deberíamos esforzarnos en mejorar nuestras capacidades de detección de ciberataques (Podzins et al. 2019).

No hay duda de que los sistemas de detección de intrusos(IDS¹) pueden ayudarnos a ello y son importantes para asegurar y proteger los activos de las organizaciones y de todo el tráfico de red. Sirven de salvaguarda para defender el acceso a los recursos de red de una organización.

El presente trabajo buscará implementar en nuestra organización (una delegación provincial de una Administración Pública Regional) un SIEM², del cual carece actualmente. De esta forma, nos será de ayuda para adecuarnos al Esquema Nacional de Seguridad (ENS), de obligado cumplimiento para todas las Administraciones Públicas desde el año 2010.

La idea es que haga un trabajo correctivo (mediante la detección de vulnerabilidades), pre-

¹Intrusion Detection System

²System Information Event Management

ventivo (mediante el bastionado de servidores), proactivo (mediante la recolección de logs, integridad de ficheros o análisis de malware en tiempo real) e informativo (notificándonos cuando suceda algún incidente de seguridad). Para ello, se analizarán las distintas opciones existentes en el mercado y se escogerá aquella que mejor se ajuste a nuestras necesidades, posibilidades e indicaciones de la dirección para su implantación.

1.2. Objetivos

Podemos distinguir entre objetivos generales y específicos.

1.2.1. Objetivos generales

Los objetivos generales de este trabajo son:

- Ser capaces de planificar el trabajo de forma adecuada y de seguir de forma práctica dicha planificación, alcanzando las metas propuestas en cada uno de los hitos.
- Analizar las características de los SIEMs y las opciones disponibles en el mercado.
- Determinar el más adecuado para nuestro entorno empresarial.
- Securizar nuestra organización con uno de estos sistemas, del que adolece actualmente.

1.2.2. Objetivos específicos

Los objetivos específicos de este trabajo son:

- Estudiar y determinar el mejor método de implantación del SIEM para nuestra organización: virtual vs. físico vs. contenedor.
- Estudiar y elegir la arquitectura de red más adecuada de implantación: centralizada vs. distribuida.
- Implantar un SIEM que permita monitorizar la mayoría de los distintos dispositivos de red con los que trabajamos (PCs, servidores, switches, cabina de almacenamiento, cortafuegos, etc. . .) así como los distintos sistemas operativos (Windows, Linux, etc. . .).
- Securizar el acceso a las aplicaciones web que utilicemos para realizar la monitorización, evitando, por ejemplo, accesos anónimos, cuentas genéricas o tráfico sin cifrar.
- Estudiar y poner en marcha, en la medida de lo posible, herramientas de automatización para la implantación del sistema.
- Estudiar la posibilidad de utilizar guías de seguridad con el SIEM que nos ayuden a cumplir con el ENS, para poder realizar un **bastionado** (hardening) correcto de los servidores, tanto en Windows como en Linux.
- Estudiar la posibilidad de detectar vulnerabilidades en los diferentes sistemas operativos de los que disponemos.
- Integración del SIEM con la herramienta de monitorización de sistemas ya implantada en la organización ([Nagios](#)).

- Realizar diferentes casos de uso que demuestren los beneficios del sistema: integridad de ficheros, detección de malware, detección de ataques, etc...
- Tener una herramienta de detección de intrusos que nos permita realizar notificaciones a los usuarios o grupos de usuarios correspondientes cuando se produzca una incidencia.

1.3. Metodología

La metodología que seguiremos en el presente TFM³ será la siguiente:

1. Se establecerá un plan de trabajo realista y ajustado a la planificación preestablecida por la asignatura.
2. Se analizarán los distintos SIEMs existentes en el mercado y se elegirá el que mejor se adapte a nuestra organización.
3. Una vez elegido, se diseñará el mejor método de implantación (virtual, físico o mediante contenedor) y se determinará la arquitectura de red más adecuada.
4. Se instalará dicho sistema en nuestra organización, securizándolo de la mejor manera posible e intentando obtener el máximo partido de él; primero en un entorno de pruebas y luego en producción.
5. Una vez implantado, se visualizarán los resultados y los beneficios obtenidos con su implantación.
6. Se extraerán las conclusiones pertinentes y posibles mejoras que reflejaremos en la memoria final.

1.4. Planificación: fases y tareas

En la tabla 1.1 se muestran las distintas fases y tareas de las que consta el trabajo, así como su duración y fechas de inicio y fin.

	Fases y Tareas	Duración	Inicio	Fin
1	FASE DE DEFINICIÓN	12	19/02/20	01/03/20
1.1	Contexto y justificación del trabajo	2	19/02/20	20/02/20
1.2	Definición de objetivos	2	21/02/20	22/02/20
1.3	Metodología a seguir	2	23/02/20	24/02/20
1.4	Recursos a utilizar	2	25/02/20	26/02/20
1.5	Estado del arte	4	27/02/20	01/03/20
2	FASE DE PLANIFICACIÓN	5	27/02/20	02/03/20
2.1	Identificación de tareas	2	27/02/20	28/02/20
<i>Continúa en la página siguiente</i>				

³Trabajo Fin de Máster

	Fases y Tareas	Duración	Inicio	Fin
2.2	Cálculo de tiempos de tareas y entregas	2	29/02/20	01/03/20
2.3	Elaboración de Diagrama de Gantt	1	02/03/20	02/03/20
2.4	Redacción de la Fase de Definición y Planificación	8	24/02/20	02/03/20
2.5	Entrega del Plan de Trabajo (PEC1)	HITO	03/03/20	03/03/20
2.6	Revisión y corrección del Plan de Trabajo por el director	4	04/03/20	07/03/20
3	FASE DE INVESTIGACIÓN Y ANÁLISIS	10	04/03/20	13/03/20
3.1	Estudio y análisis general de los SIEM	2	04/03/20	05/03/20
3.2	Necesidad de un SIEM en la empresa	2	06/03/20	07/03/20
3.3	Requisitos del SIEM a implantar	2	08/03/20	09/03/20
3.4	Elección del SIEM	2	10/03/20	11/03/20
3.5	Análisis del SIEM elegido	2	12/03/20	13/03/20
3.6	Redacción de la Fase de Investigación y Análisis	5	09/03/20	13/03/20
4	FASE DE DISEÑO Y TOMA DE DECISIONES	6	14/03/20	20/03/20
4.1	Elección de implantación: virtual, física o contenedor	3	14/03/20	16/03/20
4.2	Elección de arquitectura de red: centralizada o distribuida	3	17/03/20	20/03/20
5	FASE DE INSTALACIÓN	10	21/03/20	30/03/20
5.1	Instalación en pruebas del SIEM	4	21/03/20	24/03/20
5.2	Instalación en producción del SIEM securizado	6	25/03/20	30/03/20
5.3	Redacción de fases de Diseño e Instalación	10	21/03/20	30/03/20
5.4	Entrega Fase de Análisis, Diseño e Instalación (PEC2)	HITO	31/03/20	31/03/20
5.5	Revisión y corrección de la PEC2 por el director	4	31/03/20	3/04/20
6	FASE DE DESPLIEGUE Y CASOS DE USO	29	31/03/20	28/04/20
6.1	Despliegue automático de agentes	4	31/03/20	3/04/20
6.2	Detección de vulnerabilidades	5	04/04/20	8/04/20
6.3	Bastionado con guías de seguridad	6	09/04/20	14/04/20
6.4	Integración con Nagios	2	15/04/20	16/04/20
6.5	Otros Casos de Uso	9	17/04/20	25/04/20
6.6	Notificaciones	2	26/04/20	27/04/20

Continúa en la página siguiente

Fases y Tareas		Duración	Inicio	Fin
6.7	Redacción de la Fase de Despliegue y Casos de Uso	10	18/04/20	27/04/20
6.8	Entrega Fase de Despliegue y Casos de Uso (PEC3)	HITO	28/04/20	28/04/20
6.9	Revisión y corrección de la PEC3 por el director	4	29/04/20	02/05/20
7	FASE DE REDACCIÓN Y ENTREGA MEMORIA	35	29/04/20	02/06/20
7.1	Conclusiones y mejoras	5	29/04/20	03/05/20
7.2	Redacción de la Memoria Final	29	04/05/20	01/06/20
7.3	Entrega de la Memoria Final	HITO	02/06/20	02/06/20
8	FASE DE PRESENTACIÓN Y DEFENSA	17	03/06/20	19/06/20
8.1	Elaboración de presentación en vídeo	6	03/6/20	08/06/20
8.2	Entrega presentación en vídeo	HITO	09/06/20	09/06/20
8.3	Defensa del TFM	HITO	15/06/19	19/06/19

Tabla 1.1: Planificación del TFM

En la página 6, en la figura 1.1, representamos el Diagrama de Gantt con el cronograma del TFM.

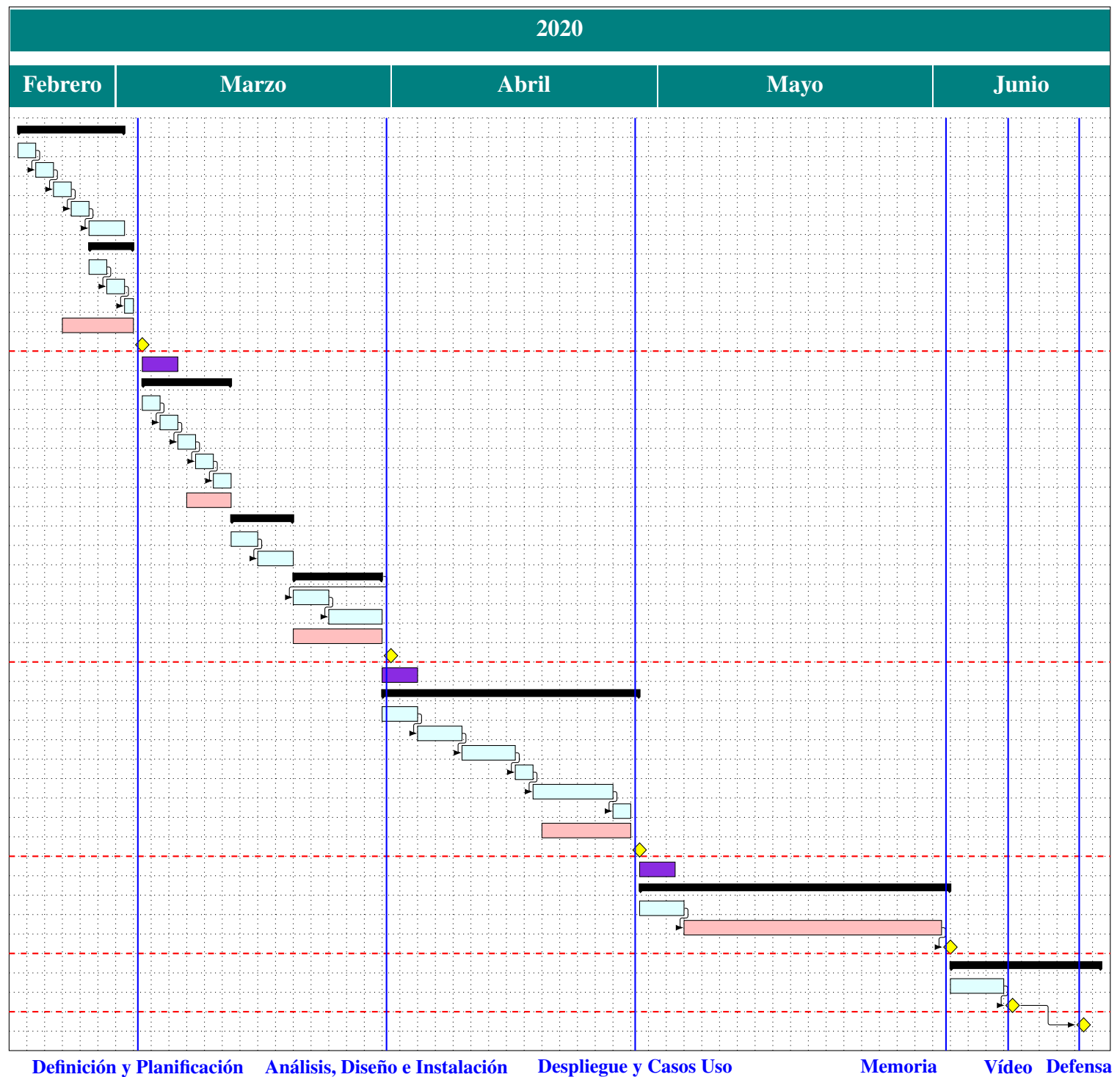


Figura 1.1: Diagrama de Gantt: Planificación temporal del trabajo

1.5. Recursos software y hardware

Podemos distinguir entre recursos software y hardware.

1.5.1. Recursos software

Se utilizarán, a priori⁴, los siguientes recursos software:

1. Sistemas Operativos
 - Microsoft Windows: 2012 R2, 2008 R2, 10 (Pro y Home), 7 Pro
 - GNU/Linux: Ubuntu Server LTS (18.x, 16.x, 14.x)
2. Virtualización/Contenedores: Hyper-V, VirtualBox, Docker
3. Edición y maquetación de memoria: L^AT_EX, TeXstudio 2.12.22, MiKTeX 2.9
4. Gestor de referencias bibliográficas: Zotero 5.0.82
5. Diseño gráfico y diagramas de red: Microsoft Visio 2007, Inkscape 0.92.3

1.5.2. Recursos hardware

Se utilizarán o accederán a los siguientes recursos hardware que forman parte de la empresa:

Servidores: Lenovo CTOX3650M5, DELL PowerEdge T610, HP XW4600

PCs: HP ProDesk 400 Gx

Electrónica de red: Switches HP 2530 48G-POE y 3800 48G-POE.

Cabina de Almacenamiento: EMC VNX5200

Cortafuegos: Fortigate 50E

1.6. Estado del arte

La necesidad de contar con sistemas SIEM en las empresas se ha incrementado en los últimos años especialmente por cómo están evolucionando los ciberataques sobre las empresas, pudiendo causar, entre otras: discontinuidad de sus servicios, fuga de datos y daños a su reputación.

Actualmente, los entornos TIC de las compañías se están volviendo cada vez más complejos e involucran a más dispositivos para los que hay que tener en cuenta una estrategia de seguridad en el proceso de negocio. Por tanto, las organizaciones han comenzado a invertir en los SIEMs para mejorar su seguridad (Mokalled et al. 2019).

El término SIEM fue acuñado en 2005 por Mark Nicolett y Amrit Williams, en el informe para Gartner llamado [Improve IT Security With Vulnerability Management](#), donde propusieron un sistema de información de seguridad que combinaba dos sistemas que hasta entonces habían estado separados:

Security Information Management (SIM): análisis y gestión de logs de seguridad

⁴Se irá ampliando la lista a lo largo del trabajo

Security Event Management (SEM): gestión de eventos de seguridad

Estaban limitados en cuanto a la cantidad de datos que podían procesar y la sofisticación en las alertas y visualizaciones que generaban. Ésta sería **la primera generación de SIEMs**, de las tres que identifican los analistas a día de hoy (Casetto, 2019).

La segunda generación de SIEMs está mejor equipada para manejar grandes cantidades de datos y de volúmenes de logs históricos. Estos SIEMs pueden correlacionar datos de logs históricos con eventos en tiempo real.

La tercera generación de SIEMs, propuesta por Gartner en 2017, combina las capacidades de los SIEMs tradicionales con dos nuevas tecnologías:

- **UEBA:** *User and Entity Behavioral Analytics*, que usa *machine learning* para establecer líneas base de comportamiento de los usuarios o de los sistemas de tecnologías de la información e identificar anomalías que podrían indicar incidencias de seguridad.
- **SOAR:** *Security Orchestration and Automation Response*, la automatización de la seguridad, orquestación y respuesta que puede ayudar a los analistas a investigar rápidamente incidentes y activar herramientas de seguridad para responder automáticamente a ellos.

Normalmente, los SIEMs de tercera generación tendrán un coste asociado y no nos será posible implementarlos en nuestra organización, con lo que nos limitaremos a los SIEM de segunda generación con algunas características de los de tercera, que cubrirán totalmente nuestras necesidades.

1.7. Estructura del documento

El documento está estructurado de la siguiente manera:

- En el capítulo 1 se llevará a cabo el plan de trabajo del Máster, realizando la definición y planificación de éste.
- En el capítulo 2 se realizará un estudio y análisis general de los SIEMs, por qué se necesita uno en la empresa, cuáles serán sus requisitos y, finalmente, por cuál nos hemos decantado. Una vez elegido, se describirán sus principales características.
- En el capítulo 3 se elegirá donde se montará el SIEM (máquina virtual, física o contenedor) y con qué arquitectura de red (centralizada o distribuida).
- En el capítulo 4 se instalará el SIEM en nuestra red, primero en un entorno de pruebas y luego en producción y se securizará el sistema.
- En el capítulo 5 se desplegarán los agentes de forma automática.
- En el capítulo 6 se usará el SIEM para detectar vulnerabilidades, el correcto bastionado de los servidores, integrarlo con Nagios y otros casos de uso que nos sean útiles. Asimismo, se estudiarán las notificaciones que el SIEM puede proporcionar.
- En el capítulo 7 se presentarán las conclusiones obtenidas al realizar el trabajo y las posibles mejoras que se podrían efectuar.
- En los apéndices A, B, C, D, E, F, G y H están los anexos al TFM, necesarios para su comprensión pero demasiado extensos para ser incluidos en el cuerpo del documento.

A large red graphic element consisting of a horizontal bar that turns upwards at a 45-degree angle, ending in a white number '2'.

INVESTIGACIÓN Y ANÁLISIS

2.1. Estudio y análisis general de los SIEM

2.1.1. Definición de un SIEM

Un SIEM es una herramienta que nos permite centralizar la interpretación de los registros relevantes de seguridad.

Las amenazas que detectan los SIEMs se engloban en algunos de los siguientes grupos:

- Vulnerabilidades
- Protocolos vulnerables
- Fallos de configuración por parte de los administradores
- Introducción de errores por el usuario de forma consciente o no
- Amenazas internas
- Amenazas externas

Nos permiten recopilar, normalizar y correlacionar eventos de seguridad, proporcionan inteligencia en materia de seguridad, descartan falsos positivos, evalúan el impacto de un ataque, unifican la gestión de la seguridad, centralizan la información e integran herramientas de detección de amenazas e intrusos.

2.1.2. Definición de un IDS

La detección de intrusiones es el acto de detectar eventos que se consideran inapropiados o no bienvenidos para el sistema. Dicha detección puede llevarse a cabo de forma manual, inspeccionando el tráfico de red y los registros de cada uno de los recursos o automáticamente usando herramientas. Una herramienta que automatiza el procesado de la información relacionada con las intrusiones se identifica como un Sistema de Detección de Intrusos (IDS). Los IDS se clasifican en:

1. **HIDS** (Host Intrusion Detection System): sistema de detección de intrusos a nivel de equipo.
2. **NIDS** (Network Intrusion Detection System): sistema de detección de intrusos a nivel de red.

No todos los sistemas de detección de intrusos de red pueden llevar a cabo una acción como resultado de la generación de una alerta. Estas características avanzadas son a veces la diferencia entre un NIDS y un **NIPS** (Network Intrusion Prevention System).

Un HIDS detecta eventos en un servidor o estación de trabajo y puede generar alertas como un NIDS pero además es capaz de inspeccionar el flujo de comunicaciones de forma completa y no se le aplican las técnicas de evasión de NIDS como los ataques de fragmentación. Asimismo, las comunicaciones encriptadas pueden monitorizarse porque el HIDS inspecciona el tráfico antes de que sea encriptado.

Un HIDS debería poder hacer los siguientes chequeos:

- Chequeo de integridad de ficheros
- Monitorización del registro del sistema
- Detección de rootkits
- Respuesta activa

2.1.3. Arquitectura general de un SIEM

Un SIEM debería proporcionar los siguientes servicios:

- Administración de logs: recolección, análisis, correlación y análisis forense.
- Cumplimiento de leyes y normas **IT**¹
- Monitorización de los registros de las aplicaciones
- Auditoría de acceso a objetos
- Alerta en tiempo real
- Monitorización de las actividades del usuario
- Tableros de visualización (dashboards)
- Informes
- Monitorización de la integridad de ficheros

¹Tecnologías de la Información

- Monitorización de logs y registros del sistema
- Seguridad en el punto final (endpoints)
- etc...

En la figura 2.1 podemos ver de forma gráfica los elementos con los que debe contar un SIEM.

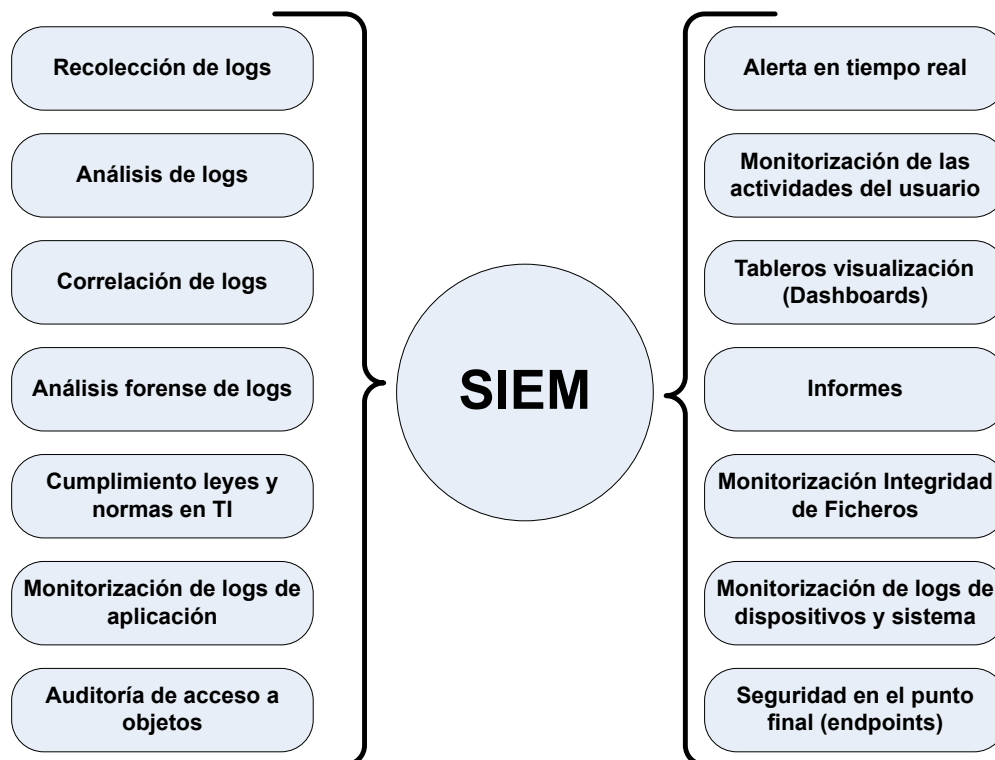


Figura 2.1: Arquitectura de un SIEM

2.2. Necesidad de un SIEM en la empresa

El [Real Decreto 3/2010 de 8 de Enero](#) por el que se regula el Esquema Nacional de Seguridad en el ámbito de la Administración Electrónica indica en su artículo 3, Ámbito de aplicación, que será de aplicación a lo establecido a su vez en el artículo 2 de la [Ley 11/2007 de acceso electrónico de los ciudadanos a los Servicios Públicos](#). En concreto en su apartado a):

- a) *A las Administraciones Públicas, entendiéndose por tales la Administración General del Estado, las Administraciones de las Comunidades Autónomas y las Entidades que integran la Administración Local, así como las entidades de derecho público vinculadas o dependientes de las mismas.*

Por lo tanto, el ENS² es de aplicación en nuestra organización desde el año 2010.

Dentro del Esquema Nacional de Seguridad, en el Anexo II, están reflejadas las medidas de seguridad a aplicar para lograr el cumplimiento de los principios básicos y requisitos mínimos establecidos por éste. Dichas medidas se dividen en tres grupos:

²Esquema Nacional de Seguridad

1. Marco organizativo
2. Marco operacional
3. Medidas de protección.

Dentro del marco operacional, que son las medidas a tomar para proteger la operación del sistema como un conjunto integral de componentes, se encuentra la Monitorización del Sistema. En dicho apartado se nos indica que para sistemas de categoría media y alta, *se dispondrán de herramientas de detección o de prevención de intrusión*. Por lo tanto, la implementación de dicho sistema, nos ayudará a cumplir con el ENS.

Por otra parte, y centrándonos en nuestra organización, la obligatoriedad del cumplimiento del ENS implica la necesidad de que el organismo tenga una política de seguridad establecida; lo cual fue plasmado en el [Decreto 1/2011 de 11 de Enero del BOJA](#)³ por el que se establece la política de seguridad de las tecnologías de la información y comunicaciones en la Administración de la Junta de Andalucía. En dicho decreto se indica que cada entidad deberá establecer su propia política de seguridad. Es en octubre del 2019 cuando se aprueba la política de seguridad de la información de la Consejería en la [Orden de 21 de Octubre de 2019, BOJA nº 211 de 31/10/2019](#). Concretamente y relacionado con la temática de este trabajo se indica en el artículo 25, apartado 2:

2. *Dado que los servicios pueden degradarse rápidamente debido a incidentes, aquellos deben estar sometidos a monitorización de manera continua para detectar anomalías en sus niveles de prestación y así poder actuar con celeridad.*

Por lo tanto, es evidente que la monitorización de sistemas y la implementación de un SIEM (que englobe a un sistema de detección y prevención de intrusos) es una necesidad en nuestra organización.

2.3. Requisitos del SIEM a implantar

Existen dos requisitos fundamentales que debe tener el SIEM elegido y son debidas a las políticas de nuestra organización. La Junta de Andalucía viene fomentando la difusión y utilización de software libre, siendo éste uno de los factores contemplados en el [Decreto 72/2003 de 18 de marzo, de Medidas de Impulso de la Sociedad del Conocimiento en Andalucía](#). De igual forma, en la [Orden de 21 de Febrero de 2005, sobre disponibilidad pública de los programas informáticos de la administración de la Junta de Andalucía y de sus organismos autónomos](#) en el artículo 1. Objeto, se indica que se va a:

- *poner a disposición pública el código fuente de los programas y aplicaciones informáticas y la documentación asociada a los mismos que sean propiedad de la Administración de la Junta de Andalucía y de sus Organismos Autónomos, que tendrán el carácter de software libre, así como establecer las condiciones para su libre uso y distribución.*

Es decir, que el software desarrollado por y para la Junta de Andalucía tendrá el carácter de software libre, creando un repositorio público y libre, así que parece lógico que el sistema que implantemos también lo sea.

El otro requisito es que en nuestra Delegación Provincial se busca un sistema que no tenga ningún coste asociado, lo cual casa con el requisito anterior. Por lo tanto el sistema a implantar

³Boletín Oficial de la Junta de Andalucía

deberá ser un SIEM de software libre sin coste asociado.

Además, según los objetivos específicos vistos en el apartado 1.2.2, como requisitos adicionales tendríamos:

- Que pueda implantarse en nuestros sistemas, soportando tecnología de virtualización, contenedor Docker, etc. . .
- Que soporte diferentes arquitecturas de implantación: centralizada, distribuida.
- Que pueda monitorizar la mayor parte de los dispositivos de red y de sistemas operativos que tenemos.
- Que permita ser securizado, evitando accesos anónimos, cuentas genéricas o tráfico sin cifrar.
- Que pueda ser automatizado en el despliegue.
- Que nos ayude a cumplir con el ENS.
- Que pueda detectar vulnerabilidades en los diferentes sistemas que tenemos.
- Que pueda integrarse con la herramienta de monitorización Nagios.
- Que permita detección de malware, ataques o integridad de ficheros.
- Que permita notificaciones.

En base a todos estos requisitos, elegiremos el SIEM que cumpla la mayoría de ellos.

2.4. Elección del SIEM

Si observamos en el cuadrante mágico de Gartner para 2020 en la figura 2.2, los líderes en los sistemas SIEM son: Splunk, IBM, Exabeam, Securonix, Rapid7, LogRhythm y Dell Technologies (RSA).



Figura 2.2: Cuadrante mágico de Gartner para SIEM 2020

Dichos SIEM no cumplen los dos requisitos fundamentales de la sección 2.3: que sean software libre (open source) y que no tengan coste asociado, por lo que se desestiman.

Hemos considerado los siguientes SIEM open source:

1. AlienVault OSSIM
2. Apache Metron (anteriormente OpenSOC de Cisco)

3. MozDef
4. Prelude OSS
5. SIEMonster Community Edition
6. Wazuh + ELK Stack

Los analizamos en las tablas [2.1](#), [2.2](#), [2.3](#), [2.4](#), [2.5](#) y [2.6](#).

AlientVault OSSIM	
Ventajas	Desventajas
Construido sobre proyectos open source probados	Falta de muchas características de versión de pago
Gran comunidad de usuarios y desarrolladores	No soporta plataformas en la nube como AWS/Azure
	Sin gestión de logs, visualizaciones, automatización o integraciones con terceros
	Arquitectura de servidor centralizado
	Problemas en el escalado

Tabla 2.1: Ventajas y desventajas de AlientVault OSSIM

Apache Metron	
Ventajas	Desventajas
Automatización con Ansible	Solo puede instalarse en determinados entornos y sistemas operativos
Instalación via Docker (Windows/Mac)	El interfaz no es muy intuitivo

Tabla 2.2: Ventajas y desventajas de Apache Metron

SIEMonster (Community Edition)	
Ventajas	Desventajas
Construido sobre proyectos open source probados	Versión limitada: 100 Endpoints y 2 informes
	Falta documentación online
	Interfaces poco homogéneas

Tabla 2.3: Ventajas y desventajas de SIEMonster (CE)

Prelude OSS

Ventajas	Desventajas
Probado en el tiempo: desarrollado desde 1998	Mucho más limitado que otras soluciones open source en cuanto a rendimiento, características y escalabilidad
Soporta un amplio rango de formatos de logs	Destinado a pruebas, evaluaciones y pequeños escenarios
Normaliza datos al formato IDMEF ¹ (útil para intercambio con otros IDS)	

¹ Intrusion Detection Message Exchange Format

Tabla 2.4: Ventajas y desventajas de Prelude OSS

MozDef

Ventajas	Desventajas
Sin agente: logs JSON como entrada	Nuevo y menos estable que otras soluciones
Fácil escalado basado en el volumen de eventos	Sólo instalable via docker o en CentOS 7
Soporta orígenes de datos basados en la nube (AWS CloudTrail y GuardDuty)	Configuración y aprendizaje de uso elevados
Desarrollado por Mozilla, de confianza en el mundo open source	

Tabla 2.5: Ventajas y desventajas de MozDef

Wazuh

Ventajas	Desventajas
Basado y compatible con OSSEC	Requiere un despliegue de ELK Stack además del servidor Wazuh
Soporta despliegues Docker, Puppet, Chef y Ansible	
Soporta monitorización infraestructura en la nube (de pago) (AWS y Azure)	
Conjunto de reglas que detecta muchos tipos de ataques comunes	
Cumplimiento con PCI DSS v3.1 y CIS	
Detección de vulnerabilidades	
Arquitectura centralizada o distribuida	
Excelente documentación online actualizada	

Tabla 2.6: Ventajas y desventajas de Wazuh

En nuestro caso, elegimos Wazuh + ELK Stack por varias razones:

- Es open source y sólo se pagaría por soporte o monitorización en la nube pero no los necesitamos ya que buscamos una solución **on-premise**.
- No es una versión limitada (como sucede con OSSIM, Prelude OSS o SIEMonster). Posee todas las características.
- Está basado en un HIDS de reconocido prestigio, como es OSSEC.
- Documentación online actualizada y muy completa.
- Fácilmente escalable mediante clúster, tanto Wazuh como ELK Stack.
- Sobre todo, cumple todos los requisitos indicados en la sección 2.3, [Requisitos del SIEM a implantar](#).

2.5. Análisis del SIEM elegido: Wazuh + Elk Stack

2.5.1. Capacidades de Wazuh

Wazuh proporciona las siguientes capacidades:

Análisis de seguridad Wazuh se utiliza para recolectar, agregar, indexar y analizar datos de seguridad, ayudando a las organizaciones a detectar intrusiones, amenazas y comportamientos anómalos.

