

Endurecimiento de redes y máquinas en IsardVDI

Introducción

El objetivo de esta práctica es, utilizando la plataforma de virtualización IsardVDI, simular una pequeña LAN (red local) de una empresa, compuesta por 3 servidores (base de datos, web y proxy) y tres clientes, que serían los trabajadores de la empresa. En el siguiente punto explicaremos con detalle la estructura de nuestra red.

Una vez creada la LAN, la securizaremos añadiendo un IDS que monitorizará todas las peticiones que lleguen al nodo de entrada y instalaremos el servicio Goaccess, el cual nos proporcionará un dashboard para visualizar estadísticas detalladas y gráficas interactivas sobre el tráfico y la actividad del servidor proxy.

Acto seguido, aplicaremos el concepto de seguridad ofensiva y pondremos a prueba la seguridad de nuestra red, buscando sus vulnerabilidades y probando varios tipos de ataques contra ella.

Por último, viendo las métricas que nos proporcionará Goaccess, analizaremos el impacto que han tenido los ataques en ella.

Informe

1. Crear una red virtual y configurar las máquinas

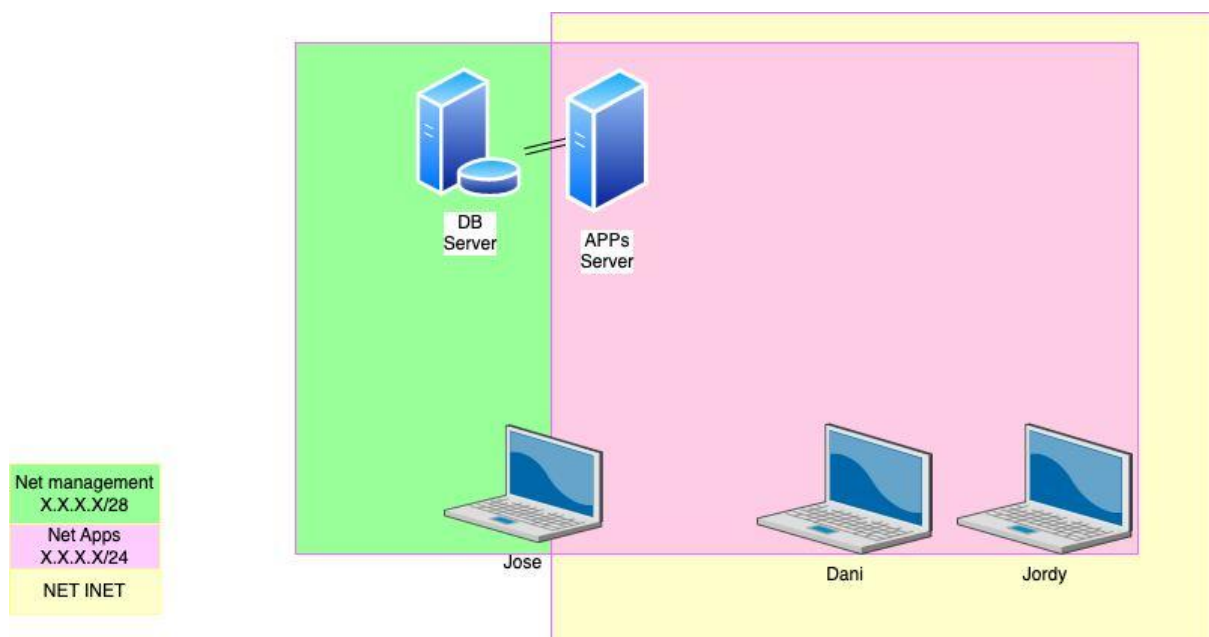
En primer lugar diseñamos el esquema de nuestro sistema máquinas/redes que hemos decidido. De manera resumida la estructura será la siguiente:

Máquinas:

- 1 servidor de base de datos (Alpine)
- 1 servidor web (Alpine)
- 1 servidor proxy inverso (Alpine)
- 3 clientes (Ubuntu y Windows)

Redes:

- una de gestión (/24 cantidad de hosts)
- una de servicio (/28)
- una default para la conexión a Internet



Despliegue de máquinas

En IsardVDI desplegamos las siguientes máquinas

- 1 servidor de base de datos (Alpine)
- 1 servidor aplicaciones (Alpine)
- 1 servidor proxy inverso (Alpine)
- 3 clientes (Ubuntu y Windows)

Las máquinas que utilizaremos como servidores utilizarán la distribución Alpine, mientras que las máquinas que usaremos como clientes serán Ubuntu o Windows.

