TRABAJO FIN DE GRADO

INGENIERÍA INFORMÁTICA

Integración de los procesos de Hacking ético en la plataforma de ciberseguridad de Wazuh

Análisis de las distintas herramientas y procedimientos de offensive security y su integración con el software de Wazuh, así como el diseño y creación de un laboratorio de pruebas para test de penetración

Autor
Francisco Navarro Morales

Director
Alberto Guillén Perales

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Septiembre de 2020

Integración de los procesos de Hacking ético en la plataforma de ciberseguridad de Wazuh

Análisis de las distintas herramientas y procedimientos de offensive security y su integración con el software de Wazuh, así como el diseño y creación de un laboratorio de pruebas para test de penetración.

Autor
Francisco Navarro Morales

Director
Alberto Guillén Perales

Integración de los procesos de Hacking ético en la plataforma de ciberseguridad de Wazuh: Análisis de las distintas herramientas y procedimientos de offensive security y su integración con el software de Wazuh, así como el diseño y creación de un laboratorio de pruebas para test de penetración.

Francisco Navarro Morales

Palabras clave: palabra_clave1, palabra_clave2, palabra_clave3, ...

Resumen

Poner aquí el resumen.

Etic Hacking procedures integration with Wazuh OpenSoruce Security platform: Offesntive security's tools & procedures analysis as well as their integration with Wazuh software. Design and development of a lab for penetration test.

Francisco Navarro Morales

Keywords: Keyword1, Keyword2, Keyword3, ...

Abstract

Write here the abstract in English.

Yo, **Francisco Navarro Morales**, alumno de la titulación en ingeniería informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 54202078W, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Francisco Navarro Morales

Granada a 20 de Septiembre de 2020 .

D. **Alberto Guillén Perales**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Integración de los procesos de Hacking ético en la plataforma de ciberseguridad de Wazuh, Análisis de las distintas herramientas y procedimientos de offensive security y su integración con el software de Wazuh, así como el diseño y creación de un laboratorio de pruebas para test de penetración.***, ha sido realizado bajo su supervisión por **Francisco Navarro Morales**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de mes de 201 .

Los directores:

Alberto Guillén Perales

Agradecimientos

Poner aquí agradecimientos...

Glosario

Cloud La computación en la nube o *cloud* (del inglés cloud computing), conocida también como servicios en la nube, informática en la nube, nube de cómputo o simplemente «la nube», es un paradigma que permite ofrecer servicios de computación a través de una red, que usualmente es internet.. 15, 20

compilance 'el cumplimiento' de las normativas o leyes referentes a la seguridad de los datos que una empresa pueda almacenar o gestionar. 21, 24

CPA Certified Public Accountant. 24

ingeniería inversa Es una técnica que consiste en intentar obtener información sobre cómo está hecho o cómo funciona un producto a partir del propio producto en si. Intentando adivinar cómo funciona por medio de su uso y de pruebas que corroboren las suposiciones que se hagan.. 16

IOT The Internet of Things. 15

Malware O software "malicioso" es todo aquél programa o código que pretende (de forma intencionada) causar daños y/o sacar beneficios de un sistema. 15

network sniffing Es un tipo de distribución de Software en el que las actualizaciones son continuas en lugar de depender de un versionado discreto. Los cambios se van añadiendo de forma incremental conforme van siendo disponibles en lugar de ir emitiendo nuevas versiones con todos los cambios desde la anterior.. 28

network sniffing Es la acción de "atender" al tráfico indiscriminado que circula por una red como si se fueran todos los posibles destinatarios con el objetivo de obtener una información que no fuera destinada a nosotros. 16

obfuscation Ofuscación, ocultación, anonimato. Es el acto de evitar ser descubierto mientras realizas un ataque o una auditoría. Bien eliminando los rastros que puedas dejar u ocultándolos por medio de falsos rastros que escondan los tuyos.. 34

OpenSource OpenSource o código abierto es un tipo de software liberado con una licencia que asegura el derecho de los usuarios a usar, estudiar, cambiar y distribuir el mismo con cualquier propósito.. 15

OpenSource Una Pull Request es la acción de validar un código que se va a mergear de una rama a otra. Por ejemplo, de una rama de desarrollo en un Fork de un proyecto a una rama oficial.. 22

Ransomware Es un tipo de malware que pretende hacer públicos o inaccesibles (por medio de encriptación, por ejemplo) los datos de la víctima hasta que esta pague un "rescate".. 15

Vagrant 'una herramienta diseñada para el despliegue y configuración de entornos de máquinas virtuales (utilizando diversos proveedores como virtualbox, qemu, aws, etc.... 32

Capítulo 1

Introducción

1.1. *Hacking* ético y seguridad ofensiva

En un contexto marcado por el explosivo crecimiento de las tecnologías de la información, nuestras vidas se encuentran cada vez más ligadas a los dispositivos electrónicos y a sus posibles vulnerabilidades: la aparición del comercio electrónico y la banca *online*, el uso del correo electrónico como alternativa al tradicional o la creciente expansión del internet de las cosas (IOT), Cloud o el creciente uso de los dispositivos móviles, entre otros, crean un entorno en el que aquellos dispuestos a buscar y explotar las vulnerabilidades de nuestros sistemas pueden lucrarse y perjudicar gravemente si no se toman medidas para evitarlo.

En este marco surgen constantemente herramientas para asegurar la protección de nuestros dispositivos, tanto de los datos sensibles como de la disponibilidad de los mismos (ya sea porque un ataque inhabilite un servicio que ofrecemos o cierto tipo de Malware nos impida acceder a nuestro dispositivo de forma regular, véase Ransomware). Entre todas estas herramientas surge ***Wazuh***, una plataforma OpenSource de ciberseguridad que pretende convertirse en un estándar y una referencia a la hora de proteger nuestros sistemas y que engloba distintos tipos de módulos o herramientas que nos ofrecen, entre otras cosas, recolección y análisis de los registros (*logs*) de nuestros *endpoints*, detección de vulnerabilidades en el software instalado, control de la integridad de archivos sensibles o detección de intrusiones.