Detección de intrusiones Los agentes de Wazuh escanean los sistemas monitorizados buscando malware, rootkits y anomalías sospechosas. Pueden detectar ficheros y procesos ocultos, puertos de red a la escucha no registrados e inconsistencias en las respuestas a las llamadas del sistema. Además de estas capacidades de los agentes, el servidor usa firmas para detectar intrusiones, usando su motor de expresiones regulares para analizar los datos de los logs almacenados y buscar indicadores de anomalías.

Análisis de los datos de logs Los agentes de Wazuh leen los logs de las aplicaciones y del sistema operativo y los envían de forma segura al servidor central para almacenarlos y realizar un análisis basado en reglas. Dichas reglas ayudan a tener conocimiento de errores del sistema o de aplicación, fallos de configuración, intentos y/o éxito de actividades maliciosas, violaciones de la política de seguridad y otra serie de problemas de operación o seguridad.

Monitorización de la integridad de ficheros Wazuh monitoriza el sistema de ficheros identificando cambios en el contenido, los permisos, la propiedad y los atributos de los ficheros que le indiquemos. Además, identifica de forma nativa los usuarios y aplicaciones usados para crear o modificar ficheros.

Detección de vulnerabilidades Los agentes de Wazuh envían los datos del inventario de software al servidor donde se correlaciona con las bases de datos [CVE⁴](#) continuamente actualizadas, para identificar el software vulnerable conocido. La evaluación de las vulnerabilidades de forma automática nos ayuda a encontrar los puntos vulnerables en los activos críticos y tomar las acciones correctivas necesarias antes de que sean explotadas por los atacantes.

Evaluación de la configuración Wazuh monitoriza el sistema y la configuración de las aplicaciones para asegurar que cumplen con las políticas de seguridad, estándares o

⁴Common Vulnerabilities and Exposure

guías de bastionado. Los agentes llevan a cabo escaneos periódicos para detectar las aplicaciones que se conoce que son vulnerables, no parcheadas o configuradas de forma insegura. Además, los chequeos pueden personalizarse y hacerse a medida para adecuarse a nuestra organización.

Respuesta a incidentes Wazuh proporciona una serie de respuestas ya preparadas para usar, que llevan a cabo contramedidas que gestionan amenazas activas, tales como bloquear el acceso a un sistema del origen de la amenaza cuando se cumple cierto criterio. Además, Wazuh puede usarse para ejecutar comandos remotamente o consultas al sistema, identificando **indicadores de compromiso (IOCs)** y ayudando a otras actividades forenses o tareas de respuesta a incidentes.

Cumplimiento normativo Wazuh proporciona algunos de los controles de seguridad necesarios para cumplir con los estándares de la industria y la normativa. Wazuh puede utilizarse para cumplir con los requisitos **PCI DSS**, **GPG13**⁵ o **GDPR**⁶, utilizando su interfaz de usuario web que proporciona informes y paneles de control (dashboards) que pueden ayudarnos a dicho cumplimiento.

Monitorización de la seguridad en la nube Wazuh ayuda a la monitorización de la infraestructura en la nube de proveedores como Amazon AWS, Azure o Google Cloud.

Seguridad en contenedores Wazuh también proporciona visibilidad de la seguridad en los equipos y contenedores Docker, monitorizando su comportamiento y detectando amenazas, vulnerabilidades y anomalías.

En nuestro caso, al ser una instalación **on-premise**, y no tener ni equipos ni contenedores Docker, no haremos uso de las dos últimas características mencionadas.

2.5.2. Componentes de Wazuh

Wazuh es una solución integral que consta de tres componentes principales: OSSEC HIDS, OpenSCAP y Elastic Stack.

2.5.2.1. OSSEC HIDS

OSSEC HIDS es un sistema de detección de intrusos **HIDS** que se usa para la detección, visibilidad y monitorización del cumplimiento de eventos de seguridad. Está basado en un agente multiplataforma que envía datos del sistema (mensajes de logs, hashes de ficheros, anomalías detectadas) a un gestor central, donde posteriormente se analiza y procesa, dando como resultado las alertas de seguridad. Los agentes envían su información a través de canales seguros y autenticados.

Además, OSSEC HIDS proporciona un servidor syslog centralizado y sistemas de monitorización de la configuración sin agente que nos permite detectar eventos de seguridad y cambios en dispositivos tales como firewalls, routers, switches, etc. . .

⁵Guía 13 de Buenas Prácticas

⁶Reglamento General de Protección de Datos

2.5.2.2. OpenSCAP

OpenSCAP es un intérprete de [OVAL](#)⁷ y [XCCDF](#)⁸ que se usa para chequear las configuraciones del sistema y para detectar aplicaciones vulnerables. Es una herramienta diseñada para chequear el cumplimiento de seguridad y el bastionado de los sistemas al usar guías de seguridad estándar en la industria para entornos empresariales.

2.5.2.3. Elastic Stack

Elastic Stack es un conjunto de software (Filebeat, Elasticsearch, Kibana) que se usa para recolectar, comparar, indexar, almacenar, buscar y presentar datos de logs. Proporciona un entorno web que da una vista a un alto nivel mediante paneles de control (dashboards) de los eventos que nos permite realizar análisis avanzados y minería de datos.

2.5.3. Arquitectura de Wazuh

Wazuh tiene dos componentes centrales principales a instalar: el manager de Wazuh y Elastic Stack. El tipo de instalación hace que Wazuh permite dos tipos de arquitectura:

Arquitectura centralizada: se ejecutan el servidor de Wazuh y Elastic Stack en el mismo servidor.

Arquitectura distribuida: se ejecuta el servidor de Wazuh y el cluster de Elastic Stack (uno o más servidores) en diferentes sistemas.

En la figura 2.3 podemos ver la arquitectura en un único host y en la figura 2.4 podemos ver la arquitectura distribuida.

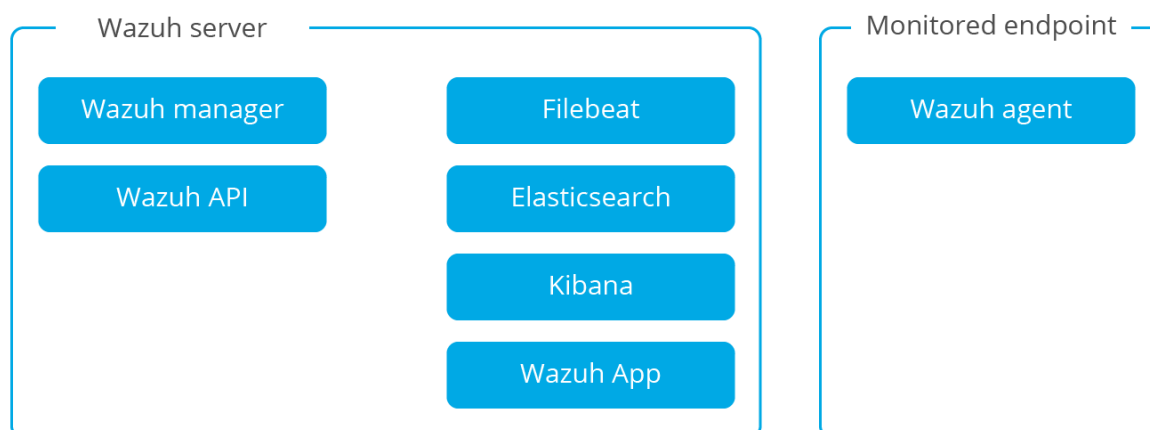


Figura 2.3: Arquitectura de Wazuh centralizada

La decisión sobre qué tipo de arquitectura conviene a nuestra organización se tomará en el capítulo 3 de [Diseño y toma de decisiones](#).

⁷Open Vulnerability Assesment Language

⁸eXtensible Configuration Checklist Description Format

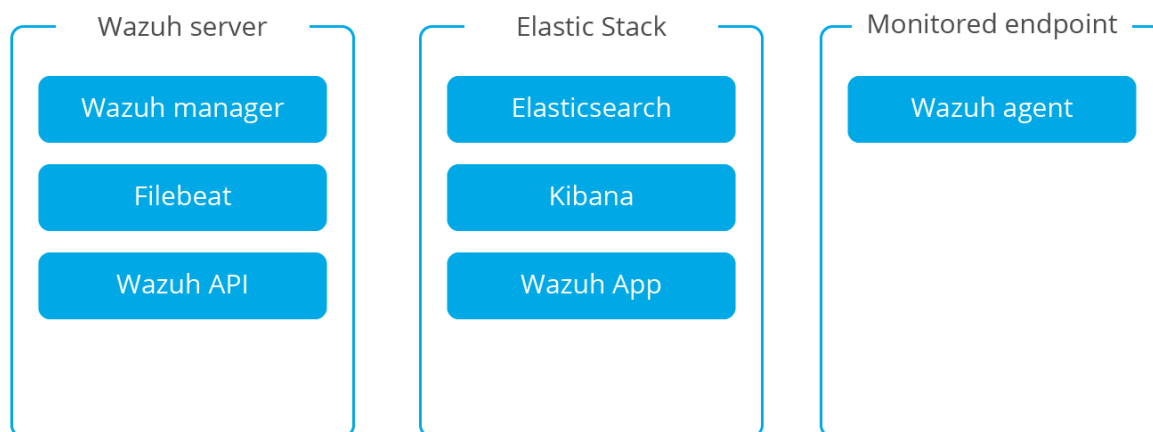


Figura 2.4: Arquitectura de Wazuh distribuida

2.5.3.1. Escalabilidad y disponibilidad: Cluster Wazuh

Wazuh también permite configurar un cluster cuando necesitamos incrementar la capacidad de procesamiento debido a que tenemos miles de agentes en el sistema informando al manager de Wazuh. En este caso, existiría un nodo central (Master) y varios nodos trabajadores (workers) a los que se les distribuiría la carga de los agentes. La arquitectura sería la que se muestra en la figura 2.5, con un balanceador de carga delante del cluster de Wazuh que enviaría los informes de los agentes a uno u otro worker. Esto también nos da disponibilidad porque si algún worker se cae, otro puede reemplazarlo. Lo que no cubriría este esquema es una caída del nodo maestro (Master), aunque es una característica (tener múltiples masters y alta disponibilidad (HA⁹)) que tienen pendiente de implementar los desarrolladores de Wazuh, como se puede ver en el hilo de github [Cluster:Multiple masters #811](#).

Al igual que para la arquitectura a utilizar vista en el apartado 2.5.3, la decisión sobre si implementar un cluster de Wazuh o no en nuestra organización se tomará en la fase de [Diseño y toma de decisiones](#), en base al esquema de red de nuestra organización y el número de agentes a desplegar.

⁹High Availability

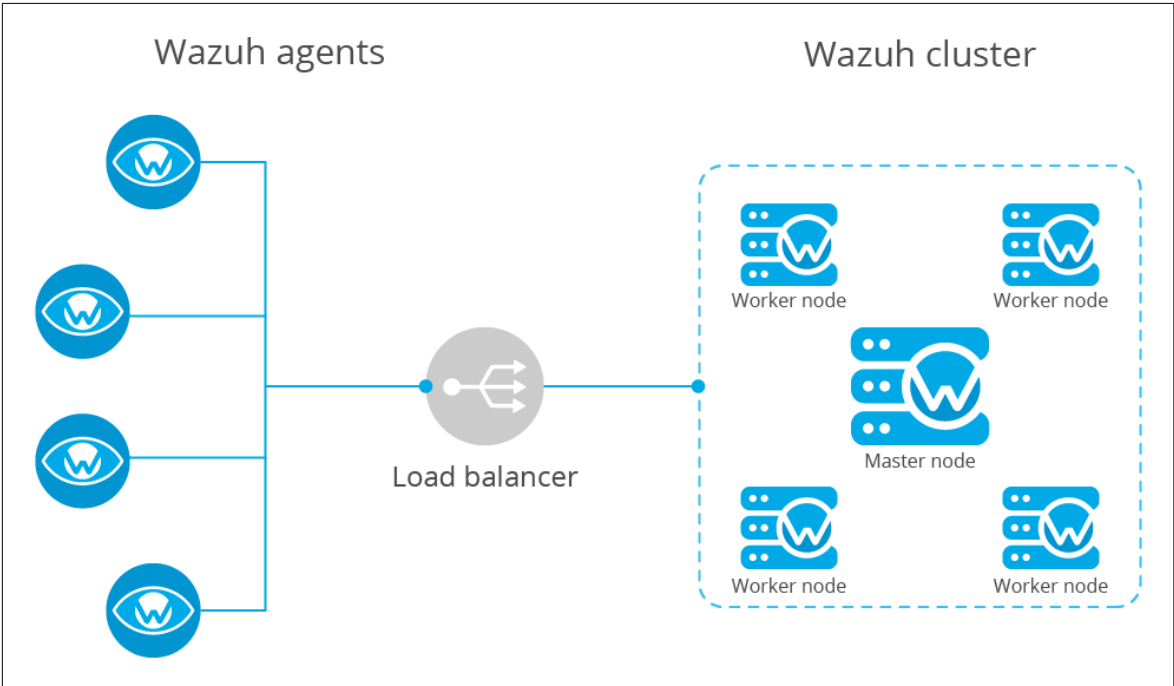


Figura 2.5: Cluster Wazuh



3

DISEÑO Y TOMA DE DECISIONES

3.1. Arquitectura de red de la organización

La organización en nuestra Delegación Provincial consta de dos sedes entre las cuales existe visibilidad a nivel de red:

1. La sede principal, con direccionamiento 10 . x . x . x /23.
2. La sede remota de formación, con direccionamiento 10 . x . x . x /24.

Asimismo, existen dos tipos de usuarios con acceso remoto mediante conexión por VPN¹ utilizando doble factor de autenticación: certificado digital más usuario y contraseña corporativos. Son:

- Los usuarios que solo tienen acceso a su equipo de trabajo a través de Escritorio Remoto usando el protocolo RDP².
- Los usuarios que tienen acceso a toda el direccionamiento de la sede (determinados usuarios del Departamento de Informática).

El esquema de red de la sede principal se muestra en la figura 3.1 y el de la sede de formación se encuentra en la figura 3.2.

Por seguridad, se han anonimizado todos los direccionamientos en el TFM.

¹Virtual Private Network

²Remote Desktop Protocol

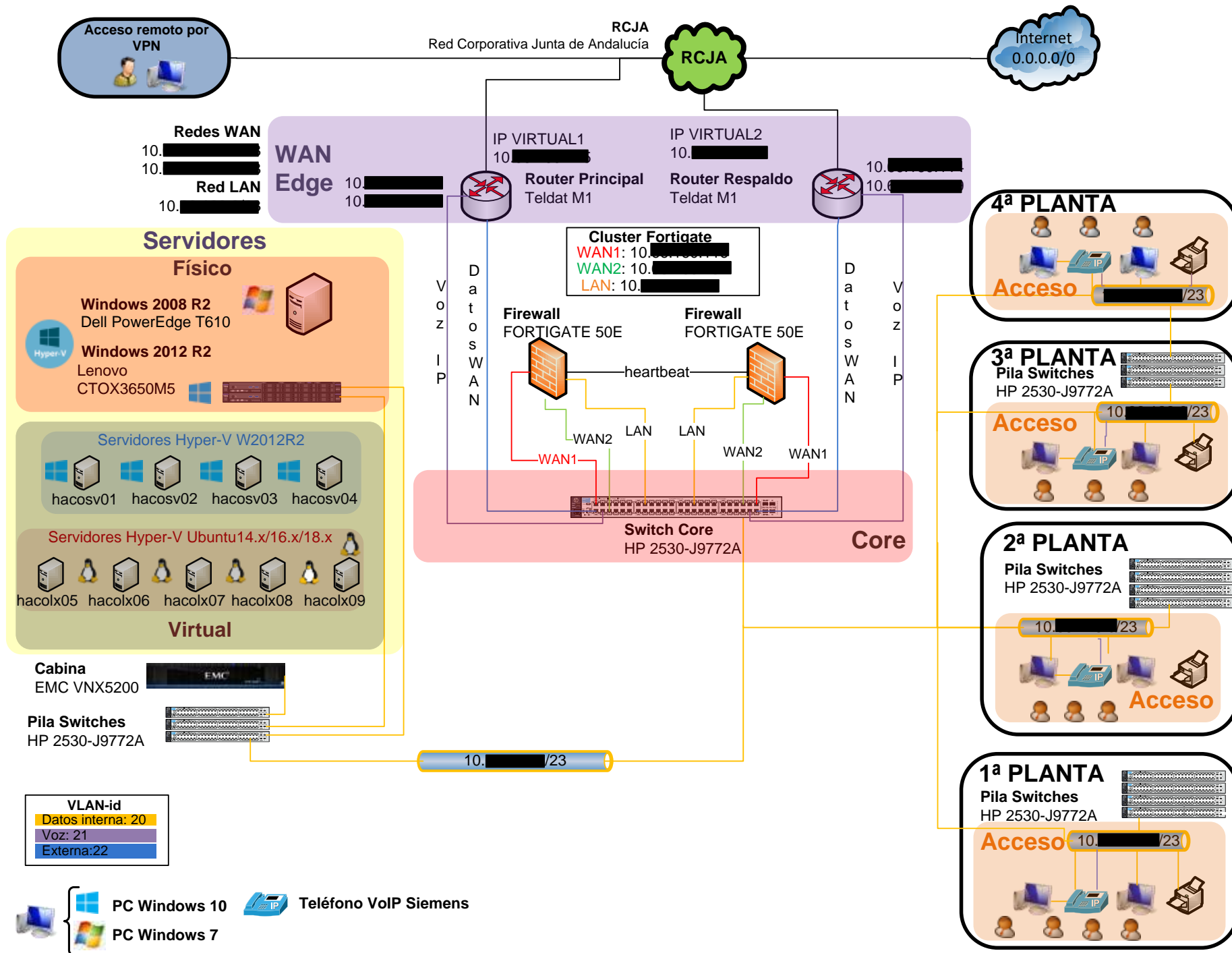


Figura 3.1: Esquema de red sede principal

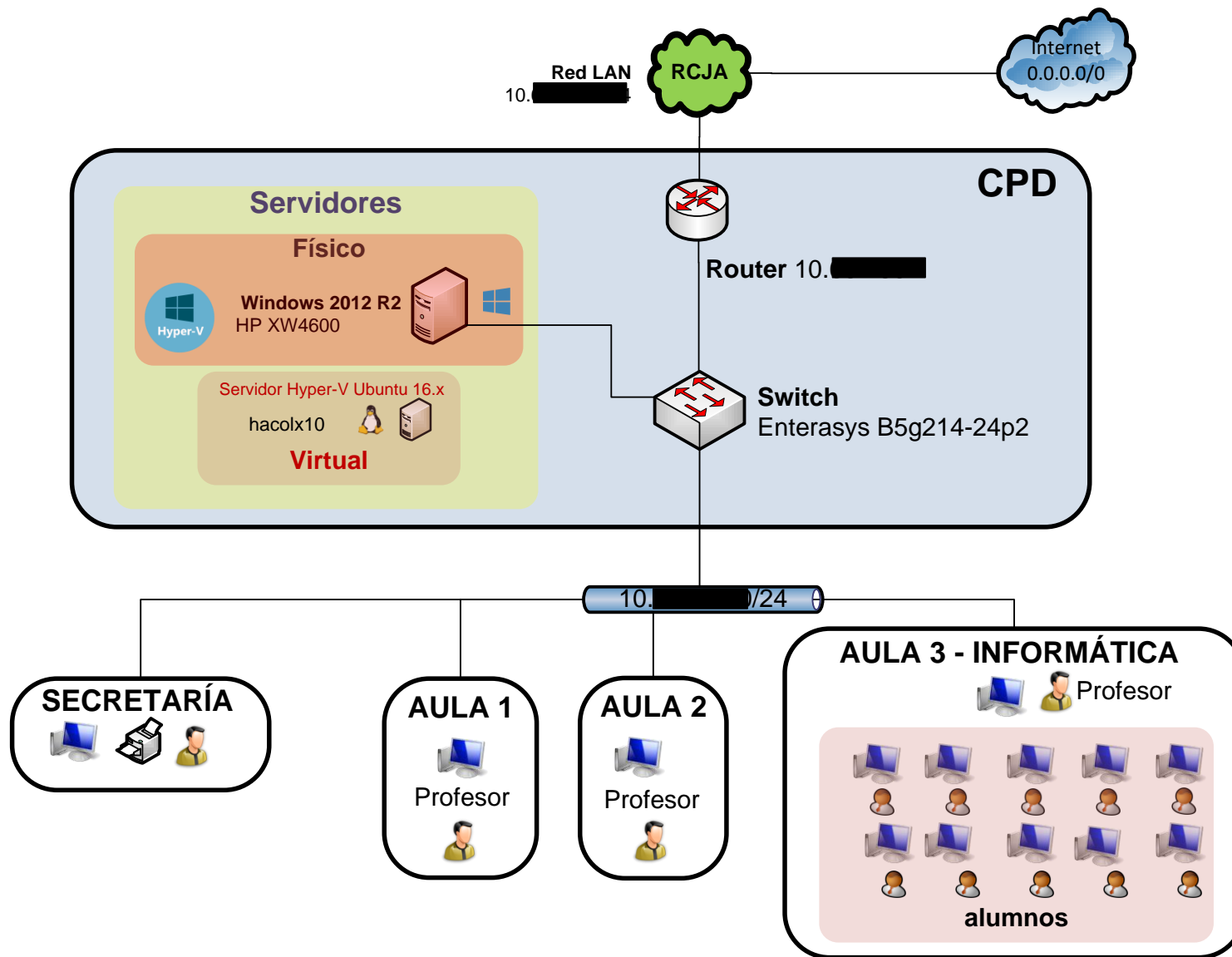


Figura 3.2: Esquema de red sede formación

En base a estos esquemas de red, se tomará la decisión de cómo implantar Wazuh en nuestra organización.

3.2. Elección de implantación: virtual, física o contenedor

Para implantar Wazuh existen 3 posibilidades: implantación en una máquina física, en una virtual o en un contenedor. Las analizamos en los siguientes apartados.

3.2.1. Implantación en una máquina física

En nuestra infraestructura no contamos con servidores físicos que puedan manejar los requisitos recomendados por Wazuh en el hilo [Requisitos Wazuh](#). En concreto se recomiendan:

- **Wazuh manager:** 4 núcleos, 16 GB RAM y 1 TB de espacio en disco por nodo.
- **Elastic Stack:** 8 núcleos, 32 GB de RAM mínimo y 64 GB máximo, 1 TB mínimo de espacio en disco por nodo (aunque esto último dependerá de los datos almacenados en Elastic).

Estos requisitos recomendados hacen inviable para nosotros desplegarlo en un servidor físico.

3.2.2. Implantación en un contenedor Docker

Existe un contenedor Docker para Wazuh que utiliza [Docker Compose](#), que tiene a su vez 4 contenedores:

1. Wazuh
2. Elasticsearch
3. Kibana
4. Nginx: es el servidor web proxy inverso que se utilizar para acceder a kibana de forma encriptada con usuario y contraseña.

En un principio se ha intentado utilizar Wazuh con Docker en Windows mediante la tecnología [Hyper-V](#) en un PC (que es la que se utiliza en los servidores de nuestra organización) a través de [Docker Desktop](#). Sin embargo, nos daba errores al iniciarlo y creamos un hilo en el foro de Wazuh para hacer la consulta al respecto. Allí nos indicaron que existen limitaciones en la virtualización al utilizar contenedores Linux en Windows. (Ver <https://github.com/wazuh/wazuh-docker/issues/292> para más información).

“Our Docker images are based on Ubuntu 16.04 LTS so executing them on Windows is not possible due to Virtualization limitations”.

Por otra parte, sería posible utilizar Wazuh con Docker en Windows mediante la tecnología de [Oracle VirtualBox](#) a través de [Docker Toolbox](#), pero se utilizarían versiones antiguas de VirtualBox (versión 5.2.0) y, realmente estaríamos utilizando una máquina virtual dentro de VirtualBox. Para eso es más sencillo y eficiente utilizar directamente VirtualBox con la imagen .ova que tiene preparada Wazuh.

Todo lo anterior hace que desechemos utilizar contenedores Docker ya que nuestra infraestructura trabaja con servidores Windows 2012 R2 con tecnología Hyper-V para la virtualización, lo cual no es soportado actualmente por los contenedores Linux en general y de Wazuh en particular.

3.2.3. Implantación de una imagen virtual

En esta opción instalaríamos Wazuh en una máquina virtual. En este caso, existen dos posibilidades:

1. Utilizar una versión precompilada, proporcionada por Wazuh. Aquí tenemos, a su vez, tres opciones de virtualización: [VirtualBox](#), [VMware](#) e [Hyper-V](#):
 - a) **VirtualBox:** Utilizar la versión precompilada de Wazuh para VirtualBox (extensión `.ova`) que en el momento de realizar este apartado contiene los elementos definidos en la sección [3.2.3.1](#).

3.2.3.1. Elementos de la versión precompilada

- CentOS 7
- Wazuh 3.11.4
- Wazuh API 3.11.4
- Elasticsearch 7.6.1
- Filebeat 7.6.1
- Kibana 7.6.1
- Wazuh app 3.11.4-7.6.1

Además con las características de máquina virtual indicada en la sección siguiente ([3.2.3.2](#)).

3.2.3.2. Características máquina virtual precompilada

- 4 GB RAM
 - RedHat 64 bits
 - 4 procesadores
 - 16 MB Pantalla RAM
 - 40 GB disco duro (dinámico)
- b) **VMware:** Utilizar la versión precompilada de Wazuh para [VMware](#) (extensión `.ovf+.vmdk`) con el mismo contenido anterior.
 - c) **Hyper-V:** No existe imagen precompilada de Wazuh para Hyper-V (extensión `.vhdx`) pero existen tutoriales en Internet para convertir imágenes de VMware a Hyper-V. Es un proceso que no es trivial y dependemos de terceras herramientas como [Microsoft Virtual Machine Converter](#), [dsfok tools](#) y scripts de Powershell. Hemos tenido que solucionar algunos errores pero ha sido posible importar la imagen vmdk a un PC de pruebas con Hyper-V siguiendo los enlaces siguientes:
 - [Convertir VMware a Hyper-V](#)
 - [Errores al convertir VMware a Hyper-V](#).

Desechamos las dos primeras opciones porque en nuestra organización no utilizamos ni la tecnología de VirtualBox ni de VMware, sino que utilizamos Hyper-V.

2. Instalar desde cero Wazuh + ELK Stack en una o varias máquinas virtuales en nuestro entorno.

3.2.4. Método de implantación seleccionado

La decisión tomada ha sido la de instalar desde cero Wazuh + ELK Stack en una o varias máquinas virtuales en nuestros servidores con tecnología Hyper-V, por las siguientes razones:

1. Dependemos de la infraestructura actual de nuestra organización basada en la tecnología de virtualización Hyper-V.
2. Tendremos alta disponibilidad, al tener dos nodos físicos y un cluster virtual activo-activo.
3. El hardware utilizado (2 servidores Lenovo CTOX3650M5, Intel Xeon CPU E5-2640 v4 (con 2 procesadores c/u³), con 512 GB RAM c/u y 15 TB de espacio de almacenamiento compartido en la cabina EMC VNX5200) es el mejor del que disponemos.
4. Las imágenes precompiladas vienen bien para tener una idea general del sistema y una puesta en marcha rápida, pero para conocerlo a fondo y dominarlo, es mejor realizar una instalación propia desde cero.
5. Nuestros servidores Linux son Ubuntu Server y tenemos un mejor conocimiento de dichos sistemas (.deb) que del sistema CentOS (.rpm) que se monta en las versiones precompiladas.
6. No existe imagen precompilada para Hyper-V de Wazuh como hemos visto en el apartado 3.2.3.2 y, aunque puede obtenerse con varios [workarounds](#), tampoco sería una imagen “limpia” suministrada por Wazuh, lo que podría ocasionarnos problemas. Además, necesitaríamos invertir un tiempo en la conversión desde la imagen de VMware a Hyper-V que es claramente compensado con el necesario para instalar desde cero Wazuh + ELK Stack.

La o las máquinas virtuales que se creen, (depende de la arquitectura utilizada, que se decidirá en la sección 3.3), deberán tener como requisitos mínimos los indicados como características en el apartado 3.2.3.2.

3.3. Arquitectura de red: centralizada o distribuida

Como vimos en el apartado 2.5.3, Wazuh puede instalarse con una arquitectura centralizada (compartiendo el mismo servidor Wazuh y Elastic Stack) o distribuida (separando el servidor de Wazuh de la pila Elastic). En nuestro caso, ya tenemos experiencia con uno de los componentes de la pila de Elastic, en concreto con Elasticsearch, al usarlo como buscador en un Mediawiki de la organización. Dicho componente consume bastantes recursos (en especial memoria) y consideramos que es preferible que esté separado del servidor de Wazuh para tener una monitorización individual de ambas máquinas y, si es necesario, poder modificar sus recursos y/o configuración sin que se vea afectada la otra. Por tanto, en principio se optará por instalar el servidor de Wazuh en una máquina virtual y Elastic Stack en otra.

Por otra parte, otro aspecto a considerar en cuanto a la arquitectura es el tema de la escalabilidad y disponibilidad utilizando un cluster Wazuh, como vimos en el apartado 2.5.3.1. En cuanto a la escalabilidad, en nuestro caso y analizando los esquemas de red de nuestra organización (figuras 3.1 y 3.2), el número de [endpoints](#) a desplegar no será superior a 50. Este

³cada uno

número de endpoints es de sobra manejado sin necesidad de utilizar el cluster de Wazuh (con un maestro y varios trabajadores, además de un balanceador). Su utilización es recomendable y necesaria cuando vamos a desplegar cientos o miles de endpoints, que no es nuestro caso.

En cuanto a la disponibilidad, hay que destacar que el cluster de Wazuh no proporciona alta disponibilidad, ya que si cae el Master se cae el sistema, como vimos en el apartado 2.5.3.1. Por lo tanto, no es una razón para desplegarlo. En nuestro caso, la alta disponibilidad quedará cubierta ya que instalaremos las máquinas virtuales en producción en el cluster de conmutación por error que tenemos desplegado en los servidores Windows 2012 R2, para que, en el caso de que se caiga uno de los dos nodos físicos (servidores Lenovo), se pasen al otro las máquinas virtuales y no se caiga el sistema.

3.4. Diseño final de implantación y arquitectura

La implantación final del servidor Wazuh será el despliegue de dos máquinas virtuales, una para Wazuh Server y otra para Elastic Stack en el cluster de conmutación por error de Windows 2012 R2. Ambas contarán con Ubuntu Server en su última versión estable (**LTS**), instaladas desde cero y teniendo como requisitos mínimos los utilizados por las imágenes precompiladas que ofrece Wazuh y que vimos en la sección 3.2.3.2.

La arquitectura final de los servidores físicos y virtuales de las sedes con Wazuh implementado será la mostrada en la figura 3.3 con independencia de que los agentes finalmente sean desplegados en más endpoints (por ejemplo en los PCs de los profesores del aula de formación o en los de los alumnos) o que se monitoricen sin agente (**agentless**) otros dispositivos (por ejemplo los switches) mediante el envío de los logs al servidor de syslog de Wazuh.