Configuración de máquinas servidor en IsardVDI

Una vez desplegadas las máquinas y completada las configuraciones iniciales de redes procedemos a deshabilitar servicios (dhcpd) e instalar servicios/programas que necesitaremos (nano, docker-compose, docker, etc...).

Máquina para Maria DB

PEQUEÑÍSIMA INTRO DE PARA QUÉ QUEREMOS UNA MÁQUINA CON MARIA DB

Tal y como explicamos en el primer punto del informe, uno de los servidores Alpine alojará una base de datos, que será la que se utilizará nuestra web para almacenar la información que llegue vía formulario web.

Para empezar vamos a realizar una actualización del sistema para mantener los paquetes actualizados. Seguidamente instalaremos MariaDB, y crearemos la base de datos que utilizará nuestra web. Esta base de datos tendrá una única tabla con dos campos, usuario y contraseña. Crearemos también un usuario, que será con el que se realizarán todas las conexiones a la base de datos.

Configuración de redes y máquinas cliente en IsardVDI

Nuestra plataforma de virtualización (IsardVDI), tiene 6 VLANs (redes LAN virtuales) preconfiguradas, una default con salida a internet y 5 redes internas (puigcastellar1-5). Cuando realizamos el despliegue de la máquina, podemos elegir cuántas interfaces de red queremos que tenga, y en qué red tienen que estar cada una de ellas.

En nuestro caso, configuraremos nuestros clientes para que utilicen dos tarjetas de red:

- eth0 (Default en IsardVDI) -> Será la que se utilizara para salir a la WAN. (solo la utilizaran los servidores)
- eth1 (puigcastellar5 en IsardVDI) -> Será la LAN interna que permitirá la conexión con los servidores.

Modificamos también los resolvers para que nos resuelva los dominios el DNS que tenemos en la LAN, sino no podremos acceder a los paneles de administración del DNS/Proxy.

Tras configurar todas las máquinas, las IP quedarían así:

Administrador	Dani	Jose	Jordy
Servidor	Página Web	Proxy	Base de datos
SO	Alpine	Alpine	Alpine
eth0 (Default)	192.168.120.179	192.168.120.178	192.168.120.180
eth1 (puigcastellar5)	10.0.0.2	10.0.0.3	10.0.0.1
Cliente	Windows	Kali	Kali
eth1 (puigcastellar5)	10.0.0.202	10.0.0.203	10.0.0.201