Wazuh se utiliza para asegurar la seguridad de nuestros entornos analizando los eventos de interés que ocurren en el sistema y generando alertas cuando sea necesario. Puesto que se trata de un software que pretende alertar de posibles ataques e intrusiones, sería muy interesante buscar un enfoque que se adecue a las necesidades de aquellos que se dedican a poner a prueba los sistemas. Wazuh podría ser una herramienta esencial para llevar a cabo análisis de penetración de sistemas desde una perspectiva externa, desde la llamada "seguridad ofensiva".

El objetivo principal de este proyecto es investigar y analizar las principales herramientas disponibles para aquellos que trabajan en auditorías de ciberseguridad y determinar cómo puede encajar Wazuh dentro del marco de las mismas con la idea de **crear un entorno de desarrollo para test de penetración de sistemas** capaz de extraer, analizar e indexar la información proporcionada por las herramientas disponibles en el mismo, de forma que se facilite el almacenamiento e intercambio de la información relevante de **todo aquello que se ha probado durante la auditoría**, bien para intercambio de información entre los miembros del equipo que esté llevando a cabo la misma o bien como una fuente importante de información que entregar a un cliente cuando se realiza una auditoría, así como una base para posibles informes y reportes del mismo.

1.1.1. Seguridad ofensiva

A veces la mejor defensa puede ser un buen ataque. Pese a que las medidas que un administrador pueda tomar 'desde dentro del sistema' pueden ser más que suficientes para proteger un dispositivo, cada vez están más presentes en el ámbito de la ciberseguridad aquellos que se ponen en la piel de un atacante e intentan encontrar puntos flacos en las defensas de un sistema '**desde fuera**'. Son los llamados ***hacker éticos***. Actuando dentro del marco legal y con permiso explícito de los responsables de un sistema, tratan de atacarlo como si quisieran sacar algún provecho de estos o causar daños, utilizando, en muchas ocasiones, técnicas de ingeniería social para encontrar puntos de acceso al sistema a partir de malas prácticas de usuarios con acceso a los mismos.

El mensaje ha calado con fuerza en la comunidad y diversos grupos de *hackers* han desarrollado herramientas para lo que llaman "*tests* de penetración de sistemas", en los que buscan extraer toda la información posible de un sistema por medio de técnicas como el escaneo de puertos, descifrado de contraseñas, análisis de redes (network sniffing) o ingeniería inversa.

Muchas de estas técnicas se pueden llevar a cabo por medio de herramientas OpenSource que suelen estar disponibles de un modo u otro en las distintas distribuciones de Linux. Es por ello que surgen diversas distribuciones especializadas en test de penetración como **Kali Linux** (basada en Debian) o *Black Arch Linux*, un derivado de *Arch Linux* con repositorios y herramientas pensadas para este tipo de tests.

1.2. Estado del arte

1.3. Motivación

Kali Linux (así como otras distribuciones Linux orientadas a seguridad ofensiva) y la mayoría de sus herramientas son libres y gratuitas, al igual

que la plataforma Wazuh. Dado que esta pretende ofrecer a sus usuarios una **única plataforma de ciberseguridad** que reúna todo aquello que puedan necesitar para la seguridad de sus sistemas en único software (evitando así tener multitud de herramientas para cada necesidad distinta), sería interesante para ambas comunidades, la de Wazuh y la de hackers éticos que hacen uso de estas distribuciones, que se acortaran las distancias entre estas dos herramientas y se facilitara el uso conjunto de ambas. De esta forma, las dos herramientas podrían beneficiarse una de la otra y crear una comunidad conjunta sólida y robusta y el proyecto Wazuh (con sede en Granada) podría introducirse en la la escena de la ciberseguridad ofensiva basada en test de penetración.

1.3.1. Registro de nuestros movimientos

Una de las principales características de Wazuh es que está diseñado para, fácilmente, indexar en un motor de búsqueda como Elasticsearch eventos de seguridad de interés. Así pues, cabe destacar que un aspecto fundamental de las auditorías de seguridad (penetration testing) es el registro (o logging) de nuestros pasos durante el test por varias razones entre las cuales quiero destacar:

nota, source: https://www.contextis.com/us/blog/logging-like-a-lumberjack

Reporting Unos buenos logs nos pueden servir como referencia a la hora de escribir o mejorar un reporte.

Responsabilidad Si ocurre algún problema durante el test de penetración. Tener un buen registro de todo lo que se ha hecho durante la auditoría puede servir para probar que no hemos sido nosotros los causantes de algún daño que se haya producido durante la auditoría.

Contrato Es posible que los clientes especifiquen en el propio contrato del trabajo que es necesario registrar todos los pasos tomados.

Replicidad Llevando un registro de nuestros tests podremos replicarlos o imitarlos en un futuro si fuera necesario.

1.3.2. Faraday: un entorno colaborativo de penetración de sistemas y administración de vulnerabilidades

Existe un proyecto opensource llamado **Faraday**[1] que pretende ofrecer una experiencia multiusuario de entorno de desarrollo para test de penetración de sistemas. Esta diseñado para analizar datos extraídos de distintas herramientas de ciberseguridad e indexar los resultados del análisis de forma que tengan un formato analizable y fácil de compartir.

[1] Ver `https://github.com/infobyte/faraday`

Este proyecto es muy similar a la idea en la que se basa el proyecto, con la salvedad de que está diseñado "from scratch.[en] lugar de partir de herramientas más potente como Wazuh y Elasticsearch (con su interfaz web Kibana).

Usando Wazuh junto con Elasticsearch, Kibana y la Wazuh-app, podemos aspirar a ofrecer un servicio similar al de Faraday con muchas ventajas.

Análisis de Faraday

1.4. Justificación

Wazuh consta de dos partes fundamentales: un *agente*, que se instala en el *endpoint* a monitorizar y recolecta información sensible para la seguridad del mismo, y un *manager*, que recibe la información de distintos agentes y la procesa como eventos de interés que pueden (o no) generar alertas al usuario en función de la gravedad del evento.