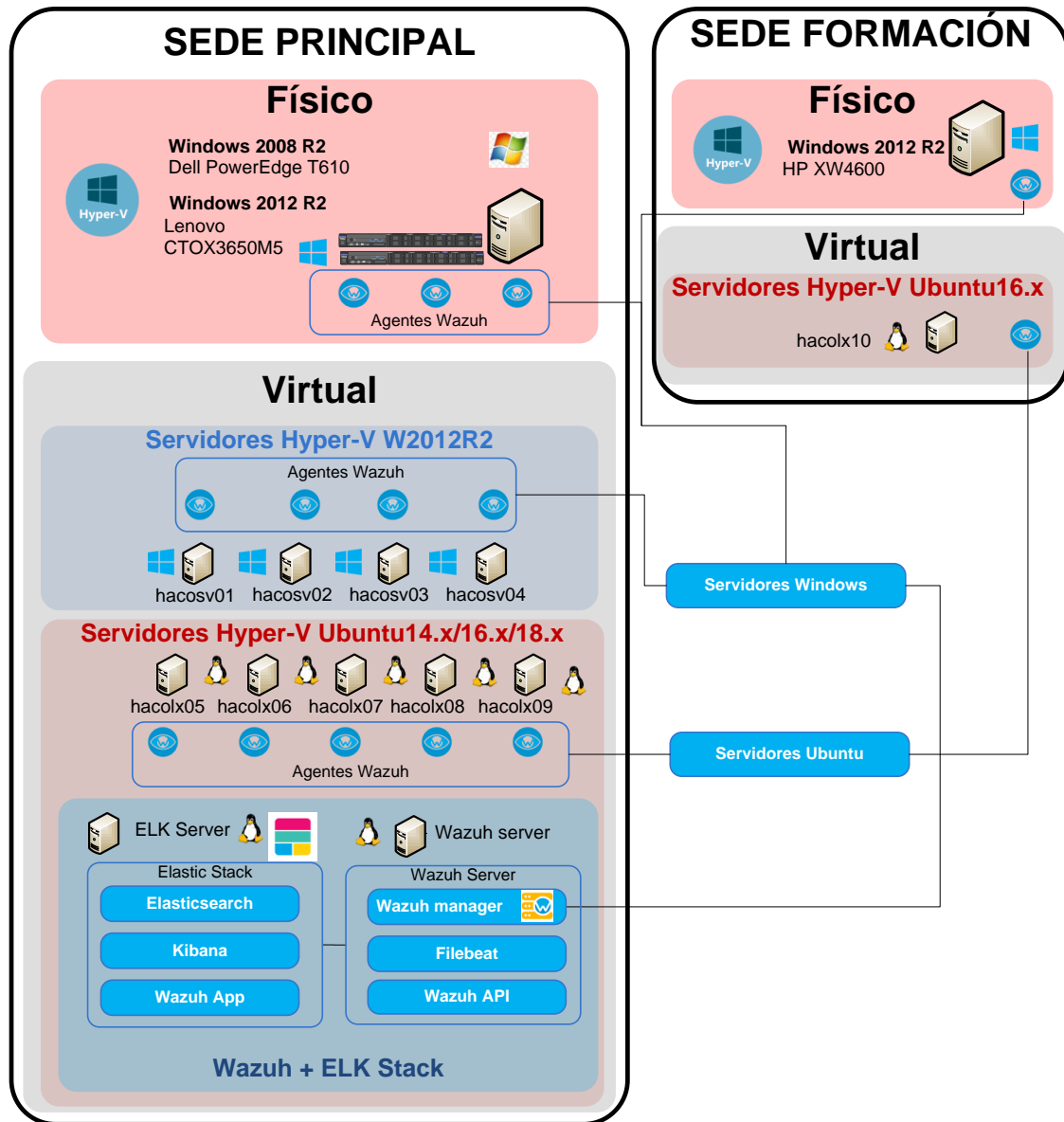


Figura 3.3: Diseño final de implantación y arquitectura Wazuh+ELK Stack



4

INSTALACIÓN

4.1. Instalación en pruebas del SIEM

La instalación en pruebas de Wazuh y de ELK Stack se realizará en mi equipo de trabajo, que tiene las siguientes características:

- Windows 10 Pro 64 bits (admite Hyper-V, a diferencia de la versión Home que no)
- HP 400 g5, i5-8500 3 Ghz con 8 GB de RAM
- Disco duro SSD 256 GB

Se han creado dos máquinas virtuales con Hyper-V, llamadas *wazuh-server-pruebas* y *wazuh-elk-pruebas* como se indica en el anexo A. A dichas máquinas, se les ha instalado Ubuntu Server 18.04.4 LTS, siguiendo las instrucciones en el anexo B.

Una vez tenemos las dos máquinas virtuales operativas, actualizadas y con una IP fija visible por el resto de equipos de la organización, podemos comenzar con la instalación. Al estar en pruebas, iremos realizando *snapshots*¹ de las máquinas virtuales por si tenemos que regresar a un punto anterior.

4.1.1. Instalación de Wazuh Server

Para la instalación seguiremos la documentación online ([Wazuh, 2020](#)). En primer lugar, para Ubuntu Server tenemos dos opciones:

1. Instalar desde los repositorios
2. Instalar desde código fuente

¹instantáneas

En principio para una instalación en pruebas y viendo que la versión de los repositorios está actualizada y es la misma que si la instalamos desde código fuente, optamos por la primera opción que es más sencilla y rápida. Dicha instalación está documentada en el anexo C.

El siguiente paso sería instalar la pila de Elastic.

4.1.2. Instalación del servidor ELK Stack

Al igual que para la [Instalación de Wazuh Server](#), tenemos la opción de instalar el servidor de Elastic Stack desde los paquetes precompilados o desde los binarios en formato `tar.gz`, pero se nos indica en la documentación que esta segunda opción no tiene soporte y no es la recomendada. Además de los componentes de la pila Elastic, es necesario instalar la aplicación de Wazuh (a través de un plugin de Kibana). Dicha instalación está documentada en el anexo D.

4.1.3. Actualización de la versión de Wazuh y de ELK

Durante el desarrollo del trabajo fin de Máster, se ha actualizado la versión de Wazuh, pasando de la 3.11.4 a la 3.12.0-2, así como de la pila ELK, pasando de la 7.6.1 a 7.6.2. Esto se ha documentado en el anexo C en la sección C.2 para Wazuh Server y Wazuh API, en el anexo D en la sección D.5 para la actualización de Kibana y de Elasticsearch y en la sección D.6 para el plugin de Wazuh para Kibana.

4.1.4. Problemas de seguridad en una instalación por defecto

Podemos observar que la instalación inicial presenta varios problemas de seguridad:

1. No existe autenticación ni encriptación para acceder al portal de Kibana.
2. No existe una comunicación encriptada entre el servidor Wazuh (Wazuh API) y el plugin de Wazuh para Kibana.
3. Se están usando usuarios por defecto (usuario:foo y contraseña: bar) para acceder a la API de Wazuh.
4. El servidor de Elasticsearch no está securizado.

Debido a que para securizar el sistema tenemos que generar certificados a través de nuestra CA², establecer FQDNs para los servidores en el servidor de nombres, etc. . . , esto ya lo haremos en producción.

Por otra parte, hemos comprobado que la instalación inicial del servidor Wazuh (Ubuntu+Wazuh Server) ocupa < 3 GB y la del servidor con ELK Stack ocupa < 8 GB. Dicho espacio irá creciendo paulatinamente debido a los logs, eventos, etc. . . pero nos da una idea del tamaño inicial adecuado para las máquinas virtuales. También hemos observado que la máquina de Elastic consume toda la memoria suministrada y quizás en producción debamos aumentarla.

²Autoridad Certificadora

4.2. Instalación en producción del SIEM securizado

Para la instalación en producción seguimos los mismos pasos que para la instalación en pruebas que vimos en la sección 4.1 solamente cambiando las IPs, nombres y características de las máquinas virtuales (como se indica en los Anexos A y B).

Una vez instaladas, se han añadido ambas al clúster de Windows 2012 R2 de los dos servidores físicos, para que no haya falta de disponibilidad si alguno de los dos servidores físicos no está levantado (parada por mantenimiento, reinicios programados por actualizaciones del sistema, etc. . .). Esto está documentado en la sección A.3 en el anexo A.

Además, se han creado los registros necesarios en el DNS para que puedan obtenerse los certificados digitales necesarios en base a su FQDN para securizar la aplicación y las comunicaciones. Esto se ha documentado en la sección B.2.3 del anexo B.

Para solucionar los problemas encontrados en la sección 4.1.4, debemos realizar 3 acciones:

1. Securizar la API de Wazuh
2. Securizar Elastic Stack
3. Dotar de autenticación a Elastic Stack

Esto lo vemos en las secciones siguientes.

4.2.1. Securizar la API de Wazuh

4.2.1.1. Activar https en la API de Wazuh

En primer lugar, debemos activar https. Para ello, tenemos dos opciones: generar nuestro propio certificado de forma manual o automáticamente con un script que proporciona Wazuh en su documentación online.

En nuestro caso, ya tenemos creada una CA en la organización, instalada como *Entidad de certificación raíz de confianza* en los clientes y que nos permite generar certificados para nuestros servidores. Por tanto, usaremos dicha autoridad de certificación para generar el certificado.

El certificado para el servidor wazuh-server (utilizado tanto para securizar la API de Wazuh como Filebeat), se ha generado en el anexo E en la sección E.1. Consta de dos ficheros: una clave privada (wazuh-server.key) y una clave pública (wazuh-server.crt).

Para securizar la API de Wazuh, debemos copiar los ficheros donde nos indica el de configuración /var/ossec/api/configuration/config.js, el cual dejamos así:

```
// Use HTTP protocol over TLS/SSL. Values: yes, no.
config.https = "yes";

// HTTPS Certificates
config.https_key = "configuration/ssl/wazuh-server.key"
config.https_cert = "configuration/ssl/wazuh-server.crt"
config.https_use_ca = "yes"
config.https_ca = "configuration/ssl/rootCA.pem"
```

Ahora copiamos los ficheros de los certificados de Wazuh y de la CA:

```
root@wazuh-server:~# cp wazuh-server.key /var/ossec/api/configuration/ssl/
root@wazuh-server:~# cp wazuh-server.crt /var/ossec/api/configuration/ssl/
root@wazuh-server:~# cp rootCA.pem /var/ossec/api/configuration/ssl/
```

Y reiniciamos el servicio:

```
root@wazuh-server:~# systemctl restart wazuh-api
```

Ahora nos debemos ir al servidor ELK y decirle al plugin de Wazuh de Kibana que vamos a usar https en lugar de http. Editamos el fichero: /usr/share/kibana/optimize/wazuh/config/wazuh.yml y cambiamos a https lo que estaba en http:

```
hosts:
- default:
  url: https://10.x.x.57
  port: 55000
  user: foo
  password: bar
```

4.2.1.2. Cambiar las credenciales por defecto

Por defecto, las credenciales de conexión para la API de Wazuh son foo/bar. Vamos a cambiarlas de forma manual (también sería posible mediante un script de forma automática).

```
root@wazuh-server:~# cd /var/ossec/api/configuration/auth
root@wazuh-server:/var/ossec/api/configuration/auth# node htpasswd -Bc -C 10 user WazuhCor
New password:
Re-type new password:
Adding password for user WazuhCor.
```

Ya tenemos el nuevo usuario creado llamado WazuhCor con la contraseña correspondiente.

A continuación reiniciamos el servicio de la API de Wazuh.

```
root@wazuh-server:~# systemctl restart wazuh-api
```

Luego nos iríamos al servidor ELK y editaríamos el fichero de configuración del plugin de Wazuh para Kibana, /usr/share/kibana/optimize/wazuh/config/wazuh.yml y pondríamos el usuario WazuhCor y la nueva contraseña:

```
hosts:
- default:
  url: https://10.x.x.57
  port: 55000
  user: WazuhCor
  password: <contraseña>
```


4.2.1.3. Cambiar el puerto por defecto

Por defecto el puerto que usa la API de Wazuh es el 55000. Es recomendable cambiarlo. Para ello, editamos el fichero `/var/ossec/api/configuration/config.js` y lo cambiamos al 55999 por ejemplo.

```
// TCP Port used by the API.
config.port = "55999";
```

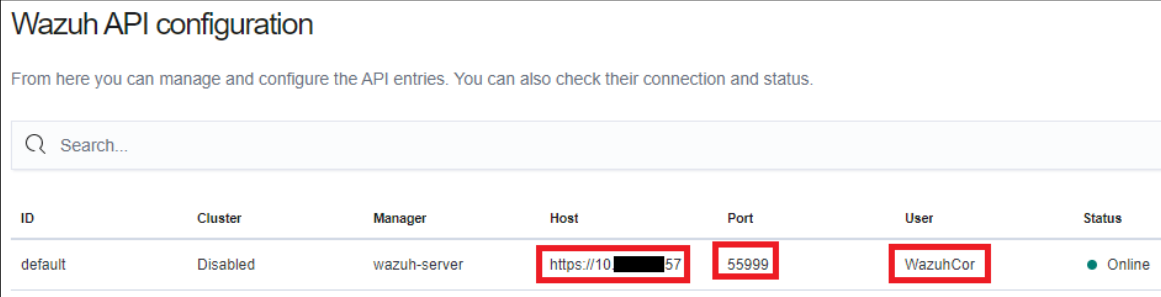
Luego lo modificamos también en el plugin de Wazuh para Kibana, en el fichero `/usr/share/kibana/optimize/wazuh/config/wazuh.yml`

```
hosts:
- default:
url: https://10.x.x.57
port: 55999
user: WazuhCor
password: <contraseña>
```

Y reiniciamos la API de Wazuh:

```
root@wazuh-server:~# systemctl restart wazuh-api
```

Y ya deberíamos tener securizado la API de Wazuh utilizando tráfico `https`, en un puerto distinto al de por defecto y con un usuario/contraseña que no son los predeterminados, como vemos en la figura 4.1



Wazuh API configuration

From here you can manage and configure the API entries. You can also check their connection and status.

Search...

ID	Cluster	Manager	Host	Port	User	Status
default	Disabled	wazuh-server	https://10.x.x.57	55999	WazuhCor	● Online

Figura 4.1: Securización de Wazuh API

4.2.2. Securizar Elastic Stack

4.2.2.1. Por qué securizar Elastic Stack

Es muy recomendable securizar Elastic Stack para que no sucedan fugas de datos como las que se produjeron en la noticia https://elpais.com/tecnologia/2020/02/25/actualidad/1582642099_471012.html.

Fueron debidas a un servidor Elastic Search no securizado, como se detalla en <https://www.vpnmentor.com/blog/report-decathlon-leak/>

Para securizar Elastic Stack hay tres opciones:

1. X-Pack
2. Search Guard
3. Nginx como proxy inverso para Kibana (autenticación sencilla y con SSL)

4.2.2.2. X-Pack

Hemos elegido X-Pack porque añade autenticación y encriptación de forma nativa. Anteriormente era una característica por la que había que pagar (en ese caso se utilizaba Search Guard que no tenía coste asociado) pero a partir de la versión 7.1.0 la seguridad puede configurarse sin coste alguno. (<https://www.elastic.co/es/blog/security-for-elasticsearch-is-now-free>).

Para añadir autenticación y encriptación necesitamos una CA y certificados digitales para el servidor Wazuh (para Filebeat), Elasticsearch y Kibana. En nuestro caso, como Elasticsearch y Kibana están en el mismo servidor, solo serían necesarios dos certificados, uno por máquina virtual.

En la documentación de Wazuh (Wazuh, 2020), se utiliza una herramienta que genera todos los certificados necesarios, incluyendo una CA, llamada `elasticsearch-certutil`. Como nosotros ya tenemos una CA en nuestra organización que además se añade por directivas de grupo a los navegadores de los equipos y ya hemos generado el certificado del servidor Wazuh en el apartado 4.2.1, únicamente tendríamos que generar un certificado para el servidor Elk (`wazuh-elk`). Esto lo hemos realizado en el anexo E en el apartado E.2.

Una vez tenemos los certificados, los copiamos al sitio adecuado.

En nuestro caso tendremos los siguientes certificados:

- `rootCA.key`: clave privada del certificado CA de la organización
- `rootCA.pem`: clave pública del certificado CA de la organización
- `wazuh-server.key`: clave privada del certificado para el servidor Wazuh (utilizado en `wazuh-api` y `filebeat`)
- `wazuh-server.crt`: clave pública del certificado para el servidor Wazuh
- `wazuh-elk.key`: clave privada del certificado para el servidor ELK (utilizado en Elasticsearch y kibana)
- `wazh-elk.crt`: clave pública del certificado para el servidor ELK

Por tanto, en primer lugar copiamos la clave pública del certificado CA y la privada y pública del servidor ELK al directorio de Elasticsearch y configuramos los permisos:

```
root@wazuh-elk:~# mkdir /etc/elasticsearch/certs/ca -p
root@wazuh-elk:~# cp rootCA.pem /etc/elasticsearch/certs/ca
root@wazuh-elk:~# cp wazuh-elk.crt /etc/elasticsearch/certs
root@wazuh-elk:~# cp wazuh-elk.key /etc/elasticsearch/certs
root@wazuh-elk:~# chown -R elasticsearch: /etc/elasticsearch/certs
root@wazuh-elk:~# chmod -R 770 /etc/elasticsearch/certs
```

Añadimos las configuraciones para las capas de transporte y de http en el fichero `/etc/elasticsearch/elasticsearch.yml`:

```
xpack.security.transport.ssl.enabled: true
xpack.security.transport.ssl.verification_mode: certificate
xpack.security.transport.ssl.key: /etc/elasticsearch/certs/wazuh-elk.key
xpack.security.transport.ssl.certificate: /etc/elasticsearch/certs/wazuh-elk.crt
xpack.security.transport.ssl.certificate_authorities: [ "/etc/elasticsearch/certs/ca/rootCA.pem" ]

xpack.security.http.ssl.enabled: true
xpack.security.http.ssl.verification_mode: certificate
xpack.security.http.ssl.key: /etc/elasticsearch/certs/wazuh-elk.key
xpack.security.http.ssl.certificate: /etc/elasticsearch/certs/wazuh-elk.crt
xpack.security.http.ssl.certificate_authorities: [ "/etc/elasticsearch/certs/ca/rootCA.pem" ]
```

Reiniciamos Elasticsearch

```
root@wazuh-elk:~# systemctl restart elasticsearch
```

4.2.2.3. Configurar la instancia de Filebeat (Wazuh manager)

En la máquina del servidor de Wazuh (`wazuh-server`) copiamos los certificados necesarios y establecemos los permisos adecuados:

```
root@wazuh-elk:~# mkdir /etc/filebeat/certs/ca -p
root@wazuh-elk:~# cp wazuh-server.crt /etc/filebeat/certs
root@wazuh-elk:~# cp wazuh-server.key /etc/filebeat/certs
root@wazuh-elk:~# cp rootCA.pem /etc/filebeat/certs/ca
root@wazuh-elk:~# chmod 770 -R /etc/filebeat/certs
```

Añadimos la configuración necesaria en el fichero `/etc/filebeat/filebeat.yml`:

```
output.elasticsearch.hosts: ['10.x.x.40:9200']
output.elasticsearch.protocol: https
output.elasticsearch.ssl.certificate: "/etc/filebeat/certs/wazuh-server.crt"
output.elasticsearch.ssl.key: "/etc/filebeat/certs/wazuh-server.key"
output.elasticsearch.ssl.certificate_authorities: ["/etc/filebeat/certs/ca/rootCA.pem"]
```

Podemos probar Filebeat con el comando `"filebeat test output"` para ver que nos conectamos de forma segura y encriptada al servidor de Elasticsearch:

```
root@wazuh-server:~# filebeat test output
elasticsearch: https://10.x.x.40:9200...
parse url... OK
connection...
parse host... OK
dns lookup... OK
addresses: 10.x.x.40
dial up... OK
TLS...
security: server's certificate chain verification is enabled
handshake... OK
TLS version: TLSv1.3
dial up... OK
```

```
talk to server... OK
version: 7.6.2
```

Y por último reiniciamos Filebeat:

```
root@wazuh-server:~# systemctl restart filebeat
```

4.2.2.4. Configurar la instancia de Kibana

En la máquina con ELK Stack creamos el directorio `/etc/kibana/certs` y copiamos la autoridad de certificación y los certificados de `wazuh-elk` ahí.

```
root@wazuh-elk:~# mkdir /etc/kibana/certs/ca -p
root@wazuh-elk:~# cp rootCA.pem /etc/kibana/certs/ca
root@wazuh-elk:~# cp wazuh-elk.crt /etc/kibana/certs
root@wazuh-elk:~# cp wazuh-elk.key /etc/kibana/certs
root@wazuh-elk:~# chown -R kibana: /etc/kibana/certs
root@wazuh-elk:~# chmod -R 770 /etc/kibana/certs
```

Añadimos la configuración necesaria en `/etc/kibana/kibana.yml`

```
elasticsearch.hosts: ["https://10.x.x.40:9200"]
elasticsearch.ssl.certificateAuthorities: ["/etc/kibana/certs/ca/rootCA.pem"]
elasticsearch.ssl.certificate: "/etc/kibana/certs/wazuh-elk.crt"
elasticsearch.ssl.key: "/etc/kibana/certs/wazuh-elk.key"

server.ssl.enabled: true
server.ssl.certificate: "/etc/kibana/certs/wazuh-elk.crt"
server.ssl.key: "/etc/kibana/certs/wazuh-elk.key"
```

Reiniciamos Kibana

```
root@wazuh-elk:~# systemctl restart kibana
```

Una vez hecho esto ya tendríamos comunicación `https` entre todas las partes implicadas. A continuación, añadiríamos la autenticación.

4.2.3. Autenticación para Elastic Stack

Para añadir autenticación a Elastic Stack hacemos lo siguiente:

- Añadimos la línea siguiente al fichero `/etc/elasticsearch/elasticsearch.yml`

```
xpack.security.enabled: true
```

- Reiniciamos Elasticsearch y esperamos hasta que el servicio esté listo:

```
root@wazuh-elk:~# systemctl restart elasticsearch
```

- Generamos credenciales aleatorias para todos los usuarios y roles precargados en la pila Elastic con la utilidad `elasticsearch-setup-passwords`:

```
root@wazuh-elk:~# /usr/share/elasticsearch/bin/elasticsearch-setup-passwords auto
```

```
Initiating the setup of passwords for reserved users
-> elastic,apm_system,kibana,logstash_system,beats_system,remote_monitoring_user.
The passwords will be randomly generated and printed to the console.
Please confirm that you would like to continue [y/N]
```

- Anotamos la contraseña del usuario `elastic` y guardamos todas las demás en un lugar seguro.
- Establecemos las credenciales del usuario `elastic` en Filebeat. Para ello, editamos el fichero `/etc/filebeat/filebeat.yml` (en la máquina `wazuh-server`) y:

```
output.elasticsearch.username: "elastic"
output.elasticsearch.password: "password_generated_for_elastic"
```

- Reiniciamos Filebeat:

```
root@wazuh-elk:~# systemctl restart filebeat
```

- Establecemos también las credenciales para Kibana. Añadimos al fichero `/etc/kibana/kibana.yml`:

```
xpack.security.enabled: true
elasticsearch.username: "elastic"
elasticsearch.password: "password_generated_for_elastic"
```

- Reiniciamos Kibana:

```
root@wazuh-elk:~# systemctl restart kibana
```

Ya tendríamos securizadas todas las comunicaciones y la autenticación establecida, con lo que podríamos acceder al interfaz web para logarnos en la URL <https://wazuh-elk.<dominio>:5601> y se nos presentaría la imagen de la figura 4.3.

Como vemos, estamos usando conexión `https` con un certificado válido firmado por nuestra Autoridad Certificadora (Figura 4.2).

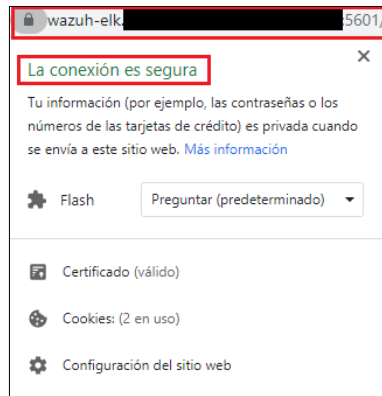


Figura 4.2: Conexión segura portal Kibana

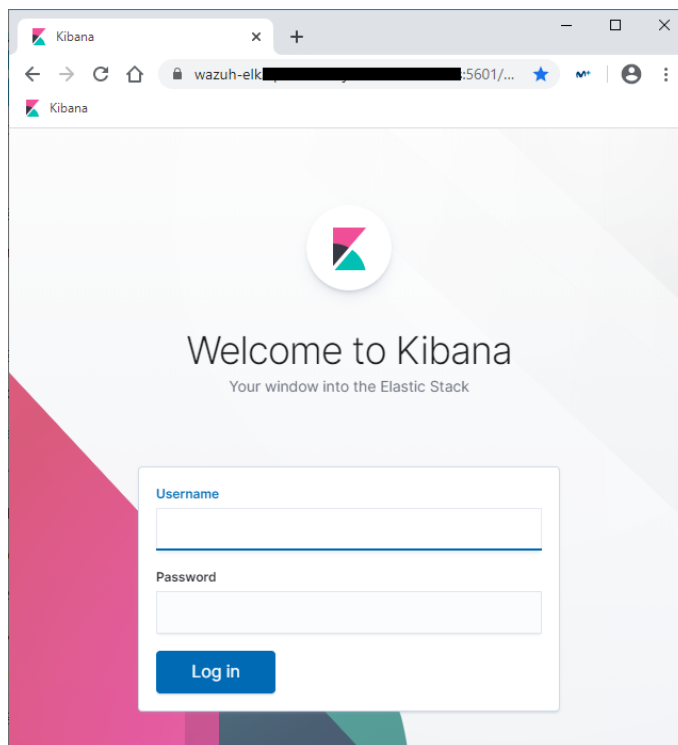


Figura 4.3: Portal autenticación Kibana con X-Pack

4.3. Monitorización de los servidores Wazuh en Nagios

Es recomendable que monitoricemos ambos servidores por varias razones:

- En primer lugar, es necesario que sepamos si algún servicio del servidor de Wazuh (wazuh-manager, wazuh-api) o del servidor ELK (Elasticsearch, Kibana) está caído y necesita ser levantado.
- En segundo lugar, es recomendable monitorizar en ambos servidores los recursos del sistema tales como la carga de CPU, memoria y espacio utilizados, etc. . . para ver si existe algún cuello de botella y tenemos que escalar el sistema, ya sea aumentando memoria, procesadores, espacio en disco, etc. . .

Para monitorizar los servidores con Nagios utilizaremos [NRPE](#)³. El proceso de instalación y configuración, al no estar directamente relacionado con Wazuh, se describe en el anexo F.

Una vez realizada la configuración descrita en dicho anexo, conseguimos una monitorización de los servidores Wazuh como las mostradas en las figuras 4.4 y 4.5.

Service Status Details For Host 'wazuh-server'							
Host	Service	Status	Last Check	Duration	Attempt	Status Information	
wazuh-server	Current Load	OK	03-27-2020 18:44:46	0d 0h 27m 48s	1/4	OK - load average: 0.05, 0.02, 0.00	
	Current Users	OK	03-27-2020 18:46:18	0d 0h 27m 30s	1/4	USERS OK - 2 users currently logged in	
	Mem usage	OK	03-27-2020 18:45:04	0d 0h 18m 16s	1/4	[MEMORY] Total: 3933 MB - Used: 2415 MB - 61% [SWAP] Total: 3932 MB - Used: 0 MB - 0%	
	NRPE	OK	03-27-2020 18:47:39	0d 0h 29m 20s	1/4	TCP OK - 0,001 second response time on 10.66.128.57 port 5666	
	PING	OK	03-27-2020 18:47:39	0d 0h 27m 47s	1/4	PING OK - Packet loss = 0%, RTA = 0.50 ms	
	Root Partition	OK	03-27-2020 18:47:39	0d 0h 21m 38s	1/4	DISK OK - free space: / 31131 MB (81% inode=96%):	
	SSH	OK	03-27-2020 18:44:47	0d 0h 27m 30s	1/4	TCP OK - 0,001 second response time on 10.66.128.57 port 22	
	Swap	OK	03-27-2020 18:46:20	0d 0h 21m 38s	1/4	SWAP OK - 100% free (3932 MB out of 3932 MB)	
	Total Procs	OK	03-27-2020 18:47:39	0d 0h 29m 19s	1/4	PROCS OK: 117 processes	
	wazuh-api	OK	03-27-2020 18:47:39	0d 0h 0m 48s+	1/4	TCP OK - 0,005 second response time on 10.66.128.57 port 55999	
	wazuh-manager	OK	03-27-2020 18:47:39	0d 0h 0m 48s+	1/4	TCP OK - 0,037 second response time on 10.66.128.57 port 1515	

Figura 4.4: Servidor Wazuh monitorizado en Nagios

Service Status Details For Host 'wazuh-elk'							
Host	Service	Status	Last Check	Duration	Attempt	Status Information	
wazuh-elk	Current Load	OK	03-27-2020 18:48:15	0d 0h 23m 17s	1/4	OK - load average: 0.01, 0.02, 0.00	
	Current Users	OK	03-27-2020 18:48:15	0d 0h 28m 45s	1/4	USERS OK - 1 users currently logged in	
	Mem usage	OK	03-27-2020 18:48:15	0d 0h 20m 58s	1/4	[MEMORY] Total: 3933 MB - Used: 3154 MB - 80% [SWAP] Total: 3932 MB - Used: 87 MB - 2%	
	NRPE	OK	03-27-2020 18:48:15	0d 0h 31m 49s	1/4	TCP OK - 0,001 second response time on 10.66.128.40 port 5666	
	PING	OK	03-27-2020 18:48:15	0d 0h 30m 16s	1/4	PING OK - Packet loss = 0%, RTA = 0.60 ms	
	Root Partition	OK	03-27-2020 18:48:15	0d 0h 23m 17s	1/4	DISK OK - free space: / 29860 MB (78% inode=92%):	
	SSH	OK	03-27-2020 18:48:15	0d 0h 28m 25s	1/4	TCP OK - 0,001 second response time on 10.66.128.40 port 22	
	Swap	OK	03-27-2020 18:48:15	0d 0h 23m 17s	1/4	SWAP OK - 98% free (3845 MB out of 3932 MB)	
	Total Procs	OK	03-27-2020 18:48:15	0d 0h 31m 48s	1/4	PROCS OK: 106 processes	
	elasticsearch-REST	OK	03-27-2020 18:48:15	0d 0h 3m 15s+	1/4	TCP OK - 0,004 second response time on 10.66.128.40 port 9200	
	elasticsearch-nodos	OK	03-27-2020 18:48:15	0d 0h 3m 15s+	1/4	TCP OK - 0,004 second response time on 10.66.128.40 port 9300	
	kibana	OK	03-27-2020 18:48:15	0d 0h 3m 15s+	1/4	TCP OK - 0,007 second response time on 10.66.128.40 port 5601	

Figura 4.5: Servidor ELK monitorizado en Nagios

³Nagios Remote Plugin Executor



5

DESPLIEGUE

5.1. Despliegue automático de agentes

Una vez tenemos los servidores Wazuh y Elastic funcionando, el siguiente paso es desplegar los agentes en los equipos ([endpoints](#)) para la recogida de la información y, como se indica en los objetivos de la sección [1.2](#), intentando utilizar herramientas de automatización.

En nuestro caso, los sistemas operativos donde se desplegarán los agentes son dos: Windows y Linux. A continuación estudiaremos el despliegue en cada uno de estos sistemas.

5.2. Despliegue automático en sistemas Linux

Wazuh permite el despliegue automático de agentes Linux con dos opciones: Puppet y Ansible. En nuestro caso hemos elegido Ansible por su sencillez basada en ssh. La instalación y configuración inicial de Ansible se detalla en el anexo [G](#). Una vez hecha dicha instalación y configuración, podemos pasar a instalar el playbook y el rol del agente de Wazuh.

5.2.0.1. Obtener los playbooks y roles de Wazuh

Wazuh tiene creados unos playbooks y roles para Ansible con los que es posible instalar los componentes del servidor Wazuh, de Elastic Stack o de los agentes, clonando para ello el repositorio en `/etc/ansible/roles`. Como prerequisite necesitamos la herramienta `git` que ya está instalada por defecto en nuestro servidor Wazuh. En primer lugar, configuramos nuestro proxy para `git`:

```
jpcozar@wazuh-server:~$ sudo git config --global http.proxy http://10.x.x.x:3128
```

A continuación, creamos el directorio para los roles de Ansible y clonamos el repositorio.

```

jpcozar@wazuh-server:~$ sudo mkdir /etc/ansible/roles; cd /etc/ansible/roles
jpcozar@wazuh-server:/etc/ansible/roles$ sudo git clone
↪ https://github.com/wazuh/wazuh-ansible.git

```

Ahora podríamos instalar cualquiera de los roles que nos hemos descargado.