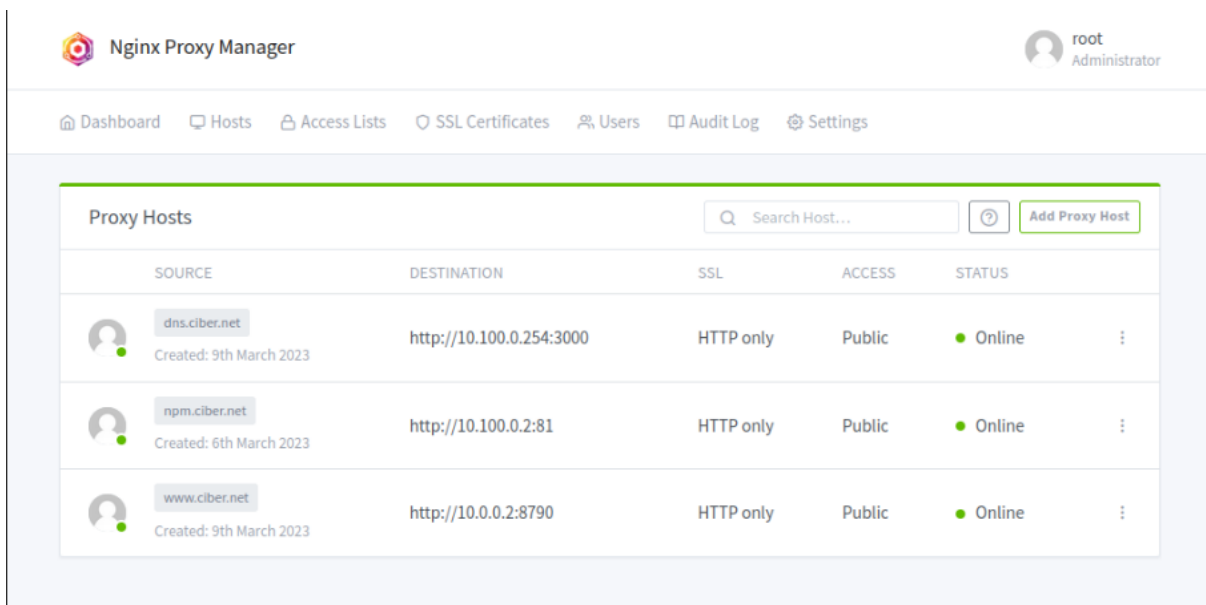
Configuraciones de los contenedores

Para realizar esta práctica, hemos decidido instalar los servicios que ofrecen nuestros servidores en contenedores, que se deciden lanzar utilizando un archivo docker compose. Se crean contenedores para los siguientes servicios:




- Proxy (Nginx)
- DNS (Adguard)
- Base de datos (MariaDB)
- Página Web (Grav)

Configuraremos el DNS para que reenvie al Proxy todas las peticiones que tengan como destino el dominio ciber.net, en el proxy crearemos 3 subdominios:

- www.ciber.net (Servidor Web)
- npm.ciber.net (Proxy)
- dns.ciber.net (DNS)



The screenshot shows the Nginx Proxy Manager web interface. At the top, the logo and name 'Nginx Proxy Manager' are on the left, and the user 'root Administrator' is on the right. Below the header is a navigation menu with items: Dashboard, Hosts, Access Lists, SSL Certificates, Users, Audit Log, and Settings. The main content area is titled 'Proxy Hosts' and contains a table with the following data:

SOURCE	DESTINATION	SSL	ACCESS	STATUS
 dns.ciber.net Created: 9th March 2023	http://10.100.0.254:3000	HTTP only	Public	● Online
 npm.ciber.net Created: 6th March 2023	http://10.100.0.2:81	HTTP only	Public	● Online
 www.ciber.net Created: 9th March 2023	http://10.0.0.2:8790	HTTP only	Public	● Online

2. Configurar un sistema de detección de intrusos

En nuestro caso, tras analizar los posibles IDS que podríamos instalar, decidimos utilizar Suricata, al ser uno de los más conocidos y fáciles de configurar.

Lo primero que tenemos que hacer es actualizar el repositorio de reglas predeterminadas que ofrece este programa. En este caso, utilizaremos las siguientes:

oisf/trafficid: La regla "oisf/trafficid" en Suricata es una regla de detección de tráfico genérico que se utiliza para identificar y alertar sobre tráfico de red que no coincide con ninguna otra regla específica.

et/open: Es una de las reglas disponibles en Suricata. Es una regla de detección de intrusiones que se enfoca en la detección de amenazas de código abierto y público conocido.

ptresearch/attackdetection: Se utiliza para detectar posibles ataques de red, incluyendo intentos de exploit y tráfico malicioso. Esta regla es parte del conjunto de reglas de detección de amenazas desarrollado por Positive Technologies Research (PT Research).

sslb/ssl-fp-blacklist: Es una regla de detección de tráfico SSL malicioso basado en huellas dactilares SSL, también conocidas como "huellas dactilares de certificados". Estas huellas dactilares se generan a partir de información específica del certificado SSL, como el emisor, el nombre común del sujeto, la fecha de caducidad y otros detalles.