Es decir, la principal utilidad de Wazuh no es proteger activamente tu sistema (como haría un antivirus) sino informar y dejar constancia de eventos críticos para la seguridad del mismo que hayan tenido lugar, para que podamos actuar en consecuencia.

Wazuh está diseñado para funcionar junto con *Elasticsearch*, un motor de indexación que permite almacenar e indexar las alertas generadas por el manager para que sean fácilmente accesibles (por medio de consultas o a través de su interfaz gráfica, *Kibana*, accesible a través de un navegador

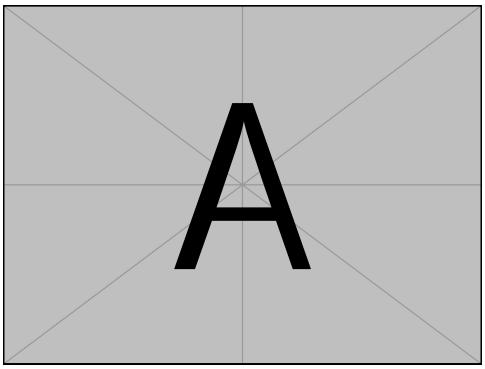

Figura 1.1: Ejemplo de visualización de alertas de seguridad indexadas en Elasticsearch por un manager Wazuh en el navegador usando Kibana

Por un lado, sería interesante poder realizar test de penetración sobre sistemas monitorizados por un agente *Wazuh* y comprobar hasta qué punto este es capaz de detectar las intrusiones que se realicen desde el exterior y notificar o responder a eventos sensitivos.

Por otro lado, Wazuh podría servir a alguien sin acceso a un sistema a realizar un test de penetración sobre este analizando los logs de las distintas

herramientas utilizadas durante el test y generando alertas según la relevancia de los eventos que se detectaran. Esto serviría para tener un registro de todo lo que se ha probado y los resultados obtenidos que podría servir como base para desarrollar un informe para una compañía para la que estuviéramos trabajando o simplemente para dejar constancia de los resultados obtenidos.

Además, uno de los puntos más sensibles de un test de penetración es la responsabilidad del auditor para con la empresa que le contrata. Si sucede algo o se produce algún daño durante la auditoría, el hacker ético puede verse en problemas. Utilizando Wazuh como un sistema de control de las acciones llevadas a cabo durante la auditoría serviría tanto para compartir los resultados obtenidos como para dejar constancia en tiempo real de las pruebas que se hacen en el sistema, como una forma de demostrar qué se ha hecho y qué no se ha hecho durante la auditoría.

Así mismo, se podría aprovechar el potencial que ofrece el formato definido de alertas de Wazuh con Elasticsearch y Kibana para generar gráficas y *dashboards* relacionados con los resultados del análisis. La propia base de datos de Elasticsearch, junto con su interfaz Kibana, podrían servir a modo de informe sobre un test realizado sobre varios sistemas.

Así pues, Wazuh es una herramienta potencialmente valiosa para aquellas distribuciones de Linux orientadas al test de penetración. Potencial porque Wazuh utiliza un *ruleset* con decodificadores de logs y reglas para generar alertas que, actualmente, no da soporte para la mayoría de herramientas de que disponen estos sistemas. Como hemos señalado, Wazuh está más orientado a buscar eventos sensibles en los logs de los servicios que correrían en un endpoint o en *firewalls*, antivirus, etc... y no en la información que podría dar una herramienta de test de penetración.

Si realizamos una investigación de aquello que le falta a Wazuh para ser útil en este tipo de análisis, un posible desarrollo de decodificadores y reglas para esas herramientas podría suponer un paso enorme en el desarrollo del proyecto. Llevándolo a los administradores del mismo, es posible que estén dispuestos a integrarlo en su software e incluso, eventualmente, dar soporte oficial para aquellas funcionalidades que se desarrollaran a lo largo de este trabajo de fin de grado.

1.4.1. Hacking ético y el modelo de software libre empresarial

Wazuh es una compañía cuyo producto es completamente libre y gratuito. Eso significa que debe encontrar otros medios para sustentarse diferentes al venta de licencias.

El modelo actual de Wazuh, y que la compañía defiende mantener, consiste en tener una comunidad Open Source de calidad y muy activa, ofreciendo soporte técnico especializado y gratuito a todos aquellos usuarios que

lo soliciten y ofreciendo siempre las últimas actualizaciones a sus usuarios de forma gratuita. Para obtener beneficios, aprovechan la fama que proporciona dicha comunidad para vender 'paquetes' de soporte técnico *premium*, así como cursos de formación.

Además de esto, el proyecto ha desarrollado una plataforma Cloud que permite al usuario que la contrata abstraerse de todos los problemas derivados de la instalación y mantenimiento de los servidores para utilizar Wazuh y ofrece a la compañía otra forma más de ingresos sin renunciar a un modelo en que su software es completamente libre y gratuito.

Sin embargo, estas opciones no siempre son suficientes y conviene buscar otras alternativas. Si este proyecto tiene el potencial que se espera podría presentarse a los directivos de la empresa como una iniciativa para implantar en Wazuh un equipo de *pen-testing* especializado cuyos servicios puedan ser contratados por empresas que conozcan el proyecto para buscarles vulnerabilidades.

Podría ser una oportunidad de negocio de gran interés para la empresa y una forma de aportar valor a la misma por mi parte y de hacer que este trabajo no quede solo en un ámbito académico sino que escale al ámbito profesional.

1.5. Objetivos

Así pues, este proyecto sería en parte un proyecto que conste de los siguientes pasos:

- Investigación sobre entornos de pruebas para pentest. Estado del arte, ejemplos de entornos viables para experimentar y análisis de ventajas e inconvenientes de los mismos. Instalación de Wazuh sobre estos entornos de pruebas para un análisis de los resultados obtenidos (desde la perspectiva del administrador de sistemas) en cuanto a alertas se refiere durante una auditoría o test de penetración.