5.2.0.2. Instalar el agente

Ahora podemos ver los roles y los playbooks preconfigurados dentro del directorio `/etc/ansible/roles/wazuh-ansible`:

```

jpcozar@wazuh-server:~$ cd /etc/ansible/roles/wazuh-ansible
jpcozar@wazuh-server:/etc/ansible/roles/wazuh-ansible$ ls -l playbooks
total 24
-rw-r--r-- 1 root root 430 abr 6 12:16 wazuh-agent.yml
-rw-r--r-- 1 root root 2750 abr 6 12:16 wazuh-elastic_stack-distributed.yml
-rw-r--r-- 1 root root 446 abr 6 12:16 wazuh-elastic_stack-single.yml
-rw-r--r-- 1 root root 163 abr 6 12:16 wazuh-elastic.yml
-rw-r--r-- 1 root root 145 abr 6 12:16 wazuh-kibana.yml
-rw-r--r-- 1 root root 210 abr 6 12:16 wazuh-manager.yml

```

Nosotros usaremos el playbook `wazuh-agent.yml` y el rol `wazuh-agent`.

El fichero de configuración por defecto es `/etc/ansible/roles/wazuh-ansible/roles/wazuh/ansible-wazuh-agent/defaults/main.yml`

El fichero para configurar los clientes, será `/etc/ansible/roles/wazuh-ansible/playbooks/wazuh-agent.yml`.

5.2.0.3. Preparar el playbook

Debemos crear un fichero YAML similar o modificar el que hay para adaptarlo a la configuración. En este caso, renombramos el original:

```

jpcozar@wazuh-server:~$ cd /etc/ansible/roles/wazuh-ansible/playbooks; sudo cp wazuh-agent.yml
↪ wazuh-agent.yml.original

```

Ahora modificamos el fichero `/etc/ansible/roles/wazuh-ansible/playbooks/wazuh-agent.yml` estableciendo los host clientes de Ansible (en nuestro caso todos los servidores Linux), el manager de Wazuh y el proxy:

```

---
- hosts: hacolx05, hacolx06, hacolx07, hacolx08, hacolx09, hacolx10
  roles:
    - ../roles/wazuh/ansible-wazuh-agent
  vars:
    wazuh_managers:
      - address: 10.x.x.57
        port: 1514
        protocol: udp
        api_port: 55999

```

```

api_proto: 'https'
api_user: WazuhCor
wazuh_agent_authd:
  registration_address: 10.x.x.57
  enable: true
  port: 1515
  ssl_agent_ca: null
  ssl_auto_negotiate: 'no'
environment:
  http_proxy: http://10.x.x.x:3128
  https_proxy: http://10.x.x.x:3128

```

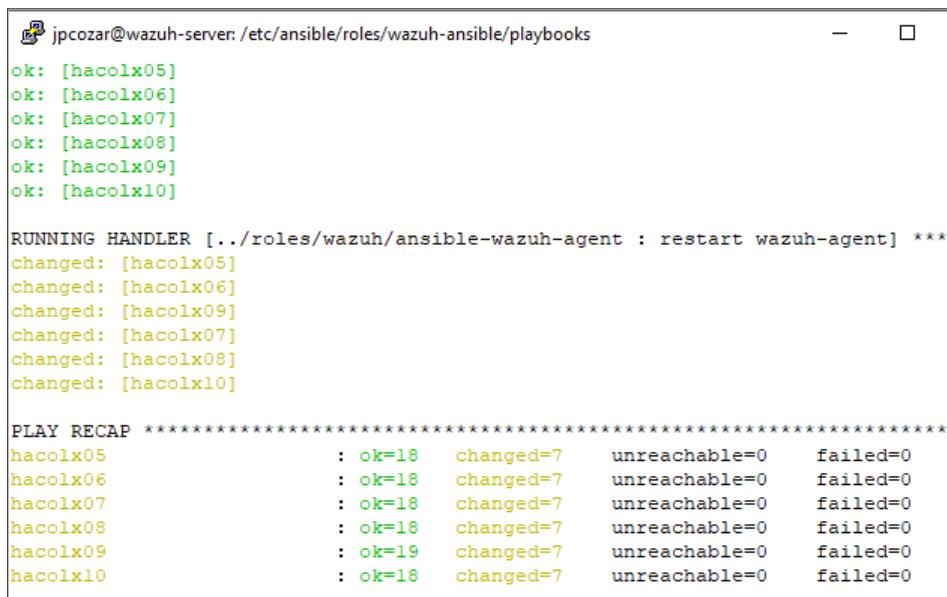
Y ahora ya podemos ejecutar el comando de Ansible para instalar el rol en cada uno de los servidores:

```

jpcozar@wazuh-server:/etc/ansible/roles/wazuh-ansible/playbooks$ ansible-playbook
↪ wazuh-agent.yml -b -K
SUDO password:

```

Una vez ejecutado con éxito, obtendremos una imagen como la de la figura 5.1.



```

jpcozar@wazuh-server:/etc/ansible/roles/wazuh-ansible/playbooks
ok: [hacolx05]
ok: [hacolx06]
ok: [hacolx07]
ok: [hacolx08]
ok: [hacolx09]
ok: [hacolx10]

RUNNING HANDLER [../roles/wazuh/ansible-wazuh-agent : restart wazuh-agent] ***
changed: [hacolx05]
changed: [hacolx06]
changed: [hacolx09]
changed: [hacolx07]
changed: [hacolx08]
changed: [hacolx10]

PLAY RECAP *****
hacolx05           : ok=18   changed=7   unreachable=0   failed=0
hacolx06           : ok=18   changed=7   unreachable=0   failed=0
hacolx07           : ok=18   changed=7   unreachable=0   failed=0
hacolx08           : ok=18   changed=7   unreachable=0   failed=0
hacolx09           : ok=19   changed=7   unreachable=0   failed=0
hacolx10           : ok=18   changed=7   unreachable=0   failed=0

```

Figura 5.1: Ejecución exitosa de Ansible

En el servidor de Wazuh, ejecutamos el siguiente comando para ver que se han registrado los agentes:

```

root@wazuh-server:~# /var/ossec/bin/manage_agents -l

Available agents:
ID: 001, Name: hacolx05, IP: 10.x.x.x
ID: 002, Name: hacolx06, IP: 10.x.x.x
ID: 003, Name: hacolx07, IP: 10.x.x.x
ID: 004, Name: hacolx09, IP: 10.x.x.x

```

```
ID: 005, Name: hacolx08, IP: 10.x.x.x
ID: 006, Name: hacolx10, IP: 10.x.x.x
```

Y en cada uno de los servidores que tienen el agente podríamos ver que está instalado y ejecutándose con el comando:

```
root@wazuh-server:~# systemctl status wazuh-agent
```

Y se nos mostraría lo que vemos en la figura 5.2

```

jpcozar@hacolx05: ~
● wazuh-agent.service - Wazuh agent
   Loaded: loaded (/etc/systemd/system/wazuh-agent.service; enabled; vendor pres
   Active: active (running) since mié 2020-04-01 23:54:44 CEST; 6h ago
   Process: 13773 ExecStop=/usr/bin/env ${DIRECTORY}/bin/ossec-control stop (code
   Process: 13835 ExecStart=/usr/bin/env ${DIRECTORY}/bin/ossec-control start (co
   Tasks: 27
   Memory: 431.4M
   CPU: 2min 12.776s
   CGroup: /system.slice/wazuh-agent.service
           └─14656 /var/ossec/bin/ossec-execd
             └─14667 /var/ossec/bin/ossec-agentd
               └─14683 /var/ossec/bin/ossec-syscheckd
                 └─14699 /var/ossec/bin/ossec-logcollector
                   └─14718 /var/ossec/bin/wazuh-modulesd

abr 01 23:54:39 hacolx05 systemd[1]: Starting Wazuh agent...
abr 01 23:54:39 hacolx05 env[13835]: Starting Wazuh v3.12.0...
abr 01 23:54:40 hacolx05 env[13835]: Started ossec-execd...
abr 01 23:54:40 hacolx05 env[13835]: Started ossec-agentd...
abr 01 23:54:41 hacolx05 env[13835]: Started ossec-syscheckd...
abr 01 23:54:41 hacolx05 env[13835]: Started ossec-logcollector...
abr 01 23:54:42 hacolx05 env[13835]: Started wazuh-modulesd...
abr 01 23:54:44 hacolx05 env[13835]: Completed.
lines 1-23

```

Figura 5.2: Agente Wazuh en ejecución

Y ya podríamos ver los servidores Linux registrados en Kibana (figura 5.3).

ID	Name	IP	Status	Group	OS name	OS version	Version	Registration date	Last keep alive
001	hacolx05	10.███	Active	default	Ubuntu	16.04.6 LTS	Wazuh v3.12.0	2020/04/01 23:54:09	2020/04/01 23:55:43
002	hacolx06	10.███	Active	default	Ubuntu	16.04.6 LTS	Wazuh v3.12.0	2020/04/01 23:54:09	2020/04/01 23:55:43
003	hacolx07	10.███	Active	default	Ubuntu	16.04.6 LTS	Wazuh v3.12.0	2020/04/01 23:54:09	2020/04/01 23:55:43
004	hacolx08	10.███	Active	default	Ubuntu	16.04.6 LTS	Wazuh v3.12.0	2020/04/01 23:54:10	2020/04/01 23:55:43
005	hacolx09	10.███	Active	default	Ubuntu	14.04.6 LTS, Trusty Tahr	Wazuh v3.12.0	2020/04/01 23:54:10	2020/04/01 23:55:42
006	hacolx10	10.███	Active	default	Ubuntu	16.04.6 LTS	Wazuh v3.12.0	2020/04/01 23:54:10	2020/04/01 23:55:42

6 items (3.01 seconds)

Figura 5.3: Agentes Wazuh de servidores Linux en Kibana

5.3. Despliegue automático en sistemas Windows

5.3.1. Despliegue del agente Wazuh por GPO

Para desplegar el agente en los servidores Windows utilizaremos la implementación de software por directiva de grupo (GPO)¹ para todos los servidores de la sede principal que forman parte de nuestro dominio. Esto se ha documentado en el anexo H.

5.3.1.1. Instalación manual en servidor independiente

En el caso de la sede de formación, se hará una instalación manual ya que se trata de un servidor independiente, sin dominio. En este último caso, ejecutaremos el siguiente comando en Powershell como administrador:

```
PS D:\Wazuh-agent.\wazuh-agent-3.12.0-1.msi /q WAZUH_MANAGER="10.x.x.57"
↪ WAZUH_REGISTRATION_SERVER="10.x.x.57"
```

Una vez ejecutada la directiva y el comando anterior, tendríamos todos los servidores Windows registrados, como vemos en la figura 5.4.

007	HACO...	10.██.██.██	Active	default	Microsoft Windows Server 2012 R2 Standard	6.3.9600	Wazuh v...	2020/04/02 09:03:...	2020/04/02 16:...
008	HACO...	10.██.██.██	Active	default	Microsoft Windows Server 2012 R2 Standard	6.3.9600	Wazuh v...	2020/04/02 09:11:55	2020/04/02 16:...
009	HACO...	10.██.██.██	Active	default	Microsoft Windows Server 2012 R2 Standard	6.3.9600	Wazuh v...	2020/04/02 09:18:...	2020/04/02 16:...
010	HACO...	10.██.██.██	Active	default	Microsoft Windows Server 2012 R2 Standard	6.3.9600	Wazuh v...	2020/04/02 10:29:...	2020/04/02 16:...
011	HACO...	10.██.██.██	Active	default	Microsoft Windows Server 2012 R2 Standard	6.3.9600	Wazuh v...	2020/04/02 10:39:...	2020/04/02 16:...
012	HACO...	10.██.██.██	Active	default	Microsoft Windows Server 2012 R2 Standard	6.3.9600	Wazuh v...	2020/04/02 14:07:...	2020/04/02 16:...
013	JESSICA	10.██.██.██	Active	default	Microsoft Windows Server 2008 R2 Standard Ed...	6.1.7601	Wazuh v...	2020/04/02 14:24:...	2020/04/02 16:...
014	HACO...	10.██.██.██	Active	default	Microsoft Windows Server 2012 R2 Standard	6.3.9600	Wazuh v...	2020/04/02 16:56:...	2020/04/02 16:...

14 items (2.69 seconds)

Figura 5.4: Agentes Wazuh de servidores Windows en Kibana

5.4. Creación de grupos en Wazuh

Una vez tenemos todos los servidores a monitorizar registrados, es conveniente crear grupos para tenerlo todo mejor organizado y poder aplicar en el futuro políticas de grupo. En nuestro caso crearemos un grupo llamado LinuxServers y otro WindowsServers. Tanto la creación como la asignación de grupos se puede hacer desde terminal o desde el entorno de Kibana. Nosotros lo haremos desde Kibana ya que es más sencillo y rápido. Para ello nos vamos a Management y seleccionamos Groups, creando dos grupos:

1. LinuxServers
2. WindowsServers

Y a continuación asignamos los agentes ya registrado a cada uno de sus grupos.

Con esto terminamos la fase de despliegue y ya podemos empezar a ver los casos de uso.

¹Group Policy Object

6

CASOS DE USO

6.1. File Integrity Monitoring: FIM

Mediante la monitorización de la integridad de los ficheros (FIM), detectamos y alertamos cuando se producen cambios en los ficheros del sistema y de las aplicaciones. Por defecto viene activado para determinados directorios tanto para Linux como para Windows, y nos indica si los ficheros monitorizados han sido añadidos, borrados o modificados.

Podemos añadir los ficheros que queramos monitorizar. En nuestro caso, por ejemplo, tenemos un fichero de contraseñas departamental generado con la aplicación [Keepass](#) que nos interesa tener controlado. El fichero, llamado contraseñas.kdbx, está en un recurso Samba, el cual montamos a través del fichero `/etc/fstab` del servidor Linux `hac01x05` en el directorio `/mnt/KEEPASS`.

Ahora editamos el fichero `ossec.conf` en el servidor linux (`hac01x05`) y añadimos lo siguiente para poder monitorizar el directorio con el fichero de contraseñas:

```
<syscheck>
...
<directories >/mnt/KEEPASS</directories>
<skip_nfs>no</skip_nfs>
...
</syscheck>
```

Para poder chequear recursos NFS o CIFS necesitamos utilizar la opción `<skip_nfs>`. Hay que notar que no podemos utilizar la opción de chequear los ficheros en tiempo real (como haremos al usar VirusTotal) porque se utiliza `inotify` en Linux para detectar los cambios. Si modificamos el fichero del recurso CIFS desde Linux donde está montado, si veríamos la modificación en tiempo real. Pero lo normal será hacer la modificación desde Windows que

es donde tenemos instalado el programa Keepass. Por tanto, inotify no detectará el cambio y no funcionará la notificación en tiempo real. Lo que haremos será reducir el intervalo de syscheck a 15 minutos.

Una vez hecho esto, reiniciamos el agente y probamos a modificar el fichero:

```
root@wazuh-server:~# systemctl restart wazuh-agent
```

Y ya debería aparecernos en las alertas y en Kibana cuando modificamos el fichero, como vemos en la figura 6.1

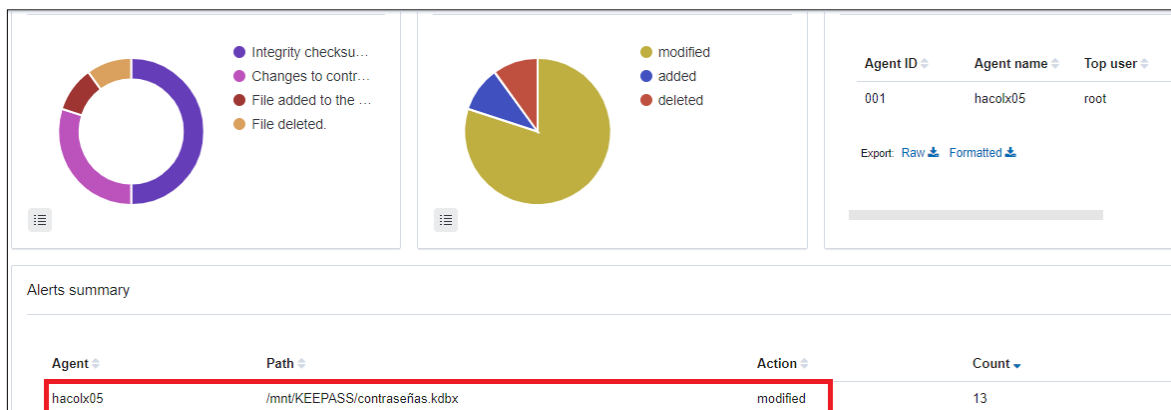


Figura 6.1: Integridad de fichero de contraseñas Keepass

Además podemos generar una alerta que nos avise con un determinado nivel de severidad. Para ello creamos una regla en el fichero `/var/ossec/etc/local_rules.xml` con el siguiente contenido:

```
<group name="local,syscheck,ossec">
<rule id="100004" level="12">
<if_group>syscheck</if_group>
<match>/mnt/KEEPASS/contraseñas.kdbx</match>
<description>Changes to contraseñas.kdbx - Critical file!</description>
</rule>
</group>
```

Y cuando se produzca el evento anterior, nos emitirá una alerta de nivel 12, con un ID de regla de 100004, como vemos en la figura 6.2

Veremos otro ejemplo de la monitorización de la integridad del sistema de ficheros en la integración con VirusTotal en el apartado 6.5.

A continuación vemos una extensión del FIM, que nos permite, aparte de monitorizar el fichero, hacer una auditoría de quién y cómo lo hizo. Esta característica se denomina `who-data` y está disponible tanto para Windows como para Linux.


```

t decoder.name      syscheck_integrity_changed
t full_log          >
                   File '/mnt/KEEPASS/contrasenyas.kdbx' modified
                   Hard links: /mnt/KEEPASS/contraseñas.kdbx
                   Mode: scheduled
                   Changed attributes: size,mtime,inode,md5,sha1,sha256
                   Size changed from '42222' to '42270'
                   Old modification time was: '1587213837', now it is '1587214329'
                   Old inode was: '14' now it is '9465738553196558'
t id                1587214495.497928864
t input.type        log
t location          syscheck
t manager.name      wazuh-server
t rule.description  Changes to contraseñas.kdbx - Critical file!
# rule.firedtimes   6
t rule.groups       local, syscheck, ossec
t rule.id           100004
# rule.level        12
@ rule.mail         true
t syscheck.changed_attributes size, mtime, inode, md5, sha1, sha256
t syscheck.event    modified
t syscheck.gid_after 0
t syscheck.gname_after root
t syscheck.hard_links /mnt/KEEPASS/contraseñas.kdbx
t syscheck.inode_after 9465738553196558
t syscheck.inode_before 14
t syscheck.md5_after  c4c70244c6dc2f3534ed530cc3b51788
t syscheck.md5_before 5cbba2299beb420f1ce65a02ed299352
@ syscheck.mtime_after Apr 18, 2020 @ 14:52:09.000

```

Figura 6.2: Alerta en el fichero de contraseñas Keepass

6.2. Auditoría mediante who-data

Mediante esta opción de Wazuh podemos ver qué usuario hizo los cambios en los ficheros que se están monitorizando, el nombre del programa o proceso que se usó para llevarlo a cabo y en qué momento se realizó.

6.2.1. Auditoría who-data en Linux

Para poder realizar la auditoría en Linux, necesitamos tener instalado el servicio Audit en los servidores Linux. En nuestro caso, utilizamos de nuevo Ansible para desplegarlo, como vemos en el anexo G, sección G.3.

Una vez instalado, si por ejemplo queremos monitorizar quién realiza cambios en el directorio /etc hasta el nivel 1 de profundidad (por ejemplo nos interesa saber quien ha modificado los ficheros /etc/passwd, /etc/shadow, /etc/hosts.allow, etc...), debemos realizar cambios en los agentes en el fichero de configuración /var/ossec/etc/ossec.conf. Como queremos hacerlo para todos los Linux, usaremos la configuración centralizada (que se verá de nuevo en el apartado 6.4). Editamos en el servidor de Wazuh, el fichero común para todos los agentes de los servidores Linux que está en /var/ossec/etc/shared/LinuxServers/agent.conf y añadimos el siguiente bloque:

```
<agent_config>
...
<syscheck>
<directories check_all="yes" whodata="yes"
↳ recursion_level="0">/etc</directories>
</syscheck>
...
</agent_config>
```

Le indicamos que queremos monitorizar quién accede a qué del directorio /etc pero sólo del primer nivel (recursión 0). Una vez hecho esto, verificamos la configuración del agente y reiniciamos el servicio de wazuh-manager en el servidor y los agentes de forma remota:

```
root@wazuh-server:# /var/ossec/bin/verify-agent-conf
root@wazuh-server:# systemctl restart wazuh-manager
root@wazuh-server:# /var/ossec/bin/agent_control -R -a
```

Ahora en uno de los agentes Linux, podemos comprobar si se está aplicando la regla:

```
root@wazuh-server:# auditctl -l|grep wazuh_fim
-w /etc/ -p wa -k wazuh_fim
```

Posteriormente, modificamos por ejemplo el fichero /etc/hosts.allow del servidor hacolx05 y, aparte de ver que se ha modificado el fichero en el apartado de Integrity monitoring de Kibana, si buscamos la alerta generada, podemos ver quién lo ha hecho (jpcozar) de qué forma (mediante un sudo), con qué programa (mediante el editor vim) y en qué momento, como vemos en la figura 6.3

t	rule.id	550
#	rule.level	7
⊙	rule.mail	false
t	rule.nist_800_53	SI.7
t	rule.pci_dss	11.5
	syscheck.audit.effective_user.id	0
	syscheck.audit.effective_user.name	root
t	syscheck.audit.group.id	0
t	syscheck.audit.group.name	root
t	syscheck.audit.login_user.id	1001
t	syscheck.audit.login_user.name	jpcozar
t	syscheck.audit.process.id	40425
t	syscheck.audit.process.name	/usr/bin/vim.gnome
t	syscheck.audit.process.ppid	40424
t	syscheck.audit.user.id	0
t	syscheck.audit.user.name	root
t	syscheck.changed_attributes	size, mtime, inode, md5, sha1, sha256
t	syscheck.event	modified
t	syscheck.gid_after	0
t	syscheck.gname_after	root
t	syscheck.inode_after	266206
t	syscheck.inode_before	262155
t	syscheck.md5_after	b635b0fb913d9f1761ecfcd265312110
t	syscheck.md5_before	d0cfb796d371b0182cd39d589b1c1ce3
📅	syscheck.mtime_after	Apr 16, 2020 @ 18:00:46.000
📅	syscheck.mtime_before	Jun 6, 2017 @ 11:24:24.000
t	syscheck.path	/etc/hosts.allow

Figura 6.3: Auditoría de ficheros Linux

6.2.2. Auditoría who-data en Windows

De forma análoga, podríamos monitorizar en tiempo real quién modificó determinados directorios y ficheros en Windows. Por ejemplo, si queremos monitorizar el fichero de hosts de Windows (C:\Windows\system32\drivers\etc\hosts), añadiríamos el bloque siguiente en el fichero de configuración común /var/ossec/etc/shared/WindowsServers/agent.conf de los servidores Windows:

```
<agent_config>
...
<syscheck>
...
  <directories check_all="yes" whodata="yes">%WINDIR%\SysNative\drivers\etc</directories>
</syscheck>
...
</agent_config>
```

Una vez hecho esto, verificamos la configuración del agente y reiniciamos el servicio de wazuh-manager en el servidor y los agentes de forma remota:

```
root@wazuh-server:~# /var/ossec/bin/verifiy-agent-conf
root@wazuh-server:~# systemctl restart wazuh-manager
root@wazuh-server:~# /var/ossec/bin/agent_control -R -a
```

Ahora si modificamos el fichero de hosts de Windows C:\Windows\system32\drivers\etc\hosts, veríamos la alerta en la pestaña de Integrity Monitoring como vemos en la figura 6.4

HACONF11	c:\windows\sysnative\drivers\etc\hosts	modified	1
----------	--	----------	---

Figura 6.4: Auditoría de ficheros en Windows - Integrity Monitoring

Y la alerta con la auditoría indicando quién (el Administrador), cómo (usando el notepad) y cuándo se modificó el fichero, como vemos en la figura 6.5

t	rule.id	550
#	rule.level	7
🔊	rule.mail	false
t	rule.nist_800_53	SI.7
t	rule.pci_dss	11.5
t	syscheck.attrs_after	ARCHIVE
t	syscheck.audit.process.id	7688
t	syscheck.audit.process.name	C:\Windows\System32\notepad.exe
t	syscheck.audit.user.id	S-1-5-21-382363961-1676483636-2948957613-500
t	syscheck.audit.user.name	Administrador
t	syscheck.changed_attributes	size, mtime, md5, sha1, sha256
t	syscheck.event	modified
t	syscheck.md5_after	8b9d3b0b12b7d8e7c04ccffdce0a8c4c
t	syscheck.md5_before	7f92cff2be77510324c394137855776d
📅	syscheck.mtime_after	Apr 22, 2020 @ 21:19:18.000
📅	syscheck.mtime_before	Apr 22, 2020 @ 20:41:34.000
t	syscheck.path	c:\windows\sysnative\drivers\etc\hosts
t	syscheck.sha1_after	d7218535c1f0ad006a6f89f57cd5f2312cd0bbae
t	syscheck.sha1_before	59e09acf04f91a2f66cbde15fa14aadbeca2f4ac
t	syscheck.sha256_after	435028aad0bc3062d64d30f5a869acd57cc7efc93e43e739f7c0dac291f66400
t	syscheck.sha256_before	af2bffc6e34d852856bbc299efce902375bba5ac9ad034634df39b6e556db92
#	syscheck.size_after	882
#	syscheck.size_before	902
t	syscheck.uid_after	S-1-5-18

Figura 6.5: Auditoría de ficheros en Windows

6.3. Detección de vulnerabilidades

Es una característica muy interesante de Wazuh que nos permite detectar vulnerabilidades conocidas tanto en sistemas Linux (Debian, Ubuntu, Red Hat) como en sistemas Windows. Para los primeros, existen ficheros de bases de datos propios proporcionados por Debian, Canonical y Red Hat respectivamente y para Windows se hace uso de la National Vulnerability Database (NVD)¹.

Existen dos maneras de que Wazuh obtenga dichos ficheros de vulnerabilidades: de forma online o de forma offline si estamos detrás de algún cortafuegos o proxy (como es nuestro caso).

6.3.1. Detección de vulnerabilidades en servidores Ubuntu

6.3.1.1. Configuración del Wazuh Manager

En primer lugar tenemos que activar la detección de vulnerabilidades en general en el servidor Wazuh, en el fichero `/var/ossec/etc/ossec.conf`.

```
...
<vulnerability-detector>
  <enabled>yes</enabled>
  <interval>5m</interval>
  <ignore_time>6h</ignore_time>
  <run_on_start>yes</run_on_start>
...
```

Y en el mismo bloque xml, deberíamos activar los sistemas (llamados providers) para los que queremos que se detecten vulnerabilidades. Podemos indicarlo por el nombre de versión o por el número. Como vamos a monitorizar servidores Ubuntu 14.x (Trusty Tahr), 16.x (Xenial) y 18.x (Bionic) estos son los que habilitaremos únicamente:

```
...
<provider name="canonical">
  <enabled>yes</enabled>
  <os>trusty</os>
  <os>xenial</os>
  <os>bionic</os>
  <update_interval>1h</update_interval>
</provider>
...
```

Estas opciones lo que hacen es descargarse los ficheros para Bionic, Xenial y Trusty, respectivamente:

- <https://people.canonical.com/~ubuntu-security/oval/com.ubuntu.bionic.cve.oval.xml>
- <https://people.canonical.com/~ubuntu-security/oval/com.ubuntu.xenial.cve.oval.xml>
- <https://people.canonical.com/~ubuntu-security/oval/com.ubuntu.trusty.cve.oval.xml>

¹National Vulnerability Database

Pero el bug que hemos reportado en <https://github.com/wazuh/wazuh/issues/4855>², es que los ficheros que suministra Canonical en esas direcciones y que se espera que sean xml están comprimidos y son en realidad .xml.bz2. Por tanto hemos tenido que optar por una configuración offline propia. En este caso, los pasos son:

1. Creamos un directorio de descarga de vulnerabilidades para Ubuntu:

```
root@wazuh-server:# mkdir -p /var/ossec/offline-vulnerabilities/ubuntu
```

2. Creamos un script que realice las descargas y descompresión de los ficheros para Ubuntu que llamamos ubuntu-generator.sh:

```
#!/bin/bash
#####
# Uso: Script para descargar los feeds de Canonical en xml.bz2
# Fecha: Abril 2020
# Autor: Javier Polo Cózar
#####

WGET=`which wget`
BUNZIP2=`which bunzip2`
DIR="/var/ossec/offline-vulnerabilities/ubuntu"
URL="https://people.canonical.com/~ubuntu-security/oval/"
pre="com.ubuntu."
pos=".cve.oval.xml"

for i in bionic xenial trusty
do
    $WGET $URL/$pre$i$pos -O $DIR/$pre$i$pos.bz2
    $BUNZIP2 -df $DIR/$pre$i$pos.bz2
done
```

3. Ahora, ponemos una tarea en el crontab, que los descargue cada hora por ejemplo:

```
@hourly /var/ossec/offline-vulnerabilities/ubuntu-generator.sh >/dev/null
```

4. Ya podemos modificar el fichero /var/ossec/etc/ossec.conf en el servidor de Wazuh y cambiar el provider, indicándole que tomará los feeds de forma local. Lo dejamos así:

```
<provider name="canonical">
  <enabled>yes</enabled>
  <os path="/var/ossec/offline-vulnerabilities/ubuntu/com.ubuntu.trusty.cve.oval.xml">
    ↪ trusty</os>
  <os path="/var/ossec/offline-vulnerabilities/ubuntu/com.ubuntu.xenial.cve.oval.xml">
    ↪ xenial</os>
  <os path="/var/ossec/offline-vulnerabilities/ubuntu/com.ubuntu.bionic.cve.oval.xml">
    ↪ bionic</os>
  <update_interval>1h</update_interval>
</provider>
```

5. Cuando hemos acabado, reiniciamos el manager de Wazuh.

²El problema fue resuelto por el equipo de Wazuh en la versión 3.12.1

```
root@wazuh-server:/var/ossec/etc# systemctl restart wazuh-manager
```

6.3.1.2. Configuración de los agentes para servidores Linux

En cada uno de los agentes en los que queramos ejecutar el análisis de vulnerabilidades tenemos que verificar que el módulo usado para recolectar los paquetes está instalado en el sistema. Para ello, editamos el fichero `/var/ossec/etc/ossec.conf` y verificamos que el bloque está así:

```
<wodle name="syscollector">
<disabled>no</disabled>
<interval>1h</interval>
<scan_on_start>yes</scan_on_start>
<hardware>yes</hardware>
<os>yes</os>
<network>yes</network>
<packages>yes</packages>
<ports all="no">yes</ports>
<processes>yes</processes>
</wodle>
```

Y ya deberían empezar a chequearse las vulnerabilidades. Si tenemos algún problema, debemos habilitar en el servidor Wazuh el debug de módulos en el fichero `/var/ossec/etc/internal_options.conf` poniendo la opción:

```
wazuh_modules.debug=2
```

y reiniciando a continuación el servicio de `wazuh-manager`. Una vez hecho esto, realizaríamos una búsqueda en el log del módulo de detección de vulnerabilidades, con la cadena:

```
root@wazuh-server:# grep vulnerability-detector /var/ossec/logs/ossec.log

2020/04/06 15:48:17 wazuh-modulesd:vulnerability-detector: INFO: (5452): Starting vulnerability
↔ scanning.
2020/04/06 15:51:03 wazuh-modulesd:vulnerability-detector: INFO: (5461): Starting Ubuntu Bionic
↔ database update.
2020/04/06 15:51:17 wazuh-modulesd:vulnerability-detector: INFO: (5494): The update of the
↔ Ubuntu Bionic feed finished successfully.
2020/04/06 15:51:17 wazuh-modulesd:vulnerability-detector: INFO: (5461): Starting Ubuntu Xenial
↔ database update.
```

Como observamos en la figura 6.6 van apareciendo las vulnerabilidades encontradas en los servidores Ubuntu de nuestra organización, pudiendo filtrar por servidor, vulnerabilidad CVE, paquete afectado, etc. . .

6.3.2. Detección de vulnerabilidades en servidores Windows

Para detectar las vulnerabilidades en los sistemas Windows se hace uso de la base de datos de la **NVD**. Igualmente tenemos la opción online u offline.