Una vez habilitadas estas reglas, actualizamos el servicio y lo probamos conectándonos a la siguiente URL: <http://testmynids.org/uid/index.html>

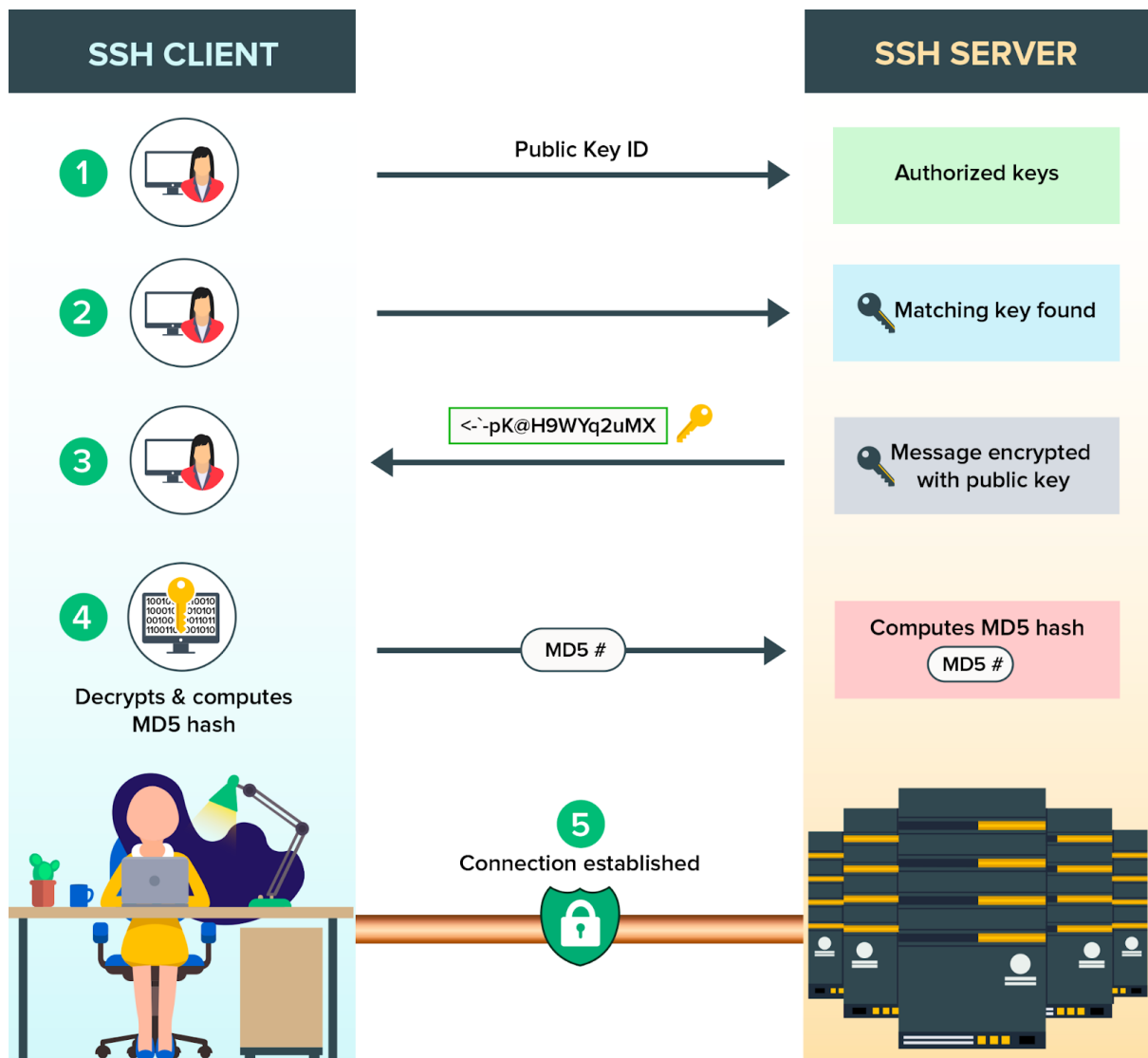
Efectivamente, vemos que Suricata nos ha bloqueado esta conexión, al detectar que el servidor nos está enviando tráfico malicioso.

3. Implementar cifrado

La conexión entre servidores, que están ubicados en la misma red, debe hacerse de la forma más segura posible. Para esto debemos de implementar medidas que nos permitan asegurar que estas conexiones se hacen sin implicar contraseñas, ya que en caso de robo de información el atacante suele ir en la búsqueda de contraseña o patrones para poder hacerse con ellas.