- Investigación sobre las distintas distribuciones orientadas a test de penetración y sus herramientas. Clasificación de las herramientas según su utilidad y selección de aquellas que resulten de mayor interés. Realización de pruebas de las mismas sobre los distintos entornos de pruebas analizados en el primer punto.

- Desarrollo de reglas y decoders para integrar las herramientas de test de penetración en Wazuh. Así mismo, aprovecharemos los experimentos para encontrar casos que no estén actualmente contemplados por Wazuh y tratar de cubrirlos con las reglas que sea necesario. Todo el trabajo desarrollado se presentará a la compañía con el objetivo de integrarlo en el código oficial.

- Creación de una o varias imágenes de las distribuciones de Linux seleccionadas que integre Wazuh y aquellas herramientas de interés que no vengan por defecto y tratar de hacerlas llegar a la comunidad. La generación de dichas imágenes deberá automatizarse de forma que se puedan generar nuevas versiones de la misma conforme el software de Wazuh evolucione.

- Realización de un estudio de test de penetración sobre los entornos de pruebas definidos y creación de un protocolo de actuación para este tipo de estudios. Así mismo, se generarán informes con los resultados del test utilizando todas las herramientas creadas hasta el momento.

- Análisis de los aspectos legales ligados al desarrollo del trabajo y de toda aquella práctica posiblemente derivada del mismo. Entender qué peligros hay en llevar a cabo test de penetración de sistemas sin el consentimiento previo de la compañía o individuo cuyo sistema se va a poner a prueba y ofrecer formas seguras de formarse en esta temática sin riesgo de cometer un delito.

1.6. Planificación

El desarrollo del trabajo constará de las siguientes fases.

1.6.1. Definición y creación del entorno de pruebas: 50 horas

Queremos hacer un entorno de pruebas realista. Investigaremos sobre aquellos tipos de compilance más importantes (relacionados con finanzas o con datos médicos de los pacientes de un hospital, por ejemplo) y definiremos uno o varios escenarios que constarán de varias máquinas con distintos sistemas operativos (Linux, Unix, Windows, macOS, Solaris...) y distinto software según el escenario.

Utilizaremos un enfoque de infraestructura as a code, de forma que el entorno sea fácilmente desplegable y configurable tanto para el proyecto como para posibles interesados en hacer uso de él.

Las máquinas desplegadas tendrán todas instalados agentes de Wazuh, de forma que podamos comprobar en todo momento cómo responde el software a un ataque externo y si es capaz de generar todas las alertas que cabría esperar e incluso responder por medio de su sistema de 'respuesta activa'.

Estimo que el tiempo requerido será de unas cincuenta horas, teniendo en cuenta una parte de labor de investigacion y definición de los entornos y otra parte programando el despliegue y aprovisionamiento de las máquinas. Crearé un repositorio en GitHub con el código que utilice y un servidor Jenkins para automatizar el despligue de las instancias que hagan falta, así como su aprovisionamiento y destrucción.

1.6.2. Investigación sobre herramientas y distribuciones de pentesting: 30 horas

Una vez creado el entorno, descargaré algunas de las distribuciones más famosas en cuanto a test de penetración se refiere y probaré sus herramientas sobre el entorno de pruebas. La idea aquí sería realizar una comparativa de las distintas distribuciones y sus herramientas y hacer una clasificación y selección de aquellos programas que nos puedan servir como base para el proyecto.

Estimo que las pruebas y redacción de las comparativas y conclusiones puede requerir unas 30 horas.

1.6.3. Desarrollo de reglas y decoders: 30 horas

Una vez seleccionadas las herramientas y distribuciones a utilizar, instalaremos Wazuh en ellas y empezaremos a hacer pruebas sobre el laboratorio de test. La idea es ir desarrollando decoders y reglas para que Wazuh pueda analizar el output de los distintos comandos y generar alertas cuando detecte que alguna información relevante ha sido descubierta durante el test, dejando un registro claro de ello en Elasticsearch y Kibana en forma de alerta.

Así mismo, comprobaremos desde una perspectiva 'interna' si los agentes de Wazuh instalados en las máquinas del entorno de pruebas son capaces de reportar todos los ataques que realicemos y en caso de detectar cualquier tipo de fisura en el software la reportaremos y trataremos de añadir las reglas o decoders que hagan falta, abriendo si es necesario un OpenSource en el repositorio oficial del *ruleset* de Wazuh.

Esta parte no tendría un tiempo fijo definido, cuanto más tiempo se le dedique, más reglas y decoders podríamos añadir. En principio voy a asignarle 30 horas y, si sobrara tiempo, podría seguir trabajando en esta parte para dar soporte a más herramientas y casos de uso.

1.6.4. Dashboards e informes en Kibana: 50 horas

Wazuh dispone de un plugin para Kibana que podríamos utilizar como base para añadir una nueva pestaña dedicada al test de penetración, con un Dashboard que contenga algunas tablas y gráficos de interés, extraídos de las alertas generadas durante el test de penetración.

No tengo muchos conocimientos sobre Kibana y cómo se programan estos elementos, por lo que pienso que esto me puede acabar llevando más tiempo de lo que debería. Así pues, considero que 50 horas puede ser una buena estimación.

1.6.5. Creación de una o varias imágenes de Linux con Wazuh preparado para test de penetración: 20 horas

Esta parte sería automatizar la generación y exportación de dichas imágenes (de forma que se puedan generar nuevas imágenes cuando se libere una nueva versión de Wazuh o se actualice la imagen base de la distribución).

En principio se seleccionaría una de las distribuciones y si fuera bien de tiempo quizá se podría hacer lo mismo con una o dos más, dependiendo del tiempo que se tarde en hacer una de ellas.

También sería interesante en este punto compartir con la comunidad dichas imágenes, publicar algún artículo en Medium, Dev, o alguna plataforma de blog técnicos que puedan llegar a aquellos a los que pueda resultarle interesante.

Estimo que esta parte puede llevarse unas 20 horas

1.6.6. Fase final: estudio del test de penetración, 60 Horas

En esta fase y, basándonos en las conclusiones extraídas hasta el momento, definiríamos un protocolo de actuación para realizar un test de penetración (herramientas a utilizar y en qué orden, según qué obtengamos de cada una qué opciones llevar a cabo, etc..)