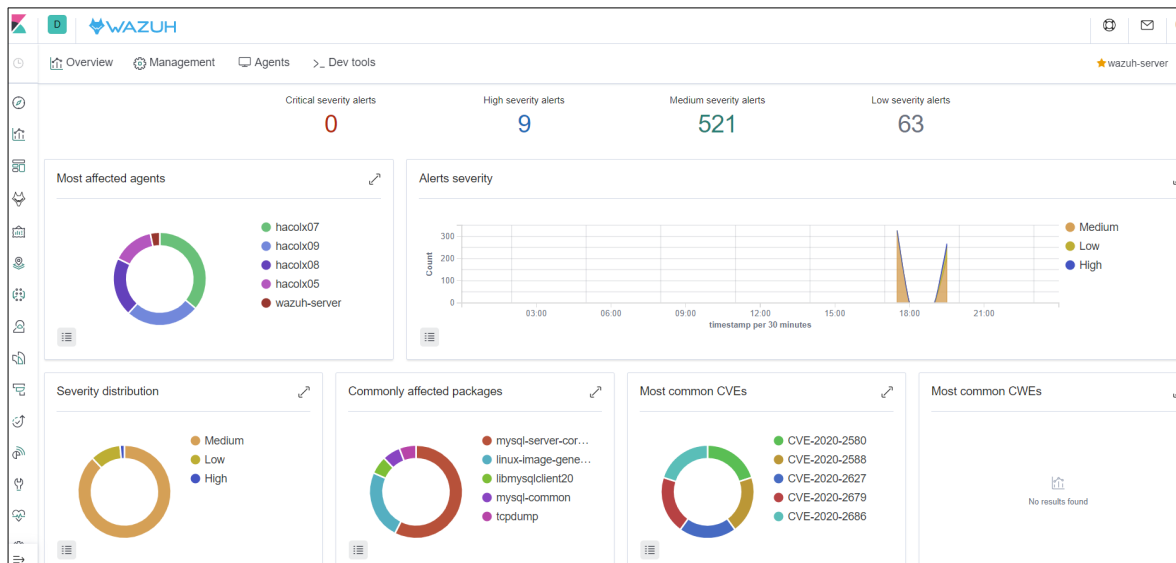


Figura 6.6: Vulnerabilidades en servidores Linux/Ubuntu

De nuevo detectamos un bug en la configuración offline que reportamos en <https://github.com/wazuh/wazuh/issues/4850>³.

La configuración online sí funcionaba en casa (al carecer de proxy y firewall) pero no en la organización. Así que hemos tenido que aplicar el *workaround*⁴ explicado en <https://github.com/wazuh/wazuh/issues/1112> para poder detectar las vulnerabilidades en Windows. Consiste en establecer el proxy en el fichero de configuración `/etc/ossec-init.conf` para que se puedan obtener los feeds de Internet a través del proxy.

6.3.2.1. Configuración del servidor Wazuh

Configuramos el servidor para que obtenga las actualizaciones desde el año 2010 y que se actualice cada hora:

```
<provider name="nvd">
  <enabled>yes</enabled>
  <update_from_year>2010</update_from_year>
  <update_interval>1h</update_interval>
</provider>
```

6.3.2.2. Configuración de los agentes para servidores Windows

En este caso cada agente debe tener una configuración similar a ésta.

```
<wodle name="syscollector">
<disabled>no</disabled>
<interval>1h</interval>
<scan_on_start>yes</scan_on_start>
<hardware>yes</hardware>
```

³Se consideró finalmente un bug en la documentación de Wazuh

⁴rodeo

```

<os>yes</os>
<network>yes</network>
<packages>yes</packages>
<ports all="no">yes</ports>
<processes>yes</processes>
<hotfixes>yes</hotfixes>
</wodle>

```

Es la que trae por defecto, a excepción del tag `<hotfixes>yes</hotfixes>` que nos permite detectar los hotfixes aplicados. Como debemos realizar dicho cambio en cada uno de los servidores Windows monitorizados, en vez de hacerlo a mano, vamos a utilizar de nuevo la configuración centralizada.

6.4. Configuración centralizada

Al tener creados los grupos, los agentes pueden configurarse remotamente usando el fichero `agent.conf`. En nuestro caso, queremos modificar los agentes del grupo Windows, así que editamos el fichero `agent.conf` que reside en `/var/ossec/etc/shared/WindowsServers`. Dicho fichero lo dejamos así:

```

<agent_config os="Windows">

  <!-- System inventory -->
  <wodle name="syscollector">
    <disabled>no</disabled>
    <interval>1h</interval>
    <scan_on_start>yes</scan_on_start>
    <hardware>yes</hardware>
  </wodle>
  <os>yes</os>
  <network>yes</network>
  <packages>yes</packages>
  <ports all="no">yes</ports>
  <processes>yes</processes>
  <hotfixes>yes</hotfixes>
</agent_config>

```

Y en pocos minutos tendríamos en cada uno de los servidores Windows, dicho fichero en la ruta `C:\Program Files (x86)\ossec-agent\shared`. Para comprobar que el agente con ID 007 - Servidor Windows, está sincronizado ejecutamos:

```

root@wazuh-server:~# /var/ossec/bin/agent_groups -S -i 007
Agent '007' is synchronized.

```

Una vez desplegado, ya empezaría a detectar las vulnerabilidades en los sistemas Windows como vemos en la figura 6.7.

A modo de ejemplo, hemos instalado el software Adobe Reader XI y Google Chrome, para comprobar que no sólo detecta parches para el sistema operativo, sino también vulnerabilida-

des en el software. Esto lo hace a través del **CPE** (Common Platform Enumeration), creando Wazuh un diccionario auxiliar que traduce el inventario realizado por Syscollector de los agentes Windows a un formato válido para los feeds obtenidos de la National Vulnerability Database.

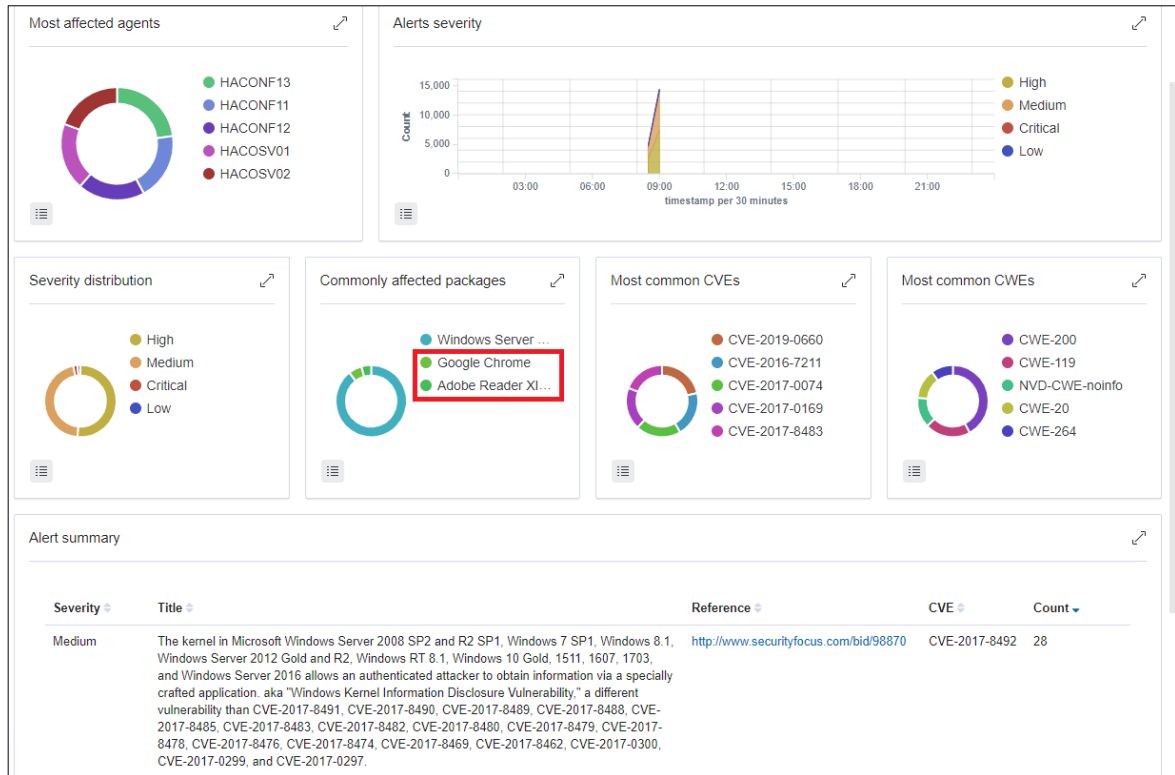


Figura 6.7: Vulnerabilidades en servidores Windows

6.5. Integración con VirusTotal

Wazuh puede integrarse con el servicio online **VirusTotal** de análisis de ficheros para la detección de virus, gusanos, troyanos y otro software malicioso. Existe una API pública con las siguientes restricciones:

1. Se establece un límite de peticiones de 4 por minuto, 1000 al día y 30000 al mes.
2. Se le da un acceso de baja prioridad a las peticiones hechas por esta API.
3. La API no puede usarse en productos o servicios comerciales.

Y luego existe una API privada sin limitaciones.

En nuestro caso, ya que contamos con un antivirus corporativo y queremos únicamente probarlo en algunos directorios (por ejemplo en uno de los wikis corporativos donde los usuarios pueden subir ficheros) usaremos la API pública. Para obtener la API key necesitamos registrarnos en la comunidad de VirusTotal. Una vez hecho esto, editamos el fichero `/var/ossec/etc/ossec.conf` y añadimos al final, antes del tag `</ossec_config>` el siguiente bloque:

```

<ossec_config>
...
<integration>
<name>virustotal</name>
<api_key>"API_GENERADA"</api_key>
<group>syscheck</group>
<alert_format>json</alert_format>
</integration>
...
</ossec_config>

```

6.5.1. Usar FIM para monitorizar un directorio

En los wikis los usuarios pueden subir ficheros con estas extensiones: pdf, png, gif, jpg, jpeg, doc, xls, mpp, ppt, tiff, bmp, docx, xlsx, pptx, ps, odt, ods, odp, odg, svg.

Dichos ficheros se guardan en la ruta `/var/www/html/wiki-sief/images/` en varios subdirectorios que se van creando. Así que nos iríamos al servidor Linux que alberga el wiki (en este caso haco1x08) y editaríamos el fichero de agente `/var/ossec/etc/ossec.conf` añadiendo lo siguiente para monitorizar el directorio de subidas en tiempo real:

```

<syscheck>
...
<directories check_all="yes"
  ↪ realtime="yes">/var/www/html/wiki-sief/images</directories>
...
</syscheck>

```

Existen ficheros de prueba con malware que nos permiten testear los sistemas antivirus. Uno de ellos está disponible para descarga en la web <https://www.eicar.org/>. El problema es que los ficheros allí suministrados tienen extensión `.com`, `.txt` o `.zip`, con lo cual no valdrían para subirlo al wiki ya que no tienen la extensión permitida y cambiarla nos indicará un error de tipo MIME ya que Mediawiki realiza una comprobación de la extensión del fichero con la cabecera.

Pero un usuario malintencionado podría incrustar dicho malware en un fichero doc y convertirlo a pdf, que ha sido la solución que hemos encontrado [aquí](#). Este fichero pdf es el que hemos usado como pruebas. Una vez subido al Mediawiki, vemos que es detectado, tanto en las alertas (figura 6.8) como en la interfaz del agente en concreto (figura 6.9).

```

{
  "agent.ip": "10.███",
  "agent.name": "hacolx08",
  "data.integration": "virustotal",
  "data.virustotal.found": 1,
  "data.virustotal.malicious": 1,
  "data.virustotal.permalink": "https://www.virustotal.com/file/86a96ec03ba8242c1486456d67ee17f919128754846dbb3bdf5e836059091dba/analysis/1585210969/",
  "data.virustotal.positives": 40,
  "data.virustotal.scan_date": "2020-03-26 08:22:49",
  "data.virustotal.sha1": "0fa681a24df1b6ee6960bf1098af9689cfb8a576",
  "data.virustotal.source.alert_id": "1586258410.793188968",
  "data.virustotal.source.file": "/var/www/html/wiki-sief/images/3/38/Pdf-doc-vba-eicar-dropper.pdf",
  "data.virustotal.source.md5": "a1ddc9ebe19a3d43ec25889085ad3ed8",
  "data.virustotal.source.sha1": "0fa681a24df1b6ee6960bf1098af9689cfb8a576",
  "data.virustotal.total": 61,
  "decoder.name": "json",
  "id": "1586258414.793273481",
  "input.type": "log",
  "location": "virustotal",
  "manager.name": "wazuh-server",
  "rule.description": "VirusTotal: Alert - /var/www/html/wiki-sief/images/3/38/Pdf-doc-vba-eicar-dropper.pdf - 40 engines detected this file"
}
    
```

Figura 6.8: Alerta virus con VirusTotal

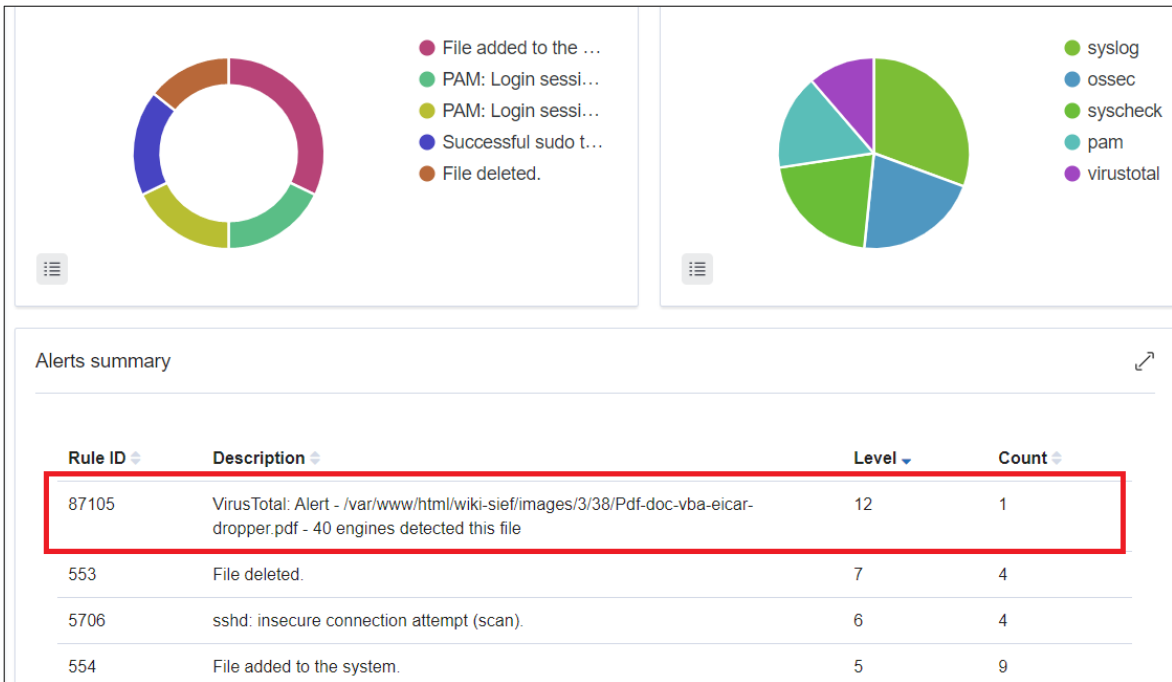


Figura 6.9: Top alertas hacolx08: virus detectado

6.6. Monitorización sin agente (agentless)

En nuestra organización existen dispositivos a los que no se les puede instalar el agente pero que queremos monitorizar. Por ejemplo los switches. En concreto queremos monitorizar los intentos de autenticación erróneos a dichos switches, modelo HP 2530. Lo que haremos será enviar sus logs (restringiendo por `severity=warning` y `facility=auth`) al servidor de syslog de Wazuh, donde los analizaremos y lanzaremos las alertas necesarias.

Para realizar esta tarea nos hemos documentado con el libro *OSSEC Host-Based Intrusion Detection Guide* (Bray et al. 2008), en concreto en el capítulo 4 *Working with rules*, donde se trabaja con los eventos y en la definición de nuevos decodificadores (decoders) y reglas (rules).

El proceso seguido ha sido, en primer lugar, conectarse al switch y configurarlo para que envíe los eventos adecuados al servidor de Wazuh:

```
CHAP-COR-001-SWLAN12# config
CHAP-COR-001-SWLAN12(config)# logging 10.x.x.57 control-descr wazuh-server
CHAP-COR-001-SWLAN12(config)# logging facility auth
CHAP-COR-001-SWLAN12(config)# logging severity warning
CHAP-COR-001-SWLAN12(config)# write mem
```

En dicho servidor habilitamos la captura de logs por syslog en el fichero `/var/ossec/etc/ossec.conf` desde la IP del switch:

```
<remote>
<connection>syslog</connection>
<allowed-ips>10.x.x.246</allowed-ips>
</remote>
```

Y para comprobar que nos están llegando los eventos, en el servidor de Wazuh tenemos que activar temporalmente la opción `logall` en el fichero `/var/ossec/ossec.conf`:

```
<ossec_config>
<global>
...
<logall>yes</logall>
...
</global>
</ossec_config>
```

Una vez hecho esto, ya deberíamos ver los intentos de autenticación erróneos en el switch en el fichero `/var/ossec/logs/archives/archives.log`

```
jpcozar@wazuh-server:/var/ossec/logs/archives$ tail -f archives.log |grep "10.x.x.246"
2020 Apr 07 15:49:00 wazuh-server->10.x.x.246 Apr 7 17:49:00 10.x.x.246 00419 auth: Invalid
↔ user name/password on SSH session User 'admin' is trying to login from 10.x.x.x
```

Pero no nos ha generado ninguna alerta ni en los ficheros `alert.(json|log)` ni en Kibana. Esto es debido a que no existe ningún decodificador ni regla que trate este evento así que tenemos que construir los nuestros.

6.6.1. Creación del decoder

Consideración importante: Hay que tener muy en cuenta al construir el decoder que en el archivo de logs visto anteriormente **se inserta un encabezado** (header) que **no forma parte del log a analizar**. En este caso, aunque la entrada en el archivo de log es:

```
2020 Apr 07 15:49:00 wazuh-server->10.x.x.246 Apr 7 17:49:00 10.x.x.246 00419 auth: Invalid
↳ user name/password on SSH session User 'admin' is trying to login from 10.x.x.x
```

Existe una parte de encabezado que la inserta Wazuh y que hay que eliminar para construir correctamente el decoder:

```
2020 Apr 07 15:49:00 wazuh-server->10.x.x.246
```

Y nos quedaríamos con:

```
Apr 7 17:49:00 10.x.x.246 00419 auth: Invalid user name/password on SSH session User 'admin'
↳ is trying to login from 10.x.x.x
```

También hay que tener cuenta al construir el decoder el espacio en blanco al principio.

Para crear el decoder, abrimos el fichero `/var/ossec/etc/decoders/local_decoder.xml` y creamos el siguiente decodificador:

```
<decoder name="hp">
  <prematch> \w+\s\d+ \d+\d+:\d+\d+:\d+\d+ </prematch>
  <regex offset="after_prematch">(\d+.\d+.\d+.\d+) \d+ auth: Invalid user name/password on SSH
  ↳ session User '(\S+)' is trying to login from (\d+.\d+.\d+.\d+)</regex>
  <order>dstip,user,srcip</order>
</decoder>
```

Las expresiones regulares (OSSEC, 2020)⁵ utilizadas las vemos en la tabla 6.1

Expresiones utilizadas	
\w	A-Z, a-z, 0-9, '-' '@'
\d	0-9
\s	Espacio en blanco
\S	Cualquier carácter excepto \s
Modificadores	
+	una o más ocurrencias (\w+ o \d+)

Tabla 6.1: Expresiones regulares en decoder HP

⁵Para una lista completa de expresiones regulares, consultar <https://www.ossec.net/docs/syntax/regex.html>

En el tag <prematch> tomamos hasta la IP destino, el timestamp: Apr 7 17:49:00. Las expresiones que van entre paréntesis se corresponden con los campos que queremos extraer. En nuestro caso serán la ip de destino (dstip), el usuario que ha intentado logarse (user) y la ip de origen (srcip). Vemos la correspondencia del decoder en la tabla 6.2.

Matching del log (prematch)		
\w+\s+\d+	Apr 7	
\d+\d+:\d+\d+:\d+\d+	17:49:00	
Matching del log (regex)		
(\d+\.\d+\.\d+\.\d+)	10.x.x.246	dstip
\d+	00419	
'(\S+)'	'admin'	user
(\d+\.\d+\.\d+\.\d+)	10.x.x.x	srcip

Tabla 6.2: Matching del decoder

Una vez tenemos el decoder, lo comprobamos con la herramienta /var/ossec/bin/ossec-logtest, pasándole la entrada del log sin el encabezado:

```

**Phase 1: Completed pre-decoding.
full event: ' Apr 7 17:49:00 10.x.x.246 00419 auth: Invalid user name/password
↳ on SSH session User 'admin' is trying to login from 10.x.x.x'
timestamp: '(null)'
hostname: 'wazuh-server'
program_name: '(null)'
log: ' Apr 7 17:49:00 10.x.x.246 00419 auth: Invalid user name/password on SSH
↳ session User 'admin' is trying to login from 10.x.x.x'

**Phase 2: Completed decoding.
decoder: 'hp'
dstip: '10.x.x.246'
dstuser: 'admin'
srcip: '10.x.x.x'

```

Como vemos, utiliza el decoder que hemos creado y obtiene los campos que queríamos.

6.6.2. Creación de reglas: atomic y composite rules

6.6.2.1. Regla atómica: fallos de autenticación

Una regla atómica es aquella que no depende de ninguna otra. En nuestro caso, crearíamos una regla para detectar cada uno de los fallos de autenticación en el switch como eventos independientes.

Para crear dicha regla, editamos el fichero /var/ossec/etc/rules/local_rules.xml y creamos una regla que cuando utilice el decodificador hp, lance una alerta de nivel 5 (según

la configuración por defecto de Wazuh, debe ser mayor de 3 para ser registrada) y con un ID de 100002 (las reglas de usuario deben crearse a partir de la 100000).

```
<group name="hp,sshd,syslog">
  <rule id="100002" level="5">
    <decoded_as>hp</decoded_as>
    <description>sshd: authentication failed HP Switch</description>
    <group>authentication_failed</group>
  </rule>
</group>
```

Y ahora, si lanzamos la herramienta de test de logs de nuevo, se habría añadido una nueva fase que es la que nos indica que nos lanzaría una alerta de nivel 5:

```
**Phase 3: Completed filtering (rules).
Rule id: '100002'
Level: '5'
Description: 'sshd: authentication failed HP Switch'
**Alert to be generated.
```

Una vez hecho esto, si intentamos autenticarnos de forma errónea en el switch, deberíamos ver la alerta tanto en los ficheros alerts. (json|log)

```
root@wazuh-server:/var/ossec/logs/alerts# cat alerts.log|grep "Rule: 100002"

Rule: 100002 (level 5) -> 'sshd: authentication failed HP Switch'
```

como en Kibana, como vemos en la figura 6.10.

```

t _id          KOx4WnEBEFb-HjGB5jPz
t _index       wazuh-alerts-3.x-2020.04.08
# _score       -
t _type        _doc
t agent.id     000
t agent.name   wazuh-server
t data.dstip   10.████████.246
t data.dstuser admin
t data.srcip   10.████████
t decoder.name hp
t full_log     Apr  8 17:46:44 10.████████.246 00419 auth: Invalid user name/password on SSH session User
'admin' is trying to login from 10.████████
t id           1586360803.977720020
t input.type   log
t location     10.████████.246
t manager.name wazuh-server
t rule.description sshd: authentication failed HP Switch
# rule.firedtimes 1
t rule.groups  hp, authentication_failed
t rule.id      100002
# rule.level   5
rule.mail      false
timestamp     Apr 8, 2020 @ 17:46:43.276

```

Figura 6.10: Intento de autenticación erróneo en switch HP

6.6.2.2. Regla compuesta: 5 fallos de autenticación en 10 minutos

Ahora queremos que aparte de generar una alerta por cada fallo de autenticación, si se suceden 5 intentos (por ejemplo desde el mismo usuario o la misma IP) en menos de 10 minutos, lo consideremos como un ataque de fuerza bruta y se lance una alerta con una severidad mayor. Esto será un tipo de regla compuesta, ya que dependemos de otras reglas.

Para ello, editamos el fichero `/var/ossec/etc/rules/local_rules.xml` y creamos una nueva regla, dentro del grupo anterior, con el siguiente contenido:

```

<rule id="100003" level="10" frequency="5" timeframe="600">
  <if_matched_sid>100002</if_matched_sid>
  <same_source_ip />
  <description>5 Failed passwords within 10 minutes HP Switch </description>
  <description>from same IP</description>
  <group>authentication_failures</group>
</rule>

```

Como vemos, para que esta regla se ejecute, se tiene que cumplir la regla atómica con SID 100002 definida anteriormente, con una frecuencia de 5 veces en un tiempo de 600 segundos, o sea 10 minutos desde la misma ip (<same_source_ip />).

Probamos dicha regla primero con el tester, /var/ossec/bin/ossec-logtest, y copiamos el log anterior 5 veces para ver si salta la regla compuesta:

```

**Phase 3: Completed filtering (rules).
Rule id: '100003'
Level: '10'
Description: '5 Failed passwords within 10 minutes HP Switch from same IP'
**Alert to be generated.

```

Como vemos, funciona correctamente y se genera una alerta de nivel 10. Ahora lo ponemos en práctica, intentando logarnos en el switch 5 veces desde la misma IP de forma errónea y vemos que se genera la alerta tanto en los logs:

```

root@wazuh-server:/var/ossec/logs# cat alerts/alerts.log|grep "Rule: 100003"
Rule: 100003 (level 10) -> '5 Failed passwords within 10 minutes HP Switch from same IP'

```

como en Kibana (ver figura 6.11).

```

t data.srcip      10.██████████
t decoder.name   hp
t full_log       Apr 10 11:48:06 10.██████████.246 00419 auth: Invalid user name/password on SSH session User 'admin' is trying to login from
10.██████████
t id             1586512086.434419448
t input.type     log
t location       10.██████████.246
t manager.name   wazuh-server
t previous_output >
Apr 10 11:47:59 10.66.129.246 00419 auth: Invalid user name/password on SSH session User 'admin' is trying to login from
10.██████████
Apr 10 11:47:37 10.66.129.246 00419 auth: Invalid user name/password on SSH session User 'pepe' is trying to login from
10.██████████
Apr 10 11:47:31 10.66.129.246 00419 auth: Invalid user name/password on SSH session User 'pepe' is trying to login from
10.██████████
Apr 10 11:47:24 10.66.129.246 00419 auth: Invalid user name/password on SSH session User 'pepe' is trying to login from
10.██████████
t rule.description 5 Failed passwords within 10 minutes HP Switch from same IP
# rule.firedtimes 1
# rule.frequency 5
t rule.groups      hp, sshd, syslogauthentication_failures
t rule.id          100003
# rule.level       10
# rule.mail        false
t timestamp        Apr 10, 2020 @ 11:48:06.655

```

Figura 6.11: Alerta regla compuesta: 5 intentos de acceso ssh en 10 minutos switch HP

6.7. Active Response

La respuesta activa consiste en que, cuando se detecte un evento, actuemos en consecuencia, no seamos meros espectadores.

Para ello vamos a suponer que, en uno de los servidores Linux, queremos bloquear una IP que intenta conectarse repetidamente a nuestro servidor por SSH durante un tiempo que nosotros establezcamos. Realizaríamos los siguientes pasos.

6.7.1. Detección del ataque

En este caso no hace falta que definamos nuevas reglas (como hicimos para el switch en el apartado 6.6) ya que Wazuh trae reglas por defecto para detectar ataques ssh. En concreto en la ruta `/var/ossec/ruleset/rules` tenemos el fichero `095-sshd_rules.xml`. En él, tenemos dos reglas que nos interesan. La regla con ID 5710, que detecta el intento de logado erróneo por ssh debido a que no existe el usuario:

```
<rule id="5710" level="5">
  <if_sid>5700</if_sid>
  <match>illegal user|invalid user</match>
  <description>sshd: Attempt to login using a non-existent user</description>
  <group>invalid_login,authentication_failed,pci_dss_10.2.4,pci_dss_10.2.5,pci_dss_10.6.1,
  gpg13_7.1,gdpr_IV_35.7.d,gdpr_IV_32.2,hipaa_164.312.b,nist_800_53_AU.14,nist_800_53_AC.7,
  nist_800_53_AU.6,</group>
</rule>
```

Y la regla compuesta con ID 5712, que se lanza si ha saltado la regla 5710 8 veces, en un tiempo de 2 minutos y se ignorará durante 1 minuto.

```
<rule id="5712" level="10" frequency="8" timeframe="120" ignore="60">
<if_matched_sid>5710</if_matched_sid>
<description>sshd: brute force trying to get access to </description>
<description>the system.</description>
<same_source_ip />
<group>authentication_failures,pci_dss_11.4,pci_dss_10.2.4,pci_dss_10.2.5,gdpr_IV_35.7.d,
gdpr_IV_32.2,hipaa_164.312.b,nist_800_53_SI.4,nist_800_53_AU.14,nist_800_53_AC.7,</group>
</rule>
```

Una vez detectado el ataque habría que definir el comando a ejecutar.

6.7.2. Definición del comando

Wazuh viene con una serie de scripts predefinidos para utilizar en las respuestas activas en el directorio `/var/ossec/active-response/bin`.

Utilizaremos el script `firewall-drop.sh` que permite bloquear una IP maliciosa mediante iptables. Una vez tenemos el script, comprobamos que el comando esté definido en el fichero `/var/ossec/etc/ossec.conf`:

```

<command>
  <name>firewall-drop</name>
  <executable>firewall-drop.sh</executable>
  <expect>srcip</expect>
  <timeout_allowed>yes</timeout_allowed>
</command>

```

6.7.3. Definir la respuesta activa

Ahora definimos la respuesta activa en el mismo fichero `ossec.conf` en el servidor de Wazuh. Los elementos principales a incluir son:

command: el comando definido anteriormente `firewall-drop`

location: donde se ejecutará el comando. Como será en el agente que reporta el evento, será `local`.

rules_id: el comando se ejecutará cuando se ejecute la regla con ID 5712.

timeout: bloquearemos la IP durante 300 segundos (5 minutos).

Por tanto la respuesta activa quedará así:

```

<active-response>
  <command>firewall-drop</command>
  <location>local</location>
  <rules_id>5712</rules_id>
  <timeout>300</timeout>
</active-response>

```

Ahora si intentamos logarnos desde la misma IP y fallamos durante 8 veces en menos de 2 minutos, nos bloqueará 5.

En el servidor de Wazuh podemos ver la alerta generada:

```

root@wazuh-server:/var/ossec/logs# cat alerts/alerts.log |grep "Rule: 5712"
Rule: 5712 (level 10) -> 'sshd: brute force trying to get access to the system.'

```

6.7.4. Generar una alerta cuando se dispara una respuesta activa

En el agente que ha ejecutado la respuesta activa podemos verla registrada en el log `/var/ossec/logs/active-responses.log`.

```

vie abr 10 12:47:46 CEST 2020 /var/ossec/active-response/bin/firewall-drop.sh add - 10.x.x.x
↔ 1586515666.494549866 5712
vie abr 10 12:52:47 CEST 2020 /var/ossec/active-response/bin/firewall-drop.sh delete - 10.x.x.x
↔ 1586515666.494549866 5712

```

Como vemos, se ha creado la regla del bloqueo y a los 5 minutos, la de desbloqueo.

Sin embargo, el manager de Wazuh no es consciente de si se ha disparado o no una respuesta activa en los agentes. Para ello, debemos configurar el servidor de Wazuh para que lea el fichero `/var/ossec/logs/active-responses.log` de cada uno de los agentes. Para no tener que ir agente por agente cambiando la configuración, usaremos de nuevo la configuración centralizada vista en el apartado 6.4.