Claves pública-privada

En este sentido las claves públicas y privadas, nos permiten que tanto servidor de origen y destino, tengan “huellas” para poder validarse unas a otras. esto se hace la siguiente manera:



Generamos un par de claves pública-privada en los servidores utilizando la herramienta OpenSSL, una vez generadas las claves solo nos faltaria compartir nuestra clave pública con el resto de servidores, asi podran añadirla al archivo `known_hosts` y por tanto, conectarse a nuestro servidor por ssh sin tener que introducir siempre el password. Utilizamos el comando `scp` para enviar el archivo a las máquinas remotas.

4. Jornada de ataques

Tal y como comentamos en la introducción, para poner a prueba la seguridad de nuestra red, utilizaremos el concepto de “offensive security”, es decir lanzar diversos tipos de ataques contra ella. En nuestro caso, ninguno de ellos ha conseguido vulnerar gravemente la seguridad de la red.

Los ataques que se han lanzado son los siguientes:

Escaneo de vulnerabilidades del servidor web

Objetivo: Escanear el servidor web, para encontrar posibles vulnerabilidades a explotar en un futuro ataque

Tipo de ataque: Reconocimiento

Herramienta: OWASP ZAP

Resultados: Localizamos un par de vulnerabilidades importantes en la web, concretamente vemos que el formulario puede ser atacado mediante SQL injection, ya que no está protegido.

En nuestro caso, utilizaremos el programa OWASP ZAP para que nos analice el dominio donde se aloja el servidor web (www.ciber.net). Encontramos vulnerabilidades graves en el archivo insert.php, que es el que se encarga de introducir en nuestra base de datos el texto que se introduce en el formulario de la web. Parece que uno de los ataques que podemos intentar es un SQL injection, ya que nuestro formulario no está protegido contra este tipo de ataques.

Ataque contra el form de la página web www.ciber.net

Objetivo: Formulario sencillo en archivo index.php.

Tipo de ataque: Ataque SQL Injection

Herramienta: SQLmap, Owasp

Resultados: 146 envíos de formulario con su correspondiente inclusión en la DB

Dado que en el anterior ataque de reconocimiento se ha detectado que el form que utiliza la web es inseguro, lanzamos este ataque para intentar insertar código SQL en la base de datos.

Utilizamos herramientas como SQLmap para intentar realizar un ataque de tipo SQL injection al form que tenemos en nuestra web. Esta herramienta lo que hace es lanzar parejas de usuarios y contraseñas utilizando el formulario, intentando insertar código SQL en los dos campos de la base de datos.

Ataque de inyección de código Javascript desde el form

Objetivo: Intentar inyectar código malicioso en el form para poder abrir un reverse shell con el que podamos ejecutar comandos en el servidor.

Tipo de ataque: Inyección de código

Herramienta: Ninguna (inyectamos el código desde el navegador web).

Resultados: Conseguimos inyectar código en el servidor, pero no abrir un reverse shell

Probamos de insertar código javascript en el campo "usuario" de nuestro formulario. Como no está protegido, lo ejecutará.

Al enviar la información, vemos que efectivamente se ejecuta el código javascript. En este caso, nuestro código solo nos abre un pop-up, pero podemos insertar código malicioso para que por ejemplo nos abra un reverse shell con el que podamos ejecutar comandos en el servidor, esto finalmente no lo logramos.

Ataque utilizando un exploit contra el proxy

Objetivo: Vulnerar la seguridad del servidor proxy, para mediante un exploit poder abrir un reverse shell con el que poder ejecutar comandos en el servidor.