Después, desplegaríamos uno o varios entornos 'limpios' y realizaríamos sobre ellos un test de penetración simulando que fuera uno real y extrayendo del mismo conclusiones con las que generar un reporte (haciendo uso de las alertas generadas y de los dashboard de Kibana) que pudiera presentarse a la empresa detrás del entorno si este fuera un test real.

En esta parte se deben emplear muchas horas puesto que durante el estudio puede surgir la necesidad de añadir más herramientas a nuestra imagen o más reglas y decoders a Wazuh, pueden descubrirse defectos en las reglas oficiales que podrían arreglarse, etc..

Desarrollo

En total, se ha estimado un reparto de 240 horas. El resto, 60 horas, serían destinados al desarrollo de este informe, planificación, reuniones y correcciones, así como a posibles contingencias que puedan hacer que alguno de los puntos anteriores se alargue más de lo esperado.

Como el plan es llevar a cabo el proyecto entre los meses de Julio de 2020 y Junio de 2021, y dado que actualmente me encuentro trabajando a jornada completa, he decidido repartir las 300 horas entre 11 meses, disponiendo así de unas 27 horas al mes, que serían más o menos 7 horas a la semana, pudiendo dedicar una hora diaria o mover si es necesario algunas horas en días laborales al fin de semana o al revés.

1.7. Security vs Compilance

Cuando hablamos en términos de ciberseguridad necesitamos destacar el término Compilance, que podría traducirse como 'el cumplimiento' de las normativas o leyes referentes a la seguridad de los datos que una empresa pueda almacenar o gestionar. Cualquier compañía con acceso a datos sensibles de sus clientes está obligada a asegurar unos mínimos requisitos de seguridad en sus entornos y ser capaz de demostrar que los cumple. [Dob]

Compilance significa la capacidad que tiene una empresa de detallar su estado de seguridad de acuerdo a una serie de requisitos regulados en forma de legislación o regulaciones de la industria o simplemente en base a unos estándar creados a partir de buenas prácticas aceptadas por la comunidad.

Algunos ejemplos importantes de *'compilances'*

HIPAA (Health Insurance Portability and Accountability Act) Se aplica a compañías del ámbito sanitario y regulariza cómo dichas compañías deben manejar y asegurar la seguridad de los datos médicos personales de sus pacientes.

PCI DSS (Payment Card Industry Data Security Standard) Creado por algunas compañías de la industria de pago con tarjetas de crédito que quiso normalizar como se asegurar la integridad de la información financiera de sus clientes. Wazuh [tea] cuenta con herramientas para asegurar este *compilance* asegurando (entre otros) la detección de *rootkits*

SOC Reports Se trata de unos reportes de control de sistemas y organizaciones verificable, llevado a cabo por una Certified Public Accountant (CPA) para asegurar el cumplimiento de las normativas de seguridad cuando la compañía maneja datos sensibles de sus clientes.

1.8. Fases de una auditoría de ciberseguridad

1.8.1. Recopilación de información

La primera fase de toda auditoría de ciberseguridad o test de penetración sería la recopilación de toda la información posible sobre los objetivos. Necesitamos llegar a conocer puntos de acceso al sistema (IPs de los distintos servidores, servicios utilizados por estos y abiertos a peticiones externas, websites, etc...), así como posible vulnerabilidades relacionadas con los usuarios del mismo (correos electrónicos que podrían ser hackeables, contraseñas inseguras o nombres de usuario que aparezcan en bases de datos de la deep web, etc...)

En un caso real, empezaríamos por investigar a la empresa a la que vamos a monitorizar y recopilar toda la información posible sobre sus sistemas

y usuarios. Como el estudio se realizará sobre máquinas virtuales y no existe una empresa real a la que investigar, en nuestro caso la primera fase consistirá principalmente en recopilar información a partir de las IPs, conocidas (suponemos realizada la fase de descubrimiento de hosts) de las mismas. Es decir, empezaremos por un escaneo de puertos y descubriremos aquellos servicios reconocibles desde el exterior, también intentaremos obtener información de la versión del sistema operativo del host y de los distintos servicios que ofrece, e intentaremos sobrepasar los firewalls.

1.9. Aspectos legales del pentesting

En primer lugar discutiremos los aspectos legales del trabajo de un hacker ético o pen-tester. Es importante tenerlos en cuenta para evitar problemas legales en un futuro y también una de las razones más importantes por las que se requiere un **entorno de pruebas seguro**.

1.9.1. Introducción y aspectos generales

1.9.2. Leyes en Europa

1.9.3. Leyes en España

1.9.4. HoneyPots y ADS

Capítulo 2

Fase 1: Entornos de pruebas y aspectos legales del pentesting

2.1. Introducción

Existen multitud en entornos de pruebas disponibles en internet para practicar los aspectos relacionados con el hacking ético. Una de las principales razones para esto es la posibilidad de meterse en problemas legales si se trata de practicar con entornos reales y equipos que no nos pertenecen. Estos aspectos se discutiran en un apartado posterior.

2.1.1. Entornos de pruebas disponibles en internet

1. "TheCyberMentor"https://github.com/hmaverickadams/Beginner-Network-Pentesting
2. "Hackthissite"https://www.hackthissite.org/
3. "Metasploitable"https://sourceforge.net/projects/metasploitable/
4. "Sliim pentest-env"https://github.com/Sliim/pentest-env
5. "Sliim pentest-lab"https://github.com/Sliim/pentest-lab

Comparativa de varios entornos

2.1.2. Cursos interesantes sobre pentesting

1. Metasploit unleashed Metasploit Unleashed https://www.offensive-security.com/metasploit-unleashed/
2. MITRE ATT&CK https://attack.mitre.org/

3. TheCyberMentor beginner curse https://github.com/hmaverickadams/Beginner-Network-Pentesting

2.2. Wazuh como una herramienta de loggin, monitorización y reporte durante nuestro análisis

Uno de los aspectos claves de Wazuh que nos han llevado a plantearnos su viabilidad como una herramienta de apoyo al test de penetración es su capacidad para analizar logs. Si se usa adecuadamente, Wazuh puede filtrar por nosotros logs de interés y generar alertas en tiempo real para eventos importantes que tengan lugar durante el análisis.