En este caso, como queremos que se aplique al grupo de servidores Linux, editaremos el fichero `/var/ossec/etc/shared/LinuxServers/agent.conf` y añadimos lo siguiente:

```
<localfile>
<log_format>syslog</log_format>
<location>/var/ossec/logs/active-responses.log</location>
</localfile>
```

Reiniciamos el manager de Wazuh y los agentes:

```
root@wazuh-server:# systemctl restart wazuh-manager
root@wazuh-server:# /var/ossec/bin/agent_control -R -a
```

Ahora si volvemos a realizar la misma operación, además, podremos verlo tanto en el fichero de alertas de Wazuh (sabiendo que la regla que saltará será la de ossec de active response, la 601) como en Kibana (figura 6.12).

```
root@wazuh-server:/var/ossec/logs/alerts# cat alerts.log|grep "Rule: 601 "
Rule: 601 (level 3) -> 'Host Blocked by firewall-drop.sh Active Response'
```

```

t data.srcip      10.██████████
t data.type      add
t decoder.name   ar_log
t full_log       vie abr 10 12:47:46 CEST 2020 /var/ossec/active-response/bin/firewall-drop.sh add - 10.██████████ 1586515666.494549866 571
2
t id             1586515666.494552496
t input.type     log
t location       /var/ossec/logs/active-responses.log
t manager.name   wazuh-server
t rule.description Host Blocked by firewall-drop.sh Active Response
# rule.firedtimes 1
t rule.gdpr      IV_35.7.d
t rule.gpg13     4.13
t rule.groups    ossec, active_response
t rule.id        601
# rule.level     3
@ rule.mail      false
t rule.nist_800_53 SI.4
t rule.pci_dss   11.4
📅 timestamp     Apr 10, 2020 @ 12:47:46.199

```

Figura 6.12: Bloqueo de IP con active response

6.8. Bastionado de servidores

Una de las mejores maneras de asegurar los servidores es reduciendo la superficie de vulnerabilidad. Esto se consigue mediante la evaluación de la configuración de la seguridad. A este proceso se le denomina bastionado o hardening. Wazuh dispone de 3 maneras para hacerlo:

SCA módulo desarrollado por el equipo de Wazuh para superar las limitaciones de los otros dos y que ha reemplazado a `rootcheck`, que dependía del servicio `syscheck` cuyos feeds sobre políticas de seguridad estaban desactualizados muy a menudo. Disponible tanto para equipos Windows como Linux.

OpenSCAP sólo disponible para equipos Linux, no para Windows.

CIS-CAT es una herramienta propietaria que depende de una licencia externa para su uso.

En nuestro caso usaremos SCA, por estar incluida con Wazuh, ser open source y cubrir tanto equipos Windows como Linux.

6.8.1. Monitorización de políticas de seguridad con SCA

Cada agente tiene su base de datos local donde almacena el estado actual de cada chequeo: superado (`passed`), fallido (`failed`) o no aplicable (`not-applicable`). De esta forma, los agentes sólo envían las diferencias entre escaneos. Las políticas se encuentran en los agentes en:

- Linux: `/var/ossec/ruleset/sca`
- Windows: `C:\Archivos de programa (x86)\ossec-agent\ruleset\sca`

6.8.1.1. SCA en servidores Windows

Por defecto, el agente Windows solo instala la política `sca_win_audit` que audita sistemas Windows en general pero no específicamente servidores Windows 2012 R2 como es nuestro caso. Para ello, nos descargamos las políticas CIS para servidores Windows 2012 R2, según el rol que tengan (ya sean controladores de dominio o servidores miembro) y según un nivel de perfil L1 o L2 (que extiende el anterior). Podemos encontrar más información en [CIS Microsoft Windows Server 2012 R2](#).

- `cis_win2012r2_domainL1`: CIS benchmark for Windows 2012 R2 Domain Controller L1
- `cis_win2012r2_domainL2`: CIS benchmark for Windows 2012 R2 Domain Controller L2
- `cis_win2012r2_memberL1`: CIS benchmark for Windows 2012 R2 Member Server L1
- `cis_win2012r2_memberL2`: CIS benchmark for Windows 2012 R2 Member Server L2

```
root@wazuh-server:~# cd /var/ossec/etc/shared/; mkdir sca-policies
root@wazuh-server:/var/ossec/etc/shared/sca-policies# wget https://raw.githubusercontent.com/wazuh/wazuh/master/etc/sca/windows/cis_win2012r2_domainL1.yml
...
root@wazuh-server:/var/ossec/etc/shared/sca-policies# chown -R ossec.ossec
↪ /var/ossec/etc/shared/sca-policies
```

Una vez descargadas y establecido el usuario ossec como propietario, las podemos distribuir a los servidores con la configuración centralizada, como vimos en el apartado 6.4. Pero antes, tenemos que crear dos nuevos grupos dentro de los servidores Windows, distinguiendo si son controladores de dominio o servidores miembro. No aplicaremos dichas políticas ni al servidor Windows 2008 ni al servidor independiente.

Los nuevos grupos serán:

2012DomainControllerServers contiene a HACOSV01 y HACOSV02

2012DomainMemberServers contiene a HACONF11, HACONF12, HACOSV03 y HACOSV04.

Ahora ya podemos añadir al fichero de configuración `agent.conf` de cada grupo, las políticas correspondientes. Por ejemplo, para los servidores miembros del dominio, editamos el fichero `/var/ossec/etc/shared/2012DomainMemberServers/agent.conf` y lo dejamos poniendo el path relativo donde se copiarán en los agentes. Será en `C:\Archivos de programa (x86)\ossec-agent\shared` ya que originalmente las busca en `C:\Archivos de programa (x86)\ossec-agent\ruleset\sca`.

```
<agent_config>
<sca>
<policies>
<policy>/../../shared/sca-policies/cis_win2012r2_memberL1.yml</policy>
<policy>/../../shared/sca-policies/cis_win2012r2_memberL2.yml</policy>
</policies>
</sca>
</agent_config>
```

Y para los controladores del dominio, el fichero a editar es `/var/ossec/etc/shared/2012DomainMemberServers/agent.conf` y el contenido:

```
<agent_config>
<sca>
<policies>
<policy>/../../shared/sca-policies/cis_win2012r2_domainL1.yml</policy>
<policy>/../../shared/sca-policies/cis_win2012r2_domainL2.yml</policy>
</policies>
</sca>
</agent_config>
```

Y copiamos las políticas a los directorios de cada grupo:

```
root@wazuh-server:/var/ossec/etc/shared# cp ./sca-policies/*member* ./2012DomainMemberServers
root@wazuh-server:/var/ossec/etc/shared# cp ./sca-policies/*domain*
↪ ./2012DomainControllerServers
```

Estas configuraciones se mezclarán con las existentes en el bloque `sca` de cada agente y se añadirán a la antigua configuración. Al cabo de poco tiempo, tendremos evaluadas dichas políticas, que nos aparecerán en Kibana en una nueva pestaña llamada SCA dentro de cada uno de los agentes en el apartado: Auditing and Policy Monitoring → Security Configuration Assessment.

Como vemos en la figura 6.13 nos muestra las tres políticas SCA que le está aplicando al servidor. La de por defecto y las dos nuevas que hemos añadido.

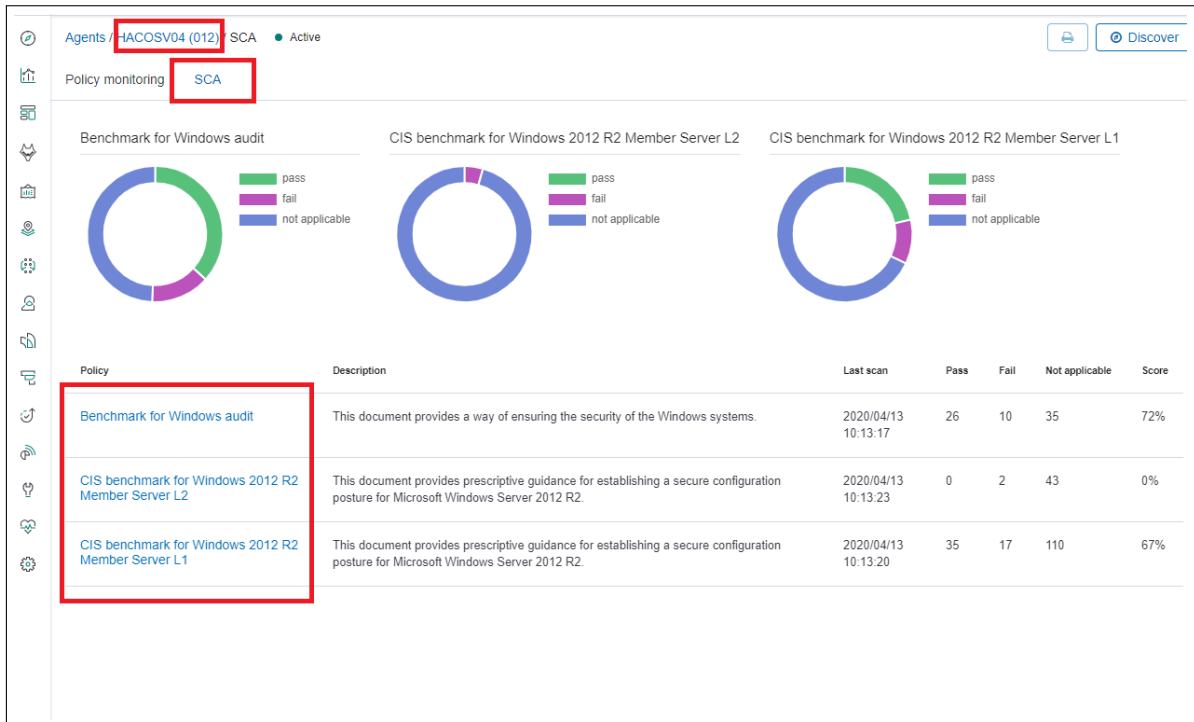


Figura 6.13: Políticas SCA servidor miembro Windows 2012 R2

Si hacemos clic en cada política, vemos en detalle el resultado de cada test: failed, passed o not applicable. Esto lo vemos en la figura 6.14.

6.8.1.2. SCA en servidores Linux

En el caso de los servidores Linux no hay que hacer ningún paso adicional ya que las políticas que vienen por defecto para Debian, son compatibles con Ubuntu. En el caso de los servidores Ubuntu 16.x se han aplicado las políticas:

- CIS benchmark for Debian/Linux 9 L1
- CIS benchmark for Debian/Linux 9 L2

Y en el caso del único Ubuntu 14.x las políticas:

- CIS benchmark for Debian/Linux 8 L1
- CIS benchmark for Debian/Linux 8 L2

Un ejemplo de cada una lo podemos ver en las imágenes 6.15 y 6.16 respectivamente.

CAPÍTULO 6. CASOS DE USO

Agents / HACOSV04 (012) / SCA ● Active

Policy monitoring SCA

← CIS benchmark for Windows 2012 R2 Member Server L1 Pass: 35 Fail: 17 Not applicable: 110 Score: 67% 2020-04-13

Policy checksum: a5078795cd577f4b6f611c8d512ba03bfc03195e467919e5f787877d17ccc474

Filter checks...

ID	Title	File	Result
12161	Ensure 'No auto-restart with logged on users for scheduled automatic updates installations' is set to 'Disabled'	-	failed
12160	Ensure 'Configure Automatic Updates: Scheduled install day' is set to '0 - Every day'	-	passed
12159	Ensure 'Configure Automatic Updates' is set to 'Enabled'	-	passed
12158	Ensure 'Disallow WinRM from storing RunAs credentials' is set to 'Enabled'	-	Not applicable
12157	Ensure 'Allow unencrypted traffic' is set to 'Disabled'	-	Not applicable
12156	Ensure 'Allow Basic authentication' is set to 'Disabled'	-	Not applicable
12155	Ensure 'Disallow Digest authentication' is set to 'Enabled'	-	Not applicable
12154	Ensure 'Allow unencrypted traffic' is set to 'Disabled'	-	Not applicable
12153	Ensure 'Allow Basic authentication' is set to 'Disabled'	-	Not applicable
12152	Ensure 'Turn on PowerShell Transcription' is set to 'Disabled'	-	Not applicable
12151	Ensure 'Turn on PowerShell Script Block Logging' is set to 'Disabled'	-	Not applicable
12150	Ensure 'Sign-in last interactive user automatically after a system-initiated restart' is set to 'Disabled'	-	passed
12149	Ensure 'Always install with elevated privileges' is set to 'Disabled'	-	Not applicable
12148	Ensure 'Allow user control over installs' is set to 'Disabled'	-	Not applicable

Figura 6.14: SCA - CIS benchmark W2012R2 Member Server L1

Agents / hacolx05 (001) / SCA ● Active

Policy monitoring System auditing SCA

← CIS benchmark for Debian/Linux 9 L1 Pass: 37 Fail: 60 Not applicable: 2 Score: 38% 2020-04-1

Policy checksum: b24b5a4ae8dbce8e6f3e8cca85ef5a64901699d76fc0973dce68d14599a31e4c

Filter checks...

ID	Title	File	Result
3098	Ensure shadow group is empty	/etc/group	passed
3097	Ensure root is the only UID 0 account	/etc/passwd	passed
3096	Ensure no legacy "+" entries exist in /etc/group	/etc/group	passed
3095	Ensure no legacy "+" entries exist in /etc/shadow	/etc/shadow	passed
3094	Ensure no legacy "+" entries exist in /etc/passwd	/etc/passwd	passed
3093	Ensure password fields are not empty	/etc/shadow	passed
3092	Ensure access to the su command is restricted	/etc/pam.d/su	failed
3091	Ensure default group for the root account is GID 0	/etc/passwd	passed
3090	Ensure inactive password lock is 30 days or less	-	failed
3089	Ensure password expiration warning days is 7 or more	/etc/login.defs	passed
3088	Ensure minimum days between password changes is 7 or more	/etc/login.defs	failed
3087	Ensure password expiration is 365 days or less	/etc/login.defs	failed
3086	Ensure password reuse is limited	/etc/pam.d/common-password	failed
3085	Ensure lockout for failed password attempts is configured	/etc/pam.d/common-auth	failed

Figura 6.15: SCA - CIS benchmark Debian 9 L1

Agents / haco1x09 (004) / SCA ● Active

Policy monitoring System auditing SCA

← CIS benchmark for Debian/Linux 8 L1 Pass: 25 Fail: 58 Not applicable: 18 Score: 30% 2020-04

Policy checksum: 4f6b33d62bcd1e809137af4c2ba33d5bac31ec5ce28ee1dc571d6e2f941fb07

Filter checks...

ID	Title	File	Result
2100	Ensure shadow group is empty	/etc/group	passed
2099	Ensure root is the only UID 0 account	/etc/passwd	passed
2098	Ensure no legacy "+" entries exist in /etc/group	/etc/group	passed
2097	Ensure no legacy "+" entries exist in /etc/shadow	/etc/shadow	passed
2096	Ensure no legacy "+" entries exist in /etc/passwd	/etc/passwd	passed
2095	Ensure password fields are not empty	/etc/shadow	passed
2094	Ensure access to the su command is restricted	/etc/pam.d/su	failed
2093	Ensure default group for the root account is GID 0	/etc/passwd	passed
2092	Ensure inactive password lock is 30 days or less	-	failed
2091	Ensure password expiration warning days is 7 or more	/etc/login.defs	passed
2090	Ensure minimum days between password changes is 7 or more	/etc/login.defs	failed
2089	Ensure password expiration is 365 days or less	/etc/login.defs	failed
2088	Ensure password reuse is limited	/etc/pam.d/common-password	failed
2087	Ensure lockout for failed password attempts is configured	/etc/pam.d/common-auth	failed

Figura 6.16: SCA - CIS benchmark Debian 8 L1

6.9. Notificaciones de alertas y generación de informes

Todo sistema de monitorización debe ser capaz de realizar algún tipo de notificación para que los administradores sean conscientes de que se ha disparado alguna alerta o para tener un informe general de ellas. En el caso de Wazuh, podemos configurar tanto un informe diario de alertas como correos específicos cuando se dispara una alerta según su nivel, ID, pertenencia a un grupo de agentes, etc. . .

Wazuh no permite el uso de servidores de correo con autenticación así que es necesario utilizar un servidor relay. Como ese problema ya lo teníamos resuelto en nuestra organización, disponemos de un servidor que hace de relay de correo electrónico, haco1x05, que es el que utilizaremos para el envío de las notificaciones.

6.9.1. Notificación de alertas

Habilitamos las alertas en el fichero `/var/ossec/etc/ossec.conf`:

```
<ossec_config>
<global>
...
<email_notification>yes</email_notification>
<smtp_server>haco1x05</smtp_server>
<email_from>usuario_envio@organizacion.es</email_from>
<email_to>usuario_recepcion@organizacion.es</email_to>
```

```

<email_maxperhour>12</email_maxperhour>
<email_log_source>alerts.log</email_log_source>
</global>
...
</ossec_config>

```

Y luego definimos el nivel de alertas a partir del cual, queremos que se nos envíen mensajes en el mismo fichero. Por ejemplo, como queremos que se nos envíen las alertas de los intentos de ataque de fuerza bruta por ssh que hicimos en la sección 6.7 que tenía un nivel 10, lo dejamos así:

```

<alerts>
  <email_alert_level>10</email_alert_level>
</alerts>

```

Con la configuración anterior, todas las alertas de nivel ≥ 10 le llegarían al usuario indicado. Wazuh permite una mayor granularidad y podríamos hacer que las alertas de un determinado servidor le llegaran a un correo específico. Por ejemplo, realizamos el ataque ssh en el servidor hacolx07 y queremos que le llegue al usuario usuario_recepcion. En el mismo fichero haríamos:

```

<email_alerts>
  <event_location>hacosv07</event_location>
  <do_not_delay />
</email_alerts>

```

Probamos a hacer el ataque ssh y recibiríamos el correo que vemos en la figura 6.17

6.9.2. Envío de informes diarios

Podemos enviar un informe diario con un resumen de las alertas que se han lanzado cada día y personalizarlo según nuestras necesidades. Editamos el fichero `/var/ossec/etc/ossec.conf` y configuramos el apartado de informes así:

```

<ossec_config>
<reports>
<level>10</level>
<title>Informe diario: Alertas con nivel mayor que 10</title>
<email_to>informe@correo-organizacion.es</email_to>
</reports>
</ossec_config>

```

Esta configuración enviaría un informe diario de las alertas generadas con un nivel ≥ 10 , como vemos en la figura 6.18.



Figura 6.17: Alerta de ataque por fuerza bruta ssh por correo

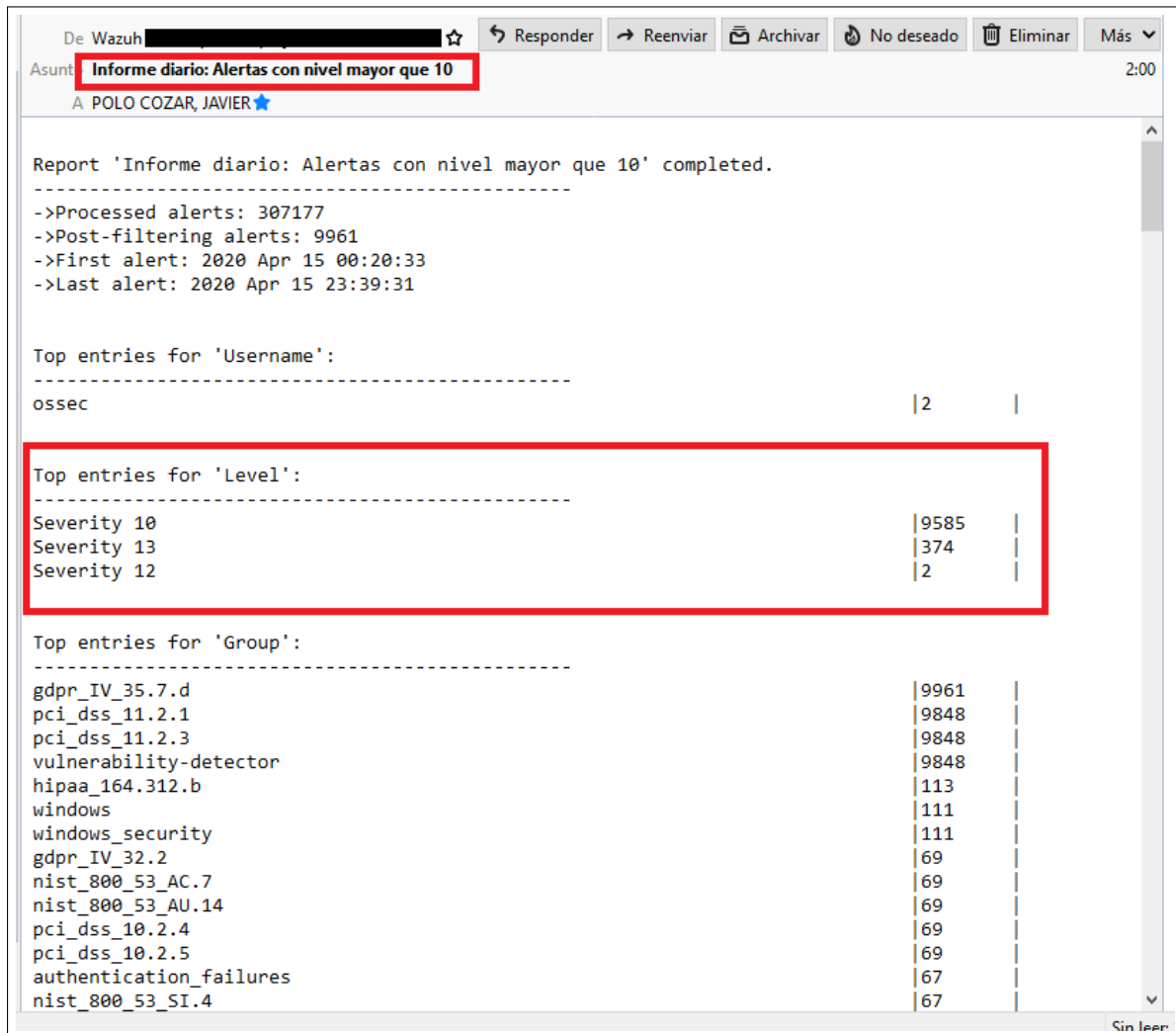


Figura 6.18: Informe diario alertas nivel 10 o más Wazuh

6.10. Integración con APIs externas: Slack

Wazuh también permite conectarnos a APIs externas y herramientas de alerta como [Slack](#), [PagerDuty](#) o [VirusTotal](#) (visto en el apartado 6.5). En nuestra organización usamos Slack como herramienta colaborativa del Departamento de Informática, así que vamos a integrar Wazuh con ella, creando un nuevo canal en nuestro espacio de trabajo de Slack llamado #wazuh y recibiendo ahí las alertas que genere.

La conexión se hará a través de la tecnología [Incoming Webhooks](#). Necesitamos crear uno, que nos dará una URL única (llamada `hook_url`) donde enviaremos un [JSON](#) con el mensaje de texto y algunas opciones adicionales⁶. Una vez tenemos la `hook_url`, en el fichero `/var/ossec/etc/ossec.conf` configuramos una nueva integración, añadiéndole los filtros correspondientes: `rule_id`, `level`, `group`, `event_location`. En nuestro caso, filtraremos por las alertas que tengan un nivel ≥ 12 :

```
<integration>
<name>slack</name>
<hook_url>https://hooks.slack.com/services/"hook_url_individual"</hook_url>
↵
<level>12</level>
```

Una vez configurada la integración, reiniciamos el manager de Wazuh:

```
root@wazuh-server:# systemctl restart wazuh-manager
```

Y al cabo de poco tiempo nos llegarán al canal #wazuh de nuestro espacio de trabajo las notificaciones con el nivel de alerta indicado, como podemos ver en la figura 6.19.



Figura 6.19: Notificación en Slack de alertas de Wazuh

⁶Más información en <https://api.slack.com/messaging/webhooks>



CONCLUSIONES Y MEJORAS

7.1. Conclusiones

Se han realizado los siguientes objetivos propuestos:

- ✓ Se ha podido seguir la planificación establecida.
- ✓ Se ha estudiado y determinado el mejor método de implantación de Wazuh en nuestra organización.
- ✓ Se han instalado los servidores Wazuh y ELK Stack con una arquitectura distribuida.
- ✓ Se han securizado los servidores Wazuh y ELK Stack (incluido la autenticación en el acceso al portal web Kibana) y las comunicaciones entre ellos utilizando la tecnología X-Pack para la parte de ELK.
- ✓ Se han integrado los servidores Wazuh y Elastic en la herramienta Nagios de monitorización de nuestra organización.
- ✓ Se han desplegado de forma automática los agentes de Wazuh tanto en servidores Windows (mediante directivas de grupo) como Linux (mediante Ansible).
- ✓ Se ha utilizado la configuración centralizada que propone Wazuh para distribuir opciones específicas a los agentes de forma automática en vez de ir uno por uno de forma manual.
- ✓ Se han monitorizado ficheros considerados críticos, alertándonos de su alteración en tiempo real cuando ha sido posible. También se ha realizado la auditoría de dichas modificaciones,

permitiéndonos conocer quién, cómo, cuándo y de qué manera se realizó la modificación (who-data), tanto en Windows como en Linux.

- ✓ Se han monitorizado dispositivos que no permiten la instalación de agentes específicos (agentless) (switches HP) mediante el servidor syslog de Wazuh y creando decodificadores y reglas específicos.
- ✓ Se ha integrado la herramienta VirusTotal con Wazuh para un análisis en tiempo real de determinados directorios.
- ✓ Se han podido detectar vulnerabilidades tanto en sistemas Windows (mediante la base de datos NVD) como en Linux (mediante los OVAL proporcionados por Canonical).
- ✓ Se han definido mecanismos de respuesta activa para responder a determinadas situaciones (ataque ssh).
- ✓ Se ha utilizado la tecnología SCA para el análisis de las políticas de seguridad, tanto en servidores Windows como Linux.
- ✓ Se han configurado los sistemas de notificaciones tanto por correo electrónico como con herramientas externas (Slack).

Además este proyecto nos ha servido para:

- Implementar en nuestra organización un SIEM de software libre del que carecía y ver las enormes posibilidades que ofrece.
- Encontrar un sistema versátil, ampliable y/o adaptable a nuestras necesidades mediante la activación/desactivación de los diferentes módulos con los que cuenta, creación de reglas y decodificadores específicos, dashboards personalizados, etc. . .
- Ver las posibilidades de integración con herramientas y APIs externas (VirusTotal y Slack, respectivamente y en nuestro caso) que incrementan su valor como SIEM.
- Apreciar en su justo valor las herramientas de automatización y de configuración centralizada, que nos ayudan a un despliegue rápido y eficiente.
- Contribuir a dicho proyecto reportando los bugs encontrados.
- Instalar una herramienta que nos ayude a cumplir con el ENS.

7.2. Mejoras

- La instalación por defecto de Wazuh genera muchas alertas e información de monitorización (por ejemplo cada logado con éxito en un sistema Windows) que es necesario que depuremos para separar lo realmente importante y no sobrecargar el sistema así como eliminar los falsos positivos.
- Al cabo de poco tiempo de estar en producción el sistema (1 mes aproximadamente), el espacio ocupado por los datos de Elasticsearch en /var/data/elasticsearch, ha crecido bastante, con lo que hemos tenido que migrar los datos a una nueva unidad virtual /mnt/es_data/lib/elasticsearch para mantenerla independiente del servidor y monitorizada por Nagios. Además hemos tenido que configurar la rotación y eliminación de índices a través de Kibana. Es necesaria una planificación y estudio más exhaustivo de este tema para evitar cuellos de botella.

- Existen características adicionales de Wazuh (por ejemplo [Osquery](#)) que podrían sernos útiles en nuestro entorno y que implementaremos en el futuro.
- Será necesario personalizar Kibana generando dashboards propios para nuestra organización que nos avisen de alertas definidas por nosotros (por ejemplo los ataques por ssh a los servidores Linux, o intentos de logado erróneo a los switches, etc. . .).
- Habrá que aplicar el caso de uso del switch HP (sección 6.6) al resto de switches de la organización y crear reglas o decodificadores adicionales si es necesario.
- Estudiar la posibilidad de utilizar Wazuh para monitorizar los dispositivos en los que compartimos la gestión (firewall y cabina de almacenamiento), previo acuerdo con el resto de administradores.
- Realizar un manual o guía de usuario del entorno web instalado (Kibana+plugin Wazuh) para los operadores e implementar dashboards preparados específicamente para ellos.

BIBLIOGRAFÍA

Libros

- Bray, R., Cid, D. y Hay, A. (2008, 9 de abril). *OSSEC Host-Based Intrusion Detection Guide* (Edición: Pap/Cdr). Syngress. (Vid. pág. 62).
- Chhajed, S. (2015, 26 de noviembre). *Learning ELK Stack: Build mesmerizing visualizations, analytics, and logs from your data using Elasticsearch, Logstash, and Kibana*. Packt Publishing.
- Gormley, C. y Tong, Z. (2015, 7 de febrero). *Elasticsearch: The Definitive Guide* (Edición: 1). O'Reilly Media.
- Kruegel, C., Valeur, F. y Vigna, G. (2005). *Intrusion detection and correlation: Challenges and solutions*. Springer US. <https://doi.org/10.1007/b101493>
- Lhotsky, B. (2013, 1 de enero). *Instant OSSEC host-based intrusion detection system* [Google-Books-ID: X80hLF8zL4EC]. Packt Publishing Ltd.
- Miell, I. y Sayers, A. H. (2019, 28 de febrero). *Docker in Practice, Second Edition* (Edición: 2nd). Manning.
- Miller, D. R., Harris, S., Harper, A., VanDyke, S. y Blask, C. (2010, 5 de noviembre). *Security Information and Event Management* (Edición: 1). McGraw-Hill Education.
- Northcutt, S. y Novak, J. (2002, 6 de septiembre). *Network Intrusion Detection* (3 edition). Sams Publishing.
- Paro, A. (2019, 30 de abril). *Elasticsearch 7.0 Cookbook: Over 100 recipes for fast, scalable, and reliable search for your enterprise, 4th Edition*. Packt Publishing.
- Shukla, P. y N, S. K. M. (2019, 31 de mayo). *Learning Elastic Stack 7.0: Distributed search, analytics, and visualization using Elasticsearch, Logstash, Beats, and Kibana, 2nd Edition* (Edición: 2). Packt Publishing.

Artículos

- Mokalled, H., Catelli, R., Casola, V., Debortol, D., Meda, E. y Zunino, R. (2019). The Applicability of a SIEM Solution: Requirements and Evaluation. *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2019 IEEE 28th International Conference on, WETICE*, 132-137. <https://doi.org/10.1109/WETICE.2019.00036> (vid. pág. 7)
- Podzins, O. y Romanovs, A. (2019). Why SIEM is Irreplaceable in a Secure IT Environment? *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream), Electrical, Electronic and Information Sciences (eStream), 2019 Open Conference of*, 1-5. <https://doi.org/10.1109/eStream.2019.8732173> (vid. pág. 1)

Recursos web

- A2Secure. (2019, 12 de febrero). *Sistemas IDS, IPS, HIDS, NIPS, SIEM ¿Qué son?* [A2Secure] [Library Catalog: www.a2secure.com Section: Blog]. Consultado el 3 de marzo de 2020, desde <https://www.a2secure.com/blog/ids-ips-hids-nips-siem-que-es-esto/>
- Casetto, O. (2019, 7 de febrero). *A SIEM security primer: Evolution and next-gen capabilities* [Exabeam] [Library Catalog: www.exabeam.com Section: SIEM]. Consultado el 28 de febrero de 2020, desde <https://www.exabeam.com/siem/a-siem-security-primer-evolution-and-next-gen-capabilities/>. (Vid. pág. 8)
- OSSEC. (2020). *OSSEC HIDS documentation*. Consultado el 25 de febrero de 2020, desde <https://ossec-documentation.readthedocs.io/en/latest/>. (Vid. pág. 63)
- Red Hat. (2020). *Ansible Documentation*. Consultado el 25 de febrero de 2020, desde https://docs.ansible.com/ansible/latest/user_guide/. (Vid. pág. G-1)
- Verizon. (2019, mayo). *2019 data breach investigations report* [Verizon enterprise solutions] [Library Catalog: enterprise.verizon.com]. Consultado el 25 de febrero de 2020, desde <https://enterprise.verizon.com/resources/reports/2019-data-breach-investigations-report.pdf>. (Vid. pág. 1)
- Wazuh. (2020). *Wazuh documentation*. Consultado el 25 de febrero de 2020, desde <https://documentation.wazuh.com/3.11/>. (Vid. págs. 29, 34, D-4)

Apéndices



HYPER-V

A.1. Creación de switch virtual en Hyper-V

En *Hyper-V* por defecto al crear una nueva máquina virtual, se utiliza la conexión de red Ethernet llamada *Default Switch* que es de tipo NAT¹, lo cual hace que el servidor Wazuh no sea accesible por los agentes. Por tanto, antes de crear la máquina virtual, tenemos que crear un nuevo switch virtual si no está ya creado. En nuestra organización en los servidores en producción ya existe, pero en pruebas en nuestro equipo no. Para hacerlo, realizamos los siguientes pasos:

1. Abrimos el Administrador de Hyper-V
2. Administrador de conmutadores virtuales
3. Tipo de conexión: Externo
4. Nombre: External vSwitch
5. ¿Con qué desea conectar este conmutador virtual?
 - Red externa: Realtek PCIe GBe Family Controller
6. Marcar: Permitir que el sistema operativo de administración comparta este adaptador de red

Debe quedarnos como se indica en la figura [A.1](#)

Una vez creado, cambiaremos el adaptador de red en la máquina virtual de *Default Switch* a *External vSwitch* y ya debe darnos una IP del rango de la sede, no una dirección privada.