Tipo de ataque: Exploit

Herramienta: Metaexploit

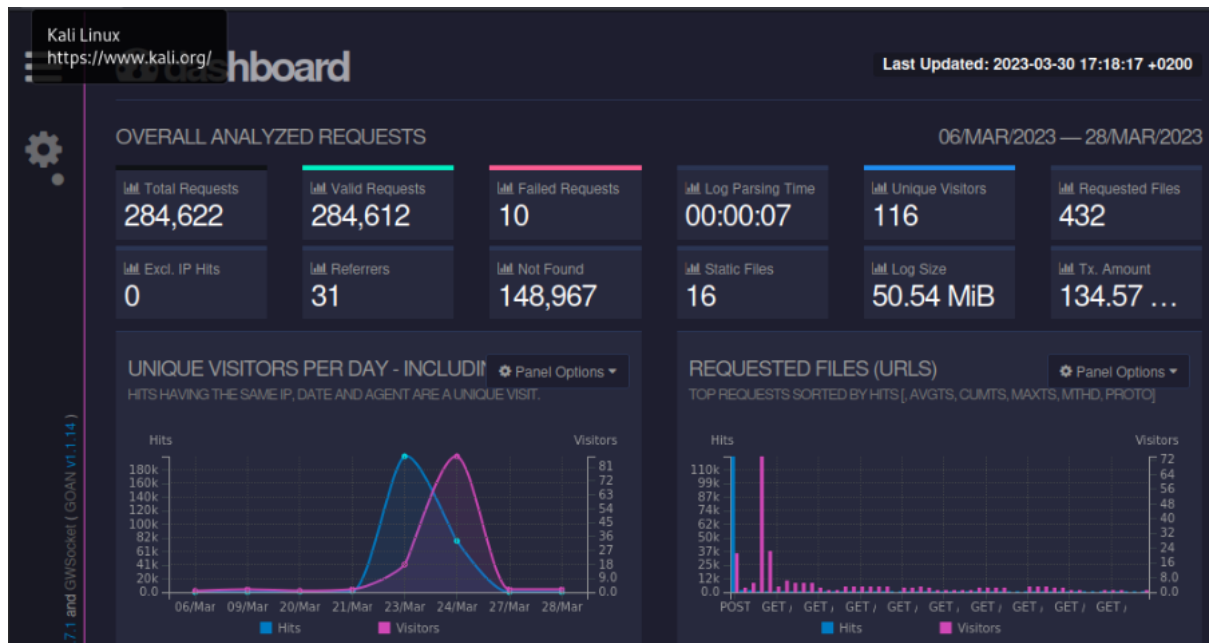
Resultados: No conseguimos abrir un reverse shell en el servidor, nuestro exploit no consigue explotar ninguna vulnerabilidad del servidor.

Utilizando la herramienta metaexploit, probamos varios exploits que nos ofrece esta herramienta para atacar este tipo de servidores, con el objetivo de poder abrir un reverse shell en el servidor, y así poder ejecutar comandos en el mismo.

5. Monitoreo y análisis de las métricas

Se utilizará la herramienta goaccess para monitorizar todas las peticiones que lleguen al servidor proxy, para ello instalaremos el servicio en un contenedor docker, utilizando un archivo docker-compose para realizar el despliegue.

Este servicio nos muestra el siguiente dashboard:



Analysis

Vemos que durante toda la jornada de ataques se han realizado un total de 284,622 peticiones. El número es tan elevado porque herramientas como SQLmap realizan muchísimas peticiones al servidor en muy poco tiempo.

En la gráfica de la izquierda vemos el número de peticiones por día, observamos claramente que todas las peticiones se han realizado entre el día 21 y el 27 de marzo. Por otro lado, en el cuadro de la derecha, vemos que casi todas las peticiones han sido de tipo POST (envío de datos). En nuestro caso, estas peticiones son las que han enviado los datos por el formulario.

Conclusión Final

Al acabar esta práctica, hemos visto que el punto más débil de este tipo de infraestructuras de red sencillas es el formulario del servidor web, que tiene que estar bien securizado para no permitir accesos ilícitos a la base de datos. Por otro lado, descubrimos muchas de las funcionalidades que tiene la plataforma IsardVDI y lo útil que es para realizar este tipo de prácticas, ya que a parte de máquinas, utilizándola podemos desplegar también diferentes redes fácilmente.

Realizando este trabajo, hemos aprendido que docker es una herramienta muy poderosa, que nos ha permitido desplegar fácilmente todas las máquinas con sus respectivos servicios, sin tener que instalarlos en el propio servidor. También hemos aprendido que Alpine es un SO muy indicado para realizar este tipo de despliegues, ya que consume muy pocos recursos y ofrece muy buen rendimiento.