Por tanto, si decidimos monitorizar toda acción o comando ejecutado en un shell de nuestro entorno de pruebas (Kali Linux) con Wazuh, podremos registrar cada acción llevada a cabo durante los tests y reportar aquellas que sean de interés para quien lo necesite. Bien como una forma de compartir información entre los miembros de un equipo de pentesting (de forma similar a lo que haría el software **Fargate** o bien como una forma de reportar a nuestros clientes **qué hemos hecho durante el análisis** y **cómo lo hemos hecho**. Así, si hubiera algún problema en los sistemas testeados durante nuestra auditoría y alguien tratara de hacernos responsables de hecho tendríamos una baza muy importante en juego: un sistema que registra nuestras acciones y las reporta en tiempo real a algún responsable dentro de la empresa que nos contrata cualquier cosa que sea de su interés.

2.3. Entorno de trabajo: Kali Linux y Wazuh

Aunque una parte importante del entorno de pruebas son aquellas máquinas que vamos a tratar de atacar, la principal herramienta que usaremos en nuestro estudio será una imagen de Kali Linux a la que instalaremos un agente de Wazuh capaz de monitorizar el output de todos los comandos que nos interesen.

Kali Linux es una Network sniffing, lo que significa que las actualizaciones se hacen de forma incremental y no tiene un versionado discreto. Para este estudio he utilizado una ISO etiquetada como **2020.03** y la he instalado en una máquina virtual limpia. A continuación, y siguiendo la guía oficial de Wazuh para la instalación en sistemas operativos basados en Debian, he instalado un agente de Wazuh en mi máquina virtual. He instalado la versión más reciente en este momento (3.13.1) aunque la intención es ir actualizándola conforme salgan nuevas versiones a lo largo del desarrollo del trabajo.

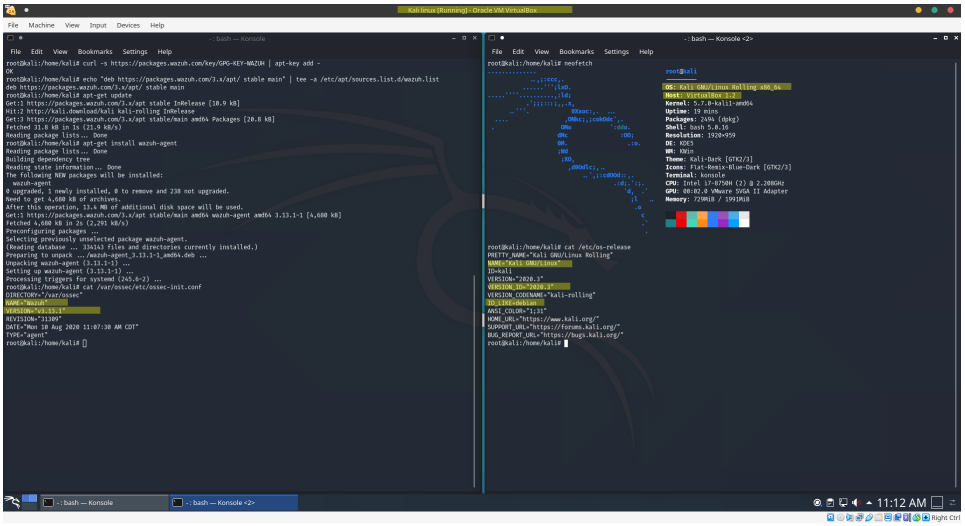

Figura 2.1: Captura de pantalla del entorno virtualizado de Kali Linux mostrando alguna información del sistema y la instalación de Wazuh en varios terminales.

2.4.

Análisis de los resultados de las ejecuciones de los distintos comandos

2.4.1. Primera aproximación utilizando el comando script

Para que Wazuh pueda generar alertas a partir del output de un comando ejecutado manualmente en un terminal, dicho output debe ser redirigido a un fichero que el agente pueda enviar al Manager para analizarlo. Para conseguir esto de forma sencilla he intentado utilizar la utilidad "script", un comando de Linux que nos permite registrar cada comando y su output en un fichero. Haciendo algunas modificaciones simples para que al iniciar una terminal interactiva el comando script se ejecute por defecto registrando los logs en un directorio concreto que será monitorizado por Wazuh y modificaremos la configuración por defecto del agente instalado en la instancia de Kali Linux para que solamente monitorice ese directorio.

El primer problema encontrado al utilizar *script* para monitorizar todas las acciones llevadas a cabo durante una sesión es que este comando crea un entorno de trabajo (una shell) nuevo en cada ejecución del mismo, leyendo y ejecutando los archivos de configuración correspondientes al inicio de la misma. Esto se traduce en un bucle infinito si tratamos de comenzar una sesión de *script* al inicio de una sesión bash, por ejemplo. Para solucionarlo,

Listing 2.1: Session recorder (/etc/profile)

```
if [ "x$SESSION_RECORD" = "x" ]
the
timestamp=$(date +%d-%m-%Y-%T)n
session_log=/var/log/session/session.$USER.$$.$timestamp
SESSION_RECORD=started
export SESSION_RECORD
script -t -f -q 2>${session_log}.timing $session_log
exit
fi
```

No obstante, esto no es suficiente. Script no nos permtie confiurar el formato del output que será enviado a Wazuh, y aunque podríamos usar herramientas extra para hacer este formateo, Script además escribe en el archivo de logs caracteres especiales de los terminales que sirven para codificar, entre otros, el color del texto, símbolos especiales, etc.. Por si fuera poco, Script guarda en sus logs el contenido tal y como aparecería en pantalla, lo que incluye lineas en blanco cuando se ejecuta un comando çlear", por ejemplo, y saltos de linea en el output de cada comando cuando este es muy largo.