¹Network Address Translation - Traducción de direcciones de red

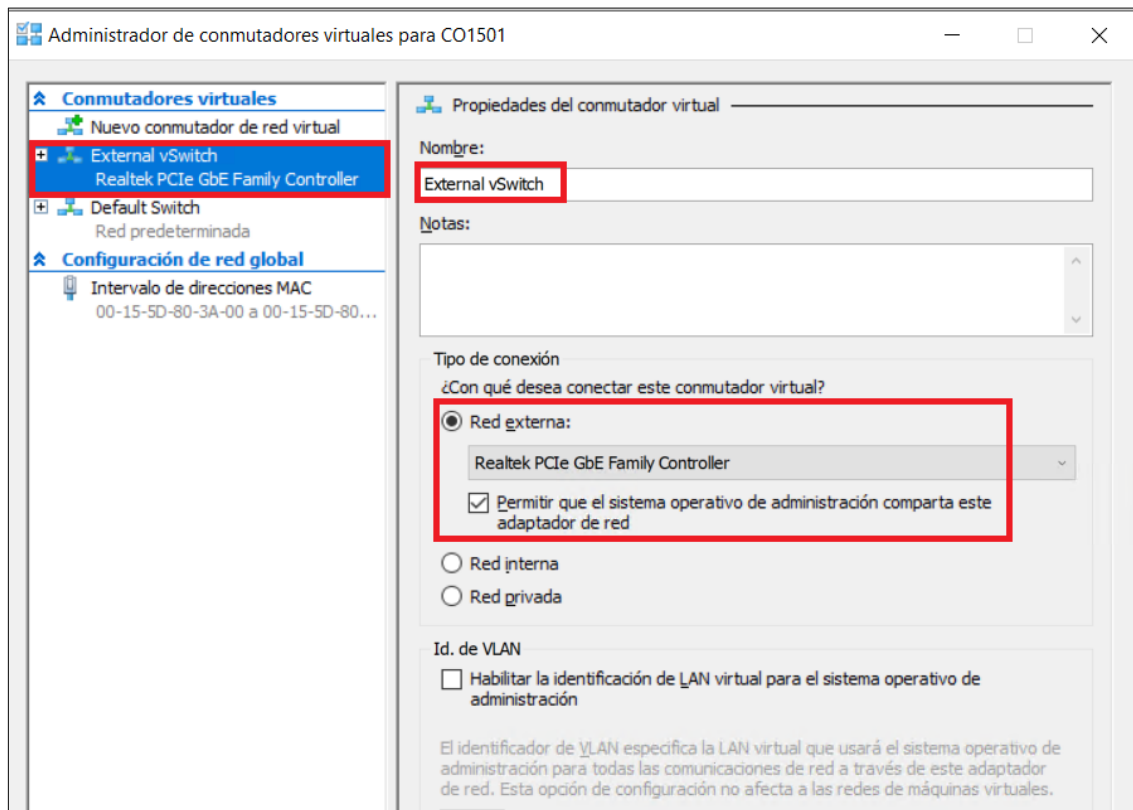


Figura A.1: Creación de switch virtual en Hyper-V

A.2. Creación de máquinas virtuales

A.2.1. Servidor Wazuh en pruebas

1. Abrimos el Administrador de Hyper-V de nuestro equipo
2. Creamos una máquina virtual con las siguientes características:
 - Nombre: wazuh-server-pruebas
 - Ubicación: D:\Wazuh-pruebas
 - Generación: 1
 - Procesadores: 1 procesador virtual
 - Memoria RAM: 2 GB
 - Marcar: Usar la memoria dinámica para esta máquina virtual
 - Conexión: External vSwitch (creado en el apartado A.1).
 - Crear un disco duro virtual: Wazuh-server-pruebas.vhdx - 40 GB
 - Instalar un sistema operativo desde un CD/DVD ROM de arranque - Archivo de imagen ISO: ubuntu-18.04.4-live-server-amd64.iso
 - Arrancamos la máquina virtual y debe empezar a instalar la ISO de Ubuntu Server
 - Seguir los pasos indicados en el anexo B

A.2.2. Servidor Wazuh en producción

1. Abrimos el Administrador de Hyper-V del servidor
2. Creamos una máquina virtual con las siguientes características:
 - Nombre: Wazuh Server
 - Ubicación: C:\ClusterStorage\Volume2
 - Generación: 1
 - Procesadores: 4 procesadores virtuales
 - Memoria RAM: 4 GB
 - Marcar: Usar la memoria dinámica para esta máquina virtual
 - Conexión: vSwitch1
 - Crear un disco duro virtual: Wazuh-server.vhdx - 40 GB
 - Instalar un sistema operativo desde un CD/DVD ROM de arranque. Archivo de imagen ISO: ubuntu-18.04.4-live-server-amd64.iso
 - Arrancamos la máquina virtual y debe empezar a instalar la ISO de Ubuntu Server
 - Seguir los pasos indicados en el anexo B

A.2.3. Servidor ELK Stack en pruebas

1. Abrimos el Administrador de Hyper-V
2. Creamos una máquina virtual con las siguientes características:
 - Nombre: ELk-stack-pruebas
 - Ubicación: D:\Wazuh-Elk-Pruebas
 - Generación: 1
 - Procesadores: 1 procesador virtual
 - Memoria RAM: 3 GB
 - Marcar: Usar la memoria dinámica para esta máquina virtual
 - Conexión: External Switch (creado en el apartado A.1).
 - Crear un disco virtual: ELK-stack-pruebas.vhdx - 40 GB
 - Instalar un sistema operativo desde un CD/DVD ROM de arranque. Archivo de imagen ISO: ubuntu-18.04.4-live-server-amd64.iso
 - Arrancamos la máquina virtual y debe empezar a instalar la ISO de Ubuntu Server
 - Seguir los pasos indicados en el anexo B

A.2.4. Servidor ELK Stack en producción

1. Abrimos el Administrador de Hyper-V
2. Creamos una máquina virtual con las siguientes características:
 - Nombre: ELk Server
 - Ubicación: C:\ClusterStorage\Volume2
 - Generación: 1
 - Procesadores: 4 procesadores virtuales
 - Memoria RAM: 8 GB
 - Marcar: Usar la memoria dinámica para esta máquina virtual
 - Conexión: vSwitch1
 - Crear un disco virtual: ELK-server.vhdx - 40 GB

- Instalar un sistema operativo desde un CD/DVD ROM de arranque. Archivo de imagen ISO: ubuntu-18.04.4-live-server-amd64.iso
- Arrancamos la máquina virtual y debe empezar a instalar la ISO de Ubuntu Server
- Seguir los pasos indicados en el anexo B

A.3. Clúster de conmutación por error: alta disponibilidad

Para configurar la alta disponibilidad y que las máquinas virtuales en producción (tanto el servidor Wazuh con el de la pila ELK) pasen de un nodo físico a otro en el caso de que haya algún problema (caída, parada, reinicio programado, etc. . .), es necesario que las añadamos al *Administrador de clúster de conmutación por error* de Windows 2012 R2.

Una vez creadas las máquinas, nos vamos al Administrador de clúster de conmutación por error y le indicamos que queremos crear un nuevo rol. Nos aparece el asistente de la figura A.2, donde le indicamos que queremos configurar la alta disponibilidad para una máquina virtual.

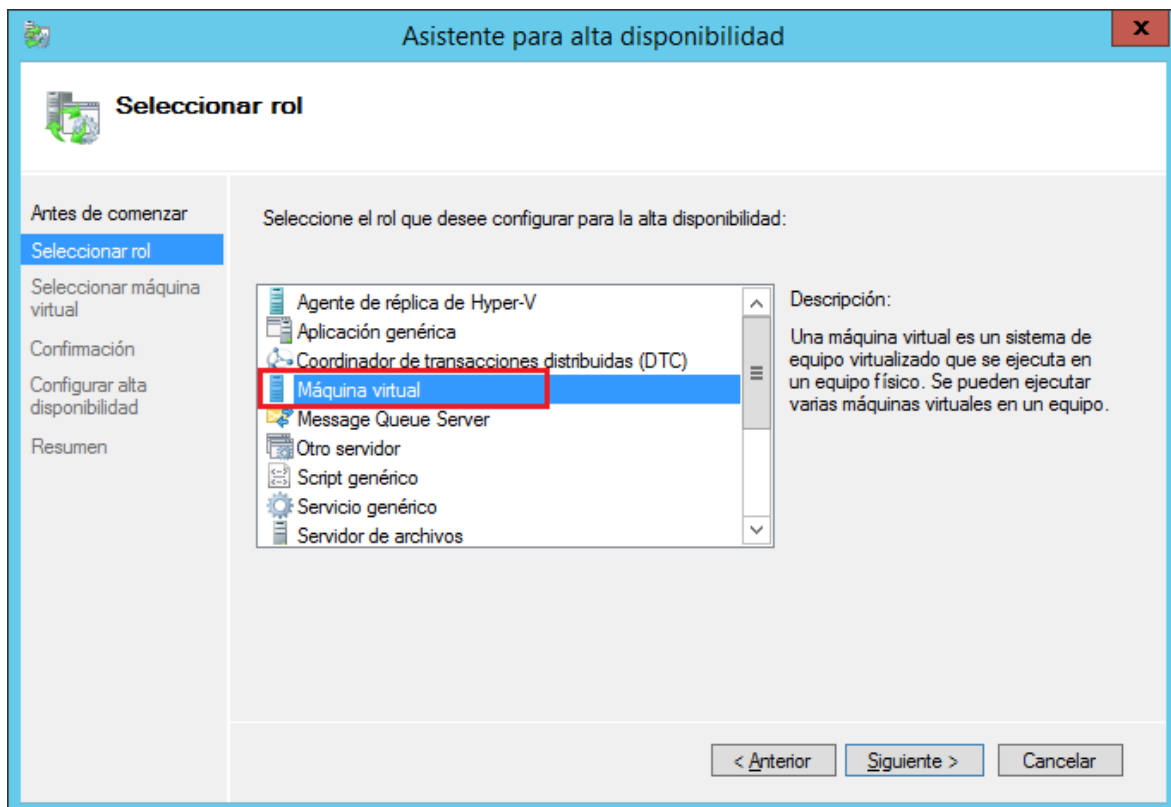


Figura A.2: Seleccionar rol para configurar alta disponibilidad

A continuación, seleccionamos las dos máquinas que queremos que tengan alta disponibilidad, como vemos en la figura A.3

Por último, comprobamos que se han agregado dos nuevos roles en el administrador de clúster de conmutación por error, uno por cada máquina virtual como vemos en la figura A.4.

Ahora, en el caso de que se produzca una caída del nodo propietario de las máquinas virtuales, se transferirán dichas máquinas al otro nodo.

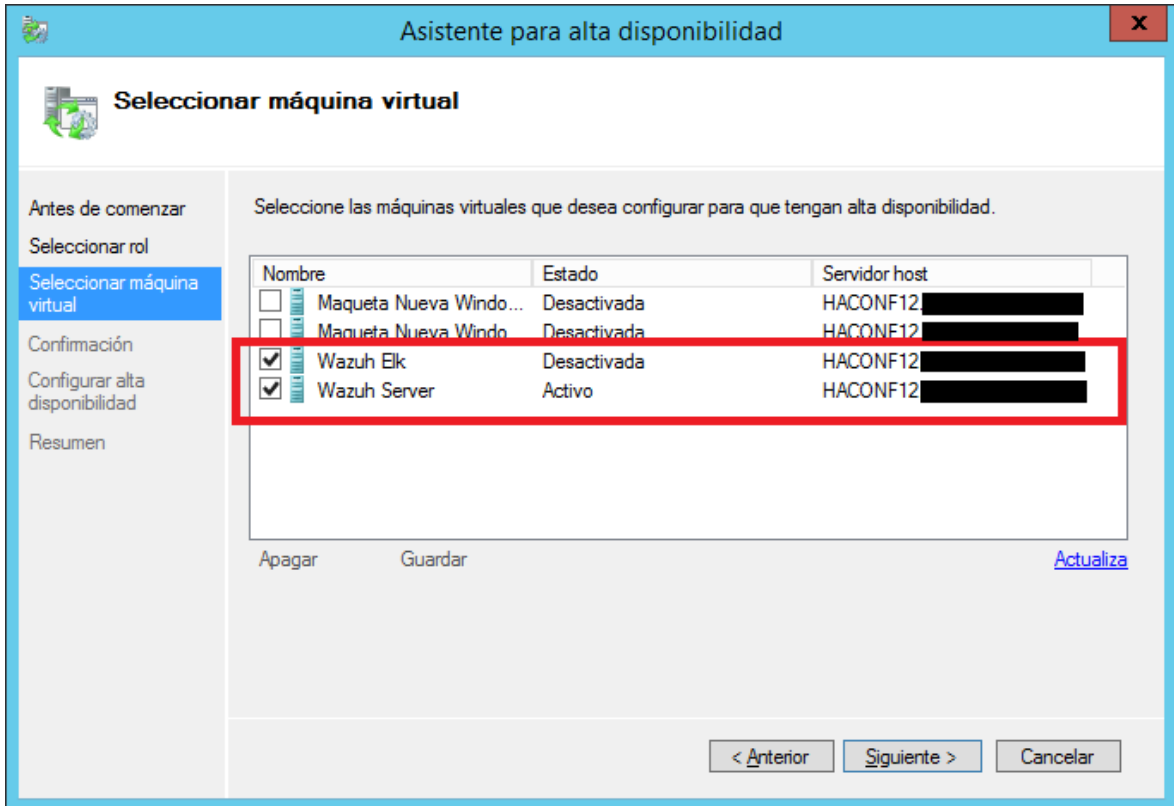


Figura A.3: Máquinas con alta disponibilidad

Roles (12)				
<input type="text" value="Buscar"/>				
Nombre	Estado	Tipo	Nodo propietario	Prioridad
Wazuh Server	En ejecución	Máquina virtual	HACONF12	Media
Wazuh Elk	En ejecución	Máquina virtual	HACONF12	Media

Figura A.4: Roles creados para alta disponibilidad



UBUNTU SERVER

18.04.4 LTS

B.1. Instalación Ubuntu Server 18.04.4 LTS en Hyper-V

Los pasos para instalar Ubuntu Server 18.04.4 LTS son:

- Descargamos la última versión estable de Ubuntu Server: *ubuntu-18.04.4-live-server-amd64.iso*
- Creamos la máquina virtual en Hyper-V como se indica en el anexo A
- Arrancamos la máquina virtual y nos salta el instalador de Ubuntu Server
- Idioma: Español
- Layout: Español
- Variant: Español
- Conexión de red: eth0
 - Subnet: 10.x.x.0/23
 - Address: dependiendo de la máquina virtual que estemos instalando, será:
 - Elk Stack-pruebas: 10.x.x.x
 - Wazuh-server-pruebas: 10.x.x.x
 - ELK Stack producción: 10.x.x.40
 - Wazuh-server producción: 10.x.x.57
 - Gateway: 10.x.x.x
 - Name Servers: 10.x.x.x, 10.x.x.x
 - Search domains: <dominio>
- Proxy address: `http://10.x.x.x:3128`

- Mirror address: <http://archive.ubuntu.com/ubuntu>
- Installer update available: Continue without updating
- Configuración del sistema de archivos: Use An Entire Disk
- Seleccionamos el disco local
- Dejamos las particiones creadas por defecto
- Your name: Javier Polo Cózar
- Your server's name: dependiendo de la máquina virtual que estemos instalando, será:
 - wazuh-server-pruebas
 - wazuh-elk-pruebas
 - wazuh-server
 - wazuh-elk
- pick a username: jpcozar
- Password y confirmación
- Marcar: Install Open SSH server
- Import SSH identity: No
- No marcar ningún snap

Una vez instalado, actualizamos los paquetes

```
jpcozar@wazuh-elk-pruebas:~$ sudo apt-get update
jpcozar@wazuh-elk-pruebas:~$ sudo apt-get upgrade
```

B.2. Configurar Ubuntu 18.04.4 LTS

B.2.1. Deshabilitar cloud-init

cloud-init es un servicio en la nube que no necesitamos y ralentiza el inicio del sistema así que lo deshabilitamos.

```
jpcozar@wazuh-elk-pruebas:~$ sudo touch /etc/cloud/cloud-init.disabled
```

B.2.2. Configuración de la red

Si tuviéramos que cambiar la configuración de red (IP estática, DNS, gateway) en Ubuntu 18.x está en el fichero `/etc/netplan/50-cloud-init.yaml`.

Para wazuh-elk-pruebas sería:

```
network:
  ethernets:
    eth0:
      addresses:
        - 10.x.x.x/23
      gateway4: 10.x.x.x
      nameservers:
        addresses:
          - 10.x.x.x
          - 10.x.x.x
```

```
search:
- <dominio>
version: 2
```

Para aplicar la configuración de red si cambiamos el fichero anterior, sería:

```
jpcozar@wazuh-elk-pruebas:~$ sudo netplan apply
```

B.2.3. Configuración del FQDN - DNS

Para las máquinas en producción, es necesario que, aparte de la IP, tengan configurado correctamente su nombre de dominio. Esto será necesario para la creación de certificados, resolución de nombres directa e inversa, etc. . . . Para ello, hacemos lo siguiente:

- Abrimos el Administrador de DNS
- Abrimos el Servidor de DNS
- Abrimos las Zonas de búsqueda directa
- Abrimos nuestro dominio <dominio>
- Creamos un nuevo host, pulsando botón derecho: Host nuevo (A o AAAA)
- Introducimos los datos para el servidor de Wazuh (como vemos en la figura B.1)
- Introducimos los datos para el servidor de ELK (como vemos en la figura B.2)
- Comprobamos tanto en la zona de búsqueda directa como en la zona de búsqueda inversa que se han creado los registros (por ejemplo, en la figura B.3 vemos el registro creado en la zona de búsqueda inversa para el servidor Wazuh).
- Comprobamos que resolvemos los equipos tanto de forma directa como inversa. Por ejemplo, para el servidor wazuh-elk:

```
PS C:\Users\administrador.<NETBIOS>> nslookup wazuh-elk
Servidor:  hacosv01.<dominio>
Address:  10.x.x.x

Nombre:  wazuh-elk.<dominio>
Address:  10.x.x.40
```

```
PS C:\Users\administrador.<NETBIOS>> nslookup 10.x.x.40
Servidor:  hacosv01.<dominio>
Address:  10.x.x.x

Nombre:  wazuh-elk.<dominio>
Address:  10.x.x.40
```

Ya tendríamos las máquinas virtuales listas para comenzar la instalación del SIEM.

Host nuevo

Nombre (si se deja en blanco, se usa el nombre del dominio primario):
wazuh-server

Nombre de dominio completo (FQDN):
wazuh-server.

Dirección IP:
10.57

Crear registro del puntero (PTR) asociado

Permitir a cualquier usuario autenticado actualizar registros DNS con el mismo nombre de propietario

Agregar host Cancelar

Figura B.1: Agregar entrada en DNS servidor Wazuh

Host nuevo

Nombre (si se deja en blanco, se usa el nombre del dominio primario):
wazuh-elk

Nombre de dominio completo (FQDN):
wazuh-elk.

Dirección IP:
10.40

Crear registro del puntero (PTR) asociado

Permitir a cualquier usuario autenticado actualizar registros DNS con el mismo nombre de propietario

Agregar host Cancelar

Figura B.2: Agregar entrada en DNS servidor ELK

10.66.128.57	Puntero (PTR)	wazuh-server.	static
--------------	---------------	---------------	--------

Figura B.3: Registro de zona de búsqueda inversa Wazuh

B.2.4. Cambiar zona horaria

Por defecto la zona horaria es UTC (-2h) y lo suyo es que esté en la nuestra (CEST). Para cambiarla, hacemos:

```
jpcozar@wazuh-server:~$ sudo timedatectl set-timezone Europe/Madrid
```

```
jpcozar@wazuh-server:~$ timedatectl
```

```
Local time: vie 2020-04-03 18:44:08 CEST
Universal time: vie 2020-04-03 16:44:08 UTC
RTC time: vie 2020-04-03 18:44:07
```

```
Time zone: Europe/Madrid (CEST, +0200)
System clock synchronized: no
systemd-timesyncd.service active: yes
```

```
RTC in local TZ: no
```



INSTALACIÓN DE WAZUH SERVER

C.1. Instalación desde repositorios

En primer lugar, tenemos que añadir el repositorio de Wazuh

C.1.1. Añadir el repositorio de Wazuh

Existen una serie de requisitos que tenemos que tener instalados en el sistema antes de instalar Wazuh. Como estos comandos tienen que ser ejecutados como el usuario administrador (*root*), o nos cambiamos a dicho usuario o ejecutamos `sudo`¹ antes de cada uno. En nuestro caso, nos convertimos en administrador e instalamos los paquetes auxiliares:

```
jpcozar@wazuh-server-pruebas:~$ sudo su -  
root@wazuh-server-pruebas:~# apt-get update  
root@wazuh-server-pruebas:~# apt-get install curl apt-transport-https lsb-release gnupg2
```

Luego instalamos la llave GPG² para que nos permita instalar los paquetes. Para ello usaremos la herramienta *curl*. Esta herramienta accederá a Internet y necesitamos establecer el proxy para que pueda salir. Para establecerlo de forma permanente, creamos el fichero `~/.curlrc` con el siguiente contenido:

¹super user do

²Gnu Privacy Guard

```
proxy=10.x.x.x:3128
```

Una vez configurado el proxy, ya podemos ejecutar el comando:

```
root@wazuh-server-pruebas:~# curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | apt-key add
↵ -
OK
```

Añadimos el repositorio y actualizamos la información de los paquetes:

```
root@wazuh-server-pruebas:~# echo "deb https://packages.wazuh.com/3.x/apt/ stable main" | tee -a
↵ /etc/apt/sources.list.d/wazuh.list
root@wazuh-server-pruebas:~# apt-get update
```

C.1.2. Instalación del Manager de Wazuh

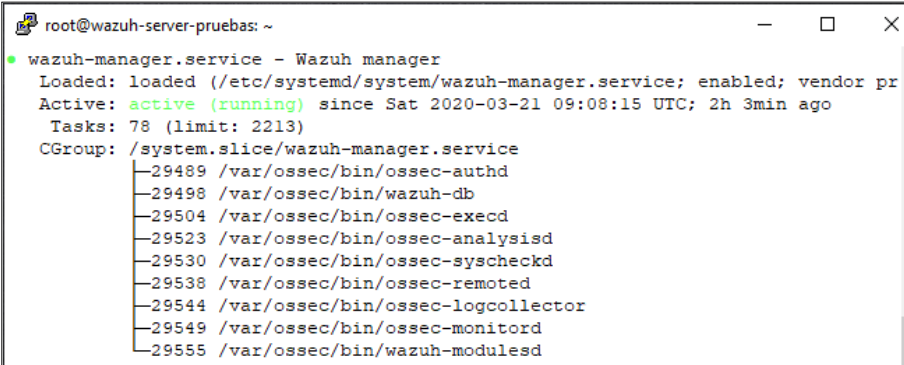
A continuación instalamos el Manager de Wazuh:

```
root@wazuh-server-pruebas:~# apt-get install wazuh-manager
```

Una vez instalado, comprobamos que se ha lanzado el servicio automáticamente:

```
root@wazuh-server-pruebas:~# systemctl status wazuh-manager
```

Y nos devolvería el resultado de la figura C.1



```
root@wazuh-server-pruebas: ~
● wazuh-manager.service - Wazuh manager
   Loaded: loaded (/etc/systemd/system/wazuh-manager.service; enabled; vendor pr
   Active: active (running) since Sat 2020-03-21 09:08:15 UTC; 2h 3min ago
     Tasks: 78 (limit: 2213)
   CGroup: /system.slice/wazuh-manager.service
           └─29489 /var/ossec/bin/ossec-authd
             └─29498 /var/ossec/bin/wazuh-db
               └─29504 /var/ossec/bin/ossec-execd
                 └─29523 /var/ossec/bin/ossec-analysisd
                   └─29530 /var/ossec/bin/ossec-syscheckd
                     └─29538 /var/ossec/bin/ossec-remoted
                       └─29544 /var/ossec/bin/ossec-logcollector
                         └─29549 /var/ossec/bin/ossec-monitor
                           └─29555 /var/ossec/bin/wazuh-modulesd
```

Figura C.1: systemctl status wazuh-manager

C.1.3. Instalación de la API de Wazuh

Para ejecutar la API de Wazuh necesitamos [NodeJS](#) en una versión igual o superior a la 4.6.1. En nuestro caso, al ser un Ubuntu recién instalado no está presente, así que lo instalamos, añadiendo en primer lugar el repositorio oficial de NodeJS:

```
root@wazuh-server-pruebas:~# curl -sL https://deb.nodesource.com/setup_10.x | bash -
root@wazuh-server-pruebas:~# apt-get install nodejs
```

Y por último, instalamos la API de Wazuh y comprobamos que se ha lanzado automáticamente (figura C.2):

```
root@wazuh-server-pruebas:~# apt-get install wazuh-api
root@wazuh-server-pruebas:~# systemctl status wazuh-api
```

```
root@wazuh-server-pruebas:~# systemctl status wazuh-api
● wazuh-api.service - Wazuh API daemon
   Loaded: loaded (/etc/systemd/system/wazuh-api.service; enabled; vendor preset
   Active: active (running) since Sat 2020-03-21 11:41:07 UTC; 48s ago
     Docs: https://documentation.wazuh.com/current/user-manual/api/index.html
   Main PID: 37863 (nodejs)
    Tasks: 11 (limit: 2213)
   CGroup: /system.slice/wazuh-api.service
           └─37863 /usr/bin/nodejs /var/ossec/api/app.js
```

Figura C.2: systemctl status wazuh-api

C.1.4. Desactivar actualizaciones automáticas de Wazuh

Es recomendable desactivar el repositorio de Wazuh o bloquear los paquetes (ponerlos en estado *hold*) para prevenir actualizaciones accidentales. En nuestro caso, optamos por bloquear los paquetes para poder hacer siempre una actualización manual usando el comando `apt-get install`.

```
root@wazuh-server-pruebas:~# echo "wazuh-manager hold" | dpkg --set-selections
root@wazuh-server-pruebas:~# echo "wazuh-api hold" | dpkg --set-selections
```

C.1.5. Instalar Filebeat

Filebeat es la herramienta en el servidor Wazuh que envía de forma segura las alertas y los eventos almacenados a Elasticsearch. Para instalarlo:

1. Añadimos el repositorio de Elastic, su llave GPG y actualizamos paquetes:

```
root@wazuh-server-pruebas:~# curl -s https://artifacts.elastic.co/GPG-KEY-elasticsearch |
↪ apt-key add -
OK
root@wazuh-server-pruebas:~# echo "deb https://artifacts.elastic.co/packages/7.x/apt
↪ stable main" | tee /etc/apt/sources.list.d/elastic-7.x.list
root@wazuh-server-pruebas:~# apt-get update
```

2. Instalamos Filebeat:

```
root@wazuh-server-pruebas:~# apt-get install filebeat=7.6.1
```

3. Descargamos el fichero de configuración de Filebeat del repositorio de Wazuh. Está preconfigurado para enviar alertas Wazuh a Elasticsearch:

```
root@wazuh-server-pruebas:~# curl -so /etc/filebeat/filebeat.yml https://\
↪ raw.githubusercontent.com/wazuh/wazuh/v3.11.4/extensions/filebeat/7.x/filebeat.yml
```

4. Descargamos la plantilla de alertas para Elasticsearch:

```
root@wazuh-server-pruebas:~# curl -so /etc/filebeat/wazuh-template.json
↪ https://raw.githubusercontent.com/wazuh/wazuh/v3.11.4/extensions/elasticsearch/7.x/\
↪ wazuh-template.json
```

5. Descargamos el módulo de Wazuh para Filebeat:

```
root@wazuh-server-pruebas:~# curl -s
↪ https://packages.wazuh.com/3.x/filebeat/wazuh-filebeat-0.1.tar.gz | sudo tar -xvz -C
↪ /usr/share/filebeat/module
```

6. Editamos el fichero `/etc/filebeat/filebeat.yml` y reemplazamos: `YOUR_ELASTIC_SERVER_IP` con la IP del servidor Elasticsearch.

```
output.elasticsearch.hosts: ['http://10.x.x.x:9200']
```

7. Activamos, arrancamos y comprobamos que está lanzado el servicio de Filebeat:

```
root@wazuh-server-pruebas:~# systemctl daemon-reload
root@wazuh-server-pruebas:~# systemctl enable filebeat.service
root@wazuh-server-pruebas:~# systemctl start filebeat.service
root@wazuh-server-pruebas:~# systemctl status filebeat
```

C.2. Actualización Wazuh Manager, Wazuh API y Filebeat

Como teníamos los paquetes retenidos, para actualizar hacemos:

```
jpcozar@wazuh-server:~$ sudo apt-get install filebeat wazuh-manager wazuh-api
```

Ya tendríamos la última versión de Wazuh Manager y Wazuh API funcionando, la 3.12 y la 7.6.2 de Filebeat.



INSTALACIÓN DEL SERVIDOR ELK STACK

D.1. Pre-requisitos ELK Stack

Como estamos en otra máquina virtual, tenemos que establecer el proxy corporativo (crear el fichero `~/.curlrc`, instalar los paquetes que son prerequisites, así como la clave GPG y añadir el repositorio de Elastic. Todo esto lo haremos con el usuario `root`.

```
jpcozar@wazuh-elk-pruebas:~$ sudo su -  
[sudo] password for jpcozar:  
root@wazuh-elk-pruebas:~# apt-get install curl apt-transport-https  
root@wazuh-elk-pruebas:~# curl -s https://artifacts.elastic.co/GPG-KEY-elasticsearch | apt-key  
↳ add -  
OK  
root@wazuh-elk-pruebas:~# echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" |  
↳ tee /etc/apt/sources.list.d/elastic-7.x.list  
root@wazuh-elk-pruebas:~# apt-get update
```

Ahora ya podemos instalar los dos componentes: Elasticsearch y Kibana.