Todo esto nos hace desechar esta opción y buscar una alternativa.

2.4.2. Segunda aproximación usando "trampas"de Linux (comando trap)

Una posible alternativa al comando script es utilizar las "trampas"(traps) de Linux, utilizando el comando **trap** podemos definir otro comando que se ejecutará antes de cualquier comando introducido en una shell. Así, podemos registrar el output de cada comando (así como el comando introducido en si) y calcular información como el nombre del usuario que lo ejecutó (**whoami**) o el timestamp del momento en que se ejecuta dicho comando.

SOURCE: https://unix.stackexchange.com/questions/250713/modify-all-bash-commands-through-a-program-before-executing-them

Listing 2.2: Session recorder (on bashrc file

```
shopt -s extdebug

preexec_invoke_exec () {
    [ -n "$COMP_LINE" ] && return  # do nothing if completing
    [ "$BASH_COMMAND" = "$PROMPT_COMMAND" ] && return # don't cause a
        preexec for $PROMPT_COMMAND
    local this_command=`HISTTIMEFORMAT= history 1 | sed -e "s/^[
        ]*[0-9]*[ ]*//"`;

    # So that you don't get locked accidentally
    if [ "shopt -u extdebug" == "$this_command" ]; then
        return 0
    fi

    if [[ "${this_command}" =~ \S*=.* ]]; then
      this_command_output=""
      echo "$(date '+%Y-%m-%d %H:%M:%S') $(whoami)@$(pwd)# ${
          this_command}: $this_command_output"
```

```
16        return 0
17      fi
18
19      this_command_output=$(eval "${this_command}" | tee /dev/tty)
20      this_command_output=$(echo "${this_command_output}" | tr '\n' ' ')
21
22      echo "$(date '+%Y-%m-%d %H:%M:%S') $(whoami)@$(pwd)# ${
            this_command}: $this_command_output"
23      # Modify $this_command and then execute it
24      return 1 # This prevent executing of original command
25  }
26  trap 'preexec_invoke_exec' DEBUG
```

problemas encontrados con el código original de stack overflow: 1) programas como vim neceitan que el output se redirija por pantalla antes de que finalice el comando -¿solución, usar **tee** — 2) si defines variables dentro de la función, esas variables no se exportarán a la shell que llama a la función. -¿solución, si el comando tiene el formato "palabra=loquesea"se permite su ejecución y se logea solo el input. 3) clear command shouldnt be logged because it will print white spaces 4) cd command should be regularly executed as well as other commands like

2.4.3. Wazuh Manager, Elasticsearch y Kibana app

Para que el agente de Wazuh instalado en nuestra máquina con Kali Linux nos sea útil, este debe ser registrado en un Wazuh Manager, preferiblemente uno que esté conectado a un nodo de Elasticsearch y Kibana. Para simplificar todo el despliegue de dicha infraestructura (así como la actualización de la misma si fuera necesario) voy a utilizar el repositorio oficial de Docker de Wazuh. Utilizando la herramienta **Docker-Compose** desplegaré fácilmente un entorno con

— INCISO — ¿Merece la pena un agente en kali conectado a un manager en vetetuasaber dónde? Por qué no usar un manager directamente en Kali? Y si quiero un entorno de pruebas distribuido para un equipo y quiero que todos compartan alertas? Puedo?

2.5. Pasos para la creación de la imagen custom

-install kali linux base -install kde plasma desktop environment -install terminator -install docker and docker-compose -set up logging system on all users -install wazuh agent, remove all unused modules and connect it to the manager -add crontab job to delete log files older than a week (for example) -export image!

2.6. Primera iteración: Metasploitable 3

En la primera iteración de este estudio utilizaremos una imagen llamada **Metasplotable 3** [rap] para crear una máquina virtual vulnerable de forma rápida y sin complicaciones.

Se trata de una tercera versión de un proyecto creado con la intención de ser utilizado para entrenar en el ámbito del test de penetración y sabemos de antemano que el software instalado en sus imágenes las hace fácilmente vulnerables.

Metasploitable3 está diseñado utilizado Vagrant, una herramienta que nos permitirá desplegar máquinas virtuales (con el software de virtualización que deseemos) fácilmente. Utilizaremos el archivo de configuración o Vagrantfile proporcionado en el repositorio de GitHub de **rapid7**, autor de esta herramienta y **Virtualbox** como software de virtualización.

Metasploitable3 contiene dos imágenes, una **Ubuntu 14.04** y otra **Windows server 2018**, las probaremos las dos en este orden y trataremos de ganar acceso a las mismas simulando un ataque o una auditoría de seguridad.

Capítulo 3

Fase 2: Análisis de las herramientas para el test de penetración

3.1. Análisis de input/output utilizando Wazuh

En primer lugar, he creado un decodificador de logs sencillo para simplemente extraer el input introducido por el usuario y su output. Para comandos más interesantes (como nmap), utilizaremos un decodificador específico que sea capaz de separar la distinta información relevante.

Listing 3.1: Custom base decoder for command execution logger

```
<decoder name="shell_log">
<program_name>shell_log</program_name>
</decoder>

<decoder name="shell_command">
<parent>shell_log</parent>
<regex># (\.*) -> (\.*)</regex>
<order>command, output</order>
</decoder>
```

Este decodificador, combinado con una regla que genere alertas de bajo nivel (3, que es el mínimo para que aparezcan en la interfaz de Wazuh) nos permite empezar a guardar información de todo lo que ocurra en el sistema.

Listing 3.2: Custom base rule for command execution logger

```
<group name="shell_execution">
  <rule id="66600" level="3">
  <decoded_as>shell_log</decoded_as>
  <description>$(command) executed. Output: $(output)</description>
</rule>
and, output</order>
</decoder>
```

No obstante, hay que tener cuidado con no llenar nuestros discos con información irrelevante o redundante. Por eso, cuando el proyecto esté más avanzado, modificaremos el nivel de las alertas basandose en el comando ejecutado, deforma que no guardemos la información de comandos como ls, top, pwd, cat, etc..

En las siguientes secciones describiremos algunas herramientas de interés en tests de penetración de sistemas así como las reglas y/o decodificadores desarrollados para las mismas.

3.2. Distribuciones de Linux para pentesting

3.3. Clasificación de herramientas

3.3.1. Recopilación de información: descubrimiento de hosts y escaneo de puertos

A la hora de detectar los hosts visibles dentro de una red y qué servicios ofrece (y son visibles) podemos encontrar diversas herramientas:

ping Commando simple que permite enviar ICMP .echo requests.a hosts en la red para ver si están en funcionamiento.

nmap Herramienta de escaneo de redes que utiliza paquetes IP (raw) para determinar los hosts disponibles en una red y los servicios que ofrecen.

metasploit framework :)

3.4. Análisis de las distintas herramientas

3.4.1. Nmap

Nmap es una herramienta muy versátil y que puede ofrecer muchísima información sobre los diferentes endpoints en una red (aunque puede analizar hosts individualmente, parte de su interés radica en que puede ser utilizada para analizar redes enteras). Permite detectar aquellos servidores con determinados puertos abiertos o aquellos puertos que tiene abiertos cada servidor, las versiones del sistema operativo del endpoint o de los servicios que ofrece e incluso muchas vulnerabilidades de los mismos. Además, tiene numerosas opciones que permiten modificar el tipo de comunicación que se hará durante el escaneo e incluso la velocidad del mismo (un escaneo muy rápido podría suponer problemas con el firewall) o tratar de anonimizar los paquetes enviados durante el escaneo para conseguir ocultar nuestras acciones (Obfuscation)

Algunos ejemplos del uso de nmap que pueden ser de interés para el estudio y que podemos utilizar para empezar a desarrollar las reglas de Wazuh que generarán alertas cuando nmap detecte información relevante.

flags	example	usage
	nmap 172.1.1.0/24	Default scan a network using CIDR notation
-sS	nmap 172.1.1.11 -sS	TCP SYN port scan (Default)
-sT	nmap 192.1.10.10 -sT	TCP connect port scan
-sA	nmap 122.122.1.1 -sA	TCP ACK port scan
-p	nmap 122.128.1.1 -p 21-100	scan ports in a range
-O	nmap -O 172.10.1.22	OS detection
-Tn	nmap 192.168.1.10 -T2	n=0,1,..,5 regulates the scan speed

Cuadro 3.1: Some examples of nmap usage

Capítulo 4

Fase 3: Estudio práctico del test de penetración y análisis del mismo

4.1. Recopilación de información

En primer lugar, tratamos de recopilar toda la información posible del host de interés. Usando la utilidad "pingçomprobamos que el servidor está operativo y procedemos al escaneo de puertos, para el cual emplearemos **nmap**.

Metasploitable3, como cualquier servidor real, dispone de varios servicios corriendo en el momento de levantarse. Sin embargo, se encuentra conectada a la red a local a través de una interfaz de Ethernet, y además dispone de un firewall instalado por defecto, lo cual dificulta un poco el análisis.

Suponiendo que no conocemos ninguna IP del servidor metasploitable3, deberíamos comenzar utilizando nmap para un escaneo simple con el objetivo de determinar en qué IP se encuentra escuchando el servidor.

Utilizando la opción **-sn** (no port scan), indicamos a nmap que queremos hacer un escaneo de la red solo para ver qué hosts se encuentran en ella, así localizaremos nuestro host objetivo. Dado que nuestro propio host (Kali linux) estará también en esta red, añadiremos una opción para excluirlo del escaneo.

A continuación, una vez detectado el nodo que nos interesa, haremos un escaneo más concreto utilizando una opción diferente de nmap, por ejemplo, el flag '-sS', que realiza un escaneo utilizando TCP sobre un mayor rango de puertos y completando el handshake del protocolo, de forma que puede confirmar qué puertos están abiertos y cuales están cerrados.

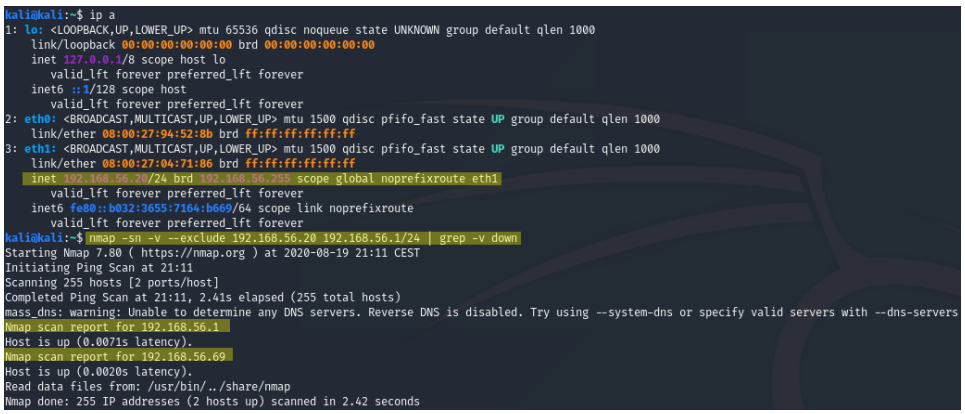

Figura 4.1: Visualización del escaneo de puertos inicial usando nmap. En él podemos ver que en la red local hay dos nodos además de aquél ejecutando Kali Linux, uno de ellos será el router de la red local (192.168.56.1) y el otro, 192.168.56.69, nuestro objetivo, corriendo metasploit3.

Bibliografía

[Dob] Bojana Dobran. *Security vs Compilance*. URL: `https://phoenixnap.com/blog/security-vs-compliance`. (accessed: 14.06.2020).

[rap] rapid7. *metasplotable3*. URL: `https://github.com/rapid7/metasploitable3`. (accessed: 10.08.2020).

[tea] Wazuh team. *Using Wazuh for PCI DSS*. URL: `https://documentation.wazuh.com/3.12/pci-dss/index.html`. (accessed: 14.06.2020).