D.2. Elasticsearch

1. Instalar el paquete de Elasticsearch

```
root@wazuh-elk-pruebas:~# apt-get install elasticsearch=7.6.1
```

2. Por defecto Elasticsearch solo escucha en localhost. Para que lo haga en la IP de nuestro servidor ELK, editamos el fichero `/etc/elasticsearch/elasticsearch.yml`, descomentamos y establecemos la línea `network.host` con la IP adecuada:

```
network.host: 10.x.x.x
```

3. Además tenemos que indicar los nodos del cluster (en este caso sólo habrá uno). Para ello, cambiamos o añadimos las líneas siguientes en el mismo fichero:

```
node.name: nodo1-pruebas
cluster.initial_master_nodes: ["nodo1-pruebas"]
```

4. Ya podemos activar y arrancar el servicio de Elasticsearch:

```
root@wazuh-elk-pruebas:~# systemctl daemon-reload
root@wazuh-elk-pruebas:~# systemctl enable elasticsearch.service
root@wazuh-elk-pruebas:~# systemctl start elasticsearch.service
```

5. Una vez que Elasticsearch está levantado, se recomienda cargar la plantilla de Filebeat donde esté instalado. En nuestro caso, está en el servidor de Wazuh, en la otra máquina virtual. Nos vamos a dicha máquina y ejecutamos:

```
root@wazuh-server-pruebas:~# filebeat setup --index-management -E
↪ setup.template.json.enabled=false
```

6. Ahora comprobamos, por ejemplo desde el servidor de Wazuh, que Elasticsearch está escuchando en el puerto 9200:

```
root@wazuh-server-pruebas:~# curl http://10.x.x.x:9200
```

7. Y nos devolvería el resultado de la figura D.1


```

root@wazuh-server-pruebas:~# curl http://10.█:9200
{
  "name" : "nodol-pruebas",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "31sQOeZdSVCTAKUbBmrZig",
  "version" : {
    "number" : "7.6.1",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "aa751e09be0a5072e8570670309b1f12348f023b",
    "build_date" : "2020-02-29T00:15:25.529771Z",
    "build_snapshot" : false,
    "lucene_version" : "8.4.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}

```

Figura D.1: Elasticsearch

D.3. Kibana

Kibana es el interfaz web que utilizaremos para visualizar los eventos y archivos almacenados en Elasticsearch.

1. Instalar el paquete de Kibana:

```
root@wazuh-elm-pruebas:~# apt-get install kibana=7.6.1
```

2. Instalamos el plugin de Wazuh para Kibana:

```

root@wazuh-elm-pruebas:~# cd /usr/share/kibana/
root@wazuh-elm-pruebas:~# sudo -u kibana https_proxy="http://10.x.x.x:3128"
→ bin/kibana-plugin install
→ https://packages.wazuh.com/wazuhapp/wazuhapp-3.11.4_7.6.1.zip

```

3. Kibana solo escuchará en el localhost por defecto. Para poder acceder a Kibana desde fuera, tenemos que modificar el fichero `/etc/kibana/kibana.yml` y descomentar y establecer la variable `server.host` y cambiar su valor:

```
server.host: "10.x.x.x"
```

4. Indicamos el host de Elasticsearch en el mismo fichero:

```
elasticsearch.hosts: ["http://10.x.x.x:9200"]
```

Además, al estar el servidor de Wazuh en otra máquina, tenemos que configurar el plugin de Wazuh para Kibana que acabamos de instalar. Para ello, editamos el fichero `/usr/share/kibana/plugins/wazuh/wazuh.yml` y cambiamos `localhost` por la IP del servidor de Wazuh:

```
hosts:
- default:
  url: http://10.x.x.x
  port: 55000
  user: foo
  password: bar
```

5. Activamos el servicio de Kibana pero no lo arrancaremos todavía:

```
root@wazuh-elk-pruebas:~# systemctl daemon-reload
root@wazuh-elk-pruebas:~# systemctl enable kibana.service
```

6. Por último, al igual que para el servidor de Wazuh, vamos a bloquear (hold) los paquetes de Elasticsearch y de Kibana para que no se actualicen automáticamente, pero que nos permita actualizarlo manualmente:

```
root@wazuh-elk-pruebas:~# echo "elasticsearch hold" | dpkg --set-selections
root@wazuh-elk-pruebas:~# echo "kibana hold" | dpkg --set-selections
```

D.4. Problemas con Kibana

El servicio de Kibana escucha en el puerto 5601 con lo que si en nuestro navegador nos vamos a <http://10.x.x.x:5601> deberíamos ver el interfaz web de Kibana con el plugin de Wazuh. Sin embargo, nos aparecía el mensaje **Kibana server is not ready yet**.

D.4.1. Kibana no arranca: Kibana server is not ready yet

Se nos ha presentado el problema, siguiendo las instrucciones de la documentación de Wazuh (Wazuh, 2020) de que Kibana no terminaba de estar listo y no mostraba la interfa web. La solución a este bug se encuentra en <https://github.com/wazuh/wazuh-kibana-app/issues/2145> donde nos indican que debemos editar el fichero `/etc/default/kibana` y añadir al final:

```
NODE_OPTIONS="--max-old-space-size=4096"
```

Si el arranque de Kibana se queda a medias, se nos puede presentar el siguiente error.

D.4.2. Another Kibana instance appears to be migrating the index

El error *Another Kibana instance appears to be migrating the index. Waiting for that migration to complete. If no other Kibana instance is attempting migrations, you can get past this message by deleting index .kibana_1 and restarting Kibana.* Este error se ha solucionado de la siguiente manera (hilo <https://github.com/elastic/kibana/issues/25464>):

```
jpcozar@wazuh-elk-pruebas:~$ sudo curl -XDELETE 'http://10.x.x.40:9200/.kibana_task_manager_1'
```

Una vez hecho esto, reiniciamos el servicio con `sudo systemctl restart kibana` y ya vemos el entorno de Kibana, en concreto el plugin de Wazuh, como observamos en la figura D.2.

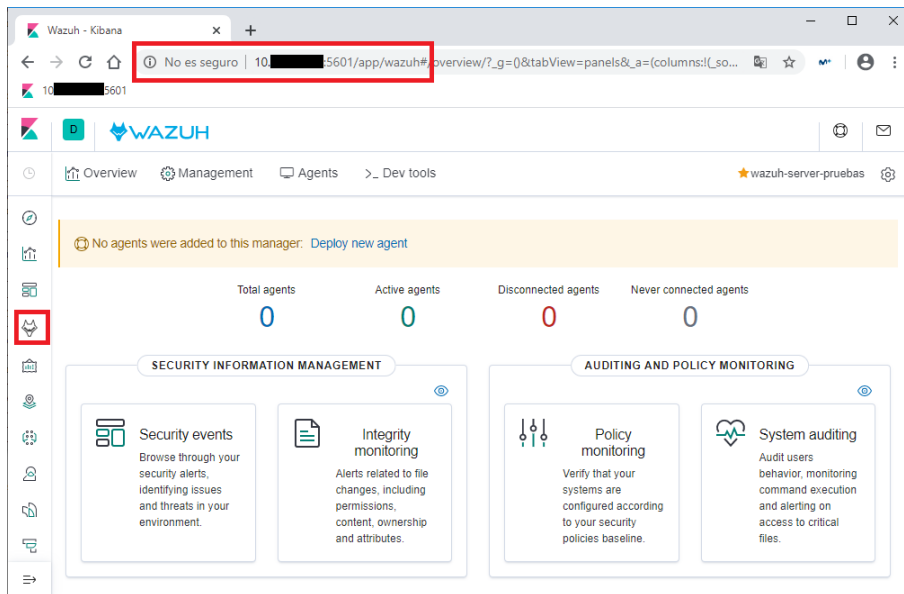


Figura D.2: Kibana desplegado en pruebas

D.5. Actualización Kibana y Elasticsearch

Para actualizar Kibana y Elasticsearch en el servidor Elk:

```
jpcozar@wazuh-elk:~$ sudo systemctl stop kibana
jpcozar@wazuh-elk:~$ sudo systemctl stop elasticsearch
jpcozar@wazuh-elk:~$ sudo apt-get install kibana=7.6.2
jpcozar@wazuh-elk:~$ sudo apt-get install elasticsearch=7.6.2
```

También habrá que actualizar el plugin de Kibana de Wazuh como se indica en la sección siguiente (D.6).

D.6. Actualización del plugin de Kibana

- Nos convertimos en administrador y paramos kibana:

```
jpcozar@wazuh-elk:~$ sudo su -
root@wazuh-elk:~# systemctl stop kibana
```

- Copiamos el fichero `wazuh.yml` a su nueva localización.

```
root@wazuh-elk:~# mkdir -p /usr/share/kibana/optimize/wazuh/config
root@wazuh-elk:~# cp /usr/share/kibana/plugins/wazuh/wazuh.yml
↪ /usr/share/kibana/optimize/wazuh/config/wazuh.yml
```

- Borramos el plugin de Wazuh para Kibana y los bundles generados

```
root@wazuh-elk:~# cd /usr/share/kibana
root@wazuh-elk:~# sudo -u kibana bin/kibana-plugin remove wazuh
root@wazuh-elk:~# rm -rf /usr/share/kibana/optimize/bundles
```

- Actualizamos los permisos de los ficheros.

```
root@wazuh-elk:~# chown -R kibana:kibana /usr/share/kibana/optimize
root@wazuh-elk:~# chown -R kibana:kibana /usr/share/kibana/plugins
```

- Instalamos el plugin de Wazuh 3.12 para la 7.6.2:

```
root@wazuh-elk:~# cd /usr/share/kibana
root@wazuh-elk:~# sudo -u kibana https_proxy="http://10.x.x.x:3128" bin/kibana-plugin
↔ install https://packages.wazuh.com/wazuhapp/wazuhapp-3.12.0_7.6.2.zip
```

- Actualizamos los permisos de los ficheros de configuración:

```
root@wazuh-elk:~# chown kibana:kibana /usr/share/kibana/optimize/wazuh/config/wazuh.yml
root@wazuh-elk:~# chmod 600 /usr/share/kibana/optimize/wazuh/config/wazuh.yml
```

- El último paso ya lo tenemos realizado debido al error que nos daba Kibana, que consiste en incrementar el tamaño de la pila de Kibana:

```
root@wazuh-elk:~#cat >> /etc/default/kibana << EOF
NODE_OPTIONS="--max_old_space_size=4096"
EOF
```

- Reiniciamos Kibana

```
root@wazuh-elk:~# systemctl daemon-reload
root@wazuh-elk:~# systemctl restart kibana
```

- Ya deberíamos tener la última versión estable de Elasticsearch y Kibana (y el plugin de Wazuh) funcionando, la 7.6.2.



CERTIFICADOS DIGITALES

E.1. Generación de certificados para wazuh-server

Preparamos un fichero `wazuh-server_csr.txt` con los datos que tendrá el certificado especificando como nombres alternativos tanto la IP del servidor como el **FQDN** (nombre DNS). Tendrá el siguiente contenido:

```
[req]
default_bits=2048
prompt=no
req_extensions=san
distinguished_name=dn
x509_extensions=san
extensions=san

[dn]
C=ES
ST=Cordoba
L=Cordoba
O=<O>
OU=<OU>
emailAddress=<email_address>
CN=wazuh-server.<dominio>

[san]
```

```
subjectAltName = @alt_names

[alt_names]
DNS.1=wazuh-server.<dominio>
IP.1=10.x.x.57
```

Ahora, generamos la clave privada y la petición de certificado de una vez con el fichero anterior:

```
root@wazuh-server:~# openssl req -new -sha256 -nodes -out wazuh-server.csr -newkey rsa:2048
↳ -keyout wazuh-server.key -config wazuh-server_csr.txt
```

Esto nos genera los ficheros `wazuh-server.key` y `wazuh-server.csr`. Verificamos que en la petición de certificado tenemos los datos correctos, incluyendo la IP y el nombre DNS:

```
root@wazuh-server:~# openssl req -text -noout -in wazuh-server.csr

...
Requested Extensions:
X509v3 Subject Alternative Name:
DNS:wazuh-server.<dominio>, IP Address:10.x.x.57
...
```

Ahora firmamos la petición de certificado con la CA nuestra, indicando tanto el fichero de salida (`wazuh-server.crt`) como las extensiones utilizadas:

```
root@wazuh-server:~# openssl x509 -req -in wazuh-server.csr -CA ./rootCA.pem -CAkey ./rootCA.key
↳ -out wazuh-server.crt -days 730 -sha256 --extensions san --extfile wazuh-server_csr.txt
↳ -CAcreateserial
```

Ya tenemos el certificado generado, `wazuh-server.crt`. Lo verificamos:

```
root@wazuh-server:~# openssl x509 -text -noout -in wazuh-server.crt
```

Ya tenemos el par de ficheros que necesitaremos para securizar el servidor de Wazuh: `wazuh-server.key` y `wazuh-server.crt`.

E.2. Generación de certificados para wazuh-elk

Preparamos un fichero `wazuh-elk_csr.txt` con los datos que tendrá el certificado especificando como nombres alternativos tanto la IP del servidor como el **FQDN** (entrada DNS). Tendrá el siguiente contenido:

```
[req]
default_bits=2048
prompt=no
req_extensions=san
```

```
distinguished_name=dn
x509_extensions=san
extensions=san

[dn]
C=ES
ST=Cordoba
L=Cordoba
O=<O>
OU=<OU>
emailAddress=<email_address>
CN=wazuh-elk.<dominio>

[san]
subjectAltName=@alt_names

[alt_names]
DNS.1=wazuh-elk.<dominio>
IP.1=10.x.x.40
```

Luego creamos la petición de certificado con esas características:

```
root@wazuh-server:~# openssl req -new -sha256 -nodes -out wazuh-elk.csr -newkey rsa:2048 -keyout
↳ wazuh-elk.key -config wazuh-elk_csr.txt
```

Verificamos que tenemos tanto el DNS como la IP:

```
root@wazuh-server:~# openssl req -text -noout -in wazuh-elk.csr
...
X509v3 Subject Alternative Name:
DNS:wazuh-elk.<dominio>, IP Address:10.x.x.40
...
```

Ahora creamos el certificado, firmándolo con el certificado raíz.

```
root@wazuh-server:~# openssl x509 -req -in wazuh-elk.csr -CA ./rootCA.pem -CAkey ./rootCA.key
↳ -out wazuh-elk.crt -days 730 -sha256 --extensions san --extfile wazuh-elk_csr.txt
↳ -CAcreateserial
```

Ya tenemos el par de claves para el servidor de la pila ELK, los ficheros `wazuh-elk.crt` y `wazuh-elk.key`.



MONITORIZACIÓN CON NAGIOS

Para poder monitorizar los servidores con Nagios, en primer lugar necesitamos tener instalados sus plugins.

F.1. Instalación plugins de Nagios

```
root@wazuh-server:~# apt-get install nagios-plugins
```

Una vez instalados, para poder chequearlos remotamente, necesitamos el plugin [NRPE](#) (Nagios Remote Plugin Executor). En el servidor de Nagios en producción ya está instalado, así que solo tenemos que instalar el cliente en cada uno de los servidores a monitorizar.

F.2. Instalación cliente NRPE v3.2.1

El cliente de NRPE v.3.2.1 (aunque se llame paradójicamente `nagios-nrpe-server`, es el que se instala en el equipo remoto, siendo `nagios-nrpe-plugin` el que se instala en el servidor de Nagios) ya se encuentra en los repositorios de Ubuntu:

```
root@wazuh-server:~# apt-get install nagios-nrpe-server
```


Editamos el fichero `/etc/nagios/nrpe.cfg` y autorizamos al servidor de Nagios, con IP `10.x.x.8`, a monitorizar los servidores Wazuh así como definimos los comandos a los que llamará el servidor Nagios (chequeo del número de usuarios, la carga del sistema, total de procesos, etc...):

```
...
allowed_hosts=127.0.0.1,10.x.x.8
...
dont_blame_nrpe=1
...
command[check_users]=/usr/lib/nagios/plugins/check_users -w 5 -c 10
command[check_load]=/usr/lib/nagios/plugins/check_load -r -w .15,.10,.05 -c .30,.25,.20
command[check_zombie_procs]=/usr/lib/nagios/plugins/check_procs -w 5 -c 10 -s Z
command[check_total_procs]=/usr/lib/nagios/plugins/check_procs -w 150 -c 200
command[check_local_disk]=/usr/lib/nagios/plugins/check_disk -w 20% -c 10% /
command[check_swap]=/usr/lib/nagios/plugins/check_swap -w 20% -c 10%
command[check_mem]=/usr/lib/nagios/plugins/check_mem -w 90 -c 95
```

Copiamos del servidor Nagios el script `check_mem` que no viene en la instalación por defecto de los plugins para poder chequear la memoria utilizada (un parámetro crítico en los servidores ELK Stack):

```
root@wazuh-server:~# scp jpcozar@hacolx05:/usr/lib/nagios/plugins/check_mem
↪ /usr/lib/nagios/plugins
```

Instalamos el servicio al inicio y comprobamos que está lanzado:

```
root@wazuh-server:~# systemctl enable nagios-nrpe-server
root@wazuh-server:~# systemctl restart nagios-nrpe-server
root@wazuh-server:~# systemctl status nagios-nrpe-server
```

Ahora desde el servidor Nagios `hacolx05` comprobamos que podemos chequear tanto el servidor Wazuh (`10.x.x.57`) como el Elk (`10.x.x.40`):

```
root@hacolx05:~# /usr/local/nagios/libexec/check_nrpe -H 10.x.x.57
NRPE v3.2.1
root@hacolx05:~# /usr/local/nagios/libexec/check_nrpe -H 10.x.x.40
NRPE v3.2.1
```

Una vez tenemos todo instalado, es el momento de añadir los servidores de Wazuh a Nagios. Para ello, editamos el fichero `/usr/local/nagios/etc/objects/linux.cfg` y añadimos las siguientes entradas para cada uno de los servidores:

```
define host{
use          linux-server
host_name    wazuh-elk
alias        wazuh-elk.<dominio>
address      10.x.x.40
}
```

```

define host{
use          linux-server
host_name    wazuh-server
alias        wazuh-server.<dominio>
address      10.x.x.57
}

```

Al agregarlo al grupo `linux-server` automáticamente realizará el chequeo por NRPE de varios parámetros: carga CPU, usuarios conectados, uso de memoria, espacio ocupado en la partición raíz, etc... Además de estos parámetros generales, también nos interesa monitorizar si están lanzados los servicios, tanto para el servidor Wazuh como para el servidor ELK. En concreto tendremos que monitorizar los puertos indicados en la tabla F.1.

Servicios Wazuh+ELK a monitorizar en Nagios

Servidor	Servicio	Puerto
wazuh-server	wazuh-manager	1515
wazuh-server	wazuh-api	5599
wazuh-elk	kibana	5601
wazuh-elk	elasticsearch REST	9200
wazuh-elk	elasticsearch nodes	9300

Tabla F.1: Servicios Wazuh+ELK a monitorizar en Nagios

Para ello, añadimos al fichero `/usr/local/nagios/etc/objects/linux.cfg` lo siguiente:

```

define service{
use          local-service          ; Name of service template to use
host_name    wazuh-server
service_description    wazuh-manager
check_command check_tcp!1515
}

define service{
use          local-service          ; Name of service template to use
host_name    wazuh-server
service_description    wazuh-api
check_command check_tcp!55999
}

define service{
use          local-service
host_name    wazuh-elk
service_description    kibana
check_command check_tcp!5601
}

define service{
use          local-service

```

APÉNDICE F. MONITORIZACIÓN CON NAGIOS

```

host_name          wazuh-elk
service_description elasticsearch-REST
check_command      check_tcp!9200
}

define service{
use                local-service
host_name          wazuh-elk
service_description elasticsearch-nodos
check_command      check_tcp!9300
}
    
```

Y a continuación, reiniciamos Nagios:

```
root@hacolx05:~# service nagios reload
```

Ya tendríamos monitorizados los servidores, como vemos en las figuras F.1 y F.2:

Service Status Details For Host 'wazuh-server'							
Host	Service	Status	Last Check	Duration	Attempt	Status Information	
wazuh-server	Current Load	OK	03-27-2020 18:44:46	0d 0h 27m 48s	1/4	OK - load average: 0.05, 0.02, 0.00	
	Current Users	OK	03-27-2020 18:46:18	0d 0h 27m 30s	1/4	USERS OK - 2 users currently logged in	
	Mem usage	OK	03-27-2020 18:45:04	0d 0h 18m 16s	1/4	[MEMORY] Total: 3933 MB - Used: 2415 MB - 61% [SWAP] Total: 3932 MB - Used: 0 MB - 0%	
	NRPE	OK	03-27-2020 18:47:39	0d 0h 29m 20s	1/4	TCP OK - 0,001 second response time on 10.66.128.57 port 5666	
	PING	OK	03-27-2020 18:47:39	0d 0h 27m 47s	1/4	PING OK - Packet loss = 0%, RTA = 0.50 ms	
	Root Partition	OK	03-27-2020 18:47:39	0d 0h 21m 38s	1/4	DISK OK - free space: / 31131 MB (81% inode=96%):	
	SSH	OK	03-27-2020 18:44:47	0d 0h 27m 30s	1/4	TCP OK - 0,001 second response time on 10.66.128.57 port 22	
	Swap	OK	03-27-2020 18:46:20	0d 0h 21m 38s	1/4	SWAP OK - 100% free (3932 MB out of 3932 MB)	
	Total Procs	OK	03-27-2020 18:47:39	0d 0h 29m 19s	1/4	PROCS OK: 117 processes	
	wazuh-api	OK	03-27-2020 18:47:39	0d 0h 0m 48s+	1/4	TCP OK - 0,005 second response time on 10.66.128.57 port 55999	
	wazuh-manager	OK	03-27-2020 18:47:39	0d 0h 0m 48s+	1/4	TCP OK - 0,037 second response time on 10.66.128.57 port 1515	

Figura F.1: Servidor Wazuh monitorizado en Nagios

Service Status Details For Host 'wazuh-elk'							
Host	Service	Status	Last Check	Duration	Attempt	Status Information	
wazuh-elk	Current Load	OK	03-27-2020 18:48:15	0d 0h 23m 17s	1/4	OK - load average: 0.01, 0.02, 0.00	
	Current Users	OK	03-27-2020 18:48:15	0d 0h 28m 45s	1/4	USERS OK - 1 users currently logged in	
	Mem usage	OK	03-27-2020 18:48:15	0d 0h 20m 58s	1/4	[MEMORY] Total: 3933 MB - Used: 3154 MB - 80% [SWAP] Total: 3932 MB - Used: 87 MB - 2%	
	NRPE	OK	03-27-2020 18:48:15	0d 0h 31m 49s	1/4	TCP OK - 0,001 second response time on 10.66.128.40 port 5666	
	PING	OK	03-27-2020 18:48:15	0d 0h 30m 16s	1/4	PING OK - Packet loss = 0%, RTA = 0.60 ms	
	Root Partition	OK	03-27-2020 18:48:15	0d 0h 23m 17s	1/4	DISK OK - free space: / 29860 MB (78% inode=92%):	
	SSH	OK	03-27-2020 18:48:15	0d 0h 28m 25s	1/4	TCP OK - 0,001 second response time on 10.66.128.40 port 22	
	Swap	OK	03-27-2020 18:48:15	0d 0h 23m 17s	1/4	SWAP OK - 98% free (3845 MB out of 3932 MB)	
	Total Procs	OK	03-27-2020 18:48:15	0d 0h 31m 48s	1/4	PROCS OK: 106 processes	
	elasticsearch-REST	OK	03-27-2020 18:48:15	0d 0h 3m 15s+	1/4	TCP OK - 0,004 second response time on 10.66.128.40 port 9200	
	elasticsearch-nodos	OK	03-27-2020 18:48:15	0d 0h 3m 15s+	1/4	TCP OK - 0,004 second response time on 10.66.128.40 port 9300	
	kibana	OK	03-27-2020 18:48:15	0d 0h 3m 15s+	1/4	TCP OK - 0,007 second response time on 10.66.128.40 port 5601	

Figura F.2: Servidor ELK monitorizado en Nagios



ANSIBLE

Para realizar este anexo, hemos utilizado la documentación online de Ansible ([Red Hat, 2020](#)).

G.1. Instalación Ansible en el servidor Wazuh

Instalaremos Ansible en el servidor de Wazuh, desde el que distribuiremos el agente Wazuh a los equipos Linux a monitorizar. Para ello hacemos:

```
root@wazuh-server:~# apt install ansible -y
root@wazuh-server:~# ansible --version
ansible 2.5.1
```

G.2. Configuración del servidor Ansible y sus clientes

G.2.1. Intercambio de claves entre servidores por ssh

Ya que toda la comunicación en Ansible se hace por ssh, necesitamos intercambiar dichas claves entre el servidor Ansible y sus clientes. Como estos son servidores Linux en los cuales está deshabilitado el ssh para el usuario root por motivos de seguridad, utilizaremos un usuario del grupo sudo, el usuario jpcozar.

Generamos las claves ssh para el servidor Wazuh con el usuario jpcozar:

```
jpcozar@wazuh-server:~$ ssh-keygen
```

Ya tenemos generado la clave privada en `/home/jpcozar/.ssh/id_rsa` y la clave pública que vamos a distribuir a los clientes en `/home/jpcozar/.ssh/id_rsa.pub`

A continuación, enviamos la clave pública al usuario `jpcozar` que existe en todos los servidores Linux en los que queremos instalar los agentes. Serán los indicados en la figura 3.3: `hacolx05`, `hacolx06`, `hacolx07`, `hacolx08`, `hacolx09` y `hacolx10`. Usaremos el comando `ssh-copy-id`:

```
jpcozar@wazuh-server:~$ ssh-copy-id jpcozar@hacolx05
jpcozar@wazuh-server:~$ ssh-copy-id jpcozar@hacolx06
```

Por ejemplo, para el servidor `hacolx09` tendríamos lo mostrado en la figura G.1.

```
jpcozar@wazuh-server:~$ ssh-copy-id jpcozar@hacolx09
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/jpcozar/.ssh/id_rsa.pub"
The authenticity of host 'hacolx09 (10.███)' can't be established.
ECDSA key fingerprint is SHA256:Dioau20UhgDPcMf4GnyG8dl+efjFYVBRYlkf18+BsdU.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
jpcozar@hacolx09's password:
Number of key(s) added: 1
Now try logging into the machine, with: "ssh 'jpcozar@hacolx09'"
and check to make sure that only the key(s) you wanted were added.
```

Figura G.1: `ssh-copy-id` a servidor `hacolx09`

Una vez hecho esto, ya podríamos logarnos en dichos servidores con este usuario sin que nos pidiera contraseña desde el servidor de Wazuh.

```
jpcozar@wazuh-server:~$ ssh-copy-id jpcozar@hacolx09
jpcozar@hacolx09:~$
```

G.2.1.1. Creación del fichero de inventario de hosts Ansible

A continuación en el servidor Wazuh, creamos el fichero `/etc/ansible/hosts` con los servidores Linux a los que instalaremos el agente con el siguiente contenido:

```
[wazuh-agents]
hacolx05 ansible_ssh_user=jpcozar
hacolx06 ansible_ssh_user=jpcozar
hacolx07 ansible_ssh_user=jpcozar
hacolx08 ansible_ssh_user=jpcozar
hacolx09 ansible_ssh_user=jpcozar
hacolx10 ansible_ssh_user=jpcozar
```

Comprobamos que todos los servidores están accesibles por Ansible con el comando:

```

jpcozar@wazuh-server:~$ ansible all -m ping
hacolx08 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
hacolx07 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
hacolx09 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
hacolx06 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
hacolx05 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
hacolx10 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}

```

G.3. Instalación de Audit

Creamos un directorio para playbooks de Audit:

```

root@wazuh-server:# mkdir -p /etc/ansible/roles/audit-ansible/playbooks

```

Y creamos el playbook con un fichero llamado `install-auditd.yml` con el siguiente contenido:

```

- hosts: hacolx05, hacolx06, hacolx07, hacolx08, hacolx09, hacolx10

tasks:

- name: install auditd
  package:
    name: "{{item}}"
    state: latest
  with_items:
- auditd

environment:
  http_proxy: http://10.x.x.x:3128
  https_proxy: http://10.x.x.x:3128

```

Ya podríamos ejecutar el playbook:

```
root@wazuh-server:~# ansible-playbook install-auditd.yml -b -K
```

Y tendríamos el servicio de auditoría instalado en todos los servidores Linux, como vemos en la figura G.2.

```
TASK [install auditd] *****
changed: [hacolx09] => (item=auditd)
changed: [hacolx08] => (item=auditd)
changed: [hacolx07] => (item=auditd)
changed: [hacolx06] => (item=auditd)
changed: [hacolx10] => (item=auditd)
changed: [hacolx05] => (item=auditd)

PLAY RECAP *****
hacolx05      : ok=2    changed=1    unreachable=0    failed=0
hacolx06      : ok=2    changed=1    unreachable=0    failed=0
hacolx07      : ok=2    changed=1    unreachable=0    failed=0
hacolx08      : ok=2    changed=1    unreachable=0    failed=0
hacolx09      : ok=2    changed=1    unreachable=0    failed=0
hacolx10      : ok=2    changed=1    unreachable=0    failed=0
```

Figura G.2: Despliegue de auditd en servidores Ubuntu mediante Ansible



DESPLIEGUE DEL AGENTE WAZUH POR GPO

H.1. Problema implementación de software GPO

El problema que tiene la Implementación de software por GPO es que, por defecto, no permite pasar parámetros al ejecutable `.msi`. En nuestro caso necesitamos indicarlo al menos el servidor manager de Wazuh (variable `WAZUH_MANAGER`) y el servidor de registro de Wazuh (variable `WAZUH_REGISTRATION_SERVER`), para poder instalar y registrar el agente de una sola vez de forma silenciosa y automática. Por tanto, tenemos que recurrir a herramientas externas como [SuperOrca](#) o [InstEd](#), que permiten crear ficheros de transformaciones que pasarán dichos parámetros al ejecutable durante la instalación.

H.2. Fichero de transformación `.mst` con InstEd

En nuestro caso hemos usado InstEd y el proceso es el siguiente:

1. Abrimos el fichero `wazuh-agent-3.12.0-1.msi` con InstEd

2. Seleccionamos en el menú Transform->New Transform y guardamos un fichero de transformación llamado wazuh-agent-3.12.0-1.mst
3. A la izquierda seleccionamos la tabla Property
4. A la derecha creamos una nueva fila con la Propiedad WAZUH_MANAGER y el valor 10.x.x.57
5. A la derecha creamos una nueva fila con la Propiedad WAZUH_REGISTRATION_SERVER y el valor 10.x.x.57.
6. Una vez lo tenemos todo como en la figura H.1 ya podemos guardar el fichero .mst

H.3. Creación de la directiva EQU_InstalarAgenteWazuh

Una vez tenemos el fichero y su transformada, los colocamos en un sitio accesible para los equipos a los que se le va aplicar y creamos la directiva EQU_InstalarAgenteWazuh.

En la directiva establecemos un Filtro de seguridad, indicando que únicamente se va a aplicar a los Controladores del dominio y a los Equipos del dominio.

Para crearla, nos vamos a:

- Configuración del equipo → Directivas → Configuración de software → Instalación de software
- Nuevo paquete y seleccionamos el fichero original wazuh-agent-3.12.0-1.msi en el sitio accesible.
- En la pestaña Modificaciones, seleccionamos el fichero de transformación creado con InstEd, llamado wazuh-agent-3.12.0-1.mst.
- Abrimos el Administrador de directivas de grupo y vinculamos la directiva de grupo a las unidades organizativas donde tenemos los servidores Windows 2012 R2 que queremos monitorizar con Wazuh.
- Por último, realizamos una actualización de directivas con:

```
PS D:\ gpupdate /force
```

que nos pedirá un reinicio y ya tendríamos los agentes de Wazuh de los servidores Windows instalados y registrados en el servidor Wazuh y visibles en Kibana.

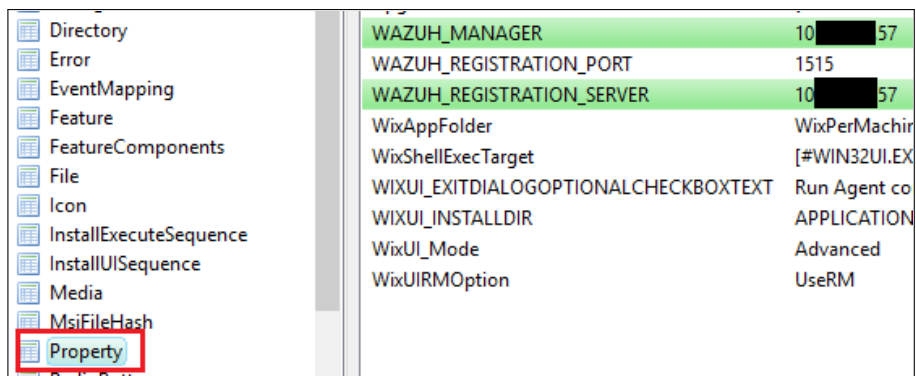


Figura H.1: Creación Transformación en InstEd